THE SINGLE BIGGEST PROBLEM IN COMMUNICATION IS THE ILLUSION THAT IT HAS TAKEN PLACE.

GEORGE BERNARD SHAW

WORDS EMPTY AS THE WIND ARE BEST LEFT UNSAID.

HOMER

EVERYTHING BECOMES A LITTLE DIFFERENT AS SOON AS IT IS SPOKEN OUT LOUD.

HERMANN HESSE

LANGUAGE IS A VIRUS FROM OUTER SPACE.

WILLIAM S. BURROUGHS
COMMUNICATION COMPLEXITY AND APPLICATIONS (EARLY DRAFT)
for our families
Contents

Introduction 11

Conventions and Preliminaries 15

I Communication 21

1 Deterministic Protocols 23

Some Protocols 25
Rigorously Defining Communication Protocols 28
Counting Arguments 30
Rectangles 31
Balancing Protocols 33
From Rectangles to Protocols 34
Lower Bounds Based on Rectangles 36
Rectangle Covers 43

2 Rank 51

Communication Complexity and Rank 54
Upper bounds on Communication From Rank 56
Non-negative Rank 61
Compressing a Single Round of Communication 147
Internal Compression of Protocols 155
Direct Sums in Randomized Communication Complexity 159
Other Methods to Compress Protocols 160

II Applications 163

8 Circuits and Proofs 165

Boolean Circuits 165
Karchmer-Wigderson Games 166
Lower Bounds on the Depth of Monotone Circuits 168
Monotone Circuit Depth Hierarchy 170
Boolean Formulas 171
Boolean Depth Conjecture 174
Proof Systems 175
Resolution Refutations 176
Cutting Planes 179

9 Memory Size 185

Lower Bounds on Streaming Algorithms 188
Lower Bounds on Branching Programs 193

10 Data Structures 197

Dictionaries 197
Maintaining a Set of Numbers 198
Lower Bounds on Static Data Structures 205
Lower bounds on Dynamic Data Structures 211
Graph Connectivity 216
<table>
<thead>
<tr>
<th>11</th>
<th><strong>Extension Complexity of Polytopes</strong></th>
<th>223</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Basic Properties and Features of Polytopes</td>
<td>223</td>
</tr>
<tr>
<td></td>
<td>Transformations of Polytopes</td>
<td>225</td>
</tr>
<tr>
<td></td>
<td>Algorithms from Polytopes</td>
<td>229</td>
</tr>
<tr>
<td></td>
<td>Extension Complexity of a Polytope</td>
<td>232</td>
</tr>
<tr>
<td></td>
<td>Slack Matrices</td>
<td>238</td>
</tr>
<tr>
<td></td>
<td>Upper Bounds on Extension Complexity</td>
<td>240</td>
</tr>
<tr>
<td></td>
<td>Lower Bounds on Extension Complexity</td>
<td>241</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>12</th>
<th><strong>Distributed Computing</strong></th>
<th>253</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Coloring the Network</td>
<td>253</td>
</tr>
<tr>
<td></td>
<td>Computing the Diameter of the Network</td>
<td>255</td>
</tr>
<tr>
<td></td>
<td>Computing the Girth of the Network</td>
<td>256</td>
</tr>
</tbody>
</table>

**List of Figures**  259

**Bibliography**  265

**Index**  277
Introduction

Communication complexity is a mathematical theory that addresses a basic question:

If two or more people want to compute something about the information they possess, how long does their conversation need to be?

The philosophical value of this investigation is almost self-evident—the ability to efficiently communicate information is one of our key advantages as a species, and communication complexity provides a principled way to quantify this efficiency. In addition, the theory is powerful enough that it allows us to identify the limits of computational processes that are not directly related to communication. This is because every computational process seems to inherently involve communication. For example, the human brain, that most familiar of computational devices, contains much more white matter than gray matter. It is the white matter that facilitates communication between different regions of the brain. If we can understand something about the rate of communication that needs to take place between two or more parts of the brain to carry out a particular computation, we can infer limits on the capability of the brain to effectively handle the computation.

In the years following the basic definitions by Yao¹, communication complexity has established itself as an essential tool for identifying the limitations of computation. The theory is general enough that it captures something important about all computational processes, yet simple and natural enough that beautiful ideas from a wide range of mathematical disciplines can be used to understand it. For example, the theory can be used to prove fundamental facts about boolean circuits, proofs, data structures, linear programs, distributed systems and streaming algorithms. This diversity of applications strongly suggests that communication complexity is the right abstraction to understand general models of computation. In this book, we guide the reader through communication complexity along a path that includes many exquisite highlights from mathematics. We shall see ideas from

¹ Yao, 1979
Two sets are disjoint if they have no common elements.

Like this.

Acknowledgements

Thanks to Anil Ananthaswamy Morgan Dixon, Abe Friesen, Mika Göös, Jeff Heer, Pavel Hrubeš, Weston Karnes, Guy Kindler, Vincent Liew, Venkatesh Medabalimi, Or Meir, Shay Moran, Aram Odeh,
Rotem Oshman, Sebastian Pokutta, Kayur Patel, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, Thomas Rothvoß, and Makrand Sinha for many contributions to this book.
In this chapter, we set up notation and explain some standard facts that are used throughout the book.

Sets, Numbers and Functions

For a positive integer \( k \), we use \([k]\) to denote the set \( \{1, 2, \ldots, k\} \). We denote by \( 2^{[k]} \) the power set of \([k]\), namely the family of all subsets of \([k]\). Following the convention in computer science, we refer to the numbers 0 and 1 as bits. All logarithms are computed base 2 unless otherwise specified.

Random variables are denoted by capital letters (like \( A \)) and values they attain are denoted by lower-case letters (like \( a \)). Events in a probability space will be denoted by calligraphic letters (like \( \mathcal{E} \)). Given \( a = a_1, a_2, \ldots, a_n \), we write \( a \leq i \) to denote \( a_1, \ldots, a_i \). We define \( a < i \) similarly. We write \( a_S \) to denote the projection of \( a \) to the coordinates specified by the set \( S \subseteq [n] \).

A function \( f : D \to R \) is an object that maps every element \( x \) of the set \( D \) to a unique element \( f(x) \) of the set \( R \). A boolean function is a function that evaluates to a bit. Given two real valued functions \( f(n), g(n) > 0 \), meaning that \( f(n) \) and \( g(n) \) are real numbers, we write \( f(n) \leq O(g(n)) \) if there are numbers \( n_0, c > 0 \), such that if \( n > n_0 \) then \( f(n) \leq cg(n) \). We write \( g(n) \geq \Omega(f(n)) \) when \( f(n) \leq O(g(n)) \). We write \( f(n) \leq o(g(n)) \), if \( \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0 \).

Graphs

A graph on the set \([n]\), called the vertices, is a collection of sets of size 2, called edges. A clique \( C \subseteq [n] \) in the graph is a subset of the vertices such that every subset of \( C \) of size 2 is an edge of the graph. An independent set \( I \subseteq [n] \) in the graph is a set that does not contain any edges. A path in the graph is a sequence of vertices \( v_1, \ldots, v_n \) such that \( \{v_i, v_{i+1}\} \) is an edge for each \( i \). A cycle is a path whose first and last vertices are the same. A cycle is called simple if all of its edges are distinct. A graph is said to be connected if there is a path
between every two distinct vertices in the graph. A graph is called a tree if it is connected and has no simple cycles. The degree of a vertex in a graph is the number of edges it is contained in. A leaf in a tree is a vertex of degree one. Every tree has at least one leaf. It follows by induction on \( n \) that every tree of size \( n \) has exactly \( n - 1 \) edges.

**Probability**

Throughout this book, we consider only finite probability spaces, or uniform distributions on intervals of real numbers.

Let \( p \) be a probability distribution on a finite set \( \Omega \). That is, \( p \) is a function \( p : \Omega \to [0, 1] \) and \( \sum_{a \in \Omega} p(a) = 1 \). Let \( A \) be a random variable chosen according to \( p \). That is, for each \( a \in \Omega \) we have \( \Pr[A = a] = \Pr_p[A = a] = p(a) \). We sometimes use the notation \( p(a) \) to denote both the distribution of the variable \( A \), and the number \( \Pr[A = a] \). The meaning will be clear from context. For example, if \( \Omega = \{0, 1\}^2 \) and \( A \) is uniformly distributed in \( \Omega \) then \( p(a) \) sometimes denotes the uniform distribution on \( \Omega \), and if \( a = (0, 0) \) then \( p(a) \) denotes the number \( 1/4 \).

Similarly, we write \( p(a|b) \) to denote either the distribution of \( A \) conditioned on the event \( B = b \), or the number \( \Pr[A = a|B = b] \). Given a distribution \( p(a, b, c, d) \), we write \( p(a, b, c) \) to denote the marginal distribution on the variables \( a, b, c \) (or the corresponding probability). We often write \( p(ab) \) instead of \( p(a,b) \) for conciseness of notation. In the example above, if \( B = A_1 + A_2 \), and \( b = 1 \), then \( p(a|b) \) may denote the uniform distribution on \( \{(0,1), (1,0)\} \) when \( a \) is a free variable, and when \( a = (0,1) \) then \( p(a|b) = 1/2 \).

If \( \mathcal{E} \) is an event, we write \( p(\mathcal{E}) \) to denote its probability according to \( p \). We denote by \( E_{p(a)}[g(a)] \) the expected value of \( g(a) \) with respect to \( p \). We write \( A = M = B \) to assert that

\[
p(amb) = p(m) \cdot p(a|m) \cdot p(b|m).
\]

The statistical distance (also known as total variational distance) between \( p(x) \) and \( q(x) \) is defined to be:

\[
|p - q| = (1/2) \sum_x |p(x) - q(x)| = \max_T p(T) - q(T),
\]

where the maximum is taken over all subsets \( T \) of the universe.

For example, if \( p \) is uniform on \( \Omega = \{0, 1\}^2 \) and \( q \) is uniform on \( \{(0,1), (1,0), (1,1)\} \subset \Omega \) then when \( a \) is a free variable \( |p(a) - q(a)| \) denotes the statistical distance between the distributions, which is \( 1/4 \), and when \( a = (0,0) \) we have \( |p(a) - q(a)| = 1/4 \).

We sometimes write \( p(x) \approx q(x) \), to indicate that \( |p(x) - q(x)| \leq \epsilon \).

Suppose \( A, B \) are two random variables in a probability space \( p \). For

The reason we use this notation is to reduce the amount of symbols used; later on, we shall consider quite complicated scenarios, where there are several random variables with a complicated conditioning structure. In those cases, it will be helpful to use as few symbols as possible.

\( A - M - B \) is often called a Markov chain.

The proof of the second equality is an exercise.
ease of notation, we write $p(a|b) \approx p(a)$ for average $b$ to mean that
\[
\mathbb{E}_{p(b)} [\| p(a|b) - p(a) \|] \leq \epsilon.
\]

Some Useful Inequalities

Markov’s Inequality

Suppose $X$ is a non-negative random variable, and $\gamma$ is a number. Markov’s inequality bounds the probability that $X$ exceeds $\gamma$ in terms of the expected value of $X$:
\[
\mathbb{E} [X] > p(X > \gamma) \cdot \gamma \implies p(X > \gamma) < \frac{\mathbb{E} [X]}{\gamma}.
\]

Chernoff-Hoeffding Bound and Estimates on Binomial Coefficients

The Chernoff-Hoeffding bound controls the concentration of the sum of independently distributed bits around its expectation. Suppose $X_1, \ldots, X_n$ are independent identically distributed bits. Let $\mu = \mathbb{E} [\sum_{i=1}^n X_i]$. The bound says that for any $0 < \delta < 1$,
\[
\Pr \left[ \left| \sum_{i=1}^n X_i - \mu \right| > \delta \mu \right] \leq e^{-\delta^2 \mu / 3}.
\]

The following form of the bound applies when $\delta > 1$,
\[
\Pr \left[ \sum_{i=1}^n X_i > (1 + \delta)\mu \right] \leq e^{-\delta \mu / 3}.
\]

When $X_1, \ldots, X_n$ are uniformly random bits, the bounds above give estimates on binomial coefficients. For a number $0 \leq a \leq n/2$, we have:
\[
\sum_{k \in \mathbb{Z}_+ : |k - n/2| > a} \binom{n}{k} \leq 2^n \cdot e^{-4a^2 / 3n}.
\]

We also have the following upper bounds on binomial coefficients: for all $k \in [n],$
\[
\binom{n}{k} \leq \frac{2^{n+1}}{\sqrt{\pi n}}.
\]

Approximating Linear Functions by Exponentials

We will often need to approximate linear functions with exponentials: $e^{-x} \geq 1 - x$ when $x \geq 0$, and $1 - x \geq 2^{-2x}$ when $0 \leq x \leq 1/2$. 

\[
\begin{align*}
\text{\tabular}{c} e^{-x} \geq 1 - x \\
1 - x \geq 2^{-2x}
\end{align*}
\]
Cauchy-Schwartz Inequality

The Cauchy-Schwartz inequality says that for two vectors \( x, y \in \mathbb{R}^n \), their inner product is at most the products of their lengths:

\[
\left| \sum_{i=1}^{n} x_i y_i \right| = |\langle x, y \rangle| \leq \|x\| \cdot \|y\| = \sqrt{\sum_{i=1}^{n} x_i^2} \cdot \sqrt{\sum_{i=1}^{n} y_i^2}.
\]

Jensen’s Inequality and Convexity

A function \( f : \mathbb{R} \rightarrow \mathbb{R} \) is said to be convex if

\[
\frac{f(x) + f(y)}{2} \geq f \left( \frac{x + y}{2} \right),
\]

for all \( x, y \) in the domain. It is said to be concave if

\[
\frac{f(x) + f(y)}{2} \leq f \left( \frac{x + y}{2} \right).
\]

Some convex functions: \( x^2, e^x, x \log x \). Some concave functions: \( \log x, \sqrt{x} \).

Jensen’s inequality says if a function \( f \) is convex, then \( \mathbb{E} [f(X)] \geq f(\mathbb{E} [X]) \), for any real-valued random variable \( X \). Similarly, if \( f \) is concave, then \( \mathbb{E} [f(X)] \leq f(\mathbb{E} [X]) \). In this book, we often say that an inequality follows by convexity when we mean that it can be derived by applying Jensen’s inequality to some function.

A consequence of Jensen’s inequality is the Arithmetic-Mean Geometric-Mean inequality:

\[
\frac{\sum_{i=1}^{n} a_i}{n} \geq \left( \prod_{i=1}^{n} a_i \right)^{1/n},
\]

which can be proved using the concavity of the log function:

\[
\log \left( \frac{\sum_{i=1}^{n} a_i}{n} \right) \geq \frac{\sum_{i=1}^{n} \log a_i}{n} = \log \left( \prod_{i=1}^{n} a_i^{1/n} \right).
\]

Basic Facts from Algebra

A few places in this book require knowledge about polynomials and finite fields. We cannot give a comprehensive introduction to these topics here, but we do state some basic facts that are relevant to this book.

A field \( \mathbb{F} \) is a set containing 0 and 1 that is endowed with the operations of addition, multiplication, subtraction and division. A field must satisfy that \( a + b, ab, a - b \) must all be elements of \( \mathbb{F} \) for any
Let $a, b \in \mathbb{F}$, and $a/b \in \mathbb{F}$ as long as $b \neq 0$. We require that $a - a = 0$ for all $a \in \mathbb{F}$, and $a/a = 1$ for all $a \neq 0$. The simplest example of a field is the field of rational numbers.

In applications, it is often useful to consider fields that have a finite number of elements. The simplest example of a finite field is a prime field. For a prime number $p$, $\mathbb{F}_p$ is the field containing the $p$ elements $0, 1, 2, \ldots, p - 1$. These numbers can be added and multiplied modulo $p$ to get the addition and multiplication operations. One can define division as well.

**Vector Spaces, Subspaces and Duals**

Given a field $\mathbb{F}$, the set $\mathbb{F}^n$ can be viewed as a vector space over $\mathbb{F}$. We have the usual notions of vectors being linearly independent or dependent, the only difference is that linear combinations are taken using coefficients from $\mathbb{F}$. Given a subspace $V \subseteq \mathbb{F}$, we define its dual subspace

$$V^\perp = \left\{ w \in \mathbb{F}^n : \sum_{i=1}^n v_i w_i = 0, \text{ for all } v \in V \right\}.$$

The following fact is useful: If $V \subseteq \mathbb{F}^n$ is a subspace, the sum of the dimensions of $V$ and $V^\perp$ is exactly $n$.

**Polynomials**

A polynomial over the variables $X_1, X_2, \ldots, X_n$ is an expression like

$$aX_1X_2X_3^2 + bX_3X_7X_5 - cX_1X_4^2.$$

It is a linear combination of monomials, where the coefficients $a, b, c$ are elements of a field. Every polynomial corresponds to a function that can be computed by evaluating the polynomials using a particular setting of $X_1, \ldots, X_n$.

A polynomial is called multilinear if every monomial is a product of distinct variable. For example:

$$X_1X_2X_3 + 3X_3X_7X_5 - 2X_1X_4.$$

A useful fact is that every function $f : \{0,1\}^n \to \mathbb{F}$ can be uniquely represented as a multilinear polynomial where the coefficients come from $\mathbb{F}$.
Part I

Communication
Deterministic Protocols

The concept of a conversation is so universal that one might be tempted to say that it requires no definition. However, to model a conversation mathematically, we need to isolate its critical features, and give rigorous definitions that capture these features. Even though many real-world conversations have no well-defined purpose, in this book we are interested in modeling conversations that do have a definite aim. Such a conversation takes place between two or more people, and it is productive only when the participants, or at least one of the participants, learns something new when the conversation is over.

An important feature of conversations is that they are inherent in computation. This is the reason communication complexity is so relevant to computer science. To give some intuition for why communication often plays a key role in computation, let us discuss one of the applications of communication complexity to proving lower bounds. We shall discuss how it can be used to give a lower bound on the area required to lay out digital chips. A chip-design specifies how to compute a function \( f(x_1, \ldots, x_n) \) by laying out the components of the chip on a grid, as in Figure 1.1. Each component either stores one of the inputs to the function, or does some computation on the values coming from adjacent components. It is vital to minimize the area used in the design, because the area directly relates to the cost, power consumption, reliability and speed of the chip.

Since there are \( n \) inputs, we certainly need area \( n \) to compute the function \( f \). Can we always find chip designs with area proportional to \( n \)? The framework of communication complexity can be used to show that many functions require much larger area, even as large as \( n^2 \), no matter what chip-design is used! The crucial insight is that the chip can be viewed as a protocol for two people to have a conversation that determines the value of \( f \). If a chip-design has area \( A \), one can argue that there must be a way to cut the chip into two

---

1 Thompson, 1979

Figure 1.1: Any chip arranged on a grid can be divided into two pieces with a small number of crossing wires.
parts containing a similar number of inputs, so that only \( \approx \sqrt{A} \) wires are cut, as in the figure. Imagine that one side of the chip represents one person, and the other side represents another person. Then the chip-design describes how \( f \) can be computed by two people, each of whom know only part of the input to \( f \), with a conversation whose length is proportional to the number of wires that were cut. If we can use ideas from communication complexity to show that \( f \) requires a conversation of length \( t \), this proves that the area \( A \) must be at least \( \approx t^2 \), no matter how the components of the chip are laid out. In this book, we shall develop ideas that can be used to argue that most functions require conversations of length proportional to \( n \), and so area on the chip that is proportional to \( n^2 \).

A second example comes from a classical result about Turing machines. Turing machines are widely regarded as universal models of computation—anything that is efficiently computable is efficiently computable by a Turing machine. A Turing machine can be thought of as a program written for a computer that has access to one or more tapes. Each tape has a head that points at a location on the tape. In each step of computation, the machine can read or write a symbol at the location corresponding to a head, or move the head to an adjacent location. Clearly, a Turing machine with more tapes is more powerful than a Turing machine with fewer tapes, but how much more powerful? A classical result shows that one can simulate a Turing machine that has access to 2 tapes with a Turing machine that has access to just 1 tape. However, the simulation may increase the number of steps of the computation by a factor of \( n \), where \( n \) is the length of the input. One can use communication complexity to show that this slow-down is unavoidable.

To see why this is the case, let us consider the problem of computing disjointness. Here the input is two sets of elements coming from an \( n \) element universe, \( X, Y \subseteq [n] \). We want to compute whether or not the sets are disjoint—namely whether or not there is an element that is in both sets. See Figure 1.2. We shall see that if Alice knows \( X \) and Bob knows \( Y \), they must communicate \( \Omega(n) \) bits in order to compute whether or not \( X \) and \( Y \) are disjoint. Now, a Turing machine with access to two tapes can compute disjointness in \( O(n) \) steps. If the sets are represented by their indicator vectors \( x, y \in \{0, 1\}^n \), with \( x_i = 1 \) if and only if \( i \in X \) and \( y_i = 1 \) if and only if \( i \in Y \), then the machine can copy \( y \) to the second tape, and scan both \( x \) and \( y \) to find an index \( i \) with \( x_i = 1 = y_i \). All of these operations can be carried out in \( O(n) \) steps.

However, one can use communication complexity to prove that a Turing machine with 1 tape must take at least \( \Omega(n^2) \) steps to compute disjointness. The idea is to show that any such machine that
computes disjointness in $\ll n^2$ steps can be used by Alice and Bob to compute disjointness using $\ll n$ bits of communication. Intuitively Alice and Bob can write down their inputs on the single tape of the machine and try to simulate the execution of the machine. Neither of them knows the contents of the whole tape, but they can still execute the machine with a small amount of communication—every time the machine transitions to reading from Alice’s part of the tape to Bob’s part, she sends him a short message to indicate the line of code that should be executed next. Bob then continues executing the machine. One can show that this simulation can be carried out in such a way that each message sent between Alice and Bob corresponds to $\Omega(n)$ steps of the Turing machine. So, we end up a protocol that computes $\ll n$ bits and still computes disjointness—this is impossible.

The two examples we have discussed above give some feel for communication problems and why we are interested in studying them. Now we discuss several other interesting examples of communication problems. Following these examples, we will begin giving rigorous definitions to specify exactly what we mean by communication complexity.

Some Protocols

A communication protocol specifies a way for a set of people to have a conversation. Each person has access to a different source of information, which is modeled as an input to the protocol. The protocol itself is assumed to be known to all the people that are involved in executing it. Their goal is to learn some feature of all the information
that they collectively know.

**Equality** Suppose two people named Alice and Bob are given two $n$-bit strings. Alice is given $x$ and Bob is given $y$, and they want to know if $x = y$. There is a trivial solution: Alice can send her input $x$ to Bob, and Bob can let her know if $x = y$. This is a deterministic protocol that takes $n + 1$ bits of communication. Interestingly, we shall prove that no deterministic protocol is more efficient. On the other hand, for every number $k$, there is a randomized protocol that uses only $k + 1$ bits of communication and errs with probability at most $2^{-k}$: the parties can use randomness to hash their inputs and compare the hashes. More on this in Chapter 3.

**Median** Suppose Alice is given a list of numbers from $[n]$ and Bob is given a different list of numbers from $[n]$. They want to compute the median element of these lists. If $t$ is the total number of elements in their lists, this is the $\lceil t/2 \rceil$'th element after the lists are combined and sorted. There is a simple protocol that takes $O(\log n \cdot \log t)$ bits of communication. In the first step, Alice and Bob each announce how many of their elements are at most $n/2$. It takes $O(\log t)$ bits of communication to encode these numbers. If there are $k$ elements that are at most $n/2$ and $k \geq \lceil t/2 \rceil$, then Alice and Bob can safely discard the all the elements that are larger than $n/2$ and recurse on the numbers that remain. If $k < \lceil t/2 \rceil$, then Alice and Bob can recurse after throwing out all the numbers that are at most $n/2$. Continuing in this way, each step takes $O(\log t)$ bits of communication, and there can be at most $O(\log n)$ steps before all the numbers have been discarded, or all elements are the same.

**Clique and Independent Sets** Here Alice and Bob are given a graph $G$ on $n$ vertices. In addition, Alice knows a clique $C$ in the graph, and Bob knows an independent set $I$ in the graph. They want to know whether $C$ and $I$ share a common vertex or not, and they want to know this using a short conversation. Describing $C$ or $I$ takes about $n$ bits, because in general the graph may have $2^n$ cliques or $2^n$ independent sets. So, if Alice and Bob try to tell each other what $C$ or $I$ is, that will lead to a very long conversation.

Here we discuss a clever interactive protocol allowing Alice and Bob to have an extremely short conversation for this task. They will send at most $O(\log^2 n)$ bits. If $C$ contains a vertex $v$ of degree less than $n/2$, Alice sends Bob the name of $v$. This takes just $O(\log n)$ bits of communication. Now, either $v \in I$, or Alice and Bob can safely discard all the non-neighbors of $v$, since these cannot be a part of $A$. This eliminates at least $n/2$ vertices from the graph. Similarly, if $I$ contains a vertex $v$ of degree at least $n/2$, the terms deterministic and randomized will be formally defined later in the book.
In Chapter 4, we shall prove that when \( k > \log n \), it is enough for each party to announce the number of elements in \( \mathbb{N} \) that are in \( i \) of the sets visible to her for \( i = 0, 1, \ldots, k \).

Bob sends Alice the name of \( v \). Again, either \( v \in C \), or Alice and Bob can safely delete all the neighbors of \( v \) from the graph, which eliminates about \( n/2 \) vertices. If all the vertices in \( C \) have degree more than \( n/2 \), and all the vertices in \( I \) have degree less than \( n/2 \), then \( C \) and \( I \) do not share a vertex. The conversation can safely terminate. So, in each round of communication, either the parties know that \( C \cap I = \emptyset \), or the number of vertices is reduced by a factor of 2. After \( k \) rounds, the number of vertices is at most \( n/2^k \). If \( k \) exceeds \( \log n \), the number of vertices left will be less than 1, and Alice and Bob will known if \( C \) and \( I \) share a vertex or not. This means that at most \( \log n \) vertices can be announced before the protocol ends, proving that at most \( O(\log^2 n) \) bits will be exchanged before Alice and Bob learn what they wanted to know.

One can show if the conversation involves only one message from each party, then at least \( \Omega(n) \) bits must be revealed for the parties to discover what they want to know. So, interaction is vital to bringing down the length of the conversation.

**Disjointness with sets of size \( k \)** Alice and Bob are given two sets \( A, B \subseteq \mathbb{N} \), each of size \( k \ll n \), and want to know if the sets share a common element. Alice can send her set to Bob, which takes \( \log \binom{n}{k} \approx k \log n \) bits of communication. There is a randomized protocol that uses only \( O(k) \) bits of communication. Alice and Bob sample a random sequence of sets in the universe, Alice announces the name of the first set that contains \( A \). If \( A \) and \( B \) are disjoint, this eliminates half of \( B \). In Chapter 3, we prove that repeating this procedure gives a protocol with \( O(k) \) bits of communication.

**Disjointness with \( k \) parties** The input is \( k \) sets \( A_1, \ldots, A_k \subseteq \mathbb{N} \), and there are \( k \) parties. The \( i \)'th party knows all the sets except for the \( i \)'th one. The parties want to know if there is a common element in all sets. We know of a clever deterministic protocol with \( O(n/2^k) \) bits of communication, and we know that \( \Omega(n/4^k) \) bits of communication are required. No randomized protocol can have communication less than \( \Omega(\sqrt{n}/2^k) \), but we do not know how to use randomness to reduce the communication.

**Summing 3 numbers** The input is three numbers \( x, y, z \in \mathbb{N} \). Alice knows \((x, y)\), Bob knows \((y, z)\) and Charlie knows \((x, z)\). The parties want to know whether or not \( x + y + z = n \). Alice can tell Bob \( x \), which would allow Bob to announce the answer. This takes \( O(\log n) \) bits of communication. There is a clever deterministic protocol that communicates \( \sqrt{\log n} \) bits, and one can show that the length of any deterministic conversation must increase with \( n \). To contrast, there is a randomized protocol that solves the problem
with a conversation whose length is a constant.

**Pointer Chasing** The input consists of two functions \( f, g : [n] \to [n] \), where Alice knows \( f \) and Bob knows \( g \). Let \( a_0, a_1, \ldots, a_k \in [n] \) be defined by setting \( a_0 = 1 \), and \( a_i = f(g(a_{i-1})) \). The goal is to compute \( a_k \). There is a simple \( k \) round protocol with communication \( O(k \log n) \) that solves this problem, but any protocol with fewer than \( k \) rounds requires \( \Omega(n) \) bits of communication.

**Rigorously Defining Communication Protocols**

Here we define exactly what we mean by a 2-party deterministic communication protocol. The definition is meant to capture a conversation between the two parties. To be useful, the definition must be fairly general. It should capture settings where the parties have access to different information, and want to use that information to generate meaningful messages. Later messages should be allowed to depend on earlier ones.

Suppose Alice’s input is an element from a set \( \mathcal{X} \) and Bob’s input is an element from a set \( \mathcal{Y} \). A communication protocol (see Figure 1.4) is an algorithm to generate a conversation between Alice and Bob. A protocol \( \pi \) is specified by a rooted binary tree. Every internal vertex \( v \) has 2 children. Every internal vertex \( v \) is associated with either the first or second party, and a function \( f_v : \mathcal{X} \to \{0, 1\} \), or \( f_v : \mathcal{Y} \to \{0, 1\} \) mapping the input known to that party to a bit, that can be viewed as a child of \( v \).

Given inputs \( (x, y) \in \mathcal{X} \times \mathcal{Y} \), the outcome of the protocol \( \pi(x, y) \) is a leaf in the protocol tree, computed as follows. The parties begin by setting the current vertex to be the root of the tree. If the first party is associated with the current vertex \( v \), she announces the value \( f_v(x) \). Although there are many other reasonable ways to define a communication protocol, they are all captured by the definitions we give here, up to some small change in the length of the communication.

The setup is analogous for \( k \) party protocols. Let \( \mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k \) be \( k \) sets. A \( k \)-party communication protocol defines a way for \( k \) parties to communicate information about their inputs, where the \( i \)’th party gets an input from the set \( \mathcal{X}_i \). Every vertex \( v \) is associated with a party \( i \) and a function \( f_v : \mathcal{X}_i \to \{0, 1\} \).
Similarly, if the second party is associated with \( v \), he announces the value \( f_v(y) \). Both parties set the new current vertex to be the child of \( v \) indicated by the announced value of \( f_v \). This process is repeated until the current vertex is a leaf, and this leaf is the outcome of the protocol. So, the inputs \((x,y)\) induce a path from the root of the protocol tree to the leaf \( \pi(x,y) \). This path corresponds to the conversation between the parties.

Notice that a protocol itself is not a conversation as we usually understand it—rather the protocol produces a conversation when it is executed. Thus, the protocol encodes all possible messages that may be sent by the parties during the conversation that they have about their inputs. It might seem strange to require that the protocol must encode all possible messages; after all, in real life we rarely know exactly what we would say at a particular point in a conversation until it is time to speak. Even when messages are generated by algorithms, as in the example of chip-design, they usually are not computed until they need to be transmitted. The point of the definition we have given here is that the focus is firmly on the number of bits communicated. We do not account for the methods used to generate the messages sent, or the time it takes to compute them. We allow the protocol designer to pick the best possible message to send ahead of time. This choice gives us a clean model, and it allows us to bring many tools from mathematics to bear on understanding the model. So, we choose to give the protocol an unusual amount of power in order to obtain a mathematically clean model.

The length of the protocol \( \pi \), denoted \( \| \pi \| \), is the depth of the protocol tree\(^3\). It is the length of the longest possible conversation that may occur when the protocol is executed. The number of rounds of the protocol is \( k \) if \( k \) is the smallest number such that the execution of the protocol involves the parties exchanging at most \( k \) binary strings. In other words, it is the maximum, over all root-leaf paths, of the number of alternations between the players along the path. In some practical applications—for example protocols that run between computers on the internet—conversations consist of only a few rounds of interaction, so it makes sense to limit the discussion to bounded round protocols. In other applications—like algorithms that exchange information between two parts of the same chip—it is not very expensive to have many rounds of interaction.

Given a function \( g : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z} \) we say that \( \pi \) computes \( g \) if \( \pi(x,y) \) determines \( g(x,y) \) for every input \((x,y) \in \mathcal{X} \times \mathcal{Y}\). In other words, we say \( \pi \) computes \( g \) if there is a map \( D \) from the leaves of the protocol tree to \( \mathcal{Z} \) so that \( D(\pi(x,y)) = g(x,y) \) for all \( x,y \). The communication complexity of a function \( g \) is \( c \) if there is protocol of

\(^3\)The length of the longest path from root to leaf.

For example, suppose Alice sends 2 bits, then Bob sends 3 bits, and then Alice sends 1 bit to end the protocol. The length is 6 and the number of rounds is 3.

We could have chosen a different definition of when the protocol computes a function—we might require only that any party that knows the messages of the protocol and one of the inputs can deduce \( g(x,y) \). This distinction is sometimes important, but for boolean functions it is not important, because any party that knows the value of \( g(x,y) \) can announce it with one more bit of communication.
length $c$ that computes $g$, but no protocol can compute the function with less than $c$ bits of communication.

Let us make some basic observations that follow from these definitions:

**Fact 1.1.** The number of rounds in $\pi$ is always at most $\|\pi\|$. 

Since the number of leaves in a rooted binary tree of depth $d$ is at most $2^d$, we have:

**Fact 1.2.** The number of leaves in the protocol tree of $\pi$ is at most $2^{\|\pi\|}$. 

### Counting Arguments

One way to understand how easy or hard it is to compute functions with efficient protocols is via a counting argument. The argument will show that almost all functions require very large communication by comparing the number of possible functions with the number of functions that can possibly be computed by short protocols.

Let us say that we are interested in computing a function $f : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$. There are $2^{2n}$ such functions. Now, suppose we are interested in functions that can be computed using $c$ bits of communication. We wish to find an upper bound on the number of such functions.

One way to do this is by counting the number of deterministic protocols. Every protocol can compute at most 1 function, so the number of protocols is certainly an upper bound on the number of functions of communication complexity $c$.

A protocol tree of depth $c$ has

$$1 + 2 + 4 + \ldots + 2^c = 2^{c+1} - 1 \leq 2^{c+1}$$

nodes. Each node belong either to Alice or Bob, and is associated with a function describing how they compute the next bit to transmit. So, the number of choices for each node is $2 \cdot 2^{2^i} = 2^{2^i+1} \leq 2^{2^i+1}$. This gives at most

$$\left(2^{2n+1}\right)^{2^{c+1}} = 2^{2n+c+2}$$

functions that can be computed with communication $c$. We see that the fraction of such functions among all functions is

$$\frac{2^{2n+c+2}}{2^{2n}} = 2^{n+c+2-2n},$$

which is extremely small whenever $c < n - 2$. 

Counting arguments are a standard way to establish the existence of hard functions in a given computational model.

To count the number of such functions $f$, note that there are $2^{2n}$ different inputs, and 2 choices for available for each input.

In general, $1 + x + x^2 + \ldots + x^r = (x^{r+1} - 1)/(x - 1)$. 
Rectangles

The concept of combinatorial rectangles plays a crucial role in our understanding of communication complexity. A rectangle is a subset of the form $R = A \times B \subseteq \mathcal{X} \times \mathcal{Y}$. See Figure 1.5.

**Lemma 1.3.** A set $R \subseteq \mathcal{X} \times \mathcal{Y}$ is a rectangle if and only if whenever $(x, y), (x', y') \in R$, we have $(x', y), (x, y') \in R$.

**Proof.** If $R = A \times B$ is a rectangle, then $(x, y), (x', y') \in R$ means that $x, x' \in A$ and $y, y' \in B$. Thus $(x, y'), (x', y) \in A \times B$. On the other hand, if $R$ is an arbitrary set with the given property, if $R$ is empty, it is a rectangle. If $R$ is not empty, let $(x, y) \in R$ be an element. Define $A = \{x' : (x', y) \in R\}$ and $B = \{y' : (x, y') \in R\}$. Then by the promised property of $R$, we have $A \times B \subseteq R$, and for every element $(x', y') \in R$, we have $x' \in A, y' \in B$, so $R \subseteq A \times B$. Thus $R = A \times B$. 

If a function is defined by a rectangle $A \times B$, then it certainly has a very simple communication protocol. Indeed, if Alice and Bob want to know if their inputs belong to the rectangle or not, Alice can send a bit indicating if $x \in A$, and Bob can send a bit indicating if $y \in B$. These two bits determine whether or not $(x, y) \in A \times B$.

The importance of rectangles stems from the fact that every protocol can be described using rectangles. For every vertex $v$ in a protocol $\pi$, let $R_v \subseteq \mathcal{X} \times \mathcal{Y}$ denote the set of inputs $(x, y)$ that would lead the
Figure 1.6: A partition of the space into 6 rectangles.

protocol to pass through the vertex $v$ during the execution, and let

\[ X_v = \{ x \in \mathcal{X} : \exists y \in \mathcal{Y} \ (x, y) \in R_v \}, \]
\[ Y_v = \{ y \in \mathcal{Y} : \exists x \in \mathcal{X} \ (x, y) \in R_v \}. \]

Lemma 1.4. For every vertex $v$ in the protocol tree, the set $R_v$ is a rectangle with $R_v = X_v \times Y_v$. Moreover, the rectangles given by all the leaves of the protocol tree form a partition of the set of inputs $\mathcal{X} \times \mathcal{Y}$.

Proof. The lemma follows by induction. For the root vertex $r$, we see that $R_r = \mathcal{X} \times \mathcal{Y}$, so indeed the lemma holds. Now consider an arbitrary vertex $v$ such that $R_v = X_v \times Y_v$. Let $u, w$ be the children of $v$ in the protocol tree. Suppose the first party is associated with $v$, and $u$ is the vertex that the players move to when $f_v(x) = 0$. Define:

\[ X_u = \{ x \in X_v : f_v(x) = 0 \}, \]
\[ X_w = \{ x \in X_v : f_v(x) = 1 \}. \]

We see that $X_u$ and $X_w$ form a partition of $X_v$, and $R_u = X_u \times Y_u$ and $R_w = X_w \times Y_w$ form a partition of $R_v$. In this way, we see that the leaves in the protocol tree induce a partition of the entire space of inputs into rectangles.

Often, the purpose of the protocol is to compute a function $g : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$. In this case, it is useful to understand the concept of a monochromatic rectangle. We say that a rectangle $R \subset \mathcal{X} \times \mathcal{Y}$ is monochromatic with respect to $g$ if $g$ is constant on $R$. See Figure 1.7.

We say that the rectangle is 1-monochromatic if $g$ only takes the value 1 on the rectangle, and 0-monochromatic if $g$ only takes the value 0 on $R$.

Fact 1.5. If a protocol $\pi$ computes a function $g : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, and $v$ is a leaf in $\pi$, then $R_v$ is a monochromatic rectangle.
Combining this fact with Lemmas 1.2 and 1.4 gives:

**Theorem 1.6.** If the communication complexity of \( g : \mathcal{X} \times \mathcal{Y} \to \{0, 1\} \) is \( c \), then \( \mathcal{X} \times \mathcal{Y} \) can be partitioned into at most \( 2^c \) monochromatic rectangles.

**Balancing Protocols**

**Lemma 1.2** is sharp when the protocol tree is a full binary tree; then the number of leaves in the protocol tree is exactly \( 2^c \). Does it ever make sense to have a protocol tree that is not balanced? It turns out that one can always balance an unbalanced tree while approximately preserving the number of nodes in the tree.

**Theorem 1.7.** If \( \pi \) is a protocol with \( \ell \) leaves, then there is a protocol that computes the outcome \( \pi(x, y) \) with length at most \( \lceil 2 \log_{3/2} \ell \rceil \).

To prove the theorem, we need a well-known lemma about trees.

**Lemma 1.8.** In every protocol tree with \( \ell > 1 \) leaves, there is a vertex \( v \) such that the subtree rooted at \( v \) contains \( r \) leaves, and \( \ell/3 \leq r < 2\ell/3 \).

**Proof.** Consider the sequence of vertices \( v_1, v_2, \ldots \) defined as follows. The vertex \( v_1 \) is the root of the tree, which is not a leaf by the assumption on \( \ell \). For each \( i > 0 \), the vertex \( v_{i+1} \) is the child of \( v_i \) that has the most leaves under it, breaking ties arbitrarily. Let \( \ell_i \) denote the number of leaves in the subtree rooted at \( v_i \). Then, \( \ell_{i+1} \geq \ell_i/2 \), and \( \ell_{i+1} < \ell_i \). Since \( \ell_1 = \ell \), and the sequence is decreasing until it hits 1, there must be some \( i \) for which \( \ell_i/3 \leq \ell_i < 2\ell_i/3 \).

**Proof of Theorem 1.7.** In each step of the balanced protocol (see Figure 1.9), the parties pick a vertex \( v \) as promised by Lemma 1.8, and decide whether \( (x, y) \in R_v \) using two bits of communication. That is, Alice sends a bit indicating if \( x \in \mathcal{X}_v \) and Bob sends a bit indicating if \( y \in \mathcal{Y}_v \). If \( x \in \mathcal{X}_v \) and \( y \in \mathcal{Y}_v \), then the parties repeat the procedure at the subtree rooted at \( v \). Otherwise, the parties delete the vertex \( v \)
and its subtree from the protocol tree and continue the simulation. In each step, the number of leaves of the protocol tree is reduced by a factor of at least $\frac{2}{3}$, so there can be at most $\log_{3/2} \ell$ such steps. \hfill \square

\textit{From Rectangles to Protocols}

Given \textbf{Theorem 1.6}, one might wonder whether every partition of the inputs to rectangles can be realized by a protocol. While this is not true in general—see Figure 1.10 for an example—we can show that a small partition yields an efficient protocol\textsuperscript{4}.

\textbf{Theorem 1.9.} If $X \times Y$ can be covered by $2^c$ monochromatic rectangles with respect to $g$, then there is a protocol that computes $g$ with $O(c^2)$ bits of communication.

One can prove \textbf{Theorem 1.9} by reduction to the clique versus independent set problem—we leave the details to Exercise 1.2. The theorem is actually tight, as we discuss later in this chapter—see Claim 1.31 and the discussion that follows it.

Here we prove a closely related statement:

\textbf{Theorem 1.10.} Let $\mathcal{R}$ be a collection of $2^c$ rectangles that form a partition of $X \times Y$. For $(x, y) \in X \times Y$, let $R_{x,y}$ be the unique rectangle in $\mathcal{R}$ that contains $(x, y)$. Then there is a protocol of length $O(c^2)$ that on input $(x, y)$ computes $R_{x,y}$.

Recent work\textsuperscript{5} has shown that there is function $g$ under which the inputs can be partitioned into $2^c$ monochromatic rectangles, yet no protocol can compute $g$ using $o(c^2)$ bits of communication, showing that \textbf{Theorem 1.10} is tight.

\textsuperscript{4}Yannakakis, 1991; and Aho et al., 1983

\textsuperscript{5}Göös et al., 2015; and Kothari, 2015
We now prove Theorem 1.10. The parties are given inputs \((x, y)\) and know a collection of rectangles \(\mathcal{R}\) that partition the set of inputs. The aim of the protocol is to find the unique rectangle containing \((x, y)\). In each round of the protocol, one of the parties announces the name of a rectangle in \(\mathcal{R}\). We shall ensure that each such announcement allows the parties to discard at least half of the remaining rectangles.

A key concept we need is that of rectangles intersecting horizontally and vertically. We say that two rectangles \(R = A \times B\) and \(R' = A' \times B'\) intersect horizontally if \(A\) intersects \(A'\), and intersect vertically if \(B\) intersects \(B'\). The basic observation is that if \(x \in A \cap A'\) and \(y \in B \cap B'\), then \((x, y) \in A \times B\) and \((x, y) \in A' \times B'\). This proves:

**Fact 1.11.** If \(R, R'\) are disjoint rectangles, they cannot intersect both horizontally and vertically.

This leads to the following definition:

**Definition 1.12.** Say that a rectangle \(R = (A \times B) \in \mathcal{R}\) is

- horizontally good if \(x \in A\), and \(R\) horizontally intersects at most half of the rectangles in \(\mathcal{R}\), and

- vertically good if \(y \in B\), and \(R\) vertically intersects at most half of the rectangles in \(\mathcal{R}\).

Say that \(R\) is good if it is either horizontally good or vertically good.

We claim that there is always at least one good rectangle:

**Claim 1.13.** \(\mathcal{R}_{x,y}\) is good.

**Proof.** Fact 1.11 implies that every rectangle in \(\mathcal{R}\) does not intersect \(\mathcal{R}_{x,y}\) both horizontally and vertically. Thus either at most half of the
rectangles in $\mathcal{R}$ intersect $R_{x,y}$ horizontally, or at most half of them intersect $R_{x,y}$ vertically. See Figure 1.11.

In each step of the protocol, one of the parties announces the name of a good rectangle $R$, which must exist, since $\mathcal{R}_{x,y}$ is good. This leads to half of the rectangles in $\mathcal{R}$ being discarded. If $R$ is horizontally good, then the players can discard all the rectangles that do not intersect $R$ horizontally. Otherwise, they discard all the rectangles that do not intersect $R$ vertically.

When only one rectangle remains, the protocol achieves its goal. Since $\mathcal{R}$ can survive at most $c$ such discards, and a rectangle in the family can be described with $c$ bits of communication, the communication complexity of the protocol is at most $O(c^2)$.

Lower Bounds Based on Rectangles

We turn to proving that some problems do not have efficient protocols. The easiest way to prove a lower bound is to use the characterization provided by Theorem 1.6. If we can show that the inputs cannot be partitioned into $2^c$ monochromatic rectangles, or do not have large monochromatic rectangles, then that proves that there is no protocol computing the function with $c$ bits of communication.

Size of Monochromatic Rectangles

Equality Consider the equality function $\text{EQ} : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}$ defined as:

$$\text{EQ}(x,y) = \begin{cases} 1 & \text{if } x = y, \\ 0 & \text{otherwise.} \end{cases} \quad (1.1)$$

Alice can send Bob her input, and Bob can respond with the value of a function, giving a protocol of length $n + 1$.

Is there a protocol with complexity $n$? Since any such protocol induces a partition into $2^n$ monochromatic rectangles, a first attempt at proving a lower bound might try to find and show that there is no large monochromatic rectangle. If we could prove that, then we could argue that many monochromatic rectangles are needed to cover the whole input. However, the equality function does have large monochromatic rectangles. For example, the rectangle $R = \{(x,y) : x_1 = 0, y_1 = 1\}$. This is a rectangle with density $\frac{1}{4}$, and it is monochromatic, since $\text{EQ}(x,y) = 0$ for every $(x,y) \in R$.  

**Figure 1.13**: A protocol finding the unique rectangle containing the input from a partition of the space into rectangles.

**Figure 1.14**: The equality function does have large monochromatic rectangles.
The solutions is to show that equality does not have a large 1-monochromatic rectangle, and argue that this is good enough to prove a lower bound.

**Claim 1.14.** If $R$ is a 1-monochromatic rectangle, then $|R| = 1$.

**Proof.** Observe that if $x \neq x'$, then the points $(x, x)$ and $(x', x')$ cannot be in the same monochromatic rectangle. Otherwise, by Lemma 1.3, the element $(x, x')$ would also have to be included in this rectangle. Since the rectangle is monochromatic, we would have $\text{EQ}(x, x') = \text{EQ}(x, x)$, which is a contradiction.  

Since there are $2^n$ inputs $x$ with $\text{EQ}(x, x) = 1$, this means $2^n$ rectangles are needed to cover such inputs. There is also at least one more 0-monochromatic rectangle. So, we need more than $2^n$ monochromatic rectangles to cover all the inputs. We conclude:

**Theorem 1.15.** The deterministic communication complexity of $\text{EQ}$ is $n + 1$.

**Inner-Product** The Hadamard Matrix is a well known example of a matrix that has many nice combinatorial properties. It corresponds to the inner-product function $\text{IP} : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}$

$$\text{IP}(x, y) = \sum_{i=1}^{n} x_i y_i \mod 2. \quad (1.2)$$

This function can be viewed as the inner product $\text{IP} : \mathbb{F}_2^n \times \mathbb{F}_2^n \rightarrow \mathbb{F}_2$. Here we can leverage some linear algebra to place a bound on the size of the largest monochromatic rectangle.

We shall show in Chapter 5 that this function satisfies the strong property that the fraction of 1’s is very close to the fraction of 0’s in every rectangle that is not exponentially small.
Disjointness

Next, consider the disjointness function $\text{Disj} : 2^{[n]} \times 2^{[n]} \to \{0, 1\}$ defined by:

$$
\text{Disj}(X, Y) = \begin{cases} 
1 & \text{if } X \cap Y = \emptyset, \\
0 & \text{otherwise.}
\end{cases}
$$

(1.3)

Alice can send her whole set $X$ to Bob, which gives a protocol with communication $n + 1$. Can we prove that this is optimal? Once again, this function does have large monochromatic rectangles, for example the rectangle $R = \{(X, Y) : 1 \in X, 1 \in Y\}$, but we shall show that there are no large monochromatic 1-rectangles. Indeed, suppose $R = A \times B$ is a 1-monochromatic rectangle. Let $X' = \cup_{X \in A} X$ and $Y' = \cup_{Y \in B} Y$. Then $X'$ and $Y'$ must be disjoint, so $|X'| + |Y'| \leq n$. On the other hand, $|A| \leq 2^{|X'|}$, $|B| \leq 2^{|Y'|}$, so $|R| = |A||B| \leq 2^n$. We have shown:

Claim 1.18. Every 1-monochromatic rectangle of $\text{Disj}$ has size at most $2^n$. In this proof, we use several facts from linear algebra that are discussed in the Conventions chapter of this book.
On the other hand, the number of disjoint pairs \((X, Y)\) is exactly \(3^n\). That’s because for every element of the universe, there are 3 possibilities: to be in \(X\), be in \(Y\) or be in neither. Thus, at least \(3^n/2^n = 2^{(\log 3 - 1)n}\) monochromatic rectangles are needed to cover the 1’s of \(\text{Disj}\), and so:

**Theorem 1.19.** The deterministic communication complexity of \(\text{Disj}\) is at least \((\log 3 - 1)n\).

### Richness

Sometimes we need to understand asymmetric communication protocols, where we need separate bounds on the communication complexity of Alice and Bob. The concept of *richness*\(^6\) is useful here:

**Definition 1.20.** A function \(g : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}\) is said to be \((u, v)\)-rich if there is a set \(V \subseteq \mathcal{Y}\) of size \(|V| = v\) such that for all \(y \in V\), we have \(|\{x \in \mathcal{X} : g(x, y) = 1\}| \geq u\).

Richness allows us to carefully control the shape of large 1-monochromatic rectangles induced by asymmetric protocols:

**Lemma 1.21.** If \(g : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}\) is \((u, v)\)-rich with \(u, v > 0\), and if there is a protocol for computing \(g\) where Alice sends at most \(a\) bits and Bob sends at most \(b\) bits, then \(g\) admits a \(\frac{u}{2}\times\frac{v}{2^{a+b}}\) 1-monochromatic rectangle.

**Proof.** The statement is proved inductively. For the base case, if the protocol does not communicate at all, then \(g(x, y) = 1\) for all \(x \in \mathcal{X}, y \in \mathcal{Y}\), and the statement holds.

If Bob sends the first bit of the protocol, then Bob partitions \(\mathcal{Y} = \mathcal{Y}_0 \cup \mathcal{Y}_1\). One of these two sets must have at least \(v/2\) of the inputs \(y\) that show that \(g\) is \((u, v)\)-rich. By induction, this set contains a \(\frac{u}{2}\times\frac{v/2}{2^{a+b+1}}\) 1-monochromatic rectangle, as required. On the other hand, if Alice sends the first bit, then this bit partitions \(\mathcal{X}\) into two sets \(\mathcal{X}_0, \mathcal{X}_1\). Every input \(y \in \mathcal{Y}\) that has \(u\) ones must have \(u/2\) ones in either \(\mathcal{X}_0\) or \(\mathcal{X}_1\). Thus there must be at least \(v/2\) choices of inputs \(y \in \mathcal{Y}\) that have \(u/2\) ones for \(g\) restricted to \(\mathcal{X}_0 \times \mathcal{Y}\) or for \(g\) restricted to \(\mathcal{X}_1 \times \mathcal{Y}\). By induction, we get that there is a 1-monochromatic rectangle with dimensions \(\frac{u/2}{2^{a-1}} \times \frac{v/2}{2^{b+1}}\), as required.

Now let us see some examples where richness can be used to prove lower bounds.

**Lopsided Disjointness** Suppose Alice is given a set \(X \subseteq \{n\}\) of size \(k < n\), and Bob is given a set \(Y \subseteq \{n\}\), and they want to compute whether the sets are disjoint or not. Now the obvious protocol is for Alice to send her input to Bob, which takes \(\log \binom{n}{k}\) bits.\(^7\) How-

---

\(^6\)Miltersen et al., 1998

\(^7\)In Chapter 2, we show that the communication complexity of this problem is at least \(\log \binom{n}{k}\).
ever, what can we say about the communication of this problem if Alice is forced to send much less than $\log(\binom{n}{k})$ bits?

To prove a lower bound, we need to analyze rectangles of a certain shape. We restrict our attention to special family of sets for Alice and Bob, as in Figure 1.17. Let $n = 2kt$, and suppose $Y$ contains exactly one element of $2i-1, 2i$, for each $i$, and that $X$ contains exactly one element from $2t(i-1)+1, \ldots, 2ti$ for each $i \in [k]$.

Claim 1.22. If $A \times B$ is a 1-monochromatic rectangle, then $|B| \leq 2^{kt-k|A|^{1/k}}$.

Proof. We claim that $|\bigcup_{X \in A} X| \geq k|A|^{1/k}$. Indeed, if the union $\bigcup_{X \in A} X$ has $a_i$ elements in $\{2t(i-1)+1, \ldots, 2ti\}$, then

$$\left|\bigcup_{X \in A} X\right| = \sum_{i=1}^{k} a_i \geq k \left(\prod_{i=1}^{k} a_i\right)^{1/k} \geq k|A|^{1/k},$$

by the arithmetic-mean geometric-mean inequality.

$\bigcup_{X \in A} X$ cannot contain both $2i, 2i+1$ for any $i$, since one of these two elements belongs to a set in $B$. Thus, the number of possible choices for sets in $B$ is at most $2^{kt-k|A|^{1/k}}$. $\square$

The disjointness matrix here is at least $(t^k, 2^{kt})$-rich, since every choice $Y$ allows for $t^k$ possible choices for $X$ that are disjoint. By Lemma 1.21, any protocol where Alice sends $a$ bits and Bob sends $b$ bits induces a 1-monochromatic rectangle with dimensions $t^k/2^a \times 2^{kt-a-b}$, so Claim 1.22 gives:

$$2^{kt-a-b} \leq 2^{kt-kt/2^{a/k}}$$

$$\Rightarrow a + b \geq \frac{n}{2^{a/k+1}}.$$

We conclude:

Theorem 1.23. If $X, Y \subseteq [n], |X| = k$ and Alice sends at most $a$ bits and Bob sends at most $b$ bits in a protocol computing $\text{Disj}(X, Y)$, then $a + b \geq \frac{n}{2^{a/k+1}}$. 
For example, for \( k = 2 \), if Alice sends at most \( \log n \) bits to Bob, then Bob must send at least \( \Omega(\sqrt{n}) \) bits to Alice in order to solve lopsided disjointness.

**Span** Suppose Alice is given a vector \( x \in \{0,1\}^n \), and Bob is given a \( n/2 \) dimensional subspace \( V \subseteq \{0,1\}^{n/2} \). Their goal is figure out whether or not \( x \in V \). As in the case of disjointness, we start by claiming that the inputs do not have 1-monochromatic rectangles of a certain shape:

**Claim 1.24.** If \( A \times B \) is a 1-monochromatic rectangle, then \(|B| \leq 2^{n^2/2-n \log |A|}/2^{n^2/2} \).

**Proof.** The set of \( x \)'s in the rectangle spans a subspace of dimension at least \( \log |A| \). The number of \( n/2 \) dimensional subspaces that contain this span is thus at most \( \left(2^{n^2/2} - n \log |A|\right) \leq 2^{n^2/2-n \log |A|}/2^{n^2/2} \).

The problem we are working with is at least \((2^{n^2/2}, 2^{n^2/4}/n!)\)-rich, since there are at least \( 2^{n^2/4}/n! \) subspaces, and each contains \( 2^{n^2/2} \) vectors. Applying Lemma 1.21 and Claim 1.24, we get that if there is a protocol where Alice sends \( a \) bits and Bob sends \( b \) bits, then

\[
2^{n^2/4-a-b}/n! \leq 2^{n^2/2-n \log 2^{n^2/2-a}}
\]

\[
\Rightarrow b \geq n^2/4 - a(n+1) - n \log n.
\]

**Theorem 1.25.** If Alice sends \( a \) bits and Bob sends \( b \) bits to solve the span problem, then \( b \geq n^2/4 - a(n+1) - n \log n \).

For example, if Alice sends at most \( n/8 \) bits, then Bob must send at least \( \Omega(n^2) \) bits in order to solve the span problem. One of the players must send a linear number of the bits in their input.

**Fooling Sets**

A set \( S \subseteq X \times Y \) is called a **fooling set** if every monochromatic rectangle can share at most 1 element with \( S \). Fooling sets can be used to prove several basic lower bounds on communication.

**Greater-than** Our first example using fooling sets is the greater-than function, \( GT : [n] \times [n] \rightarrow \{0,1\} \), defined as:

\[
GT(x,y) = \begin{cases} 
1 & \text{if } x > y, \\
0 & \text{otherwise.}
\end{cases}
\]

The trivial protocol computing greater-than has complexity \( 1 + \lceil \log n \rceil \) bits, and we shall show that this is essentially tight. The
methods we used for the last two examples will surely not work here, because GT has large 0-monochromatic rectangles (like $R = \{(x, y) : x < n/2, y > n/2\}$) and large 1-monochromatic rectangles (like $R = \{(x, y) : x > n/2, y < n/2\}$). Instead we shall use a fooling set to prove the bound. Consider the set of $n$ points $S = \{(x, x)\}$. We claim:

**Claim 1.26.** Two points of $S$ cannot lie in the same monochromatic rectangle.

Indeed, if $R$ is monochromatic, and $x < x'$, but $(x, x), (x', x') \in R$, then since $R$ is a rectangle, $(x', x) \in R$. This contradicts the fact that $R$ is monochromatic, since $GT(x', x) \neq GT(x', x')$. So once again, we have shown that the number of monochromatic rectangles must be at least $n$, proving:

**Theorem 1.27.** The deterministic communication complexity of $GT$ is at least $\log n$.

**Disjointness** Fooling sets also allow us to prove tighter lower bounds on the communication complexity of disjointness. Consider the set $S = \{(X, \overline{X}) : X \subseteq |n|\}$, namely the set of pairs of sets and their complements. No monochromatic rectangle can contain two such pairs, because if such a rectangle contained $(X, \overline{X}), (Y, \overline{Y})$ for $X \neq Y$, then it would also contain both $(X, \overline{Y}), (Y, \overline{X})$, but at least one of the last pair of sets must intersect, while the first two pairs are disjoint. Since $|S| = 2^n$, and at least one more 0-monochromatic rectangle is required, this proves:

**Theorem 1.28.** The deterministic communication complexity of disjointness is $n + 1$.

**Krapchenko’s Method**

We end the lower bounds part of this chapter with a method for non-boolean relations. Let $\mathcal{X} = \{x \in \{0,1\}^n : \sum_{i=1}^n x_i = 0 \mod 2\}$ and $\mathcal{Y} = \{y \in \{0,1\}^n : \sum_{i=1}^n y_i = 1 \mod 2\}$. Since $\mathcal{X}$ and $\mathcal{Y}$ are disjoint, for every $x \in \mathcal{X}, y \in \mathcal{Y}$, there is an index $i$ such that $x_i \neq y_i$. Suppose Alice is given $x$ and Bob is given $y$, and they want to find such an index $i$. How much communication is required?

Perhaps the most trivial protocol is for Alice to send Bob her entire string, but we can use binary search to do better. Notice that

$$\sum_{i \leq n/2} x_i + \sum_{i > n/2} x_i \neq \sum_{i \leq n/2} y_i + \sum_{i > n/2} y_i \mod 2.$$ 

Alice and Bob can thus exchange $\sum_{i \leq n/2} x_i \mod 2$ and $\sum_{i \leq n/2} y_i \mod 2$. If these values are not the same, they can safely restrict their
attention to the strings \( x \leq n/2, y \leq n/2 \) and continue. On the other hand, if the values are the same, they can continue the protocol on the strings \( x > n/2, y > n/2 \). In this way, in every step they communicate 2 bits and eliminate half of their input string, giving a protocol of communication complexity \( 2 \log n \).

It is easy to see that \( \log n \) bits of communication are necessary, because that’s how many bits it takes to write down the answer. Now we shall prove that \( 2 \log n \) bits are necessary, using a variant of fooling sets. Consider the set of inputs

\[
S = \{ (x, y) \in X \times Y : x, y \text{ differ in only 1 coordinate} \}.
\]

\( S \) contains \( n \cdot 2^{n-1} \) inputs, since one can pick an input of \( S \) by picking \( x \in X \) and flipping any of the \( n \) coordinates. We will not be able to argue that every monochromatic rectangle must contain only one element of \( S \) or bound the number of elements in any way. Instead, we will prove that if such a rectangle does contain many elements of \( S \), then it is big:

Claim 1.29. Suppose \( R \) is a monochromatic rectangle that contains \( r \) elements of \( S \). Then \( |R| \geq r^2 \).

The key observation here is that two distinct elements \((x, y), (x, y') \) in \( S \) cannot be in the same monochromatic rectangle. For if the rectangle was labeled \( i \), then \((x, y), (x, y') \) must disagree in the \( i \)'th coordinate, but since they both belong to \( S \) we must have \( y = y' \). Similarly we cannot have two distinct elements \((x, y), (x', y) \) in \( S \) that belong to the same monochromatic rectangle. Thus, if \( R = A \times B \) has \( r \) elements of \( S \), we must have \(|A| \geq r, |B| \geq r\), proving that \(|R| \geq r^2\).

Now suppose there are \( t \) monochromatic rectangles that partition the set \( S \), and the \( i \)'th rectangle covers \( r_i \) elements of \( S \). Then \( |S| = \sum_{i=1}^{t} r_i \), but since the rectangles are disjoint, \( 2^{2n-2} \geq \sum_{i=1}^{t} r_i^2 \). Using these facts and the Cauchy-Schwartz inequality:

\[
2^{2n-2} \geq \sum_{i=1}^{t} r_i^2 \geq \left( \frac{\sum_{i=1}^{t} r_i}{\sqrt{t}} \right)^2 = n^2 2^{2n-2}/t,
\]

proving that \( t \geq n^2 \). This shows that the binary search protocol is the best one can do.

Rectangle Covers

Given that rectangles play such a crucial role in the communication complexity of protocols, it is worth studying alternative ways
to use them to measure the complexity of functions. Here we investigate what one can say if we count the number of monochromatic rectangles needed to cover all of the inputs.

**Definition 1.30.** We say that a boolean function has a 1-cover of size \(C\) if there are \(C\) monochromatic rectangles whose union is all of the inputs that evaluate to 1. We say that the function has a 0-cover of size \(C\) if there are \(C\) monochromatic rectangles whose union is all of the inputs that evaluate to 0.

By Theorem 1.6, every function that admits a protocol with communication \(c\) also admits a 1-cover of size at most \(2^c\) and a 0-cover of size at most \(2^c\). Conversely, Theorem 1.9 shows that small covers can be used to give small communication.

Can the logarithm of the cover number be significantly different from the communication complexity? Consider the disjointness function, defined in (1.3). For \(i = 1, 2, \ldots, n\), define the rectangle \(R_i = \{(X, Y) : i \in X, i \in Y\}\). Then we see that \(R_1, R_2, \ldots, R_n\) form a 0-cover for disjointness. So there is a 0-cover of size \(n\), yet (Theorem 1.28) the communication complexity of disjointness is \(n + 1\). In fact, by the proof of Theorem 1.9, this must mean that any 1-cover for disjointness must have at least \(2^{\Omega(\sqrt{n})}\) rectangles.

Another interesting example is the \(k\)-disjointness function. Here Alice and Bob are given sets \(X, Y \subseteq [n]\), each of size \(k\). We shall see in Chapter 2 that the communication complexity of \(k\)-disjointness is at least \(\log\binom{n}{k} \approx k\log(n/k)\). As above, there is a 0-cover of \(k\)-disjointness using \(n\) rectangles.

**Claim 1.31.** \(k\)-disjointness has a 1-cover of size \(2^{2k} \cdot \ln\binom{n}{k}\).

We prove Claim 1.31 using the probabilistic method. Sample a random 0-rectangle by picking a set \(S \subseteq [n]\) uniformly at random, and using the rectangle \(R = \{(X, Y) : X \subseteq S, Y \subseteq [n] - S\}\). Namely, the set of all inputs \(X, Y\) where \(X\) is contained in \(S\), and \(Y\) is contained in the complement of \(S\). Now sample \(t = 2^{2k} \cdot \ln\binom{n}{k}\) such rectangles independently. The probability that a particular disjoint pair \((X, Y)\) is included in any single rectangle is \(2^{-2k}\). So the probability that the
pair is excluded from all the rectangles is

\[(1 - 2^{-2k})^t < e^{-2^{-2k}t} \leq \left(\frac{n}{k}\right)^{-2},\]

by the choice of \(t\). Since the number of disjoint pairs \((X,Y)\) is at most \(\binom{n}{k}^2\), this means that the probability that any disjoint pair is excluded by the \(t\) rectangles is less than 1. So there must be \(t\) rectangles that cover all the 1 inputs.

Setting \(k = \log n\), we have found 1-cover with \(t = O(n^2 \log^2 n)\) rectangles. This means that all the entries of the matrix can be covered with \(2^{O(\log n)}\) monochromatic rectangles. However, we shall see in Chapter 2 that the communication complexity of \(k\)-disjointness is exactly \(\log \left(\frac{n}{k}\right) = \Omega(\log^2 n)\). This example shows that Theorem 1.9 is tight, at least when it comes to covers.

**Direct-sums in Communication Complexity**

**The direct-sum question** is about the complexity of solving several copies of a given problem—if a function requires \(c\) bits of communication, how much communication is required to compute \(k\) copies of the function?

Given a function \(g : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}\), we define

\[g^k : (\{0,1\}^n)^k \times (\{0,1\}^n)^k \rightarrow \{0,1\}^k\]

by

\[g^k((x_1, \ldots, x_k), (y_1, \ldots, y_k)) = (g(x_1, y_1), g(x_2, y_2), \ldots, g(x_k, y_k)).\]

If the communication complexity of \(g\) is \(c\), then the communication complexity of \(g^k\) is at most \(kc\). Can it be lower? Nontrivial examples are known that show that one can find cheaper protocols for solving many copies of certain kinds of tasks, though no such examples are known for computing functions.

Let us explore one such example. Suppose Alice is given a set \(S \subseteq [n]\) of size \(n/2\), with \(n\) even. Bob has no input. The goal of the players is to output an element of \(S\). Alice can send Bob the minimum element of her set. This can be done with communication \(\lceil \log(n/2 + 1) \rceil\), since the elements \(n/2 + 2, \ldots, n\) can never be the minimum of \(S\). Moreover, \(\lceil \log(n/2 + 1) \rceil\) bits are necessary. Indeed, if fewer bits are sent, then the set of elements \(P\) that Bob could potentially output is of size at most \(n/2\), and so the protocol would fail if Alice is given the complement of \(P\) as input.

On the other hand, we show that the parties can solve \(k\) copies of this problem with \(k + \log(nk)\) bits of communication, while the naive
The protocol would take \( k \log(n/2 + 1) \) bits of communication. The key claim is:

**Claim 1.32.** There is a set \( Q \subseteq [n]^k \) of size \( nk2^k \) with the property that for any \( S_1, S_2, \ldots, S_k \), each of size \( n/2 \), there is an element \( q \in Q \) such that \( q_i \in S_i \) for every \( i = 1, 2, \ldots, k \).

This claim gives the protocol—Alice simply sends Bob the name of the element of \( Q \) with the required property.

**Proof.** To find such a set \( Q \), we pick \( |Q| \) elements from \( [n]^k \) by sampling each element uniformly at random. For any fixed \( S_1, \ldots, S_k \), the property that \( Q \) misses this tuple is

\[
(1 - (1/2)^k)|Q| \leq e^{- (1/2)^k |Q|}.
\]

Setting \( |Q| = nk2^k \), the probability that \( Q \) does not have an element that works for some tuple is at most

\[
2^{nk} e^{- (1/2)^k |Q|} \leq 2^{nk} e^{- nk} < 1,
\]

so such a \( Q \) does exist. \( \square \)

Nevertheless, we shall use the concept of rectangle covers to prove \(^9\):

**Theorem 1.33.** If \( g \) requires \( c \) bits of communication, then \( g^k \) requires at least \( k(\sqrt{c} - \log n - 1) \) bits of communication.

In fact, one can show that even computing the two bits \( \bigwedge_{i=1}^k g(x_i, y_i) \), and \( \bigvee_{i=1}^k g(x_i, y_i) \) requires \( k(\sqrt{c} - \log n - 1) \) bits of communication.\(^10\)

The heart of the proof is the following lemma:

**Lemma 1.34.** If \( g^k \) can be computed with \( \ell \) bits of communication, then the inputs to \( g \) can be covered by \( \left\lceil 2n \cdot 2^{\ell/k} \right\rceil \) monochromatic rectangles.

Theorem 1.9 and Lemma 1.34 imply that \( g \) has a protocol with communication \((\ell/k + \log n + 1)^2\). Thus,

\[
c \leq (\ell/k + \log n + 1)^2
\]

\[
\Rightarrow \ell \geq k(\sqrt{c} - \log n - 1),
\]

as required.

Now we turn to proving Lemma 1.34. We find rectangles that cover the inputs to \( g \) iteratively. Let \( S \subseteq \{0, 1\}^n \times \{0, 1\}^n \) denote the set of inputs to \( g \) that have not yet been covered by one of the monochromatic rectangles we have already found. Initially, \( S \) is the set of all inputs. We claim:

**Claim 1.35.** There is a rectangle that is monochromatic under \( g \) and covers at least \( 2^{-\ell/k} |S| \) of the inputs from \( S \).
A permutation matrix is a 0/1 matrix where every row contains exactly one 1 and every column of \( A \) contains exactly one 1.

**Proof.** Since \( g^k \) can be computed with \( \ell \) bits of communication, by Theorem 1.6, the set \( S^k \) can be covered by \( 2^\ell \) monochromatic rectangles. So there must be some monochromatic rectangle \( R \) that covers at least \( 2^{-\ell}|S|^k \) of these inputs. For each \( i \), define

\[
R_i = \{(x, y) \in \{0, 1\}^n \times \{0, 1\}^n : \exists (a, b) \in R, a_i = x, b_i = y\},
\]

which is a rectangle, since \( R \) is a rectangle. \( R_i \) is simply the projection of the rectangle \( R \) to the \( i \)’th coordinate. Moreover, since this rectangle is monochromatic under \( g^k \), it must be monochromatic under \( g \).

Since

\[
\prod_{i=1}^{k} |R_i \cap S| \geq |R \cap S^k| \geq 2^{-\ell}|S|^k.
\]

there must be some \( i \) for which \( |R_i \cap S| \geq 2^{-\ell/k}|S| \).

We repeatedly pick rectangles using Claim 1.35 until all of the inputs to \( g \) are covered. After \( \lceil 2n2^{\ell/k} \rceil \) steps, the number of uncovered inputs is at most

\[
2^{2n} \cdot (1 - 2^{-\ell/k})2^{2^{\ell/k}} \leq 2^{2n}e^{-2^{-\ell/k}2^{2^{\ell/k}}} = 2^{2n}e^{-2n} < 1.
\]

since \( 1 - x \leq e^{-x} \) for all \( x \).

**Exercise 1.1**

Given a function \( f : \{0, 1\}^{2n} \to \{0, 1\} \), and a subset \( S \subseteq [2n] \), let \( C(f, S) \) be the communication complexity for computing \( f \), when Alice is given the bits of the input that correspond to \( S \), and Bob is given the bits that correspond to the complement of \( f \). In the introduction to this Chapter, we saw that the area of a chip computing \( f \) can be related to \( C(f, S) \), for some set \( S \) containing close to half of the inputs.

1. Use a counting argument to show that there is a universal constant \( \epsilon \) such that for most functions \( f \), \( C(f, S) \geq \epsilon n \), for every set \( S \) with \( 2n/3 \leq |S| \leq 4n/3 \).

2. Give an explicit example of a function \( f : \{0, 1\}^{2n} \to \{0, 1\} \) for which \( C(f, S) \geq \Omega(n) \) for all sets \( S \) with \( 2n/3 \leq |S| \leq 4n/3 \).

**HINT:** Consider functions of the type \( f(A, x) = x^TAx \), where \( A \) is a \( 2n \times 2n \) boolean permutation matrix. Show that for every choice of \( S \), there is a choice of \( A \) for which computing \( f \) corresponds to computing the two party inner product function on a linear number of inputs, that we have discussed in the chapter. Moreover, argue that \( A \) can be encoded with a linear number of bits.

**Exercise 1.2**
Show that if \( g : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\} \) is such that \( g^{-1}(1) \) can be partitioned into \( 2^c \) rectangles, then \( g \) has communication complexity at most \( O(c^2) \).

**Exercise 1.3**

Prove Theorem 1.9 by using the protocol for the cliques and independent sets problem discussed at the beginning of the chapter. *Hint: define a graph where the vertices correspond to (some of the) rectangles in the rectangle cover.*

**Exercise 1.4**

Suppose Alice and Bob each get a subset of size \( k \) of \([n]\), and want to know whether these sets intersect or not. Show that at least \( \log(\lfloor n/k \rfloor) \) bits are required.

**Exercise 1.5**

Suppose Alice gets a string \( x \in \{0, 1\}^n \) which has more 0’s than 1’s, and Bob gets a string \( y \in \{0, 1\}^n \) that has more 1’s than 0’s. They wish to communicate to find a coordinate \( i \) where \( x_i \neq y_i \). Show that at least \( 2 \log n \) bits of communication are required.

**Exercise 1.6**

Show that almost all functions \( f : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\} \) require communication \( \Omega(n) \).

**Exercise 1.7**

Let \( X \) and \( Y \) be families of subsets of \([n]\). Assume for all \( x \in X \) and \( y \in Y \) the intersection of \( x \) and \( y \) contains at most 1 element, that is, \( |x \cap y| \leq 1 \). Define the communication problem as follows. Alice receives \( x \in X \), Bob receives \( y \in Y \), and they wish to evaluate the function \( f : X \times Y \rightarrow \{0, 1\} \) defined as \( f(x, y) = |x \cap y| \). Show the deterministic complexity of \( f \) is \( O(\log^2(n)) \).

**Exercise 1.8**

Recall that for a simple undirected graph \( G \), the chromatic number \( \chi(G) \) is the minimum number of colors needed to color the vertices of \( G \) so that no two adjacent vertices have the same color. Show that \( \log \chi(G) \) is at most the deterministic communication complexity of \( G \)'s adjacency matrix.

**Exercise 1.9**

Alice and Bob receive inputs \( x, y \in \mathbb{Z}^n \), where \( x_i, y_i \geq 0 \) for all \( i \).
They want to compute an index \( m \in [n] \) minimizing
\[
\left| \sum_{j<i} x_j + y_j - \sum_{j>i} x_j + y_j \right|.
\]

Exhibit a deterministic protocol with \( O(\log n) \) bits of communication for finding \( m \) as above. Show no protocol can do asymptotically better. What happens when the union and median are as sets?

**Exercise 1.10**

Show that for every \( 0 \leq \alpha < 1 \), any deterministic protocol for estimating the Hamming distance between two strings \( x, y \in \{0, 1\}^n \) up to an additive error \( \alpha n \) must have length at least \( \Omega(n) \).

**Exercise 1.11**

Consider the partial function \( f : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\} \), where the input to each party is interpreted as two \( n/2 \) bit strings, defined by
\[
f(x, x', y, y') = \begin{cases} 
1 & \text{if } x = y \text{ and } x' \neq y', \\
0 & \text{if } x \neq y \text{ and } x' = y'.
\end{cases}
\]

Show that there are \( 2^n \) monochromatic rectangles under \( f \). Use fooling sets to show that the communication complexity of \( f \) is at least \( \Omega(n) \). This proves that an analogue of Theorem 1.9 does not hold for partial functions.

**Exercise 1.12**

For a boolean function \( g \), define \( g^{\land k} \) by \( g(x_1, x_2, \ldots, x_k, y_1, \ldots, y_k) = \land_{i=1}^k g(x_i, y_i) \). Show that if \( g^{\land k} \) has a 1-cover of size \( 2^\ell \), then \( g \) has a 1-cover of size \( 2^\ell/k \).

**Exercise 1.13**

Show that if \( g : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\} \) requires \( c \) bits of communication, then any protocol computing both \( \land_{i=1}^k g(x_i, y_i) \) and \( \lor_{i=1}^k g(x_i, y_i) \) requires \( k(\sqrt{c/2} - \log n - 1) \) bits of communication. **Hint:** Find a small 1-cover using the protocol for computing \( \lor_{i=1}^k g(x_i, y_i) \), and 0-cover using the protocol for computing \( \land_{i=1}^k g(x_i, y_i) \).
Representations of finite groups, spectral graph theory and the linear algebra method in combinatorics are a few examples of how matrices can be used to represent objects that are not themselves matrices.

Sometimes it is convenient to use $M_{ij} = (-1)^{g(i,j)}$ instead. If the function depends on the inputs of $k$ parties, the natural representation is by a $k$-dimensional tensor rather than a matrix.

Throughout this chapter, rank is over the reals, unless explicitly stated otherwise.

When we refer to the communication complexity of $M$, we mean the communication complexity of the associated function $g$.

\section{Rank}

A matrix is a powerful and versatile way to represent a mathematical object. Once an object is encoded as a matrix, the many tools of linear algebra can be used to infer properties of the object. This broad approach has a long history in mathematics, and we shall use it to understand communication complexity as well.

We can represent a function $g : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ by an $m \times n$ matrix $M$, where $m = |\mathcal{X}|$ is the number of rows and $n = |\mathcal{Y}|$ is the number of columns, and the $(i, j)'$th entry of $M$ is $M_{ij} = g(i, j)$. If we interpret the inputs to the parties as unit column vectors $e_i, e_j$, we have

$$g(i, j) = e_i^\top M e_j.$$

The rank of a matrix is the maximum size of a set of linearly independent rows in the matrix. We write $\text{rank}(M)$ to denote the rank of a matrix $M$. The key question of this chapter is—How is the communication complexity of a function related to the rank of the corresponding matrix?

\subsection*{Properties of Rank}

One reason why the rank of a matrix is such a useful concept is that it has many equivalent interpretations:

**Fact 2.1.** For an $m \times n$ matrix $M$, the following are equivalent:

- $\text{rank}(M) = r$.

- $r$ is the smallest number such that $M$ can be expressed as $M = AB$, where $A$ is an $m \times r$ matrix, and $B$ is an $r \times n$ matrix.

- $r$ is the smallest number such that $M$ can be expressed as the sum of $r$ matrices of rank 1.

- $r$ is the largest number such that $M$ has $r$ linearly independent columns.

Representations of finite groups, spectral graph theory and the linear algebra method in combinatorics are a few examples of how matrices can be used to represent objects that are not themselves matrices.
The rank of a matrix is quite robust to many kinds of operations. Scaling the entries of a matrix by the same non-zero quantity, or reordering the rows, or reordering the columns does not change the rank.

The second characterization of rank discussed above already seems suggestive of communication complexity—Alice and Bob can use a factorization $M = AB$, where $A$ is an $m \times r$ matrix and $B$ is an $r \times n$, to get a protocol for computing $g$. To compute $g(i, j) = e_i^\top M e_j = e_i^\top A B e_j$, Alice can send Bob $e_i^\top A$, and then Bob can multiply this vector with $B e_j$. This involves transmitting a vector of at most $r$ numbers, so it seems like we have shown that the communication complexity is at most $r$. However, each of the numbers in the vector may require lots of bits to encode, so the communication of this protocol may be large. To correctly relate the rank of the matrix to its communication complexity, we need to make a few more observations about the rank. The rank has some very nice properties that make it easy to work with:

**Fact 2.2.** If a matrix $A$ is obtained from $B$ by permuting rows or columns then $\text{rank}(A) = \text{rank}(B)$.

$A$ is called a submatrix of $B$ if $A$ can be obtained by repeatedly deleting rows and columns of $B$. In other words, it is the matrix specified by a subset of the columns of $B$ and a subset of the rows of $B$.

**Fact 2.3.** If $A$ is a submatrix of $B$ then $\text{rank}(A) \leq \text{rank}(B)$.

**Fact 2.4.** $\text{rank} \left( \begin{bmatrix} A & C \\ 0 & B \end{bmatrix} \right) \geq \text{rank}(A) + \text{rank}(B)$.

**Fact 2.5.** $|\text{rank}(A) - \text{rank}(B)| \leq \text{rank}(A + B) \leq \text{rank}(A) + \text{rank}(B)$.

A consequence of Fact 2.5 is that the constants used to represent a function $g(i, j)$ do not change the rank in a big way. For example, if $M$ is a matrix with 0/1 entries, one can define a matrix $M'$ of the same dimensions by

$$M'_{ij} = (-1)^{M_{ij}}.$$  

This operation replaces 1’s with −1’s and 0’s with 1’s. Now observe that $M' = J - 2M$, where $J$ is the all 1’s matrix, and so

**Fact 2.6.** $|\text{rank}(M') - \text{rank}(M)| \leq \text{rank}(J) = 1$.

The tensor product of an $m \times n$ matrix $M$ and an $m' \times n'$ matrix $M'$ is the $mm' \times nn'$ matrix $T = M \otimes M'$ whose entries are indexed by tuples $(i, i'), (j, j')$ defined by

$$T_{(i,i'),(j,j')} = M_{ij} \cdot M'_{i'j'}.$$
Try to prove Fact 2.7.

For example, you can check that the matrix
\[
M = \begin{bmatrix}
1 & 1 & 0 \\
1 & 0 & 1 \\
0 & 1 & 1
\end{bmatrix}
\]
has rank 2 over \(\mathbb{F}_2\), but rank 3 over the reals.

The tensor product multiplies the rank, a fact that is very useful for proving lower bounds.

**Fact 2.7.** \(\text{rank}(M \otimes M') = \text{rank}(M) \cdot \text{rank}(M')\).

The matrices we are working with have 0/1 entries, so one can view these entries as coming from any field—for instance, we can view them as real numbers, rational numbers or elements of the finite field \(\mathbb{F}_2\). This is important, because the value of the rank may depend on the field used. In this chapter, unless explicitly stated otherwise, we adopt the convention that the matrix entries are interpreted as real numbers.

However, it will be useful to have the following lemma:

**Lemma 2.8.** If \(M\) has 0/1 entries, then the rank of \(M\) over the reals is equal to its rank over the rationals, which is at least as large as its rank over \(\mathbb{F}_2\).

**Proof.** The proof of the first equality follows from Gaussian elimination: If the rank over the rationals is \(r\), we can always apply an invertible linear transformation to the rows using rational coefficients to bring the matrix into this form:

\[
M = \begin{bmatrix}
1 & 0 & 0 & \ldots & 0 & M_{1,r+1} & \ldots & M_{1,n} \\
0 & 1 & 0 & \ldots & 0 & M_{2,r+1} & \ldots & M_{2,n} \\
0 & 0 & 1 & \ldots & 0 & M_{3,r+1} & \ldots & M_{2,n} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \ldots & 1 & M_{r,r+1} & \ldots & M_{r,n} \\
0 & 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots
\end{bmatrix}
\]

This transformation does not affect the rank over the reals, and now it is clear that the rank over the reals is exactly \(r\).

To prove that the rank over the rationals is at least as large as the rank over \(\mathbb{F}_2\), observe that if any set of rows is linearly dependent over the rationals, then we can find an integer linear dependence between them, and so get a linear dependence over \(\mathbb{F}_2\). So if \(r\) rows are linearly independent over \(\mathbb{F}_2\), they must also be linearly independent over the rationals and the reals.

One consequence of Lemma 2.8 is:

**Lemma 2.9.** A matrix with 0/1 entries of rank \(r\) has at most \(2^r\) distinct rows, and at most \(2^r\) distinct columns.

**Proof.** Since the rank over \(\mathbb{F}_2\) is also at most \(r\), every row must be expressible as the linear combination of some \(r\) rows over \(\mathbb{F}_2\). There are only \(2^r\) such linear combinations possible, so there can be at most \(2^r\) distinct rows. 

\(\square\)
Communication Complexity and Rank

The basic properties of rank that we have discussed so far already allow us to prove some relationships between rank and communication complexity:

**Theorem 2.10.** If a matrix has rank $r$, then its communication complexity is at most $r + 1$.

*Proof.* If the matrix has rank $r$, then by Lemma 2.9, it has at most $2^r$ distinct rows. So Alice need only announce which of these distinct rows corresponds to her input. This takes $r$ bits of communication. Bob can then respond with the value of the function. □

The rank of a matrix also gives a lower bound on its communication complexity, via the following lemma:

**Lemma 2.11.** If the 1’s of a 0/1 matrix $M$ can be partitioned into $2^c$ monochromatic rectangles, then its rank is at most $2^c$.

*Proof.* For every rectangle $R = A \times B$, let $M_R$ denote the 0/1 matrix whose $(i, j)$ entry is 1 iff $(i, j) \in R$. We have $\text{rank}(M_R) = 1$. Moreover, $M$ can be expressed as the sum of $2^c$ such matrices. By Fact 2.1, this means $\text{rank}(M) \leq 2^c$. □

Since every function with low communication gives rise to a partition into monochromatic rectangles (Theorem 1.6), we immediately get:

**Theorem 2.12.** If a matrix has rank $r > 0$, then its communication complexity is at least $\log r$.

Theorem 2.12 allows us to prove lower bounds on the communication complexity of many of the examples we have already considered. Let us revisit some of them.

**Equality** We start with the equality function, defined in (1.1). The matrix of the equality function is just the identity matrix. The rank of the matrix is $2^n$, proving that the communication complexity of equality is $n + 1$.

**Greater-than** Consider the greater than function, defined in (1.4). The matrix of this function is the upper-triangular matrix which is 1 above the diagonal and 0 on all other points. Once again the matrix has full rank. This proves that the communication complexity is more than $\log n$. 

- Theorem 2.10 is far from the last word on the subject. By the end of this chapter, we will prove that the communication is bounded from above by a quantity closer to $\sqrt{r}$.
- Can you prove a lemma similar to Lemma 2.11, when the matrix has $+1/-1$ entries?
Disjointness Consider the disjointness function, defined in (1.3). Let \( D_n \) be a 0/1 matrix that represents disjointness. Let us order the rows and columns of the matrix in lexicographic order so that the last rows/columns correspond to sets that contain \( n \). We see that \( D_n \) can be expressed as:

\[
D_n = \begin{bmatrix}
D_{n-1} & D_{n-1}
\end{bmatrix}
\begin{bmatrix}
D_{n-1} & 0
\end{bmatrix}
\]

In other words \( D_n = D_1 \otimes D_{n-1} \), and so \( \text{rank}(D_n) = 2 \cdot \text{rank}(D_{n-1}) \) by Fact 2.7. We conclude that \( \text{rank}(D_n) = 2^n \), proving that the communication complexity of disjointness is \( n + 1 \).

\( k \)-disjointness Consider the disjointness function restricted to sets of size at most \( k \). In this case, the matrix is a \( \sum_{i=0}^{k} \binom{n}{i} \times \sum_{i=0}^{k} \binom{n}{i} \) matrix. Let us write \( D_{n,k} \) to represent the matrix for this problem. We shall use polynomials to prove that this matrix too has full rank.

For two sets \( X, Y \subseteq [n] \) of size at most \( k \), define the monomial \( m_X(z_1, \ldots, z_n) = \prod_{i \in X} z_i \), and the string \( z_Y \in \{0, 1\}^n \) such that \( (z_Y)_i = 0 \) if and only if \( i \in Y \). This ensures that \( \text{Disj}(X, Y) = m_X(z_Y) \). Any non-zero linear combination of the rows corresponds to a linear combination of the monomials we have defined, and so gives a non-zero polynomial \( f \). We show that for any such polynomial \( f \), there is a set \( Y \) of size at most \( k \) so that \( f(z_Y) \neq 0 \), so \( f \) is non-zero. This proves that the rank of the matrix is full.

To show this, let \( X \) be a set that corresponds to a monomial of maximum degree in \( f \). Let us restrict the values of all variables.
outside $X$ to be equal to 1. This turns $f$ into a non-zero polynomial that only depends on the variables corresponding to $X$. In this polynomial, let $X'$ denote the set of variables in a minimal monomial that has a non-zero coefficient. Consider the assignment $z_Y$ for $Y = X - X'$. Now, $f(z_Y)$ is equal to the coefficient of this minimal monomial, which is non-zero.

**Inner-product** Our final example is the inner product function $IP : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}$ defined by

$$IP(x,y) = \sum_{i=1}^{n} x_i y_i \mod 2. \quad (2.1)$$

Again, it will be helpful to use Fact 2.6. If $P_n$ represents the matrix of $IP$ after sorting the rows and columns lexicographically, and replacing 1 with $-1$ and 0 with $-1$, we see that

$$P_n = \begin{bmatrix} P_{n-1} & P_{n-1} \\ P_{n-1} & -P_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes P_{n-1},$$

and so by Fact 2.7, $\text{rank}(P_n) = 2 \cdot \text{rank}(P_{n-1})$. This proves that $\text{rank}(P_n) = 2^n$, and so by Fact 2.6 the communication complexity of $IP$ is at least $n$.

**Upper bounds on Communication From Rank**

Lovász and Saks conjectured$^1$ that Theorem 2.12 is closer to the truth than Theorem 2.10:

**Conjecture 2.13.** There is a constant $\alpha$ such that the communication complexity of a non-constant matrix $M$ is at most $\log^\alpha \text{rank}(M)$.

This is the famous log-rank conjecture. We do know of examples that requires $\alpha \geq 2$ in such an inequality$^2$, so we cannot expect the communication complexity of a matrix to be proportional to the logarithm of its rank.

Our main goal in this section is to prove the following theorem$^3$:

**Theorem 2.14.** If the rank of a matrix is $r > 1$, then its communication complexity is at most $O(\sqrt{r} \log^2 r)$.

To prove Theorem 2.14, we appeal to a beautiful theorem from convex geometry called John’s theorem$^4$. Informally, John’s theorem says that if a convex set is sufficiently round, then it cannot be too long. A little more precisely—if the largest ellipsoid contained in a symmetric convex body in $r$ dimensions is the unit sphere, then every element of the body has length at most $\sqrt{r}$. We use John’s theorem to prove the following lemma$^5$, which applies to any low rank matrix with 0/1 entries.

Can you come up with a similar proof that shows that the matrix that computes disjointness on sets of size exactly $k$ also has full rank?
Lemma 2.15. Any \( m \times n \) matrix that has 0/1 entries and rank \( r \geq 0 \) must contain a monochromatic submatrix of size at least \( mn \cdot 2^{-20\sqrt{r\log r}} \).

Before proving the lemma, let us see\(^6\) how to use it to get a protocol (see Figure 2.2 for an outline).

Proof of Theorem 2.14. Assume \( r > 9 \), since otherwise the proof is complete by Theorem 2.10. Let \( R \) be the rectangle promised by Lemma 2.15. Rearranging the rows and columns, we can write the matrix as:

\[
M = \begin{bmatrix} R & A \\ B & C \end{bmatrix}.
\]

Since the rank of \( R \) is at most 1, Fact 2.5 and Fact 2.4 can be used to show:

\[
\text{rank} \left( \begin{bmatrix} R \\ B \end{bmatrix} \right) + \text{rank} \left( \begin{bmatrix} R & A \end{bmatrix} \right) \leq \text{rank} \left( \begin{bmatrix} R & A \\ B & C \end{bmatrix} \right) + 3 \quad (2.2)
\]

This is because:

\[
\text{rank} \left( \begin{bmatrix} R \\ B \end{bmatrix} \right) + \text{rank} \left( \begin{bmatrix} R & A \end{bmatrix} \right) \\
\leq \text{rank}(A) + \text{rank}(B) + 2 \\
\leq \text{rank} \left( \begin{bmatrix} 0 & A \\ B & C \end{bmatrix} \right) + 2 \\
\leq \text{rank} \left( \begin{bmatrix} R & A \\ B & C \end{bmatrix} \right) + \text{rank} \left( \begin{bmatrix} -R & 0 \\ 0 & 0 \end{bmatrix} \right) + 2 \\
\leq \text{rank} \left( \begin{bmatrix} R & A \\ B & C \end{bmatrix} \right) + 3.
\]

Now if

\[
\text{rank} \left( \begin{bmatrix} R \\ B \end{bmatrix} \right) \leq \text{rank} \left( \begin{bmatrix} R & A \end{bmatrix} \right),
\]

then Bob can tell Alice if his input is consistent with \( R \). If it is consistent, then the players can make a lot of progress by setting

\[
M = \begin{bmatrix} R \\ B \end{bmatrix},
\]

and continuing the protocol on the matrix \( M \). They have reduced\(^7\) the rank of the matrix by at least a factor of \( \frac{2}{3} \). If Bob’s input is not consistent with \( R \), then the players set

\[
M = \begin{bmatrix} A \\ C \end{bmatrix},
\]

Input: Alice knows \( i \), Bob knows \( j \).
Output: \( M_{i,j} \).

\begin{algorithm}
\begin{algorithmic}
\While{\text{rank}(M) > 9}
\State Find a monochromatic rectangle \( R \) as promised by Lemma 2.15;
\State Write \( M = \begin{bmatrix} R & A \\ B & C \end{bmatrix} \);
\If{\text{rank}\left( \begin{bmatrix} R \\ B \end{bmatrix} \right) > \text{rank}\left( \begin{bmatrix} R & A \end{bmatrix} \right)}
\If{i is consistent with \( R \)}
\State Both parties replace \( M \) with \( \begin{bmatrix} R & A \end{bmatrix} \);
\Else
\State Both parties replace \( M \) with \( \begin{bmatrix} B & C \end{bmatrix} \);
\EndIf
\EndIf
\Else
\If{j is consistent with \( R \)}
\State Both parties replace \( M \) with \( \begin{bmatrix} R \\ B \end{bmatrix} \);
\Else
\State Both parties replace \( M \) with \( \begin{bmatrix} A \\ C \end{bmatrix} \);
\EndIf
\EndIf
\State The parties exchange at most 10 bits to compute \( M_{i,j} \), using Theorem 2.10;
\EndWhile
\end{algorithmic}
\end{algorithm}

\(^6\) Lovász, 1990; and Nisan and Wigderson, 1995

\(^7\) \( (t+3)/2 \leq 2t/3 \), when \( t \geq 9 \).
and recursively continue the protocol on \( M \). In this case, they reduced the size of the matrix by at least a factor of \( 1 - 2^{-20\sqrt{t\log r}} \).

The length of the protocol described above may be large. However, we shall prove that it does have a small number of leaves. By Lemma 2.9, we can assume that the matrix \( M \) has at most \( 2r \) rows and columns. The number of transmissions in this protocol where Alice or Bob says that their input is inconsistent with the monochromatic rectangle is at most \( 2r \ln 2 \cdot 2^{20\sqrt{t\log r}} \), since after that many transmissions, the number of entries in the matrix have been reduced to

\[
2^{2r} (1 - 2^{-20\sqrt{t \log r}}) 2^{2r} 2^{20\sqrt{t \log r}} < 2^{2r} e^{-2 - 20\sqrt{t \log r}} 2r \ln 2 \cdot 2^{20\sqrt{t \log r}} = 1.
\]

The number of transmissions where Alice or Bob announces that their input is consistent with the rectangle is at most \( O(\log r) \), since after that many transmissions, the rank of the matrix is reduced to less than 9. Thus, the number of leaves in this protocol is at most

\[
O(\log r) \leq 2^{O(\sqrt{t \log^2 r})}.
\]

Finally, by Theorem 1.7, we can balance the protocol tree to obtain a protocol computing \( M \) with length \( O(\sqrt{t \log^2 r}) \).

It only remains to prove Lemma 2.15. It is no loss of generality to assume that the matrix has more 0’s than 1’s—if this is not the case, we can always work with the matrix \( J - M \).

The lemma is proved in two steps. First we show that \( M \) must contain a large rectangle that is almost monochromatic:

**Claim 2.16.** If at least half of the entries in \( M \) are 0’s, then there is a submatrix \( T \) of \( M \) of size at least \( mn2^{-16\sqrt{t \log r}} \) such that the fraction of 1’s in \( T \) is at most \( 1/r^3 \).

The second claim shows how to find a large zero rectangle in a matrix with low rank and few ones.

**Claim 2.17.** If \( T \) is 0/1 matrix of rank \( r \) so that at most \( 1/r^3 \) of the entries are 1’s, then there is a 0-submatrix consisting of at least half of the rows and half of the columns of \( T \).

**Proof.** Call a row of \( T \) good if the fraction of 1’s in it is at most \( 2/r^3 \). At least half the rows of \( T \) must be good, or else \( T \) would have more than \( 1/r^3 \) fraction of 1’s overall. Let \( T' \) be the submatrix obtained by restricting \( T \) to the good rows. Since \( \text{rank}(T') \leq r \), it has \( r \) rows \( A_1, \ldots, A_r \) that span all the other rows of \( T' \). Each row \( A_i \) has at most \( 2/r^3 \) fraction of 1’s, and at most \( r \cdot 2/r^3 \leq 1/2 \) fraction of the columns can contain a 1 in one of these \( r \) rows. Let \( T'' \) be the submatrix obtained by restricting \( T' \) to the columns that do not have a 1 in the rows \( A_1, \ldots, A_r \). Then \( T'' \) must be 0, since every row of \( T' \) is a linear combination of \( A_1, \ldots, A_r \). □

---

*If we replace \( M \) with \( J - M \), where \( J \) is the all 1’s matrix, this can increase the rank by at most 1, but now the role of 0’s and 1’s has been reversed.*

*Gavinsky and Lovett, 2014*
It only remains to prove Claim 2.16, which relies on some interesting results from convex geometry. A set $K \subseteq \mathbb{R}^r$ is called convex if whenever $x, y \in K$, then all the points on the line segment between $x$ and $y$ are also in $K$. The set is called symmetric if whenever $x$ is in $K$, $-x$ is also in $K$. An ellipsoid centered at 0 is a set of the form:

$$E = \left\{ x \in \mathbb{R}^r : \sum_{i=1}^r \langle x, u_i \rangle^2 / a_i^2 \leq 1 \right\},$$

where $u_1, \ldots, u_r$ is an orthonormal basis of $\mathbb{R}^r$, and $a_1, \ldots, a_r$ are non-zero numbers.

**Theorem 2.18** (John’s Theorem). Let $K \subseteq \mathbb{R}^r$ be a symmetric convex body such that the unit ball is the most voluminous of all ellipsoids contained in $K$. Then every element of $K$ has Euclidean length at most $\sqrt{r}$.

John’s theorem can be used to show that any matrix with 0/1 entries has a useful factorization:

**Lemma 2.19.** Any boolean matrix $M$ of rank $r$ can be expressed as $M = AB$, where $A$ is an $m \times r$ matrix whose rows are vectors of length at most $\sqrt{r}$, and $B$ is an $r \times n$ matrix whose columns are vectors of length at most 1.

**Proof.** Since the matrix has rank $r$, we know that $M$ can be expressed as $M = A'B'$, where $A'$ is an $m \times r$ matrix, and $B'$ is an $r \times n$ matrix. Moreover, the rows of $A'$ must all be linearly independent, and columns of $B'$ must all be linearly independent, or the rank of $M$ would be less than $r$. The matrices $A', B'$ do not necessarily satisfy the length constraints. Let $v'_1, \ldots, v'_m$ be the rows of $A'$, and $w'_1, \ldots, w'_n$ be the columns of $B'$.

Let $K'$ be the convex hull of $\{\pm v'_1, \ldots, \pm v'_m\}$. Our first goal is to modify these vectors so that the ellipsoid of maximum volume in $K'$ is the unit ball. Although this may not be the case initially, we show how to transform $A', B'$ to $A, B$ and consequently $K'$ to $K$ so that the ellipsoid of maximum volume contained in $K$ is the unit ball.

Suppose

$$E' = \left\{ x \in \mathbb{R}^r : \sum_{i=1}^r \langle x, u_i \rangle^2 / a_i^2 \leq 1 \right\}$$

is the ellipsoid of maximum volume contained in $K'$. We apply a linear transformation to the space of rows of $A'$ to get the rows of $A$. Each row $v'_i = \sum_{j=1}^r \beta_i u_{ij}$ in $A'$ is replaced by the row $v_i = \sum_{j=1}^r \alpha_i \beta_i u_{ij}$. This has the desired effect on the ellipsoid of the maximal volume, the ellipsoid of maximum volume in the convex hull $K$ of $\{\pm v_1, \ldots, \pm v_m\}$ is now

$$E = \left\{ x \in \mathbb{R}^r : \sum_{i=1}^r \langle x, u_i \rangle^2 \leq 1 \right\}.$$
which is the unit ball. To compensate for scaling the rows of $A'$, we also need to scale the columns of $B'$. Each column $w'_i = \sum_{j=1}^r \gamma_i u_i$ in $B'$ is replaced by the column $w_i = \sum_{j=1}^r (1/\alpha) \gamma_i u_i$. This ensures that:

$$\langle v_i, w'_j \rangle = \sum_{k=1}^r \alpha_i \beta_i (1/\alpha) \gamma_i u_i = \sum_{k=1}^r \beta_i \gamma_i u_i = \langle v'_i, w'_j \rangle,$$

and so $M = A'B' = AB$. By John’s theorem, every vector $v_i$ must have length at most $\sqrt{r}$.

It only remains to argue that vectors $w_1, \ldots, w_n$ are of length at most 1. This is where we use the fact that the matrix has entries of small magnitude. Consider any $w_i$, and the unit vector in the same direction $e_i = w_i / \|w_i\|$. The length of $w_i$ can be expressed as $\langle w_i, e_i \rangle$. Since $e_i$ is in the unit ball, it is also contained in $K$, so $e_i = \sum_j \mu_j v_j + \sum_j \kappa_j (-v_j)$ is a convex combination of the $v_j$'s.

Thus

$$\langle w_i, e_i \rangle = \left( \sum_j \mu_j v_j \right) + \left( \sum_j \kappa_j (-v_j) \right) = \sum_j \mu_j \langle v_i, v_j \rangle + \sum_j \kappa_j \langle w_i, -v_j \rangle \leq \sum_j \mu_j + \sum_j \kappa_j = 1,$$

where the inequality follows from the fact that $M$ has 0/1 entries. \qed

Now let us use Lemma 2.19 to complete the proof.

Proof of Claim 2.16. Let $A, B$ be the two matrices guaranteed by Lemma 2.19. Let $v_1, \ldots, v_m$ be the rows of $A$ and $w_1, \ldots, w_n$ be the columns of $B$. Let $\theta_{ij} = \arccos \left( \frac{\langle v_i, w_j \rangle}{\|v_i\| \|w_j\|} \right)$ be the angle between the unit vectors in the directions of $v_i$ and $w_j$. We claim that

$$\theta_{ij} \begin{cases} \frac{\pi}{2} & \text{if } M_{ij} = 0, \\ \leq \frac{\pi}{2} - \frac{2\pi}{\sqrt{r}} & \text{if } M_{ij} = 1. \end{cases}$$

Figure 2.3: One can always apply a linear transformation so that the unit ball is the inscribed ellipsoid of maximum volume inside the given convex body.

Figure 2.4: If some $w_i$ has length bigger than 1, then there must be a vector $v_j$ such that $\langle v_j, w_i \rangle > 1$. This is impossible.
When \(v_i, w_j\) are orthogonal, the angle is \(\pi/2\). When the inner product is closer to 1, we use the fact that \(\arccos(a) \leq \pi/2 - 2\pi a/7\), which implies that the angle is at most \(\arccos(\frac{1}{\sqrt{7}}) \leq \pi/2 - \frac{2\pi}{\sqrt{7}}\).

The existence of the rectangle we seek is proved via the probabilistic method. Consider the following experiment. Sample \(t\) vectors \(z_1, \ldots, z_t \in \mathbb{R}^r\) of length 1 uniformly and independently at random, and use them to define the rectangle

\[
R = \{(i, j) : \forall k \in [t], \langle v_i, z_k \rangle > 0, \langle w_j, z_k \rangle < 0\}.
\]

For fixed \((i, j)\) and \(k\), the probability that \(\langle v_i, z_k \rangle > 0\) and \(\langle w_j, z_k \rangle < 0\) is exactly \(\frac{1}{4} - \frac{\pi/2 - \theta_{ij}}{2\pi}\). So for a fixed \((i, j)\) we get

\[
\Pr_{z_1, \ldots, z_t}[(i, j) \in R] = \left(\frac{1}{4}\right)^t \cdot \text{Pr}(M_{ij} = 0), \quad \left(\frac{1}{4} - \frac{1}{\sqrt{7}}\right)^t \cdot \text{Pr}(M_{ij} = 1).
\]

Let \(R_1\) denote the number of 1’s in \(R\) and \(R_0\) denote the number of 0’s. Set \(t = \lceil 7\sqrt{r} \log r \rceil\). By what we have just argued,

\[
\mathbb{E}[R_0] \geq \frac{mn}{2 \cdot 4^t},
\]

and using the Fact that \(1 - x \leq e^{-x}\) for \(0 < x < 1\),

\[
\mathbb{E}[R_1] \leq \frac{mn}{2 \cdot 4^t} \cdot \left(1 - \frac{4}{\sqrt{7}}\right)^t \leq \frac{mn}{2 \cdot 4^t} \cdot e^{\frac{-\sqrt{7}}{\sqrt{7}}} \leq \frac{mn}{2 \cdot 4^t} \cdot r^{-4\log e}.
\]

Now let \(Q = R_0 - r^4 R_1\). By linearity of expectation, we have

\[
\mathbb{E}[Q] \geq \frac{mn}{2 \cdot 4^t} \cdot (1 - 1/r) \geq \frac{mn}{2 \cdot 4^t} \cdot 2^{-16\sqrt{r} \log r}.
\]

There must be some rectangle \(R\) realizing this value of \(Q\). Only a \(1/r^3\) fraction of such a rectangle can correspond to 1 entries of the matrix, or else \(Q\) would be negative.

\[\square\]

**Non-negative Rank**

Non-negative rank gives another way to measure the complexity of a matrix. The non-negative rank of an \(m \times n\) boolean matrix \(M\), denoted \(\text{rank}_+(M)\), is the smallest number \(r\) such that \(M = AB\), where \(A, B\) are matrices with non-negative entries, such that \(A\) is an \(m \times r\) matrix and \(B\) is an \(r \times n\) matrix. Equivalently, it is the smallest number of non-negative rank 1 matrices that sum to \(M\).

Since any such non-negative factorization \(M = AB\) would establish the rank of \(M\) is also at most \(r\), we have:

**Fact 2.20.** \(\text{rank}(M) \leq \text{rank}_+(M)\).
However, \( \text{rank}(M) \) and \( \text{rank}_+(M) \) may be quite different. For example, given a set of numbers \( X = \{x_1, \ldots, x_n\} \) of size \( n \), consider the \( n \times n \) matrix defined by \( M_{i,j} = (x_i - x_j)^2 = x_i^2 + x_j^2 - 2x_ix_j \). Since \( M \) is the sum of three rank 1 matrices, namely the matrices corresponding to \( x_i^2 \), \( x_j^2 \) and \(-2x_ix_j \), we have \( \text{rank}(M) \leq 3 \). On the other hand, we can show by induction on \( n \) that \( \text{rank}_+(M) \geq \log n \).

The proof of Theorem 2.21 is similar to that of Theorem 2.12.

The non-negative rank could give stronger lower bounds on communication complexity than the rank:

**Theorem 2.21.** If a matrix has non-negative rank \( r > 1 \), then its communication complexity is greater than \( \log r \).

When the non-negative rank is small, the matrix has polylogarithmic communication complexity\(^{10}\):

**Theorem 2.22.** The communication complexity of \( M \) is at most

\[
O(\log(\text{rank}_+(M)) \cdot \log(\text{rank}(M))) \leq O(\log^2(\text{rank}_+(M))).
\]

**Proof.** If a matrix \( M \) with 0/1 entries has small non-negative rank \( r \), we must have \( M = R_1 + \ldots + R_r \), where \( R_1, \ldots, R_r \) are non-negative rank 1 matrices. The set of non-zero entries of each matrix \( R_i \) must form a monochromatic rectangle in \( M \) with value 1. Thus, \( M \) must admit a 1-cover of size \( r \). We shall use this 1-cover to define an efficient communication protocol.

The protocol is similar to the one used to prove Theorem 1.9. If the rank of the matrix is \( r < 9 \), the parties use a constant number of bits to compute the output. If \( r > 9 \), then for every rectangle \( R \) in the 1-cover, we can write

\[
M = \begin{bmatrix} R & A \\ B & C \end{bmatrix}.
\]

As in 2.2, either

\[
\text{rank} \left( \begin{bmatrix} R \\ A \end{bmatrix} \right) \leq (\text{rank}(M) + 3)/2, \quad (2.3)
\]

or

\[
\text{rank} \left( \begin{bmatrix} R \\ B \end{bmatrix} \right) \leq (\text{rank}(M) + 3)/2. \quad (2.4)
\]

\(^{10}\)Lovász, 1990
So in each step of the protocol, if Alice sees an \(R\) that is consistent with her input and satisfying (2.3), she announces its name, or if Bob sees a rectangle \(R\) in the cover that is consistent with his input and satisfying (2.4), he announces its name.

Both parties then restrict their attention to the appropriate sub-matrix, which reduces the rank of \(M\) by a factor of \(2/3\). This can continue for at most \(O(\log \text{rank}(M))\) steps before the rank of the matrix becomes less than 9.

On the other hand, if neither party finds such an \(R\), then there is no such \(R\) that covers their input, so they can safely conclude that the entry they seek is 0.

\[\square\]

**Exercise 2.1**

Show that there is a boolean matrix of rank \(r\) with \(2^r\) distinct rows, and \(2^r\) distinct columns. Conclude that Lemma 2.9 is sharp.

**Exercise 2.2**

Let \(M\) be a 0/1 matrix with exactly \(t\) ones in each row and column. Show that you can cover the zeros of \(M\) using \(O(t(\log |X| + \log |Y|))\) monochromatic rectangles.

**Exercise 2.3**

Show that the protocol in Figure 2.2 goes through even if we weaken Lemma 2.15 to only guarantee a rectangle with rank at most \(r/8\) (instead of rank at most one, or monochromatic).

**Exercise 2.4**

For any symmetric matrix \(M \in \{0, 1\}^{n \times n}\) with ones in all diagonal entries, show that

\[2^c \geq \frac{n^2}{|M|}\]

where \(c\) is the deterministic communication complexity of \(M\), and \(|M|\) is the number of ones in \(M\).

**Exercise 2.5**

For any boolean matrix \(M\), define \(\text{rank}_2(M)\) to be the rank of \(M\) over the field with two elements \(F_2\). Exhibit an explicit family of matrices \(M \in \{0, 1\}^{n \times n}\) with the property that \(c \geq \text{rank}_2(M)/10\), where \(c\) is the deterministic communication complexity of \(M\). This falsifies the analogue of log-rank conjecture for \(\text{rank}_2\).

**Exercise 2.6**
Show that if $f$ has fooling set of size $s$ then $rk(M_f) \geq \sqrt{s}$. *Hint: tensor product.*

**Exercise 2.7**

For a real matrix $M$ and $\epsilon > 0$ define the $\epsilon$-approximate rank of $M$ to be $\text{rank}_\epsilon(M) = \min \{ \text{rank}(A) : |A_{i,j} - M_{i,j}| \leq \epsilon \text{ for all } i,j \}.$

1. Find a boolean matrix with rank $r$ and $1/3$-approximate rank at most $O(\log r)$. *Hint: the equality function.*

2. Prove the following strengthening of Theorem 2.14: The communication complexity of a matrix $M$ is at most $O(\sqrt{\text{rank}_{1/3}(M)} \log^2 \text{rank}(M)).$
3

Randomized Protocols

Access to randomness is an enabling feature in many computational processes. Randomized algorithms are often easier to understand and more elegant than their deterministic counterparts. In this chapter, we show how randomization can be used to give communication protocols that are far more efficient than the best possible deterministic protocols for many problems. We start by giving some examples of randomized protocols. Later, we give rigorous definitions for randomized protocols, and prove some basic properties.

We do not discuss any lower bounds on randomized communication complexity in this chapter. Lower bounds for randomized protocols can be found in Chapter 5 and Chapter 6.

Some Protocols

Equality Suppose Alice and Bob are given access to $n$ bit strings $x$, $y$, and want to know if these strings are the same or not (1.1). In Chapter 1 we showed that the deterministic communication complexity of this problem is $n + 1$.

There is a simple randomized protocol, where Alice and Bob use a shared random string. Alice and Bob use the shared randomness to sample a random function $h : \{0, 1\}^n \rightarrow \{0, 1\}^k$. Then, Alice sends $h(x)$ to Bob, and Bob responds with a bit indicating whether or not $h(y) = h(x)$. If $h(x) = h(y)$, they conclude that $x = y$. If $h(x) \neq h(y)$, they conclude that $x \neq y$. The number of bits communicated is $k + 1$. The probability of making an error is at most $2^{-k}$—if $x = y$ then $h(x) = h(y)$, and if $x \neq y$ then the probability that $h(x) = h(y)$ is at most $2^{-k}$.

This protocol may seem less than satisfactory, because the number of shared random bits required to sample $h$ is quite large. We can
reduce the number of random bits used if we use an error correcting code. This is a function $C : \{0,1\}^n \to [2^k]^m$ such that if $x \neq y$, then $C(x)$ and $C(y)$ differ in all but $m/2^{\Omega(k)}$ coordinates. It can be shown that if $m$ is set to $10n$, then for any $k$, most functions $C$ will be error correcting codes.

Given the code, Alice can pick a random coordinate of $C(x)$ and send it to Bob, who can then check whether this coordinate is consistent with $C(y)$. This takes $\log m + \log 2^k = O(\log n + k)$ bits of communication, and again the probability of making an error is at most $2^{-\Omega(k)}$. In this protocol, the players do not require a shared random string, since the choice of $C$ is made once and for all when the protocol is constructed, and before the inputs are seen.

**Greater-than** Suppose Alice and Bob are given numbers $x, y \in [n]$ and want to know which one is greater (1.4). We have seen that any deterministic protocol for this problem requires $\log n$ bits of communication. However, there is a randomized protocol that requires only $O(\log \log n)$ bits of communication.

Here we describe a protocol that requires only $O(\log \log n \cdot \log \log \log n)$ communication. The inputs $x, y$ can be encoded by $\ell$-bit binary strings, where $\ell = O(\log n)$. To determine whether $x \geq y$, it is enough to find the most significant bit where $x$ and $y$ differ. To find this bit, we use the randomized protocol for equality described above, along with binary search. In the first step, Alice and Bob will use the protocol for equality to exchange $k$ bits that determine whether the $n/2$ first (most significant) bits of $x$ and $y$ are the same. If they are the same, the parties continue with the last $n/2$ bits. If not, the parties discard the second half of their strings. In this way, after $O(\log n)$ steps, they find the first bit of difference in their inputs. In order to ensure that the probability of making an error in all of these $O(\log n)$ steps is small, we set $k = O(\log \log n)$. By the union bound, this guarantees that the protocol succeeds with high probability.

**$k$-Disjointness** Suppose Alice and Bob are given 2 sets $X, Y \subseteq [n]$ of size at most $k$, and want to know if these sets intersect or not. In Chapter 2, we used the rank method to argue that at least $\log \binom{n}{k} \approx k \log(n/k)$ bits of communication are required. Here we give a randomized protocol\(^1\) that requires only $O(k)$ bits of communication, which is more efficient when $k \ll n$.

Alice and Bob start by exchanging 2 bits to announce whether or not either of them has the empty set as an input. If one of them has the empty set as input, then their sets are disjoint, and the protocol terminates. If neither of their sets is empty, they use shared randomness to sample a sequence of sets $R_1, R_2, \ldots \subseteq [n]$.

Many beautiful explicit constructions of error correcting codes are also known.

---

**Exercise 3.1**

- **Input:** Alice knows $x \in \{0,1\}^n$.
- **Output:** Largest $i$ such that $x_i \neq y_i$, if such an $i$ exists.

Let $J$ be the first $|J|/2$ elements of $J$.

Both parties use shared randomness to sample a random function $h : \{0,1\}^{|J|/2} \to \{0,1\}^{2 \log \log n}$; Alice sends $h(x_J)$, which is $h$ evaluated on the bits in $J$; Bob announces whether or not $h(x_J) = h(y_J)$; if $h(x_J) = h(y_J)$ then Alice and Bob replace $J = J - J'$; else Alice and Bob replace $J = J'$.

Both parties announce $x_J, y_J$ to decide if which of $x, y$ is greater.

---

\(^1\) Hästad and Wigderson, 2007

In Chapter 6, we show that $\Omega(k)$ bits are required.
uniformly at random. Alice announces the index $i$ of the first set $R_i$ that contains her set, and Bob announces the index $j$ of the first set $R_j$ that contains his set. This can be done with at most $2(\log(i) + \log(j) + 2)$ bits of communication. Now Alice can safely replace her set with $X \cap R_j$, and Bob can replace his set with $Y \cap R_i$—if the sets were disjoint, they remain disjoint, and if they were not disjoint, they must still be intersect. They repeat the above process.

We argue that if the sets are disjoint, this process must end, at least in expectation, after $O(k)$ bits of communication. Suppose $X, Y$ are disjoint. Let us start by analyzing the expected number of bits that are communicated in the first step. We claim:

**Claim 3.1.** $\mathbb{E}[i] = 2^{|X|}, \mathbb{E}[j] = 2^{|Y|}.$

**Proof.** The probability that the first set of the sequence contains $X$ is exactly $2^{-|X|}$. In the event that it does not contain $X$, we are picking the first set that contains $X$ from the rest of the sequence. Thus:

$$\mathbb{E}[i] = 2^{-|X|} \cdot 1 + (1 - 2^{-|X|}) \cdot (\mathbb{E}[i] + 1)$$

($\Rightarrow \mathbb{E}[i] = 2^{|X|}.$

The bound on the expected value of $j$ is the same. □

So, by Jensen’s inequality applied to the log function, the expected length of the first step is at most

$$2\mathbb{E}[\log(i) + \log(j)] \leq 2(\log(\mathbb{E}[i]) + \log(\mathbb{E}[j])) = 2(|X| + |Y|).$$

(3.1)

**Claim 3.2.** If $X \cap Y = \emptyset$, the expected number of bits communicated by the protocol is at most $8|X| + 8|Y| + 4.$

**Proof.** We show that for every positive integer $L$, if $(X, Y)$ are sets in $[n]$ so that $X \cap Y = \emptyset$ and $|X| + |Y| \leq L$ then the expected

Alice can send $i$ using $2\lceil \log i \rceil$ bits by sending the binary encoding of $i$ bit by bit. In each step, she sends a bit to indicate whether or not the transmission is over, and the next bit of the encoding. In fact, there is a more efficient encoding using $\log(i) + O(\log \log(i))$ bits of communication.

**Input:** Alice knows $X \subseteq [n], \text{ Bob knows } Y \subseteq [n].$

**Output:** Whether or not $X \cap Y = \emptyset.$

**while** $|X| > 1 \text{ and } |Y| > 1 \text{ and at most } 120k + 20 \text{ bits have been communicated so far do}$

| Alice and Bob use shared randomness to sample random subsets $R_1, R_2, \ldots \subseteq [n]$; |
| Alice sends Bob the smallest $i$ such that $X \subseteq R_i$; |
| Bob sends Alice the smallest $j$ such that $Y \subseteq R_j$; |
| Alice replaces $X = X \cap R_i$; |
| Bob replaces $Y = Y \cap R_j$; |

**end**

**if** $X = \emptyset$ or $Y = \emptyset$ **then**

| Alice and Bob conclude that the sets were disjoint; |

**else**

| Alice and Bob conclude that the sets were intersecting; |

**end**

**Figure 3.4:** When $X, Y$ are disjoint, typically half of $Y$ will be eliminated by $R_i$.

**Figure 3.5:** Public-coin protocol for $k$-disjointness.
number of bits communicated is at most \( C(L) \leq 8L + 4 \). Similarly, the bound clearly holds when \( L = 0 \).

The proof proceeds by induction on \( L \). For the base case, if \( |X| + |Y| \leq L = 1 \) then one of the sets is empty and indeed at most \( 2 \leq 8L + 4 \) bits are communicated.

For the inductive step, assume \( L \geq 2 \). Equation (3.1) shows that the expected number of bits communicated in the first step of the protocol is at most \( 4 + 2|X| + 2|Y| \leq 4 + 2L \). The two new sets are \( X \cap R_i \) and \( Y \cap R_i \). By induction, the expected communication can be bounded:

\[
C(L) = \sum_{L' = 0}^{\lfloor |X| + |Y| \rfloor} \Pr[|X \cap R_i| + |Y \cap R_i| = L'] \cdot C(L') \\
\leq \frac{C(L)}{2^L} + \sum_{L' = 0}^{\lfloor |X| + |Y| \rfloor - 1} \Pr[|X \cap R_i| + |Y \cap R_i| = L'] \cdot (8L' + 4) \\
\leq \frac{C(L)}{2^L} + \mathbb{E}[8(|X \cap R_i| + |Y \cap R_i|) + 4].
\]

We have \( \mathbb{E}[|X \cap R_i|] = |X|/2 \) and \( \mathbb{E}[|Y \cap R_i|] = |Y|/2 \), so \( \mathbb{E}[|X \cap R_i| + |Y \cap R_i|] = L/2 \). Thus, we have

\[
C(L) \leq \frac{1}{1 - 2^{-L}} \cdot (4L + 4) \leq \frac{4}{3} \cdot (4L + 4) \leq 8L + 4,
\]

for \( L \geq 2 \).

\( \square \)

Claim 3.2 means that if \( X, Y \) are disjoint, the expected communication is at most \( 8|X| + 8|Y| + 4 \). By Markov’s inequality, if the sets are disjoint then the probability that the protocol communicates more than \( 10 \cdot (8|X| + 8|Y| + 4) \) bits is at most \( 1/10 \). Thus, if we run this process until \( 160k + 40 \) bits have been communicated (or it terminates because one of the sets is empty), the probability of making an error is at most \( 1/10 \).

**Gap-Hamming** Suppose Alice and Bob are given two strings \( x, y \in \{+1, -1\}^n \), and want to estimate the Hamming-distance:

\[
\Delta(x, y) = |\{ i \in [n] : x_i \neq y_i \}| = \frac{n - (x, y)}{2}.
\]

We say that a protocol \( \pi \) approximates the Hamming distance up to a parameter \( m \), if \( |\pi(x, y) - \Delta(x, y)| \leq m \). In Exercise 1.10, we studied the relationship between \( m \) and the deterministic communication complexity of \( \pi \), and showed that for any \( \alpha < 1 \), approximating the Hamming distance up to \( \alpha n \) requires communication \( \Omega(n) \). Here we show that there is a significantly better randomized protocol.
Alice and Bob use shared randomness to sample \(i_1, \ldots, i_k \in [n]\) uniformly at random, and then communicate \(2k\) bits to compute

\[
\gamma = (1/k) \cdot |\{ j \in [k] : x_{i_j} \neq y_{i_j} \}|.
\]

They output \(\gamma n\).

We now analyze the probability that this protocol makes an error, when \(\Delta(x, y) \leq n/2\). Define \(Z_1, \ldots, Z_k\) by

\[
Z_j = \begin{cases} 
1 & \text{if } x_{i_j} \neq y_{i_j}, \\
0 & \text{otherwise.}
\end{cases}
\]

The expected value of each \(Z_j\) is \(\Delta(x, y)/n\). So if \(m \leq \Delta(x, y)\), we can apply the Chernoff-Hoeffding bound to conclude:

\[
\Pr[|\pi(x, y) - \Delta(x, y)| > m] \leq e^{-\frac{2m^2}{\Delta(x, y)}} \cdot \frac{1}{2}
\]

If \(m > \Delta(x, y)\), we have

\[
\Pr[|\pi(x, y) - \Delta(x, y)| > m] \leq e^{-\frac{m^2}{\Delta(x, y)}} \cdot \frac{1}{2}
\]

Since \(m \leq n\) and \(\Delta(x, y) \leq n\), in either case this probability is at most \(e^{-\frac{m^2}{3n^2}}\). Thus, if, for example, we set \(k = 3n^2/m^2\), we obtain a protocol whose probability of making an error is at most \(1/e\).

Variants of Randomized Communication Complexity

A randomized protocol is a deterministic protocol where each party has access to a random string, in addition to the inputs to the protocol. The random string is sampled independently from the inputs, but may have an arbitrary distribution known to the players.

We say that a protocol uses public coins if all parties have access to a common shared random string. We say that the protocol uses private coins if each party privately samples an independent random string. Every private coin protocol can be simulated by a public coin protocol, and we shall soon see a partial converse: every public coin protocol can be simulated with private coins, with a small increase in the communication.

There are two established ways to measure the probability that a randomized protocol makes an error:

For \(m = n^{0.6}\) we get a protocol of length \(O(n^{0.8})\), and for \(m = \sqrt{n}/\epsilon\) the length is \(O(\epsilon^2 n)\).

One can always simulate any randomized protocol by a protocol that uses uniformly random bits as the shared randomness.

If a randomized protocol never makes an error, we can fix the randomness to obtain a deterministic protocol that is always correct.
Worst-case  We say that a randomized protocol has error \( \epsilon \) in the worst-case if the probability that the protocol makes an error is at most \( \epsilon \) on every input.

Average-case  Given a distribution on inputs \( \mu \), we say that a protocol has error \( \epsilon \) with respect to \( \mu \) if the probability that the protocol makes an error is at most \( \epsilon \) when the inputs are sampled from \( \mu \).

In both cases, the length of the protocol is always defined to be the maximum depth of all of the deterministic protocol trees that the protocol may generate.

When a protocol has error \( \epsilon < 1/2 \) in the worst case, we can run it several times and output the most common outcome that we see. This reduces the probability of making an error. If we run the protocol \( k \) times, and output the most frequent output in all of the runs, there will be an error in the output only if at least \( k/2 \) of the runs computed the wrong answer. By the Chernoff bound, the probability of error is thus reduced to at most \( 2^{-\Omega(k(1/2-\epsilon)^2)} \).

The worst-case and average-case complexity of a problem are related by via Yao’s minimax principle:

**Theorem 3.3.** The communication complexity of a function \( g \) in the worst-case with error at most \( \epsilon \) is equal to the maximum, over all distributions \( \mu \), of the average-case communication complexity of \( g \) with error at most \( \epsilon \) with respect to \( \mu \).

To prove Theorem 3.3, we appeal to a famous minimax principle due to von Neumann:\(^2\)

**Theorem 3.4.** Let \( M \) be an \( m \times n \) matrix with entries that are real numbers. Let \( A \) denote the set of \( 1 \times m \) row vectors with non-negative entries, such that \( \sum_i x_i = 1 \), and let \( B \) denote the set of \( n \times 1 \) column vectors with non-negative entries such that \( \sum_j y_j = 1 \). Then

\[
\min_{x \in A} \max_{y \in B} xMy = \max_{y \in B} \min_{x \in A} xMy.
\]

Theorem 3.4 has a very intuitive interpretation in terms of zero-sum games. The matrix \( M \) encodes the rules of the game, and \( x, y \) represent strategies for playing the game. There are two players: a row player and a column player. The row player privately chooses a row \( i \) of the matrix, and the column player privately chooses a column \( j \). The outcome of the game is determined by \( i \) and \( j \): the column player gets a payoff of \( M_{i,j} \), and the row player gets a payoff of \( -M_{i,j} \). This is why it is a zero-sum game—the sum of payoffs is zero. In such a game, the column player chooses \( j \) in order to maximize \( M_{i,j} \) and the row player chooses \( i \) in order to minimize \( M_{i,j} \).

The players are allowed to use randomness to play the game, by sampling a row or column according to a distribution of their

\[
\begin{pmatrix}
R & P & S \\
R & 0 & -1 & 1 \\
P & 1 & 0 & -1 \\
S & -1 & 1 & 0
\end{pmatrix}
\]

The worst-case error is \( \epsilon \) if and only if the error is \( \epsilon \) under *every* distribution on inputs.

In some contexts, it makes sense to measure the expected number of bits exchanged by the parties as well.

For example, if the error is \( 1/3 \), then repeating the protocol \( O(\log(1/\delta)) \) reduces the error to at most \( \delta \).

This procedure is sometimes called confidence amplification by repetition.

The minimax principle can be applied to almost any randomized computational process. It does not rely on any features of the communication model.

\(^*\) von Neumann, 1928

von Neumann’s minimax principle can be seen as a consequence of linear programming duality.

This may seem like a game that is too simple to represent real-life games, but von Neumann observed that in fact such games represent almost all two-player games: “In other words, the player knows before hand how he is going to act in a precisely defined situation: he enters the game with a theory worked out in detail.”

The classic Rock-Paper-Scissors game is a zero-sum game with the following payoff matrix:

\[
\begin{align*}
R & \quad P & \quad S \\
R & 0 & -1 & 1 \\
P & 1 & 0 & -1 \\
S & -1 & 1 & 0
\end{align*}
\]
The pair of strategies \((x^*, y^*)\) is called an equilibrium—neither player has an incentive to unilaterally deviate from this pair of strategies. What are the equilibrium strategies for Rock-Paper-Scissors?

The vector \(x\) in the theorem corresponds to a distribution on the rows that the row player may use to choose his row, and \(y\) corresponds to a distribution that the column player may use. Then \(\min_{x \in A} \max_{y \in B} xMy\) gives the expected value of the payoff when the row player announces his choice for \(x\) first, and commits to it before the column player picks \(y\). In this case, the row player picks \(x\) to minimize \(\max_{y \in B} xMy\), and the column player then picks \(y\) to maximize \(xMy\). Similarly, the quantity \(\max_{y} \min_{x} xMy\) measures the expected payoff if the column player commits to a strategy \(y\) first, and the row player \(x\) gets to pick the best strategy \(x\) after seeing \(y\).

The first quantity \(\min_{x} \max_{y} xMy\) can only be larger than the second \(\max_{y} \min_{x} xMy\)—the row player wishes to minimize the value, and in the second case they have more information available to help make their choice.

Somewhat surprisingly, the min-max theorem states that they two quantities must be equal. In other words, there is a strategy \(y^*\) for the column player that guarantees a payoff that is equal to the amount she would get if she knew the strategy of the row player. There is a similar strategy \(x^*\) for the row player.

Now we leverage this powerful theorem to prove Yao’s min-max principle:

**Proof of Theorem 3.3.** If there is a protocol that computes \(g\) with error \(\epsilon\) in the worst case, then the same protocol must compute \(g\) with error \(\epsilon\) in the average case, no matter what the input distribution is. So, the average-case complexity is at most the worst-case complexity.

Conversely, suppose that for every distribution on inputs, the average-case complexity of the problem with error \(\epsilon\) is at most \(c\). Consider the matrix \(M\), where every row corresponds to an input to the protocol, and every column corresponds to a deterministic communication protocol of length at most \(c\), defined by

\[
M_{i,j} = \begin{cases} 
1 & \text{if protocol } j \text{ computes } g \text{ correctly on input } i, \\
0 & \text{otherwise.}
\end{cases}
\]

A distribution on the inputs corresponds to a choice of \(x\). Since a randomized protocol can be thought of as a distribution on deterministic protocols, a randomized protocol corresponds to a choice of \(y\). The success probability of a fixed randomized protocol \(y\) on inputs distributed according to \(x\) is exactly \(xMy\). So, by assumption, we know that \(\min_{x} \max_{y} xMy \geq 1 - \epsilon\). Theorem 3.4 implies that \(\max_{y} \min_{x} xMy \geq 1 - \epsilon\) as well—there is a fixed randomized protocol \(y^*\) that has error at most \(\epsilon\) under every distribution on inputs.
Public Coins vs Private Coins

Every private coin protocol can certainly be simulated by a public coin protocol, by making the private randomness visible to both parties. It turns out that every public coin protocol can also be simulated by a private coin protocol with only a small increase in communication\(^3\).

**Theorem 3.5.** If \( g : \{0,1\}^n \times \{0,1\}^n \to \{0,1\} \) can be computed with \( c \) bits of communication, and error \( \epsilon \) in the worst case, then it can be computed by a private coin protocol with \( c + \log(n/\epsilon^2) + O(1) \) bits of communication, and error \( 2\epsilon \) in the worst case.

**Proof.** We use the probabilistic method to find the required private coin protocol. Suppose the public coin protocol uses a random string \( r \) as the source of all randomness, drawn from some distribution \( \mu \).

To design the private coin protocol, we start by picking \( t \) independent random strings \( r_1, \ldots, r_t \) sampled from \( \mu \).

For any fixed input \( (x,y) \), some of these \( t \) random strings lead to the public coin protocol computing the right answer, and some of the lead to the protocol computing the wrong answer. However, the probability that \( r_i \) gives the right answer is at least \( 1 - \epsilon \). Thus, by the Chernoff bound, the probability that \( 1 - 2\epsilon \) fraction of the \( t \) strings lead to the wrong answer is at most \( 2^{\Omega(\epsilon^2)} \). We set \( t = O(2n/\epsilon^2) \) to be large enough so that this probability is less than \( 2^{-2n} \). Then by the union bound, we get that the probability that more than \( 2\epsilon t \) of these strings give the wrong answer for any input is less than 1. Thus, there must be some fixed strings with this property.

The private coin protocol is now simple. We fix \( r_1, \ldots, r_t \) with the property that for any input \( (x,y) \), the fraction of strings giving the wrong answer is at most \( 2\epsilon \). Alice samples a uniformly random element \( i \in \{1,2,\ldots,t\} \), and sends \( i \) to Bob, which takes at most \( \log(n/\epsilon^2) + O(1) \) bits. Alice and Bob then run the original protocol using the randomness \( r_i \).


\[^3\text{Newman, 1991}\]

It is known that computing whether or not two \( n \)-bit strings are equal requires \( \Omega(\log n) \) bits of communication if only private coins are used. This shows that Theorem 3.5 is tight.

Notice that only one party uses randomness.

Nearly Monochromatic Rectangles

Monochromatic rectangles proved to be a very useful concept for understanding deterministic protocols. A similar role is played by nearly monochromatic rectangles when trying to understand randomized protocols.

Given a randomized protocol, and a distribution on inputs \( \mu \), one can always fix the randomness of the protocol in the way that mini-
mizes the probability of error under $\mu$. The result is a deterministic protocol whose error under $\mu$ is at most the error of the randomized protocol.

The following theorem describes some aspects of the connection between randomized protocols and nearly monochromatic rectangles. This is studied in greater detail in future chapters, where lower bounds on randomized communication complexity are proved.

**Theorem 3.6.** If there is a deterministic $c$-bit protocol $\pi$ with error at most $\epsilon$ under a distribution $\mu$, and a set $S$ such that

$$\Pr[\pi(X,Y) \in S] > 2\sqrt{\epsilon},$$

then there exists a rectangle $R$ such that

- $\pi$ has the same outcome for all inputs in $R$, and this outcome is in $S$.
- $\Pr[(X,Y) \in R] \geq \sqrt{\epsilon} \cdot 2^{-c}$.
- $\Pr[\pi \text{ makes an error}|(X,Y) \in R] \leq \sqrt{\epsilon}$.

**Proof.** By Theorem 1.6, we know that the protocol induces a partition of the space into $t \leq 2^c$ rectangles $R_1, R_2, \ldots, R_t$, and in each of these rectangles, the outcome of the protocol is determined.

For each rectangle $R_i$ in the collection, define the number

$$\epsilon(R_i) = \Pr[\text{the protocol makes an error}|(X,Y) \in R_i].$$

Let $\rho(R_i)$ denote the number $\Pr[(X,Y) \in R_i]$. If $R$ denotes the rectangle that the inputs $X, Y$ belong to, we have that $\mathbb{E}[\epsilon(R)] \leq \epsilon$, so Markov’s inequality gives $p(\epsilon(R) > \sqrt{\epsilon}) \leq \sqrt{\epsilon}$. Thus,

$$\mathbb{E}\left[\frac{1}{\rho(R)}\right] = \sum_{i=1}^{t} \Pr[R = R_i] \cdot \frac{1}{\Pr[(X,Y) \in R_i]} = t,$$

so by Markov’s inequality, we get $p(1/\rho(R) > t/\sqrt{\epsilon}) \leq \sqrt{\epsilon}$.

Since

$$\Pr[\pi(X,Y) \in S] > 2\sqrt{\epsilon} > \Pr[\epsilon(R) > \sqrt{\epsilon}] + \Pr[1/\rho(R) > t/\sqrt{\epsilon}],$$

there must be a rectangle $R^*$ in the collection corresponding to an outcome in $S$, with $\epsilon(R^*) \leq \sqrt{\epsilon}$ and $\rho(R^*) \geq \sqrt{\epsilon}/t \geq \sqrt{\epsilon} \cdot 2^{-c}$, as required.

- **Exercise 3.1**

  In this exercise, we design a randomized protocol for finding the first difference between two $n$-bit strings. Alice and Bob are given $n$ bit strings $x \neq y$ and want to find the smallest $i$ such that $x_i \neq y_i$. We
already saw how to accomplish this using \( O(\log n \log \log n) \) bits of communication. Here we do it with \( O(\log n) \) bits of communication. For simplicity, assume that \( n \) is a power of two.

Define a rooted tree as follows. Every vertex \textit{corresponds} to an interval of coordinates from \([n]\). The root corresponds to the interval \( I = [n] \). Every internal vertex corresponding to the interval \( I \) will have two children, the left child corresponding to the first half of \( I \) and the right child corresponding to the right half of \( I \). This defines a tree of depth \( \log n \), where the leaves correspond to intervals of size 1 (i.e. coordinates) of the input. At each leaf, attach a path of length \( 3 \log n \). Every vertex of this path represents the same interval of size 1. The depth of the tree is now \( 4 \log n \).

1. Fill in the details of the following protocol. Prove an upper bound on the expected number of bits communicated and a lower bound on the success probability.

The players use their inputs and hashing to start at the root of the tree and try to navigate to the smallest interval that contains the index \( i \) that they seek. In each step, the players either move to a parent or move to a child of the node that they are at. When the players are at a vertex that corresponds to the interval \( I \), they should first exchange \( O(1) \) hash bits to confirm that the first difference does lie in \( I \). If this hash shows that the first difference does not lie in \( I \), they should move to the parent of the current node. Otherwise, they exchange \( O(1) \) hash bits to decide to which child to move to.

2. Argue that as long as the number of nodes where the protocol made the \textit{right} choice exceeds the number of nodes where the players made the \textit{wrong} choice by \( \log n \), the protocol you defined succeeds in computing \( i \).

3. Use the Chernoff-Hoeffding bound to argue that the number of hashes that give the right answer is high enough to ensure that the protocol succeeds with high probability on any input.

\textbf{Exercise 3.2}

Show that if the inputs to greater-than are sampled uniformly and independently, then there is a protocol that communicates only \( O(\log(1/\epsilon)) \) bits and has error at most \( \epsilon \) under this distribution.
4

Numbers On Foreheads

When more than two parties communicate, there are several different scenarios that may arise. The number-on-forehead model is one way to generalize the case of two party communication to the multiparty setting. In this model, there are $k$ parties communicating, and the $i$’th party has an input drawn from the set $X_i$ written on her forehead. So, each party can see all of the $k$ inputs except the one written on her forehead.

Since each party can see most of the inputs, the parties often do not need to communicate much. Proving lower bounds for this model is usually, therefore, more difficult than for the two party case. Indeed, we do not yet know of any explicit examples of functions that require the maximum communication complexity in this model, in stark contrast to the models we have discussed before.

Some Protocols

We start with some examples of clever number-on-forehead protocols.

Equality We have seen that every deterministic protocol for computing equality in the two party setting must have complexity $n + 1$. Perhaps surprisingly, the complexity of equality is quite different in the number-on-forehead model. Suppose 3 parties each have an $n$ bit string written on their foreheads. Then there is a very efficient protocol for computing whether all three strings are the same: Alice announces whether or not Bob and Charlie’s strings are the same, and Bob announces whether or not Alice and Charlie’s strings are the same. This computes equality with 2 bits of communication.

Intersection size Suppose there are $k$ parties, and the $i$’th party has a subset $X_i \subseteq [n]$ on their forehead. The parties want to compute the
size of the intersection $\bigcap_i X_i$. We shall describe a protocol\(^2\) that requires only $O(k^4(1 + n/2^k))$ bits of communication.

We start by describing a protocol that requires only $O(k^2 \log n)$ bits of communication, as long as $n < \binom{k}{k/2}$. It is helpful to think of the input as a $k \times n$ boolean matrix. Each of the parties knows all but one row of this matrix, and they wish to compute the number of all 1’s columns. Let $C_{i,j}$ denote the number of columns containing $j$ ones that are visible to the $i$’th party. The parties compute and announce the values of $C_{i,j}$, for each $i, j$. Since this involves each player announcing $k + 1$ numbers, the communication complexity of the protocol is at most $(k + 1)k \log n \leq O(k^2 \log n)$.

The following claim allows us to prove that the protocol achieves its goal.

**Claim 4.1.** Let $A_j$ denote the actual number of columns with $j$ ones in them. If there are two valid solutions $A_k, \ldots, A_0$ and $A'_k, \ldots, A'_0$ that are both consistent with the values $C_{i,j}$, then either $A'_k = A_k$, or for each $j$, $|A_j - A'_j| \geq \binom{k}{j}$.

**Proof.** Suppose $A'_k \neq A_k$. Then $|A_k - A'_k| \geq 1 = \binom{k}{k}$. We prove the claim by induction on $j = k, k-1, k-2, \ldots, 0$. Since a column of weight $j$ is observed as having weight $j-1$ by $j$ parties, and having weight $j$ by $k-j$ parties, we have:

$$(k-j)A_j + (j+1)A_{j+1} = \sum_{i=1}^{k} C_{i,j} = (k-j)A'_j + (j+1)A'_{j+1}$$

$$\Rightarrow |A_j - A'_j| \geq \binom{j+1}{k-j} |A_{j+1} - A'_{j+1}|$$

$$\geq \binom{j+1}{k-j} \binom{k}{j+1} = \binom{k}{j},$$

as required. \(\square\)

Claim 4.1 shows that knowing $C_{i,j}$ for all $i, j$ allows one to determine the number of all 1’s columns. Indeed, if $n < \binom{k}{k/2}$, then there can only be one possible value for $A_k$, since otherwise $|A'_{k/2} - A_{k/2}| > n$, which is not possible.

To obtain a protocol for general $n$, the parties divide the columns of the matrix into blocks of size $\binom{k}{k/2} - 1$, and count the number of all 1’s columns in each block using the above idea separately. The total communication is then at most

$$\left(\frac{n}{\binom{k}{k/2}} + 1\right) \cdot k^2 \log \left(\frac{k}{k/2}\right) \leq O(k^4(1 + n/2^k)),$$

as claimed.

---

\(^2\) Grolmusz, 1998; and Babai et al., 2003

In Chapter 5, we prove that at least $n/4^k$ bits of communication are required.

Recall that $\binom{k}{k/2} \approx 2^k/\sqrt{k}$. 
**Exactly n** Suppose 3 parties each have a number from \([n]\) written on their forehead, and want to know whether these numbers sum to \(n\) or not. A trivial protocol is for one of the parties to announce one of the numbers she sees, and then the relevant party announces the answer, which takes \(\log n\) bits. Here we use ideas of Behrend\(^3\) to show that one can do it with just \(O(\sqrt{\log n})\) bits of communication. Behrend’s ideas lead to a coloring of the integers that avoids monochromatic three-term arithmetic progressions:

**Theorem 4.2.** One can color the set \([m]\) with \(2^{O(\sqrt{\log m})}\) colors, such that for any \(a, b \in [m]\), if the numbers \(a, a + b, a + 2b\) are all in \([m]\), then they do not have the same color.

First we explain how the coloring from Theorem 4.2 can be used to get a protocol for the exactly \(n\) problem with communication \(O(\sqrt{\log n})\). Suppose the three inputs are \(x, y, z\). Consider the numbers

\[
x' = n - y - z, \quad y' = n - x - z.
\]

Alice can compute \(x'\), and Bob can compute \(y'\). We have \(x - x' = y - y' = x + y + z - n\). So, \(x = x'\) and \(y = y'\) exactly when \(x + y + z = n\). In the protocol, Alice announces the color of \(x' + 2y\) in the coloring promised by Theorem 4.2, when applied to the set \([3n]\). Bob and Charlie just send a bit to indicate whether the announced color is the same as the color of \(x' + 2y'\) or \(x + 2y\). If all three colors are the same, the players conclude that the sum of their numbers is \(n\). If the colors are not the same, they conclude that the sum is not \(n\). The communication complexity of the protocol is at most \(O(\sqrt{\log n})\).

The reason the protocol works is that if \(x + y + z \neq n\), then \(x - x' = y - y' \neq 0\), and

\[
\begin{align*}
x + 2y, \\
x + 2y + x' - x = x' + 2y, \\
x' + 2y + 2(y' - y) = x + 2y'
\end{align*}
\]

form an arithmetic progression. Thus, by the property of the coloring, all three colors cannot be the same. On the other hand, if \(x + y + z = n\), then \(x' = x\) and \(y' = y\), and all three colors must be the same.

Now we turn to proving Theorem 4.2. A triple of points \(a, a + b, a + 2b\) can be also thought of as a triple of the form \(x, (x + y)/2, y\).

In other words, we want to find a coloring of \([m]\) so that if \(x, y\) are of the same color, then \((x + y)/2\) has a different color. The basic observation is that the points on a sphere satisfy a similar property—if \(x, y\) are two vectors that lie on the same sphere, then

\[x + y = 2\sqrt{2}\]

The average of two distinct points on a sphere cannot lie on the sphere—they must lie in the interior.
(x + y)/2 must be shorter, and so cannot lie on the same sphere. The idea for the proof is to discretize this property of Euclidean length.

**Proof of Theorem 4.2.** We shall choose parameters \( d, r \) with \( d^r > m \) and \( d \) is divisible by 4. To carry out the above intuition, we need to convert each number \( x \in [m] \) into a vector. To do this, we write each number \( x \in [m] \) in base \( d \), using at most \( r \) digits. We express

\[
x = \sum_{i=0}^{r-1} x_i d^i,
\]

where \( x_i \in [d - 1] \). One can thus interpret \( x \in [m] \) as a vector \( v(x) \in \mathbb{R}^r \) whose \( i \)’th coordinate \( x_i \). We approximate each of these vectors using a vector where every coordinate is off by at most \( d/4 \): let \( w(x) \) be the vector where the \( i \)’th coordinate is the largest number of the form \( jd/4 \) such that \( jd/4 \leq x_i \) and \( j \) is an integer.

Color each number \( x \in [n] \) by the vector \( w(x) \) and the integer \( \|v(x)\|^2 = \sum_{i=0}^{r-1} x_i^2 \). The number of choices for \( w(x) \) is at most \( 2^{O(r)} \), and the number of possible values for \( \|v(x)\|^2 \) is at most \( O(rd^2) \), so the total number of possible colors is at most \( 2^{O(r + \log d)} \).

Setting \( r = \sqrt{\log m} \), \( d = 2^{O(\sqrt{\log m})} \) gives the required bound.

It only remains to check that the coloring avoids arithmetic progressions. For the sake of finding a contradiction, suppose \( a, b \in [m] \) are such that \( a, a + b, a + 2b \) all get the same color. Then we must have \( \|v(a)\| = \|v(a + b)\| = \|v(a + 2b)\| \), so the three vectors \( v(a), v(a + b), v(a + 2b) \) all lie on a sphere. We will get a contradiction by proving that \( v(a + b) = \frac{v(a) + v(a + 2b)}{2} \).

To prove this, we need to use the fact that the points also satisfy: \( w(a) = w(a + b) = w(a + 2b) \). We get a contradiction by proving that the following quantity is 0:

\[
v(a + 2b) + v(a) - 2v(a + b)
= v(a + 2b) + v(a) - 2v(a + b) - w(a + 2b) - w(a) + 2w(a + b)
= (v(a + 2b) - w(a + 2b)) + (v(a) - w(a)) - 2(v(a + b) - w(a + b)).
\]

Let \( W(x) = \sum_{i=0}^{r-1} w(x)_i d^i \). Then we see that for any \( x \), \( v(x) - w(x) \) is just the base \( d \) representation of \( x - W(x) \), and this number is at most \( d/4 \) in each coordinate. This means that \( (v(a + 2b) - w(a + 2b)) + (v(a) - w(a)) - 2(v(a + b) - w(a + b)) \) is simply the base \( d \) representation of \( a + 2b - W(w(a + 2b)) + a - W(w(a)) - 2(a + b - W(w(a + b))) \) = 0. So, it must be 0, as required.

\( \square \)
Cylinder Intersections

The basic building blocks of protocols in the number-on-forehead model are cylinder intersections. They play the same role that rectangles play in the case that the number of parties is 2. Any set $S \subseteq X_1 \times \cdots \times X_k$ can be described using its characteristic function:

$$\chi_S(x_1, \ldots, x_k) = \begin{cases} 1 & \text{if } (x_1, \ldots, x_k) \in S, \\ 0 & \text{otherwise.} \end{cases}$$

We can then define cylinder intersections as:

**Definition 4.3.** $S \subseteq X_1 \times \cdots \times X_k$ is called a cylinder if $\chi_S$ does not depend on one of its inputs. $S$ is called a cylinder intersection if it can be expressed as an intersection of cylinders.

If $S$ is a cylinder intersection, we can always express

$$\chi_S(x_1, \ldots, x_k) = \prod_{i=1}^{k} \chi_i(x_1, \ldots, x_k),$$

where $\chi_i$ is a boolean function that does not depend on the $i$’th input.

**Fact 4.4.** The intersection of two cylinder intersections is also a cylinder intersection.

Just as for rectangles, we say that a cylinder intersection is monochromatic with respect to a function $g$, if $g(x) = g(y)$ for every two inputs $x, y$ in the cylinder intersection. In analogy with the 2 party case, we have the following theorem:

**Theorem.**
Theorem 4.5. If the deterministic communication complexity of \( g : \mathcal{X}_1 \times \cdots \times \mathcal{X}_k \rightarrow \{0, 1\} \) is \( c \), then \( \mathcal{X}_1 \times \cdots \times \mathcal{X}_k \) can be partitioned into at most \( 2^c \) monochromatic cylinder intersections with respect to \( g \).

Indeed, for every outcome of the protocol \( m \), it is easy to verify that the set of inputs that are consistent with that outcome form a cylinder intersection.

Lower Bounds from Ramsey Theory

Cylinder intersections are more complicated than rectangles, and this makes proving lower bounds in the number-on-forehead model challenging. Here we explore some lower bounds in this model is by appealing to arguments from Ramsey Theory.

Let us consider the \( \text{Exactly } n \) problem: Alice, Bob and Charlie are each given a number from \( [n] \), written on their foreheads, and want to know if their numbers sum to \( n \). We have shown that there is a protocol that computes this function using \( O(\sqrt{\log n}) \) bits of communication. Here we show that \( \Omega(\log \log \log n) \) bits of communication are required.

Denote by \( c_n \) the deterministic communication complexity of the exactly \( n \) problem. To understand the behavior of \( c_n \), we need to relate it to a combinatorial structure studied in Ramsey Theory. Three points in \( [n] \times [n] \) form a corner if they are of the form \( (x, y), (x + d, y), (x, y + d) \) for some integer \( d \). A coloring of \( [n] \times [n] \) with \( C \) colors is a function \( g : [n] \times [n] \rightarrow [C] \). We say that the coloring avoids monochromatic corners if there is no corner such that all three points get the same color. Let \( C_n \) be the minimum number of colors required to avoid monochromatic corners in \( [n] \times [n] \). We claim that \( C_n \) essentially captures the value of \( c_n \):

\[ C_n \text{ essentially captures the value of } c_n. \]

\[ \text{Figure 4.5: A cylinder intersection.} \]
\[ \text{Watch an animation.} \]

\[ \text{Figure 4.6: Figure 4.5 viewed from above.} \]

\[ \text{Figure 4.7: Figure 4.5 viewed from the left.} \]

\[ \text{Figure 4.8: Figure 4.5 viewed from the right.} \]

In Chapter 5 we discuss the discrepancy method, which leads to the strongest known lower bounds in the number-on-forehead model.

\footnote{Chandra et al., 1983}
Lemma 4.6. $\log C_{n/3} \leq c_n \leq 2 + \log C_n$.

Proof. To prove $\log C_n \geq c_n$, suppose there is a coloring with $C_n$ colors that avoids monochromatic corners. As in the protocol discussed at the beginning of this chapter, Alice can compute $x' = n - y - z$, and Bob can compute $y' = n - x - z$. Alice will then announce the color of $(x', y')$, and Bob and Charlie will say whether this color is the same as the color of $(x, y')$ and $(x, y)$. The three points $(x, y'), (x', y), (x, z)$ form a corner, since $x' - x = n - x - y - z = y' - y$. So all three points have the same color if and only if all three points are the same, which can only happen when $x + y + z = n$.

To prove that $\log C_{n/3} \leq c_n$, suppose there is a protocol solving the exactly-$n$ problem with $c$ bits of communication. Then by Theorem 4.5, every input to the protocol can be colored by one of $2^c$ colors that is the name of the corresponding cylinder intersection. This induces a coloring of $[n/3] \times [n/3]$; color $(x, y)$ by the name of the cylinder intersection containing the point $(x, y, n - x - y)$. We claim that this coloring avoid monochromatic corners. Indeed, if $(x, y), (x + d, y), (x, y + d)$ is a monochromatic corner with $d \neq 0$, then $(x, y, n - x - y), (x + d, y, n - x - y - d), (x, y + d, n - x - y - d)$ must all belong to the same cylinder intersection. But then $(x, y, n - x - y - d)$ must also be in the same cylinder intersection, since it agrees with each of the three points in two coordinates. That contradicts the correctness of the protocol, since $x + y + n - x - y = n$ but $x + y + n - x - y - d \neq n$. \hfill $\Box$

Next we prove \textsuperscript{5}:

Theorem 4.7. $C_n \geq \Omega \left( \frac{\log \log n}{\log \log \log n} \right)$.

We prove the theorem by induction. However, to carry out the proof, we need to first strengthen the statement in order for the induction to go through. We prove that the matrix must either contain a monochromatic corner, or a structure called a rainbow-corner. A rainbow-corner with $r$ colors and center $(x, y)$ is specified by a set of $r$ distinct colors, and numbers $d_1, \ldots, d_{r-1}$, such that $(x + d_i, y)$ and $(x, y + d_i)$ are both colored using the $i$'th color, and $(x, y)$ is colored by the $r$'th color.

Proof of Theorem 4.7. We shall prove by induction that as long as $C > 3$, if $n \geq 2C^{2r}$, then any coloring of $[n] \times [n]$ with $C$ colors must contain either a monochromatic corner, or a rainbow-corner with $r$ colors. When $r = C + 1$, this means that if $n \geq 2C^{2C + 1}$, then $[n] \times [n]$ must contain a monochromatic corner, proving that $C_n \geq \Omega \left( \frac{\log \log n}{\log \log \log n} \right)$.

\textsuperscript{5} Graham, 1980; and Graham et al., 1980

\begin{figure}[h]
\centering
\includegraphics[width=0.2\textwidth]{monochromatic_corner.png}
\caption{A monochromatic corner.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=0.2\textwidth]{rainbow_corner.png}
\caption{A rainbow-corner.}
\end{figure}
For the base case, when \( r = 2 \), since \( n \geq 2^{2^r} > C \), two of the points of the type \( (x, n - x) \) must have the same color. If \((x, n - x)\) and \((x', n - x')\) have the same color, with \( x > x' \), then \((x', n - x), (x, n - x), (x', n - x')\) are either a monochromatic corner, or a rainbow-corner with 2 colors.

For the inductive step, assume \( n = 2^{2^r} \). The set \( [n] \) can be partitioned to \( m = 2^{2^r - C^{2^{2^{r-1}}}} \) intervals \( I_1, I_2, \ldots, I_m \), each of size exactly \( 2^{2^{2^{r-1}}} \). By induction, each of the sets \( I_j \times I_j \) must have either a monochromatic corner, or a rainbow-corner with \( r - 1 \) colors. If one of them has a monochromatic corner, we are done, so suppose they all have rainbow-corners with \( r - 1 \) colors. Since a rainbow-corner is specified by choosing the center, choosing the colors and choosing the offsets for each color, there are at most

\[
(2^{2^{2^{r-1}}})^2 \cdot C^r \cdot (2^{2^{2^{2^{r-1}}}})^C = 2^{2^{2^{2^{2^{r-1}+r}}}} + C^{2^{r-1}} < m
\]

potential rainbow-corners in each of these sets. Since the number of possible rainbow corners is less than \( m \), there must be some \( j < j' \) that have exactly the same rainbow corner with the same coloring. These two rainbow corners induce a monochromatic corner centered in the box \( I_j \times I_j' \), or a rainbow-corner with \( r \) colors (see Figure 4.11).

\[ \square \]

**Exercise 4.1**

Give an NOF protocol that allows to compute whether or not there is a column with \( k/2 \) ones. HINT: The protocol is identical to the protocol for disjointness—we only need to argue that the information revealed by the players is enough to count the number of columns with \( k/2 \) ones.

**Exercise 4.2**

Define the generalized inner product function \( GIP \) as follows. For \( k \) inputs \( x_1, \ldots, x_k \in \{0, 1\}^n \),

\[
GIP(x) = \sum_{j=1}^{n} \prod_{i=1}^{k} x_{ij} \mod 2.
\]

This exercise outlines a number-on-forehead \( GIP \) protocol using \( O(n/2^k + k) \) bits. It will be convenient to think about the input \( X \) as a \( k \times n \) matrix with rows corresponding to \( x_1, \ldots, x_k \).

- Suppose the players know a string \( z \in \{0, 1\}^k \) with the property that no column of \( X \) is equal to \( z \). Show that the players can use \( z \) to compute the number of all 1’s columns as follows. Assume the first \( t \) coordinates of \( z \) are zeroes and the rest are ones. For
\( \ell \in \{0, 1, \ldots, k - 1\} \) define \( c_\ell \) as the number of columns in \( X \) with \( \ell \) zeroes, followed by either a one or zero, followed by \( k - \ell - 1 \) ones. Prove that \( \sum_{\ell=0}^{k-1} c_\ell \equiv \text{GIP}(x) \mod 2 \). Use these ideas to find a protocol to compute \( \text{GIP}(x) \) using \( O(k) \) bits assuming the players know \( z \).

- Exhibit an overall protocol for \( \text{GIP} \) by showing that the players can agree upon a vector \( z \) and communicate to determine \( c_\ell \mod 2 \) using \( O(n/2^k + k) \) bits.

### Exercise 4.3

Given a function \( g \), recall that we define \( g^r \) to be the function that computes \( r \) copies of \( g \). This exercise explores, in the number-on-forehead model, what we can say about the communication required by \( g^r \), knowing the communication complexity of \( g \).

The approach taken in the proof of Theorem 1.33 does not work because cylinder intersections do not tensorize nicely like rectangles do. Fortunately, we can appeal to a powerful result from Ramsey theory called the Hales-Jewett Theorem to prove that the communication complexity of \( g^r \) must increase as \( r \) increases.

For an arbitrary finite set \( S \), the Hales-Jewett Theorem gives insight into the structure of the cartesian product \( S^n = S \times S \times \cdots \) as \( n \) grows large. For the precise statement we need the notion of a combinatorial line. The combinatorial line specified by a nonempty set of indices \( I \subseteq [n] \) and a vector \( v \in S^n \) is the set of all \( x \in S^n \) so that \( x_i = v_i \) for every \( i \notin I \) and \( x_i = x_j \) for every \( i, j \in I \). For example, when \( S = \{3\} \) and \( n = 4 \) then the set \( \{1132, 2232, 3332\} \) is a combinatorial line with \( I = \{1, 2\} \) and \( v_3 = 3, v_4 = 2 \).

Given a set \( S \) and a number \( t \), the Hales-Jewett theorem says that as long as \( n \) is large enough, any coloring of \( S^n \) with \( t \) colors must contain a monochromatic combinatorial line.

Assume that the communication complexity of \( g \) is strictly greater than the number of players. Define \( c_n \) to be communication required to compute the \( \text{AND} \) of \( n \) copies of \( g \). Prove that

\[
\lim_{n \to \infty} c_n = \infty.
\]

### Exercise 4.4

A three player NOF puzzle demonstrates that unexpected efficiency is sometimes possible.

**Inputs:** Alice has a number \( i \in [n] \) on her forehead, Bob has a number \( j \in [n] \) on his forehead, and Charlie has a string \( x \in \{0, 1\}^n \) on his forehead.
Output: On input \((i, j, x)\) the goal is for Charlie to output the bit \(x_k\) where \(k = i + j \mod n\).

Question: Find a deterministic protocol in which Bob sends one bit to Charlie, and Alice sends \(\left\lfloor \frac{n}{2} \right\rfloor\) bits to Charlie; Alice and Bob must send Charlie their message simultaneously. Charlie is then able to output the correct answer.

Exercise 4.5

Show that any degree \(d\) polynomial over \(\mathbb{F}_2\) over the variables \(x_1, \ldots, x_n\) can be computed by \(d + 1\) players with \(O(d)\) bits in the number-on-forehead model, for any partition of \(x_1, \ldots, x_n\) to \(d + 1\) parts, where each party has \(n/(d + 1)\) bits on their forehead (you may assume \(d + 1\) divides \(n\)).
Besides being useful in communication complexity, the concept of discrepancy shows up in several other fields, like geometry and learning theory.

5
Discrepancy

If the elements of a large set are randomly colored with two colors, then we would expect roughly half of the elements to get one color and half to get the other color. The discrepancy of a set refers to the degree to which a given coloring of the set does not look random in this sense. The techniques we have developed for proving lower bounds in prior chapters all rely on the fact that efficient communication protocols lead to partitions of the space into a small number of monochromatic sets. A monochromatic set has large discrepancy—it does not look like it was colored randomly. In this chapter, we shall prove lower bounds by arguing that for a given function, the discrepancy of large rectangles or cylinder intersections must be small.

The ideas we develop will lead to lower bounds for randomized protocols, and the best known lower bounds in the number-on-forehead model. Discrepancy measures the degree to which a fixed object resembles a truly random object. In our context, a truly random 0/1 valued function is approximately balanced on every specific set—the number of 0 entries in the set is close to the number of 1 entries.

Suppose \( g : D \to \{0, 1\} \) is a function and \( \mu \) is a distribution on \( D \). Suppose \( S \subseteq D \) is a subset of the domain, and let \( \chi_S \) be the characteristic function of the set \( S \), so

\[
\chi_S(x) = \begin{cases} 
1 & \text{if } x \in S, \\
0 & \text{otherwise}.
\end{cases}
\]

The discrepancy of \( g \) with respect to \( S \) and \( \mu \) is defined as

\[
\left| \mathbb{E}_\mu [\chi_S(x) \cdot (-1)^{g(x)}] \right|.
\]

Roughly speaking, the discrepancy can be small for two reasons:

Besides being useful in communication complexity, the concept of discrepancy shows up in several other fields, like geometry and learning theory.
either the set $S$ is small, or $g$ is close to being balanced on $S$.

For every fixed $S$ and $\mu$, a random function has low discrepancy with high probability. However, every $g : D \to \{0,1\}$ is monochromatic on a set of $\mu$-measure at least $1/2$, so the discrepancy is certainly large for some sets. We shall therefore restrict our attention to sets of a specific structure, like rectangles or cylinder intersections.

For example, we shall consider the domain $D = \mathcal{X} \times \mathcal{Y}$ and the maximum discrepancy over all rectangles in $D$.

To relate discrepancy to protocols, recall that the bias of a set $S$ is defined to be

$$\text{bias}(S) = \max_b \Pr[g(x) = b | x \in S].$$

A large set with large bias must lead to high discrepancy:

**Fact 5.1.** If $S$ is a set of inputs with $\Pr[\mu | x \in S] \geq \delta$, and $\text{bias}(S) \geq 1 - \epsilon$, then the discrepancy of $g$ with respect to $S$ is at least $(1 - 2\epsilon)\delta$.

**Proof.** Only points inside $S$ contribute to its discrepancy. Since a $(1 - \epsilon)$ fraction of these points have the same value under $g$, the discrepancy is at least $\delta(1 - \epsilon - \epsilon) = \delta(1 - 2\epsilon)$.

Functions that have small discrepancy with respect to rectangles must have large randomized communication complexity:

**Theorem 5.2.** For a fixed distribution $\mu$, if the discrepancy of $g$ with respect to every rectangle is at most $\gamma$, then any protocol computing $g$ with error $\epsilon$ when the inputs are drawn from $\mu$ must have communication complexity at least $\log \left( \frac{1 - 2\epsilon}{\gamma} \right)$.

**Proof.** Suppose $\pi$ is a deterministic protocol of length $c$ with error $\epsilon$ with respect to $\mu$, and let $\pi(x,y)$ denote its output. By Lemma 1.4, the leaves of the protocol $\pi$ correspond to some rectangles $R_1, \ldots, R_t$, with $t \leq 2^c$. Then we have

$$1 - 2\epsilon \leq \Pr[\mu | \pi(x,y) = g(x,y)] - \Pr[\mu | \pi(x,y) \neq g(x,y)]$$

$$= \mathbb{E}_{\mu} \left[ (-1)^{\pi(x,y) + g(x,y)} \right]$$

$$= \mathbb{E}_{\mu} \left[ (-1)^{\pi(x,y)} \cdot (-1)^{g(x,y)} \right]$$

$$= \mathbb{E}_{\mu} \left[ \sum_{i=1}^{t} \chi_{R_i}(x,y) \cdot \varphi(R_i) \cdot (-1)^{g(x,y)} \right],$$

where $\varphi(R_i) = -1$ if the protocol outputs 1 on inputs from $R_i$, and
\( \varphi(R_i) = 1 \) if the protocol outputs 0. We can continue to bound:

\[
1 - 2\epsilon \leq \sum_{i=1}^t |E[R_i(x, y) \cdot (-1)^g(x, y)]|
\leq 2^c \cdot \max_R |E[R(x, y) \cdot (-1)^g(x, y)]|,
\]

where the maximum is taken over all choices of rectangles. Rearranging, we get

\[
2^c \geq \frac{1 - 2\epsilon}{\max_R |E[R(x, y) \cdot (-1)^g(x, y)]|}.
\]

Given Theorem 5.2, we only need to bound the discrepancy of functions with respect to rectangles and cylinder intersections to prove communication lower bounds. Simple counting arguments show that most functions have small discrepancy with respect to rectangles and cylinder intersections. Our goal in this chapter is to prove that some explicit functions have small discrepancy.

In this chapter, we shall explore two techniques to bound the discrepancy. First, we show how to appeal to Jensen’s inequality to bound discrepancy. Later, we show one can use concentration bounds from probability theory to control it. To begin, let us explore some examples where Jensen’s inequality is useful in combinatorics.

**Jensen’s Inequality in Combinatorics**

Jensen’s inequality plays a key role in understanding many combinatorial properties of graphs. One kind of question where it is very useful is: how many edges must a graph have before it is forced to contain a small cycle?

Given a graph on \( n \) vertices, the maximum number of edges it can have is \( \binom{n}{2} \). We say that the graph has edge density \( \epsilon \) if it has \( \epsilon \binom{n}{2} \) edges.

While there are graphs with constant edge density that have no 3-cycles, there are no graphs with constant density that avoid 4-cycles.

**Lemma 5.3.** Every \( n \)-vertex graph with at least \( \frac{2}{\sqrt{n}} \cdot \binom{n}{2} \) edges contains a 4-cycle.

**Proof.** Let \( G \) be a graph with \( n \) vertices and \( \frac{2}{\sqrt{n}} \cdot \binom{n}{2} \) edges. When \( n < 4 \), the theorem is vacuously true, since \( \frac{2}{\sqrt{n}} > 1 \). So assume \( n \geq 4 \). For two vertices \( x, y \), let \( E(x, y) = 1 \) when there is an edge

Figure 5.1: A dense graph with no 3-cycles.

The point-line incidence graph of a projective plane has \( n \) vertices, approximately \( n^{3/2} \) edges and no 4-cycles, which means that the lemma is sharp up to constants.
between the vertices $x$ and $y$, and 0 otherwise. Let $X, X', Y$ be 3 vertices chosen independently and uniformly at random from the graph. Use Jensen’s inequality to estimate

$$\mathbb{E}_{x,x',y} \left[ \mathbb{E}_{y} \left[ E(x,y) \cdot E(x',y) \right] \right] = \mathbb{E}_{y} \left[ \mathbb{E}_{x,x'} \left[ E(x,y) \cdot E(x',y) \right] \right] \geq \mathbb{E}_{y} \left[ E(x,y) \right]^2.$$

This last quantity is at least $\left( \frac{2}{\sqrt{n}} - \frac{1}{n} \right)^2$, since $E(x,y)$ is 0 only when $x = y$ or $\{x, y\}$ is not an edge. Using the fact that $n \geq 4$, we compute:

$$\left( \frac{2}{\sqrt{n}} - \frac{1}{n} \right)^2 \geq \frac{4}{n} - \frac{4}{n^{3/2}} + \frac{1}{n^2} \geq \frac{4}{n} - \frac{2}{n} + \frac{1}{n^2} > \frac{2}{n}.$$

For the sake of proving a contradiction, assume $G$ has no 4-cycles. Then for each $x \neq x'$, there is at most one $y$ with $E(x,y) \cdot E(x',y) = 1$, and so, $\mathbb{E}_{y} \left[ E(x,y) \cdot E(x',y) \right] \leq \frac{1}{n}$. The probability that $x = x'$ is exactly $1/n$. Thus, we get

$$\mathbb{E}_{x,x',y} \left[ \mathbb{E}_{y} \left[ E(x,y) \cdot E(x',y) \right] \right] \leq \frac{1}{n} + \frac{1}{n} \leq \frac{2}{n},$$

contradicting the bound we proved above. \qed

We can use similar ideas as above to prove that every dense bipartite graph must contain a reasonably large bipartite clique. Here is a slightly different way to find a clique in a bipartite graph:

**Lemma 5.4.** Suppose $G$ is a bipartite graph with edge density $e > 0$, and bipartition $A, B$ with $|A| = m, |B| = n$, and let $k \leq \frac{\log n}{\log(2e/c)}$. Then if $e \geq 2k/m$, there are subsets $Q \subseteq A, R \subseteq B$ with

$$|Q| \geq k, |R| \geq \sqrt{n}$$

such that every pair of vertices $q \in Q, r \in R$ is connected by an edge.

**Proof.** Pick a uniformly random subset $Q \subseteq A$ of size $k$, and let $R$ be all the common neighbors of $Q$. Given any vertex $b \in B$ that has degree $d \geq k$, the probability that $b$ is included in $R$ is exactly

$$\frac{\binom{d}{k}}{\binom{em}{k}} \geq \left( \frac{d/k}{em/k} \right)^k = \left( \frac{d}{em} \right)^k.$$

Let $d_b$ be the degree of the vertex $b \in B$. The expected size of the set $R$ is at least

$$\mathbb{E} \left[ |R| \right] \geq \sum_{b \in B, d_b \geq k} \left( \frac{d_b}{em} \right)^k \geq n \cdot \left( \frac{1}{n} \sum_{b \in B, d_b \geq k} \frac{d_b}{em} \right)^k = n \cdot \left( \frac{1}{emn} \sum_{b \in B, d_b \geq k} d_b \right)^k.$$

By applying Jensen’s inequality to the function $x^k$. Since $(\frac{x}{c})^k \leq (\frac{x}{c})^k$ for $k > 0$.
Observe that $\sum_{b \in B, d_b \geq k} d_b$ counts all the edges of the graph, except those that touch vertices of degree less than $k$, so this quantity is at least $emn - kn \geq emn/2$, since $\epsilon \geq 2k/m$. Thus,

$$E[|R|] \geq n \cdot \left(\frac{1}{emn} \cdot \frac{emn}{2}\right)^k = n \cdot \left(\frac{\epsilon}{2e}\right)^k \geq \sqrt{n}.$$ 

So there must be some choice of $Q, R$ that proves the lemma. \hfill \square

**Lower Bounds for Inner-Product**

Say Alice and Bob are given $x, y \in \{0, 1\}^n$ and want to compute

$$IP(x, y) = \langle x, y \rangle \mod 2.$$

We have seen that this requires $n + 1$ bits of communication using a deterministic protocol. Here we show that it requires at least $\approx n/2$ bits of communication even using a randomized protocol. To prove this, we shall prove that the discrepancy of the inner product function with respect to the uniform distribution is exponentially small, for every rectangle.

**Lemma 5.5.** For any rectangle $R$, the discrepancy of inner product with respect to $R$ and the uniform distribution is at most $2^{-n/2}$.

**Proof.** Since $R$ is a rectangle, we can write its characteristic function as the product of two functions $A : \{0, 1\}^n \rightarrow \{0, 1\}$ and $B : \{0, 1\}^n \rightarrow \{0, 1\}$. Thus we can write:

$$\left(\mathbb{E}_{x, y} [\chi_R(x, y) \cdot (-1)^{\langle x, y \rangle}]\right)^2 = \left(\mathbb{E}_{x, y} [A(x) \cdot B(y) \cdot (-1)^{\langle x, y \rangle}]\right)^2$$

$$= \left(\mathbb{E}_x [A(x)] \mathbb{E}_y [B(y) \cdot (-1)^{\langle x, y \rangle}]\right)^2$$

$$\leq \mathbb{E}_x \left[A(x)^2 \left(\mathbb{E}_y [B(y) \cdot (-1)^{\langle x, y \rangle}]\right)^2\right].$$

by Jensen’s inequality applied to the function $z^2$.

Since $0 \leq A(x) \leq 1$, we can drop $A(x)$ from this expression to get:

$$\left(\mathbb{E}_{x, y} [\chi_R(x, y) \cdot (-1)^{\langle x, y \rangle}]\right)^2 \leq \mathbb{E}_x \left[\left(\mathbb{E}_y [B(y) \cdot (-1)^{\langle x, y \rangle}]\right)^2\right]$$

$$= \mathbb{E}_{x, y, y'} [B(y)B(y') \cdot (-1)^{\langle x, y \rangle + \langle x, y' \rangle}]$$

$$= \mathbb{E}_{x, y, y'} [B(y)B(y') \cdot (-1)^{\langle x, y + y' \rangle}] .$$
In this way, we have completely eliminated the dependence on the set $A$ from the calculation! We can also eliminate the set $B$ too and write:

$$\left( \mathbb{E}_{x,y} \left[ \chi_R(x,y) \cdot (-1)^{|x \cdot y|} \right] \right)^2 \leq \mathbb{E}_{x,y,y'} \left[ B(y)B(y') \cdot (-1)^{|x \cdot y + y'|} \right] \leq \mathbb{E}_{y,y'} \left( \mathbb{E}_x \left[ (-1)^{|x \cdot y + y'|} \right] \right).$$

Now, whenever $y + y'$ is not 0 modulo 2, the expectation over $x$ is 0. On the other hand, the probability that $y + y'$ is 0 modulo 2 is exactly $2^{-n}$. So

$$\mathbb{E}_{y,y'} \left( \mathbb{E}_x \left[ (-1)^{|x \cdot y + y'|} \right] \right) = 2^{-n},$$

proving the lemma.

Lemma 5.5 and Theorem 5.2 together imply:

**Theorem 5.6.** Any 2-party protocol that computes the inner product function with error at most $\epsilon$ over the uniform distribution must have communication at least $n/2 - \log(1/(1 - 2\epsilon))$.

Similar ideas can be used to show that the communication complexity of the generalized inner product must be large in the number-on-forehead model\(^1\). Here each of the $k$ players is given a binary string $x_i \in \{0, 1\}^n$. They want to compute

$$GIP(x) = \sum_{j=1}^{k} \prod_{i=1}^{n} x_{i,j} \mod 2.$$ 

We can show:

**Lemma 5.7.** The discrepancy of the generalized inner product function with respect to the uniform distribution and any cylinder intersection is at most $e^{-n/4k^{k-1}}$.

**Proof.** Let $S$ be a cylinder intersection. Then its characteristic function can be expressed as the product of $k$ 0/1 valued functions $\chi_S = \prod_{i=1}^{k} \chi_i$, where $\chi_i$ does not depend on $x_i$. Thus we can write:

$$\left( \mathbb{E}_x \left[ \chi_S(x) \cdot (-1)^{GIP(x)} \right] \right)^2 = \left( \mathbb{E}_{x_1,\ldots,x_{k-1}} \left[ \chi_k(x) \mathbb{E}_{x_k} \left[ \prod_{i=1}^{k-1} \chi_i(x) \cdot (-1)^{GIP(x)} \right] \right] \right)^2 \leq \mathbb{E}_{x_1,\ldots,x_{k-1}} \left[ \chi_k(x)^2 \left( \mathbb{E}_{x_k} \left[ \prod_{i=1}^{k-1} \chi_i(x) \cdot (-1)^{GIP(x)} \right] \right)^2 \right].$$

\(^1\) Babai et al., 1989

Each vector $x_i$ can be interpreted as a subset of $[n]$. The protocol for computing the set intersection size gives a protocol for computing generalized inner product with communication $O(k^4(1 + n/2^k))$. 

by Jensen’s inequality applied to the function $z^2$. 

Now we can drop $\chi_k(x)$ from this expression to get:

\[
\left( \mathbb{E}_x \left[ \chi_S(x) \cdot (-1)^{GIP(x)} \right] \right)^2
\leq \mathbb{E}_{x_1,\ldots,x_{k-1}} \left[ \mathbb{E}_{x_k} \left[ \prod_{i=1}^{k-1} \chi_i(x) \cdot (-1)^{GIP(x)} \right]^2 \right]
= \mathbb{E}_{x_1,\ldots,x_k,x'_k} \left[ \prod_{i=1}^{k-1} \chi_i(x) \chi_i(x') \cdot (-1)^{\sum_{j=1}^{n} x_i + x'_i} \prod_{i=1}^{k-1} x_i \right],
\]

where here $x'_k$ is uniformly distributed and independent of $x_1,\ldots,x_k$, and $x' = (x_1,\ldots,x_{k-1},x'_k)$. In this way, we have completely eliminated the function $\chi_k$ from the calculation! Repeating this trick $k-1$ times gives the bound

\[
\left( \mathbb{E}_x \left[ \chi_S(x) \cdot (-1)^{GIP(x)} \right] \right)^{2^{k-1}}
\leq \mathbb{E}_{x_2,x'_2,\ldots,x_k,x'_k} \left[ \mathbb{E}_{x_1} \left[ (-1)^{\sum_{j=1}^{n} x_i + x'_i} \prod_{i=1}^{k-1} x_i \right] \right].
\]

Whenever there is a coordinate $j$ for which

\[
\prod_{i=2}^{k-1} (x_{ij} + x'_{ij}) = 1 \mod 2,
\]

the inner expectation is 0. When

\[
\prod_{i=2}^{k-1} (x_{ij} + x'_{ij}) = 0 \mod 2
\]

for all coordinates $j$, then the inner expectation is 1. On the other hand, the probability that $\prod_{i=2}^{k-1} (x_i + x'_i) = 0 \mod 2$ is exactly $(1 - 2^{-k+1})^n$. So we get

\[
\left( \mathbb{E}_x \left[ \chi_S(x) \cdot (-1)^{GIP(x)} \right] \right)^{2^{k-1}} \leq (1 - 2^{-k+1})^n \leq e^{-n/2^{k-1}},
\]

since $1 - x \leq e^{-x}$ for all $x$.

By Lemma 5.7 and Theorem 5.2:

**Theorem 5.8.** Any randomized protocol for computing the generalized inner product in the number-on-forehead model with error $\epsilon$ over the uniform distribution requires at least $n/4^{k-1} - \log(1/(1 - 2\epsilon))$ bits of communication.

### Disjointness in the Number-on-Forehead Model

The discrepancy method may seem too weak to prove lower bounds against functions like disjointness, which do have large
monochromatic rectangles. Suppose Alice and Bob are given two sets $X, Y \subseteq [n]$ and want to compute disjointness. If we use a distribution on inputs that gives intersecting sets with probability at most $\epsilon$, then there is a trivial protocol with error at most $\epsilon$—the parties guess that the sets intersect without communicating. On the other hand, if the probability of intersection is at least $\epsilon$, then by averaging there must be some fixed coordinate $i$ such that an intersection occurs in coordinate $i$ with probability at least $\epsilon/n$. Setting $R = \{(X, Y) : i \in X, i \in Y\}$, we get

$$\mathbb{E} \left[ \chi_R(X, Y) \cdot ( -1)^{\text{Disj}(X, Y)} \right] = \mathbb{E} [\chi_R(X, Y)] \geq \epsilon / n.$$ 

So, the discrepancy method can only give a lower bound of $\Omega(\log n)$ if we follow the same approach that was used for the inner product function.

Nevertheless, one can use discrepancy to give a lower bound on the communication complexity of disjointness\(^2\), even when the protocol is allowed to be randomized, by studying the discrepancy of a function that is related to disjointness under a carefully chosen distribution.

Consider the following distribution on sets. Suppose that the universe consist of disjoint sets $I_1, \ldots, I_m$. Alice gets $m$ independently sampled sets $X_1, \ldots, X_m$, where $X_j$ is a uniformly random subset of $I_j$, and Bob gets $m$ independent random sets $Y_1, \ldots, Y_m$, each of size 1, where $Y_j \subset I_j$ for all $j$. Let $X = \bigcup_{j=1}^m X_j$ and $Y = \bigcup_{j=1}^m Y_j$. We start by proving a weak bound on the discrepancy:

**Lemma 5.9.** For any rectangle $R$,

$$\mathbb{E} \left[ \chi_R(X, Y) \cdot ( -1)^{\sum_{j=1}^m \text{Disj}(X_j, Y_j)} \right] \leq \sqrt{\frac{1}{\prod_{j=1}^m |I_j|}}.$$

**Proof.** As usual, we express $\chi_R(X, Y) = A(X) \cdot B(Y)$ and carry out a convexity argument. We get:

$$\left( \mathbb{E} \left[ \chi_R(X, Y) \cdot ( -1)^{\sum_{j=1}^m \text{Disj}(X_j, Y_j)} \right] \right)^2$$

$$= \left( \mathbb{E} \left[ A(X) \cdot B(Y) \cdot ( -1)^{\sum_{j=1}^m \text{Disj}(X_j, Y_j)} \right] \right)^2$$

$$\leq \mathbb{E} \left[ A(X)^2 \left( \mathbb{E} \left[ B(Y) \cdot ( -1)^{\sum_{j=1}^m \text{Disj}(X_j, Y_j)} \right] \right)^2 \right]$$

$$\leq \mathbb{E}_{X, Y, Y'} \left[ B(Y)B(Y') \cdot ( -1)^{\sum_{j=1}^m \text{Disj}(X_j, Y_j) + \text{Disj}(X_j, Y'_j)} \right]$$

$$\leq \mathbb{E}_{Y, Y'} \left[ \mathbb{E}_X \left[ ( -1)^{\sum_{j=1}^m \text{Disj}(X_j, Y_j) + \text{Disj}(X_j, Y'_j)} \right] \right],$$

by Jensen’s inequality applied to the function $z^2$.

If the probability of intersections is at most $\epsilon$, then the trivial rectangle $R = X \times Y$ has high discrepancy:

$$\left| \mathbb{E} \left[ \chi_R(X, Y) \cdot ( -1)^{\text{Disj}(X, Y)} \right] \right|$$

$$= |\Pr[X \cap Y = \emptyset] - \Pr[X \cap Y \neq \emptyset]|$$

$$\geq 1 - 2\epsilon.$$

\(^2\) Sherstov, 2012; and Rao and Yehudayoff, 2015
Previously, we used discrepancy to prove randomized lower bounds. Here we use it to prove a deterministic lower bound that does not apply to randomized protocols.

In fact, the argument we present here is stronger: it gives a lower bound on the size of a 1-cover for disjointness.
convexity argument to bound:

\[
\left( \mathbb{E} \left[ X_5(X) \cdot (-1)^{\sum_{j=1}^{m} \text{Disj}(X_{j,i}, X_{k,j})} \right] \right)^2 \\
\leq \mathbb{E}_{X_1, \ldots, X_{k-1}} \left[ \chi_k(X)^2 \cdot \left( \mathbb{E}_{X_k} \left[ \prod_{i=1}^{k-1} \chi_i(X) \cdot (-1)^{\sum_{j=1}^{m} \text{Disj}(T_j)} \right] \right)^2 \right] \\
\leq \mathbb{E}_{X_1, \ldots, X_{k-1}} \left[ \left( \mathbb{E}_{X_k} \left[ \prod_{i=1}^{k-1} \chi_i(X) \cdot (-1)^{\sum_{j=1}^{m} \text{Disj}(T_j)} \right] \right)^2 \right] \\
= \mathbb{E}_{X_1, \ldots, X_{k-1}, X_k, X_k'} \left[ \prod_{i=1}^{k-1} \chi_i(X) \chi_i(X') \cdot (-1)^{\sum_{j=1}^{m} \text{Disj}(T_j) + \text{Disj}(T_j')} \right],
\tag{5.1}
\]

where \( X_k' \) is an independent copy of \( X_k \) conditioned on \( X_1, \ldots, X_{k-1} \), \( X' = (X_1, \ldots, X_{k-1}, X_k') \) and \( T_j' = (X_{1,ji}, \ldots, X_{k-1,ji}, X_{k,ji}') \).

Now, let \( v_j \) denote the common intersection point of \( X_{2,ji}, \ldots, X_{k,ji} \), and let \( v'_j \) denote the common intersection point of \( X_{2,ji}, \ldots, X_{k-1,ji}, X_{k,ji}' \). Whenever \( v_j = v'_j \), we have \( \text{Disj}(T_j) = \text{Disj}(T_j') \), and so the \( j \)'th term of the sum is 0 modulo 2. On the other hand, when \( v_j \neq v'_j \), then any intersection in \( T_j \) must take place in the set \( X_{k,ji} - X_{k,ji}' \), and any intersection in \( T_j' \) must take place in \( X_{k,ji}' - X_{k,ji} \). So we can ignore the part of the sets \( X_1, \ldots, X_{k-1} \) in \( X_k \cap X_k' \) and in \( X_k' \cap X_k \) and use induction to bound the discrepancy as follows. Let \( Z_j \) be the random variable defined as

\[
Z_j = \begin{cases} 
1 & \text{if } v_j = v'_j, \\
\frac{(2^{k-2} - 1)^2}{\sqrt{|X_k - X_k'| \cdot |X_{k,ji} - X_{k,ji}'|}} & \text{otherwise.}
\end{cases}
\]

Since the \( Z_j \)'s are independent of each other, the inductive hypothesis gives:

\[
(5.1) \leq \mathbb{E} \left[ \prod_{j=1}^{m} Z_j \right] \leq \prod_{j=1}^{m} \mathbb{E} \left[ Z_j \right].
\]

We need a technical claim next:

**Claim 5.11.** Suppose a set \( Q \subseteq I_j \) is sampled by including a random element \( v \in I_j \) and adding every other element to \( Q \) independently with probability \( \gamma \neq 0 \). Then \( \mathbb{E} \left[ \frac{1}{|Q|} \right] \leq \frac{1}{\gamma |I_j|} \).

**Proof.**

\[
\mathbb{E} \left[ \frac{1}{|Q|} \right] = \sum_{Q, e} \frac{(1/|I_j|) \cdot \gamma |Q|^{-1}(1 - \gamma)^{|I_j| - |Q|}}{|Q|} \\
= \frac{1}{\gamma |I_j|} \sum_{Q \neq \emptyset} \gamma |Q| (1 - \gamma)^{|I_j| - |Q|} \leq \frac{1}{\gamma |I_j|} (1 - \gamma + \gamma |I_j|) = \frac{1}{\gamma |I_j|}.
\]
We now use the claim. To bound $\prod_{i=1}^{m} \mathbb{E} [Z_j]$. Conditioned on the value of $Q_j = X_{2,j} \cap \ldots \cap X_{k-1,j}$, the probability that $v_j = v_j'$ is exactly $1/|Q_j|$. Given the common intersection point, every element of $I_j$ is included in $Q_j$ independently with probability $\frac{1}{2^{k-1}}$. So by Claim 5.11,

$$\Pr[v_j = v_j'] = \mathbb{E} \left[ \frac{1}{|Q_j|} \right] \leq \frac{2^{k-1} - 1}{|I_j|}.$$ 

On the other hand, when $v_j \neq v_j'$, we can bound

$$Z_j = \frac{(2^{k-2} - 1)^2}{\sqrt{|X_{k,j} - X'_k|} \cdot |X'_{k,j} - X_{k,j}|} \leq \frac{(2^{k-2} - 1)^2}{2} \cdot \left( \frac{1}{|X_{k,j} - X'_k|} + \frac{1}{|X'_{k,j} - X_{k,j}|} \right).$$

by the arithmetic-mean geometric-mean inequality.

In this case, let $Q_j = X_{k,j} - X'_{k,j}$ Once again we see that $Q_j$ is sampled by picking the value of $v_j$ uniformly, and then every other element is included in $Q_j$ independently with probability $\frac{2^{k-2} - 1}{2^{k-1}}$. So using Claim 5.11 again,

$$\mathbb{E} \left[ \frac{1}{|X_{k,j} - X'_k|} \right] \leq \frac{2(2^{k-1} - 1)}{(2^{k-2} - 1)|I_j|}.$$ 

By symmetry, we also get:

$$\mathbb{E} \left[ \frac{1}{|X'_{k,j} - X_{k,j}|} \right] \leq \frac{2(2^{k-1} - 1)}{(2^{k-2} - 1)|I_j|}.$$ 

Combining these bounds, we get

$$\mathbb{E} [Z_j] \leq \Pr[v = v'] + \frac{(2^{k-2} - 1)^2}{2} \cdot \left( \frac{1}{|X_{k,j} - X'_k|} + \frac{1}{|X'_{k,j} - X_{k,j}|} \right) \leq \frac{2^{k-1} - 1}{|I_j|} + \frac{2(2^{k-2} - 1)(2^{k-2} - 1)^2}{(2^{k-2} - 1)|I_j|} = \frac{(2^{k-1} - 1)^2}{|I_j|}.$$ 

□

Lemma 5.10 can be used to prove a lower bound of $\Omega(n/4^k)$ on the number-on-forehead deterministic communication complexity of disjointness. Suppose a deterministic protocol for disjointness has communication $c$. Then there are at most $2^c$ monochromatic cylinder intersections $S_1, \ldots, S_T$ that cover all the 1's. Whenever $X_1, \ldots, X_k$ are disjoint, we have that $\sum_{j=1}^{m} \text{Disj}(X_{1,j}, X_{2,j}, \ldots, X_{k,j}) = m$. On the other hand, the probability that $X_1, \ldots, X_k$ are disjoint is exactly $2^{-m}$. Thus, in fact, it proves a lower bound of $2^{\Omega(n/4^k)}$ on the number of monochromatic cylinder intersections needed to cover the 1's of disjointness.
we get
\[ 2^{-m} \leq E \left[ \sum_{i=1}^{T} \chi_{S_i}(X) \cdot (-1)^{\sum_{j=1}^{m} \text{Disj}(X_{i,j}, \ldots, X_{k,j})} \right] \]
\[ \leq \sum_{i=1}^{T} \left| \mathbb{E} \left[ \chi_{S_i}(X) \cdot (-1)^{\sum_{j=1}^{m} \text{Disj}(X_{i,j}, \ldots, X_{k,j})} \right] \right| \]
\[ \leq 2^{c} \cdot \left( \prod_{j=1}^{m} \frac{2^{k-1} - 1}{\sqrt{|I_j|}} \right). \]

Setting \(|I_j| = 16 \cdot (2^{k-1} - 1)^2\) for all \(j\), we get that
\[ c \geq m = \frac{n}{16(2^{k-1} - 1)^2}. \]

**Theorem 5.12.** Any deterministic protocol for computing disjointness in the number-on-forehead model with \(k\) players over a universe of size \(n\) requires \(\frac{n}{16(2^{k-1} - 1)^2}\) bits of communication.

The best lower bound on the randomized communication complexity is given by the following theorem:\(^3\)

**Theorem 5.13.** Any randomized protocol for computing disjointness in the worst case with error \(1/3\) in the number-on-forehead model with \(k\) players over a universe of size \(n\) requires \(\Omega\left(\sqrt{\frac{n}{k^2}}\right)\) bits of communication.

Theorem 5.13 is proved using discrepancy estimates as above, combined with ideas from approximation theory.

**Using Concentration Bounds to Control Discrepancy**

Several tools from probability theory can be used to control discrepancy. Here we explore two examples: one using Chernoff-Hoeffding bounds and the other using Talagrand’s inequality.

**The Randomized Communication Complexity of Disjointness**

We start by trying to prove lower bounds on the randomized communication complexity of 2-party protocols computing the disjointness function. We have already discussed a major obstacle for this approach—for any distribution on inputs there is a rectangle with large discrepancy. Nevertheless, we can prove some lower bounds by estimating the discrepancy of disjointness with respect to rectangles that mostly contain disjoint inputs.

We shall prove:\(^4\)

\(^3\) Sherstov, 2014; and Rao and Yehudayoff, 2015

\(^4\) Babai et al., 1986

Later on we shall prove a sharp \(\Omega(n)\) lower bound on the randomized communication complexity of disjointness. We present this proof here for two reasons. First, it applies to a product distribution on the inputs, and is tight for this case. Second, it sets the stage for the lower bound on the Gap-Hamming problem that is explained in the next section.
**Theorem 5.14.** Any randomized 2-party protocol computing disjointness with error $1/3$ must have communication $\Omega(\sqrt{n})$.

To prove this theorem, we start by defining a hard distribution on inputs. For a parameter $\gamma$, we independently sample sets $X, Y \subseteq [n]$ by including each element in each set independently with probability $\gamma$. We set $\gamma \approx 1/\sqrt{n}$ so that the probability that the sets are disjoint is exactly

$$
(1 - \gamma^2)^n = \frac{1}{2}.
$$

The heart of the proof is the following lemma, which shows that we cannot have a large rectangle with too many disjoint pairs of inputs:

**Lemma 5.15.** There are constants $0 < \alpha, \beta < 1$ such that for any rectangle $R$ so that

$$
\Pr[(X, Y) \in R] \geq e^{-\alpha\sqrt{n}},
$$

we have

$$
\Pr[X, Y \text{ are disjoint} \mid (X, Y) \in R] < 1 - \beta.
$$

The lemma implies the theorem:

**Proof of Theorem 5.14.** We can always reduce the error of the protocol by repeating it a constant number of times, so suppose for the sake of contradiction that there is a protocol with communication $c$ that computes disjointness with error at most $e$, for some small constant $e$. Since the probability that $X, Y$ are disjoint is $1/2$, the protocol must output that the sets are disjoint with probability at least $1/2 - e$.

Theorem 3.6 implies that there is a rectangle of density $\sqrt{e} \cdot 2^{-e}$ that consists almost entirely of disjoint inputs. By Lemma 5.15, any such rectangle must have density at most $e^{-\alpha\sqrt{n}}$, proving that $c$ must be at least $\Omega(\sqrt{n})$. 

To prove Lemma 5.15, we repeatedly use the Chernoff-Hoeffding bound to find a useful subset of the rectangle:

**Proof of Lemma 5.15.** We shall set $\alpha, \beta$ to be small enough constants during the proof. Let $R = A \times B$ be any rectangle of density $e^{-\alpha/\sqrt{n}}$. Let $\beta = \Pr[X, Y \text{ are intersecting} \mid (X, Y) \in R]$. Define:

$$
A' = \{x \in A : \Pr_Y[x, Y \text{ are intersecting} \mid Y \in B] < 2\beta\}.
$$

By Markov’s inequality,

$$
\Pr[X \in A'] \geq \Pr[X \in A'](X, Y) \in R \cdot \Pr[(X, Y) \in R]
\geq (1/2) \cdot \Pr[(X, Y) \in R] \geq e^{-\alpha/\sqrt{n}}/2.
$$

For $\gamma \leq 1/2$,

$$
2^{-2\gamma^2 n} \leq (1 - \gamma^2)^n \leq e^{-\gamma^2 n}.
$$

So, we have the estimates:

$$
\frac{1}{\sqrt{2n}} \leq \gamma \leq \frac{\ln 2}{\sqrt{n}}.
$$

Can you think of a rectangle that proves that the lemma is tight?

Figure 5.2: The set $A'$ contains many sets that do not intersect each other too much.

Figure 5.3: There will be a set in $B$ that intersects many of the sets $X_1, \ldots, X_\ell$. 

---

**DISCREPANCY 97**
Claim 5.16. Let $\ell = \lceil \frac{1}{\eta} \rceil$. If $\alpha$ is small enough, there are sets

$$X_1, X_2, \ldots, X_\ell \in A'$$

such that for all $i$,

$$\frac{1}{2} \cdot \gamma n \leq |X_i| \leq \frac{3}{2} \cdot \gamma n,$$

and

$$|X_i - \bigcup_{j=1}^{i-1} X_j| \geq \frac{\gamma n}{4}.$$

Proof. We find the sequence of sets $X_1, \ldots, X_\ell$ inductively. In the $i$’th step, consider the experiment of picking $X_i$ according to the distribution where each element is included in $X_i$ independently with probability $\gamma$. Note that this distribution does not depend on $A'$. The expected size of $X_i$ is $\gamma n$. So by the Chernoff-Hoeffding bound,

$$\Pr[||X_i| - \gamma n| > \gamma n/2] \leq 2e^{-(1/2)^2 \gamma n/3}.$$

The size of $\bigcup_{j=1}^{i-1} X_j$ is at most $\frac{1}{\eta} \cdot \frac{3 \gamma n}{2} = \frac{n}{6}$. So if $\mu$ is the expected number of elements in $X_i$ that are not in $\bigcup_{j=1}^{i-1} X_j$, we have $\mu \geq \gamma 5n/6$. So, the probability that fewer than $\gamma n/4$ elements are in $X_i$ but not in the union is at most

$$\Pr\left[\left|X_i - \bigcup_{j=1}^{i-1} X_j\right| < \gamma n/4 \right] \leq e^{-(\frac{2\mu}{\gamma n} - \frac{1}{4})^2 \gamma n/18}.$$

We set $\alpha$ to be small enough so that

$$\Pr[X \in A'] > 2e^{-(1/2)^2 \gamma n/3} + e^{-(\frac{2\mu}{\gamma n} - \frac{1}{4})^2 \gamma n/18}.$$

This ensures that there is some $X_i \in A$ with the claimed properties.

Let $X_1, \ldots, X_\ell$ be as promised by Claim 5.16, and for each $i$, define

$$Z_i = X_i - \bigcup_{j=1}^{i-1} X_j.$$

The sets $Z_1, \ldots, Z_\ell$ are disjoint, and each is of size at least $\gamma n/4$.

Now, assume towards a contradiction that $\beta < (1 - e^{-1/8})/8$.

Define

$$B' = \{y \in B : y \text{ intersects at most } 4\beta \ell \text{ of the sets } Z_1, \ldots, Z_\ell\}.$$

On the one hand, if we pick $Y$ at random by including each element in $Y$ with probability $\gamma$, the probability that $Y$ is disjoint from a specific $Z_i$ is at most

$$1 - (1 - \gamma)^{\gamma n/4} \leq e^{-\gamma^2 n/4} \leq e^{-1/8}.$$
So, the expected number of the sets $Z_1, \ldots, Z_\ell$ that $Y$ intersects is at least $\ell(1 - e^{-1/8})$. Applying the Chernoff-Hoeffding bound once more, we get that
\[
\Pr[Y \in B'] \leq e^{-\left((1/2)\ell(1-e^{-1/8})/3 \right)} < e^{-\alpha n/2},
\]
if we choose $\alpha$ to be a small enough constant.

On the other hand, by the definition of $A'$, a random element $Y \in B$ intersects less than $2\beta\ell$ of the sets $Z_1, \ldots, Z_\ell$ in expectation. Thus, the probability that $Y$ intersects at least $4\beta\ell$ fraction of the sets is at most $1/2$ by Markov’s inequality. We finally get a contradiction:
\[
\Pr[Y \in B'] = \Pr[Y \in B'] \cdot \Pr[Y \in B] \geq (1/2) \cdot \Pr[(X, Y) \in R] > e^{\alpha n/2}.
\]

\[\square\]

**The Gap-Hamming Problem**

Although the bounds we obtained for the randomized communication complexity of disjointness are not tight, a similar approach gives tight bounds for the Gap-Hamming problem. Suppose Alice and Bob have inputs $x, y \in \{+1, -1\}^n$. They wish to estimate the inner product $\langle x, y \rangle$ of the two strings. This task is equivalent to computing the Hamming distance between $x$ and $y$, defined to be $|\{i \in [n] : x_i \neq y_i\}|$. Indeed, the Hamming distance is exactly $n - \langle x, y \rangle$.

The fooling set method can be used to show that the deterministic communication complexity of this problem is $\Omega(n)$. In Chapter 3, we showed that there is a randomized protocol that can estimate the hamming distance up to an additive factor of $\sqrt{n}/\epsilon$, with communication $O(\epsilon^2 n)$. Here we prove that this protocol is essentially the best we can hope for.

We say that a randomized protocol $\pi$ approximates inner-product up to a parameter $m$ if for every $x, y$, we have $\Pr[|\langle x, y \rangle - \pi(x, y)| > m] \leq 2/3$.

We shall prove $^5$:

**Theorem 5.17.** Any randomized protocol that approximates inner-product up to $\sqrt{n}$ must have communication complexity $\Omega(n)$.

Let $X, Y \in \{+1, -1\}^n$ be independent and uniformly random. As in the lower bound for disjointness, the key step in the argument is to prove that there are no large rectangles where the magnitude of the inner product is small.

The lower bound follows from the following lemma:

---

Lemma 5.18. There are constants $0 < \alpha, \beta < 1$ and an integer $t$ such that if $R$ is a rectangle with
\[
\Pr[(X, Y) \in R] > 2^{-\alpha n},
\]
then
\[
\Pr \left[ |\langle X, Y \rangle| \leq \sqrt{n}/t \mid (X, Y) \in R \right] \leq 1 - \beta.
\]

The proof strategy of Lemma 5.18 is similar to that of Lemma 5.15 used to prove the lower bound for the disjointness function. However, the proof of Lemma 5.15 is more elaborate and relies on a beautiful result from geometry and probability called Talagrand’s inequality, as well as the singular value decomposition of matrices. Before discussing how to use these ideas to prove the lemma, let us first show how to use the lemma to prove the theorem.

Proof of Theorem 5.17. We start by increasing the confidence of the protocol by repetition. Let $\ell = 9t^2 n$, where $t$ is as in Lemma 5.18. Assume towards a contradiction that there is a protocol $\pi$ with communication complexity $o(\ell)$ that approximates the inner product of $x', y' \in \{+1, -1\}^\ell$ up to $\sqrt{\ell}$. By repeating the protocol and taking the majority outcome, we obtain a protocol with communication $o(\ell)$ that approximates inner-product up to $\sqrt{\ell}$, and makes an error with probability $o(1)$.

Now, for inputs $x, y \in \{+1, -1\}^n$, Alice and Bob can repeat each coordinate of their input $9t^2$ times to obtain inputs $x', y' \in \{+1, -1\}^\ell$, and run the protocol $\pi$ on these inputs. We have $\langle x', y' \rangle = 9t^2 \cdot \langle x, y \rangle$, so if the estimated inner product is $z$, and the protocol does not make an error, we have:
\[
3t \sqrt{n} = \sqrt{\ell} \geq |z - \langle x', y' \rangle| = 9t^2 \cdot \frac{z}{9t^2} - \langle x, y \rangle,
\]
so $|\frac{z}{9t^2} - \langle x, y \rangle| \leq \sqrt{n}/3t$. Alice and Bob compute $z/9t^2$ to obtain a good estimate for the inner product.

There is a significant probability that the inner product will have a magnitude that is at most $\sqrt{n}/3t$. To see this, let $Z_1, \ldots, Z_n$ be bits defined by
\[
Z_i = \begin{cases} 1 & \text{if } X_i \neq Y_i, \\ 0 & \text{otherwise.} \end{cases}
\]
Then $Z = \sum_{i=1}^n Z_i = \frac{n - \langle X, Y \rangle}{2}$. Moreover, the expected value of $Z$ is $n/2$. So, by the Chernoff-Hoeffding bound,
\[
\Pr[|\langle X, Y \rangle| > \sqrt{n}/3t] = \Pr \left[ \left| Z - \frac{n}{2} \right| > \sqrt{n}/6t \right] \leq e^{-\frac{t^2}{3}}.
\]

This means that the protocol must compute an outcome of magnitude at most $\sqrt{n}/3t$ with probability at least $1 - e^{-\frac{t^2}{3}} - o(1)$. So by
Theorem 3.6, there must be a rectangle $R$ of density at least $2^{-o(n)}$, where the protocol computes an estimate of magnitude at most $\sqrt{n}/3t$, and yet makes an error with probability $o(1)$. Lemma 5.18 implies that this is impossible.

It only remains to prove Lemma 5.18. The first technical tool we need is called Talagrand’s inequality. Talagrand’s inequality concerns the length of the projection of a uniformly random vector $X \in \{+1, -1\}^n$ to a given vector space $V$. If $V$ is a $d$-dimensional vector space and $x \in \mathbb{R}^n$, let $\text{proj}_V(x) \in V$ denote the projection of $x$ to $V$. Namely it is the vector in $V$ such that $\text{proj}_V(x)$ and $x - \text{proj}_V(x)$ are orthogonal. Figure 5.4 shows various ways of projecting the cube $\{+1, -1\}^n$ to different 2-dimensional subspaces.

The expected value of the square of the length of the projection $\|\text{proj}_V(x)\|_2^2$ is $d$: if $e_1, e_2, \ldots, e_d$ is an orthonormal basis for $V$, then

$$E \left[ \|\text{proj}_V(x)\|_2^2 \right] = E \left[ \sum_{i=1}^d (x, e_i)^2 \right] = \sum_{i=1}^d E \left[ (x, e_i)^2 \right],$$

and for each $e_i$,

$$E \left[ (x, e_i)^2 \right] = E \left[ \left( \sum_{j=1}^n e_{ij}x_j \right)^2 \right] = \sum_j E \left[ (e_{ij}x_j)^2 \right] + \sum_{j \neq f} E \left[ (e_{ij}x_j)(e_{if}x_f) \right] = \sum_{j=1}^n E \left[ e_{ij}^2 \right] = ||e_i||^2 = 1.$$

This might lead us to guess that length of the projection should typically be about $\sqrt{d}$. Talagrand’s inequality shows that the length of the projection is concentrated around this quantity:

**Theorem 5.19.** There is a constant $\gamma > 0$ such that for any $d$-dimensional vector space $V \subseteq \mathbb{R}^n$,

$$\Pr \left[ ||\text{proj}_V(x)|| - \sqrt{d} \geq s \right] < 4e^{-\gamma s^2},$$

where $x$ is uniformly distributed in $\{+1, -1\}^n$.

Intuitively Talagrand’s inequality suggests that a statement like Lemma 5.18 ought to hold. If $R = A \times B$ is not exponentially small, then for $k = \Omega(n)$, one should be able to find vectors $x_1, \ldots, x_k \in A$ that are essentially orthogonal to each other—we can choose $x_1, \ldots, x_k$ iteratively, since a uniformly random vector $x_i \in \{+1, -1\}^n$ will have a small projection onto the previous vectors except with exponentially small probability, so the probability that it falls in $A$ and has small projection onto the previous vectors will be positive.
Now if $x_1, \ldots, x_k$ were perfectly orthogonal, we could apply Talagrand’s inequality once again to prove the lemma. Consider the experiment of picking a uniformly random $Y$. Since $Y$’s projection to the space spanned by $x_1, \ldots, x_k$ must be of length about $\sqrt{k} \approx \sqrt{n}$ by Talagrand’s inequality, we must have $|\langle y, x_i \rangle| > \sqrt{k}$ for most coordinates $i$, except with exponentially small probability. Since $B$ is not exponentially small, the probability that $Y$ is in $B$ and still has the above properties is significant.

Of course, $x_1, \ldots, x_k$ need not be perfectly orthogonal, so to turn these intuitions into a proof, we need to use the concept of singular value decompositions—SVD for short—of a matrix. Let $M$ be an arbitrary $m \times n$ matrix with real valued entries and $m \leq n$. Then, one can always express

$$ M = \sum_{i=1}^{m} \sigma_i \cdot u_i v_i^\top, $$

where $u_1, \ldots, u_m$ are orthogonal $m \times 1$ column vectors with $\|u_i\| = 1$, $v_1, \ldots, v_m$ are orthogonal $1 \times n$ column vectors with $\|v_i\| = 1$, and $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_m \geq 0$ are real numbers called the singular values of $M$.

The singular value decomposition gives a a nice way to interpret

![Figure 5.4: Four examples showing projections of the set $\{+1, -1\}^{15}$ onto different 2-dimensional vector spaces. The circles are centered at the origin and have radius $\sqrt{2}$.](image)

If $m > n$, one can apply the decomposition to the transpose of $M$ to get a similar statement.

The vectors $u_1, \ldots, u_m$ are the eigenvalues of the symmetric matrix $MM^T$, the vectors $v_1, \ldots, v_m$ are the eigenvectors of $M^T M$, and the singular values are the square roots of the eigenvalues of both $MM^T$ and $M^T M$. 
the action of $M$ on a $n \times 1$ column vector $y$:

$$My = \sum_{i=1}^{m} \sigma_i u_i v_i^T y = \sum_{i=1}^{m} \sigma_i \langle v_i, y \rangle \cdot u_i.$$  

In words, $My$ can be viewed as the result of scaling and rotating $y$. To compute $My$, express $y$ in the basis given by $v_1, \ldots, v_m$ and scale the coefficients of $y$ using the singular values. Then rotate the result to the basis given by $u_1, \ldots, u_m$.

The singular values characterize how much the matrix $M$ can stretch an $n \times 1$ column vector $y$. Indeed, the properties of the decomposition allow us to conclude:

$$\|My\|^2 = y^T M^T My$$

$$= y^T \left( \sum_{i=1}^{m} \sigma_i \cdot v_i u_i^T \right) \left( \sum_{j=1}^{m} \sigma_j \cdot u_j v_j^T \right) y$$

$$= y^T \left( \sum_{i=1}^{m} \sigma_i^2 \cdot v_i v_i^T \right) y = \sum_{i=1}^{m} \sigma_i^2 \cdot (y^T v_i)^2. \quad (5.2)$$

Moreover, convexity implies that $\|My\| \leq \sigma_1 \cdot \|y\|$, and thus for a given value of $\|y\|$, the length $\|My\|$ is maximized when $y$ is proportional to $v_1$.

**Proof of Lemma 5.18.** For simplicity of notation, in this proof, we assume that several expressions of the form $\delta n$ for some $\delta$ are integers.

Let $R = A \times B$ be the given rectangle. We shall set $\alpha, \beta, t$ as needed in the proof. We assume towards a contradiction that $R$ has density at least $2^{-\alpha n}$, and yet a uniformly random $(X, Y)$ in $R$ satisfy $|\langle X, Y \rangle| \leq \sqrt{n}/t$ with probability $1 - \beta$ for $\beta$ a small constant.

Define:

$$A' = \left\{ x \in A : \Pr_{Y \in B}[|\langle x, Y \rangle| > \sqrt{n}/t] \leq 2\beta \right\}.$$  

By Markov’s inequality, we must have that

$$\Pr[X \in A'] \geq \frac{1}{2} \cdot \Pr[(X, Y) \in R] \geq 2^{-\alpha n - 1}.$$  

Now, use Talagrand’s inequality to find a set of nearly orthogonal vectors in $A'$:

**Claim 5.20.** If $k = \frac{n}{16}$, there are strings $x_1, x_2, \ldots, x_k \in A'$ such that for all $i$, if $V_i$ denotes the span of $x_1, \ldots, x_i$, then

$$\|\text{proj}_{V_{i-1}}(x_i)\| \leq \frac{\sqrt{n}}{2}.$$
Proof. We find the sequence \( x_1, \ldots, x_k \in \{ +1, -1 \}^n \) inductively.

In the \( i \)th step, consider the experiment of picking \( x_i \) according to the uniform distribution. The dimension of \( V_{i-1} \) is at most \( k \), so by Theorem 5.19, the probability that the length of the projection exceeds \( \sqrt{n}/2 \) is at most \( 4e^{-\gamma(\sqrt{n}/4)^2} \).

We set \( \alpha \) to be a small enough constant such that \( \Pr\left[ X \in A' \right] \geq 2 - \alpha n^{-1} \), to guarantee that there must be some \( x_i \in A' \) satisfying the requirement.

Let \( x_1, \ldots, x_k \) be as in the claim above. For each subset \( S \subseteq [k] \) of size \( |S| = m \) with \( m = k - 4\beta k \), we define the set

\[
B_S = \{ y \in B : |\langle x_i, y \rangle| \leq \sqrt{n}/t \text{ for all } i \in S \}.
\]

By definition of \( A' \) and Markov's inequality, at least half the \( y \)'s in \( B \) must satisfy the property that the number of coordinates \( i \) for which \( |\langle x_i, y \rangle| > \sqrt{n}/t \) is at most \( 4\beta k \). So, by averaging, there must be one set \( S \) for which

\[
\Pr[Y \in B_S] \geq \frac{2^{-an-1}}{\binom{n}{k}} = \frac{2^{-an-1}}{\binom{n}{4\beta k}}.
\] (5.3)

Without loss of generality, we may assume that \( \{1, 2, \ldots, m\} \) is such a set.

Now, we use the singular value decomposition. Define the \( m \times n \) matrix \( M \) whose rows are \( x_1, \ldots, x_m \), and write it as

\[
M = \sum_{i=1}^{m} \sigma_i \cdot u_i^T v_i,
\]

with \( \sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_m \geq 0 \). We claim that the largest singular values of \( M \) cannot be too different from each other, by establishing two bounds:

\[
\sum_{i=1}^{m} \sigma_i \geq m \sqrt{n}/2,
\] (5.4)

and

\[
\sum_{i=1}^{m} \sigma_i^2 = mn.
\] (5.5)

Let us see how to use these bounds to completely the proof of Lemma 5.18.

We claim that there must be at least \( m/16 \) singular values of
magnitude at least $\sqrt{n}/4$, since

$$\sum_{i>m/16} \sigma_i = \sum_{i=1}^m \sigma_i - \sum_{i=1}^{m/16} \sigma_i \geq \frac{m\sqrt{n}}{2} - \sqrt{\sum_{i=1}^{m/16} \sigma_i^2 \cdot \sqrt{m/16}}$$

by Cauchy-Schwartz and (5.4).

$$\geq \frac{m\sqrt{n}}{2} - \sqrt{m\cdot \sqrt{m/4}} = m\sqrt{n}/4.$$

So, $\sigma_{m/16+1} > \sqrt{n}/4$.

Now, let $V$ denote the span of $v_1,\ldots,v_{m/16}$. If $y \in \{+1,-1\}^n$ is uniformly random, then by Theorem 5.19,

$$\Pr[||\text{proj}_V(y)|| - \sqrt{m}/4 \geq \sqrt{m}/8] < 4e^{-\gamma m/64} < 2^{-an-1}/(\beta^k),$$

if $a, \beta$ are small enough. Thus, by (5.3) there must be $y \in B_S$ with $||\text{proj}_V(y)|| \geq \sqrt{m}/8$. But, on the other hand, by (5.2) and the definition of $B_S$,

$$\frac{mn}{t^2} \geq \sum_{i=1}^m \langle x_i, y \rangle^2 = ||My||^2 \geq \sum_{i=1}^{m/16} \langle y^T v_i \rangle^2 \geq \frac{n}{16} \sum_{i=1}^{m/16} \langle y^T v_i \rangle^2 = \frac{n}{16} \cdot ||\text{proj}_V(y)||^2,$$

which is a contradiction if $t \geq 32$.

It only remains to prove (5.4) and (5.5). To prove (5.4), let $z_1,\ldots,z_m$ be the orthogonal vectors obtained by setting $z_1 = x_1$, and for $i > 1$,

$$z_i = x_i - \text{proj}_{V_{i-1}}(x_i).$$

Now, let $Z$ be the matrix with rows $z_1,\ldots,z_m$. On one hand, we have:

$$\text{tr}(MZ^T) = \sum_{i=1}^m \langle x_i, z_i \rangle = \sum_{i=1}^m \langle x_i, x_i \rangle - \sum_{i=1}^m \langle x_i, \text{proj}_{V_{i-1}}(x_i) \rangle \geq mn - \sum_{i=1}^m ||x_i|| \cdot ||\text{proj}_{V_{i-1}}(x_i)|| \geq mn - mn \cdot \sqrt{n} \cdot \sqrt{n}/2 = mn/2.$$
On the other hand, we have
\[
\text{tr}(MZ^T) = \text{tr}((\sum_{i=1}^{m} \sigma_i \cdot u_i v_i^T)Z^T)
\]
\[
= \sum_{i=1}^{m} \sigma_i \cdot \text{tr}(u_i v_i^T Z^T)
\]
\[
= \sum_{i=1}^{m} \sigma_i \cdot \langle u_i, Z v_i \rangle \leq \sum_{i=1}^{m} \sigma_i \cdot \|u_i\| \cdot \|Z v_i\|.
\]
by Cauchy-Schwartz.

The rows of $Z$ are orthogonal, and of length at most $\sqrt{n}$, so we get:
\[
\|Z v_i\|^2 = v_i Z Z^T v_i^T = \sum_{j=1}^{n} v_i^2 j \|z_i\|^2 \leq \sum_{j=1}^{n} v_i^2 j n \leq n.
\]

Thus, we have
\[
\text{tr}(MZ^T) \leq \sqrt{n} \cdot \sum_{i=1}^{m} \sigma_i,
\]
which proves (5.4).

To prove (5.5), on one hand,
\[
\text{tr}(MM^T) = \sum_{i=1}^{m} x_i^T x_i = mn.
\]

On the other hand, it is the same as the trace
\[
\text{tr}\left(\sum_{i=1}^{m} \sigma_i^2 u_i u_i^T\right) = \sum_{i=1}^{m} \sigma_i^2 \cdot \text{tr}(u_i u_i^T) = \sum_{i=1}^{m} \sigma_i^2.
\]

Exercise 5.1

Use counting to bound discrepancy.

Exercise 5.2

Let $G = (V, E)$ be an $n$-vertex $d$-regular expander graph. That is, for every $A \subseteq V$ of size $|A| \leq n/2$ the number of edges of the form $\{a, b\} : a \in A, b \notin A$ is at least $|A|/10$. Define a version of the inner product function over $G$: for $z : V \to \{0, 1\}$ set
\[
f(z) = \sum_{\{v, u\} \in E} z_u z_v \mod 2.
\]

For a one-to-one map $\pi$ from the variables $\{x_1, \ldots, x_{n/2}\} \cup \{y_1, \ldots, y_{n/2}\}$ to the variables $\{z_v : v \in V\}$, define
\[
f_{\pi}(x, y) = f(\pi(x, y)).
\]

Prove: for every $\pi$ and for any rectangle $R$, the discrepancy of $f$ with respect to $R$ and the uniform distribution is at most $2^{-\Omega(n)}$. 

Exercise 5.3

Suppose you are given $n$ sets $S_1, S_2, \ldots, S_n \subseteq [n]$. Use the Chernoff-Hoeffding bound to show that there is an element $x \in \{1, -1\}^n$ such that the maximum discrepancy

$$\max_{i \in [n]} \left| \sum_{j \in S_i} x_j \right|$$

is at most $O(\sqrt{n \log n})$.

A more clever argument shows that there is an $x$ whose discrepancy as defined in this exercise is at most $O(\sqrt{n})$. 
6

Information

Not all messages convey the same amount of information to the receiver. Sometimes a conversation or message reveals a lot of new information to the listener, and sometimes the conversation is almost predictable. But what is the best way to mathematically quantify how much information is conveyed by a given message or conversation? This is the motivating question of this chapter.

In Shannon's seminal work\(^1\), he defined the notion of entropy, which has had a big impact on many areas in mathematics. Shannon's goal was to quantify the amount of information or entropy contained in a random variable \(X\). His definition leads to a theory that is both elegant and useful in many areas, including communication complexity. We begin this chapter with some simple examples that help to demonstrate the usefulness of this theory. Later, we show how these concepts can be used to understand communication complexity.

Entropy

In many cases, the amount of information contained in a message is not the same as the length of the message. Here are some examples:

- Suppose Alice's first message to Bob is a \(c\)-bit string that is always 0, no matter what her input is. This message does not convey any information to Bob. Alice and Bob may as well imagine that this first message has already been sent, and so reduce the communication of the first step to 0.

- Suppose Alice's first message to Bob is a uniformly random string from a set \(S \subseteq \{0, 1\}^c\), with \(|S| \ll 2^c\). In this case, the parties could use \(\log |S|\) bits to index the elements of the set, reducing the communication from \(c\) to \(\log |S|\).

\(^1\) Shannon, 1948
• Suppose Alice’s first message to Bob is the string $0^n$ with probability $1 - \epsilon$, and is a uniformly random $n$ bit string with the probability $\epsilon$. In this case, one cannot encode every message using fewer than $n$ bits. However, Alice can send the bit 0 to encode the string $0^n$, and the string $1x$ to encode the $n$ bit string $x$. Although the first message is still quite long in the worst case, the expected length of the message is $1 - \epsilon + \epsilon(n + 1) = 1 + \epsilon n \ll n$.

Shannon’s definition of entropy gives a general way to compute the optimal encoding length for messages. Given a random variable $X$ with probability distribution $p(x)$, the entropy of $X$ is defined to be

$$H(X) = \sum_x p(x) \cdot \log \frac{1}{p(x)} = \mathbb{E}_{p(x)} \left[ \log \frac{1}{p(x)} \right],$$

This definition may seem technical at first sight, but as we shall see, it enjoys some simple, intuitive and useful properties. One simple property is that the entropy is always non-negative, since every term in the sum is non-negative.

Suppose $X$ is a uniformly random element of $[n]$. Then its entropy is

$$H(X) = \sum_{x \in [n]} \frac{1}{n} \log n = \log n.$$

In fact, the uniform distribution has the maximum entropy of any distribution on a finite set: if $X \in [n]$ then

$$H(X) = \mathbb{E}_{p(x)} \left[ \log \frac{1}{p(x)} \right] \leq \log \mathbb{E}_{p(x)} \left[ \frac{1}{p(x)} \right] = \log n,$$

where the inequality follows by Jensen’s inequality applied to the convex function log. This property of the entropy function makes it particularly useful as a tool to count the size of sets, since it relates entropy to the size of a set.
An Axiomatic Definition

A fundamental property of the entropy function is that it can be axiomatically defined. Let $H(X)$ be any notion of the entropy of a random variable $X$. Then it is natural to require this notion of entropy to satisfy the following axioms:

- **Symmetry** $H(\pi(X)) = H(X)$ for all bijections $\pi$. Intuitively, an invertible transformation should not change the amount of information in $X$.

- **Continuity** $H$ should be continuous in the distribution of $X$. Intuitively, an infinitesimally small change to the distribution of $X$ should result in an infinitesimally small change in the information contained in $X$.

- **Monotonicity** If $X$ is uniform over a set of size $n$, then $H(X)$ increases as $n$ increases. This axiom ensures that larger strings have higher entropy.

- **Chain-Rule** If $X = (Y, Z)$ then
  
  $$H(X) = H(Y) + \sum_y p(Y = y) \cdot H(Z|Y = y).$$

  This axiom asserts that the entropy of a random variable $X$ that consists of $Y$ and $Z$ equals the entropy of $Y$ plus the expected entropy of $Z$ given that we know $Y$. Intuitively $X$ can be described by describing $Y$ and then describing $Z$ after $Y$ has been determined.

Shannon proved that any notion of entropy satisfying these axioms must be proportional to the entropy function that he defined.

Several other axiomatic definitions of the entropy are known. For example, Gromov gives an axiomatic definition that extends to a quantum notion of entropy.

Coding

Shannon showed that the entropy of $X$ characterizes the expected number of bits that need to be transmitted to encode $X$. We start with an intuitive explanation for why this should be true, and then give a more rigorous proof. On the one hand, if there is an encoding of $X$ that has expected length $k$, then $X$ can be encoded by a string of length at most $10^k$ most of the time. So, one would expect that $X$ takes one of $2^{O(k)}$ values most of the time, and thus the expected value of $\log(1/p(x))$ should be at most $O(k)$. On the other hand, if $H(X) = k$, we may assume that $X \in [n]$ and $p(1) \geq p(2) \geq \cdots \geq p(n)$. Since $\sum_{j=1}^{n} p(j) \leq 1$ for all $i$, it follows that $p(i) \leq 1/i$. Writing the

\[\log(1/p(x)) \leq O(k)\]

by Markov’s inequality.
integer $i$ takes about $\log i$ bits, so the expected length of the encoding should be about $\sum_{i \in [n]} p(i) \cdot \log i \leq H(X)$. Formally, we can prove $^4$

### Theorem 6.1

The random variables $X$ can be encoded using a message whose expected length is at most $H(X) + 1$. Conversely, every encoding of $X$ has expected length at least $H(X)$.

**Proof.** Without loss of generality, suppose that $X$ is an integer from $[n]$, with $p(i) \geq p(i + 1)$ for all $i$. Let $\ell_i = \lceil \log(1/p(i)) \rceil$. To prove that $X$ can be encoded using messages of length $H(X) + 1$, we shall construct a protocol tree for Alice to send $X$ to Bob. Each $i$ in the tree will correspond to a vertex $v_i$ at depth $\ell_i$. The expected length of the message will thus be

$$\sum_{i} p(i) \cdot \ell_i \leq \sum_{i} p(i)(\log(1/p(i)) + 1) = H(X) + 1.$$  

The encoding is done greedily. In the first step, we pick a vertex $v_1$ of the complete binary tree at depth $\ell_1$ and let that vertex represent 1. We delete all of $v_1$’s descendants, so that $v_1$ becomes a leaf. Next, we find an arbitrary vertex $v_2$ at depth $\ell_2$ that has not been deleted, and use it to represent 2. We continue in this way, until every element of $[n]$ has been encoded.

This process can fail only if for some $j$ there are no available leaves at depth $\ell_j$. We show that this never happens. For $i < j$, the number of vertices at depth $\ell_j$ that are deleted in the $i$'th step is exactly $2^{\ell_j - \ell_i}$. So, the number of vertices at depth $j$ that are deleted before the $j$'th step is

$$\sum_{i=1}^{j-1} 2^{\ell_j - \ell_i} = 2^{\ell_j} \left( \sum_{i=1}^{j-1} 2^{-\ell_i} \right) \leq 2^{\ell_j} \sum_{i=1}^{j-1} p(i) < 2^{\ell_j}.$$ 

This proves that some vertex will always be available at the $j$'th step.

To show that no encoding can have expected length less than $H(X)$, suppose $X$ can be encoded in such a way that $i$ is encoded using $\ell_i$ bits. Then the expected length of the encoding is:

$$\mathbb{E}_{p(i)}[\ell_i] = \mathbb{E}_{p(i)}[\log(1/p(i))] - \mathbb{E}_{p(i)}[\log(2^{\ell_i}/p(i))]$$ 

$$\geq H(X) - \log \left( \mathbb{E}_{p(i)} \left[ 2^{\ell_i}/p(i) \right] \right)$$ 

$$= H(X) - \log \left( \sum_i 2^{-\ell_i} \right).$$ 

We claim that $\sum_i 2^{-\ell_i} \leq 1$. Imagine sampling a random path by starting from the root of the protocol tree, and picking one of the two children uniformly at random, until we reach a leaf. This random...
Later, we will use these properties to show that the projections onto one of the $xy$, $yz$ or $zx$ planes must be large; this proof is much easier using the entropy function.

**Chain Rule and Conditional Entropy**

The entropy function has several properties that make it particularly useful. To illustrate some of the properties, we use an example from geometry. Let $S$ be a set of $n^3$ points in $\mathbb{R}^3$, and let $S_x, S_y, S_z$ denote the projections of $S$ onto the $x, y, z$ axes.

**Claim 6.2.** One of $S_x, S_y, S_z$ must have size at least $n$.

This claim has an easy proof:

$$n^3 = |S| \leq |S_x| \cdot |S_y| \cdot |S_z|,$$

and so one of the three projections must be of size $n$. However, to introduce some properties of the entropy function, we seek a proof using entropy.

Let $(X, Y, Z)$ be a uniformly random point from $S$. We prove the claim by reasoning about $H(XYZ)$ and exploiting the properties of entropy. The property of the entropy function we need is called **subadditivity**. For any random variables $A, B$, we have:

$$H(AB) \leq H(A) + H(B). \quad (6.2)$$

This follows from the concavity of the log function:

$$H(A) + H(B) - H(AB) = \mathbb{E}_{p(ab)} \left[ \log \frac{1}{p(a)} \right] + \mathbb{E}_{p(ab)} \left[ \log \frac{1}{p(b)} \right] - \mathbb{E}_{p(ab)} \left[ \log \frac{1}{p(ab)} \right] = - \mathbb{E}_{p(ab)} \left[ \log \frac{p(a) \cdot p(b)}{p(ab)} \right] \geq - \log \sum_{a,b} p(a) \cdot p(b) = - \log 1 = 0.$$

Subadditivity is a very powerful property. It can be used even when many variables are involved:

$$H(A_1 A_2 \ldots A_k) \leq H(A_1) + H(A_2 \ldots A_k) \leq H(A_1) + H(A_2) + H(A_3 \ldots A_k) \leq \ldots \leq \sum_{i=1}^{k} H(A_i).$$

To prove Claim 6.2, recall that $(X, Y, Z)$ is a uniformly random element of $S$. So,

$$3 \log n = \log |S| = H(XYZ) \leq H(X) + H(Y) + H(Z),$$

by subadditivity.
proving that one of the three terms \( H(X), H(Y), H(Z) \) must be at least \( \log n \). By (6.1), the projection onto the corresponding coordinate must be supported on at least \( n \) points. This proves Claim 6.2.

Of course, this is not very surprising—we did not need the definition of entropy to reach this conclusion. Things become more interesting when we study the projections of \( S \) to the \( xy, yz \) and \( zx \) planes. Let \( S_{xy}, S_{yz}, S_{zx} \) denote these projections. Then we claim:

**Claim 6.3.** One of \( S_{xy}, S_{yz}, S_{zx} \) must have size at least \( n^2 \).

To proceed, we need to define the notion of *conditional entropy*. For two random variables \( A \) and \( B \), the entropy of \( B \) conditioned on \( A \) is

\[
H(B|A) = \mathbb{E}_{p(a,b)} \left[ \log \frac{1}{p(b|a)} \right].
\]

This is the expected entropy of \( B \), conditioned on the event \( A = a \), where the expectation is over \( a \). The chain rule for entropy states that:

\[
H(AB) = H(A) + H(B|A).
\]

The proof follows from Baye’s rule and linearity of expectation:

\[
H(AB) = \mathbb{E}_{p(ab)} \left[ \log \frac{1}{p(ab)} \right] \\
= \mathbb{E}_{p(ab)} \left[ \log \frac{1}{p(a) \cdot p(b|a)} \right] \\
= \mathbb{E}_{p(ab)} \left[ \log \frac{1}{p(a)} + \log \frac{1}{p(b|a)} \right] = H(A) + H(B|A).
\]
The chain rule shows that the entropy of $AB$ is the entropy of $A$ plus the entropy of $B$ given that we know $A$. The chain rule is an extremely useful property of entropy. It allows us to express the entropy of a collection of random variables in terms of the entropy of individual variables.

To better understand why the conditioning is necessary, consider the following example. Suppose $A$, $B$ are two uniformly random bits that are always equal. Then $H(AB) = H(A) = H(B) = 1$, so $H(AB) > H(A) + H(B)$. Nevertheless, $H(B|A) = 0$ and hence $H(AB) = H(A) + H(B|A) < H(A) + H(B)$.

Conditional entropy satisfies another intuitive property—conditioning can only decrease entropy:

$$H(B|A) \leq H(B)$$

Indeed, using the chain rule and subaddtivity, we have

$$H(A) + H(B|A) = H(AB) \leq H(A) + H(B),$$

which proves (6.3).

We now have enough tools to prove Claim 6.3. As before, let $(X,Y,Z)$ be a uniformly random point of $S$. Then $H(XYZ) = \log |S| = 3 \log n$. Repeatedly using the fact that conditioning cannot increase entropy, (6.3) we have:

\[
\begin{align*}
H(X) &+ H(Y|X) \leq H(X) + H(Y|X), \\
H(X) &+ H(Z|XY) \leq H(X) + H(Z|X), \\
H(Y|X) &+ H(Z|XY) \leq H(Y) + H(Z|Y).
\end{align*}
\]

Adding these inequalities together and applying the chain rule gives

$$6 \log n = 2 \cdot H(XYZ) \leq H(XY) + H(XZ) + H(YZ).$$

Thus, one of three terms on the right hand side, must be at least $2 \log n$. The projection onto the corresponding plane must be of size at least $n^2$.

**Combinatorial Applications**

The entropy function has found many applications in combinatorics, where it can be used to give simple proofs. Here we give a few examples that illustrate its power and versatility.

**Counting paths and cycles in a graph** Suppose we have a graph on $n$ vertices with $m$ edges. Since the sum of the degrees of the vertices is $2m$, the average degree of the vertices is $d = 2m/n$. Here we recall that the number of neighbors of a vertex is called its degree.

In Lemma 5.3, we proved a lower bound on the number of 4-cycles in the graph.
prove lower bounds on the number of paths and cycles in the graph.

Let $X, Y, Z$ be a random path of length 2 in the graph, obtained by sampling a uniformly random edge $X, Y$, and then a uniformly random neighbor $Z$ of $Y$. After fixing $Y$, the vertex $Z$ is independent of the choice of $X$. Thus, we can use the chain rule to write:

$$H(XYZ) = H(XY) + H(Z|XY) = \log m + H(Z|Y).$$

To bound $H(Z|Y)$, we use convexity. If $d_v$ denotes the degree the vertex $v$, we have:

$$H(Z|Y) = \sum_v \frac{d_v}{2m} \cdot \log d_v = \frac{n}{2m} \cdot \sum_v \frac{1}{n} \cdot d_v \log d_v \geq \frac{n}{2m} \cdot d \cdot \log d = \log \frac{2m}{n}.$$ 

Thus, we have

$$H(XYZ) \geq \log \frac{2m^2}{n},$$

proving that the support of $XYZ$ must contain at least $\frac{2m^2}{n}$ elements. Some of the elements in the support of $XYZ$ do not correspond to paths of length 2, since we could have $x = z$. However, there are at most $2m$ sequences $x, y, z$ that correspond to such a redundant choice. Moreover, each path of length 2 can be expressed in two ways as a sequence $X, Y, Z$. After correcting for these counts, we are left with at least

$$\left( \frac{2m^2}{n} - 2m \right) / 2 = m\left( \frac{n}{m} - 1 \right)$$

paths of length 2.

Next, we turn to proving a lower bound on the number of 4-cycles. Sample $X, Y, Z$ as before, and then independently sample $W$ using the distribution of $Y$ conditioned on the values of $X, Z$. Then,

$$H(XYZW) = H(XYZ) + H(W|XZ) = H(XYZ) + H(XWZ) - H(XZ) \geq 2 \cdot H(XYZ) - 2 \log n.$$ 

Combining this with our bound for $H(XYZ)$, we get

$$H(XYZW) \geq \log \frac{4m^4}{n^4}.$$ 

This does not quite count the number of 4-cycles, because there could be some settings of $XYZW$ where two of the vertices are the
same. We could have $X = Z$ or $Y = W$. However, there are at most $2n^3$ possible elements in the support of $XYZW$ where that can happen. Each cycle can be expressed in at most 4 different ways as $XYZW$. After accounting for these facts, we are left with at least

$$\left(\frac{4m^4}{n^4} - 2n^3\right)/4 = \frac{m^4}{n^4} - \frac{n^3}{2}$$

distinct cycles.

Bounding the girth of a graph Another interesting parameter of a graph is its girth: the length of the smallest cycle. Suppose we have a graph with $n$ vertices and $m$ edges, and let $d = 2m/n$ denote the average degree. Here we show how to give an upper bound on the girth $g$ in terms of $n$ and $m$, using an argument based on entropy.

For simplicity, let us assume that $g$ is an odd number $g$. We also make the necessary assumption that $d > 2$, since if $d \geq 2$, there are graphs which have very large girth.

If every vertex in the graph has the same degree, then the vertices at distance $\frac{g-1}{2}$ from any fixed vertex must form a rooted $(d-1)$-ary tree, or else the graph would have a cycle of length less than $g$. This proves that $(d-1)^{\frac{g-1}{2}} \leq n$ and so $g \leq \frac{2\log n}{\log(d-1)} + 1$.

We shall prove that the same bound holds even when the vertices do not all have the same degree. We start by observing that we may assume that $d_v \geq 2$ for each vertex $v$. Indeed, if $v$ is a vertex with $d_v = 1$, then by deleting $v$ from the graph we obtain a graph with fewer vertices, larger average degree and yet the same girth.

Let $X = (X_0, X_1, \ldots, X_{g-1})$ be a random path in the graph, sampled as follows. Let $X_0, X_1$ be a random edge, and for $i > 1$, let $X_i$ be a random neighbor of $X_{i-1}$ that is not the same as $X_{i-2}$. This is a non-backtracking path—the path never returns along an edge that it took in the last step.

Given $X_1$, we see that $X_2$ and $X_0$ are identically distributed. Hence, each edge of this path is identically distributed. The chain rule gives

$$H(X|X_0) = \sum_{i=1}^{g-1} H(X_i|X_{i-1}).$$

If $d_v$ denotes the degree of the vertex $v$, we can calculate each term

There are at most $n^3$ choices for $XYZW$ with $X = Z$, and at most $n^3$ with $Y = W$.  

6 Babu and Radhakrishnan, 2010; and Alon et al., 2002

The ideas given here can easily be modified to apply when $g$ is even.

Can you think of a graph with $d = 2$ that has large girth?

Figure 6.3: If the vertices near a fixed vertex do not form a tree, then the graph contains a short cycle.
$H(X_i|X_{i-1})$ as

$$H(X_i|X_{i-1}) = \sum_v \frac{d_v}{2m} \cdot \log(d_v - 1)$$

$$= \frac{1}{d} \cdot \sum_v \frac{d_v}{n} \cdot \log(d_v - 1)$$

$$\geq \frac{1}{d} \cdot d \log(d - 1) = \log(d - 1).$$

Putting these bounds together, we get:

$$H(X|X_0) \geq \frac{g-1}{2} \cdot \log(d-1).$$

Since the girth of the graph is $g$, there can be at most $n$ distinct paths of length $\frac{g-1}{2}$ that begin at $X_0$. Thus,

$$\log n \geq H(X|X_0) \geq \frac{g-1}{2} \cdot \log(d-1),$$

proving that $g \leq \frac{2 \log n}{\log(d-1)} + 1$.

An isoperimetric inequality for the hypercube

An isoperimetric inequality identifies the shape of a given volume that has the smallest boundary. For example, if we are working with the geometry of the plane, the shape of unit area with the smallest boundary is a disc. Here we prove a similar fact, albeit in a discrete geometric space.

The $n$-dimensional hypercube is the graph whose vertex set is $\{0, 1\}^n$, and whose edge set consists of pairs of vertices that disagree in exactly one coordinate. The hypercube contains $2^n$ vertices and $\frac{n}{2} \cdot 2^n$ edges. Given a subset $S$ of the edges in the hypercube, we define its volume to be $|S|$. The boundary of $S$ consists of the set of edges that go from inside $S$ to outside $S$. We write $\delta(S)$ to denote the size of the boundary of $S$. We want to understand the sets $S$ that minimize $\delta(S)$ for a given value of $|S|$.

A $k$-dimensional subcube of the hypercube is a subset of the vertices given by fixing $n-k$ coordinates of the vertices to some fixed value and, allowing $k$ of the coordinates to take any value. The volume of such a subcube is exactly $2^k$. Each vertex of the subcube has $n-k$ edges that leave the subcube, so the boundary of the subcube is of size $(n-k)2^k$.

We shall prove that the subcube has the smallest boundary of any set of the same volume:

**Theorem 6.4.** For any subset $S$ of the vertices, if $|S| \geq 2^k$, then $\delta(S) \geq (n-k)2^k$.

**Proof.** Let $e(S)$ be the number of edges contained in $S$. Then we have

$$\delta(S) = n|S| - 2e(S),$$

since the function $z \log(z-1)$ is convex for $z \geq 2$. Prove it by computing derivatives.
so it suffices to prove that the subcube maximizes \( \epsilon(S) \) when \( |S| = 2^k \).

Let \( X \) be a uniformly random element of \( S \). For every vertex \( x \in S \) and \( y \) such that \( \{x, y\} \) is an edge of the hypercube, and \( x, y \) disagree in the \( i \)'th coordinate, we have

\[
H(X_i | X_{<i} = x_{<i}) = \begin{cases} 
1 & \text{if } \{x, y\} \subset S, \\
0 & \text{otherwise}.
\end{cases}
\]

By subadditivity, and since conditioning does not increase the entropy,

\[
\log |S| = H(X) = \sum_{i=1}^{n} H(X_i | X_{<i}) \geq \sum_{i=1}^{n} \frac{1}{|S|} H(X_i | X_{<i}) = \frac{2\epsilon(S)}{|S|}.
\]

This proves that \( \epsilon(S) \geq \frac{|S| \log |S|}{2} \), which is exactly the parameter achieved by a subcube. \( \square \)

**Shearer’s Inequality**

Shearer’s inequality is a generalization of the subadditivity of entropy. Suppose \( X = X_1, \ldots, X_k \) is a collection of \( n \) jointly distributed random variables, and \( S \subseteq [k] \) is a set of coordinates sampled independently of \( X \). Then we write \( X_S \) to denote the collection of variables that correspond to \( S \). One way to interpret the subadditivity of entropy (6.2) is that when \( S \) is uniformly random subset of size 1, then

\[
H(X_S | S) \geq \frac{1}{n} H(X).
\]

Shearer’s inequality is a generalization of this fact:

**Lemma 6.5.** If \( p(i \in S) \geq \epsilon \) for every \( i \in [n] \), then \( H(X_S | S) \geq \epsilon \cdot H(X) \).

**Proof.** Since conditioning does not increase entropy,

\[
H(X_S | S) \geq \mathbb{E}_S \left[ \sum_{i \in S} H(X_i | X_{<i}) \right] = \sum_{i=1}^{n} p(i \in S) \cdot H(X_i | X_{<i}) \geq \epsilon \cdot H(X). \quad \square
\]

We shall show how Shearer’s inequality can be used to understand families of graphs that have structured intersections. First, a warmup: suppose \( \mathcal{F} \) is a family of subsets of \( [n] \) such that any two sets from \( \mathcal{F} \) intersect. How large can such a family be?
Claim 6.6. $|\mathcal{F}| \leq 2^{n-1}$.

Proof. For any set $T \in \mathcal{F}$, its complement cannot be in $\mathcal{F}$. So only half of all the sets can be in $\mathcal{F}$. □

Now, let us try to study the same kind of question for families of graphs. Let $\mathcal{G}$ be a family of graphs on $n$ vertices such that every two graphs intersect in a triangle. Such a family can be obtained by including a fixed triangle, which gives a family with $2^{\binom{n}{2}}/8$ graphs. This bound is known to be tight, but here we give a simple argument that provides a partial converse:

Theorem 6.7. $|\mathcal{G}| \leq 2^{\binom{n}{2}}/4$.

Proof. Let $G$ be a uniformly random graph from the family. $G$ can be described by a binary vector of length $\binom{n}{2}$, where each bit indicates whether a particular edge is present or not. Let $S$ be a uniformly random subset of the vertices, so that each vertex $v$ is in $S$ with probability 1/2, independently of all other vertices.

Let $G_S$ denote the graph obtained from $G$ by deleting all edges that go from $S$ to the complement of $S$. Since the probability that any particular edge is retained is exactly 1/2, Shearer’s inequality gives

$$\mathbb{E}_S [H(G_S|S)] \geq H(G)/2.$$

Now, since every two graphs $G, G'$ in the family intersect in a triangle, we must have that $G_S, G'_S$ must share an edge, no matter what $S$ is, because at least one of the edges of the triangle is not deleted. This means that the number of graphs of the form $G_S$ is at most half of all possible options, by Claim 6.6. Writing

$$e(S) = \binom{|S|}{2} + \binom{n - |S|}{2}$$

for the total number of edges possible in the graph $G_S$, this means that $H(G_S|S) \leq e(S) - 1$. The expected value of $\mathbb{E} [e(S)]$ is exactly $\binom{n}{2}/2$, since each edge of the complete graph is counted in the expectation with probability 1/2. Thus, we have

$$\frac{1}{2} \cdot \binom{n}{2} = \mathbb{E}_S [e(S)] \geq \mathbb{E}_S [H(G_S|S)] + 1 \geq \frac{1}{2} \cdot H(G) + 1.$$

So $H(G) \leq \binom{n}{2} - 2$, which implies that $|\mathcal{G}| \leq 2^{\binom{n}{2}} - 2 = 2^{\binom{n}{2}}/4$. □

Divergence and Mutual Information

The concepts of divergence and mutual information are closely related to the concept of entropy. They provide a toolbox that helps to understand the flow of information in a variety of situations.
Suppose \((X, Y)\) are random inputs to a communication protocol sampled from a distribution \(p_0\), and suppose \(M\) denotes the \(T\) bits transmitted during the execution of the protocol. Consider a sequence of distributions \(p_1, p_2, \ldots, p_T\), where \(p_t\) is the distribution of \((X, Y)\) conditioned on the value of \(M_1, \ldots, M_t\). One would expect that when the protocol terminates, \(p_T\) should be far away from \(p_0\). This corresponds to the idea that the protocol has computed something about the inputs \(X, Y\). The flow of information can be thought of as the evolution in time of the distance of \(p_0\) from \(p_t\) for some appropriate notion of distance between distributions. At time \(t = 0\), this distance ought to be 0, and eventually this distance should be large. Having a good understanding of this flow of information often enables us to prove interesting statements about communication protocols. The choice of the notion of distance used here is crucial in such an analysis. It turns out that the divergence between distributions, which we define below, is the right notion.

The divergence between two distributions \(p(x)\) and \(q(x)\) is defined to be

\[
\frac{p(x)}{q(x)} = \sum_x p(x) \log \frac{p(x)}{q(x)} = \mathbb{E}_{p(x)} \left[ \log \frac{p(x)}{q(x)} \right].
\]

The divergence can be thought of as a measure of distance between the two distributions \(p\) and \(q\). In line with this intuition, we have

**Fact 6.8.** \(\frac{p(x)}{q(x)} \geq 0\), and equality holds if and only if \(p\) and \(q\) are identical.

The divergence is sometimes called Kullback-Leibler divergence or KL-divergence and is often denoted \(D(p||q)\). It is part of a larger family of functions called \(f\)-divergences.
Proof. We have
\[
\frac{p(x)}{q(x)} = -\sum_x p(x) \log \frac{q(x)}{p(x)} \geq -\log \sum_x p(x) \frac{q(x)}{p(x)} = \log 1 = 0.
\]
The inequality follows from the convexity of the log function.

Since the log function is strictly convex, the inequality is a strict inequality unless \( \frac{p(x)}{q(x)} \) is the same for every \( x \). This can happen only when \( p(x) \) and \( q(x) \) are the same distribution.

The divergence is, however, not symmetric; it is possible that \( \frac{p(x)}{q(x)} \neq \frac{q(x)}{p(x)} \).

Moreover, the divergence can be infinite, for example if \( p \) is supported on a point that has 0 probability under \( q \). If \( X \) is an \( \ell \)-bit string, we see that:
\[
H(X) = \mathbb{E}_{p(x)} [\log(1/p(x))] = \ell - \mathbb{E}_{p(x)} \left[ \log \frac{p(x)}{2^{\ell}} \right] = \ell - \frac{p(x)}{q(x)},
\]
where \( q(x) \) is the uniform distribution on \( \ell \)-bit strings. So we see that the entropy of a string is just the divergence from uniform.

More generally, we have
\[
\frac{p(x)}{q(x)} = \mathbb{E}_{p(x)} [\log(1/q(x))] - H(X).
\]

In a sense, this shows that the divergence measures the loss of efficiency incurred when one tries to encode a random variable using the wrong distribution. As we saw at the beginning of this chapter, when \( X \) is distributed according to \( p \), the optimal encoding for \( X \) encodes each sample \( x \) using \( \lceil \log(1/p(x)) \rceil \) bits. Such an encoding has an expected length of at most \( H(X) + 1 \). Let us denote this encoding as \( E(p) \). Now, the quantity \( H_{p,q} := \mathbb{E}_{p(x)} [\log(1/q(x))] \) is, up to a 1, the expectation with respect to \( p \) of the encoding length of \( E(q) \). In other words, it is the expected length of an encoding with respect to the distribution \( q \). So, the divergence \( H_{p,q} - H(p) \) can be thought of as the loss incurred when we encode \( X \) using the wrong distribution \( q \) when \( X \) is actually sampled from \( p \)—we could get an encoding length of \( H(p) \), but we only get an expected length of \( H_{p,q} \).

We can use divergence to quantify the dependence between two random variables. If \( p(a,b) \) is a joint distribution of two random variables \( A \) and \( B \), we define the mutual information between \( A \) and \( B \) to be
\[
I(A : B) = \mathbb{E}_{p(a,b)} \left[ \log \frac{p(a,b)}{p(a)p(b)} \right].
\]
By Baye’s rule, we can write this as

$$I(A : B) = \mathbb{E}_{p(a,b)} \left[ \log \frac{p(b|a)}{p(b)} \right] = \mathbb{E}_{p(a)} \left[ \frac{p(b|a)}{p(b)} \right] = H(B) - H(B|A).$$

The third expression is the expected divergence between $p(b|a)$ and $p(b)$; it measures the distance of $p(b|a)$ from $p(b)$, for an average $a$. The fourth expression says that the information measures the decrease in the entropy of $B$ when conditioning on $A$. Of course, since the information is symmetric, we have $I(A : B) = H(A) - H(A|B)$ as well.

When $A = B$, we have $I(A : B) = H(B)$. In the other extreme, $I(A : B) = 0$ exactly when $A$ and $B$ are independent. In general, the mutual information satisfies

$$0 \leq I(A : B) = H(A) - H(A|B) \leq H(A).$$

The first inequality follows from the fact that conditioning can only decrease entropy, and the second from the fact that entropy is always non-negative.

**Lower Bound for Indexing**

We have gathered enough tools to begin discussing our first lower bound on communication using information theory. We shall prove a lower bound on the *indexing problem.*

Suppose Alice has a uniformly random $n$ bit string $x$, and Bob is given an independent uniformly random index $i \in [n]$. The goal of the players is to compute the $i$’th bit $x_i$, but they are only allowed to execute a protocol of a specific form—the protocol must start with a message from Alice to Bob, and then Bob must output the answer. We prove that $\Omega(n)$ bits of communication are necessary, even if the parties are allowed to use randomness and make errors with some small probability.

Suppose there is a protocol for this problem where Alice sends a message $M$ that is $\ell$ bits long. Intuitively, since $M$ is only $\ell$ bits long, it can have only $\ell$ bits of information about $x$. So, we should be able to argue that $M$ carries only $\ell/n$ bits of information about a typical coordinate $x_i$. If indeed this is the case and $\ell/n \ll 1$, then $M$ should not be useful to help Bob determine $x_i$.

**Chain Rules for Divergence and Mutual Information**

We have already learnt a chain rule for the entropy function, and used it a few times. Divergence and mutual information have similar
chain rules that are equally useful. Since $M$ has only $\ell$ bits, the total amount of information $M$ has about $X$ is at most $\ell$. We shall use a chain rule to argue that $M$ can convey only $\ell/n$ bits of information about $X_i$.

The chain rule for divergence states that for every two distributions $p(a, b)$ and $q(a, b)$,

$$\frac{p(a, b)}{q(a, b)} = \frac{p(a)}{q(a)} + \mathbb{E}_{p^{(a)}} \left[ \frac{p(b|a)}{q(b|a)} \right].$$

The proof is a straightforward calculation using Bayes’s rule:

$$\frac{p(a, b)}{q(a, b)} = \mathbb{E}_{p^{(a,b)}} \left[ \log \frac{p(a) \cdot p(b|a)}{q(a) \cdot q(b|a)} \right] = \mathbb{E}_{p^{(a,b)}} \left[ \log \frac{p(a)}{q(a)} \right] + \mathbb{E}_{p^{(a,b)}} \left[ \log \frac{p(b|a)}{q(b|a)} \right].$$

In words, the total divergence is the sum of the divergence from the first variable, plus the expected divergence from the second variable.

Before we state the chain rule for information, it is worthwhile to think about a simple example. Suppose $A, B, C$ are three random bits that are all equal to each other. Then $I(AB : C) = 1 < 2 = I(A : C) + I(B : C)$. On the other hand, if $A, B, C$ are three random bits satisfying $A + B + C = 0 \mod 2$, we have $I(AB : C) = 1 > 0 = I(A : C) + I(B : C)$. Nevertheless, a chain rule does hold for mutual information—we need to use the right definitions. For three random variables $A, B$ and $C$, define

$$I(B : C|A) = \mathbb{E}_{p^{(a,b,c)}} \left[ \log \frac{p(b,c|a)}{p(b|a) \cdot p(c|a)} \right].$$

The chain rule for mutual information is then

$$I(AB : C) = I(A : C) + I(B : C|A).$$

This chain rule also has an intuitive meaning: the information $AB$ give about $C$ is the information $A$ gives about $C$ plus the information $B$ gives about $C$ when we already know $A$. As usual, the proof is a straightforward application of Bayes’s rule:

$$I(AB : C) = \mathbb{E}_{p^{(a,b,c)}} \left[ \log \frac{p(a,c) \cdot p(b|a,c)}{p(a)p(b|a) \cdot p(c)} \right] = I(A : C) + \mathbb{E}_{p^{(a,b,c)}} \left[ \log \frac{p(b|a,c)}{p(b|a) \cdot p(c|a)} \right] = I(A : C) + I(B : C|A).$$
Subadditivity

Unlike the entropy, the mutual information can go up under conditioning. For example, if $A, B, C$ are three random bits subject to $A + B + C = 0 \mod 2$, then $0 = I(A : B) < I(A : B | C) = 1$. Nevertheless, we can prove a subadditivity bound, under the assumption that the variables under consideration are independent.

**Theorem 6.9.** Let $A_1, \ldots, A_n$ be independent random variables, and $B$ be jointly distributed. Then,

$$\sum_{i=1}^n I(A_i : B A < i) \leq I(A_1, \ldots, A_n : B) \leq H(B).$$

**Proof.** We have

$$H(B) \geq I(A_1, \ldots, A_n : B) = H(A_1, \ldots, A_n) - H(A_1, \ldots, A_n | B).$$

The first term is exactly equal to $\sum_{i=1}^n H(A_i)$, since $A_1, \ldots, A_n$ are independent. On the other hand, the chain rule gives that

$$H(A_1, \ldots, A_n | B) = \sum_{i=1}^n H(A_i | B A < i).$$

So, we get

$$I(A_1, \ldots, A_n : B) \geq \sum_{i=1}^n H(A_i) - H(A_i | B A < i) = \sum_{i=1}^n I(A_i : B A < i),$$

as required. \qed

Returning to the indexing problem, we can apply Theorem 6.9 to $X_1, \ldots, X_n$ and $M$, to obtain

$$\mathbb{E} [I(X_i : M)] \leq \mathbb{E} [I(X_i : MX < i)] = (1/n) \sum_{i=1}^n I(X_i : MX < i) \leq \frac{\ell}{n}.$$

This inequality captures our intuition that Alice’s message does not contain much information about the bit that Bob cares about. However, the proof is not yet complete: we need to prove that if $M$ has low mutual information with $X_i$ then Bob cannot use $M$ to guess the value of $X_i$ with high probability. We need one more technical tool to prove this—Pinsker’s inequality.

**Pinsker’s Inequality**

Pinsker’s inequality bounds the statistical distance between two distributions in terms of the divergence between them.
Lemma 6.10. \( \frac{p(x)}{q(x)} \geq \frac{2}{\ln 2} \cdot |p - q|^2. \)

**Proof.** Let \( T \) be the set that maximizes \( p(T) - q(T) \), and define

\[
x_T = \begin{cases} 
1 & \text{if } x \in T, \\
0 & \text{otherwise.}
\end{cases}
\]

By the chain rule,

\[
\frac{p(x)}{q(x)} \geq \frac{p(x_T)}{q(x_T)}.
\]

Since \( |p - q| = p(T) - q(T) = p(x_T = 1) - q(x_T = 1) \), it remains to prove that

\[
\frac{p(x_T)}{q(x_T)} \geq \frac{2}{\ln 2} \cdot (p(x_T = 1) - q(x_T = 1))^2.
\]

Let \( \epsilon = p(x_T = 1) \) and \( \gamma = q(x_T = 1) \). It is enough to prove that

\[
\epsilon \log \frac{\epsilon}{\gamma} + (1 - \epsilon) \log \frac{1 - \epsilon}{1 - \gamma} - \frac{2}{\ln 2} \cdot (\epsilon - \gamma)^2 \tag{6.4}
\]

is always non-negative. \( (6.4) \) is 0 when \( \epsilon = \gamma \), and its derivative with respect to \( \gamma \) is

\[
\frac{-\epsilon}{\gamma \ln 2} + \frac{1 - \epsilon}{(1 - \gamma) \ln 2} - \frac{4(\gamma - \epsilon)}{\ln 2} = \frac{(\gamma - \epsilon)}{\ln 2} \left( \frac{1}{\gamma (1 - \gamma)} - 4 \right).
\]
Since $\frac{1}{\gamma (1-\gamma)}$ is always at most 4, the derivative is non-positive when $\gamma \leq \epsilon$, and non-negative when $\gamma \geq \epsilon$. This proves that (6.4) is indeed always non-negative.

Pinsker’s inequality implies that two variables that have low information with each other are statistically close to being independent:

**Corollary 6.11.** If $A, B$ are random variables then on average over $b$,

$$p(a|b) \approx p(a),$$

where $\epsilon = \sqrt{\frac{\ln 2 \cdot I(A:B)}{2}}$.

Another useful corollary is that conditioning on a low entropy random variable cannot change the distribution of many other independent random variables:

**Corollary 6.12.** Let $A_1, \ldots, A_n$ be independent random variables, and $B$ be jointly distributed. Let $i \in [n]$ be uniformly random and independent of all other variables. Then on average over $i, b, a, < i$,

$$p(a_i|b, a, < i) \approx p(a_i),$$

where $\epsilon \leq \sqrt{\frac{H(B) \ln 2}{2n}}$.

**Proof.** By Theorem 6.9, $H(B) \geq \sum_{j=1}^{n} I(A_j : BA < j)$. Thus we get that for a uniformly random coordinate $i$,

$$\mathbb{E} \left[ I(A_i : BA < i) \right] \leq H(B) / n.$$

The bound then follows from Corollary 6.11.

We are finally ready to prove the lower bound we wanted. By Corollary 6.12, on average over $m$ and $i$,

$$p(x_i|m) \approx p(x_i),$$

with $\epsilon = \frac{\ell \ln 2}{2n}$. Since $p(x_i)$ is uniform for each $i$, the probability that Bob makes an error in the $i$'th coordinate must be at least $1/2 - |p(x_i|m) - p(x_i)|$. So the probability that Bob makes an error is at least $1/2 - \sqrt{\frac{\ell \ln 2}{2n}}$, proving that at least $\Omega(n)$ bits must be transmitted if the protocol has a small probability of error.

**Round Elimination**

The ideas we have seen for proving lower bounds on the indexing functions are quite powerful. They show that the message sent by Alice sends the majority of all her bits, that bit is equal to a random coordinate with probability $1/2 + \Omega(1/\sqrt{n})$. See Exercise 6.2.

Note that if Alice has a random set from a family of sets of size $2^{\Omega(n)}$, the lower bound for indexing would still hold. The lower bound even extends to the case that Bob knows $x_1, \ldots, x_{i-1}$. The square-root dependence is tight: If
Alice provides no useful information about the coordinate of Alice’s input that Bob cares about. These ideas are powerful enough to prove lower bounds on multi-round protocols as well. We illustrate this using the greater-than function.

Recall that the greater than function $\text{GT}(x, y)$ gets inputs $x, y \in [n]$ and computes:

$$
\text{GT}(x, y) = \begin{cases} 
1 & \text{if } x > y, \\
0 & \text{otherwise}.
\end{cases}
$$

In Chapter 1, we showed that every deterministic protocol for GT requires $\log n$ bits of communication. In Chapter 3, we discussed a randomized protocol to compute the greater than function with $O(\log \log n)$ bits of communication. Here we show\textsuperscript{11} that randomized protocols require much more communication if the protocols used involve a small number of rounds of communication.

**Theorem 6.13.** Any randomized $k$-round protocol for computing greater-than requires communication at least

$$
\Omega \left( \frac{(\log n)^{1/k}}{k^2} \right).
$$

To prove the theorem, we define a sequence of hard distributions on numbers $x, y$. In the first distribution $\mu(x, y)$, let $m$ be odd, and let $x \in [m]$ be a uniformly random even integer, and let $y \in [m]$ be a uniformly random odd integer. It is easy to see that the probability that $x > y$ is exactly 1/2. Thus, any protocol that computes $\text{GT}(x, y)$ without communicating must make an error with probability at least 1/2.

Now, to find a hard distribution for $k$-round protocols with $k = 1$, where Alice sends a single message to Bob of length $c$, set $t = \lceil c/\epsilon^2 \rceil$. We sample $t$ independent samples

$$(x_1, y_1), (x_2, y_2), \ldots, (x_t, y_t)$$

from the distribution $\mu$, sample $i \in [t]$ uniformly at random, and set

$$x = (x_1 - 1) \cdot n^{t-1} + (x_2 - 1) \cdot n^{t-2} + \ldots + (x_t - 1).$$

Intuitively, this is the number whose digits are obtained by concatenating the digits of $x_1, x_2, \ldots, x_t$. We set

$$y = (x_1 - 1) \cdot n^{t-1} + (x_2 - 1) \cdot n^{t-2} + \ldots + x_{i-1} \cdot n^{t-i+1} + y_i \cdot n^{t-i}.$$

Thus, $y$ is the number whose digits are obtained by concatenating the digits of $x_1, x_2, \ldots, x_{i-1}, y_i, 0, 0, \ldots, 0$. Since the most significant digits of $x, y$ are the same, and $x_i \neq y_i$, only the digits of $x_i, y_i$
matter when determining whether $x$ or $y$ is greater. Consequently, $\text{GT}(x, y) = \text{GT}(x_i, y_i)$.

We claim that the protocol must make an error with probability at least $1/2 - \epsilon$ when the inputs are sampled according to the above distribution. Indeed, since $x_1, \ldots, x_i$ are independent, and Alice’s first message $m$ has entropy at most $c$, by Corollary 6.12, we have on average over $i, m, x_{<i}$,

$$p(x_i|m, x_{<i}) \leq p(x_i).$$

Thus, there must be some fixed value of $i, m, x_{<i}$ for which this error is achieved. Since $y_i$ is independent of $x_1, \ldots, x_i$ after $x_i$ is fixed, we shall argue that

$$p(x_{<i}, y_i|m, x_{<i}) \leq p(x_{<i}, y_i).$$

Indeed, $p(y_i|x_1, \ldots, x_i) = p(y_i|x_i)$, so we have

$$|p(x_{<i}, y_i|m, x_{<i}) - p(x_{<i}, y_i)| = \sum_{x_i, y_i} |p(x_i, y_i|m, x_{<i}) - p(x_i, y_i)|$$

$$= \sum_{x_i, y_i} |p(x_i|m, x_{<i}) \cdot p(y_i|x_{<i}) - p(x_i) \cdot p(y_i|x_i)|$$

$$= \sum_{x_i, y_i} p(y_i|x_i) \cdot |p(x_i|m, x_{<i}) - p(x_i)|$$

$$= \sum_{x_i} |p(x_i|m, x_{<i}) - p(x_i)|.$$

These fixed values induce a 0-round protocol for computing $\text{GT}(x, y)$, and so they give a protocol for computing $\text{GT}(x_i, y_i)$ with error at most $\epsilon$ more than the original protocol. Since the error of a 0 round protocol must be at least 1/2, the 1 round protocol must make an error with probability at least $1/2 - \epsilon$.

The above argument can be repeated for general $k$ to get a distribution on inputs where any $k$ round protocol must make an error with probability at least $1/2 - \epsilon k$. In each step, we obtain a distribution that is hard for $k$ round protocols using a distribution that is hard for $k - 1$ round protocols. The size of the input increases in each step from $n$ to $n^{(c^2)/2}$, so if when $k = 0$ we let $x, y$ come from the set $[3]$, we obtain our final distribution on a universe of size $3^{(c^2)/2}$. Setting $\epsilon = 1/8k$, when $c \leq \frac{(\log n)^{\frac{1}{k}}}{128k^2}$, we get that the inputs are supported on a set of size at most $n$, as required. The argument proves that the error for any such protocol must be at least $1/2 - 1/8 = 1/4$.

**Randomized Communication of Disjointness**

*One of the triumphs of* information theory is its ability to prove optimal lower bounds on the randomized communication complexity
of functions like disjointness\textsuperscript{12}, which we do not know how to prove any other way.

**Theorem 6.14.** Any randomized protocol that computes disjointness function with error $1/2 - \varepsilon$ must have communication $\Omega(\varepsilon^2 n)$.

**Obstacles to Proving Theorem 6.14**

The most natural way to prove lower bounds on randomized protocols is to find a hard distribution on the inputs, such that any protocol with low communication must make an error a significant fraction of the time. If we adopt this approach, we need not worry about the protocol being randomized, since any randomized protocol that works on average over the hard distribution implies the existence of a deterministic protocol as well.

This is the approach we took when we proved lower bounds on the inner-product function (Theorem 5.6), where we used the uniform distribution. The uniform distribution is also hard for the pointer-chasing problem (Theorem 6.18). The uniform distribution, however, is not a hard distribution for disjointness: two uniformly random sets $X, Y$ will intersect with very high probability, so the protocol can output 0 without communicating and still have very low error. In fact, it can be shown that any distribution where $X$ and $Y$ are independent cannot be used to prove a strong lower bound. Therefore, the hard distribution, if it exists, must involve strong correlations between $X$ and $Y$.

Given these constraints, we shall use a natural distribution on correlated sets. The distribution of $X, Y$ will be a convex combination of two distributions:

1. two random disjoint sets, and

2. two sets that intersect in exactly one element.

Once we restrict our attention to such a distribution, we have a second challenge: the pairs of variables $X_i, Y_i$ and $X_j, Y_j$ are not independent for $i \neq j$. This makes arguments involving subadditivity much harder to carry out, because subadditivity of information crucially relies on independence. The subtleties in the proof arise from circumventing these obstacles.

**Proving Theorem 6.14**

Given a randomized protocol with error $1/2 - \varepsilon$, one can make the error an arbitrarily small constant by repeating the protocol $O(1/\varepsilon^2)$ times and outputting the majority outcome. This means

\textsuperscript{12} Kalyanasundaram and Schnitger, 1992; Razborov, 1992; Bar-Yossef et al., 2004; and Braverman and Moitra, 2013

By Theorem 3.3, Theorem 6.14 is equivalent to the existence of such a hard distribution.

Suppose $X, Y$ are represented using their characteristic vectors. Intuitively, when $X, Y$ are independent, for typical $i$, on the one hand if the entropy $H(X_i | X_{<i}), H(Y_i | Y_{<i}) \ll 1$ then Alice can encode the relevant coordinates of her set (those where there is a good chance of an intersection) in much less than $n$ bits and send them to Bob. On the other hand, if this entropy is typically close to 1, then the sets will intersect with high probability, so Alice and Bob can output 0 without communicating.
that, it suffices to show that any protocol with error \( \frac{1}{32} \) must have communication \( \Omega(n) \).

We start by defining a hard distribution on inputs. View the sets \( X, Y \) as \( n \)-bit strings, by setting \( X_i = 1 \) if and only if \( i \in X \). Pick an index \( T \in [n] \) uniformly at random, and let \( X_T, Y_T \) be uniformly random and independent bits. For \( i \neq T \), sample \( (X_i, Y_i) \) to be one of \((0,0), (0,1), (1,0)\) with equal probability, and independent of all other pairs \((X_j, Y_j)\). The random sets \( X \) and \( Y \) intersect in at most 1 element, and they intersect with probability \( \frac{1}{2} \).

Let \( M \) denote the messages of a deterministic protocol whose communication complexity is \( \ell \) and probability of error is at most \( 1/32 \). We shall prove that the protocol conveys a significant amount of information about \( X_t \) or \( Y_t \), when the sets are disjoint. Let \( D \) denote the event that \( X, Y \) are disjoint. The key claim of the proof is that the following sum of informations must be large:

\[
I(X_T : M|T, X_{<T}Y_{\geq T}, D) + I(Y_T : M|T, X_{<T}, Y_{\geq T}, D) \geq \Omega(1). \tag{6.5}
\]

Before proving (6.5), we use it together with the subadditivity of mutual information to prove that \( \ell \geq \Omega(n) \).

We start by using the chain rule to prove:

**Lemma 6.15.** Let \( X = (X_1, \ldots, X_n) \) and \( Y = (Y_1, \ldots, Y_n) \) be random variables such that the \( n \) tuples \((X_1, Y_1), \ldots, (X_n, Y_n)\) are mutually independent. Let \( M \) be an arbitrary random variable. Then

\[
\sum_{i=1}^{n} I(X_i : M|X_{<i}Y_{\geq i}) \leq I(X : M|Y),
\]

and

\[
\sum_{i=1}^{n} I(Y_i : M|X_{<i}Y_{\geq i}) \leq I(Y : M|X).
\]

**Proof.** Using the chain rule:

\[
\sum_{i=1}^{n} I(X_i : M|X_{<i}Y_{\geq i}) \leq \sum_{i=1}^{n} I(X_i : MY_{<i}|X_{<i}Y_{\geq i})
\]

\[
= \sum_{i=1}^{n} I(X_i : Y_{<i}|X_{<i}Y_{\geq i}) + I(X_i : M|X_{<i}Y)
\]

\[
= \sum_{i=1}^{n} I(X_i : M|X_{<i}Y) = I(X : M|Y).
\]

The second bound is proved similarly. \( \square \)

We see that \( X, Y, M|D \) satisfy the assumptions of Lemma 6.15. Moreover \( T \) is uniform and independent of \( X, Y, M \), conditioned on

The \( n \) coordinates \((X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\) are not independent. As we discussed, this is necessary for proving a strong lower bound, but makes the proof subtle.

Here is some intuition for the validity of (6.5). If this sum of informations was 0, then \( X_T, Y_T \) are both uniform and independent conditioned on \( M, D \). This means that the probability that the sets intersect is far from both 0 and 1, which should not happen if \( M \) determines disjointness.
D. So Lemma 6.15 gives:

\[
\frac{2\ell}{n} \geq \frac{I(X : M|YD) + I(Y : M|XD)}{n} \\
\geq I(X_T : M|TX_{<T}Y_{\geq T}D) + I(Y_T : M|TX_{\leq T}Y_{> T}D) \\
\geq \Omega(1),
\]

which proves that \( \ell \geq \Omega(n) \).

It only remains to prove (6.5). Let \( Z = (M, T, X_{<T}, Y_{> T}) \). The intuition for the proof is as follows. Suppose towards a contradiction that the information is small. If we sample \( Z \) conditioned on \( D \) then with high probability the resulting value \( z \) has the property that \( p(x_t, y_t|z) \) is close to the distribution of two uniformly random bits. However, this leads to a high probability of errors for the protocol, because conditioned on \( D \) the protocol must typically output that the sets are disjoint.

For any \( z \), let \( \alpha_z \) be the statistical distance of \( p(x_t, y_t|z) \) from uniform. Let

\[
I(X_T : M|T, X_{<T}Y_{\geq T}, D) + I(Y_T : M|T, X_{\leq T}, Y_{> T}, D) = 2\gamma^4/3.
\]

Let \( \mathcal{G} \) be the set of \( z \) such that \( \alpha_z \leq 2\gamma \). We shall use Pinsker’s inequality to prove:

**Claim 6.16.** \( p(z \in \mathcal{G}) \geq \frac{1 - 4\gamma}{4} \).

Conditioned on \( Z = z \) the output of the protocol is determined. In addition, when \( z \in \mathcal{G} \), we know that \( x_t, y_t \) are close to uniform, so the probability that the protocol makes an error in this case is at least \( \frac{1}{4} - 2\gamma \). Thus, the probability that the protocol makes an error overall is at least

\[
1/32 \geq p(\text{error}) \geq p(z \in \mathcal{G}) \cdot p(\text{error}|z \in \mathcal{G}) \geq \frac{1 - 4\gamma}{4} \cdot \left( \frac{1}{4} - 2\gamma \right),
\]

proving that \( \gamma \geq \Omega(1) \).

**Proof of Claim 6.16.** Observe that since \( X_T, Y_T \) are independent and \( M \) defines a rectangle, for all \( z \),

\[
p(x_t|z) = p(x_t|z, y_t = 0) = p(x_t|z, y_t = 0, D).
\]

Let \( \alpha_{xx} \) be the statistical distance of \( p(x_t|z) \) from uniform. We have

\[
\frac{2}{3} \cdot I(X_T : M|T, X_{<T}Y_{\geq T}, Y_T = 0, D) \\
\leq I(X_T : M|T, X_{<T}Y_{\geq T}, D) \\
\leq 2\gamma^4/3,
\]

since \( M \) has at most \( \ell \) bits.

by Lemma 6.15.

by (6.5).
so by convexity and Pinsker’s inequality (Lemma 6.10),
\[ \mathbb{E}_{p(z|y_t=0)} \left[ a_{z,x} \right] \leq \sqrt{\mathbb{E}_{p(z|y_t=0)} \left[ a_{z,x}^2 \right]} \leq \gamma^4 = \gamma^2. \]

In particular,
\[ \gamma \geq p(a_{z,x} > \gamma | y_t = 0) \]
\[ \geq p(x_t = 0 | y_t = 0) \cdot p(a_{z,x} > \gamma | y_t = x_t) \]
\[ = \frac{p(a_{z,x} > \gamma | x_t = 0 = y_t)}{2}, \]

so \( p(a_{z,x} > \gamma | x_t = 0 = y_t) \leq 2\gamma. \)

Let \( a_{z,y} \) be the statistical distance of \( p(y_t|z) \) from uniform. A symmetric argument proves that the probability that \( a_{z,y} > \gamma \) conditioned on \( x_t = 0 = y_t \) is at most \( 2\gamma. \)

We use the following simple lemma:

**Lemma 6.17.** Let \( a(x,y) = a(x) \cdot a(y) \) and \( b(x,y) = b(x) \cdot b(y) \) be product distributions. Then,
\[ |a(x,y) - b(x,y)| \leq |a(x) - b(x)| + |a(y) - b(y)|. \]

**Proof.** We have
\[
|a(x,y) - b(x,y)| \\
= \sum_{x,y} |a(x,y) - b(x,y)| \\
\leq \sum_{x,y} |a(x) \cdot a(y) - b(x) \cdot b(y)| \\
+ \sum_{x,y} |b(x) \cdot a(y) - b(x) \cdot b(y)|.
\]

We can bound the first sum by
\[
\sum_{x,y} |a(x) \cdot a(y) - b(x) \cdot b(y)| \\
= \sum_y a(y) \sum_x |a(x) - b(x)| = |a(x) - b(x)|.
\]

Similarly, the second sum contributes at most \( |a(y) - b(y)| \).

Since \( p(x_t, y_t|z) \) and the uniform distribution are both product distributions,
\[
\Pr[a_z > 2\gamma | x_t = 0 = y_t] \\
\leq \Pr[a_{z,x} + a_{z,y} > 2\gamma | x_t = 0 = y_t] \]
\[ \leq \Pr[a_{z,x} > \gamma | x_t = 0 = y_t] + \Pr[a_{z,x} > \gamma | x_t = 0 = y_t] \]
\[ \leq 4\gamma. \]
Hence,

\[ p(z \in \mathcal{G}) \geq p(x_i = 0 = y_i)p(z \in \mathcal{G}|x_i = 0 = y_i) \geq \frac{1}{4} \cdot (1 - 4\gamma). \]

\[ \square \]

**Stronger Lower Bound for the Number of Rounds**

Are interactive protocols more powerful than protocols that do not have much interaction?\(^{13}\) Here we show that a protocol with many rounds can have significantly less communication than a protocol with fewer rounds. The key idea is quite similar to the round-elimination discussed earlier but

**Randomized Pointer-Chasing**

The pointer chasing problem is a natural problem where having many rounds of communication is very useful. Alice is given \( x \in [n]^n \) and Bob is given \( y \in [n]^n \). The vectors \( x, y \) define a bipartite directed graph with \( 2n \) vertices, in which each vertex has exactly one edge coming out of it. The edges emanating from the vertices on the left are specified by \( x \), and the edges from the right are specified by \( y \). An example is shown in Figure 6.8.

The graph defines a path \( z(0), z(1), z(2), \ldots \) by setting \( z(0) = 1, z(1) = x_{z(0)}, z(2) = y_{z(1)}, \) and so on. Namely, \( z(i) \) is the vertex obtained by following \( i \) edges in the graph starting at the vertex \( z(0) = 1 \). Suppose the parties want to compute whether or not \( z(k) \) is even.

Since the information about the number of rounds is lost once we move to viewing a protocol as a partition into rectangles, it seems hard to prove a separation between a few rounds and many rounds using the techniques we have seen before. A protocol with low communication will have a large rectangle, so we cannot bound the size of rectangles to get a separation between interactive protocols and non-interactive protocols.

There is an obvious deterministic protocol that takes \( k \) rounds and \( k\lceil \log n \rceil \) bits of communication—in round \( i \), the relevant player announces the value of \( z_i \).

There is also a randomized protocol with \( k - 1 \) rounds and \( O((k + n/k) \log n) \) bits of communication.\(^{14}\) In the first step, Alice and Bob use shared randomness to pick \( 10n/k \) vertices in the graph and announce the edges that originate at these vertices. Alice and Bob

\[^{13}\text{Yao, 1983; Duris et al., 1987; Halstenberg and Reischuk, 1993; and Nisan and Wigderson, 1993}\]

\[^{14}\text{Nisan and Wigderson, 1993}\]
We need to bound the error in the $k$th step by $k \epsilon$. Theorem 6.18 states that any randomized $(k - 1)$-round protocol for the $k$-step pointer chasing problem that is correct with probability $1/2 + \epsilon$ requires $\frac{C_n^2}{k^2} - k \log n$ bits of communication.

**Proof.** Let $X, Y$ be distributed uniformly and independently. Let $M_t$ be the message sent at the $t$th round of some protocol. Let $R_{k-1}$ denote the random variable

$$R_{k-1} = (M_1, \ldots, M_{k-1}, Z(1), \ldots, Z(k-1)).$$

We shall prove inductively that on average over $r_{k-1}$, the distribution $p(z(k)|r_{k-1})$ is $(k \cdot \epsilon)$-close to uniform with

$$\epsilon = \sqrt{\frac{\ell + k \log n}{n}}.$$

When $k = 1$, the statement is trivial. Suppose $k \geq 2$ and $k$ is even. The proof is similar when $k$ is odd.

We shall repeatedly use the following fact about statistical distance: If $U, V$ are independent and $p(u) \approx p(v)$, then $p(g(u)) \approx p(g(v))$ for any function $g$.

Since $R_{k-2}, M_{k-1}$ contains at most $\ell + k \log n$ bits of information, Corollary 6.12 implies that if $i$ is uniformly random in $[n]$ and independent of all other variables, then on average over $i, r_{k-2}, m_{k-1}$,

$$p(y_i|r_{k-2}) \approx p(y_i) \approx p(y_i|m_{k-1}, r_{k-2}).$$

We need to bound the error in the $k$th step by $k \epsilon$. There are two cases to consider:

**Alice sends the message $m_{k-1}$.** In this case, $p(y_i|r_{k-1}) = p(y_i|r_{k-2})$, since after fixing $r_{k-2}$, we know that $Y_i$ is independent of $M_{k-1}$ and $Z(k-1)$. By induction,

$$p(z(k-1)|r_{k-2}) \approx p(i).$$
After fixing $r_{k-2}$, if we set $u = i$, $v = z(k-1)$, and $g$ to be the function with $g(j) = y_j$, then by induction $p(u)^{(k-1)e} \approx p(v)$, and so
\[ p(g(u))^{(k-1)e} \approx p(g(v)) \]
implies that
\[ p(y_{z(k-1)}|z(k-1), r_{k-2})^{(k-1)e} \approx p(y_i|i, r_{k-2}). \]
So we have:
\[ p(z(k)|r_{k-1}) = p(y_{z(k-1)}|z(k-1), r_{k-2})^{(k-1)e} \approx p(y_i|i, r_{k-2}). \]
Combining this with (6.6) gives that $p(z(k)|r_{k-1}) \approx p(y_i)$.

**Bob sends the message $m_{k-1}$.** In this case, after fixing $R_{k-2}$, we know that $Z(k-1)$ is independent of $Y$, and therefore also of $M_{k-1}$, which is a function of $Y$. So,
\[ p(z(k-1)|r_{k-2}) = p(z(k-1)|m_{k-1}, r_{k-2}). \]
By induction,
\[ p(z(k-1)|m_{k-1}, r_{k-2})^{(k-1)e} \approx p(i). \]
So, on average over $i, r_{k-2}$ and $m_{k-1}$. After fixing $r_{k-2}$, if we set $u = i$, $v = z(k-1)$, and $g$ to be the function with $g(j) = y_j$, then by induction $p(u)^{(k-1)e} \approx p(v)$, and so
\[ p(g(u))^{(k-1)e} \approx p(g(v)) \]
implies that
\[ p(y_{z(k-1)}|z(k-1), r_{k-2})^{(k-1)e} \approx p(y_i|i, r_{k-2}). \]
This proves that
\[ p(z(k)|r_{k-1}) = p(y_{z(k-1)}|z(k-1), m_{k-1}, r_{k-2})^{(k-1)e} \approx p(y_i|i, m_{k-1}, r_{k-2}). \]
Combining this with (6.6) gives that $p(z(k)|r_{k-1}) \approx p(y_i)$.

Both of these bounds imply that $p(z_k|r_{k-1})$ is $(k \cdot e)$-close to uniform, as required. □
Stronger Bounds for Deterministic Protocols

Similar intuitions can be used to show that the deterministic communication of the pointer-chasing problem is \( \Omega(n) \) if fewer than \( k \) rounds of communication are used.\(^\text{15}\) Since the proof is too technical, we omit it from this text.

**Theorem 6.19.** Any \( k - 1 \) round deterministic protocol that computes the \( k \)-step pointer-chasing problem requires \( \frac{n}{16} - k \) bits of communication.

**Exercise 6.1**

Show that for any two joint distributions \( p(x, y), q(x, y) \) with same support, we have

\[
\mathbb{E}_{p(y)} \left[ \frac{p(x|y)}{p(x)} \right] \leq \mathbb{E}_{p(y)} \left[ \frac{p(x|y)}{q(x)} \right].
\]

**Exercise 6.2**

Suppose \( n \) is odd, and \( x \in \{0,1\}^n \) is sampled uniformly at random from the set of strings that have more 1’s than 0’s. Use Pinsker’s inequality to show that the expected number of 1’s in \( x \) is at most \( n/2 + O(\sqrt{n}) \).

**Exercise 6.3**

Let \( X \) be a random variable supported on \([n]\) and \( g : [n] \to [n] \) be a function. Prove that

\[
\Pr[X \neq g(X)] \geq \frac{H(X|g(X)) - 1}{\log n}.
\]

Use this bound to show that if Alice has a uniformly random vector \( y \in [n]^n \), Bob has an independent uniformly random input \( i \in [n] \), and Alice sends Bob a message \( M \) with that contains \( \ell \) bits, the probability that Bob guesses \( y_i \) is at most \( 1 + \ell/n \log n \).

**Exercise 6.4**

Let \( \mathcal{G} \) be a family of graphs on \( n \) vertices, such that every two graphs in the family share a clique on \( r \) vertices. Show that the number of graphs in the family is at most \( 2^{\binom{n}{2}}/2^{r-1} \).

*Hint:* Partition the graph into \( r - 1 \) parts uniformly at random and throw away all edges that do not stay within a part. Analyze the entropy of the resulting distribution on graphs.

**Exercise 6.5**

Prove the data processing inequality: if \( A - B - C \) then \( l(A : B) \geq l(A : C) \).

\(^{15}\) Nisan and Wigderson, 1993
7

Compressing Communication

Is there a way to define the information of a protocol in analogy with Shannon’s definition of the entropy of a single message? We would like to extend Shannon ideas and measure the information contained in the messages of the protocol in a way that captures the interactive nature of a communication protocol.

In this chapter, we explore how to do this for 2-party communication protocols. As we shall see, these definitions lead to several results in communication complexity that do not concern information at all.

Suppose we are working in the distributional setting, where the inputs $X, Y$ to Alice and Bob are sampled from some known distribution $\mu$, and in addition the protocol is randomized. We start with several examples to illustrate what the information of protocols ought to be.

- Consider a protocol where all the messages of the protocol are 0, no matter what the inputs are. The messages of the protocol are known ahead of time, and Alice and Bob might as well not send them. This protocol does not convey any information.

- Suppose $X, Y$ are independent uniformly random $n$ bit strings. Alice sends $X$ as her first message and Bob sends $Y$ in response. In this case, the protocol cannot be simulated with less than $2n$ bits.

- Suppose $X, Y$ are independent uniformly random $n$ bit strings. In the protocol, Alice privately samples $k$ uniformly random bits, independently of $X$, and sends them to Bob. This protocol can be simulated by a randomized communication protocol with no communication. Alice and Bob can use shared randomness to sample the $k$ bit string, so that they do not have to send it.

- Suppose $X, Y$ are independent uniformly random $n$ bit strings. Alice uses private randomness to sample a uniformly random
subset $T \subseteq [n]$ of size $k$ with $k < n/2$. Alice sends $n$ bits to Bob, where the $i$'th bit is $X_i$ if $i \notin T$, and $1 - X_i$ otherwise.

One can simulate this protocol with less than $n$ bits. Alice and Bob can use public randomness to sample a set $S \subseteq [n]$ of size $k$, as well a uniformly random $n$ bit string $R$. Alice computes the number of coordinates $i \in S$ such that $R_i \neq X_i$. If there are $t$ such coordinates, she samples a uniformly random subset $T' \subseteq [n] - S$ of size $k - t$. Setting $M_i = R_i$ for all $i \in S$, $M_i = 1 - X_i$ for all $i \in T'$ and $M_i = X_i$ for all remaining $i$, Alice sends the $n - k$ bits of $M_i$ that Bob does not already know.

For any fixed value of $X$, the string $M$ is identically distributed to how it was in the original protocol, so the simulation succeeds with communication $n - k$.

• Suppose $X, Y$ are uniformly random $n$ bit strings that are always equal. Suppose Alice sends $X$ to Bob in the first message. Then this message can be simulated with 0 communication, since Bob already knows $X$.

These examples illustrate some of the difficulties with defining the information of communication protocols. Indeed, we shall see two natural ways to define the information of a protocol. Let $R$ denote the public randomness of the protocol, and let $M$ denote the messages that result from executing the protocol. The external information\(^1\) of the protocol is defined to be

$$I(XY : MR).$$

The external information measures the amount of information about the inputs that an external observer may learn about $X, Y$ from the messages and the public randomness of the protocol. The second definition is called the internal information\(^2\) of the protocol. It is defined to be

$$I(X : M|YR) + I(Y : M|XR).$$

The internal information measures the amount of information that Alice and Bob learn about each other’s inputs from the messages and public randomness of the protocol.

Intuitively, the external information is always at least as large as the internal one—the parties know more, so they must learn less:

**Theorem 7.1.** The internal information never exceeds the external information. The two quantities are equal when $X, Y$ are independent.
Proof. Apply the chain rule to express the internal information as:

\[
I(X : M|YR) + I(Y : M|XR)
= \sum_i I(X : M_i|YRM_{<i}) + I(Y : M_i|XR M_{<i}),
\]

where here \(M_1, M_2, \ldots\) are the bits of \(M\) and \(M_{<i}\) denotes the first \(i - 1\) bits of \(M\). The definition of communication protocols ensures that \(M_{<i}\) determines whether Alice or Bob sends the next bit of the protocol. For each fixed value \(m_{<i}\), if Alice sends the next bit, then

\[
I(Y : M_i|XR M_{<i}) = 0,
\]

because \(M_i\) is determined by the variables \(XR M_{<i}\) and the private randomness of Alice. Similarly, if Bob sends the next bit in the protocol, then

\[
I(X : M_i|YR M_{<i}) = 0.
\]

Moreover, if Alice sends the next bit, then by the chain rule, we have

\[
\begin{align*}
I(X : M_i|YR M_{<i}) & \leq I(X : M_i|YR M_{<i}) + I(Y : M_i|R M_{<i}) \\
& = I(XY : M_i|R M_{<i}),
\end{align*}
\]

where the inequality is an equality when \(X, Y\) are independent of each other, because in this case \(Y\) is independent of \(M_i\) after fixing \(R, M_{<i}\). Similarly, if Bob sends the next bit, we have

\[
I(Y : M_i|XR M_{<i}) \leq I(XY : M_i|R M_{<i}),
\]

and the inequality is an equality when \(X, Y\) are independent. Putting all of these observations together, we get that the internal information can be bounded

\[
I(X : M|YR) + I(Y : M|XR)
= \sum_i I(X : M_i|YRM_{<i}) + I(Y : M_i|XR M_{<i})
\leq \sum_i I(XY : M_i|R M_{<i})
= I(XY : M|R).
\]

What we are really after is an analogy of Shannon’s Theorem 6.1 from coding theory—we want to show that information characterizes communication. Theorem 6.1 is about a particular one-round deterministic protocol—Alice gets \(X\) and needs to send \(X\) to Bob. Theorem 6.1 shows that the expected communication complexity of
this problem is characterized by the entropy of $X$. In this simple case all three quantities—internal information, external information and entropy—are equal:

$$I(X : M|Y) + I(Y : M|X) = I(XY : M) = H(X).$$

In other words, we are after an interactive generalization of Shannon’s theorem where the parties also have access to randomness. Such a statement would be immensely useful, because the quantities defining information are much easier to work with than communication complexity. The chain-rule, subadditivity and Pinsker’s inequality can all be used to understand the information complexity of communication problems. A generalization of Shannon’s theorem can also be thought of as the following questions:

Can a protocol with low information be simulated by a protocol with low communication? Is it true that every protocol with external/internal information $I$ can be simulated by a protocol with communication close to $I$?

Or,

Can we optimally compress communication protocols?

**Simulations of Protocols**

A compression of a protocol is a new protocol of smaller length that simulates the original protocol. Before we describe how to compress communication protocols, we need to first define the term simulation. Intuitively, a protocol $\sigma$ simulates a protocol $\pi$ if the messages of $\sigma$ can be translated to messages of $\pi$ in a way that induces the correct distribution.

We start by setting some notation. Suppose $X, Y$ are jointly distributed random inputs to Alice and Bob. Let $\mathcal{X}$ be the possible values of $X$ and $\mathcal{Y}$ be the possible values of $Y$. Let $R_\pi$ be the public randomness used in the protocol $\pi$, and let $M_\pi$ be the messages of $\pi$. Let $\mathcal{M}_\pi$ be the set of possible messages $M$ of $\pi$, and $\mathcal{R}_\pi$ be the set of possible values of $R$. Let $\sigma$ be a two-player protocol with public randomness $R_\sigma$, and let $M_\sigma$ be the messages of $\sigma$. Let $\mathcal{M}_\sigma$ be the set of possible messages $M_\sigma$, and $\mathcal{R}_\sigma$ be the set of possible values of $R_\sigma$.

We say that $\sigma$ simulates $\pi$ with error $\epsilon > 0$ with respect to some input distribution on $X, Y$ if there are two maps

$$F_A : \mathcal{X} \times \mathcal{M}_\sigma \times \mathcal{R}_\sigma \rightarrow \mathcal{M}_\pi \times \mathcal{R}_\pi$$

There is no input $Y$ and no randomness $R$ in this case.

We shall see that private randomness and public randomness play very different roles in this discussion.

There are several subtleties involved in deciding whether or not one protocol simulates another. We discuss these issues later on.
The proofs of the Lemma 1.8 and Lemma 7.3 are similar.
Theorem 7.4. Given any deterministic protocol \( \pi \) and a distribution \( p \) on the leaves of \( \pi \), there is a deterministic protocol \( \pi_p \) such that on input \( x, y \), the protocol \( \pi_p \) computes \( \pi(x, y) \) after communicating at most \( 2 \cdot \log_{3/2}(1/p(\pi(x, y))) + 2 \) bits.

Before proving Theorem 7.4, let us show how it implies Theorem 7.2.

Proof of Theorem 7.2. Let \( \pi \) be a protocol with no private randomness. Let \( X, Y \) be inputs to the protocol sampled from some known distribution, and let \( R \) denote the public randomness of the protocol. Let \( M \) denote the messages of the protocol.

Then observe that the external information of \( \pi \) is

\[
I(XY: M|R) = H(M|R) - H(M|RXY) = H(M|R),
\]

where the second inequality follows from the fact that \( M \) is determined by \( X, Y \) and \( R \).

Our simulating protocol will sample \( R = r \), then set \( p \) to be the distribution of \( M \) given \( R = r \), and carry out the simulation promised by Theorem 7.4. The expected number of bits communicated by the simulation is thus:

\[
\mathbb{E}_{p(x,y)} [2 \log_{3/2}(1/p(xy|r))] + 2 \leq \frac{2 \log 3}{\log 2} \cdot H(M|R = r) + 2.
\]

So in expectation, the protocol communicates \( O(H(M|R) + 1) \) bits. \( \Box \)

Proof of Theorem 7.4. Consider the following protocol for computing \( \pi(x, y) \). Let \( u \) be the node of the protocol tree promised by Lemma 7.3. As we proved in Lemma 1.4, this node corresponds to a rectangle \( R_u \) in the set of inputs. Alice and Bob communicate a single bit to determine if \( (x, y) \in R_u \). If this is the case, they continue to execute the protocol after replacing \( p(xy) = p(xy|\mathcal{E}_u) \). Otherwise, they continue, setting \( p(xy) = p(xy|\neg \mathcal{E}_u) \). The protocol terminates when \( p(\pi(x, y)) \) is supported on a single leaf.

Let \( c_{x',y'} \) denote the number of bits communicated by the protocol when the inputs are \( x', y' \). We shall prove by induction on \( \log_{3/2}(1/p(x', y')) \) that the communication of the protocol is at most

\[
2 \cdot \log_{3/2}(1/p(x', y')) + 2.
\]

The base case is when \( \log_{3/2}(1/p(x', y')) < 1 \Rightarrow p(x', y') > 2/3 \).

In this case, the protocol terminates after 2 bits of communication, since the vertex found in the application of Lemma 7.3 must be a leaf. In the general case, the protocol replaces the distribution by the distribution \( q(x, y) = p(x, y|\mathcal{E}_u) \) or \( q(x, y) = p(x, y|\neg \mathcal{E}_u) \). We have that

\[
q(x', y') = \frac{p(x', y')}{p(\mathcal{E}_u)} \geq (3/2) \cdot p(x', y'),
\]

\begin{figure}[h]
\centering
\begin{tabular}{|l|}
\hline
Input: Alice gets \( x \), Bob gets \( y \). \\
Output: The leaf \( \pi(x, y) \). \\
\hline
Set \( p \) to be the distribution of the leaf \( \pi(X, Y) \); \\
while \( p \) is the distribution on more than one leaf do \\
\begin{itemize}
\item Set \( u \) to be the node promised by Lemma 7.3; \\
\item Set \( R_u \) to be the corresponding rectangle; \\
\item Alice and Bob each communicate a bit to determine if \( (x, y) \in R_u \); \\
\item if \( (x, y) \in R_u \) then \\
\begin{itemize}
\item Set \( p \) to be the distribution \( p(xy|\mathcal{E}_u) \); \\
\end{itemize}
\item else \\
\begin{itemize}
\item Set \( p \) to be the distribution \( p(xy|\neg \mathcal{E}_u) \);
\end{itemize}
\end{itemize}
end \\
\end{itemize}
Output the unique leaf in the support of \( p \). \\
\hline
\end{tabular}
\caption{A compression algorithm for deterministic protocols.}
\end{figure}
or
\[ q(x', y') = \frac{p(x', y')}{p(-\mathcal{E}_a)} \geq (3/2) \cdot p(x', y'). \]

In either case, we have \( \log_{3/2}(1/q(x', y')) \leq \log_{3/2}(1/p(x', y')) - 1 \), so by induction, the communication of the protocol is at most
\[ 2 + 2\log_{3/2}(1/q(x', y')) + 2 \leq 2\log_{3/2}(1/p(x', y')) + 2, \]
as required.

\[ \square \]

Correlated Sampling

When the information of a protocol is much less than 1, we can use the technique of correlated sampling\(^4\) to achieve this: to compress the protocol.

Suppose we are given a protocol whose communication complexity is very large, but its internal information is very close to 0. In this case, the protocol teaches Alice and Bob almost nothing about each others inputs, so they should be able to simulate its execution without communicating, and this is what we show here.

**Lemma 7.5.** There is a protocol using public randomness and no communication with the following functionality. Suppose Alice is given as input a distribution \( p(m) \) on a set \( \mathcal{U} \), and Bob is given a distribution \( q(m) \) on \( \mathcal{U} \). After the protocol terminates, Alice holds a value \( M_A \) which is distributed according to \( p \), Bob holds a value \( M_B \) which is distributed according to \( q \), and the probability that \( M_A \neq M_B \) is at most \( 2|p - q| \).

In other words, Alice samples \( M_A \) using the public randomness, Bob samples \( M_B \) using the public randomness, and if \( p, q \) are close then they sample the same value most of the time. Hence the term correlated sampling. This simulation is exact, in the sense that Alice samples from \( p \) exactly and Bob samples from \( q \) exactly.

**Proof.** We interpret the public randomness as a sequence
\[(M_1, \rho_1), (M_2, \rho_2), \ldots\]
of independent, identically distributed samples, where \( M_i \) is a uniformly random element from the support of \( \mathcal{U} \), and \( \rho_i \) is uniformly random from \([0, 1] \). Alice sets \( m^A = M_I \) where \( I \) is the minimum index for which \( \rho_I < p(M_I) \). Similarly, Bob will set \( M^B = M_J \) where \( J \) is the minimum index such that \( \rho_J < p(M_J) \).

It remains to prove that the protocol has the desired properties. First, observe that the probability that Alice and Bob find some sample acceptable is 1.

\[^4\text{Holenstein, 2007}\]

Variants of correlated sampling also appear in probability theory in the context of coupling.
Now, we claim that $M^A$ is distributed according to $p$. A similar argument proves that $M^B$ is distributed according to $q$. Let $E$ denote the event that $\rho_1 < p(M_1)$. Think of $(M_1, \rho_1)$ as a point in the plane $\mathcal{U} \times [0,1]$ distributed uniformly at random. Imagine the graph of $p$ drawn in this plane, as in Figure 7.3. The event $E$ happens when the point $(M_1, \rho_1)$ is under the graph of $p$. The total area of the plane is $u := |\mathcal{U}|$ and the total area under $p$ is 1. Thus,

$$\Pr[E] = \frac{1}{u},$$

and

$$\Pr[M^A = m | E] = \frac{\Pr[M_1 = m, \rho_1 < p(m)]}{\Pr[E]} = \frac{(1/u) \cdot p(m)}{1/u} = p(m).$$

On the other hand, by the definition of the process, the distribution of $M^A$ conditioned on $\neg E$ is the same as the distribution of $M^A$. Thus,

$$\Pr[M^A = m] = \Pr[E] \Pr[M^A = m | E] + \Pr[\neg E] \Pr[M^A = m | \neg E]$$

$$= \Pr[E] p(m) + \Pr[\neg E] \Pr[M^A = m]$$

which implies that $\Pr[M^A = m] = p(m)$.

We now bound the probability that $M^A \neq M^B$. Let $B$ be the event that $q(M_I) < \rho_1 < p(M_I)$ or $p(M_J) < \rho_J < q(M_J)$. The event that $M^A \neq M^B$ implies the event $B$, so it suffices to bound $\Pr[B]$ from above. Denote by $F$ the event that $I = 1$ or $J = 1$. In other words, $F$ is the event that $\rho_1 < \max\{p(M_I), q(M_I)\}$. As above, $\Pr[\neg F] = \Pr[F]$. Figure 7.3: An illustration of the sampling procedure. $(M_4, \rho_4)$ is selected in this case and $M^A = M^B$. Note that $\rho_5 < q(M_5)$ but $\rho_5 > p(M_5)$. 

Figure 7.3: An illustration of the sampling procedure. $(M_4, \rho_4)$ is selected in this case and $M^A = M^B$. Note that $\rho_5 < q(M_5)$ but $\rho_5 > p(M_5)$. 

Figure 7.3: An illustration of the sampling procedure. $(M_4, \rho_4)$ is selected in this case and $M^A = M^B$. Note that $\rho_5 < q(M_5)$ but $\rho_5 > p(M_5)$. 

Figure 7.3: An illustration of the sampling procedure. $(M_4, \rho_4)$ is selected in this case and $M^A = M^B$. Note that $\rho_5 < q(M_5)$ but $\rho_5 > p(M_5)$.
which implies that $\Pr[B] = \Pr[B|F]$. Finally,

$$
\Pr[B|F] = \frac{\Pr[\min\{p(M_1), q(M_1)\} \leq \rho_1 < \max\{p(M_1), q(M_1)\}]}{\Pr[\rho_1 < \max\{p(M_1), q(M_1)\}]}
= \frac{\sum_m |p(m) - q(m)|}{\sum_m \max\{p(m), q(m)\}}
\leq \frac{\sum_m |p(m) - q(m)|}{\sum_m (p(m) + q(m))/2}
= 2 \cdot |p - q|.
$$

\[ \square \]

**Compressing a Single Round of Communication**

**Next, let us consider compression of one-round protocols.** As we shall see, even this seemingly simple task is not trivial.

Since the definition of information of a protocol involves conditioning on the public randomness, it is no loss of generality to assume that the protocols we consider do not have public randomness.

**External Compression**

Suppose we would just like to compress the first message in a protocol down to its external information. If the message $M$ is sent by Alice, who has the input $X$, and Bob has the input $Y$, then the external information can be expressed as

$$
I(XY : M) = I(X : M) + I(Y : M|X)
= I(X : M).
$$

In analogy with Theorem 6.1, we prove that there is a way to simulate\(^5\) the sending of the message $M$ using $I(X : M) + O(\log I(X : M))$ bits of communication in expectation. The theorem follows from the following stronger fact:

**Theorem 7.6.** Suppose Alice knows two distributions $p, q$ over the same set $\mathcal{U}$, and Bob knows $q$. There is a protocol for Alice and Bob to sample an element according to $p$ using

$$
\frac{p}{q} + 2\log \left( \frac{p}{q} \right) + O(1)
$$

bits of communication in expectation. This is a one-round protocol in which Alice sends Bob a single message.

As a corollary, we get

\(^5\) Harsha et al., 2007; and Braverman and Garg, 2014

The factor 2 before the log in the theorem can be replaced by 1, and 1 is sharp up to the additive constant (see Exercise 7.3).

This is a perfect simulation—the players sample exactly from $p$.\n
Since after fixing $X$, the variables $Y$ and $M$ are independent.
Corollary 7.7. Alice and Bob can use public randomness to simulate sending $M$ with expected communication $I(X : M) + 2 \log I(X : M) + O(1)$. This simulation is in one round, external and without error.

The protocol we use is inspired by the correlated sampling idea. The public random tape will consist of a sequence of i.i.d. samples $(M_1, \rho_1), (M_2, \rho_2), \ldots$, where each $M_i$ is a uniformly random element from the support of the messages, and $\rho_i$ is a uniformly random number from $[0, 1]$. Given this public randomness, Alice finds the minimum index $R$ such that $\rho_R < p(m_R)$. As we proved when we analyzed correlated sampling, the value $M_R$ has exactly the correct distribution. Unfortunately, communicating $R$ can be too expensive, so Alice cannot simply send $R$ to Bob. Instead, Alice computes the positive integer

$$T = \left\lceil \frac{\rho_R}{q(M_R)} \right\rceil,$$

and sends $T$ to Bob. Given $T$, Alice and Bob can both compute the set

$$S_T = \left\{ j : T = \left\lceil \frac{\rho_j}{q(M_j)} \right\rceil \right\}.$$

Alice also sends Bob the number $K$ for which $R$ is the $K$'th smallest element of $S_T$.

The intuition for this protocol is that $\log T$ is a rough estimate of $\log \frac{p(M)}{q(M)}$ for the message $M$, and with it Alice tells Bob the $p$-to-$q$ ratio of the sample that they are after. Knowing $T$, the players focus only on the part of this universe with constant $p$-to-$q$ ratio, and this takes a constant number of experiments on average.

To analyze the expected communication of the protocol, we need two basic claims. The first claim, whose proof we sketch, is used to encode the integers sent in the protocol.

Figure 7.5: The sampling procedure of Theorem 7.6. Here $T$ is 3 and the sampled point is the 3'rd point of $S_T$.

To prove the corollary, if $r(x, m)$ denotes the joint distribution of $X, M$, let $p(m) = r(m|x)$ and $q(m) = r(m)$. Jensen's inequality yields that the expected communication of the resulting protocol is at most $I(X : M) + 2 \log I(X : M) + O(1)$.

What is the expectation of $\log R$?

**Input:** Alice knows $p, q$, Bob knows $q$.

**Output:** $M$ distributed according to $p$.

**P. Rand:** $(M_1, \rho_1), \ldots$, uniformly and independently from the universe and $[0, 1]$.

Alice sets $R$ to be the minimum index such that $\rho_R < p(m_R)$; Alice computes $T = \left\lceil \frac{\rho_R}{q(M_R)} \right\rceil$, and sets $K$ to be the smallest integer such that $R$ is the $K$'th element of $\left\{ j : T = \left\lceil \frac{\rho_j}{q(M_j)} \right\rceil \right\}$; Alice sends Bob $T, K$.

Figure 7.4: Compressing a single round of communication to its internal information.
Claim 7.8. One can encode all positive integers in such a way that at most \( \log z + 2 \log \log z + O(1) \) bits are used to encode the integer \( z \).

Proof Sketch. A naive encoding would take \( 2\lceil \log z \rceil \) bits, by repeating every bit of the binary representation of \( z \) twice. The first bit specifies whether or not there is another bit coming or not. To get a better bound, first send the integer \( \lceil \log z \rceil \) using the naive encoding, and then send \( \lceil \log z \rceil \) more bits to encode \( z \).

To argue that the expected length of \( T \) is small, we need the following claim:

Claim 7.9. For any two distribution \( p(m), q(m) \), the contribution of the terms with \( p(m) < q(m) \) to the divergence is at least \(-1\):

\[
\sum_{m: p(m) < q(m)} p(m) \log \frac{p(m)}{q(m)} \geq -1.
\]

Now, we bound the expected number of bits required to transmit \( T \). By Claim 7.8, this is at most

\[
\mathbb{E} [\log T + 2 \log \log T + O(1)] \leq \mathbb{E} [\log T] + 2 \log \mathbb{E} [\log T] + O(1),
\]

where the inequality follows from Jensen’s inequality. By Claim 7.9, we can bound

\[
\mathbb{E} [\log T] \leq \sum_m p(m) \log \left[ \frac{p(m)}{q(m)} \right] = \sum_{m: p(m) > q(m)} p(m) \left( 1 + \log \frac{p(m)}{q(m)} \right) \leq 1 + \frac{p(m)}{q(m)} - \sum_{m: p(m) < q(m)} p(m) \log \frac{p(m)}{q(m)} \leq \frac{p(m)}{q(m)} + 2.
\]

So the expected number of bits used to transmit \( T \) is at most

\[
\frac{p(m)}{q(m)} + 2 \log \left( \frac{p(m)}{q(m)} \right) + O(1).
\]

It only remains to bound the number of bits required to transmit \( K \). We shall prove that \( \mathbb{E} [K] \leq 2 \), which implies that the expected number of bits required to transmit \( K \) is a constant. Indeed, by convexity, we have \( \mathbb{E} [\log K + \log \log k] \leq \log \mathbb{E} [K] + \log \log \mathbb{E} [K] \).

Consider the event \( A \) that \( \rho_1 \leq p(M_1) \). Define the random variable

\[
Z = \begin{cases} 
1 & \text{if } 1 \in S_T, \\
0 & \text{otherwise.}
\end{cases}
\]

By encoding we mean a protocol in which Alice gets as input \( z \) and the outcome of the protocol determines \( z \). Proof of Claim 7.9: Let \( E \) denote the subset of \( m \)'s for which \( p(m) < q(m) \). Then we have

\[
\sum_{m \in E} p(m) \log \frac{p(m)}{q(m)} \geq -p(E) \cdot \sum_{m \in E} p(m|E) \log \frac{q(m)}{p(m)} \geq -p(E) \cdot \log \frac{E}{p(E)} \geq p(E) \cdot \log p(E).
\]

For \( 0 \leq x \leq 1 \), \( x \log x \) is maximized when its derivative is 0: \( \log x + \log x = 0 \). So the maximum is attained at \( x = 1/e \), proving that \( p(E) \log p(E) \geq \frac{-\log e}{p(E)} > -1 \).
When \( A \) happens, \( K = 1 \). On the other hand, the distribution of \( K - Z \) conditioned on \( \neg A \) is the same as the distribution of \( K \), so \( \mathbb{E}[K] = \mathbb{E}[K - Z|\neg A] \). Thus,

\[
\mathbb{E}[K] = \Pr[A] + \Pr[\neg A](\mathbb{E}[K] + \mathbb{E}[Z|\neg A])
\]

which implies

\[
\mathbb{E}[K] = 1 + \frac{\Pr[\neg A] \cdot \mathbb{E}[Z|\neg A]}{\Pr[A]}
\]

Now, let \( u \) be the size of the universe \( \mathcal{U} \). Then

\[
\Pr[A] = \frac{1}{u} \sum_m p(m) = \frac{1}{u}
\]

and

\[
\mathbb{E}[Z|\neg A] \cdot \Pr[\neg A] \leq \Pr[1 \in S_T]
\]

\[
= \Pr[(T - 1)q(M_1) < \rho_1 \leq Tq(M_1)]
\]

\[
= \mathbb{E} \left[ \frac{1}{u} \sum_m Tq(m) - (T - 1)q(m) \right] = \frac{1}{u}.
\]

Thus we get

\[
\mathbb{E}[K] \leq 2.
\]

**Internal Compression**

Now suppose we wish to compress a single message \( M \) sent from Alice who knows \( X \) to Bob who knows \( Y \) down to its internal information

\[
I(X : M|Y) + I(Y : M|X) = I(X : M|Y).
\]

This is strictly harder than the problem for external information, and when \( X, Y \) are independent the two problems are the same.

**Theorem 7.10.** Suppose Alice knows two distributions \( p, q \) over the same set \( \mathcal{U} \), and Bob knows \( q \). For every \( \epsilon > 0 \), there is a protocol for Alice to sample an element according to the distribution \( p \) while communicating at most

\[
\frac{p}{q} + O \left( \sqrt{\frac{p}{q}} \right) + \log(1/\epsilon) + O(1)
\]

bits in expectation such that Bob also computes the same sample, except with probability at most \( \epsilon \).

The additive square-root term in this theorem is not sharp. The proof can be altered to yield other bounds.
As a corollary, we get

**Corollary 7.11.** Alice and Bob can use public randomness to simulate sending $M$ with expected communication at most

$$I(X : M|Y) + O(\sqrt{I(X : M|Y)}) + \log(1/\epsilon) + O(1).$$

This simulation has several rounds, is internal and has error $\epsilon$.

We shall use very similar ideas to obtain a protocol as in the previous sections. However, our simulating protocol will be interactive rather than a one-round protocol, and the simulation is not perfect—Bob does not sample exactly from the correct distribution.

Alice and Bob again use public randomness to sample a sequence of points $(M_1, \rho_1), (M_2, \rho_2), \ldots$, where each $M_i$ is a uniformly random element of the support, and $\rho_i$ is a uniformly random number in $[0,1]$. As before, Alice picks the smallest index $R$ such that $\rho_R < p(M_R)$. As before, Alice would like to send Bob enough data for him to be able to recover $M_R$. Specifically, Alice would really like to compute the ratio $\lceil \frac{\rho_R}{q(M_R)} \rceil$. Unfortunately, Alice does not know $q$, so she cannot compute this ratio without interacting with Bob. Instead, Alice and Bob try to approximate this ratio. To do this, they gradually increase a threshold until the threshold is larger than this ratio. They are able to locate the correct time to stop increasing the threshold using hashing, which eventually yields some probability of error.

Before describing the protocol, we set some notation. For each index $i$, let $H(i) = (H(i)_1, H(i)_2, \ldots)$ be an infinite sequence of uniformly random bits, sampled publicly. The sequence $H(i)$ is thought of as a hash function of $i$. For a positive integer $k$, let $\rho_R = \rho(H(i)_1 H(i)_2 \cdots H(i)_k \cdots)$.
\[
Q_k = \left\{ j : 2^{k^2} \geq \frac{\rho_j}{q(M_j)} \right\}.
\]

\(Q_k\) is the set of indices with \(\rho\)-to-\(q\) ratio at most \(2^{k^2}\). For positive integers \(i, j\), let

\[
g(i, j) = \min\{\ell \in Q_i : H(\ell) \leq j = H(R) \leq j\}.
\]

Intuitively, this is the best candidate for \(R\) among elements of \(Q_i\), with respect to the hash values in the first \(j\) positions.

As we have shown in the correlated sampling section, \(M_R\) is correctly distributed. Alice always outputs \(M_R\). Bob’s output is determined after a number of rounds of communication. In round \(k\),

Alice sends Bob all the bits of \(H(R) \leq k^2 + \log(1/\epsilon)\) that she has not already sent him. Bob computes the values \(g(i, j)\) for each \(i \leq 2^{k^2}\) and \(j \leq k^2\).

If there is any index \(s \leq k\) such that \(g(s, k^2 + \log(1/\epsilon)) = g(s, (k - 1)^2)\), then Bob stops the protocol and outputs \(M_{g(s, (k - 1)^2)}\) for the smallest such index \(s\). If there is no such index \(s\), then Bob sends Alice a bit to indicate that the protocol should continue, and the parties go to the next round.

Before proving that the protocol achieves its goal, we provide some intuition. If \(k\) is small so that \(Q_k\) does not contain \(R\) then the hashes will be an evidence that \(R\) does not contain \(R\). If \(k\) is large so that \(Q_k\) contains \(R\) then the probability that any index will remain consistent with the hashes for many hashes is small.

If \(T\) is large enough so that \(Q_T\) contains \(R\), then all indices of \(Q_T\) that are less than \(R\) will eventually become inconsistent with \(H(R)\). If \(T\) is smaller, then the probability that any index will remain consistent with the hashes for many hashes is small.

First, let us analyze the probability that the protocol makes an error. The players’ outputs are different only if \(g(s, k^2 + \log(1/\epsilon)) = g(s, (k - 1)^2) \neq R\) for some integers \(k\) and \(s \leq k\). The probability of this event, for fixed \(k, s\), is at most

\[
2^{-(k^2 + \log(1/\epsilon) - 1 - (k - 1)^2)} \leq 2^{-2k - \log(1/\epsilon)}.
\]

Thus, by the union bound, the probability of an error is at most

\[
\sum_{k=1}^{\infty} \sum_{j=1}^{k} 2^{-2k - \log(1/\epsilon)} = \sum_{k=1}^{\infty} k \cdot 2^{-2k - \log(1/\epsilon)} < \epsilon.
\]

To analyze the expected communication of the protocol, let \(T\) be the smallest positive integer such that \(2T^2 \geq \frac{\rho_k}{q(M_k)}\). Let \(H\) be the minimum integer such that \(g(T, H) = R\) and \(H \geq T^2\).
We shall show that the expected value of \( H \) is small.

**Claim 7.12.** \( \mathbb{E}[H] \leq \frac{p(m)}{q(m)} + 3 \sqrt{\frac{p(m)}{q(m)}} + O(1). \)

Before proving the claim, we show how it completes the proof. For every \( k \) so that \((k - 1)^2 \geq H\), the protocol certainly terminates by round \( k \), since

\[
g(T, k^2 + \log(1/\epsilon)) = R = g(T, (k - 1)^2).
\]

The smallest value of \( k \) satisfying this inequality is at most \( \sqrt{H} + 2 \). The number of bits communicated up to round \( k \) by Alice is at most \( k^2 + \log(1/\epsilon) \), and by Bob is at most \( k \). Hence, the expected communication of the protocol is at most

\[
\mathbb{E} \left[ \left( \sqrt{H} + 2 \right)^2 + \sqrt{H} + 2 + \log(1/\epsilon) \right]
\]

\[
\leq \mathbb{E}[H] + 5 \sqrt{\mathbb{E}[H] + \log(1/\epsilon)} + O(1) \quad \text{by convexity.}
\]

\[
\leq \frac{p(m)}{q(m)} + O \left( \sqrt{\frac{p(m)}{q(m)}} \right) + \log(1/\epsilon) + O(1).
\]

**Proof of Claim 7.12.** We start by proving that

\[
\mathbb{E}[H | T, R] \leq 2 + \log \left( 1 + \frac{2 T^2 R}{u - 1} \right), \quad (7.1)
\]

where \( u > 1 \) is the size of the universe. To this end, let \( L \) denote the number of elements of \( Q_T \) that precede \( R \). We prove the stronger statement that

\[
\mathbb{E}[H | T, R, L] \leq 3 + \log(1 + L). \quad (7.2)
\]

To see that (7.2) implies (7.1), use Jensen’s inequality and that

\[
\mathbb{E}[L | T, R] \leq R \cdot \Pr[\rho_1 \leq 2 T^2 q(M_1) | \rho_1 \geq p(M_1)]
\]

\[
= R \cdot \frac{1}{u} \sum_m \Pr[p(m) \leq \rho_1 \leq 2 T^2 q(m)]
\]

\[
\leq \frac{2 T^2 R}{u - 1}.
\]

We now prove (7.2). For \( L = 0 \), we indeed have \( \mathbb{E}[H | T, R, L = 0] = 1 \leq 3 \). So we can assume that \( L > 0 \). For every integer \( h \geq 1 \),

\[
\Pr[H \geq h + 1] = 1 - \Pr[H \leq h] = 1 - \left( 1 - 2^{-h} \right)^L \leq 1 - e^{-L 2^{-h+1}}.
\]

since \( e^{-2x} \leq 1 - x \) for \( 0 \leq x \leq 1/2 \).
So for $h > 1 + \log L$ we have $\Pr[H \geq h + 1] \leq 1 - e^{-2^{-h}}$. Hence,

$$
\mathbb{E}[H] = 1 + \sum_{h=1}^{\infty} \Pr[H \geq h + 1] \\
\leq 2 + \log L + \sum_{h>0} 1 - e^{-2^{-h}} \\
\leq 2 + \log L + \sum_{h>0} 2^{-h} = 3 + \log L.
$$

Now, since

$$
\mathbb{E}\left[\log \frac{u - 1 + 2^T R}{u - 1}\right] \leq \mathbb{E}\left[\log \frac{2^T (u - 1 + R)}{u - 1}\right] \\
= \mathbb{E}[T^2] + \mathbb{E}\left[\log \frac{u - 1 + R}{u - 1}\right],
$$

by (7.4), the expectation of $H$ is at most

$$
\mathbb{E}[T^2] + \mathbb{E}\left[\log \frac{u - 1 + R}{u - 1}\right] + 3. \quad (7.3)
$$

Bound the first term in (7.3) as follows. Let $T' = \max\{\log \frac{\rho_1}{\rho_{M_1}}, 0\}$ so that

$$
T^2 \leq \left(\sqrt{T'} + 1\right)^2 = T' + 2\sqrt{T'} + 1.
$$

By convexity, we can bound

$$
\mathbb{E}[T^2] \leq \mathbb{E}[T'] + 2\mathbb{E}[T'] + 1.
$$

Bound $\mathbb{E}[T']$ as follows. As in previous sections, $\Pr[R = 1] = \frac{1}{n}$ and the distribution of $T'$ conditioned on the event $R > 1$ is the same as the distribution of $T'$. Hence,

$$
\mathbb{E}[T'] = \Pr[R = 1] \mathbb{E}[T'|R > 1] + \Pr[R > 1] \mathbb{E}[T'|R > 1] \\
= \Pr[R = 1] \mathbb{E}[T'|R = 1] + \Pr[R > 1] \mathbb{E}[T'].
$$

Since conditioned on $R = 1$, $T' > 0$ and $M_1 = m$, the variable $\rho_1$ is uniform in the interval $(q(m), p(m)]$, therefore

$$
\mathbb{E}[T'] = \mathbb{E}[T'|R = 1] \\
= \frac{1}{\Pr[R = 1]} \sum_{m:p(m) > q(m)} \Pr[M_1 = m] \int_{q(m)}^{p(m)} \log \frac{\rho}{q(m)} d\rho \\
\leq \sum_{m:p(m) > q(m)} p(m) \log \frac{p(m)}{q(m)} \\
= \frac{p(m)}{q(m)} \sum_{m:p(m) < q(m)} p(m) \log \frac{p(m)}{q(m)} \\
\leq \frac{p(m)}{q(m)} + 1.
$$
This gives:
\[
\mathbb{E}[T^2] \leq \frac{p(m)}{q(m)} + 2 \sqrt{\frac{p(m)}{q(m)}} + 3.
\]

Finally, bound the second term in (7.3) as follows. Observe that by Jensen’s inequality
\[
\mathbb{E} \left[ \log \frac{u - 1 + R}{u - 1} \right] \leq \log \frac{u - 1 + \mathbb{E}[R]}{u - 1}.
\]

Again, since \( \Pr[R = 1] = \frac{1}{u} \) and the distribution of \( R \) conditioned on \( R > 1 \) is the same as the distribution of \( R + 1 \), we have
\[
\mathbb{E}[R] = \frac{1}{u} + \left( 1 - \frac{1}{u} \right) (\mathbb{E}[R] + 1).
\]

So \( \mathbb{E}[R] = u \) and
\[
\mathbb{E} \left[ \log \frac{u - 1 + R}{u - 1} \right] \leq 3.
\]

\( \square \)

Internal Compression of Protocols

Here we describe how to compress any protocol with low internal information\(^6\). This is the most general compression we describe. The generality comes with a cost—the simulating protocol is not as efficient as the simulations we saw earlier.

Suppose we are given inputs \( X, Y \) sampled according to some known distribution, and a protocol \( \pi \) with public randomness \( R \) and messages \( M \).

As usual, it is enough to show how to compress protocols that only use private randomness. This is because for each fixing of the public randomness \( R = r \), if the internal information cost is \( I_r \), and we obtain a simulating protocol with communication \( \sqrt{I_r C \log C} \), then by convexity, the expected number of bits communicated for average \( R \) is
\[
\mathbb{E}_{p(r)} \left[ \sqrt{I_r C \log C} \right] \leq \sqrt{\mathbb{E}_{p(r)} [I_r] C \log C}
\]
\[
= \sqrt{I C \log C}.
\]

Suppose the length of the protocol is \( C \) and its internal information is
\[
I = I(X : M|Y) + I(Y : M|X).
\]

We shall prove:
Theorem 7.13. For every $\epsilon > 0$, one can simulate any such protocol $\pi$ with a protocol of length $O\left(\sqrt{IC\log(C/\epsilon)}\right)$. This simulation is internal and has error $\epsilon$.

In a nutshell, the idea for the proof is that Alice and Bob use correlated sampling to repeatedly guess the bits of the messages in the protocol without communicating. Typically, not all of their guesses are correct so they repeatedly communicate a few bits to fix the errors.

Proof. Without loss of generality we assume that the protocol tree is a full binary tree of depth $C$. To carry out the simulation, we use correlated sampling, but since we will be sampling bits and not elements of a large universe, the sampling procedure is particularly

An interesting property of the simulating protocol $\sigma$ is that an outside observer can not interpret the messages of $\sigma$ as messages of $\pi$. In other words, the simulation is not external.
simple. For each prefix $m_{<i}$ of messages, define the number
\[ \gamma(m_{<i}) = p(M_i = 1 | xy m_{<i}). \]

These numbers define the correct distribution that our simulation protocol attempts to compute: for all $m$,
\[ \prod_{i=1}^{C} \gamma(m_{<i})^{m_i} (1 - \gamma(m_{<i}))^{1-m_i} = \prod_{i=1}^{C} p(m_i | xy m_{<i}) = p(m | xy). \]

Here is a useful way to sample from this distribution. Let $\rho_1, \ldots, \rho_C$ be independent, identically distributed random variables from the interval $[0,1]$, sampled uniformly at random. Now, for each $i$, set $M_i = 1$ if $\rho_i < \gamma(M_{<i})$, and set $M_i = 0$ otherwise. It follows that $\Pr[M = m] = p(m | xy)$; that is, $M$ has exactly the same distribution as the one we want to sample from.

For now, fix $\rho_1, \ldots, \rho_C$ and the obtained $m$. The goal of the players is to communicate in order to compute $m$. Although Alice and Bob cannot compute $\gamma(m_{<i})$ without communicating, Alice can compute the number
\[ \gamma^A(s_{<i}) = p(M_i = 1 | x, M_{<i} = s_{<i}) \]
and Bob can compute the number
\[ \gamma^B(s_{<i}) = p(M_i = 1 | y, M_{<i} = s_{<i}), \]
for all $s$ and $i$. Moreover, if it is Alice’s turn to speak to send $M_i$ when $M_{<i} = s_{<i}$, then $\gamma^A(s_{<i}) = \gamma(s_{<i})$, and if it is Bob’s turn to speak, then $\gamma^B(s_{<i}) = \gamma(s_{<i})$. Thus, we have:

**Claim 7.14.** Either $\gamma(m_{<i}) = \gamma^A(m_{<i})$, or $\gamma(m_{<i}) = \gamma^B(m_{<i})$.

Alice and Bob use $\gamma^A$ and $\gamma^B$ as proxies for $\gamma$ in order to guess $m$. Alice computes $m^A$ by setting $m^A_i = 1$ if and only if $\rho_i < \gamma^A(m_{<i})$, and Bob computes $m^B$ by setting $m^B_i = 1$ if and only if $\rho_i < \gamma^B(m_{<i})$. Of course, $m^A$ and $m^B$ are likely to be quite different. However, by Claim 7.14, if they happen to be the same, then they must both be equal to $m$.

To compute $m$, the players communicate. They start by finding the first index $i$ where $m^A_i \neq m^B_i$. Using the results of Exercise 3.1, this takes $O(\log(C/\delta))$ communication, if the probability of making an error is $\delta > 0$. If $m^A_{<i}$ dictates that Alice was supposed to send the $j$’th bit, then Bob sets $m^B_j = m^A_j$ and recomputes the rest of $m^B$ using $\rho_{j+1}, \ldots, \rho_C$. Otherwise Alice sets $m^A_j = m^B_j$ and recomputes $m^A$. They repeat this procedure until $m^A = m^B = m$.

The protocol is an internal simulation of $\pi$. The simulation error is at most $C\delta := \epsilon/2$. It remains to bound from above the length of the

---

**Figure 7.9:** Compressing protocols to their internal information.
simulation. This is done via Pinsker’s inequality, which allows us to bound the expected number of errors the players need to correct in terms of $I$.

We say that the protocol made a mistake at $i$ if during its execution $m^A_i$ was found to be not equal to $m^B_i$. This happens exactly when $\rho_i$ lies in between the numbers $\gamma^A(m_{<i})$ and $\gamma^B(m_{<i})$. Note that when this happens $m_{<i}$ is distributed exactly as in $\pi$. So given that $m$ is sampled by the protocol, the probability that there is a mistake at $i$ is at most

$$\mathbb{E}_{p(xym)} \left[ |\gamma^A(m_{<i}) - \gamma^B(m_{<i})| \right]$$

$$= \mathbb{E}_{p(xym)} \left[ |p(m_i = 1|x_{<i}) - p(m_i = 1|y_{<i})| \right].$$

Now for each fixing of $m_{<i}$, if the $i$’th message is supposed to be sent by Alice, we have

$$\mathbb{E}_{p(xy|m_{<i})} \left[ |p(m_i = 1|x_{<i}) - p(m_i = 1|y_{<i})| \right]$$

$$= \mathbb{E}_{p(xy|m_{<i})} \left[ |p(m_i = 1|x_{<i}) - p(m_i = 1|y_{<i})| \right]$$

$$\leq \sqrt{I(X : M_i|Ym_{<i})}, \quad \text{By Corollary 6.11.}$$

and if the $i$’th bit was to be sent by Bob, then we have

$$\mathbb{E}_{p(xy|m_{<i})} \left[ |p(m_i = 1|x_{<i}) - p(m_i = 1|y_{<i})| \right]$$

$$= \mathbb{E}_{p(xy|m_{<i})} \left[ |p(m_i = 1|x_{<i}) - p(m_i = 1|y_{<i})| \right]$$

$$\leq \sqrt{I(Y : M_i|XM_{<i})}.$$

In either case, by convexity, the expected number of mistakes is at most

$$\sum_{i=1}^C \sqrt{I(X : M_i|YM_{<i}) + I(Y : M_i|XM_{<i})}$$

$$\leq \sqrt{C} \cdot \sqrt{\sum_{i=1}^C I(X : M_i|YM_{<i}) + I(Y : M_i|XM_{<i})} \quad \text{by the Cauchy-Schwartz inequality.}$$

$$= \sqrt{C} \cdot \sqrt{I(X : M|Y) + I(Y : M|X)} = \sqrt{IC}.$$

The communication of the protocol is $O(\sqrt{IC \log(C/\epsilon)})$ in expectation. By Markov’s inequality, the probability that the communication exceeds $2/\epsilon$ times this number is at most $\epsilon/2$, as claimed. \hfill \Box
**Direct Sums in Randomized Communication Complexity**

We already proved a direct sum theorem for deterministic communication complexity in Section 1 (Theorem 1.33). Here we prove analogous results for randomized communication complexity.

**Theorem 7.15.** If the randomized communication complexity of $g$ is $c$, then the randomized communication complexity of $g^k$ is at least $\Omega(c\sqrt{k}/\log c)$.

The theorem is proved using the compression from the previous section. Roughly speaking, the proof shows that a protocol for $g^k$ of length $\ell$ can be interpreted as a protocol for $g$ with information $I \leq \ell/k$. The intuition is that the information contained in the $\ell$ bits of the protocol are distributed over the $k$ copies of $g$, giving $\ell/k$ bits of information for an average copy of $g$. We can now compress it to a protocol for a single copy of $g$ of length $\approx \sqrt{I\ell} = \ell/\sqrt{k}$, which means that $c \lesssim \ell/\sqrt{k}$.

**Proof.** Suppose there is a randomized protocol computing $g^k$ in the worst case, with $\ell$ bits of communication, and success probability at least $3/4$.

By the minimax principle, Theorem 3.3, there is a distribution $\mu$ on inputs to $g$ such that every deterministic protocol that computes $g$ with less than $c$ bits of communication must make an error with probability more than $1/3$ over inputs from $\mu$. Let

$$(X_1, Y_1), (X_2, Y_2), \ldots, (X_k, Y_k)$$

be $k$ independent inputs sampled according to the distribution $\mu$.

We can now conclude that there is a deterministic protocol $\pi$ computing $g^k$ on with error less than $1/4$ on inputs from $\mu^k$.

Consider the following protocol for computing $g$ on inputs from the distribution $\mu$. Alice and Bob get inputs $(X', Y')$, sampled from $\mu$. The players sample $J \in [k]$ uniformly at random and sample $X_{<J}Y_{>J}$ using public randomness according to the marginal distributions of $\mu$. Alice privately samples $X_{>J}$ conditioned on $Y_{>J}$ and Bob privately samples $X_{<J}$ conditioned on $X_{<J}$, according to the conditional marginal distributions of $\mu$. Finally, they set $(X_J, Y_J) = (X', Y')$. The players now run the protocol $\pi$ on the inputs $X = (X_1, \ldots, X_k)$ and $Y = (Y_1, \ldots, Y_k)$ they thus generated.

A crucial observation is that the inputs $X, Y$ the players generated in the above protocol are distributed as $k$ independent copies of $\mu$. Now, let $M$ denote the messages of this protocol when the inputs are
sampled as above. By Lemma 6.15, we have
\[ \sum_{i=1}^{k} I(X_i : M|X_{<i}Y_{\geq i}) \leq I(X : M|Y) \leq \ell \]
and
\[ \sum_{i=1}^{k} I(Y_i : M|X_{\leq i}Y_{> i}) \leq I(Y : M|X) \leq \ell. \]
This means that the internal information cost of the protocol is
\[ I(X' : M|Y'R) + I(Y' : M|X'R) \]
\[ = I(X' : M|Y'|X_{<j}Y_{\geq j}) + I(Y' : M|X'|X_{<j}Y_{\geq j}) \]
\[ = \sum_{j=1}^{k} \left( I(X_j : M|X_{<j}Y_{\geq j}) + I(Y_j : M|X_{\leq j}Y_{> j}) \right) \leq \frac{2\ell}{k}. \]
The length of this protocol is \( O(\ell) \). Hence, by Theorem 7.13, the protocol can be simulated with arbitrarily small constant error and communication
\[ O(\sqrt{\ell} \cdot \ell/k \log \ell) = O(\ell \sqrt{1/k \log \ell}). \]
Since this quantity must be larger than \( c \), we get that
\[ \ell \geq \Omega(c \sqrt{k} / \log c). \]
\[ \square \]

**Other Methods to Compress Protocols**

**Compression of communication protocols** is a relatively new line of research that is still evolving. For this reason and for the clarity of exposition, we have not included all known compression-related results in this chapter. We conclude this chapter with a survey of these results.

The first result we state is an external compression\(^8\):

**Theorem 7.16.** For every \( \epsilon > 0 \), one can simulate any protocol with external information \( I \) and length \( C \) by a protocol of length \( O\left( \frac{\log C}{\epsilon^2} \right) \). This simulation is external and has error \( \epsilon \).

Later on, this compression was improved to be independent of the communication length, for the special case of product distributions\(^9\):

**Theorem 7.17.** For every \( \epsilon > 0 \), one can simulate any protocol with internal information \( I \) by a protocol of length \( O\left( \frac{1}{\epsilon} \log \frac{1}{\epsilon} \right) \) when the inputs \( X, Y \) are independent. This simulation is external and has error \( \epsilon \).

---

\(^8\) Barak et al., 2010

\(^9\) Kol, 2016; and Sherstov, 2016

Recall that for product distributions, the external and internal informations are equal.
For general compression, the following result gives a bound that is independent of the communication of the initial protocol:\(^\text{10}\):

**Theorem 7.18.** For every \( \epsilon > 0 \), one can simulate any protocol with internal information \( I \) by a protocol of length \( 2^{O(I/\epsilon)} \). This simulation is internal and has error \( \epsilon \).

If we measure the information learnt by Alice as \( I_A = I(Y : M|XR) \), and by Bob as \( I_B = I(X : M|YR) \), then one can compress the protocol to take advantage of a large asymmetry between these quantities:

**Theorem 7.19.** If Alice learns information \( I_A \) and Bob learns information \( I_B \), then the protocol can be simulated by a protocol of communication \( I_A \cdot 2^{O(I_B)} \). If the total communication of the original protocol is \( C \), one can carry out the simulation using communication proportional to

\[
I_A + C^{3/4} I_B^{1/4} \log C + \sqrt{C^{1/2} \cdot I_B^{1/2}}.
\]

When the protocols have only public-randomness, we have seen an optimal external compression. In this case, a different internal compression is known:\(^\text{11}\):

**Theorem 7.20.** For every \( \epsilon > 0 \), one can simulate any protocol with no private randomness, internal information \( I \) and length \( C \) by a protocol of length \( O \left( \frac{I^2 \log \log C}{\epsilon^2} \right) \). This simulation is internal and has error \( \epsilon \).

A simulation of length \( O \left( \frac{\epsilon}{\sqrt{\epsilon}} \log(C/\epsilon) \right) \) is also known in this case:\(^\text{12}\).

In the other direction, we also know some impossibility results regarding the compressibility of protocols. The following theorem shows the limitations of internal compression:\(^\text{13}\):

**Theorem 7.21.** For every \( k > 0 \), there is a protocol \( \pi \) and a distribution on inputs \( \mu \) such that the internal information of the protocol is \( O(k) \), yet every protocol simulating \( \pi \) on the same distribution of inputs with error at most \( 1/3 \) must have communication \( 2^{\Omega(k)} \).

The following theorem shows limitations of external compression:\(^\text{14}\):

**Theorem 7.22.** For every \( k > 0 \), there is a protocol \( \pi \) and an input distribution \( \mu \) such that the external information of the protocol is \( O(k) \) every simulation of the protocol with error \( 1/3 \) must have communication at least \( 2^{\Omega(k)} \).

**Exercise 7.1**

In correlated sampling, show that the expected values of \( I \) and \( J \) are proportional to the size of the universe.
Exercise 7.2

In Lemma 7.5, show that $\Pr[M^A \neq M^B] \geq |p - q|$.

Exercise 7.3

Let $X, Y$ be jointly distributed in $\{0, 1\}^n$ as follows. Let $I \in [n]$ be uniform, and let $(X, Y)$ be uniform conditioned on $X_{<I} = Y_{<I}$ and $X_I \neq Y_I$.

1. Compute $I(X : Y)$.
2. Show that if the players have shared randomness and get $(X, Y)$ as inputs, and Alice sends a message $M$ to Bob in a way that allows Bob to decode the value of $X$ from $M, Y$ then the expected length of $M$ is at least $n/2 - O(1)$.
3. Deduce that Theorem 7.6 and Corollary 7.7 are sharp up to the additive $O(1)$ term.\footnote{Braverman and Garg, 2014}

Exercise 7.4

In the sense of Claim 7.8, show that there is no encoding of the positive integers so that each integer $z$ is encoded with $\log(n) + \log \log(n) + O(1)$ bits.

Exercise 7.5

Let $X, Y$ be jointly distributed random variables. Show that there is a random variables $Z$ so that

- $Z$ is independent of $(X, Y)$.
- Conditioned on $Z = z$, the variables $Y$ becomes a deterministic function of $X$.
- $H(Y|Z) \leq I(X : Y) + 2 \log I(X : Y) + O(1)$.

Exercise 7.6

Let $X$ be a random variable taking values in the positive integers. Let $E = \mathbb{E} [\log X]$. Prove that $H(X) \leq E + \log E + O(1)$.

Exercise 7.7

Consider the compression in Theorem 7.13. Show that if the protocol $\pi$ is deterministic then the simulating protocol has length at most $O(I \log C)$ for error $1/3$. 

\[ \text{Exercise 7.2} \]

In Lemma 7.5, show that $\Pr[M^A \neq M^B] \geq |p - q|$.

\[ \text{Exercise 7.3} \]

Let $X, Y$ be jointly distributed in $\{0, 1\}^n$ as follows. Let $I \in [n]$ be uniform, and let $(X, Y)$ be uniform conditioned on $X_{<I} = Y_{<I}$ and $X_I \neq Y_I$.

1. Compute $I(X : Y)$.
2. Show that if the players have shared randomness and get $(X, Y)$ as inputs, and Alice sends a message $M$ to Bob in a way that allows Bob to decode the value of $X$ from $M, Y$ then the expected length of $M$ is at least $n/2 - O(1)$.
3. Deduce that Theorem 7.6 and Corollary 7.7 are sharp up to the additive $O(1)$ term.\footnote{Braverman and Garg, 2014}

\[ \text{Exercise 7.4} \]

In the sense of Claim 7.8, show that there is no encoding of the positive integers so that each integer $z$ is encoded with $\log(n) + \log \log(n) + O(1)$ bits.

\[ \text{Exercise 7.5} \]

Let $X, Y$ be jointly distributed random variables. Show that there is a random variables $Z$ so that

- $Z$ is independent of $(X, Y)$.
- Conditioned on $Z = z$, the variables $Y$ becomes a deterministic function of $X$.
- $H(Y|Z) \leq I(X : Y) + 2 \log I(X : Y) + O(1)$.

\[ \text{Exercise 7.6} \]

Let $X$ be a random variable taking values in the positive integers. Let $E = \mathbb{E} [\log X]$. Prove that $H(X) \leq E + \log E + O(1)$.

\[ \text{Exercise 7.7} \]

Consider the compression in Theorem 7.13. Show that if the protocol $\pi$ is deterministic then the simulating protocol has length at most $O(I \log C)$ for error $1/3$. 

\[ \text{Exercise 7.2} \]

In Lemma 7.5, show that $\Pr[M^A \neq M^B] \geq |p - q|$.

\[ \text{Exercise 7.3} \]

Let $X, Y$ be jointly distributed in $\{0, 1\}^n$ as follows. Let $I \in [n]$ be uniform, and let $(X, Y)$ be uniform conditioned on $X_{<I} = Y_{<I}$ and $X_I \neq Y_I$.

1. Compute $I(X : Y)$.
2. Show that if the players have shared randomness and get $(X, Y)$ as inputs, and Alice sends a message $M$ to Bob in a way that allows Bob to decode the value of $X$ from $M, Y$ then the expected length of $M$ is at least $n/2 - O(1)$.
3. Deduce that Theorem 7.6 and Corollary 7.7 are sharp up to the additive $O(1)$ term.\footnote{Braverman and Garg, 2014}

\[ \text{Exercise 7.4} \]

In the sense of Claim 7.8, show that there is no encoding of the positive integers so that each integer $z$ is encoded with $\log(n) + \log \log(n) + O(1)$ bits.

\[ \text{Exercise 7.5} \]

Let $X, Y$ be jointly distributed random variables. Show that there is a random variables $Z$ so that

- $Z$ is independent of $(X, Y)$.
- Conditioned on $Z = z$, the variables $Y$ becomes a deterministic function of $X$.
- $H(Y|Z) \leq I(X : Y) + 2 \log I(X : Y) + O(1)$.

\[ \text{Exercise 7.6} \]

Let $X$ be a random variable taking values in the positive integers. Let $E = \mathbb{E} [\log X]$. Prove that $H(X) \leq E + \log E + O(1)$.

\[ \text{Exercise 7.7} \]

Consider the compression in Theorem 7.13. Show that if the protocol $\pi$ is deterministic then the simulating protocol has length at most $O(I \log C)$ for error $1/3$. 

Part II

Applications
8

Circuits and Proofs

Although communication complexity ostensibly studies the amount of communication needed between parties that are far apart, it is deeply involved in our understanding of many other concrete computational models and discrete systems. In this chapter, we discuss the applications of communication complexity to boolean circuits and proof systems.

Boolean Circuits

Boolean circuits are the most natural model for computing boolean functions $f : \{0, 1\}^n \rightarrow \{0, 1\}$. A boolean circuit is a directed acyclic graph whose vertices, often called gates, are associated with boolean operators or input variables. Every gate with in-degree 0 corresponds to an input variable, the negation of an input variable, or a constant bit. All other gates compute either the logical AND (denoted by the symbol $\wedge$) or the OR (denoted by the symbol $\lor$) of the inputs that feed into them. Usually, the fan-in of the gates is restricted to being at most 2. We adopt this convention, unless we explicitly state otherwise.

Every gate $v$ in a circuit computes a boolean function $f_v$ of the inputs to the circuit. We say that a circuit computes a function $f$ if $f = f_v$ for some gate $v$ in it. Every circuit is associated with two standard complexity measures: The size of the circuit is the number of gates, and the depth of the circuit is the length of the longest directed path in the underlying graph. The size corresponds to the number of operations the circuit performs, and the depth to the parallel time it takes the computation to end, using many processors.

Understanding the complexity of computation with boolean circuits is extremely important, because they are a universal model of computation. Any function that can be computed by an algorithm

---

Figure 8.1: A circuit computing the parity $x_1 \oplus x_2 \oplus x_3$. This circuit has size 15 and depth 4.

One may consider circuits where every gate has fan-in 2 and computes an arbitrary function of its inputs. This only changes the size and depth of the circuit by a constant factor, since any function of 2 bits can be computed by a small circuit using only AND, OR and NOT gates.
in $T(n)$ steps can also be computed by circuits of size approximately $T(n)$. So, to prove lower bounds on the time complexity of algorithms, it is enough to prove that there are no small circuits that can carry out the computation. Counting arguments imply that almost every function requires circuits of exponential size\(^1\). However, we know of no explicit function for which we can prove a super-linear lower bound, highlighting the difficulty in proving such lower bounds.

We focus on two well-known types of circuits. A formula is a circuit whose underlying graph is a tree. Every circuit of depth $d$ can always be turned into a formula whose size is at most $2^d$, and depth is at most $d$. A monotone circuit is a circuit that does not use any negated variables. A monotone circuit computes a monotone function; $f(y) \geq f(x)$ whenever $y_i \geq x_i$ for all $i$.

Every boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ can be computed by a circuit\(^2\) of depth $n$ and size at most $O(2^n/n)$. A similar statement holds for the size and depth of monotone circuits computing monotone functions.

We now describe a general connection between circuit complexity and communication complexity.

**Karchmer-Wigderson Games**

Every boolean function that is not identically 0 or 1 defines a communication problem via its Karchmer-Wigderson game\(^3\). In the game defined by the function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, Alice gets $x \in f^{-1}(0)$, Bob gets $y \in f^{-1}(1)$, and they seek to find $i \in [n]$ such that $x_i \neq y_i$. When $f$ is monotone, one can define the monotone Karchmer-Wigderson game to be the problem where the inputs are $x \in f^{-1}(0), y \in f^{-1}(1)$ as before, but now Alice and Bob want to find an $i$ such that $x_i < y_i$.

The basic observation is

**Lemma 8.1.** A circuit of depth $d$ computing $f$ yields a length $d$ deterministic protocol for the associated game. If the circuit is monotone, the protocol solves the monotone game.

**Proof.** The construction of the protocol is by induction on the depth of the circuit. If the gate computing $f$ is computed as $f = g \land h$, then either $g(x) = 0$ or $h(x) = 0$, while $g(y) = h(y) = 1$. Alice can announce whether $g(x)$ or $h(x)$ is 0, and the parties can continue the protocol using $g$ or $h$. Similarly if $f = g \lor h$, Bob can announce whether $g(y) = 1$ or $h(y) = 1$, and the parties then continue with either $g$ or $h$. If $f$ is the negation of $g$, then the parties can continue

---

\(^1\) Shannon, 1949

The number of circuits of size $s$ can be bounded by $2^{O(s \log s)}$, while the number of functions $f$ is $2^{2^n}$. So if $s \ll 2^n/n$, one cannot hope to compute every function with a circuit of size $s$.

\(^2\) Lupanov, 1958

Several restricted classes of circuits are not discussed in this book. We restrict our attention to methods related to communication complexity. For example, if the circuit is allowed to have gates of arbitrarily large fan-in, then it makes sense to talk about circuits of bounded depth. Today, we know that constant depth circuits require exponential size to compute functions like parity and majority. These lower bounds are sometimes proved by taking a random restriction of the inputs, or by using methods based on polynomials.

\(^3\) Karchmer and Wigderson, 1990

In the Karchmer-Wigderson game, Alice and Bob are computing a relation rather than a function—there may be many indices $i$ with the property they seek.

Show that when $f$ is monotone, there must be an $i$ with $x_i < y_i$. 
the protocol using $g$, without communicating at all. If $f$ is the $i$'th input variable, the parties identify an index $i$ for which $x_i \neq y_i$.

When the circuit is monotone, the same simulation finds an index $i$ such that $x_i = 0, y_i = 1$, since there are no negated variables.

The topology of the circuit determines the topology of the protocol tree. Every AND gate corresponds to a node in the protocol tree where Alice speaks, every OR gate corresponds to a node where Bob speaks, and every input gate corresponds to a leaf in the protocol tree. Thus, a circuit of depth $d$ gives a protocol of length at most $d$.

Conversely, we have:

**Lemma 8.2.** If the Karchmer-Wigderson game for a non-constant function $f$ can be solved with $d$ bits of communication, then there is a circuit of depth $d$ computing $f$. If $f$ is monotone, and the monotone game can be solved with $d$ bits of communication, then there is a monotone circuit of depth $d$ computing $f$.

**Proof.** We shall prove, by induction on $d$, that for any non-empty sets $A \subseteq f^{-1}(0), B \subseteq f^{-1}(0)$, the following holds. If there is a protocol such that whenever $x \in A$ is given to Alice and, $y \in B$ is given to Bob, they can exchange $d$ bits to find $i$ such that $x_i \neq y_i$, then there is a circuit of depth $d$ computing a boolean function $g$ with $g(A) = 0$, and $g(B) = 1$. When $A = f^{-1}(0), B = f^{-1}(1)$, this implies the lemma.

If we are working with the monotone game, we shall prove that the resulting circuit is monotone.

When $d = 0$, the protocol must have a fixed output $i$, and so we must have that $x_i \neq y_i$ (or $x_i = 0, y_i = 1$ in the monotone case) for every $x \in A, y \in B$. Thus, setting $g$ to be the $i$'th variable or its negation works.

Suppose $d > 0$ and Alice speaks first. Then, her message partitions the set $A$ into two non-empty disjoint sets $A = A_0 \cup A_1$, where $A_0$ is the set of inputs that lead her to send 0 as the first message, and $A_1$ is the set of inputs that lead her to send 1. Both sets must be non-empty, since otherwise the first bit of the protocol need not be transmitted, and the lemma follows by induction.

By induction, the two children of the root node in the protocol tree correspond to boolean functions $g_0$ and $g_1$, with $g_0(A_0) = g_1(A_1) = 0$ and $g_0(B) = g_1(B) = 1$. Consider the circuit that takes the AND of the two gates obtained inductively, and denote the function it computes by $g$. Then for all $y \in B$, we have $g(y) = g_0(y) \land g_1(y) = 1 \land 1 = 1$. For all $x \in A$, either $x \in A_0$ or $x \in A_1$. In either case $g(x) = g_0(x) \land g_1(x) = 0$. If the first bit of the protocol is sent by Bob, the proof is similar, except we take the OR of the gates obtained by induction.
One immediate consequence of Lemma 8.2 is regarding the circuit depth required to compute the majority and parity functions. In Section 1 and Exercise 1.5, we proved that solving the Karchmer-Wigderson games for these functions requires at least $2\log n - O(1)$ bits of communication. This shows that both of these functions requires circuits of depth $2\log n - O(1)$, and hence the smallest formulas for these functions have size $\Omega(n^2)$.

As we shall see, the Karchmer-Wigderson connection between circuit complexity and communication complexity is a powerful tool for proving lower bounds on circuit depth.

Lower Bounds on the Depth of Monotone Circuits

One of topics we do not yet understand in circuit complexity is the power of depth:

**Open Problem 8.3.** Can every function that is computable using circuits of size polynomial in $n$ be computed by circuits of depth $O(\log n)$?

However, we do know how to prove interesting results when the underlying circuits are monotone.

**Matching**

One of the most studied combinatorial problems is finding the largest matching in a graph. A matching is a set of disjoint edges. Today, we know of several polynomial time algorithms that can find the matching of largest size in a given graph.4

Given a graph $G$ on $n$ vertices, define

$$\text{Match}(G) = \begin{cases} 1 & \text{if } G \text{ has a matching of size at least } n/3 + 1, \\ 0 & \text{otherwise.} \end{cases}$$

Since there are polynomial time algorithms for finding matchings, one can obtain polynomial sized circuits that compute Match. However, we do not know of any logarithmic depth circuits that compute Match. Here we show that there are no monotone circuits of depth $o(n)$ computing Match.5

By Lemma 8.2, it is enough to prove a lower bound on the communication complexity of the corresponding monotone Karchmer-Wigderson game. In the monotone matching game, Alice gets a graph $G$ with Match($G$) = 1 and Bob gets a graph $H$ with Match($H$) = 0. Their goal is to find an edge which is in $G$, but not in $H$. We shall prove:

---

4 Kleinberg and Tardos, 2006

5 Raz and Wigderson, 1992

Note that Match is a monotone function.
Theorem 8.4. Any randomized protocol solving the matching game must communicate \( \Omega(n) \) bits.

As a corollary, using Lemma 8.1, we get:

**Corollary 8.5.** Every monotone circuit computing \( \text{Match} \) has depth \( \Omega(n) \).

**Proof of Theorem 8.4.** The theorem is proved by reduction to the disjointness lower bound proved in Theorem 6.14. We shall show that if there is a protocol for the monotone matching game with length \( c \), then there is a randomized protocol with length \( O(c) \) solving the disjointness problem on a universe of size \( \Omega(n) \). By Theorem 6.14, this implies that \( c \geq \Omega(n) \).

Suppose Alice and Bob get inputs \( X \subseteq [m] \) and \( Y \subseteq [m] \). They encode \( X \) and \( Y \) as two graphs \( G_X \) and \( H_Y \) on the vertex set \( [3m + 2] \), and then using public randomness, they randomly permute the vertices of the graphs and feed them into the protocol for the monotone matching game.

Figure 8.2 shows an example for \( G_X \) and \( H_Y \). These graphs are constructed as follows:

**Building \( G_X \):** Alice constructs the graph \( G_X \) as follows. For each \( i \in [m] \), the graph \( G_X \) contains the edge \( \{3i, 3i-1\} \) if \( i \in X \), and has the edge \( \{3i, 3i-2\} \) if \( i \notin X \). In addition, \( G_X \) contains the edge \( \{3m + 1, 3m + 2\} \).

The construction ensures that \( G_X \) consists of \( m + 1 \) disjoint edges, and so \( G_X \) contains a matching of size \( m + 1 \).

**Building \( H_Y \):** Bob uses \( Y \) to build a graph \( H_Y \) as follows. For each \( i \in [m] \), if \( i \in Y \) then Bob connects \( 3i - 2 \) to all the other \( 3m + 1 \) vertices of the graph, and if \( i \notin Y \) then Bob connects \( 3i \) to all the other vertices.

By construction, there are \( m \) vertices so that every edge of \( H_Y \) touches one of these vertices. These are the gray vertices in Figure 8.2. So, \( H_Y \) does not contain a matching of size \( m + 1 \).
If $X$ and $Y$ are disjoint, the outcome of the protocol must be the edge corresponding to $\{3m + 1, 3m + 2\}$. On the other hand, if $X$ and $Y$ intersect in $k > 0$ elements, then there are exactly $k + 1$ edges in $G_X$ that are not in $H_Y$.

Since the graph is permuted randomly before the protocol is executed, the outcome of the protocol is equally likely to be one of these $k + 1$ edges. Indeed, let $e, e'$ be two of these $k + 1$ edges, and let $\sigma$ be a permutation of the vertices such that $\sigma(e) = \sigma(e')$ maps the edge $e$ to the edge $e'$. Then for every permutation $\tau$ for which the protocol outputs the edge $\tau(e)$ when it samples $\tau$, there is another permutation $\tau \circ \sigma$ such that the protocol must output $\tau(\sigma(e'))$ when it samples $\tau \circ \sigma$. So, the probability that the protocol outputs $e$ and $e'$ must be exactly the same.

In particular, the probability that the protocol outputs the edge corresponding to $\{3m + 1, 3m + 2\}$ is at most $1/2$. Now, if the output of the protocol is not the edge corresponding to $\{3m + 1, 3m + 2\}$, the players know that the sets are not disjoint. Repeating this experiment a constant number of times, the players are able to solve disjointness with probability of error at most $1/3$.

Monotone Circuit Depth Hierarchy

We can use the connection to communication to show that monotone circuits of large depth are strictly more powerful than circuits of small depth. Throughout this section, we work with circuits of arbitrarily large fan-in.

Let $F_{n,k}$ be the formula that is computed by the full AND-OR tree with gates of fan-in $n$ and depth $k$. This is the formula where all non-input gates have fan-in exactly $n$. The gates of odd depth are OR gates, and the gates of even depth are AND gates. Every input gate is labeled by a distinct unnegated variable. The size of $F$ is $O(n^k)$.

We shall prove that any formula of smaller depth computing $F$ must have exponential size:

**Theorem 8.6.** Any monotone circuit of depth $k - 1$ that computes $F$ must have size at least $2^{\frac{n}{16k} \log k}$.

**Proof.** It suffices to show that any protocol computing the associated monotone Karchmer-Wigderson game has communication at least $n/16 - k$. The lower bound follows, since if the size of the circuit is at most $s$, the communication of a $k - 1$ round protocol for the Karchmer-Wigderson game can be at most $(k - 1) \lceil \log s \rceil$.

We prove that the Karchmer-Wigderson game has large communication by reducing the problem to the pointer-chasing problem that

\[ \]
we studied in Chapter 6. Here Alice and Bob are given \( x, y \in [n]^n \) and want to compute \( z = z(k) \), where \( 1 = z(0), z(1), z(2), \ldots \) are inductively defined using the rule

\[
z(i) = \begin{cases} xz(i-1) & \text{if } i \text{ is odd,} \\ yz(i-1) & \text{if } i \text{ is even.} \end{cases}
\]

Given inputs \( x, y \) to the point-chasing problem, the inputs \( x', y' \) in \( \{0, 1\}[n]^k \) to \( F \) are constructed as follows. Note that every variable in the formula can be described by a string in \( v \in [n]^k \). We say that \( v \) is consistent with \( x \) if

\[
v_i = \begin{cases} x_1 & \text{when } i = 1, \\ x_{v_{i-1}} & \text{when } i \text{ is odd and not 1.} \end{cases}
\]

We say that \( v \) is consistent with \( y \) if \( v_i = y_{v_{i-1}} \) when \( i \) is even. Alice sets all the coordinates of \( x' \) that are consistent with her input to be 0, and all other coordinates to be 1. Bob sets all the coordinates of \( y' \) that are consistent with his input to be 1, and all other coordinates to be 0.

We now prove that \( F(x') = 0 \) and \( F(y') = 1 \). We focus on \( F(x') \); a similar argument works for \( F(y') \). Every a gate of depth \( d \) in the formula corresponds to a vector in \( [n]^d \). We claim that every gate that corresponds to a vector that is consistent with Alice’s input evaluates to 0. This clearly true for the gates at depth \( k \), since that is how we set the variables in \( x' \). For the gates at depth \( d < k \), if the gate is an AND gate then one of its children is consistent with \( x \) and so evaluate to 0, and if the gate is an OR gate then all of its children are consistent with Alice’s input and so evaluate to 0.

For all \( x, y \), there is a unique \( v \) that is consistent with both \( x \) and \( y \), and that’s when \( v = z(k) \), the output of the pointer-chasing problem. The only coordinate where \( x' \) is 0 and \( y' \) is 1 is the coordinate that is consistent with both \( x, y \).

Thus, any protocol for the monotone Karchmer-Wigderson game gives a protocol solving the pointer-chasing problem, and so by Theorem 6.19, we get that the communication of the game must be at least \( n/16 - k \), as required.

**Boolean Formulas**

A **formula** is a circuit whose underlying graph is a tree. Although we do not know how to prove super-linear circuit lower bounds for arbitrary circuits, we do know how to prove super-linear lower bounds for formulas. When it comes to formulas, the choice of
basis can affect the formula size by more than a constant factor. Nevertheless, one can prove super-linear lower bounds even when allowing each gate to compute an arbitrary function of two bits.

Consider the function Distinct : \([2n]^{n+1} \to \{0, 1\}\), defined as:

\[
\text{Distinct}(x_1, \ldots, x_{n+1}) = \begin{cases} 
1 & \text{if } x_1, \ldots, x_{n+1} \text{ are distinct,} \\
0 & \text{else.}
\end{cases}
\]

Distinct is a boolean function that depends on \(O(n \log n)\) bits. We shall prove\(^6\):

**Theorem 8.7.** Any formula computing Distinct must have at least \(n^2 - O(n \log n)\) input gates.

To prove the theorem, we start by proving a simple communication lower bound. Suppose Alice is given \(n\) numbers \(y_1, \ldots, y_n \in [2n]\), and Bob is given \(z \in [2n]\). They want to compute Distinct\((y_1, \ldots, y_n, z)\).

**Lemma 8.8.** If there is a 1-round protocol where Alice sends Bob \(t\) bits and Bob outputs Distinct\((y_1, \ldots, y_n, z)\), then \(t \geq \log \binom{2n}{n} \geq 2n - O(\log n)\).

**Proof.** To prove the lower bound, it is enough to consider the case when \(y_1, \ldots, y_n\) are distinct elements. In this case, Alice’s message must determine \(S = \{y_1, \ldots, y_n\}\), or else Bob will not be able to compute Distinct\((y_1, \ldots, y_n, z)\). This is because if \(S \neq S'\) are two sets of size \(n\) that are consistent with Alice’s message, then there must be an element \(z \in S\) such that \(z \notin S'\). Then \(z\) is distinct from \(S'\), but not from \(S\). Thus, the number of bits transmitted by Alice must be at least \(\log \binom{2n}{n}\), as required. Since the middle binomial coefficient is maximal, \(\binom{2n}{n} \geq \frac{2^n}{n+1}\), so \(\log \binom{2n}{n} = 2n - O(\log n)\). A more accurate bound using Stirling’s approximation gives \(\binom{2n}{n} = \Theta\left(\frac{2^n}{\sqrt{n}}\right)\).

We are ready to prove the formula lower bound:

**Proof of Theorem 8.7.** Suppose there is a formula \(F\) computing Distinct using \(s\) gates. Each input gate in the formula reads a bit of one of the numbers \(x_i\) in the input to Distinct\((x_1, \ldots, x_{n+1})\). For each \(i \in [n+1]\) we define the tree \(T_i\) as follows (see Figure 8.3). Every vertex of \(T_i\) corresponds to a gate in \(F\). Start by discarding all the gates in \(F\) that do not depend on \(x_i\). In the graph that remains, iteratively replace every gate that has only one input feeding into it with an edge connecting its input to its output.

Now, suppose Alice knows all of the input numbers except \(x_i\), Bob knows \(x_i\), and Alice and Bob want to compute Distinct\((x_1, \ldots, x_n)\). They can use the tree \(T_i\) to carry out the computation efficiently as follows. Bob already knows the values at the leaves of the \(T_i\). Every gate \(v\) in \(T_i\) a boolean function \(f_v\) which depends on gates in \(T_i\).
and some number of Alice’s inputs. There are $2^{2^2} = 2^4$ boolean functions that depend on two variables, so Alice can send 4 bits to Bob to indicate which of these functions he should use to compute $f_i(x_1, \ldots, x_{n+1})$ using the 2 inputs that correspond to gates of $T_i$. Using this information, Bob can compute $\text{Distinct}(x_1, \ldots, x_{n+1})$. The overall communication is at most 4 times the number of vertices in $T_i$.

Since $F$ has only $s$ gates, there must be some $i$ for which $T_i$ has at most $\ell = s/n$ leaves. If $m$ denotes the number of vertices of $T_i$, and $e$ the number of edges in $T_i$, then we must have $e = m - 1$, since $T_i$ is a tree. On the other hand, counting the number of edges by adding up the degrees of the vertices, we have

$$2(m - 1) = 2e \geq 3(m - \ell - 1) + \ell,$$

which implies that $m \leq 2\ell + 1 \leq 2s/n + 1$.

By Lemma 8.8, we get $2s/n + 1 \geq 2n - O(\log n)$, proving the theorem. \qed

**Circuits with Arbitrary Gates**

Similar ideas can be used to show non-trivial lower bounds even when the gates are allowed to compute arbitrary functions of $2n/3$ variables\textsuperscript{7}.

Suppose we want to express a function $f : \{0, 1\}^n \to \{0, 1\}$ as

$$f = g(g_1, \ldots, g_k),$$

where each of the functions $g_1, \ldots, g_k$ depends on $2n/3$ input bits. What is the minimum $k$ required?

---

\textsuperscript{7}Hrubes and Rao, 2015
We can represent Distinct in this form with \( k = O(\log n) \). Nevertheless, there is a closely related explicit function that requires \( k \geq n^{\Omega(1)} \). It remains an open problem to find an explicit function for which \( k = \Omega(n) \).

Assume \( n \) is a power of 2. For a subset \( S \) of \( [n \log(2n)] \) of size \( \log(2n) \) and \( b \in \{0,1\}^{n \log(2n)} \), define the scrambled distinctness function \( SDistinct(S, b) \) as follows. Use the coordinates of \( S \) in \( b \) to define a number \( z \in [2n] \). Use the remaining bits of \( b \) to define \( y_1, \ldots, y_{n-1} \subseteq [2n] \), and finally output \( Distinct(y_1, \ldots, y_{n-1}, z) \). We can prove:

**Theorem 8.9.** \( SDistinct(S, b) \) requires \( k \geq n^{\Omega(1)} \).

**Proof.** As in the formula lower bound, we shall appeal to Lemma 8.8. Suppose we can write \( SDistinct \) as \( g(g_1, \ldots, g_k) \), where each of the gates \( g_i \) depends on at most 2/3\'rds of the input variables.

We claim that if \( k \) is small, there must be some \( S \) for which every gate \( g_i \) reads at most \( 4 \log(2n)/5 \) of inputs that correspond to \( S \). Indeed, suppose we pick the elements of \( S \) uniformly at random. For any \( i \), the expected number of coordinates of \( S \) read by \( g_i \) is at most \( 2 \log(2n)/3 \). So by the Chernoff-Hoeffding bound, the probability that more than \( 4 \log(2n)/5 \) of the coordinates are read in \( g_i \) is at most \( e^{-\Omega(\log(2n))} = n^{-\gamma} \), for some constant \( \gamma \). The probability that the \( 2n \) coordinates sampled are not all distinct is at most \( \log^2(2n)/n \). So, if \( k < n^\gamma/2 \), then \( k \cdot n^{-\gamma} + \log^2(2n)/n < 1 \), and there is a set \( S \) satisfying the properties we want.

Given such a set \( S \), Alice and Bob can use the circuit to obtain a protocol solving the distinctness problem. Bob sets the coordinates of \( b \) in \( S \) according to his input, and Alice sets the remaining coordinates according to her input. Each gate \( g_i \) depends on at most \( 4 \log(2n)/5 \) of Bob’s bits. There are \( 2^{2\log(2n)/5} = 2^{O(n^{4/5})} \) boolean functions that depend on \( 4 \log(2n)/5 \) bits, so Alice can send Bob \( k \cdot O(n^{4/5}) \) bits to describe the function Bob should evaluate to compute each \( g_i \). By Lemma 8.8, we must have that \( k \cdot O(n^{4/5}) \geq \Omega(n) \).

---

**Boolean Depth Conjecture**

Can every function computable by a polynomial sized circuit also be computed by a circuit of depth \( O(\log n) \)? In a sense, this is the same as asking whether every circuit can be balanced so that it looks nearly like a tree. If a function cannot be computed by a balanced circuit, this suggests that the computation of the function cannot be parallelized.

To see this, let \( S_1, \ldots, S_k \subseteq [n] \) be sets of size \( n/2 \) so that for every \( i, j \in [n] \), there is some set of the sequence that contains both \( i, j \). One can show that a random choice of \( O(\log n) \) sets satisfies this property with positive probability. Use these sets to construct a formula. For each \( i \), let \( g_i \) be the function that reads the numbers \( x_\ell \) for \( \ell \in S_i \), and outputs 1 if and only if these numbers are distinct. Let \( g \) be the OR function.

The input to \( SDistinct \) can be encoded using \( n \log(2n) + \log^2 n \) bits.
This seemingly simple problem remains open, despite much effort to resolve it. Here we discuss an approach based on direct-sums in communication complexity to proving that there is a function that can be computed using polynomial sized circuits but cannot be computed by a circuit of depth $O(\log n)$.

The approach is quite natural and is based on the repeated composition of functions. Given functions $f : \{0,1\}^t \to \{0,1\}$ and $g : \{0,1\}^k \to \{0,1\}$, define their composition $f \circ g : \{0,1\}^{tk} \to \{0,1\}$ by

$$f \circ g(x_1, \ldots, x_t) = f(g(x_1), \ldots, g(x_t))$$

where each $x_i$ is a $k$-bit string. Inductively define $f \circ^k$ to be $f \circ f \circ^{k-1}$, the composition of $f$ with itself $k$ times, where $f \circ^1 = f$.

Now, let $f : \{0,1\}^t \to \{0,1\}$ be a function that requires circuit depth $\Omega(t)$—most functions require such depth. Consider the function $f^{t^i} : \{0,1\}^n \to \{0,1\}$ with $n = t^i$. The function $f^{t^i}$ can be computed naively by a circuit of size $O(n^2) = O(t^{2i})$. Indeed, $f$ can be computed using a circuit of size $O(2^i)$, since every function on $t$ bits can be computed by a circuit of size $O(2^i)$. The tree of evaluations of $f$ has at most $O(t^{2i})$ nodes. So, we obtain a circuit computing $f$ with $O(t^{2i} \cdot 2^i)$ gates. This number is at most $O(n^2)$.

It seems natural to conjecture that this naive circuit is best possible in terms of depth. Namely, that the depth complexity $\text{depth}(f^{t^i})$ of $f^{t^i}$ is at least $\Omega(t^2)$, and so much larger than $\log(n)$.

If we could show that there is a function $f$ as above for which the circuit depth of $f \circ f^{k-1}$ must be at least $ct$ more than the circuit depth of $f^{k-1}$, then that would imply that the circuit depth of $f^{t^i}$ is at least $ct^2 \gg t \log t = \log t^i$. This would prove that there is a boolean function depending on $n = t^i$ variables that can be computed using $O(n^2)$ gates, but cannot be computed with a circuit of depth $O(\log n)$.

In terms of communication complexity, all that is needed is an example of a function $f$ for which the communication complexity of the Karchmer-Wigderson game of $f^{t^i}$ is at least $ct$ larger than the communication complexity of the game of $f^{k-1}$. This looks quite similar to understanding the direct-sum question in communication that we studied in Chapters 1 and 7. The ideas we discussed there, unfortunately, do not seem to apply in this situation.

**Proof Systems**

Proof systems provide a framework for proving theorems and for studying the complexity of proofs. A proof system is a specific language for expressing proofs. It consists of a set of rules that allow one to logically derive a theorem from axioms. The study of proof
Resolution Refutations

Perhaps the simplest example of a proof system is resolution. Resolution can be used to refute a boolean formula $F$ expressed in conjuctive normal form, that is, to show that $F$ cannot possibly be satisfied.

For example, consider the formula

$$F = (x_2 \lor x_1) \land (\neg x_2 \lor x_1) \land (\neg x_1 \lor x_3 \lor \neg x_4) \land (\neg x_1 \lor x_3 \lor \neg x_4).$$

The formula $F$ cannot be satisfied by any boolean assignment. To prove that the formula is unsatisfiable, we repeatedly use the resolution rule. The rule derives a clause that must be true if two other clauses are both true:

$$\begin{align*}
(a \lor B) \\
(\neg a \lor C)
\end{align*} \Rightarrow B \lor C,$$

where $a$ is a variable, $B, C$ are clauses and $B \lor C$ is the derived clause obtained by including all the literals in $B$ and $C$. The resolution refutation for $F$ shown in Figure 8.4 uses this rule to give a proof that $F$ cannot be satisfied.

In general, a resolution refutation is a sequence of clauses where each clause is derived by combining two previously derived clauses using the resolution rule. The proof ends when the empty clause, which implies a contradiction, is derived. The proof is said to be tree-like if every derived clause is used only once. A tree-like proof corresponds to the concept of a formula in circuit complexity.
The problem of understanding whether a boolean formula is satisfiable (SAT) is a central problem because of its connection to the complexity classes NP and coNP. The best SAT-solvers known today try to find a satisfying solution while simultaneously trying to prove that formulas obtained after partial assignments cannot be satisfied using resolution refutations. Thus, it is important to understand what kinds of formulas can be efficiently refuted.

To study the power of a given proof system, like resolution, we need to study sequences of formulas of growing complexity. A basic example of such a sequence is the well-known pigeonhole principle.

**The Pigeonhole Principle**

The pigeonhole principle states that if \( n \) pigeons are placed into \( n - 1 \) holes, then some hole must contain at least 2 pigeons. One can use the principle to construct a sequence of unsatisfiable boolean formulas. For \( i \in [n] \) and \( j \in [n - 1] \), we have the variable \( x_{i,j} \) which indicates that the \( i \)’th pigeon is in the \( j \)’th hole. Define the following \( n + 1 \) formulas:

\[
\forall i \in [n], \quad P_i = \bigvee_{j \in [n-1]} x_{i,j},
\]

\[
H = \bigwedge_{i < i' \in [n]} \big( \neg x_{i,j} \lor \neg x_{i',j} \big).
\]

The pigeonhole principle implies that

\[
P = H \land \bigwedge_i P_i
\]

cannot be satisfied by any assignment to the variables \( x_{i,j} \).

How hard is it to prove that \( P \) is unsatisfiable? If one uses resolution, we can prove that it is very hard:

*Theorem 8.10.* Any resolution refutation of the pigeonhole principle must involve \( 2^{\Omega(n)} \) derivation steps.

We give the proof of this theorem, even though it is not directly related to communication complexity. It will help us get a feel for the basic notions in proof complexity. Later, we explain the connection some other connections of proof complexity to communication complexity.

A key idea in the proof is to give the proof system even more power. We allow the proof to assume the following axiom for free:

*Axiom 8.11.* Each hole contains exactly one pigeon, and the \( n - 1 \) pigeons that are in the holes are distinct.

If NP ≠ coNP, then it must be true that for every efficiently checkable proof system, there is an unsatisfiable formula that cannot be refuted in a polynomial number of steps. On the other hand, if P = coNP, then there is a proof system in which every unsatisfiable boolean formula can be refuted in a polynomial number of proof steps.
This can only make it easier to derive a contradiction. Indeed, Axiom 8.11 implies that for each \( i, j \),
\[
\neg x_{i,j} \iff \bigvee_{i' \neq i} x_{i',j}.
\]
This allows us to replace every negated variable in the proof with a disjunction of unnegated variables.

Consider any refutation of \( P \) that derives \( s \) clauses. Let \( C \) be one of the clauses derived in the proof. We say that \( C \) is big if there is a set \( S \subset [n] \) of size \(|S| \geq n/4 \) such that for each \( i \in S \) the number of \( j \)'s so that \( C \) contains \( x_{i,j} \) is at least \( n/4 \).

Let us see how a random assignment affects the refutation. Pick \( n/4 \) of the pigeons uniformly at random, and randomly assign them to \( n/4 \) different holes. If pigeon \( i \) is assigned to hole \( j \) in this process, then we set \( x_{i,j} = 1 \), we set \( x_{i',j} = 0 \) for all \( i' \neq i \), and \( x_{i,j'} = 0 \) for all \( j' \neq j \). In words, this amounts to making sure that the relevant pigeons and holes are not involved with any of the remaining holes and pigeons. This assignment is enforced by Axiom 8.11.

After this assignment to the variables, \( n/4 \) of the pigeon clauses become true. Moreover, several variables disappear, and the formula becomes equivalent to the corresponding formula for \( 3n/4 \) pigeons and \( 3n/4 - 1 \) holes. The resolution refutation must still derive a contradiction. We claim:

**Claim 8.12.** One of the big clauses must survive the assignment.

**Proof.** Consider the refutation of \( P \) after the random assignment. Say that a clause has pigeon complexity \( w \) if there is a set \( S \subset [n] \) of size \( w \) such that
\[
\bigwedge_{i \in S} P_i \Rightarrow C,
\]
yet no smaller set \( S \) has this property.

The contradiction can only be derived from all \( 3n/4 \) pigeon clauses that remain, since one can satisfy any strict subset of those clauses with some assignment to the variables. So, the empty clause in the proof has pigeon complexity at least \( 3n/4 \). Since the contradiction is derived from two clauses, one of the clauses used to derive the contradiction must have pigeon complexity at least \( 3n/8 \). Continuing in this way, we obtain a sequence of clauses in the proof, where each clause requires at least half as many pigeon clauses as the previous one. Since the clauses of \( P \) have pigeon complexity at most \( 1 < n/4 \), there must be a clause \( C \) in this sequence that has pigeon complexity between \( n/4 \) and \( n/2 \).

Let \( S \subset [n] \) be the minimal set of pigeon clauses that imply \( C \). Suppose \( i \in S \) and \( j \) is a hole that that did not yet receive a pigeon
during the random assignment. Since $S$ is minimal, there must be an assignment to all the variables where $\bigwedge_{i' \in S - \{i\}} P_{i'}$ is true, yet $C$ is false. This assignment places all of the pigeons of $S$ into holes, except for the $i$'th pigeon. Suppose $i' \notin S$ and $x_{i',j} = 1$ in this assignment. Consider what happens when we set $x_{i,j} = 0$ and $x_{i,j} = 1$, and leave the rest of the variables as they are. Doing so must make $C$ true, since $\bigwedge_{i \in S} P_i = 1$ in the assignment. Since $C$ is a disjunction of unnegated variables, this can only happen if $C$ contains $x_{i,j}$.

Thus for each $i \in S$, there must be at least $3n/4 - n/2 = n/4$ values of $j$ for which $x_{i,j}$ is in the clause $C$. So, $C$ is big even in the proof after the random assignment. 

\textbf{Claim 8.13.} If a clause $C$ is big, then the probability that $C$ survives the random assignment is at most $\left(\frac{63}{64}\right)^{n/8}$.

\textbf{Proof.} Consider what happens when the first pigeon is assigned to a hole. The probability that the pigeon is one of the $n/4$ pigeons relevant to $C$ is at least $1/4$. The probability that it is assigned to one of the $n/4$ holes that would imply $C$ is at least $1/4$. So the probability that $C$ becomes true after the first pigeon is assigned to a hole is at least $1/16$. Continuing in this way, we see that for each of the first $n/8$ pigeons that we assign to a hole in the random assignment, there are at least $n/4 - n/8 = n/8$ pigeons which if assigned to $n/4 - n/8 = n/8$ holes would lead to the clause becoming true. Thus, the probability that $C$ survives the first $n/8$ assignments of pigeons to holes is at most

$$\left(1 - \frac{(n/8) \cdot (n/8)}{n^2}\right)^{n/8} = \left(\frac{63}{64}\right)^{n/8},$$

as required. \qed

We are ready to prove the theorem:

\textbf{Proof of Theorem 8.10.} Suppose towards a contradiction that the refutation of $P$ has less than $(64/63)^{n/8}$ clauses. By Claim 8.13, there is an assignment of the pigeons to holes such that every big clause does not survive. On the other hand, by Claim 8.12, at least one big clause must survive. \qed

\textbf{Cutting Planes}

A stronger proof system than resolution can be obtained by reasoning about linear inequalities instead of clauses. For example,
the clause \( a \lor \neg b \lor c \) can be viewed as asserting that the boolean variables \( a, b, c \) satisfy the linear inequality

\[
a + 1 - b + c \geq 1,
\]

or equivalently,

\[
-a + b - c \leq 0.
\]

In the cutting planes proof system, we convert the clauses into inequalities and then argue about the inequalities. Suppose the variables in the proof are \( x = x_1, \ldots, x_n \). Since the variables are boolean, we allow the proof to use the inequalities \( x_i \leq 1 \) and \( \neg x_i \leq 0 \) for free. We are allowed to take positive linear combinations of two inequalities in the natural way, and round the right hand side. If we have already derived \( \langle c, x \rangle \leq t \) and \( \langle c', x \rangle \leq t' \), for any non-negative \( a, a' \) for which \( ac + a'c' \) is an integer valued vector, we can use the derivation rule to derive

\[
\begin{align*}
\langle c, x \rangle &\leq t \\
\langle c', x \rangle &\leq t'
\end{align*}
\quad \Rightarrow \quad \langle ac + a'c', x \rangle \leq at + a't'.
\]

We also allow the rounding rule:

\[
\langle c, x \rangle \leq t \Rightarrow \langle c, x \rangle \leq \lfloor t \rfloor,
\]

namely, we can replace any number on the right hand side with the closest integer that is smaller than it. This makes sense because \( x \) is boolean and \( c \) is always integer valued. One proves that the clauses are unsatisfiable by arriving at the contradiction \( 1 \leq 0 \).

The cutting planes system can efficiently simulate resolution, so it is at least as powerful as resolution. For example, consider the resolution derivation

\[
\begin{align*}
-x \lor y \lor z \\
x \lor y \lor \neg w
\end{align*}
\quad \Rightarrow \quad y \lor z \lor \neg w.
\]

Viewing the clauses as inequalities, this corresponds to

\[
\begin{align*}
x - y - z &\leq 0 \\
-x - y + w &\leq 0
\end{align*}
\quad \Rightarrow \quad -y - z + w \leq 0.
\]

This derivation does not directly follow by taking linear combinations—if we add the first two inequalities, we get \(-2y - z + w \leq 0\), which is not quite what we want. However, we can derive the inequality we seek using the rounding rule. We have:
circuits and proofs 181

Cook et al., 1987; and Jukna, 2012

\[
\begin{align*}
  x - y - z & \leq 0 \\
  w & \leq 1 \\
  -x - y + w & \leq 0 \\
  -z & \leq 0 \\
  \frac{1}{2} \cdot (x - y - z + w - 1) & \leq 1 \\
  \frac{1}{2} \cdot (-x - y - z + w) & \leq 0
\end{align*}
\] \Rightarrow \begin{align*}
  x - y - z + w & \leq 1, \\
  -x - y + z & \leq 0, \\
  x - y - z + w & \leq 0, \\
  -y + w & \leq \lfloor 1/2 \rfloor = 0.
\end{align*}

In this way, one can show:

**Lemma 8.14.** If a formula can be refuted in \( s \) steps using resolution, then it can be refuted in \( O(ns) \) steps using cutting planes.

In fact, cutting planes gives a strictly stronger proof system. For example, one can give a cutting planes refutation of the formula for the pigeonhole principle using just \( O(n^2) \) steps\(^\text{13}\). Rewriting the clauses of the pigeonhole principle as linear inequalities, we get:

\[
P_i \equiv x_1 + x_2 + \cdots + x_n \leq -1,
\]

\[
P_{i',j} \equiv x_{i,j} + x_{i',j} \leq 1.
\]

We claim that for each \( j \), we can derive the inequality

\[
L_{k,j} \equiv \sum_{i=1}^{k} x_{i,j} \leq 1
\]

in \( O(k) \) steps. The inequality \( L_{2,j} \) is \( H_{1,2,j} \). To derive \( L_{k,j} \) from previously derived inequalities, use the derivation rule \( k \) times to get

\[
(k - 1) \cdot L_{k-1,j} + \sum_{i=1}^{k-1} H_{i,k,j}
\]

\[
\equiv k(x_{1,j} + x_{2,j} + \cdots + x_{k,j}) \leq 2k - 1.
\]

Now, divide by \( k \) and round to get \( L_{k,j} \). To complete the proof, observe

\[
\sum_{j=1}^{n-1} L_{n,j} \equiv \sum_{j=1}^{n-1} \sum_{i=1}^{n} x_{ij} \leq n - 1,
\]

while

\[
\sum_{i=1}^{n} P_i \equiv -\sum_{i=1}^{n} \sum_{j=1}^{n-1} x_{ij} \leq -n.
\]

Adding these last two inequalities gives \( 1 \leq 0 \).

To summarize, cutting planes can efficiently prove the pigeonhole principle, although resolution can not. Can we find a formula that is difficult to refute using cutting planes? We shall use communication complexity to analyze such an example.
Lower Bounds on Cutting Planes

Here we give an example of an unsatisfiable formula that requires an exponential number of steps to refute in the cutting planes proof system. The formula is based on the combinatorial properties of graphs.

Given a graph, a vertex cover is a set of vertices $U$ such that every edge of the graph contains at least one vertex from $U$. A matching is a set of disjoint edges. We shall design the formula to encode the fact that given any vertex cover in a graph and any matching, the vertex cover must have more vertices than the matching has edges. Indeed, every edge in the matching must be covered by one vertex from the vertex cover, and the edges are disjoint.

We shall construct a formula that asserts that the input graph has a vertex cover of size $k-1$, as well as a matching of size $k$; this ensures that the formula is unsatisfiable. For each possible edge $e = \{v, u\} \subset [n]$, we have the variable $x_e$ which is 1 if and only if the edge $e$ is present in the graph. For $i \in [k]$ and $e$, the variable $y_{i,e}$ encodes whether $\{u, v\}$ is the $i$'th edges in the matching. For $j \in [k-1]$ and a vertex $v \in [n]$, the variable $z_{j,v}$ encodes whether $v$ is the $j$'th vertex in a cover. Now, define the following formulas:

$$C = \bigwedge_{e \in [n]^2} \left( \neg x_e \lor \bigvee_{v \in [k-1]} z_{j,v} \right),$$

Every edge is covered.

$$\forall j \in [k-1], \quad C_j = \left( \bigvee_{v \in [n]} z_{j,v} \right) \land \left( \bigwedge_{v \neq v' \in [n]} \left( \neg z_{j,v} \lor \neg z_{j,v'} \right) \right),$$

The $j$'th vertex in the cover is unique.

$$M = \bigwedge_{e, e' \in [n]^2 : |e \cap e'| = 1 \land i \neq i'} \left( \neg y_{i,e} \lor \neg y_{i,e'} \right),$$

Matching edges are disjoint.

$$\forall i \in [k], \quad M_i = \left( \bigvee_{e \in [n]^2} y_{i,e} \right) \land \left( \bigwedge_{e \neq e' \in [n]^2} \left( \neg y_{i,e} \lor \neg y_{i,e'} \right) \right),$$

The $i$'th edge of the matching is a unique edge in the graph.

$$K = \bigwedge_{e \in [n]^2, i \in [k]} (x_e \lor \neg y_{i,e}).$$

Edges in the matching are in the graph.

Finally, define the formula:

$$F = C \land \left( \bigwedge_{j=1}^{k-1} C_j \right) \land \left( \bigwedge_{i=1}^k M_i \right) \land M \land K.$$
To prove the theorem, we shall reduce the problem to the communication complexity of the matching game, for which we proved a lower bound in Theorem 8.4. In the matching game, Alice gets a graph $G$ so that that has a matching of size $k \approx n/3$, and Bob gets a graph $H$ that does not have a matching of size $k$. Their goal is to find an edge which is in $G$, but not in $H$. We prove:

**Lemma 8.16.** If there is a tree-like cutting plane proof of size $s$ showing that $F$ is not satisfiable, then there is a randomized protocol for the matching game with communication $O(\log(s)(\log(n) + \log\log(s)))$.

By the lemma and Theorem 8.4, we must have

$$\log(s)(\log(n) + \log\log(s)) \geq \Omega(n),$$

which proves Theorem 8.15.

The proof of the lemma shows how to efficiently convert a cutting planes refutation to a communication protocol for the relevant game. In fact, the proof extends to many other formulas that have similar structure, but for simplicity, we limit the discussion to this particular formula. A key step in the proof is the observation that we can efficiently check if linear inequalities that arise in the proof are true or not.

**Proof of Lemma 8.16.** Alice sets the variables $y_{i,e}$ to be consistent with her matching, and Bob sets the variables $z_{j,v}$ and $x_e$ to be consistent with the graph $H$. Under this setting of variables, all of the clauses in $M_i, M, C_j, C$ are true, but one of the clauses in $K$ must be false. This false clause specifies an edge that is in Alice’s graph $G$ but not in Bob’s graph $H$. Our goal is to find this clause using the refutation of $F$.

By Lemma 1.8, there must be some inequality $L$ in the proof that depends on at most $2s/3$ of the clauses, and on at least $s/3$ of the clauses. Let $L$ be such an inequality. Our aim is to check whether $L$ is satisfied or not under the assignment to the variables held by Alice and Bob. $L$ can be written as

$$\kappa + \sum_{i,e} \alpha_{i,e} \cdot y_{i,e} \leq \sum_{j,v} \beta_{j,v} \cdot z_{j,v} + \sum_e \gamma_e \cdot x_e.$$

All of the variables on the left hand side are known to Alice, and all the variables on the right hand side are known to Bob. Since the variables are boolean, there are at most $2^{n^3}$ possible values for the left hand side, and at most $2^{2n^3}$ possible values for the right hand side.

Alice and Bob can thus use the randomized protocol for solving the greater-than problem to compute whether or not this inequality is satisfied by their variables (see Exercise 3.1). They expend
$O(\log(n/\epsilon))$ bits of communication in order to make sure that output of their computation is correct with error $\epsilon$.

If the inequality $L$ is not satisfied, Alice and Bob can safely discard the clauses that are not used to derive $L$, and continue to find a false clause. Otherwise, all of the clauses used to derive $L$ can safely be discarded, and Alice and Bob can start their search again after discarding all the inequalities used to derive $L$.

In either case, they discard at least $s/3$ clauses. This process can repeat at most $O(\log s)$ times, so the probability that they make an error is at most $O(\epsilon \log s)$ by the union bound. Setting $\epsilon$ to be small enough so that this number is at most $1/3$, we obtain a protocol whose communication is at most $O((\log n + \log \log s)(\log s))$, as promised.

\[\square\]

**Exercise 8.1**

We showed that any formula that computes whether or not $x \in [2n]^n$ corresponds to $n$ distinct numbers requires a formula of size $\Omega(n^2)$. This gives a boolean function that depends on $m$ bits but requires $\Omega(m/\log^2 m)$ size formulas. Show how you can improve the lower bound to get a boolean function depending on $m$-bits that requires formulas of size $\Omega(m/\log m)$. HINT: Consider the element distinctness function with $x \in [n]^k$ and carefully choose the relationship between $n, k$ to prove the stronger lower bound.

**Exercise 8.2**

Show that the formula that asserts that there cannot be a graph on $[n]$ which both has a path from 1 to $n$ and a set $S \subset [n]$ with $1 \in S$ and $n \notin S$, and with no edges between $S$ and the complement of $S$ requires a super-polynomial in $n$ number of inequalities to prove in the cutting planes proof system.
9

Memory Size

The memory used by an algorithm is an important resource. In this chapter, we explore two related models that measure the amount of memory used, and prove lower bounds on the best possible algorithms optimizing this resource. As usual, our main tool is communication complexity.

The standard way to model algorithms with bounded memory is via branching programs. A branching program of length \( \ell \) and width \( w \) is a layered directed graph whose set of vertices is a subset of \([\ell + 1] \times [w]\). Each layer from 1 to \( \ell \) is associated with a variable in \( x_1, \ldots, x_n \). Every vertex in layers 1 to \( \ell \) has 2 out-going edges, each labeled by a distinct symbol from \([d]\). Edges go from layer \( u \) to layer \( u + 1 \) for \( u \leq \ell \). The vertices on layer \( \ell + 1 \) are labelled with an output of the program.

Computing a function \( f(x) \) using a branching program is straightforward. On input \( x \in [d]^n \), the program is executed by starting at the vertex \((1, 1)\) and reading the variables associated with each layer in turn. These variables define a path through the program. The program outputs the label of the last vertex on this path.

Intuitively, if an algorithm uses only \( s \) bits of memory, then it can be modeled as a branching program of width at most \( 2^s \). Every function \( f : [d]^n \rightarrow \{0, 1\} \) can be computed by a branching program of width \( d^s \) and length \( n \), and counting arguments show that most such functions require exponential width.

Another motivation for understanding branching programs comes from a powerful theorem due to Barrington:

**Theorem 9.1.** If \( f : \{0, 1\}^n \rightarrow \{0, 1\} \) can be computed by a boolean circuit of depth \( O(\log n) \), then it can be computed by a branching program of width 5 and length \( n^{O(1)} \).

Barrington’s theorem implies that if a function requires a super-polynomial length when the width is restricted to being 5, then it

Figure 9.1: A branching program computing \( x_1 \wedge x_2 \wedge x_3 \).

In the literature, the programs we define here are referred to as oblivious branching programs. In general branching programs, every vertex of the program can be associated with a different variable, and vertices in a particular layer need not read the same variable.

Carry out the counting argument yourself by estimating the number of branching programs of width \( w \) and length \( \ell \), and comparing this to the number of functions \( f \).

\(^1\) Barrington, 1986
must require super-logarithmic depth. As we discussed in Chapter 8, finding an explicit function that requires super-logarithmic depth is a major open problem.

A streaming algorithm is a specific type of branching program—one where the inputs, often called the data stream, are read exactly once, and in order: \(x_1, x_2, \ldots, x_n\). Streaming algorithms are motivated by applications where massive amounts of data need to be processed quickly. In these applications, we cannot afford to store all of the data that is coming in, so we need to process it on the fly, and yet be able to compute some function that depends on all the inputs. We also want to compute the result after making just one pass on the data. The parameter \(\log w\) is called the space of the streaming algorithm. This is because after reading \(x_1, \ldots, x_i\), the state of the program can be described with \(\lceil \log w \rceil\) bits.

We start by describing a couple of clever streaming algorithms.

**Maximum Matching** Suppose the data stream consists of a sequence of edges \(e_1, \ldots, e_m\) in a graph with vertex set \([n]\). The goal of the algorithm is to find a matching of largest size in the graph whose edges are \(e_1, \ldots, e_m\).

There is a very simple algorithm for finding a matching that is within a factor of 2 of the largest one, using space at most \(n \log n\). We store each new edge as long as it does not intersect any of the previously stored edges. At most \(n/2\) edges are stored at the end, and these edges must form a matching. The space of the algorithm is at most \((n/2) \cdot \log n^2 \leq n \log n\).

We claim that this algorithm computes a matching that is at least half as big as the largest matching in the graph. Indeed, let \(M_{\text{out}}\) be the output of the algorithm and let \(M_{\text{max}}\) be a matching in the graph of maximum size. Every edge in \(M_{\text{out}}\) can intersect at most two edges of \(M_{\text{max}}\), since the edges of \(M_{\text{max}}\) are disjoint. By construction, every edge \(e\) in \(M_{\text{max}}\) must intersect at least one of the edge of \(M_{\text{out}}\), or else \(e\) would have been included in \(M_{\text{out}}\). Thus, \(2|M_{\text{out}}| \geq |M_{\text{max}}|\).

**Frequency moments** Suppose the data stream consists of a sequence of elements \(x_1, \ldots, x_m \in [n]\). For \(i \in [n]\), let \(f(i)\) denote the number of times that \(i\) occurs in the data stream. The \(t\)'th moment is defined to be \(\sum_{i=1}^{n} f(i)^t\).

We can efficiently compute the 1'st moment \(\sum_{i=1}^{n} f(i)\). This is just \(m\), which can be computed using space \(O(\log m)\).

The 0'th moment is the number of distinct elements in the sequence. Although computing the 0'th moment requires space \(n\) in general, one can estimate it with less space using a randomized
algorithm. The randomized algorithm is based on sampling. It uses similar ideas to the protocol for the gap-hamming problem we described in Chapter 3. Let $S \subset [n]$ be a random subset obtained by sampling $k$ uniformly random independent elements of $[n]$. In Chapter 3, we showed that counting the number of distinct elements in $S$ is enough to approximate the number of distinct elements in the sequence, up to an additive error of $O(n/\sqrt{k})$, with probability at least $2/3$. The number of distinct elements within the set $S$ can be counted using space $O(k)$. Better algorithms are known, if we wish to estimate the number of distinct elements up to a small multiplicative factor.

Let us see a clever algorithm for computing the 2'nd moment efficiently.

**Theorem 9.2.** For any constant $\epsilon, \delta > 0$, we can estimate the 2'nd moment up to a multiplicative factor of $1 - \epsilon$, with probability of error $\delta$, using memory $O\left(\frac{\log m}{\epsilon^2} \right)$.

The proof is based on the following idea: if we are interested in estimating some quantity $q$, it is often useful to find an unbiased estimator of $q$; namely, a random variable with expectation $q$. If the random variable has small variance, then taking a few samples of it provides a good estimate for the value of $q$.

So, to prove the theorem, we need to come up with an unbiased estimator for the 2'nd moment that can be computed with small memory.

**Proof.** Let $M_2 = \sum_{i=1}^{n} f(i)^2$ be the second moment, which we wish to approximate. For a parameter $k$ to be determined, and $i \in [n], j \in [k]$, let $e_{ij} \in \{+1, -1\}$ be uniformly random, and independent of all other variables. Consider the parameters

$$X_j = \sum_{i \in [n]} e_{ij} f(i)$$

for $j \in [k]$. The crucial point is that $X_j^2$ is an unbiased estimator of $M_2$, and $X_j$ can be computed from the data stream using at most $\log m$ bits of memory.

Let us begin by showing that it is indeed an unbiased estimator:

$$\mathbb{E} \left[ X_j^2 \right] = \mathbb{E} \left[ \left( \sum_{i=1}^{n} e_{ij} f(i) \right)^2 \right] = \mathbb{E} \left[ \sum_{i=1}^{n} e_{ij}^2 f(i)^2 \right] + \mathbb{E} \left[ \sum_{i \neq i'} e_{ij} e_{i'j} f(i) f(i') \right] = M_2,$$

since $\mathbb{E} \left[ e_{ij}^2 \right] = 1$ and $\mathbb{E} \left[ e_{ij} e_{i'j} \right] = 0$ for $i \neq i'$.

We shall prove that the average $Z$ of $X_1^2, \ldots, X_k^2$ is a good estimate $M_2$, when $k$ is large. To see how large $k$ should be, we bound the
In general, the variance of the average of \( k \) independent identically distributed random variables is always smaller by a factor of \( k \).

As above, the odd degree terms vanish.

so, the variance of \( X_j^2 \) is

\[
\mathbb{E} \left[ X_j^4 \right] - \mathbb{E} \left[ X_j^2 \right]^2 = 4 \sum_{i \neq i'} f(i)^2 f(i')^2 \leq 2M_2^2.
\]

The variance of the average \( Z \) can be bounded:

\[
\mathbb{E} \left[ Z^2 \right] - \mathbb{E} \left[ Z \right]^2 = \mathbb{E} \left[ \left( \frac{1}{k} \cdot \sum_{j=1}^{k} X_j^2 \right)^2 \right] - \mathbb{E} \left[ \frac{1}{k} \cdot \sum_{j=1}^{k} X_j^2 \right]^2 \\
= \frac{1}{k^2} \cdot \sum_{i=1}^{j} \mathbb{E} \left[ X_j^4 \right] - \mathbb{E} \left[ X_j^2 \right]^2 \leq \frac{2}{k} \left( \sum_{i=1}^{n} f(i)^2 \right)^2.
\]

Finally, Chebyshev’s inequality implies

\[
\Pr \left[ |Z - M_2| \geq \epsilon M_2 \right] \leq \frac{2}{\epsilon^2 k} < \delta,
\]

for \( k = O \left( \frac{1}{\epsilon^2 \delta} \right) \). \( \square \)

**Lower Bounds on Streaming Algorithms**

The dominant method for proving lower bounds on the memory requirements of streaming algorithms is by appealing to lower bounds in communication complexity. One approach is to break the data stream into two parts. Alice will simulate the execution of the algorithm on the first part of the stream. She will then send Bob the contents of the memory, allowing him to continue the simulation over the second half of the data stream. Perhaps surprisingly, this simple approach often gives tight lower bounds.

**Lower Bounds for Estimating Moments**

To illustrate this basic idea, let us start with computing the frequency moments. Recall that the inputs is a stream of elements \( x_1, \ldots, x_m \in \mathbb{R} \).
[n], and \( f(i) \) denotes the number of times that \( i \in [n] \) occurs in the stream.

Consider the problem of computing the 0′th moment, which is the number of distinct elements in the stream. We prove\(^5\):

**Theorem 9.3.** Any randomized streaming algorithm estimating the number of distinct elements in the stream up to an additive error of \( \sqrt{n} \), requires memory at least \( \Omega(n) \).

*Proof.* The theorem follows by reduction to the lower bound on the communication complexity of the gap-hamming problem. Indeed, suppose Alice has a string \( x \in \{0, 1\}^n \) and Bob has a string \( y \in \{0, 1\}^n \), and they wish to estimate the hamming distance between \( x, y \). Then, viewing \( x \) as the indicator vector for a set \( S = \{i_1, i_2, \ldots, i_s\} \subseteq [n] \), Alice simulates the execution of the given algorithm on the stream \( i_1, \ldots, i_s \), and computes the contents of the memory after the algorithm made a pass on this stream. Alice sends Bob the contents of the memory, as well as \( s = |S| \). Bob continues executing the algorithm on the elements of the set \( T \) of size \( t \), obtained by viewing \( y \) as the indicator vector of \( T \). After the algorithm has finished executing, Bob recovers a number \( k \) which is equal to \( |S \cup T| \) with probability 2/3, and outputs \( 2k - s - t \). Since the hamming distance between \( x \) and \( y \) is exactly

\[
|S \cup T| - |S \cap T| = |S \cup T| - (s + t - |S \cup T|)
\]

\[
= 2 \cdot |S \cup T| - s - t,
\]

the players just solved the gap-hamming problem. By Theorem 5.17, the memory must contain \( \Omega(n) \) bits to encode. \( \square \)

Next, suppose we are interested in computing the maximum frequency: \( \max_i f(i) \). A simple reduction to the communication complexity of disjointness\(^6\) proves:

**Theorem 9.4.** Any randomized algorithm that can estimate \( \max_i f(i) \) within a multiplicative factor of 2 must use memory \( \Omega(n) \).

*Proof.* Suppose Alice and Bob have sets \( x, y \subseteq [n] \) and want to know whether the sets are disjoint or not. Alice simulates an execution of the streaming algorithm whose input stream consists of the elements of \( x \), and then sends the contents of the memory to Bob. Bob continues the simulation using the elements of \( y \). If \( x \) and \( y \) are disjoint, the maximum frequency is at most 1. If they are not disjoint, then the maximum frequency is 2. So, the output of the algorithm allows Alice and Bob to distinguish the two cases. By Theorem 6.14, the memory must contain \( \Omega(n) \) bits. \( \square \)

---

\(^5\) Indyk and Woodruff, 2003; and Chakrabarti and Regev, 2012

\(^6\) Alon et al., 1999

Observe that the maximum frequency is the \( \infty \)-moment:

\[
\lim_{t \to \infty} \left( \sum_{i=1}^{n} f(i)^t \right)^{1/t}.
\]
Lower Bounds for Computing Maximum Matchings

If the algorithm is restricted to using space $n \cdot \text{polylog}(n)$, the best known single-pass algorithm for computing the maximum matching is the algorithm we presented at the beginning of this chapter. Although we do not know whether this algorithm is optimal, we can show\(^7\) that no algorithm can approximate the size of the maximum matching up to a factor of $2/3$. In fact, it is known that there can be no algorithm\(^8\) with an approximation factor better than $1 - 1/e$.

A key combinatorial construction that is useful for proving the lower bound is a dense graph that can be covered by many induced matchings. A matching in a graph is induced if there is a subset of vertices $A$ such that an edge belongs to the matching if and only if the edge is contained in $A$.

Let us see how one can construct such a graph. Theorem 4.2 asserts that there is a subset $T \subseteq [n]$ of size at least $n/2 - \Omega(\sqrt{\log n})$ that does not contain any non-trivial arithmetic progressions of length 3. One can use the set $T$ to construct a graph. Let $A, B$ be two disjoint sets of vertices of size $3n$. Identify each of these sets with $[3n]$. Now put an edge between two vertices $x \in A, y \in B$, if $y = v - t$ and $x = v + t$, for some $n \leq v < 2n$ and $t \in T$. In this way, we get $n$ matchings, one for every fixed choice of $v$. Moreover, these matchings are induced. Consider two edges of the form $(v - t, v + t)$ and $(v - t', v + t')$. We claim that the edge $(v - t', v + t)$ is not present in the graph—it cannot be expressed as $(v' - t'', v' + t'')$ for some $v'$ and $t'' \in T$. Indeed, if this is not the case, then $(v + t) - (v - t') = t + t' = 2t'',$ so $t, t'', t'$ is an arithmetic progression of length 3, which is impossible.

The graph described above does not quite have the parameters we need to prove the strongest lower bound, but a better construction is known\(^9\).

**Theorem 9.5.** For every $\delta > 0$, there is a constant $c > 0$ such that for every $n$ there is a bipartite graph with $n$ vertices on each side that is the disjoint union of $n^{1-c/\log \log n}$ induced matchings, and each matching has at least $(1/2 - \delta)n$ edges.

Given Theorem 9.5, and ideas similar to those used to prove the communication complexity lower bound on indexing, we can prove:

**Theorem 9.6.** For any constant $\gamma > 0$, there is a constant $c > 0$ such that any randomized streaming algorithm that computes a matching whose size is at least $2/3 + \gamma$ fraction of the size of the maximum matching, with probability $2/3$, must have memory $\ell \geq \Omega(\gamma^2 n^{2-c/\log \log n})$.

**Proof.** Consider the following communication game. Alice gets a set of edges $E$ and Bob gets a set of edges $F$. They need to determine whether $E$ and $F$ have a matching of size at least $2/3$ of the size of the maximum matching. The communication complexity of this problem is at least $\Omega(\gamma^2 n^{2-c/\log \log n})$.

\(^7\) Goel et al., 2012
\(^8\) Kapralov, 2013
\(^9\) Ruzsa and Szemerédi, 1978
\(^10\) Goel et al., 2012

Many interesting upper bounds are also known for algorithms that make multiple passes on the data stream.
of edges $H$, and Bob gets a set of edges $J$. Their goal is to output a large matching that is contained in the union of their edges. In the game, Alice must send a message to Bob, and Bob must output the final matching.

Alice and Bob can always use the streaming algorithm to get a protocol for the game—Alice simulates the execution of the algorithm on her edges, and then sends the contents of the memory to Bob, who completes the execution on his edges and outputs the edges found by the algorithm.

To prove the lower bound, we find a hard distribution on the inputs $H, J$, described in Figure 9.2. We shall prove that the players must make an error with significant probability when the inputs are drawn from this distribution. As usual, it is no loss of generality to assume that the protocol of the players is deterministic, since we can always fix their randomness in the best possible way.

Let $\gamma > 0$ be as in the theorem statement, and choose $\delta > 0$ to be a small enough constant that we shall set later. Let $G$ be the graph on
2n vertices promised by Theorem 9.5, which consists of

\[
k \geq n^{1-c/\log \log n}
\]

induced matchings \(G_1, \ldots, G_k\), each of size \(t = (1/2 - \delta)n\). Let \(S\) be a new set of 2n vertices. For each \(i \in [k]\), let \(H_i\) be the random graph obtained by picking exactly \((1 - \delta)t\) of the edges in \(G_i\) uniformly and independently. The graph \(H\) is defined to be the union of \(H_1, \ldots, H_k\). Let \(I \in [k]\) be uniformly random. The graph \(J\) is obtained by matching all of the edges that do not touch the vertices in \(G_I\) to vertices in \(S\).

We claim that the largest matching in the graph is obtained by taking \(J \cup H_I\), which is of size

\[
\sigma = 2(n - t) + (1 - \delta)t \\
= 2(n - (1/2 - \delta)n) + (1 - \delta)(1/2 - \delta)n \\
= 3n/2 + (\delta/2)n + \delta^2/n \geq 3n/2.
\]

Indeed, if any matching in \(H \cup J\) includes an edge of \(G\) that is not in \(G_I\), then we can always remove that edge and replace it with two edges touching \(S\) to obtain a larger matching. So, when the algorithm does not make an error, the number of edges from \(H_I\) that Bob outputs must be at least

\[
r = (2/3 + \gamma)\sigma - 2(n - t) \\
\geq 3\gamma n/2 + n - 2t \geq \gamma n,
\]

if \(\delta\) is small enough—we can choose \(\delta = \Omega(\gamma)\).

Let \(M\) denote the \(\ell\)-bit message that Alice sends to Bob. By Theorem 6.9, since \(H_1, \ldots, H_k\) are independent,

\[
\sum_{i=1}^{k} I(H_i : M) \leq I(H_1, \ldots, H_k : M) \leq \ell.
\]

So, by applying Markov’s inequality twice, there must be an \(i \in [k]\) for which \(I(H_i : M) \leq 2\ell/k\) and the probability of making an error given \(i\) is at most 2/3. Fix such an index \(i\).

Let \(E\) be the indicator random variable for the event that the algorithm makes an error. Namely, \(E = 1\) if the algorithm outputs a matching of size at most \((2/3 + \gamma)\sigma\), and \(E = 0\) otherwise. Now,

\[
I(H_i : ME) \leq 2\ell/k + 1
\]

and yet

\[
I(H_i : ME) = H(H_i) - H(H_i|ME).
\]
We need to bound $l(H_i : ME)$ from below. First observe that since $H_i$ is a uniformly random set of $(1 - \delta)t$ edges chosen from $t$ edges, we have:

$$H(H_i) = \log \left( \frac{t}{(1 - \delta)t} \right).$$

Secondly, the conditional entropy

$$H(H_i|M, E = 1) \leq H(H_i),$$

since the $H_i$ is always a set of $(1 - \delta)t$ edges, and the uniform distribution has the maximum entropy of all such distributions. However, when $E = 0$, the entropy must be significantly lowered:

$$H(H_i|M, E = 0) \leq \log \left( \frac{t - r}{(1 - \delta)t - r} \right).$$

Since $\Pr[E = 0] \geq 1/3$, we get:

$$H(H_i|ME) \leq \frac{1}{3} \log \left( \frac{t - r}{(1 - \delta)t - r} \right) + \frac{2}{3} H(H_i).$$

Hence,

$$l(H_i : ME) \geq \log \left( \frac{t}{(1 - \delta)t} \right) - \frac{1}{3} \log \left( \frac{t - r}{(1 - \delta)t - r} \right) - \frac{2}{3} \log \left( \frac{t}{(1 - \delta)t} \right)
\geq \frac{1}{3} \log \frac{(1 - \delta)t}{(1 - \delta)t - r}.$$

The identity $\binom{\frac{a}{b}}{\frac{a-r}{b-r}} = \frac{a}{b}$ implies

$$\frac{a-j}{b-j} = \prod_{j=0}^{r-1} \frac{a-j}{b-j} = \prod_{j=0}^{r-1} \frac{a-j}{b-j} \geq \left( \frac{a}{b} \right)^r.$$

So,

$$l(H_i : ME) \geq \frac{r}{3} \log \frac{1}{1 - \delta} \geq \Omega(\gamma^2 n).$$

Since $r \geq \gamma n$ and $\log \frac{1}{1 - \delta} \geq \delta \geq \Omega(\gamma)$.

Overall,

$$l \geq \Omega \left( \gamma^2 kn \right),$$

as needed (recall $k \geq n^{1 - c/\log \log n}$).

**Lower Bounds on Branching Programs**

Communication complexity can be used to prove lower bounds on general branching programs as well. Branching programs are
more general than streaming algorithms, because a branching program may read the variables multiple times and in arbitrary order. This makes proving lower bounds harder. Here we present explicit functions that cannot be computed by branching programs that are simultaneously short and narrow.

To prove the lower bound, we first show that any branching program can be efficiently simulated, at least in some sense, by a communication protocol in the number-on-forehead model (see Chapter 4).

Let \( g : \{0,1\}^r \rightarrow \{0,1\} \) be an arbitrary function that \( k \)-parties wish to compute in the number-on-forehead model. Define the function \( g' \) by

\[
g'(x,S_1,\ldots,S_k) = g(x|S_1,\ldots,x'|S_k),
\]

where \( x \in \{0,1\}^n \), \( S_1,\ldots,S_k \) are subsets of \( [n] \) of size \( r \), and \( x|S_i \in \{0,1\}^r \) is the projection of \( x \) to the coordinates in \( S_i \). The input to \( g' \) can be described using at most \( n + O(kr \log n) \) bits.

The key claim\(^{11}\) is that any branching program computing \( g' \) can be used to obtain an efficient protocol computing \( g \) in the number-on-forehead model.

**Theorem 9.7.** There is a constant \( \gamma_0 > 0 \) such that for every \( 0 < \gamma < \gamma_0 \) and for every \( g, g' \) as above with \( k > 10\sqrt{\gamma \log n} \) and \( r \leq \sqrt{n} \) the following holds. If \( g' \) can be computed by a length \( \gamma n \log^2 n \), width \( w \) branching program, then \( g \) can be computed by \( k \) players with communication at most \( O(\gamma \log(w) \log^2(n)) \) in the number-on-forehead model.

Setting \( g \) to be the generalized inner-product function, Theorems 9.7 and the lower bound from 5.8 imply that any branching program with length \( \ell < \gamma n \log^2 n \) that computes \( g' \) must have width at least \( 2^{2^{\Omega(\ell)}} \). No better tradeoff between the length and width of branching programs is known.

**Proof.** Let \( \gamma \) be a constant that we will set to be small enough in the proof. Consider a branching program of length \( \gamma n \log^2 n \) and width \( w \) computing \( g' \). Partition the layers of the program into consecutive parts, in such a way that each part reads at most \( n/3 \) variables, and there at most \( 3\gamma \log^2 n \) parts.

Consider the bipartite graph where every vertex on the left corresponds to a part of the partition, and every vertex on the right corresponds to one of the \( n \) variables \( x_1,\ldots,x_n \) of \( g' \). Connect two vertices if the variable does not occur in the corresponding part in the partition. The degree of each vertex on the left is at least \( 2n/3 \), so the edge density is at least 2/3.

We shall prove, the following claim, which corresponds to Figure 9:

---

\(^{11}\) Babai et al., 1989; and Hrubes and Rao, 2015

---

Figure 9.3: The bipartite graph defined by the branching program can be partitioned into cliques. The \( Q_i \)'s and \( R_i \)'s in the claim are not necessarily consecutive blocks, and some of the \( Q_i \)'s may be of size 1.
Claim 9.8. One can partition the vertices on the left to k sets \(Q_1, \ldots, Q_k\), and find \(k\) pairwise disjoint sets of vertices on the right \(R_1, \ldots, R_k\), each of size \(r = \sqrt{n}\), such that for all \(1 \leq i \leq k\), every vertex of \(Q_i\) is connected to every vertex of \(R_i\) by an edge.

Before proving the claim, let us see how to use it. Given an input \((x_1, \ldots, x_k)\) to \(g\), use the branching program for \(g'\) with inputs \((x, R_1, \ldots, R_k)\) where \(x|_{R_i} = x_i\) and \(x\) is zero outside \(\bigcup R_i\). Thus, \(g(x_1, \ldots, x_k) = g'(x, R_1, \ldots, R_k)\).

The protocol for computing \(g\) proceeds as follows. Recall that player \(i\) sees \(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k\). The player that corresponds to the first \(n/3\) steps of the branching program can simulate those steps. He uses \(\lceil \log w \rceil\) bits to announce the result of this simulation. Now the player that corresponds to the next \(n/3\) steps of the program can compute the output of the next part of the program and announce it, and so on. The overall communication is at most \(3\gamma \log^2 n \cdot \lceil \log w \rceil\), and the protocol indeed computes \(g\). It only remains to prove the claim.

Proof of Claim 9.8. We prove the claim by repeatedly using Lemma 5.4. We shall use the Lemma as long as the edge density remains above 1/2, and the number of vertices on the left remains larger than \(8\sqrt{\gamma} \log n\). We set \(\gamma\) to be small enough so that \(\sqrt{\gamma} \leq 1/(16 \log(4e))\).

Initially, the edge density of the graph is 2/3. Since the edge density of the graph is at least 1/2 \(\geq 2 \cdot \frac{2\sqrt{\gamma} \log n}{8\sqrt{\gamma} \log n}\), we can apply Lemma 5.4 to find a set \(Q_1\) on the left and \(R_1\) on the right of sizes \(|Q_1| \geq 2\sqrt{\gamma} \log n\) and \(|R_1| = \sqrt{n}\) such that every vertex of \(Q_1\) is connected to every vertex of \(R_1\). Removing \(Q_1, R_1\) from the graph, we can repeat the process to find pairs of disjoint sets

\[(Q_2, R_2), (Q_3, R_3), \ldots, (Q_t, R_t)\]

with the same property, as long as the edge density remains at least 1/2 and the number of vertices on the left remains at least \(8\sqrt{\gamma} \log n\).

The number of times \(t\) we can apply this process is at most

\[t \leq t_0 = \frac{3\gamma \log^2(n)}{2\sqrt{\gamma} \log n} \leq 2\sqrt{\gamma} \log n.\]

The number of vertices on the right that are removed is at most \(t_0 \cdot r \leq n/6\), for large \(n\). This means that the edge density always remains at least 1/2, so the process ends when we reach a state where the number of vertices on the left is \(k' < 8\sqrt{\gamma} \log n\). At this point, we set \(Q_{t+1}\) to be a singleton set \(\{q_{t+1}\}\) and \(R_{t+1}\) to be a set of \(r\) of the neighbors of \(q_{t+1}\). We remove \(Q_{t+1}, R_{t+1}\) and similarly keep building \((Q_{t+2}, R_{t+2})\) as long as there are vertices on the left.

During this process the number of vertices that are removed is at
most \((t_0 + k') \cdot r \leq n/6\) so the edge density always remains at least 1/2.

Exercise 9.1

Show that the formula that asserts that there cannot be a graph which both has a \(k\)-matching and a set of size \(k - 1\) that covers every edge requires an exponential number of inequalities to prove in the cutting planes proof system.

Exercise 9.2

Show that the formula that asserts that there cannot be a graph on \([n]\) which both has a path from 1 to \(n\) and a set \(S \subset [n]\) with \(1 \in S, n \notin S\) \(k\)-matching and a set of size \(k - 1\) that covers every edge requires an exponential number of inequalities to prove in the cutting planes proof system.

Exercise 9.3

Show that any randomized algorithm that can estimate \(\max_i f(i)\) within an additive factor of 1 must use memory \(\Omega(n)\).
10
Data Structures

A data structure is a way to efficiently maintain access to data. Many well known algorithms\(^1\) rely on efficient data structures for their efficiency. Lower bounds on the performance of data structures are often proved by appealing to arguments about communication complexity. In this chapter, we explore some methods for proving such lower bounds.

There are several ways to measure the efficiency of data structures, and usually there is a tradeoff between these different measures. Here we focus on tradeoffs between the total number of memory cells used by the data structure and the time that it takes to perform operations, like updates or queries to the data. Time is measured by counting the number of memory cells that are accessed in order to perform the needed operation.

We begin this chapter with several examples of useful and clever data structures. Later on, we explain how to prove lower bounds on data structures using ideas from communication complexity.

Dictionaries

A dictionary is a data structure that maintains a subset \( S \subseteq [n] \), allowing us to add and delete elements from the set. We would also like to ask membership queries of the form “is \( i \in S \)?”

The most straightforward implementation is to maintain a string \( x \in \{0, 1\}^n \) that is the indicator vector of \( S \). This allows us to add and delete elements, as well as support membership queries in time 1, but requires \( n \) memory cells.

A more efficient randomized alternative is to use hashing. Assume that we only perform \( m \) operations, and we permit the data structure to make an error with small probability. Then, for a parameter \( \epsilon \), we can pick a random function \( h : [n] \rightarrow [n'] \) with \( n' = \lceil m^2 / \epsilon \rceil \). Now,

\(^1\) Kleinberg and Tardos, 2006

For example, interesting data structures are used in Dijkstra’s algorithm for finding the shortest path in directed graphs, and in Kruskal’s algorithm for computing the minimum spanning tree of undirected graphs.
we encode \( S \) using a string \( x \in \{0, 1\}^{n'} \), by setting \( x_j = 1 \) if and only if there is an \( i \in S \) such that \( h(i) = j \). To add \( i \), set \( x_{h(i)} = 1 \), and to delete \( i \), set \( x_{h(i)} = 0 \). This data structure uses only \( n' \) cells of memory, and if at most \( m \) operations are involved, the probability that this data structure makes an error is at most \( \epsilon \).

It is a tantalizing open problem to prove that there is no data structure beating the indicator vector:

**Open Problem 10.1.** Find a data structure that can maintain a dictionary for \( S \subseteq [n] \) where each memory cell has \( O(\log n) \) bits, and all operations can be carried out in time \( O(1) \), and the total space used is \( \ll n \) when \( |S| \ll n \). Alternatively, show that one cannot use space \( O(|S|) \) for any such data structure.

**Maintaining a Set of Numbers**

Efficient algorithms for sorting numbers are key primitives in algorithm design, with countless applications. In many of these applications, we do not actually need to sort the numbers. It is enough to be able to query some information about the sorted list that can be computed very quickly.

**Sort Statistics**

Suppose we want to maintain a set \( S \subseteq [n] \) of \( k \) numbers, so that one can quickly add and delete numbers from the set, as well as compute the minimum of the set.

A trivial solution is to store the \( k \) numbers in a list. Then adding a number is fast, but finding the minimum might take \( k \) steps in the worst case. A better solution is to maintain the numbers in a heap, as in Figure 10.1. The numbers are stored on the nodes of a balanced binary tree, with the property that every node is at most the value of its children. One can add a number to the heap by adding it at a leaf, and bubbling it up the tree. One can delete the minimum by deleting the number at the root, inserting one of the numbers at a leaf into the root, and bubbling down the number. This takes only \( O(\log k) \) time for each operation.

Another solution is to maintain the numbers in a binary search tree, as in Figure 10.2. Each memory location corresponds to a node in the binary tree. Each leaf corresponds to an element of \( x \in [n] \) and stores a binary value indicating if \( x \in S \) or not. Each node above it maintains three numbers: the number of elements of \( S \) in the corresponding subtree, the minimum of \( S \) in that subtree, and the maximum of \( S \) in that subtree. If the subtree is empty, the
minimum and maximum are set to 0. An element can be added to $S$ or deleted from $S$ in $O(\log n)$ steps, by visiting all the memory cells that correspond to the ancestors of the element in the tree. One can also compute the $i$'th smallest element of $S$ in $O(\log n)$ steps, by starting at the root and moving to the appropriate subtree.

**Predecessor Search**

Suppose we want to maintain a set of numbers $S \subseteq [n]$ and given $x \in [n]$ we want to quickly determine the predecessor of $x$, defined as

$$P(x) = P_S(x) = \max\{y \in S : y \leq x\}.$$

If we maintain the numbers using a binary search tree as in Figure 10.2, we can handle updates in $O(\log n)$ time, and answer
queries in time $O(\log n)$. In fact the queries can be computed in time $O(\log \log n)$.

One can improve the update time using van Emde Boas trees². We sketch the solution. For simplicity of the description, assume $\sqrt{n}$ is an integer. Let $I_1, I_2, \ldots, I_{\sqrt{n}}$ be $\sqrt{n}$ consecutive intervals of integers, each of size $\sqrt{n}$. We store the maximum and the minimum elements of the set $S$ in two sets. We recursively apply the whole scheme to store the set $T_5 = \{i : S \cap I_i \neq \emptyset\}$ using a van Emde Boas tree on a universe of size $\sqrt{n}$. Finally, for each $i$, we recursively store the set $S \cap I_i$ using a van Emde Boas tree on a universe of size $\sqrt{n}$. See Figure 10.3 for an illustration.

To compute $P(x)$ from the van Emde Boas tree, let $i$ be such that $x \in I_i$. If $S \cap I_i$ is empty, or $x$ is less than the minimum of $S \cap I_i$, then $P(x)$ is the maximum of $S \cap I_j$, where $j < i$ is the largest index for which $S \cap I_j$ is non-empty. To find $P(x)$, find the predecessor of the relevant interval in $T_5$, and output its maximum element. Otherwise, $P(x)$ is in $S \cap I_i$, and we can compute it recursively using the recursive structure that stores $S \cap I_i$.

In either case, we only need to make one recursive call to a smaller data structure. Since the universe goes from $n$ to $\sqrt{n}$ after each recursive call, there can be at most $O(\log \log n)$ recursive calls before $P(x)$ is found. Similarly, one can add and delete numbers to the set $S$ using at most $O(\log \log n)$ operations.

Later in this chapter, we shall prove that van Emde Boas trees are essentially optimal when it comes to the predecessor search problem. The trees can also be used to query the minimum, median and maximum of a set in time $O(\log \log n)$. Surprisingly, we still do not know whether not this the optimal way to store a set $S$ to compute these parameters, though we can prove some lower bounds on restricted types of data structures³.

³ Brodal et al., 1996; and Ramamoorthy and Rao, 2017

---

² van Emde Boas, 1975

---

Figure 10.3: An example of a van Emde Boas tree.
**Open Problem 10.2.** Find a data structure that can maintain a set $S \subseteq [n]$, support the insertion and deletion of elements, and querying the minimum or median of the set in time $\ll \frac{\log \log n}{\log \log \log n}$. Alternatively, prove that there is no such data structure that can carry out all operations in time $O(1)$.

**Union-find**

The union-find data structure allows us\(^4\) to efficiently keep track of a partition $S_1, \ldots, S_k$ of $[n]$. The initial partition is the partition to $n$ sets of size 1. The data structure supports the union operation, which forms a new partition by replacing two sets $S_i, S_j$ in the partition by their union $S_i \cup S_j$. It also supports find queries which return a unique identifier for the set $S_i$ containing $x$. This data structure has numerous applications. For example, it plays a key role in the fastest algorithms for computing the minimum spanning tree of a graph.

Here is the high-level scheme for an implementation of a union-find data structure, with operations that take time $O(\log n)$. See Figure 10.4 for an illustration. The data structure stores a table with $n$ elements, each consisting of $O(\log n)$ bits. The idea is to represent each set $S_i$ in the partition by a rooted tree $T_i$ with $|S_i|$ nodes. The nodes of $T_i$ are labelled by all elements of $S_i$. The edges of the tree $T_i$ are directed towards the root. Each element $x \in [n]$ has an entry in the table that stores a pointer to its parent, as well as the height of the subtree rooted at $x$. If $x$ is the root of the whole tree, then $x$ points to

\(^{4}\) Galler and Fisher, 1964
itself. The find operation with input \( x \in [n] \) follows the pointer from \( x \) to the root of the tree it belongs to, and outputs the name of the root. The union operation for \( S_i, S_j \) merges the corresponding trees \( T_i, T_j \) by making the root of the shorter tree (breaking ties arbitrarily) \( T_i \) a child of the root of taller tree \( T_j \), and adjusting the heights appropriately. One can show that this ensures that no tree has height more than \( O(\log n) \), and so all operations take time at most \( O(\log n) \).

Approximate Nearest Neighbor Search

In the nearest neighbor search problem, we are given a set \( S \subseteq \{0,1\}^d \) of size \( n \), and wish to store it, so that one can quickly compute the nearest neighbor \( N(x) \) of a query \( x \in \{0,1\}^d \). The nearest neighbor is an element \( y \in S \) minimizing the hamming distance \( \Delta(x,y) = |\{i \in [d] : x_i \neq y_i\}| \). We use the notation

\[
\Delta(x,S) = \min \{ \Delta(x,w) : w \in S \}.
\]

Typically \( n \gg d \).

One can always store the set using \( nd \) bits, and answer the query by querying all \( nd \) bits. We could also store a table with \( 2^d \) entries, each with \( n \) bits, recording the response for every possible query. Here we sketch the idea for a scheme to approximate the nearest neighbor\(^5\). That is, to find an element \( y \in S \) so that

\[
\Delta(x,y) \leq (1+\epsilon) \cdot \Delta(x,S).
\]

Theorem 10.3. For every constant \( \epsilon > 0 \), there is a data structure that can solve the nearest neighbor search problem up to a multiplicative factor of \( (1+\epsilon) \) by storing a table of size \( s = d \log(d)(n \log(\log(d/\epsilon)))^{O(1/\epsilon^2)} \), where each cell contains \( w = d + 1 \) bits, and the query time is \( t = O(d \log(\log(d)/\epsilon)) \).

The key idea is to use locality sensitive hashing—hashing that is useful to identify the Hamming weight of a string. This is done by choosing a random matrix \( Z \) as follow. Let \( Z \) be a random \( r \times d \) matrix, where each entry of \( Z \) is independent and

\[
Z_{i,j} = \begin{cases} 
1 & \text{with probability } \delta, \\
0 & \text{with probability } 1 - \delta.
\end{cases}
\]

Lemma 10.4. For every \( \gamma, \epsilon > 0 \), there is an \( r = O(\log(1/\gamma)/\epsilon^2) \) so that for all \( k \in [d] \) there is a bias \( \delta > 0 \) for the entries of \( Z \), and an interval \( T \subseteq \mathbb{R} \), so that for every \( d \times 1 \) column vector \( w \) with 0/1 entries:

- If \( |w| \leq k \) then \( \Pr[|Zw \mod 2| \in T] \geq 1 - \gamma \).

Prove by induction on \( h \) that any tree of height \( h \) that occurs in the data structure always has at least \( 2^h \) elements in it.

Later, we shall show that this union-find solution is essentially optimal.

\(^5\) Kushilevitz et al., 2000

The query time can be improved if the data structure is allowed to be randomized.

For a binary vector \( x \), let \( |x| \) denote the number 1’s in \( x \).
• If $|w| \geq k(1+\epsilon)$ then $\Pr[|Zw \mod 2| \in T] \leq \gamma$.

The lemma allows to efficiently check whether the Hamming distance of $x, y$ is at most $k$ or not. Indeed, if $Z$ is sampled as above, then $|(Zx \mod 2) - (Zy \mod 2)|$ suffices to verify that the hamming distance $\Delta(x, y)$ is at most $k$ or at least $(1 + \epsilon)k$.

Before proving the lemma, let us see how to use it. We actually describe a random construction that with positive probability yields the required data structure. Let $S$ be the set of vectors of size $n$. Let $\epsilon > 0$ be the approximation parameter.

We shall set $\gamma$ to be $\frac{1}{n \log (\log (d) / \epsilon)}$ times a small constant to be determined. Apply Lemma 10.4 $O(\log (d) / \epsilon)$ times with

$$k = 1, (1 + \epsilon), (1 + \epsilon)^2, \ldots$$

as long as $k \leq d$. For each value of $k$, independently sample $m$ matrices $Z^{(k,1)}, \ldots, Z^{(k,m)}$ and choose the interval $T^{(k)}$ as in the lemma, for some $m = O(d)$ to be determined below. For each $r \times 1$ binary vector $q$, and $j \in [m]$, the data structure stores a cell with $d+1$ bits that contains $y \in S$ if

$$|q - Z^{(k,j)} y \mod 2| \in T^{(k)}.$$ 

If no such $y$ is in the set $S$, the cell is left empty, and if there are several such $y$s, the structure stores just one of them. The total number of cells required is thus

$$s = O(m \log (d) 2^r / \epsilon) = d \log (d) (n \log (\log (d) / \epsilon))^{O(1/\epsilon^2)},$$

and each cell indeed contains $w = d + 1$ bits.

We now sketch how the data structure answers queries. Fix a query $x \in \{0, 1\}^d$. For each $j \in [m]$, choose $y_j \in S$ by a binary search as follows. For $k_0 \approx d/2$, by looking at the cell corresponding to $q = Z^{(k_0,m)}(x) \mod 2$, check if there is $y \in S$ so that

$$|Z^{(k_0,m)}(y - x) \mod 2| \in T^{(k_0)}.$$ 

If the answer is yes go to $k$'s that are smaller than $k_0$, and if the answer is no go to $k$'s that are larger. The element $y_j \in S$ is chosen as the last element in this binary search.

Claim 10.5. For fixed $x$ and $j$,

$$\Pr[y_j \in S, \Delta(y_j, x) \leq (1 + \epsilon)\Delta(x, S)] \geq 2/3.$$

Proof sketch. There are $n$ elements in $S$ and $O(\log (\log (d) / \epsilon))$ values of $k$ in the binary search. Lemma 10.4 implies that the probability in question is at least $1 - O(\gamma n \log (\log (d) / \epsilon)) \geq 2/3$ by the choice of $\gamma$. \qed
From the claim and by the Chernoff-Hoeffding bound, it follows that for fixed $x$, the probability that for at least $m/2$ of the $j$'s we have $y_j \in S$ and $\Delta(y_j, x) \leq (1 + \epsilon)\Delta(x, S)$ is at least $1 - 1/2^{d+1}$. The union bound over all $x$ now implies that there is a choice of matrices and intervals so that at least half of the $y_j$'s are proper solutions to the problem. For this choice, the data structure can properly answer all queries.

It only remains to prove the hashing lemma.

**Proof of Lemma 10.4.** Let $z$ be a single row of $Z$. We claim:

$$\Pr_z[\langle z, w \rangle \equiv 0 \mod 2] = \frac{1 + (1 - 2\delta)|w|}{2}.$$  

Indeed, if we set $p = \Pr_z[\langle z, w \rangle \equiv 0 \mod 2]$, then

$$2p - 1 = \mathbb{E}_z[(-1)^{\langle w, z \rangle}] = \mathbb{E}_z\left[\prod_{i=1}^d (-1)^{w_i z_i}\right]$$

$$= \prod_{z_i} \mathbb{E}_z[(-1)^{w_i z_i}]$$

$$= (1 - 2\delta)^{|w|}.$$  

Now let $\delta$ be set so that  

$$b(\delta, k) = \frac{1 + (1 - 2\delta)^k}{2} = 2/3,$$

so $(1 - 2\delta)^k = 1/3$. Observe that

$$b(\delta, k) - b(\delta, k(1 + \epsilon)) = 1/3 - (1/3)^{1+\epsilon}$$

$$= (1/3) \cdot (1 - 2^{-\epsilon \log 3})$$

$$\geq (1/3) \cdot (1 - (1 - \epsilon \cdot (\log 3)/2))$$

since $2^{-2x} \leq 1 - x$ for $0 \leq x \leq 1/2$.

Let $T$ be the interval

$$T = \left[\left(\frac{2}{3} - \frac{b(\delta, k) - b(\delta, k(1 + \epsilon))}{2}\right) r, 1\right].$$

By the Chernoff-Hoeffding bound, if $|w| \leq k$ then

$$\Pr[|Zw \mod 2| \in T] \geq 1 - e^{-\Omega(\epsilon^2 r)}.$$  

Similarly, if $|w| \geq k(1 + \epsilon)$

$$\Pr[|Zw \mod 2| \in T] \leq e^{-\Omega(\epsilon^2 r)}.$$  

Choosing $r = O(\log(1/\gamma)/\epsilon^2)$ hence completes the proof. \qed
Lower Bounds on Static Data Structures

A static data structure specifies a way to store data in memory, and to answer queries about the data, without the ability to update the data. There are three main parameters that we seek to optimize:

**Number of cells** \( s \): This is the total number of memory cells used to store the data.

**Word size** \( w \): This is the number of bits in each memory cell.

**Query time** \( t \): This is the number of cells that need to be accessed to answer a query on the data.

Ideally, we would like to minimize all three parameters.

The primary method for proving lower bounds on the parameters of static data structures is via communication complexity. In a nutshell, efficient data structures lead to efficient communication protocols. Say we are given a data structure for a particular problem. We define the corresponding data structure game as follows: Alice is given a query to the data structure, and Bob is given the data that is stored in the data structure. Using the data structure, the communication problem can be solved by a \(2^t\)-round protocol. Bob encodes the data using the data. Alice and Bob then simulate the execution of the data structure algorithm—Alice sends \(\log s\) bits to indicate the name of the memory cell she wishes to read, and Bob responds with \(w\) bits encoding the contents of the appropriate cell. After \(t\) repetitions steps, Alice and Bob know the result of the computation.

**Lemma 10.6.** If there is a data structure of size \( s \), word size \( w \), and query time \( t \) for solving a particular problem, then there is a deterministic protocol solving the related communication game with \(2^t\) rounds. In each round Alice sends \(\log s\) bits and Bob responds with \(w\) bits.

Appealing to lower bounds in communication complexity gives us lower bounds on the parameters of the data structure. Let us explore some examples.

**Set Intersection**

Suppose we wish to store an arbitrary subset \( Y \subseteq [n] \) so that on input \( X \subseteq [n] \) one can quickly compute whether or not \( X \cap Y \) is empty.\(^6\) There are several solutions one could come up with:

- We could store \( Y \) as string of \( n \) bits broken up into words of size \( w \). This would give the parameters \( s = t = \lceil n/w \rceil \).

\(^6\)Miltersen et al., 1998
• We could store whether or not $Y$ intersects every potential set $X$. This would give $s = 2^n$ and $w = t = 1$.

• For every subset $V \subseteq [n]$ of size at most $p$, we could store whether or not $Y$ intersects $V$. Since $X$ is always the union of at most $\lceil n/p \rceil$ sets of size at most $p$, this gives $s = \sum_{i=0}^{p} \binom{n}{i}$, $w = 1$, and $t = \lceil n/p \rceil$.

On the other hand, since every data structure leads to a communication protocol for computing set disjointness, for which the communication must be at least $n + 1$, we have:

**Theorem 10.7.** Any data structure that solves the set intersection problem must have $t \cdot (\lceil \log s \rceil + w) \geq n + 1$.

**Lopsided Set Intersection**

In practice, the bit complexity of the queries is often much smaller than the amount of data being stored. In the $k$-lopsided set intersection problem, the data structure is required to store a set $Y \subseteq [n]$. A query to the problem is a set $X \subseteq [n]$ of size $k \ll n$. The data structure must compute whether or not $X$ intersects $Y$.

When $k = 1$, we can get $s = \lceil n/w \rceil$ and $t = 1$, and no better parameters are possible. The problem becomes more interesting when $k > 1$. We can get a solution with $s = \lceil (\binom{n}{k})/w \rceil$ and $t = 1$ by storing whether or not $Y$ intersects each set of size $k$.

Theorem 1.23 yields the following lower bound:

**Theorem 10.8.** In any data structure solving the lopsided set intersection problem,

$$t(\log s + w) \geq \frac{n}{2^{(\log s)/k + 1}} = \frac{n}{st/k + 1}.$$  

For example, if $s \leq \left(\frac{n}{k}\right)^{(k/2)}$, then

$$t \geq \frac{n}{st/k + 1} \geq \frac{n}{\sqrt{n/k} + 1} \geq \frac{\sqrt{n}k}{2}.$$  

**The Span Problem**

In the span problem, the goal is to store $n/2$ vectors $y_1, \ldots, y_{n/2} \in \mathbb{F}_2^n$, with $n$ even. A query is a vector $x \in \mathbb{F}_2^n$. The data structure must quickly compute whether or not $x$ is a linear combination of $y_1, \ldots, y_{n/2}$.

Theorem 1.25 yields the lower bound:

**Theorem 10.9.** In any static data structure solving the span problem,

$$tw \geq n^2/4 - t \log s \cdot (n + 1) - n \log n.$$  

For example, if $s < 2^{n/8t}$, then $tw = \Omega(n^2)$.
Predecessor Search

In the predecessor search problem, the data structure is required to encode a subset $S \subseteq [n]$ of size at most $k$. The data structure should also be able to compute the predecessor $P_S(x)$ of any element $x \in [n]$.

We have seen that there is a dynamic data structure that can achieve this in time $t \leq O(\log \log n)$. Here we show that this bound is essentially tight. The proof actually proceeds by proving a lower bound on a static version of the predecessor search problem. For simplicity, let us set the word size $w = \lceil \log n \rceil$ throughout the discussion. There is a simple static data structure with space $s = n$, and time $t = 1$ for this problem—the data structure stores the value of $P_S(x)$ for every $x$. To compute $P_S(x)$, the data structure algorithm simply needs to read the relevant cell where this value has been stored.

So, to prove a lower bound, we need to appeal to the complexity of static data structures that use space much less than $n$. In fact, any efficient dynamic data structure does lead to such space-efficient data structures. Suppose the set $S$ is promised to be of size at most $\ell$. Then only $\log \binom{n}{\ell}$ bits are required to encode the set $S$, which can be much smaller than $n$. Indeed, any dynamic data structure of time $t$ for predecessor search gives a static data structure with space $O(t\ell)$ and time $t$. Intuitively, one can use the dynamic data structure to add each element of the set $S$, which takes total time $t\ell$. Then every predecessor query can be handled in time $t$ using the dynamic data structure. If $\ell \geq \log n$ and $t = O(\log \log n)$, this gives a static data structure with space at most $O(\ell \log \log n)$ and time approximately $O(\log \log n)$.

**Theorem 10.10.** Suppose that for every $\ell, n$, there is a static data structure with word size $\log n$ that solves the predecessor search problem in time $t$ and space $s$ that is a polynomial function of $\ell, t$. Then we must have

$$t \geq \Omega \left( \frac{\log \log n}{\log s \log \log n} \right).$$

We prove the theorem using the round-elimination method introduced in Chapter 6. Recall that we can view the data structure as a communication protocol. Here Alice gets the element $x \in [n]$, and Bob gets the set $S \subseteq [n]$ of size at most $k$. If there is a data structure with word size $w$, space $s$ and time $t$, then we obtain a protocol where each of message is at most $\log s$ bits, each message of Bob is at most $w$ bits, and there are $2t$ total rounds of communication to compute the predecessor of $x$ in $S$.

To prove the theorem, we shall find a distribution on inputs $x, S$, where $S$ is of size at most $\ell$ such that any protocol as above must
make an error with significant probability. To carry out round elimination, let us start with the base case. Suppose the communication protocol has 0 rounds. Then define the distribution $\mu$ on $x, S$ by setting $x \in \{2, 4\}$ to be uniformly random, and setting $S = \{1, 3\}$. $P(x)$ is then a uniformly random element of $S$, so any 0-round protocol will make an error with probability at least $1/2$ on these inputs.

Now, suppose we have already identified the hard distribution for a $k$-round protocol, and we want to allow the protocol one more round of communication. Suppose Alice sends the first message in the new protocol, and it is a message of length $\log s$. Set $r = \lceil (8t)^2 \log s \rceil$, and sample $x, S$ as follows. First sample $(x_1, S_1), \ldots, (x_r, S_r)$ independently according to the distribution $\mu$. Let $i \in [r]$ be uniformly random. Then set

$$x = \sum_{j=0}^{r-1} x_j \cdot n^{r-j-1},$$

and

$$S = \left\{ \sum_{j=0}^{i-1} x_j \cdot n^{r-j-1} + y \cdot n^{r-i-1} : y \in S_i \right\}.$$

The crucial point is that every element of $S$ has exactly the same most significant bits as $x$, so $P_S(x)$ determines $P_S(x_i)$.

If instead Bob sends the first message, the reduction is a little different. We would set $r = (8t)^2 w$ and sample $x$ and $S$ as follows. Let $(x_1, S_1), (x_2, S_2), \ldots, (x_r, S_r)$ be independently sampled from the original hard distribution, and let $i \in [r]$ be uniformly random. Set

$$x = i \cdot r + x_i,$$

and

$$S = \bigcup_{j=0}^{r-1} \{ j \cdot n + y : y \in S_j \}.$$

The point of this construction is that the predecessor of $x$ in $S$ determines the predecessor of $x_i$ in $S_i$, yet Bob has no information about $i$.

Exactly as in the lower bound for the greater than function, we claim that the fewer protocol must makes an error with probability at most $1/(8t)$ more than the probability of error of the $k + 1$ round protocol. For example, when Alice sends the first message, then her message $m$ has entropy at most $\log s$, so by Corollary 6.12, we have on average over $i, m, x_{<i}$, 

$$p(x_i|m, x_{<i}) \approx p(x_i).$$

Thus, there must be some fixed value of $i, m, x_{<i}$ for which this error is achieved. Since $y_i$ is independent of $x_1, \ldots, x_r$ after $x_i$ is fixed, we
shall argue that
\[ p(x_i, y_i | m, x_{<i}) \approx 1/8t \approx p(x_i, y_i). \]
Indeed, \( p(y_i | x_1, \ldots, x_i) = p(y_i | x_i) \), so we have
\[
|p(x_i, y_i | m, x_{<i}) - p(x_i, y_i)|
= \sum_{x_i, y_i} |p(x_i, y_i | m, x_{<i}) - p(x_i, y_i)|
= \sum_{x_i, y_i} |p(x_i | m, x_{<i}) \cdot p(y_i | m x_{<i}) - p(x_i) \cdot p(y_i | x_i)|
= \sum_{x_i, y_i} p(y_i | x_i) \cdot |p(x_i | m, x_{<i}) - p(x_i)|
= \sum_{x_i} |p(x_i | m, x_{<i}) - p(x_i)|.
\]

These fixed values induce a \( k - 1 \)-round protocol for computing the predecessor, and so they give a protocol for computing \( P_S(x_i) \) with error at most \( 1/8t \) more than the original protocol.

The above argument can be repeated \( 2t \) times to get a distribution on inputs \( x, S \) where the protocol still makes errors with significant probability.

To complete the proof of Theorem 10.10, observe that we started with a universe of size 4. Suppose \( t \ll \frac{\log \log n}{w \log \log n} \). After \( 2t \) iterations of the round-elimination argument, we obtain a distribution on \( x, S \) that is hard for \( 2t \)-round protocols. The size of the set \( S \) is always at most \( 4 \cdot ((8t^2)^{w^2})^t \leq (\log n)^{o(\log \log n)} \). The size of the universe required for the distribution is at most
\[
(8w^2)^t (4t^2 \log s)^t \leq (\log n)^{O(t)} \ll n.
\]
Thus, the distribution we have found proves the theorem!

**Approximate Nearest Neighbor Search**

Here we prove a lower bound for the nearest neighbor search problem\(^9\).

**Theorem 10.11.** For all constants \( \beta > 0 \) and \( 0 < \gamma < 1 \) the following holds. If there is a static data structure solving the approximate nearest neighbor search problem with \( n \) points in dimension \( O(\log(n)/\epsilon^2) \) up to a multiplicative factor of \( 1 + \epsilon \) with \( \epsilon > n^{-\gamma} \), then the parameters of the data structure must satisfy:
\[
s \geq n^{\Omega(1/\epsilon)} \quad \text{or} \quad t \geq \Omega(n^{1-\beta}/w).
\]

The lower bound is proved by appealing to the lower bound on the communication complexity of the lopsided disjointness problem. In Chapter 1, we proved such a lower bound for deterministic protocols. 

\(^9\) Andoni et al., 2006
To prove a lower bound matching the upper bound for approximate nearest neighbor search that we discussed earlier in this chapter, we need to appeal to the following randomized lower bound for lopsided disjointness. Suppose Alice is given a set \( X \subseteq [d] \) of size \( \ell \), and Bob is given a set \( Y \subseteq [d] \) of size \( n \), then we have:

**Theorem 10.12.** For every constants \( \beta > 0 \) and \( 0 < \gamma < 1 \), if there is a randomized protocol solving lopsided disjointness as above for the case \( \ell < n^\gamma \) and \( d \geq 2\ell n \), then either Alice sends at least \( \Omega(\ell \log n) \) bits or Bob must send \( \Omega(n^{1-\beta}) \) bits.

There is a straightforward reduction from the communication problem to the data structure problem. Suppose Alice and Bob have sets \( X, Y \subseteq [d] \) with \( |X| = \ell = 1 + 2/\epsilon \) and \( |Y| = n \), and they want to know if their sets are disjoint or not. Alice can encode \( X \) by its indicator vector \( x \in \{0, 1\}^d \) and Bob can encode \( Y \) by the set \( S = \{e_i : i \in Y\} \), where \( e_i \in \{0, 1\}^d \) is the string of Hamming weight 1 which has a 1 in the \( i \)’th coordinate. Now, if the sets \( X, Y \) are disjoint then \( \Delta(x, S) = \ell - 1 \), and otherwise \( \Delta(x, S) = \ell + 1 \). Since

\[
(1 + \epsilon)(\ell - 1) = \frac{2}{\epsilon} + 2 = \ell + 1,
\]

it suffices to find the nearest neighbor up to an multiplicative error of \( 1 + \epsilon \).

This approach is not efficient enough to prove the strong lower bound of the theorem. To get the stronger bounds, we need to use locality sensitive hashing, as in the proof of the upper bound for nearest neighbor search.

**Proof of Theorem 10.11.** The idea for the proof is similar to straightforward approach as above, but instead of storing the set \( S \), we store a hash of the set. We use Lemma 10.4 with \( k = \ell - 1 \) and \( \gamma = 1/(3n) \). Let the matrix \( Z \), the interval \( T \) and the parameter \( r = O(\log(3n)/\epsilon^2) \) be as in the lemma. By the proof of Lemma 10.4, there is \( \epsilon' = \Omega(\epsilon) \) so that for all \( y \in S \) so that \( \Delta(x, y) = k \) we have

\[
\Pr \left[ r - |Z(x - y) \mod 2| > \left( \frac{1}{3} + \epsilon' \right) r \right] \leq \frac{1}{3n'}.
\]

and for all \( y \in S \) so that \( \Delta(x, y) \geq (1 + \epsilon)k \) we have

\[
\Pr \left[ r - |Z(x - y) \mod 2| < \left( \frac{1}{3} + 2\epsilon' \right) r \right] \leq \frac{1}{3n}.
\]

Now, the players choose \( Z \) using public randomness, Bob uses the data structure to store the (random) set

\[
S' = \{Zy \mod 2 : y \in S\}
\]
and Alice encodes $x$ as
\[ x' = (J + Zx) \mod 2 \]
where $J$ is the all-ones vector. Note that
\[ \Delta(x', S') = \min\{|x' + y' \mod 2 : y' \in S'| \}
\[ = \min\{r - |Z(x - y) \mod 2| : y \in S\}. \]
Hence, by the union bound over the $n$ elements of $S$, with probability at least $2/3$, if there is $y \in S$ so that $\Delta(x, y) = k$ then $\Delta(x', S') \leq (\frac{1}{3} + \epsilon')r$, and otherwise $\Delta(x', S') \geq (\frac{1}{3} + 2\epsilon')r$. By Lemma 10.6, we obtain a protocol computing lopsided disjointness where Alice sends at most $O(t \log s)$ bits and Bob sends at most $O(tw)$ bits, with success probability at least $2/3$. Theorem 10.12 completes the proof.

Lower bounds on Dynamic Data Structures

A dynamic data structure is one that allows to both update and query on the data. The union-find data structure, the van Emde Boas tree, heaps and binary search trees are all examples of dynamic data structures. In this section, we develop methods for proving lower bounds on such data structures.

Dynamic data structures have four main parameters:

- **Number of cells $s$**: This is the total number of memory cells used to store the data.
- **Word size $w$**: This is the number of bits in each memory cell.
- **Query time $t_q$**: This is the number of cells that need to be accessed to answer a query on the data.
- **Update time $t_u$**: This is the number of cells that need to be accessed to update the data.

We allow data structures to be randomized, making decisions using random coin tosses, and this may induce errors. We say that the error of the data structure is $\epsilon$ if for every sequence of updates followed by a single query, the probability that the query is computed correctly is at least $1 - \epsilon$.

Prefix Sum and Maintaining a Sorted List of Numbers

Our first lower bound for dynamic data structures concerns a very basic problem. Although the proof does not involve communication complexity, it does use the concepts of entropy that we developed in...
Chapter 6. We include it here, because it is of fundamental importance.

Suppose we want to maintain a set $S \subseteq [n]$ so that we can add and delete elements from the set, as well as compute the $i$'th element of the set in the sorted order\textsuperscript{10}. We prove:

**Theorem 10.13.** Any data structure maintaining a sorted list of numbers with error at most $1/3$ for $m \geq 2n^2$ operations must satisfy:

$$t_q \cdot \log (t_aw) \geq \Omega(\log n).$$

In particular, if $t_u, w$ are polylogarithmic in $n$, then Exercise 10.2 asserts that $s$ can be assumed to be polynomial in $m$. This gives $t_q \geq \Omega(\log n / \log \log n)$.

We first prove a lower bound for a different task called *prefix sum*. Suppose we want to maintain a binary string $x \in \{0,1\}^n$, which is initially 0, allow to update the value of $x_i$ to be $1 - x_i$ for any $i \in [n]$, and query $q(i) = \sum_{j=1}^{i} x_j$. We prove\textsuperscript{11}:

**Theorem 10.14.** Any data structure correctly computing prefix sum of an $n$ bit string with error at most $1/3$ for $m \geq n \log n$ operations satisfies

$$t_q \cdot \log (t_u w) \geq \Omega(\log n).$$

Before proving Theorem 10.14, we show how to use it to prove a lower bound for maintaining a sorted list of numbers. To do this, we assume we are given a data structure for maintaining a sorted list of numbers, and show how to use it to solve the prefix sum problem.

Use the data structure to Initialize the set of numbers to be

$$T = \{ j \in [n^2] : j \neq 1 \mod n \}.$$ 

We also maintain the indicator vector of $T$. This allows us to add and delete elements to $T$, and check whether $j \in T$ in constant time and word size, and space $n^2$.

We associate every bit $x_i$ with the element $(i-1)n+1 \in [n^2]$. Whenever we wish to flip the value of $x_i$, we check if $(i-1)n+1 \in T$ using the indicator vector. If $(i-1)n+1 \in T$, we delete it from $T$ using the given data structure. If $(i-1)n+1 \notin T$, we add it to $T$. Note that $(i-1)n+1 = 1 \mod n$, and so $(i-1)n+1 \notin T$ initially.

Let $y_{i-1}$ denote the $(i-1)n+1$'st element of $T$ in the sorted order. This is the element returned by the data structure that maintains a sorted list when it is queried with $(i-1)n+1$. It is easy to check that

$$y_{(i-1)n+1} = i - q(i) + ((i-1)n+1).$$

Indeed, $i - q(i)$ counts the number of elements of $[(i-1)n+1] - T$, and since $T$ contains all elements in $[(i-1)n+2, in]$, we always have

\textsuperscript{10} Fredman and Saks, 1989; and Patrascu and Thorup, 2014

The theorem holds even when the error $1/3$ is replaced by an arbitrary constant that is less than $1/2$.

\textsuperscript{11} Fredman and Saks, 1989

This result applies even if the data structure is only required to compute $q(i) \mod 2$.

The indicator vector is a vector $v \in \{0,1\}^n$ with $v_j = 1$ if and only if $j \in S$.
Figure 10.5: An example of the sets $S_j$ when $k = 2, r = 5.$

$y_{i-1} n + 1 = i - q(i) + ((i - 1) n + 1).$ In particular, $i$ and $y_{i-1} n + 1$ determine $q(i).$ So, we can always use the data structure to compute $q(i)$ by making a single query.

This allows us to simulate $n \log n$ operations of the prefix sum data structure using at most $n^2 + n \log n \leq 2n^2$ operations of the given (sorted set) data structure.

**Proof of Theorem 10.14.** To prove the lower bound, we use a particular distribution on operations. By fixing the randomness used by the data structure, we can assume that it is deterministic, and makes an error on at most $\epsilon$ fraction of the random sequences of updates and query.

We set $n = k^r$ with $k \leq O(t_{uw})$ a parameter that we shall set in the proof. For $j = 0, 1, 2, \ldots, r,$ define

$$S_j = \{ a \cdot k^j + 1 : a \in \{0, 1, \ldots, k^r - j - 1\}\}.$$  

The set $S_j$ consists of $k^r - j$ uniformly spaced numbers in $[n].$

Consider the following sequence of random updates that consists of $r + 1$ rounds. In the $j$'th round we pick a uniformly random subset $T_j \subseteq S_j,$ independently of other choices. For each $i \in T_j,$ we flip the value of $x_i$ using the data structure. At the end of these $r + 1$ rounds of updates, we pick a uniformly random coordinate $L \in [n]$ and query $q(L)$ using the data structure.

We shall compute the expected number of queries the data structure needs to make to correctly compute $q(L).$ Say that a cell of the data structure belongs to round $j$ if it was last touched during the updates of round $j.$ We shall prove that for every round $j,$ the probability that a cell belonging to round $j$ is queried during the final query operation is at least $\Omega(1).$ By linearity of expectation, this implies that the expected number of cells queried is

$$E [t_q] \geq \Omega(r) \geq \Omega\left(\frac{\log n}{\log(t_{uw})}\right),$$

as required.
Intuitively, for fixed \( j \), all the information about \( T_j \) can only be encoded by cells belonging to rounds \( \geq j \). However, the number of cells belonging to rounds \( > j \) is much smaller than the entropy of \( T_j \), because the total number of operations performed by the data structure in these rounds is much smaller than \( |S_j| \). So, even accounting for the cells belonging to rounds \( > j \), the algorithm must read a cell belonging to round \( j \) if it wishes to learn information about \( T_j \). Finally, it must learn information about \( T_j \) if it is to correctly answer the query.

Now, let us make this intuition more formal. For the rest of the proof, fix a specific round \( j \). Let \( A \in \{0, 1\}^{k-j} \) be the indicator vector of \( T_j \). Let \( I \) be the maximum element in \( S_j \) that is at most \( L \). The integer \( I \) is uniformly distributed in \( S_j \), and we interpret it as the name of a coordinate of \( A \). For ease of notation, let \( D \) be a random variable encoding \( L, A_{<I}, T_1, \ldots, T_{j-1}, T_{j+1}, \ldots, T_r \) and the contents and locations of all the cells that belong to rounds \( > j \).

The key claim of the proof shows that if \( k \) is large enough, the data structure does not learn much information about \( A_I \) even knowing all of \( D \). Since \( H(A_I) = 1 \), this is equivalent to claiming:

**Claim 10.15.** \( H(A_I|D) \geq 1 - \frac{2w}{k} \).

Before proving Claim 10.15, let us use it to prove that a cell belonging to round \( j \) must be accessed with probability \( \Omega(1) \).

Define

\[
Q = \begin{cases} 
1 & \text{if the data structure queries a cell that belongs to round } j, \\
0 & \text{otherwise,}
\end{cases}
\]

and

\[
E = \begin{cases} 
1 & \text{if the data structure makes an error,} \\
0 & \text{otherwise.}
\end{cases}
\]

When \( Q = 0 \), the output of the algorithm is determined by \( D \), since all the cells that are read in order to compute \( q(L) \) are determined by \( D \). In addition, when \( Q = 0 \), the value of \( A_I \) is determined by \( E \) and \( D \)—when \( Q = 0 \) and \( D \) is fixed, \( q(L) \) is \( A_I \) plus a constant. Denote by \( \gamma \) the probability that \( Q = 1 \).

Now, by Claim 10.15, we have:

\[
1 - \frac{2w}{k} \leq H(A_I|D) \leq H(QA_I|D)
\]

\[
= H(Q|D) + H(A_I|DQ)
\]

\[
\leq H(Q) + H(A_I|DQ).
\]

Thus, by the chain rule,

\[
h(\gamma) = \gamma \cdot \log(1/\gamma) + (1 - \gamma) \cdot \log(1/(1 - \gamma)).
\]
If $k$ is set to be a large multiple of $tw$, the left hand side is close to 1. If $\epsilon, \gamma$ are small, the right hand side is close to 0. Thus we must have $\gamma = \Omega(1)$.

It only remains to prove Claim 10.15.

**Proof of Claim 10.15.** We partition $D$ into two parts:

Let

$$B = (T_1, T_2, \ldots, T_{j-1}, T_{j+1}, \ldots, T_r)$$

and let $C$ denote the locations and contents of all the cells that belong to rounds $> j$.

The number of cells that are touched during the rounds $> j$ is at most

$$tu \cdot \sum_{j'=j+1}^r k^{r-j'} = tu \cdot \frac{k^{r-j}-1}{k-1} \leq 2tu k^{r-j-1}.$$}

Given $B$, the variable $C$ can be described by specifying the contents of all cells that are touched during the updates in round $> j$, since the data structure is deterministic. Thus, we have

$$H(C|B) \leq 2tu k^{r-j-1}.$$

Hence,

$$H(A|BC) \geq H(A|B) - H(C|B) \geq k^{r-j} - 2k^{r-j-1}tuw \geq k^{r-j} \left(1 - \frac{2tw}{k}\right),$$

Now, by the chain rule,

$$\sum_{i=1}^{k^{r-j}} H(A_i|A_{<i}BC) \geq k^{r-j} \left(1 - \frac{2tw}{k}\right),$$

proving Claim 10.15.
Graph Connectivity

Efficient algorithms that maintain and operate on graphs are widely used in computer science. These provide another source of basic data structure questions.

Suppose we want to implement a static data structure that can store a graph on \( n \) vertices that allows to efficiently decide if two vertices are connected in the graph or not.

A trivial solution is to store the adjacency matrix of the graph, and then perform breadth first search using this matrix. It takes \( n^2 \) bits to encode the matrix, but might involve making \( n^2 \) probes to answer a query.

A better solution is to store a vector in \([n]^n\) which stores the name of the connected component that each vertex belongs to. This can be stored with \( n \) words, each of size \( O(\log n) \). Now connectivity for two vertices can be answered by probing two locations in the data structure.

If we want to maintain the graph using a dynamic data structure that supports addition of edges and querying whether or not two vertices are connected, we can use the union-find data structure (as in Figure 10.4). At each point in time, the partition of the vertices represents the current connected components. When a new edge is added, if the two vertices of the edge are contained in the same connected component, nothing needs to be done. Otherwise, the two connected components are merged with the union operation.

Static Connectivity in Sparse Directed Graphs

We have seen that one can solve the static graph connectivity problem on \( n \) vertices with \( t = 2, s = n, w = \log n \). Here we consider the same problem in directed graphs.

A trivial solution is to store whether or not \( u \) is connected to \( v \) for every pair \((u,v)\). This gives \( t = 1, s = n(n - 1) \) and \( w = 1 \). In general, one cannot do much better than this.

Indeed, let \( A \) and \( B \) be two disjoint sets of vertices, each of size \( n/2 \), and consider the graphs where all edges go from \( A \) to \( B \). There are \( 2^{n^2/4} \) such graphs, and the data structure must distinguish all of them, since the queries to the data structure can reconstruct all edges in such graphs. Thus we must have \( sw \geq n^2/4 \).

The problem becomes more interesting when we consider sparse graphs. What if we are guaranteed that every vertex only has \( O(\log n) \) edges coming out of it? Is there a data structure solving this version of graph connectivity with \( s = O(n \log n) \) and \( t, w = O(1) \)?
Communication complexity allows to prove that such data structures do not exist\textsuperscript{12}:

**Theorem 10.16.** In any static data structure solving the directed graph connectivity problem on graphs with at most $nw$ edges and word-size $w$, we must have

$$t \geq \Omega\left(\frac{\log n}{\log \frac{sw}{n \log w}}\right).$$

In particular, Theorem 10.16 implies that if $t = O(1)$ and $w = O(\log n)$ then $s = n^{1 + \Omega(1)}$.

**Proof.** To prove Theorem 10.16, we consider a special family of sparse directed graphs: subgraphs of the butterfly graph. A $(d, w)$ butterfly graph is a layered directed graph where each vertex corresponds to a tuple $(i, u) \in [d + 1] \times [w]^d$. Here $w$ is set to be the word size of the data structure. Each layer has a vertex for each string of length $d$ from the alphabet $[w]$. Each vertex in the $i$'th layer is connected to exactly $w$ vertices from the $i + 1$'st layer that agree in all but the $i$'th coordinate. See Figure 10.6 for an example. This graph has the feature that there is a unique path from every vertex in the first layer to a vertex in the last layer. To go from $(w_1, \ldots, w_d)$ to $(w_1', \ldots, w_d')$, the only path is

$$(w_1, \ldots, w_d) \rightarrow (w_1', w_2, \ldots, w_d)$$

$$\rightarrow (w_1', w_2', w_3, \ldots, w_d) \rightarrow \ldots \rightarrow (w_1', w_2', \ldots, w_d').$$

To describe the graph more rigorously, for any $u \in [w]^d$, let $u_{-i} = (u_1, u_2, \ldots, u_{i-1}, u_{i+1}, \ldots, u_d)$. There is an edge from $(i, u)$ to $(j, v)$ if and only if $j = i + 1$ and $u_{-i} = v_{-i}$.

Our proof of the lower bound from Chapter 1 implies:

**Theorem 10.17.** Suppose Alice is given a string $x \in [w]^k$, and Bob is given a sequence $Y$ of sets $Y_1, \ldots, Y_k \subseteq [w]$. If there a protocol that determines whether or not there is an $i$ such that $x_i \in Y_i$, with Alice sending a bits and Bob sending $b$ bits, then $a + b \geq \frac{w^k}{2^{\sqrt[k]{w} + t}}$.

We shall show how Alice and Bob can use the data structure to solve the lopsided disjointness problem with inputs $x, Y$ as in Theorem 10.17. We set $k = dw^{d-1}$, and $w$ to be the same parameter as in the butterfly graph. The size of the graph is $n \leq 2wk$ and so $d = \Theta\left(\frac{n}{\log w}\right)$ and $w^{d-1} \geq \Omega\left(\frac{n \log w}{\log w}\right).

Alice uses $x \in [w]^k$ to construct a subgraph $G$ of the butterfly graph. Each coordinate of $x$ is associated with a tuple $(i, u_{-i})$, and so $x_{(i, u_{-i})} \in [w]$. The edge from $(i, u)$ to $(i + 1, v)$ is included in $G$ if and
only if \( v_i + u_i = x_{(i,u_{-i})} \mod w \). Observe that \( G \) consists of \( w^d \) vertex disjoint paths from the first layer of the graph to the last layer.

Bob uses \( Y_1, \ldots, Y_k \) to construct a subgraph \( H \) of the butterfly graph. The edge from \((i, u)\) to \((i + 1, v)\) is included in \( H \) if and only if \( v_i + u_i \not\in Y_{(i,u_{-i})} \mod w \).

Now, if there is an \( i \) for which \( x_i \in Y_i \), this corresponds to \( w^d \) edges that are present in \( G \), but not in \( H \). Thus, Alice and Bob can determine if such an \( i \) exists by answering \( w^d \) connectivity queries. Actually, it is enough to answer just \( w^d - 1 \) connectivity queries to compute disjointness—for every vertex \( u \), Alice only needs to know whether the \( w^d - 1 \) paths that start at \((1, u_{-1})\) in \( H \) are included in \( G \) or not.

Alice and Bob can simulate the execution of these \( w^d - 1 \) queries on the data structure in parallel. In each round, Alice sends Bob \( \lceil \log (w^d - 1) \rceil \) bits to indicate the cells that she needs to look up for each of her queries. Bob responds with \( w \cdot w^d - 1 = w^d \) bits to describe the contents of those cells. This simulation gives a protocol where Alice sends \( a = t \lceil \log (w^d - 1) \rceil \) bits, and Bob responds with \( b = tw^d \) bits.

Theorem 10.17 implies that

\[
2t \log \left( \frac{s}{w^d - 1} \right) + tw^d \geq \frac{dw^d}{2\log(w^d - 1)/(dw^d - 1) + 1}.
\]

Since \( \left( \frac{s}{w^d - 1} \right)^{w^d - 1} \leq \left( \frac{es}{w^d - 1} \right)^{w^d - 1} \), we can simplify the inequality to:

\[
2tw^{d - 1} \cdot \log \frac{es}{w^d - 1} + tw^d \geq \frac{dw^d}{\left( \frac{es}{w^d - 1} \right)^{1/d} + 1}
\]

or

\[
t \left( \log \frac{es}{w^d - 1} + 1 \right) \geq \frac{d}{\left( \frac{es}{w^d - 1} \right)^{1/d} + 1}.
\]

Now, if \( t \ll \frac{d}{\log \frac{es}{w^d - 1}} \), then \( \left( \frac{es}{w^d - 1} \right)^{1/d} \leq 3 \), so the right hand side above is at least \( \Omega(d) \), but on the other hand the left hand side is \( \ll d \). Thus, we must have

\[
t = \Omega \left( \frac{d}{\log \frac{es}{w^d - 1}} \right) \geq \Omega \left( \frac{\log n}{\log \frac{su \log u}{n \log w}} \right),
\]

as required.

**Lower bound for Dynamic Graph Connectivity**

In the dynamic graph connectivity problem, the data structure is required to maintain a graph on the vertex set \([n]\), supporting addition
of edges, as well as queries that compute whether or not two vertices are connected in the graph.

Here we prove:\textsuperscript{13}

**Theorem 10.18.** Any data structure solving the graph connectivity problem with error at most $1/3$ for $m \geq n + 1$ operations must satisfy:

$$t_q \cdot \log t_u w \geq \Omega(\log n).$$

The proof shares many ideas with the proof of the lower bound for the prefix-sum problem given in Theorem 10.14. Assume we have a data structure for solving the graph connectivity problem. We shall perform a random sequence of edge additions, and then ask one random connectivity query, and prove that the data structure must probe many locations. As usual, we can assume that the data structure is deterministic, and prove that the expected number of memory cells accessed is large.

**Proof.** For a parameter set $k \leq O(t_u w)$ that we shall in the proof, we assume without loss of generality that $n = 2(k^r - 1)/(k - 1)$, for some $r$. We sample a uniformly random graph that consists of two disconnected $k$-ary trees $T_0, T_1$, each of depth $r$. The number of vertices in such a graph is $n$. One can sample such a graph by randomly relabeling the vertices of 2 such trees.

We add the edges of this graph to the data structure in $r$ rounds, labelled $j = 1, 2, \ldots, r$. In the first round, we add all $2k^r$ edges to the leaves of the trees. In the $j$'th round, we add the edges from depth $r - j$ to $r - j + 1$. After all the edges of the trees have been added, we pick two random leaves and query whether or not they are connected in the graph. The leaves will be connected when they belong to the same tree, and disconnected if they belong to distinct trees.

Say that a cell of the data structure belongs to round $j$ if it was last touched in round $j$ of the updates. We prove that for each $j$, the probability that a cell that belongs to round $j$ was queried is at least $\Omega(1)$. Thus the expected number of queries made must be at least

$$E[t_q] \geq \Omega(r) \geq \Omega\left(\frac{\log n}{\log(t_u w)}\right),$$

which completes the proof.

It remains to prove that the probability that a cell that belongs to round $j$ was queried is $\Omega(1)$. For the rest of the proof, fix a particular round $j$. Let $A \in \{0, 1\}^{2k^{r-j+1}}$ be the random variable which has a bit for vertex $v$ at depth $r - j + 1$ in the graph, describing which tree $v$
belongs to. Formally,
\[ A_v = \begin{cases} 
1 & \text{if } v \in T_1, \\
0 & \text{if } v \in T_0.
\end{cases} \]

Let \( U, V \) be two uniformly random and independent vertices at depth \( r - j + 1 \) in the graph. Set \( R = A_U + A_V \mod 2 \). Note that the final query to the data structure yields the same distribution on \( U, V \) if \( R \) encodes whether \( U, V \) are connected in the graph. Let \( D \) be the random variable encoding the following data: \( U, V \), the edges of the graph not added in the \( j \)th round, and the locations and contents of all cells that belong to rounds \( > j \).

The key claim of the proof is:

**Claim 10.19.** \( H(R|D) \geq 1 - \frac{7tuw}{k} \).

Before we prove the claim, let us see how to use it. Define
\[ Q = \begin{cases} 
1 & \text{if the data structure queries a cell that belongs to round } j, \\
0 & \text{otherwise},
\end{cases} \]
and
\[ E = \begin{cases} 
1 & \text{if the data structure makes an error,} \\
0 & \text{otherwise.}
\end{cases} \]

When \( Q = 0 \), the value of \( R \) is determined by \( E \) and \( D \). In addition, when \( Q = 0 \), the output of the algorithm is determined by \( D \), since all the cells that are read in order to compute whether \( U, V \) are connected are determined by \( D \). Denote by \( \gamma \) the probability that \( Q = 1 \).

By Claim 10.19, we have:
\[ 1 - \frac{7tuw}{k} \leq H(R|D) \leq H(QR|D) \]
\[ = H(Q|D) + H(R|QD) \]
\[ \leq H(Q) + H(R|DQ). \]

Thus,
\[ 1 - \frac{7tuw}{k} \leq H(Q) + H(R|DQ) \]
\[ \leq h(\gamma + \gamma \cdot 1 + (1 - \gamma) \cdot H(R|D, Q = 0) \]
\[ \leq h(\gamma) + \gamma + H(E|Q = 0) \]
\[ \leq h(\gamma) + \gamma + h(\epsilon/(1 - \gamma)). \]

If \( k \) is set to be a large multiple of \( tuw \), the left hand side is close to 1. If \( \epsilon, \gamma \) are small, the right hand side is close to 0. Thus we must have \( \gamma = \Omega(1) \).
**Proof of Claim 10.19.** Let $B$ be the random variable encoding all the edges not added to the graph in the $j$’th round. After fixing the value of $B$, the roots of the two trees have been fixed, the identities of the leaves have also been determined, but the graph still consists of many disjoint and full $k$-ary trees, as in Figure 10.7.

After fixing $B$, there are $2k^{r-j+1}$ vertices at depth $r - j + 1$, and exactly half of these components will get $A_v = 0$. Thus,

\[
H(A|B) = \log \left( \frac{2k^{r-j+1}}{k^{r-j+1}} \right) \\
\geq 2k^{r-j+1} - 1 - \log k^{(r-j+1)/2} \\
\geq 2k^{r-j+1} - 2k^{r-j}.
\]

since $\binom{2m}{m} \geq 2^{m-1}/\sqrt{m}$.

\[
\text{since } \log k^{(r-j+1)/2} \leq k^{(r-j+1)/2} \leq k^{r-j}.
\]

Let $C$ denote the locations and contents of all cells that belong to rounds $> j$. The number of edges added after the $j$’th round is

\[
\sum_{i=j+1}^{r} 2k^{r-i+1} = 2 \cdot \sum_{i=1}^{r-j} k^i = 2k^{r-j} - \frac{1}{k-1} \leq 4k^{r-j}.
\]

Given $B$, each of these edges can contribute at most $t_u w$ to the entropy of $C$. This is because $C$ can be described by specifying the contents of each of the cells accessed when the algorithm adds these edges. Hence,

\[
H(A|BC) \geq H(A|B) - H(C|B) \\
\geq 2k^{r-j+1} - 2k^{r-j} - 1 - 4k^{r-j}t_u w \\
\geq 2k^{r-j+1} - 6k^{r-j}t_u w.
\]

For any fixed vertex $w$ at the same depth as $U, V$, we have

\[
\Pr[w \in \{U, V\}] = 1 - \left( 1 - \frac{1}{2k^{r-j+1}} \right)^2 = \frac{1}{k^{r-j+1}} - \left( \frac{1}{2k^{r-j+1}} \right)^2.
\]
Applying Shearer’s lemma (Lemma 6.5), we conclude that
\[
H(A_U, A_V | UVBC) \\
\geq \left( \frac{1}{k^{r-j+1}} - \left( \frac{1}{2k^{r-j+1}} \right)^2 \right) \cdot \left( 2k^{r-j+1} - 6k^{r-j}t_u w \right) \\
\geq 2 - \frac{1}{2k^{r-j+1}} - \frac{6t_u w}{k} \\
\geq 2 - \frac{7t_u w}{k}.
\]
Now, \( A_U, A_V \) are determined by \( R, A_U \). So, we have
\[
H(R | D) \geq H(A_U, A_V | D) - H(A_U | D) \\
\geq 2 - \frac{7t_u w}{k} - 1 \\
= 1 - \frac{7t_u w}{k},
\]
as required.

**Exercise 10.1**

Modify the Van Emde Boas tree data structure so that it can maintain the median of \( n \) numbers, with time \( O(\log \log n) \) for adding, deleting and querying the median.

**Exercise 10.2**

Show that \( n \) operations of a dynamic data structure with parameters \( s, t_u, t_q, w \) and error \( \epsilon \) can always be simulated by another data structure with space \( n^2(t_u + t_q)/\epsilon' \), update time \( t_u \), query time \( t_q \), word size \( w \) and error \( \epsilon + \epsilon' \).
11

Extension Complexity of Polytopes

Polytope are subsets of Euclidean space that can be defined by a finite number of linear inequalities. They are fundamental geometric objects that have been studied by mathematicians for centuries. Any \( n \times d \) matrix \( A \) and \( n \times 1 \) vector \( b \) defines a polytope \( P \):

\[
P = \{ x \in \mathbb{R}^d : Ax \leq b \}.
\]

In this chapter, we explore some questions about the complexity of representing polytopes. For example, when can a complex polytope be expressed as the shadow of a simple polytope?

Besides being mathematically interesting, this question is relevant to understanding the complexity of algorithms based on linear programming as we explain in detail later on.

Basic Properties and Features of Polytopes

Polytopes have many nice properties that make them easy to manipulate and understand. For example, a polytope \( P \) is always convex—whenever \( x \) and \( y \) are in \( P \), then all the the line segment connecting \( x \) and \( y \) is also in \( P \). Indeed, if \( \gamma \in [0,1] \), then

\[
A(\gamma x + (1-\gamma) y) \leq \gamma Ax + (1-\gamma)Ay \leq \gamma b + (1-\gamma)b = b.
\]

Although the definition of the polytope seems to involve only inequalities, sets defined using equalities are also polytopes. For example, the set given by solutions \( (x, y, z) \in \mathbb{R}^3 \) with

\[
\begin{align*}
x &= y + z + 1, \\
z &\geq 0,
\end{align*}
\]

In other texts, polytopes are sometimes assumed to be bounded—it is assumed that there is a finite ball that contains the polytope. Throughout this textbook, polytopes may have infinite volume.
Figure 11.2: The anatomy of a polytope. The polytope has dimension 2. The six 1-dimensional faces are called facets. The six 0-dimensional faces are called vertices. The intersection between every two facets is a face—it is either a vertex or empty.

is a polytope, because it can be expressed as:

\[
\begin{align*}
x - y - z &\leq 1, \\
-x + y + z &\leq -1, \\
-z &\leq 0.
\end{align*}
\]

A halfspace \( H \) is a particular type of polytope—one that is defined by a single inequality, \( H = \{ x \in \mathbb{R}^d : hx \leq c \} \), where \( h \) is a \( 1 \times d \) matrix, and \( c \) is a real number. Every polytope is therefore the intersection of halfspaces, and every intersection of halfspaces is a polytope. In particular, the intersection of two polytopes is also a polytope.

Moreover, every linear inequality that the points of the polytope satisfy can be derived from the inequalities that define the polytope:

**Fact 11.1.** If a polytope \( P = \{ x \in \mathbb{R}^d : Ax \leq b \} \) is contained in a halfspace \( H = \{ x \in \mathbb{R}^d : hx \leq c \} \), then there is a \( 1 \times n \) row vector \( u \geq 0 \) such that \( uA = h \) and \( ub \leq c \).

The vector \( u \) promised by Fact 11.1 shows how to derive the inequality of the halfspace from the inequalities defining the polytope. It proves that the points of the polytope belong to the halfspace—for every \( x \in P \), we have \( hx = uAx \leq ub \leq c \).

The dimension of a polytope \( P \) is the dimension of the minimal affine subspace \( A \) such that \( P \subseteq A \). A point \( v \) is on the boundary of \( P \)
if $v \in P$, and for every $\epsilon > 0$, there is a point $u \in A - P$ at distance at most $\epsilon$ from $v$. A face of the polytope $P$ is a set of the form $F = P \cap H$, where $H$ is a halfspace that intersects $P$ only on its boundary. The faces of a polytope are also polytopes. The dimensions of the faces are smaller than that of $P$.

When the dimension of the face is exactly one less than the dimension of the polytope itself, we call it a facet. A vertex of the polytope is a non-empty face of dimension 0—it consists of a single point.

The boundary of the polytope is the union of all of its facets—every point on the boundary must belong to some facet, and every point on a facet belongs to the boundary. The inequalities defining the polytope may not correspond to the facets of the polytope. In fact, they can be redundant, since some of the inequalities may be implied by the others. Nevertheless, by Fact 11.1, a halfspace defining a facet can be derived by combining the inequalities defining the polytope.

The number of inequalities needed to express a polytope is at most the number of its facets:

Fact 11.2. If a polytope $P \subseteq \mathbb{R}^d$ has $r$ facets, it can be expressed with $r$ inequalities as $P = \{x : Ax \leq b, Cx = 0\}$, where $A$ is an $r \times d$ matrix, $b$ is an $r \times 1$ column vector, and $C$ is a $k \times d$ matrix for some $k$.

The facets can be used to generate all the other faces of the polytope:

Fact 11.3. Every face of $P$ can be expressed as the intersection of some subset of the facets of $P$. If the polytope is defined by $n$ inequalities, it can have at most $n$ facets.

One important consequence of Fact 11.3 is that a polytope with $f$ facets can have at most $2^f$ faces, since there are at most $2^f$ subsets of the facets.

Transformations of Polytopes

Polytopes behave nicely under some natural transformations. Rotating or translating a polytope gives another polytope. For example, if $P = \{x \in \mathbb{R}^d : Ax \leq b\}$ is a polytope, and $z \in \mathbb{R}^d$ is any vector, then the set $P + z = \{x + z : Ax \leq b\}$ is also polytope. This is because $y \in P + z$ exactly when $A(y - z) \leq b$, which is equivalent to $Ay \leq b + Az$. So,

$P + z = \{y \in \mathbb{R}^d : Ay \leq b + Az\}$. 
We shall see that applying an arbitrary linear transformation to a polytope gives another polytope:

**Theorem 11.4.** If $L$ is a $k \times d$ real valued matrix with $k \leq d$, and $P \subseteq \mathbb{R}^d$ is a polytope, then

$$L(P) = \{ Lx : x \in P \} \subseteq \mathbb{R}^k$$

is also a polytope. Moreover, every face of $L(P)$ is equal to $L(F)$, for some face $F \subseteq P$.

It is often much easier to describe a polytope by applying a linear transformation on another polytope. To illustrate this, let us explore a generic way to generate a polytope from a finite set of points. Given a set of points $V = \{ v_1, \ldots, v_k \} \in \mathbb{R}^d$, the convex hull of these points is the minimal convex set containing $V$. Stated differently, it is the intersection of all convex sets containing $V$. Equivalently, it is the set of points $x \in \mathbb{R}^d$ satisfying

$$x_i = \sum_{j=1}^{k} \mu_j \cdot v_i \quad \text{for } i = 1, 2, \ldots, d$$

$$\mu_j \geq 0 \quad \text{for } j = 1, 2, \ldots, k$$

$$\sum_{j=1}^{k} \mu_j = 1$$

for some $\mu_1, \ldots, \mu_k \in \mathbb{R}$. The equations above describe a polytope whose points are of the form $(x_1, \ldots, x_d, \mu_1, \ldots, \mu_k)$. Projecting this polytope onto the variables $x_1, \ldots, x_d$ is a linear transformation, so the convex hull is also a polytope.

Not every polytope is the convex hull of a finite set of points. However, every bounded polytope is the convex hull of a finite set of points. A bounded polytope is a polytope that is contained in some ball of finite radius.

**Fact 11.5.** A bounded polytope can always be expressed as the convex hull of all its vertices.

Similarly, the conical hull of a finite set $V \subseteq \mathbb{R}^d$ is the set of points that can be obtained by taking non-negative linear combinations of the points in $V$. It is the set of points $x \in \mathbb{R}^d$ satisfying:

$$x_i = \sum_{j=1}^{k} \mu_j \cdot v_i \quad \text{for } i = 1, 2, \ldots, d$$

$$\mu_j \geq 0 \quad \text{for } j = 1, 2, \ldots, k,$$

for some $\mu_1, \ldots, \mu_k \in \mathbb{R}$. Again, Theorem 11.4 implies that the conical hull of a finite set of points is a polytope.
We used the singular value decomposition of matrices when proving the linear lower bound for the gap-hamming problem in Chapter 5.

Quantifying how much the number of facets of a polytope can increase under linear transformations is the main goal of this chapter.

**Proof of Theorem 11.4.** We can assume without loss of generality that $P$ has full dimension; otherwise restrict $L$ to the affine subspace containing it.

If $k = d$ and $L$ is invertible, then the proof is straightforward. If $P = \{ x : Ax \leq b \}$, then

$$L(P) = \{ y : AL^{-1}y \leq b \},$$

so $L(P)$ is also a polytope. In addition, the structure of the polytope is also preserved: There is a one to one correspondence between the faces of $P$ and the faces of $L(P)$. Every face of $P$ given by $F = P \cap H$ corresponds to the face $L(F)$ of $L(P)$. In addition, the dimension of $L(F)$ is exactly the same as the linear dimension of $F$. So every facet or vertex of $P$ becomes a facet or vertex of $L(P)$ under the linear transformation.

When $L$ is not invertible, the theorem becomes a little more involved to prove. Recall that every $k \times d$ matrix $L$ has a singular value decomposition $L = U \cdot \Lambda \cdot V$, where $U$ is an invertible $k \times k$ matrix with $UU^T = I$, where $V$ is an invertible $d \times d$ matrix with $VV^T = I$, and where $\Lambda$ is a $k \times d$ diagonal matrix with $\Lambda_{ij} = 0$ whenever $i \neq j$.

So, $Lx$ can be computed by first applying an invertible linear transformation $V$, then projecting and scaling the polytope on to a subset of the coordinates using $\Lambda$, and then applying another invertible linear transformation $U$.

Since $\Lambda$ is diagonal, we can express it as $\Lambda = DS$, where $D$ is a $k \times d$ diagonal matrix where all non-zero entries are 1, and $S$ is a $d \times d$ diagonal matrix where all entries on the diagonal are non-zero. Then we have $L = UDSV$, where $U$, $V$ and $S$ are invertible, so they all preserve the polytope structure.

A matrix $D$ as above is called a projection, and we call $D(P)$ a projection of $P$. The projection of a polytope can actually have a different number of faces and facets than the original polytope.

The following lemma is enough to prove Theorem 11.4.

**Lemma 11.6.** If $D$ is a $k \times d$ projection matrix and $P \subseteq \mathbb{R}^d$ is a polytope, then the projection $D(P)$ is also a polytope. Moreover, every face of $D(P)$ is equal to $D(F)$, for some face $F \subseteq P$.

**Proof.** The lemma follows from repeating a process called Fourier-Motzkin elimination. It is enough to prove that the lemma holds when $d = k + 1$, since any projection onto fewer coordinates can be obtained by repeatedly projecting onto dimension smaller by one.

Suppose $P \subseteq \mathbb{R}^{k+1}$ is a polytope, and every point of $P$ can be written as $(x_1, \ldots, x_k, z)$. Suppose $D$ is the projection operation that maps such a point to $x = (x_1, \ldots, x_k)$. The inequalities defining $P$ can
be scaled so that they are of three kinds:

\[
A_i \cdot x \leq b_i, \\
A_i \cdot x + z \leq b_i, \\
A_i \cdot x - z \leq b_i.
\]

We call these inequalities of type 0, 1, or \(-1\), according to the coefficient of \(z\) in the inequality.

We now define a new set of inequalities that captures the projection \(D(P)\). An inequality of type 0 induces the same inequality on \(x \in \mathbb{R}^k\). Every two inequalities

\[
A_i \cdot x + z \leq b_i
\]

and

\[
A_j \cdot x - z \leq b_j
\]

of type 1 and \(-1\) can be combined to give a single inequality by addition:

\[
(A_i + A_j) \cdot x \leq b_i + b_j.
\]

If \(P\) was defined by \(n\) inequalities, we obtain at most \(n^2\) inequalities in this way. We claim that these inequalities define \(P\).

Every inequality we have derived is satisfied by the elements of \(P\). On the other hand, if \(x \in \mathbb{R}^d\) satisfies all of these inequalities, we shall prove that there is a choice of \(z\) such that \((x, z) \in P\), so indeed \(x \in D(P)\). Let \(\ell\) be the index maximizing \(A_\ell \cdot x - b_\ell\) over all \(\ell\) for which \(A_\ell, b_\ell\) define an inequality of type \(-1\). Set \(z = A_\ell \cdot x - b_\ell\). This choice of \(z\) ensures that \((x, z)\) satisfies all the inequalities of type \(-1\):

\[
A_j \cdot x - b_j \leq z \Rightarrow A_j \cdot x - z \leq b_j.
\]

\((x, z)\) also satisfies all the inequalities of type 0, since these do not involve \(z\). Every inequality \(A_i \cdot x + z \leq b_i\) of type 1 is also satisfied, because \(x\) satisfies

\[
(A_i + A_\ell) \cdot x \leq b_i + b_\ell
\]

\[
\Rightarrow A_\ell \cdot x - b_\ell + A_i \cdot x \leq b_i
\]

\[
\Rightarrow z + A_i \cdot x \leq b_i.
\]

It remains to argue about the faces of \(D(P)\). Suppose \(H \subseteq \mathbb{R}^k\) is a halfspace such that \(H \cap D(P)\) is a face of \(D(P)\). Then, \(H\) can be expressed as the set of points satisfying \(h \cdot x \leq c\) for some \(h \in \mathbb{R}^k\) and \(c \in \mathbb{R}\). Let \(H' \subseteq \mathbb{R}^{k+1}\) be the set of points \((x, z)\) such that \(h \cdot x \leq c\). Then \(H'\) is a halfspace, and \(D(H') = H\). Since \(D\) is the projection to the first \(k\) coordinates, \(D(H' \cap P) = H \cap D(P)\).

We need to prove that \(H' \cap P\) is a face of \(P\)—namely, that \(H' \cap P\) is contained in the boundary of \(P\). Let \(v = (x, z) \in H' \cap P\) and let \(\epsilon > 0\).
This means that $x = D(v)$ belongs to the face $H \cap D(P)$ so there is a point $u \notin D(P)$ at distance at most $\epsilon$ from $x$ recall that $P$ has full dimension. The point $(u, z)$ is not in $P$ and its distance from $v$ is at most $\epsilon$ as well.

Algorithms from Polytopes

Besides being fundamental geometric objects, polytopes are useful from the perspective of algorithm design, because many interesting computational problems can be reduced to the problem of optimizing a linear function over some polytope.

We illustrate this by showing how to reduce combinatorial problems to optimizing linear functions over polytopes. The high-level idea is as follows. Suppose we wish to solve some combinatorial optimization problem like computing the shortest path between two points. To do so, we build some polytope $P$ and some linear function $L$ so that the maximum of $L$ over $P$ is precisely the quantity we are interested in. We now provide two examples of such reductions.

Shortest Paths

Say we want to compute the distance between the two vertices 1 and $n$ in an undirected graph with vertex set $[n]$. We show how to encode this algorithmic problem as a question about optimizing a linear function over a polytope. For every pair of distinct vertices $u, v \in [n]$, define the variable $x_{u,v}$, and define the graph polytope $P$ by

$$x_{u,v} \geq 0 \quad \text{for every } u \neq v$$
$$\sum_{w \neq 1} x_{1,w} = 1$$
$$\sum_{w \neq n} x_{w,n} = 1$$
$$\sum_{w \neq u} x_{u,w} = \sum_{w \neq u} x_{w,u} \quad \text{for every } u \notin \{1, n\}$$

These equations define a polytope $P \subseteq \mathbb{R}^d$, with $d = n(n-1)$, which has at most $d$ facets, since it is defined by $d$ inequalities.

Now, given a connected graph $G$ with the edge set $E$, consider the problem of finding the point in the graph polytope that minimizes the linear function

$$L(x) = \sum_{\{u,v\} \notin E} n \cdot (x_{u,v} + x_{v,u}) + \sum_{\{u,v\} \in E} (x_{u,v} + x_{v,u}).$$

Think of the variable $x_{u,v}$ as the flow from $u$ to $v$.

(flow is nonnegative)
(flow out of 1 is 1)
(flow into $n$ is 1)
(flow into an intermediate vertex is equal to the flow coming out)
Claim 11.7. \( \min_{x \in P} L(x) \) is the distance from 1 to \( n \) in \( G \).

Proof. Without loss of generality, suppose \( e_1, e_2, \ldots, e_\ell \) are the edges of a shortest path in the graph. If we give the edges of this path weight 1 and all other edges weight 0, that gives a point \( x \) in the polytope with \( L(x) = \ell \).

To prove the claim, we show that for any other \( x \in P \), we have \( L(x) \geq \ell \). Suppose \( x \in P \) is such that \( x_{u,v} > 0 \) for some directed edge \( (u,v) \) so that \( \{u,v\} \) is not on the shortest path. If \( v \neq n \), since the flow into intermediate vertices must be equal to the flow out of intermediate vertices, there must be a vertex \( w \) so that \( x_{v,w} > 0 \) and then a vertex \( z \) such that \( x_{w,z} > 0 \), and so on.

In this way, we see that either there is a directed path \( e'_1, \ldots, e'_k \) from 1 to \( n \) with \( k \geq \ell \), where all edges get positive flow, or there is a directed cycle \( e'_1, \ldots, e'_k \) with positive flow as in Figure 11.4.

In the first case, we can reduce the flow on \( e'_1, \ldots, e'_k \) by a small amount, and increase the weight of the shortest path \( e_1, \ldots, e_\ell \) by the same amount. This gives a new point in the polytope, and the value of \( L(x) \) on it does not increase.

In the second case, we can reduce the weight of all the edges on the cycle by a small amount to get a new point in the polytope. This reduces the value of \( L \).

Repeating these operations eventually gives a point in \( P \) that only places weight on the edges \( e_1, \ldots, e_\ell \). The value of \( L \) on such a point is \( \ell \). \[\Box\]
Matchings

Suppose we wish to find the size of the largest matching in a given graph. We show that this problem can also be encoded as the problem of optimizing a linear function over some polytope. For every pair \( u, v \in [n] \) of distinct vertices we have the variable \( x_{\{u,v\}} \). Each matching corresponds to the points where \( x_{\{u,v\}} = 1 \) if \( u, v \) are matched and \( x_{\{u,v\}} = 0 \) if they are not. The convex hull \( M \) of these matching is called the matching polytope. One can prove that the facets of the polytope correspond to the inequalities:

\[
\begin{align*}
    x_{\{u,v\}} &\geq 0 & \text{for all distinct } u, v \\
    \sum_v x_{\{u,v\}} &\leq 1 & \text{for all } u \\
    \sum_{u \in v \in A} x_{\{u,v\}} &\leq \frac{|A| - 1}{2} & \text{for all } A \subseteq [n] \text{ with } |A| \text{ odd}
\end{align*}
\]

Thus, this polytope has at most \( \binom{n}{2} + n + 2^{n-1} \) facets, and is contained in \( \mathbb{R}^{\binom{n}{2}} \).

Given a graph with \( n \) vertices defined by the set of edges \( E \), let

\[
L(x) = L_E(x) = \sum_{\{u,v\} \in E} x_{\{u,v\}} - \sum_{\{u,v\} \notin E} x_{\{u,v\}}.
\]

Claim 11.8. \( \max_{x \in M} L(x) \) is the size of the largest matching in the graph.

Proof. Let \( Z \) denote the set of binary vectors \( z \in \mathbb{R}^{\binom{n}{2}} \) that correspond to matchings. Since \( M \) is the convex hull of such vectors, we can write every \( x \in M \) as \( x = \sum_{z \in T} \alpha_z \cdot z \), where \( \alpha_z \geq 0 \) for all \( z \) and \( \sum_{z \in Z} \alpha_z = 1 \). Thus, if \( x \in M \) maximizes \( L(x) \), then we have

\[
L(x) = L \left( \sum_{z \in T} \alpha_z \cdot z \right) = \sum_{z \in T} \alpha_z \cdot L(z).
\]

So some \( z \in Z \) must also achieve the maximum \( L(z) = L(x) \).

Now, \( z \) must correspond to a valid matching of the graph, since by the definition of \( L \), if this valid matching contains an edge \( e \) that is not in the graph, setting \( z_e = 0 \) gives another point in the polytope whose value under \( L \) is even larger.

So we have shown that there is a single polytope—the matching polytope—such that every \( n \)-vertex graph defines a linear function whose maximum value on the polytope is precisely the maximum size of a matching in the graph.

Unfortunately, the matching polytope has an exponential number of facets, so we cannot use linear programming to efficiently find the size of the largest matching in polynomial time this way.

\[\text{Edmonds, 1965}\]

We shall see in Exercise 11.7 that one can come up with \( O(n^2) \) equations to work with the bipartite matching polytope.
Extension Complexity of a Polytope

The complexity of solving optimization problems on polytopes is related to the number of facets of the polytope. Therefore, it is important to find polytopes that encode computational problems and have a small number of facets. This would allow us to understand the limitation of this method and identify the cases in which this approach fails to produce efficient algorithms.

A generic way to do this is via extensions of the polytope. A polytope $Q \subseteq \mathbb{R}^k$ is an extension of a polytope $P \subseteq \mathbb{R}^d$ if there is a linear map $L : \mathbb{R}^k \to \mathbb{R}^d$ such that $L(Q) = P$. The extension complexity of $P$ is the minimum number of facets achieved by any extension of $P$.

There are many polytopes that admit non-trivial extensions—extensions with fewer facets. See Figure 11.5 for an example.

In term of proving impossibility results, the following lower bound always holds:

**Claim 11.9.** The extension complexity of a polytope with $n$ faces is at least $\log n$.

**Proof.** Suppose $P$ is the projection of a polytope $Q$. By Theorem 11.4, every face of $P$ is the projection of a face of $Q$. Every face of $Q$ is the intersection of a subset of the facets of $Q$. So if $Q$ has $k$ facets, $P$ can have at most $2^k$ faces.

This lower bound is often too weak to be useful. Later on, we shall develop tools for proving stronger lower bounds for specific cases.

Next, we explore some examples of natural polytopes that have small extension complexity.

**Regular Polygons**

A polygon is a polytope in $\mathbb{R}^2$. Consider polygons with $n$ facets where $n = 2^k$ is a power of 2. One can show that there are such polygons whose extension complexity is at least $\Omega(\sqrt{n})$. Here we show that if the polygon is sufficiently symmetric, then it has low extension complexity; when the polygon is regular, its extension complexity is $O(\log n)$.

The key idea is that one can mirror a polygon without increasing its extension complexity by much. Consider any polygon $P \subseteq \mathbb{R}^2$ which is defined by $P = \{x : Ax \leq b\}$. By applying a rotation and a translation, we can assume without loss of generality that $x_1 \leq 0$ is an inequality defining a facet of $P$. Define a new polytope $Q \subseteq \mathbb{R}^3$ by replacing each inequality

$$A_{i,1} \cdot x_1 + A_{i,2} \cdot x_2 \leq b_i$$

Figure 11.6: An octagon can be built with 3 reflections.
of $P$ with the inequality
\[ A_{i,1} \cdot x_3 + A_{i,2} \cdot x_2 \leq b_i, \]
for $Q$, and add in the inequalities $-x_3 \leq x_1 \leq x_3$ to $Q$. Let $\pi$ be the projection map defined by
\[ \pi(x_1, x_2, x_3) = (x_1, x_2), \]
and define the mirror of $P$ to be
\[ \mu(P) = \pi(Q). \]

The polytope $\mu(P)$ is the union of $P$ and its reflection with respect to the line $x_1 = 0$.

The number of inequalities defining $Q$ is only 2 more than the number of inequalities defining $P$, but the number of facets of $\mu(P)$ may be a factor of 2 larger than the number of facets of $P$!

We can now construct a regular polygon with $n = 2^k$ facets using $k = \log n$ mirror operations. We start with an isosceles triangle $P_0$ with one angle of $2\pi/n$—this requires only three inequalities to define. Mirror $P_0$ to get a 4-gon $P_1 = \mu(P_0)$, mirror the 4-gon $P_1$ to get a hexagon $P_2 = \rho(P_1)$, and so forth, until the full polygon is constructed.

So, every such polygon has extension complexity at most $O(\log n)$.

By Fact 11.9, the extension complexity of such a polygon cannot be less than $\log n$, so up to constant factors, this is the best we can hope for.

**Permutahedron**

For every permutation $\sigma : [n] \to [n]$, define the point $p^\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n)) \in \mathbb{R}^n$. The **permutahedron** is the convex hull of these $n!$ points.

The dimension of this polytope is $n - 1$. To see this, first observe that the permutahedron lies in the hyperplane $\sum_{i=1}^n x_i = \binom{n}{2}$, so its dimension is at most $n - 1$. To see that the dimension is at least $n - 1$, let $p^{\text{id}} = (1, 2, 3, \ldots, n)$ be the point corresponding to the identity permutation, and let $\sigma_2, \ldots, \sigma_n$ be the $n - 1$ permutations obtained by swapping 1 with 2, $\ldots$, $n$:

\[ \sigma_i(k) = \begin{cases} 1 & \text{if } k = i, \\ i & \text{if } k = 1, \\ \sigma(k) & \text{otherwise}. \end{cases} \]

The $n - 1$ points of the form $p^{\text{id}} - p^{\sigma_i}$ are linearly independent, since $p^{\text{id}} - p^{\sigma_i}$ is the only such vector with a non-zero entry in the $i$'th
coordinate. This proves that the dimension of the permutahedron is at least \( n - 1 \).

Although the permutahedron has \( n! \) vertices, it has much fewer facets:

**Lemma 11.10.** The permutahedron is the set of points satisfying the conditions:

\[
\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} i = \binom{n}{2},
\]

\[
\sum_{i \in S} x_i \geq \sum_{i=1}^{\lfloor |S| \rfloor} i \quad \text{for all sets } S \subseteq [n] \text{ with } 0 < |S| < n.
\]

The facets of the permutahedron correspond to these \( 2^n - 2 \) inequalities.

**Proof.** Let \( Q \) denote the polytope defined by the constraints. Every permutation satisfies the constraints, so the permutahedron is contained in \( Q \). Let us prove that every point of \( Q \) is in the permutahedron.

Suppose \( x \in Q \). We shall show that \( x \) is in the convex hull of the points corresponding to the permutations. The key claim is:

**Claim 11.11.** If \( x \) satisfies the constraints

\[
\sum_{i \in S} x_i \geq \sum_{i=1}^{\lfloor |S| \rfloor} i \quad \text{for all sets } S \subseteq [n] \text{ with } 0 < |S| < n,
\]
then there is a permutation \( \sigma \), and a positive number \( \epsilon > 0 \), such that the point \( x - \epsilon \cdot p^\sigma \) also satisfies all of the constraints.

Repeatedly applying this claim, we see that \( x \) must lie in the conical hull of the points \( p^\sigma \) given by the permutation. Indeed, in each step, we reduce the sum of the entries \( \sum_{i=1}^n x_i \), until \( x \) is expressed as a conical combination of the permutations. But this implies that if \( x \in Q \), then \( x \) must lie in the convex hull of the permutations, since if

\[
x = \sum_{\sigma} \mu_\sigma \cdot p^\sigma,
\]

then

\[
\binom{n}{2} = \sum_{i=1}^n x_i = \sum_{\sigma} \mu_\sigma \sum_{i=1}^n p_i^\sigma = \binom{n}{2} \cdot \sum_{\sigma} \mu_\sigma,
\]

so \( \sum_{\sigma} \mu_\sigma = 1 \).

To see that each of the inequalities gives a facet, fix a set \( S \subset [n] \) with \( 0 < |S| < n \), and consider the points of the permutahedron satisfying the constraint corresponding to \( S \) with equality. These points form a face. For simplicity, suppose \( S = \{1, 2, \ldots, t\} \). Let \( p^{id} = (1, 2, \ldots, n) \) denote the point corresponding to the identity permutation. For \( i = 2, 3, \ldots, t, t + 2, t + 3, \ldots, n \), define \( \sigma_i \) to be the permutation that swaps the \( i \)'th element with 1 if \( i \leq t \), or the \( i \)'th element with \( t + 1 \) if \( i > t \):

\[
\sigma_i(k) = \begin{cases} 
1 & \text{if } i \leq t, k = i, \\
i & \text{if } i \leq t, k = 1 \\
t + 1 & \text{if } i > t, k = i, \\
i & \text{if } i > t, k = t + 1, \\
k & \text{otherwise.}
\end{cases}
\]

Then we see that the \( n - 2 \) vectors \( p^\sigma - p^{id} \) are linearly independent, and all of these points belongs to the face. Thus, the face has dimension \( n - 2 \), and so must be a facet.

Similarly, one can argue that every facet of the polytope corresponds to some set \( S \).

It remains to prove the claim.

**Proof of Claim 11.11.** Suppose \( x \) satisfies \( \sum_{i \in S} x_i = \sum_{|S|} i \) and \( \sum_{i \in T} x_i = \sum_{|T|} i \), for two distinct sets \( S, T \). We claim that we must
have $S \subseteq T$ or $T \subseteq S$. Otherwise we would have

$$\sum_{i \in S \cup T} x_i = \sum_{i \in S} x_i + \sum_{i \in T} x_i - \sum_{i \in T \cap S} x_i \leq |S| + |T| - \sum_{i \in T \cap S} i$$

using the inequality for $T \cap S$.

$$= \sum_{i=1}^{|S|} i + \sum_{i=|T \cap S|+1}^{|T|} i < \sum_{i=1}^{|T \cap S|} i$$

since $|T \cap S| < |S|$.

contradicting the constraint for $S \cup T$.

Thus, the sets that give constraints that $x$ satisfies with equality can be arranged into a chain: $S_1 \subset S_2 \subset \ldots \subset S_k$. Let $\sigma$ be a permutation with $\sigma(S_i) = S_j$ for $i = 1, 2, \ldots, k$. This permutation also satisfies the same equations with equality. For a small enough $\epsilon > 0$, we must have that $x - \epsilon p^\sigma \in \mathcal{Q}$, as required.

---

5 Goemans, 2015

6 Rado, 1952

Although the permutahedron has $2^n - 2$ facets, it is known\(^5\) that its extension complexity is $O(n \log n)$. This bound is tight: the polytope has $n! = 2^{\Omega(n \log n)}$ vertices, so its extension complexity must be at least $\Omega(n \log n)$ by Fact 11.9.

Here we prove\(^6\) that its extension complexity is at most $n^2$. Define the polytope $\mathcal{Q}$ using the inequalities:

$$Y_{i,j} \geq 0 \quad \text{for } i, j \in [n]$$

$$\sum_{i=1}^n Y_{i,j} = 1 \quad \text{for all } j \in [n]$$

$$\sum_{j=1}^n Y_{i,j} = 1 \quad \text{for all } i \in [n]$$

$\mathcal{Q}$ is defined by $n^2$ inequalities, so it has at most $n^2$ facets.

We shall prove that the permutahedron is a projection of $\mathcal{Q}$, and so its extension complexity is at most $n^2$. Let $v$ be the column vector $(1, 2, \ldots, n)^T$.

**Claim 11.12.** $Yv$ is an element of the permutahedron if and only if $Y \in \mathcal{Q}$.

**Proof.** Each permutation $\sigma$ corresponds to a boolean permutation matrix $Y^\sigma \in \mathcal{Q}$ where $Y^\sigma_{i,j} = 1$ if and only if $\sigma(i) = j$. Since $p^\sigma = Y^\sigma v$, every element of the permutahedron can be realized as $Yv$ for some $Y \in \mathcal{Q}$. This proves that every point of the permutahedron can be expressed as $Yv$ for some $Y \in \mathcal{Q}$, since a convex combination of the permutations can be obtained by taking the appropriate convex combination of the permutation matrices.
In the other direction, for any $Y \in Q,$

$$\sum_{i=1}^{n} (Yv)_i = \sum_{i=1}^{n} \sum_{j=1}^{n} Y_{i,j} \cdot j = \sum_{j=1}^{n} j,$$

So, every point of the form $Yv$ satisfies the first constraint of the permutahedron. It only remains to check that $Yv$ satisfies the inequalities of the permutahedron. For any set $S \subseteq [n]$ with $0 < |S| = k < n,$ we want to prove

$$\sum_{i \in S} (Yv)_i \geq \sum_{i=1}^{k} i.$$

Write

$$\sum_{i \in S} (Yv)_i = \sum_{j=1}^{n} \alpha_j \cdot j,$$

with $\alpha_j = \sum_{i \in S} Y_{i,j}.$ For all $j,$ we have

$$0 \leq \alpha_j = \sum_{i \in S} Y_{i,j} \leq \sum_{i=1}^{n} Y_{i,j} \leq 1,$$

and we have

$$\sum_{j=1}^{n} \alpha_j = \sum_{i \in S} \sum_{j=1}^{n} Y_{i,j} = k.$$

Under these constraints, the vector $(\alpha_1, \ldots, \alpha_n)$ that minimizes $\sum_{j=1}^{n} \alpha_j \cdot j$ has $\alpha_1 = \ldots = \alpha_k = 1$ and $\alpha_{k+1} = \ldots = \alpha_n = 0,$ as required. \hfill \square

**Polytopes from Boolean Circuits**

The connection between polytopes and algorithms also goes the other way—efficient algorithms lead to efficient ways to represent polytopes. To explain this connection, we need the concept of a separating polytope. Given a boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\},$ we say that the polytope $P \subseteq \mathbb{R}^n$ is separating for $f$ if $f(x) = 1$ if and only if $x \in P.$

**Claim 11.13.** If $f$ can be computed by a circuit with $s$ gates, then there is a polytope separating $f$ that has extension complexity at most $O(s).$

**Proof.** Consider the polytope $P$ obtained from the circuit as follows. For every intermediate gate $g$ in the circuit let $v_g$ be a variable, and let $v_1, \ldots, v_n$ be the variables corresponding to the $n$ inputs to $f.$ For each of these variables, we have the constraints:

$$0 \leq v_g \leq 1.$$
If $g = \neg h$, we add the constraint
\[ v_g = 1 - v_h. \]
If $g = h \lor r$, we add the constraints
\[ v_g \geq v_h, \]
\[ v_g \geq v_r, \]
\[ v_g \leq v_h + v_r. \]
If $g = h \land r$, we add the constraints
\[ v_g \leq v_h, \]
\[ v_g \leq v_r, \]
\[ v_g \geq v_h + v_r - 1. \]
Finally, we add the constraint $v_f = 1$, where $f$ denotes the gate computing the output of the circuit.

The constraints above ensure that whenever $x \in \{0, 1\}^n$ is boolean, the values $v_g$ must also be boolean, and must be equal to the values of the corresponding gates in the circuit.

When this polytope is projected onto the inputs, we obtain a polytope $P$ that we claim separates $f$. Indeed, we see that if $f(x) = 1$, then $x \in P$, since we can assign all of the variables the values computed in the circuit, and these values satisfy the inequalities we defined. On the other hand, if $f(x) = 0$, then the only way to satisfy all the constraints is to set $v_f = 0$, and so $x$ is not in the polytope.

\[ \square \]

**Slack Matrices**

A useful tool for understanding the extension complexity of a polytope is the concept of a slack matrix.

A slack matrix of a polytope $P \subset \mathbb{R}^d$ is a matrix that captures some key properties of the polytope. It is defined with respect to an $n \times d$ matrix $A$ and an $n \times 1$ vector $b$ such that $P$ is contained in the polytope $\{x : Ax \leq b\}$, and with respect to a finite set of points $V = \{v_1, \ldots, v_k\} \subseteq P$.

**Definition 11.14.** The slack matrix of the polytope with respect to $A, b, V$ is the $n \times k$ matrix $S$ with
\[ S_{i,j} = b_i - A_i \cdot v_j \]
where $A_i$ is the $i$'th row of $A$. 

\[ \text{Figure 11.10: A numbering of the facets and vertices of the } 1 \times 1 \times 1 \text{ cube. The slack matrix corresponding to the vertices and the inequalities defining the facets:} \]

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
2 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
3 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
4 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
5 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
6 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0
\end{bmatrix}
\]
The slack matrix is a non-negative matrix—all of its entries are non-negative, since by the assumptions on \( A, b, V \), we always have \( A_i v_j \leq b_i \). If all the rows of \( A \) are normalized to have length 1, then \( S_{i,j} \) is the distance of \( v_j \) from the hyperplane defined by \( A_i, b_i \)—it is the “slack” of the \( i \)'th inequality. A single polytope can have many different slack matrices, depending on the choice of \( A, b, V \).

The extension complexity of a polytope \( P \) is determined by its slack matrices\(^7\)—it is equal to maximum non-negative rank\(^8\) achievable by a slack matrix of \( P \).

**Theorem 11.15.** If \( P \) has extension complexity \( r \), then every slack matrix of \( P \) has non-negative rank at most \( r + 1 \). Conversely, suppose \( P \) is bounded, and \( P = \{ x : Ax \leq b \} \). Suppose the slack matrix corresponding to \( A, b \) and the vertex set of \( P \) has non-negative rank \( r \), then \( P \) has extension complexity at most \( r \).

The theorem gives a powerful way to prove both upper and lower bound on the extension complexity of polytopes. The lower bounds are usually proved using ideas inspired by communication complexity.

**Proof.** First, suppose the extension complexity of the polytope \( P \subseteq \mathbb{R}^d \) is \( r \). Then, there is a polytope \( Q \subseteq \mathbb{R}^{\ell} \) with \( r \) facets, and a linear transformation \( L \), represented as a \( d \times \ell \) matrix, such that \( P = L(Q) \). Without loss of generality, possibly by modifying the linear transformation, we can assume that \( Q \) has full dimension. By Fact 11.2, \( Q \) can be expressed as

\[
Q = \{ x : Cx \leq e \},
\]

where \( C \) is an \( r \times \ell \) matrix, and \( e \) is an \( r \times 1 \) vector. Thus we have \( P = \{ Lx : Cx \leq e \} \).

Now, let \( S \) be the \( n \times k \) slack matrix of \( P \) with respect to \( A, b \) and the set of points \( V = \{ v_1, \ldots, v_k \} \). For each \( i \in [n] \), since \( A_i, b_i \) give a valid inequality for \( P \), they must also give a valid inequality \( A_i L x \leq b_i \) for \( x \in Q \). By Fact 11.1, this inequality can be proved by taking linear combinations of the inequalities defining \( Q \)—there must be a non-negative \( 1 \times r \) vector \( u_i \) such that

\[
u_i C = A_i L_i,
\]

and

\[
u_i e \leq b_i \Rightarrow b_i = u_i e + a_i,
\]

with \( a_i \geq 0 \). Let \( w_1, \ldots, w_k \in Q \) be such that \( L w_j = v_j \) for \( j = 1, 2, \ldots, k \). Thus, we have

\[
S_{i,j} = b_i - A_i v_j = u_i e + a_i - u_i C w_j = u_i (e - C w_j) + a_i.
\]

\(^7\) Yannakakis, 1991

\(^8\) We already discussed non-negative rank in Chapter 2.
This representation of the entries of $S$ allows us to bound the non-negative rank of $S$. Let $U$ be the $n \times (r + 1)$ non-negative matrix whose $i$'th row is given by
\[
\begin{bmatrix}
u_i \\ r_i \end{bmatrix},
\]
and let $W$ be the $(r + 1) \times k$ non-negative matrix whose $j$'th column is given by
\[
\begin{bmatrix}
e - Cw_j \\ 1 \end{bmatrix}.
\]
Thus $S = UW$, proving that the non-negative rank of $S$ is at most $r + 1$.

Conversely, suppose $P = \{x : Ax \leq b\}$, and let $V = \{v_1, \ldots, v_k\}$ be the vertex set of $P$. Suppose the slack matrix can be expressed as $S = UW$, where $U$ has $r$ columns and $W$ has $r$ rows, both with non-negative entries. We claim that $P$ is the projection of the polytope $Q = \{(x, y) : Ax + Uy = b, y \geq 0\}$ to the first $d$ coordinates, which has at most $r$ facets. Indeed, every vertex $v_j \in P$ corresponds to the column $W_j$ of $W$, and the column of $S$ that corresponds to $v_j$ is $UW_j = b - Av_j$. So,
\[Av_j + UW_j = b,\]
which means that every $v_j$ is in the projection of $Q$, since $W_j \geq 0$. Since $P$ is bounded, this implies that $P$ is contained in the projection of $Q$. On the other hand, the projection of $Q$ is contained in $P$, since $b = Ax + Uy \geq Ax$ for all $(x, y) \in Q$.

Now let us explore some examples where Theorem 11.15 allows to prove bounds on the extension complexity of polytopes.

**Upper Bounds on Extension Complexity**

**Spanning Tree Polytope**

The spanning tree polytope is the convex hull of all trees of size $n$:
There is a variable $x_e$ for every potential edge $e \subseteq [n]$ of size 2. Every tree gives a point $x$ by setting $x_e = 1$ if $e$ is an edge of the tree, and 0 otherwise. The spanning tree polytope is the convex hull of all of these points. Its vertices correspond to the spanning trees.

The facets of the spanning tree polytope correspond\(^9\) to the inequalities:
\[
\begin{align*}
\sum_x x_e &= n - 1 \\
\sum_{e \subseteq S} x_e &\leq |S| - 1 \quad \text{for every } S \subseteq [n]
\end{align*}
\]

\(^9\) Edmonds, 1971

A tree is a connected acyclic graph.

The number of edges in any subset is at most the size of the subset minus 1.
Each facet of the polytope corresponds to a subset \( S \subseteq [n] \), and each vertex corresponds to a spanning tree \( T \). The slack of the pair \( S, T \) is exactly \( |S| - 1 - k \), where \( k \) is the number of edges of \( T \) that are contained in \( S \). In the case that \( T \) is rooted at a vertex \( a \in S \), the slack is the number of children in \( S \) whose parents are not in \( S \).

Motivated by this observation, we show how to encode the slack matrix using a small non-negative factorization. For every tree \( T \), define the vector \( v \) in \( \mathbb{R}^{(n)^3} \) as follows. For distinct \( a, b, c \in [n] \), set

\[
v_{a,b,c} = \begin{cases} 1 & \text{if } b \text{ is the parent of } c \text{ when } T \text{ is rooted at } a \\ 0 & \text{otherwise.} \end{cases}
\]

For every set \( S \), define the vector

\[
 u_{a,b,c} = \begin{cases} 1 & \text{if } a = \min S, b \notin S \text{ and } c \in S \\ 0 & \text{otherwise.} \end{cases}
\]

Thus \( \sum_{a,b,c} u_{a,b,c}v_{a,b,c} \) is exactly the slack of \( T \) from the facet of the set \( S \). So, setting \( U \) to be the matrix whose rows correspond to the vectors \( u \) for each set \( S \), and \( V \) to be the matrix whose columns correspond to the vectors \( v \) for each tree \( T \), we can express the slack matrix as \( UV \).

This proves that the non-negative rank of the slack matrix is at most \( (n)^3 \). By Theorem 11.15, the extension complexity of the spanning tree polytope is at most \( (n)^3 \).

**Lower Bounds on Extension Complexity**

Our main tool for proving lower bounds on the extension complexity of polytopes are the entropy based techniques discussed in Chapter 6.

**Separating Polytopes**

We start with a connection between proving lower bounds on non-negative rank and proving circuit lower bounds—a notoriously difficult problem.

Let \( f : \{0,1\}^n \to \{0,1\} \). Consider the \( |f^{-1}(0)| \times |f^{-1}(1)| \) matrix \( M^\epsilon \) whose entries are indexed by inputs \( x \in f^{-1}(0) \) and \( y \in f^{-1}(1) \), and whose \((x,y)\)’th entry is

\[
 M^\epsilon_{x,y} = \Delta(x,y) - \epsilon,
\]

where \( \Delta(x,y) \) is the Hamming distance between \( x, y \).
We claim\(^{10}\):

**Theorem 11.16.** If \(\text{rank}_+(M^e) \geq k\) for all \(e > 0\), then the extension complexity of every separating polytope for \(f\) is at least \(k - 1 - 2n\).

**Proof.** Let \(P_0\) be a separating polytope for \(f\). That is, \(f^{-1}(1) \subseteq P_0\) and \(f^{-1}(0) \cap P_0 = \emptyset\).

Let \(P\) be the intersection of \(P_0\) with the solid cube \([0,1]^n\). Since the cube can be defined with \(2n\) inequalities, the extension complexity of \(P\) is at most \(2n\) plus that of \(P_0\).

Consider any \(x \in f^{-1}(0)\). Extend the Hamming distance from \(x\) to all the solid cube \([0,1]^n\) by defining the function

\[
\ell_x(v) = \sum_{i=1}^n (1-x_i)v_i + x_i(1-v_i)
\]

for \(v \in [0,1]^n\). This is an affine function in \(v\), and \(\ell_x(v) = 0\) for \(v \in [0,1]^n\) if and only if \(x = v\). Since \(x \not\in P\), there is \(\epsilon_x > 0\) so that for all \(v \in P\) we have \(\ell_x(v) > \epsilon_x\).

Let \(\epsilon = \min \{\epsilon_x : x \in f^{-1}(0)\}\). Thus, for all \(x \in f^{-1}(0)\) and \(v \in P\) we have \(\ell_x(v) > \epsilon\). We have obtained \(|f^{-1}(0)|\) inequalities that all points in \(P\) satisfy. The matrix \(M^e\) is the slack matrix of \(P\) with respect to these inequalities and the set of points \(f^{-1}(1)\). Theorem 11.15 completes the proof. \(\square\)

**Correlation Polytope**

The **correlation** polytope \(C_n\) is the convex hull of cliques. More precisely, there are \(n + \binom{n}{2} = \binom{n+1}{2}\) variables of the form \(x_T\) for \(T \subseteq [n]\) of size 1 or 2. For every \(A \subseteq [n]\), define the point

\[
x_A^T = \begin{cases} 
1 & \text{if } T \subseteq A, \\
0 & \text{otherwise.}
\end{cases}
\]

The correlation polytope is the convex hull of all these \(2^n\) points. Each point of the correlation polytope can be thought of as a lower triangular matrix whose rows and columns are labelled by elements of \([n]\).

**Claim 11.17.** \(C_n\) has full dimension \(\binom{n+1}{2}\).

**Proof.** For every set \(\{i\}\) of size one, \(C\) contains the unit vector \(x^{\{i\}}\) in the corresponding direction. For every set \(\{i,j\}\) of size two, every affine subspace containing \(C\) also contains the unit vector \(x_{\{i,j\}} = x^{\{i\}} - x^{\{j\}}\). These are \(\binom{n+1}{2}\) linearly independent vectors. \(\square\)

We can use the lower bound we proved on non-negative rank, which was proved using ideas from communication complexity, to prove:

\(^{10}\) Hrubeš, 2016

Since the minimal extension complexity of a separating polytope is at most linear in the size of the smallest circuit computing \(f\), we get that the infimum \(\inf \{\text{rank}_+(M^e) : e > 0\}\) is a lower bound on the circuit complexity of \(f\).
Theorem 11.18. The extension complexity of $C_n$ is $2^{\Omega(n)}$.

Consider the inequalities:

$$\sum_{i \in B} x(i) \leq 1 + \sum_{T \in [2]^B} x_T$$

for all non-empty $B \subseteq [n]$. 

For each vertex $x^A$ of $C_n$, the left hand side is exactly $|A \cap B|$, and the right hand side is exactly $1 + (|A| + |B|)$, so the inequality always holds. Since the inequality is valid for all the vertices, it is valid for all the points of $C_n$.

Consider the slack matrix $S$ for $C_n$ that correspond to these inequalities, and the vertices of the polytope. We see that $S_{A,B} = 0$ when $A$ and $B$ intersect in one element, and $S_{A,B}$ is 1 when $A$ and $B$ are disjoint. For a parameter $0 \leq \delta \leq 1$, suppose we are given a $2^n \times 2^n$ non-negative matrix $S$ whose rows and columns are indexed by sets $x, y \subseteq [n]$, such that

$$S_{x,y} \begin{cases} = 1 & \text{if } x, y \text{ are disjoint}, \\ \leq 1 - \delta & \text{if } x, y \text{ are not disjoint}. \end{cases}$$

When $\delta = 1$, the matrix $A$ is the disjointness matrix. This is because $S_{x,y} \leq 1 - \delta$ implies $S_{x,y} = 0$. The disjointness matrix has full rank, and hence full non-negative rank. When $\delta = 0$, the matrix may have non-negative rank 1.

Here we prove that the rank of the matrix is at least exponential in $\delta n$. In fact, we shall prove the following stronger theorem, which implies Theorem 11.18.

Theorem 11.19. If $S$ is non-negative matrix with $S_{x,y} = 1$ when $x, y$ are disjoint and $S_{x,y} \leq 1 - \delta$ when $|x \cap y| = 1$ then $\text{rank}_+(S) \geq 2^{\Omega(\delta n)}$.

The proof of the theorem is an adaptation of the lower bound we proved for the randomized communication complexity of disjointness.

Proof. Consider the distribution on $x, y$ given by

$$q(xy) = \frac{S_{x,y}}{\sum_{a,b} S_{a,b}}.$$

If $S$ has non-negative rank $r$, then $S$ can be expressed as $S = \sum_{m=1}^r S(m)$, where $S(m)$ is a non-negative rank 1 matrix. In other words, $q(xy)$ can be expressed as a convex combination of $r$ product distributions, by setting

$$q(xy|m) = \frac{S(m)_{x,y}}{\sum_{a,b} S(m)_{a,b}},$$

If the entries corresponding to intersecting sets are allowed to be larger than the entries corresponding to disjoint sets, that matrix may have exponentially smaller rank. For example, if $S_{x,y} = |x \cap y| + 1$, the matrix has non-negative rank $n + 1$. This shows that for $\delta \leq 0$, the non-negative rank of $S$ can be quite small.

When $\delta < 1$, Theorem 11.19 corresponds to proving a lower bound on the extension complexity of any polytope approximating the correlation polytope.
and
\[ q(m) = \frac{\sum_{a,b} S(m)_{a,b}}{\sum_{a,b} S_{a,b}}. \]

Let \( D \) denote the event that the sets \( X, Y \) sampled in this distribution are disjoint. The key step is to prove that for every \( i \in [n] \),
\[ l(X_i : M|X_{<i}, Y_{\geq i}, D) + l(Y_i : M|X_{\leq i}, Y_{>i}, D) \geq \Omega(\delta^4) \quad (11.1) \]

Before proving (11.1), we show how to use it. Since \( S_{x,y} = 1 \) for disjoint sets \( x, y \), we know that \( q(xy|D) \) is the uniform distribution on all pairs of disjoint sets. In particular, conditioned on \( D \), the coordinates \((X_1, Y_1), \ldots, (X_n, Y_n)\) are independent. By Lemma 6.15, we get that
\[ 2 \log r \geq \sum_{i=1}^{n} l(X_i : M|X_{<i}, Y_{\geq i}, D) + l(Y_i : M|X_{\leq i}, Y_{>i}, D) \geq \Omega(\delta^4 n), \]
proving that \( r \geq 2^{\Omega(\delta^4 n)} \) as required.

Let \( Z = (M, X_{<i}, Y_{\geq i}) \). Let \( U \) denote the event that \( X \cap Y \subseteq \{i\} \).
Let \( p(xym) = q(xym|U) \). Note that for every \( z \), \( p(xy|z) \) is a product distribution. For fixed \( z \), let \( \alpha_z \) denote the statistical distance of \( p(x_i, y_i|z) \) from uniform. Set
\[ l(X_i : M|X_{<i}, Y_{\geq i}, D) + l(Y_i : M|X_{\leq i}, Y_{>i}, D) = 2\gamma^4/3. \]

Let \( G \) denote the set of \( z \) for which \( \alpha_z \leq 2\gamma \). Let \( Q \) denote the event \( i \notin X, i \notin Y \), and let \( I \) denote the event that \( i \in X, i \in Y \). We shall use Pinsker’s inequality to prove:

Claim 11.20. \( p(z \in G|Q) \geq 1 - 4\gamma \).

Whenever \( z \in G \),
\[ \frac{p(I, z)}{p(Q, z)} \geq \frac{1/4 - 2\gamma}{1/4 + 2\gamma} = \frac{1 - 8\gamma}{1 + 8\gamma}. \]

So, we have
\[ p(I) \geq \sum_{z \in G} p(I, z) \geq \frac{1 - 8\gamma}{1 + 8\gamma} \cdot \sum_{z \in G} p(Q, z) = \frac{1 - 8\gamma}{1 + 8\gamma} \cdot p(z \in G, Q) \geq \frac{1 - 8\gamma}{1 + 8\gamma} \cdot (1 - 4\gamma) \cdot p(Q) \geq (1 - O(\gamma)) \cdot p(Q). \]

On the other hand, the assumption on \( S \) implies that \( p(I) \leq (1 - \delta) \cdot p(Q) \), so we must have \( \gamma \geq \Omega(\delta) \).
To prove Claim 11.20, let $\beta_z$ denote the distance of $p(x_i|z)$ from uniform. We have
\[
\frac{2}{3} \cdot I(X_i: M|X_{<i}, Y_{>i}, Y_i = 0, \mathcal{D}) \leq I(X_i: M|X_{<i}, Y_{>i}, \mathcal{D}) \leq \frac{2}{3} \cdot \gamma^4.
\]
Since $p(z|y_i = 0)$ is identical to $q(z|y_i = 0, \mathcal{D})$, we can use convexity and Pinsker’s inequality (Lemma 6.10) to conclude
\[
\mathbb{E}_{p(z|y_i = 0)} [\beta_z] \leq \sqrt{\mathbb{E}_{p(z|y_i = 0)} [\beta_z^2]} \leq \sqrt{\gamma^4} = \gamma^2.
\]
In particular,
\[
\gamma \geq p(\alpha_m > \gamma|y_i = 0)
\geq p(x_i = 0|y_i = 0) \cdot p(\alpha_m > \gamma|x_i = 0 = y_i)
= p(\alpha_m > \gamma|x_i = 0 = y_i) \cdot \frac{1}{2},
\]
and so
\[
p(\beta_z > \gamma|x_i = 0 = y_i) \leq 2\gamma.
\]
A symmetric argument proves that the probability that the distance of $p(y_i|z)$ exceeds $\gamma$ is at most $2\gamma$. By the union bound, the probability that $x_i, y_i$ are $2\gamma$ close to uniform is at least $1 - 4\gamma$. \qed

### Matching Polytope

The matching polytope $M_n \subset \mathbb{R}^{\binom{n}{2}}$ is the convex hull of all matchings in a graph on $n$ vertices. Its coordinates are labelled by sets $\{u, v\} \subset [n]$ of size two. Here we prove\(^{11}\):

**Theorem 11.21.** The extension complexity of the matching polytope is at least $2\Omega(n)$.

Like the lower bound for the correlation polytope, the proof crucially relies on entropy based inequalities. To prove the theorem, we first identify the appropriate slack matrix of the matching polytope.

Consider the set of inequalities:
\[
\sum_{u < v \in A} z_{\{u,v\}} \leq \frac{|A| - 1}{2} \quad \text{for all } A \subseteq [n] \text{ with } |A| \text{ odd}.
\]
These inequalities hold for every matching $z$, because the number of edges contained in a set $A$ can be at most $|A|/2$, and since this number is an integer, it is at most $(|A| - 1)/2$ when $|A|$ is odd.

Now, suppose $n$ is even. Let $S$ be the slack matrix that corresponds to the inequalities defined above and the set of points in $M_n$ that correspond to perfect matchings (i.e., matchings of size $n/2$ that touch every vertex in the graph).

Given Theorem 11.15, it is enough to prove:

This is similar to the final step of the proof of Claim 6.16

\(^{11}\) Rothvoß, 2014

We make a slight change in notation here.
Lemma 11.22. $\text{rank}_+(S) \geq 2^{\Omega(n)}$.

The proof of the lower bound closely follows ideas developed to prove lower bounds on the randomized communication complexity of disjointness.

Consider the distribution on $(x, y)$ given by:

$$q(xy) = \frac{S_{xy}}{\sum_{i,j} S_{i,j}}$$

If $S$ has non-negative rank $r$, then $q(xy)$ can be expressed as a convex combination of $r$ product distributions: there is a distribution $q(m)$ on $[r]$, and for each $m \in [r]$, there is a product distribution $q(xy|m)$ on $(x, y)$, so that

$$q(xy) = \sum_m q(m) \cdot q(xy|m).$$

For ease of notation, it will be convenient to work with $4n + 6$ vertices. Let $A$ be a perfect matching and let $W$ be a subset of the vertices of size $2n + 3$ that cuts all of the edges of $A$. Let $F = (C, B_1, \ldots, B_n)$ be a uniformly random partition of $W$ such that $|C| = 3$, and $|B_i| = 2$ for all $i$. Let $A_i$ denote the set of edges of $A$ that touch $B_i$.

We say that a set $x$ is consistent with $F$ if $C \subseteq x \subseteq W$, and $x_i = x \cap B_i$ is not of size 1 for any $i$. We say that a perfect matching $y$ is consistent with $F$ if $y$ contains the 3 edges of $A$ cut by $C$, and for each $i$, either $A_i \subseteq y$, or $y$ matches $B_i$ to itself and matches the neighbors of $B_i$ under $A_i$ to themselves. We write $y_i$ to denote the edges of $y$ contained in $A_i$.

Let $(X, Y)$ be sampled according to the distribution $q$. Let $D$ denote the event that $(X, Y)$ are consistent with the partition $F$, and for each $j \neq i$, the edges of $y_j$ are not cut by $x_j$. So $D$ implies $U$.

It remains to prove $(11.2)$. Fix $i$ for the rest of the proof. Let $U$ denote the event that $(x, y)$ are consistent with the partition $F$, and for each $j \neq i$, the edges of $y_j$ are not cut by $x_j$. So $D$ implies $U$, but
under $\mathcal{U}$ the edges of $y_i$ may be cut by $x_i$. Let $Z$ denote the random variable $Z = (X_{<i}, Y_{>i}, B_{<i}, B_{>i})$. In fact, we prove the stronger statement that for each fixing $Z = z$, we have $\gamma \geq \Omega(1)$ where

$$\gamma_A/2 = I(X_i : M|Y_i,CzD) + I(Y_i : M|X_i,CzD).$$

Fix $z$ for the rest of the proof, and let

$$p(xym) = q(xym|z\mathcal{U}).$$

Given $c, m$, the distribution $p(x, y|cm)$ is supported on four possible values. Let $a_{cm}$ denote the distance of this distribution from the uniform distribution on these four values. Call a pair $(c, m)$ good if $a_{cm} \leq \gamma$. Denote by $\mathcal{G}$ the set of good pairs. Let $\mathcal{Q}$ denote the event that $X_i = \emptyset, Y_i \neq A_i$, and $\mathcal{I}$ denote the event that $X_i \neq \emptyset, Y_i = A_i$.

As in the lower bounds for disjointness and the correlation polytope, we shall use Pinsker’s inequality to prove:

Claim 11.23. $p((c, m) \in \mathcal{G}|\mathcal{Q}) \geq 1 - 4\gamma$.

We shall appeal to the combinatorial structure of matchings to argue that:

Claim 11.24. For any value of $m$, we have $|\{c : (c, m) \in \mathcal{G}\}| \leq 4$.

Let us see how to use these claims to complete the proof of the lemma. First, we need to understand the distribution of $p(x, y_i)$. If $A$ is odd sized set of vertices, and $z$ is a perfect matching with $k$ edges in $z$ that go from $A$ to its complement, we have

$$|A| = k + 2 \cdot \sum_{u < v \in A} z_{(u, v)}.$$
since every vertex of $A$ touches either one of the $k$ edges leaving $A$ or one of the edges staying within $A$. Thus, the slack $S_{A,z}$ is exactly

$$S_{A,z} = \frac{|A| - 1}{2} - \sum_{u<v \in A} z_{\{u,v\}} = \frac{k - 1}{2}. \quad (11.3)$$

We can therefore conclude the following. Let $L = (C, X_{<j}, X_{>j}, Y_{<j}, Y_{>j})$.
Then for every fixing of $L$, we have that the distribution of $X, Y$ under $p$ is determined by a $2 \times 2$ submatrix of $S$ of the form:

$$
\begin{array}{cccc}
Y_i \neq A_i & Y_i = A_i \\
X_i = \emptyset & 1 & 1 \\
X_i \neq \emptyset & 1 & 2 \\
\end{array}.
$$

(11.4)

Namely, the slack of the entry is either $1 = (3 - 1)/2$, when the 3 edges of $C$ cross the cut, or $2 = (5 - 1)/2$ when the edges of $C$ and $A_i$ cross the cut.

Here lies a crucial difficulty in the proof. In the disjointness matrix, the corresponding submatrix is of the form

$$
\begin{array}{cccc}
Y_i = 0 & Y_i = 1 \\
X_i = 0 & 1 & 1 \\
X_i = 1 & 1 & <1 \\
\end{array}.
$$

In that proof, we showed that there are many rectangles where the entry corresponding to $X_i = 1 = Y_i$ gets as much weight as the entry corresponding to $X_i = 0 = Y_i$, giving a contradiction. For the slack matrix of the matching polytope, the entry corresponding to intersections is larger than the entries corresponding to disjoint sets.

Now, it seems that there is a counterexample to our efforts. Consider the matrix $T$ whose entries are indexed by $u, v \in \{0, 1\}^n$, and where $T_{u,v} = \langle u, v \rangle + 1$. Then $T$ has non-negative rank $\leq n + 1$ by definition, yet it has the same structure as the slack matrix defined above. The freedom in the choice of $C$, and Claim 11.24 allow us to avoid this counterexample. Roughly speaking, we partition the weight of the 2 entry into many parts, thus getting a number smaller than 1. Whenever $x_{ij} \in \mathcal{G}$, we have that $p(x_{ij}|cm)$ is $\gamma$-close to uniform, so

$$
\frac{p(T, c, m)}{p(Q, c, m)} \geq \frac{1/4 - 2\gamma}{1/4 + 2\gamma} \geq \frac{1 - 8\gamma}{1 + 8\gamma}.
$$

(11.5)
Now we compute

\[
p(\mathcal{I}) = \sum_m p(\mathcal{I}, m) \geq \frac{1}{4} \cdot \sum_{(c,m) \in \mathcal{G}} p(\mathcal{I}, m)
\]

\[
= \frac{5}{3} \cdot \sum_{(c,m) \in \mathcal{G}} p(\mathcal{I}, m) \cdot p(c|\mathcal{I}, m)
\]

\[
= \frac{5}{2} \cdot \frac{1 - 8\gamma}{1 + 8\gamma} \cdot \sum_{(c,m) \in \mathcal{G}} p(Q, c, m)
\]

\[
\geq \frac{5}{2} \cdot \frac{1 - 8\gamma}{1 + 8\gamma} \cdot (1 - 4\gamma) \cdot p(Q).
\]

On the other hand, by (11.4), we have \(p(\mathcal{I}) \leq 2 \cdot p(\mathcal{Q})\). Since \(5/2 > 2\), we must have \(\gamma \geq \Omega(1)\).

Finally, we turn to proving each of the two claims.

**Proof of Claim 11.24.** Fix any value of \(m\). Under \(p\), the set \(C\) is a subset of size 3 in a universe of size 5. We claim that if \(c\) and \(c'\) are two sets with \(|c \cap c'| \leq 1\), then we cannot have \((c, m) \in \mathcal{G}\) and \((c', m) \in \mathcal{G}\).

Indeed, assume towards a contradiction that \((c, m), (c', m) \in \mathcal{G}\) are as in Figure 11.13. Since \(\sigma_{cm} \leq \gamma < 1/2\), we know that \(p(X_i = \emptyset|ct) > 0\), and the \(X\) shown in Figure 11.13 has positive probability conditioned on \(m\). Similarly, the edges shown in Figure 11.13 have positive probability conditioned on \(m\). However, since \(X_i, Y_i\) are independent conditioned on \(m\), both have positive probability conditioned on \(m\). But this cannot happen, since this configuration corresponds to an entry of \(S\) that is 0, so it has 0 probability in \(p\).

This means that all of the sets \(c, c'\) that are in \(\mathcal{G}\) must intersect in at least 2 elements. We claim that there can be at most 4 such sets. Indeed, take any two sets \(c_1 \neq c_2\) in such a family. Assume without loss of generality that \(c_1 = \{1, 2, 3\}\) and \(c_2 = \{1, 2, 4\}\). If every other subset \(c_3\) is contained in \([4]\), then indeed there are at most \(\binom{4}{3} = 4\) sets. Otherwise, there is a set \(c_3\) such that \(5 \in c_3\). Then \(\{1, 2\} \subset c_3\), otherwise \(c_3\) cannot share two elements with each of \(c_1, c_2\). Now, if there is a forth set \(c_4\) in the family then \(c_4\) cannot include both 1, 2, since then it will be equal to \(c_1, c_2\) or \(c_3\). But if \(c_4\) includes only one
of the elements of \( \{1, 2\} \), it will intersect one of \( c_1, c_2, c_3 \) in just one element, a contradiction.

**Proof of Claim 11.23.** Recall

\[
\mathbb{I}(X_i : M|Y_i CzD) + \mathbb{I}(Y_i : M|X_i CzD) = \gamma^4/2.
\]

Since both \( q(Y_i \neq A_i|zD) \) and \( q(X_i = \varnothing |zD) \) are at least \( \frac{1}{2} \), we get

\[
\mathbb{I}(X_i : M|Y_i C, Y_i \neq A_i, zD) + \mathbb{I}(Y_i : M|C, X_i = \varnothing, zD) \leq \gamma^4
\]

Since the events \( X_i = \varnothing \) and \( U \) together imply \( D \),

\[
p(xycm|X_i = \varnothing) = q(xycm|z, X_i = \varnothing, D).
\]

Let \( \beta_{cm} \) denote the statistical distance of \( p(x_i|cm) \) from uniform. By Pinsker’s inequality (Corollary 6.11):

\[
\mathbb{E}_{p(cm|X_i = \varnothing)} \left[ \beta_{cm} \right] \leq \sqrt{\mathbb{E}_{p(cm|X_i = \varnothing)} \left[ \beta_{cm}^2 \right]} \leq \sqrt{\gamma^4} \leq \gamma^2.
\]

This implies that \( p(\beta_{cm} > \gamma|X_i = \varnothing) \leq \gamma \). So

\[
p(\beta_{cm} > \gamma|X_i = \varnothing, Y_i \neq A_i) \leq \frac{p(\beta_{cm} > \gamma|X_i = \varnothing)}{p(Y_i \neq A_i|X_i = \varnothing)} \leq 2\gamma.
\]

A symmetric argument proves that the statistical distance of \( p(y_i|cm) \) is at most \( \gamma \) except with probability \( 2\gamma \). Since \( p(x_i, y_i|cm) \) is a product distribution, the claim then follows using the union bound.

**Exercise 11.1**

Prove Fact 11.1.

**Exercise 11.2**

Show that the extension complexity of any regular polygon in the plane with \( n \) facets is \( O(\log n) \).

**Exercise 11.3**

Show that the non-negative rank of the slack matrix of the permutohedron with respect to the \( 2^n - 2 \) inequalities from Lemma 11.10 and its \( n! \) vertices is \( O(n^2) \).

**Exercise 11.4**

Show that Theorem 11.16 is false if we remove the \( -2n \) from the lower bound on the extension complexity.
Exercise 11.5

The solid cube in $n$ dimensions is the convex hull of the set $\{0, 1\}^n$. Identify the facets of the solid cube. Is it possible that the extension complexity of the solid cube is $O(\sqrt{n})$?

Exercise 11.6

Show that the non-negative rank of the slack matrix of a regular $2^k$ sided polygon in the plane is at most $O(k)$ by giving an explicit factorization of the matrix into non-negative matrices.

Exercise 11.7

Given two disjoint sets $A, B$, each of size $n$, define the bipartite matching polytope to be the convex hull of all bipartite matchings: matchings where every edge goes from $A$ to $B$. Show that the extension complexity of the bipartite matching polytope is at most $O(n^2)$.

Exercise 11.8

Show that there is a $k$ for which the convex hull of cliques of size $k$ has extension complexity $2^{\Omega(n^2)}$. / 

Exercise 11.9

The cut polytope $K_n$ is the convex hull of all cuts in a graph. Here the number of variables in $\binom{n}{2}$, one variable for each potential edge $e \subseteq [n]$ of size 2. For every set $A \subseteq [n] + 1$ define the vertex

$$y_e^A = \begin{cases} 1 & \text{if } |e \cap A| = 1, \\ 0 & \text{otherwise.} \end{cases}$$

The cut polytope is the convex hull of all these points. Observe that $y_A = y_{A^c}$, so there are $2^{n-1}$ vertices. Prove that the extension complexity of the cut polytope is $2^{\Omega(n)}$. Hint: Find an invertible linear map that maps the cut polytope to the correlation polytope.

A vertex of the cut polytope:

\[
\begin{pmatrix}
\varepsilon_A & \varepsilon_A & \varepsilon_A & \varepsilon_A & \varepsilon_A & \varepsilon_A \\
\varepsilon_A & 1 & 0 & 1 & 0 & 1 \\
\varepsilon_A & 0 & 1 & 0 & 1 & 1 \\
\varepsilon_A & 1 & 0 & 1 & 0 & 1 \\
\varepsilon_A & 0 & 1 & 0 & 1 & 1 \\
\end{pmatrix}
\]
Distributed Computing

Distributed computing is the study of algorithms and protocols for computers operating in a distributed environment. In such an environment, there are \( n \) parties that are connected together by a communication network, yet no single party knows what the whole network looks like. Nevertheless, the parties wish to solve computational problems together.

More formally, the network is defined by an undirected graph on \( n \) vertices, where each of the vertices represents one of the parties. The parties communicate according to a protocol in order to achieve some common goal. Each protocol begins with each party knowing its own name, and perhaps some part of the input. The protocol proceeds in rounds. In each round, each of the parties can send a message to all of her neighbors.

The setup is often interesting even when the goal is to learn something about the structure of the network and there are no inputs besides their own names.

Coloring the Network

Suppose the parties in a distributed environment want to properly color the underlying graph—each party needs to choose its own color so that no two neighboring parties have the same color.

Here is another protocol\(^1\) that finds a proper coloring with a constant number of colors in \( O(\log^* n) \) rounds of communication. Initially, each party colors itself with its name. This is a proper coloring. The goal now is to iteratively reduce the number of colors.

In each round, the parties send all of their neighbors their current color. If \( a \in \{0, 1\}^t \) denotes the color of one of the parties in a round, and \( b, c \in \{0, 1\}^t \) denote the colors assigned to its neighbors, then the party sets \( i \) to be a number such that \( a_i \neq b_i \), and \( j \) to be a number

\(^1\) Cole and Vishkin, 1986

In general, networks may be asynchronous, and one can either charge or not charge for the length of each message. Moreover, one can also study the model in which some of the parties do not follow the protocol, or the protocol is disrupted by adversarial actions. In this chapter we stick to the model of synchronous networks, where we count both the number of rounds as well as the total communication. We also assume that all parties execute the protocol correctly, and there are no errors in the communication.
such that $a_j \neq c_j$. Its new color is set to be $(i, j, a_i, a_j)$. The new coloring is still proper. Indeed, if the neighbor whose color was $b$ also gets the color $(i, j, a_i, a_j)$, then we see that $b_i = a_i$, contradicting the choice of $i$.

In this way, the number of colors has been reduced from $t$ to $O(\lceil \log t \rceil^2)$. After $O(\log^* n)$ rounds, the number of colors is constant.

This coloring protocol can be generalized to handle arbitrary graphs of constant degree $d$. Any graph of degree $d$ can be colored using $d + 1$ colors. Here we give a protocol that uses $O(\log^* n)$ rounds to find a coloring using $O(d^2 \log d)$ colors.

The protocol relies on the following combinatorial lemma:

**Lemma 12.1.** For every $t > 0$, there is a sequence of $t$ subsets $T_1, \ldots, T_t \subseteq [m]$ with $m = 5d^2 \lceil \log t \rceil$ such that for any distinct $i_1, i_2, \ldots, i_{d+1} \in [t]$, the set $T_{i_1}$ is not contained in the union of $T_{i_2}, \ldots, T_{i_{d+1}}$.

*Linial, 1992*

**Proof.** Pick the $t$ sets at random from $[m]$, where each element is included in each set independently with probability $1/d$.

For a particular choice of $S_1, \ldots, S_{d+1}$ from the list of sets and $j \in [m]$, let $\mathcal{E}_j$ denote the event that $j \in S_1$ and $j \notin \bigcup_{\ell \geq 1} S_\ell$. It remains to show that with positive probability for every choice of distinct $S_1, \ldots, S_{d+1}$, one of the events $\mathcal{E}_1, \ldots, \mathcal{E}_m$ occurs.

The probability that $\mathcal{E}_j$ occurs is

$$\frac{1}{d} \left(1 - \frac{1}{d}\right)^d \geq \frac{d}{4}.$$  

Hence, the probability that no $\mathcal{E}_j$ occurs is at most

$$\left(1 - \frac{d}{4}\right)^m < e^{-d \log t}.$$  

On the other hand, the number of choices for $d + 1$ sets from the family is $\binom{t}{d+1} \leq t^{d+1} \leq 2^d \log t$. Thus, by the union bound, the probability that the family does not have the property we need is at most $e^{-d \log t} \cdot 2^d \log t < 1$.  

Now, in each round of the protocol, each party sends its current color to all its neighbors. If there are $t$ colors in a particular round, each party looks at the $d$ colors its received and associates each with a set from the family promised by Lemma 12.1. Its new color is an element that belongs to her own set but not to any of the others. Thus, the next round has at most $5d^2 \lceil \log t \rceil$ colors. Continuing in this way, the number of colors is reduced to $O(d^2 \log d)$ in $O(\log^* n)$ rounds.
**Computing the Diameter of the Network**

Suppose the parties in a network want to compute the diameter of the network; namely, the maximum distance between two vertices in the graph of the network. Here we show:

**Theorem 12.2.** Given any distributed algorithm for computing the diameter of an \(n\)-vertex graph, there is an input for which there are \(\Omega(n)\) edges on which \(\Omega(n^2)\) bits are transmitted in total.

This holds even if the goal is to distinguish whether the diameter of the graph is at most 2 or at least 3, and when the protocol is allowed to be randomized.

The proof is by reduction to the randomized communication complexity of disjointness, when there are just 2 parties. The idea for the reduction from \(n\) parties to 2 parties is quite general. Partition the \(n\) vertices of the graph to 2 parts, and think of one part as Alice and of the other part as Bob. The messages between the 2 parts can be viewed as communication between Alice and Bob.

Let \(X, Y \subseteq [n] \times [n]\) be two subsets of a universe of size \(n^2\). For every such pair of sets, we shall define a graph \(G_{X,Y}\). We show that if there is an efficient distributed algorithm for computing the diameter of \(G_{X,Y}\), then there is an efficient communication protocol for computing whether or not \(X\) and \(Y\) are disjoint.

The graph \(G_{X,Y}\) has \(4n + 2\) vertices (see Figure 12.1). Let \(A = \{a_1, \ldots, a_n\}, B = \{b_1, \ldots, b_n\}, C = \{c_1, \ldots, c_n\}\) and \(D = \{d_1, \ldots, d_n\}\) be disjoint cliques, each of size \(n\). Let \(v\) be a vertex that is connected to all the vertices in \(A \cup B\) and \(w\) be a vertex that is connected to all the vertices in \(C \cup D\). Connect \(v\) and \(w\) with an edge as well. For each \(i\), connect \(a_i\) to \(c_i\), and \(b_i\) to \(d_i\). Finally, connect \(a_i\) to \(b_j\) if and only if \((i, j) \notin X\), and \(c_i\) to \(d_j\) if and only if \((i, j) \notin Y\).

**Claim 12.3.** The diameter of \(G_{X,Y}\) is 2 if \(X, Y\) are disjoint, and 3 if \(X, Y\) are not disjoint.

**Proof.** The interesting parts of the claim are the distances between \(A\) and \(D\), and between \(B\) and \(C\)—the distances between all other pairs of vertices are at most 2. Here we focus on the distances between \(A\) and \(D\) the case of \(B\) and \(C\) is similar.
When \((i,j) \notin X\) or \((i,j) \notin Y\), the distance of \(a_i\) from \(d_j\) is at most 2; for example, if \((i,j) \notin X\), we have the path \(a_i \rightarrow b_j \rightarrow d_j\). Otherwise, \((i,j) \in X \cap Y\), and in this case the distance from \(a_i\) to \(d_j\) is at least 3.

Consider the protocol obtained when Alice simulates all the vertices in \(A, B\) and \(v\), and Bob simulates all the vertices in \(C, D\) and \(w\). This protocol solves the disjointness problem, and so has communication at least \(\Omega(n^2)\). This proves that the \(O(n)\) edges that cross from Alice’s part to Bob’s part must carry at least \(\Omega(n^2)\) bits of communication to compute the diameter of the graph.

**Computing the Girth of the Network**

Another basic measure associated with a graph is its girth, which is the length of the shortest cycle in the graph. Here we show:\n
**Theorem 12.4.** Any distributed protocol for computing the girth of an \(n\)-vertex graph must involve at least \(\Omega(n^22^{-O(\sqrt{\log n})})\) bits of communication.
The lower bound holds even if the goal is to detect if the girth is 3 or more—it is even hard to determine if there is a single triangle in the graph.

The proof is by reduction to disjointness in the number-on-forehead model with 3 parties. The reduction together with the lower bound from Theorem 5.12 complete the proof.

Suppose Alice, Bob and Charlie have 3 sets X, Y, Z ⊆ U written on their foreheads, where U is a set that we shall soon specify. We shall define a graph \( G_{X,Y,Z} \) that has a triangle if and only if \( X \cap Y \cap Z \) is non-empty.

The vertex set of \( G_{X,Y,Z} \) is \( A \cup B \cup C \), where \( A, B, C \) are disjoint sets, each of size \( 2n \). To construct \( G_{X,Y,Z} \) we need the coloring promised by Theorem 4.2. This is a coloring of \( [n] \) with \( 2^{O(\log n)} \) colors such that there are no 3-term monochromatic arithmetic progressions. Since such a coloring exists, there must be a subset \( Q \subseteq [n] \) of size \( n2^{-O(\log n)} \) that does not contain any non-trivial 3-term arithmetic progressions.

First define a graph\(^5\) \( G \) on the vertex set \( A \cup B \cup C \), where for each \( a \in A, b \in B, c \in C \),

\[
\begin{align*}
\{a, b\} & \in E(G) \iff b - a \in Q, \\
\{b, c\} & \in E(G) \iff c - b \in Q, \\
\{a, c\} & \in E(G) \iff \frac{c - a}{2} \in Q.
\end{align*}
\]

See Figure 12.2.

**Claim 12.5.** The graph \( G \) has at least \( n|Q| = n2^{-O(\log n)} \) triangles, and no two distinct triangles in \( G \) share an edge.

**Proof.** For each element \( q \in Q \), the vertices \( a \in A, a + q \in B, a + 2q \in C \) certainly form a triangle, as long as \( a + 2q \leq 2n \). No two of these triangles share an edge, since any edge of these triangles determines \( a, q \).

We claim that there are no other triangles. Indeed, if \( a, b, c \) was a triangle in the graph, then \( b - a = q_1 \in Q, c - b = q_2 \in Q \), and

\[
\frac{q_1 + q_2}{2} = \frac{c - b + b - a}{2} = \frac{c - a}{2} = q_3 \in Q.
\]

So \( q_1, q_3, q_2 \in Q \) form an arithmetic progression. The only way this can happen is if \( q_1 = q_2 = q_3 \). In this case we recover one of the triangles above.
The universe $U$ is the set of triangles in $G$. The graph $G_{X,Y,Z}$ is the subgraph of $G$ defined by

- $\{a,b\} \in E(G_{X,Y,Z}) \iff$ a triangle of $G$ containing $\{a,b\}$ is in $Z$,
- $\{b,c\} \in E(G_{X,Y,Z}) \iff$ a triangle of $G$ containing $\{b,c\}$ is in $X$,
- $\{a,c\} \in E(G_{X,Y,Z}) \iff$ a triangle of $G$ containing $\{a,c\}$ is in $Y$.

The following simple property holds:

**Claim 12.6.** $G_{X,Y,Z}$ contains a triangle if and only if $X \cap Y \cap Z \neq \emptyset$.

**Proof.** If $a,b,c$ are the vertices of a triangle in $X \cap Y \cap Z$, then they form a triangle in $G_{X,Y,Z}$. Conversely, if $G_{X,Y,Z}$ contains a triangle, then each edge of the triangle is contained in a single triangle of $G$, by Claim 12.5. This implies that the 3 edges define a triangle in $X \cap Y \cap Z$. \qed

Given sets $X,Y,Z$ as input, Alice, Bob and Charlie execute the protocol for detecting triangles in the network $G_{X,Y,Z}$, with Alice, Bob, Charlie simulating the behavior of the nodes in $A, B, C$ of the network, respectively. Each of the players knows enough information to simulate the behavior of these nodes—for example, Alice knows the neighbors of $A$, since she knows $Y,Z$. Finally, by Theorem 5.12, the total communication of the protocol must be at least $\Omega(|U|)$. 

![Figure 12.3: The graph $G_{X,Y,Z}$](image)
List of Figures

1 The addition, multiplication and division tables of $F_5$. 19
1.1 Any chip arranged on a grid can be divided into two pieces with a small number of crossing wires. 23
1.2 The disjointness function when $n = 8$. Each column corresponds to a set $X \subseteq [8]$, and each row corresponds to a set $Y \subseteq [8]$. The inputs are ordered lexicographically. The $X, Y$ entry is black if and only if $X$ and $Y$ are disjoint. 25
1.3 The vertices that are not neighbors of $v$ cannot be involved in any intersection between $C$ and $I$. 26
1.4 An execution of a protocol. 28
1.5 A combinatorial rectangle. 31
1.6 A partition of the space into 6 rectangles. 32
1.7 A boolean function $g$ represented as a matrix. The black entries are 1’s and the white entries are 0’s. A monochromatic rectangle is highlighted in gray. 32
1.8 Every rooted binary tree must contain a vertex that accounts for at least 1/3rd fraction, but no more than a 2/3rd fraction of all the leaves. 33
1.9 Balancing Protocols 33
1.10 A partition into rectangles that cannot be realized by any protocol. 34
1.11 Either there is a 0-rectangle consistent with $x$ that intersects at most half of the 1-rectangles vertically, or there is a 0-rectangle consistent with $y$ that intersects at most half of the 1-rectangles horizontally. 35
1.12 Rectangles 1 and 2 intersect vertically, while rectangles 1 and 3 intersect horizontally. 35
1.13 A protocol finding the rectangle containing the input from a partition of the space into rectangles. 36
1.14 The equality function does have large monochromatic rectangles. 36
1.15 The inner-production function over $F_2$ when $n = 8$. The inputs are ordered lexicographically. 37
1.16 A $(7, 4)$-rich function. 39
1.17 An input to the lopsided disjointness problem with $n = 12, k = 3, t = 2$. 40
1.18 The greater than function has large monochromatic rectangles of both types. 41
1.19 This function has a 1-cover of size 3. 44

2.1 The disjointness matrix $D_{8,3}$, which represents disjointness when restricted to sets of size at most 3 on a universe of size 8. The sets are ordered lexicographically. Compare this with Figure 1.2. 55
2.2 Protocol for low rank matrices with $2^{O(\sqrt{\log r})}$ leaves. 57
2.3 One can always apply a linear transformation so that the unit ball is the inscribed ellipsoid of maximum volume inside the given convex body. 60
2.4 If some $w_i$ has length bigger than 1, then there must be a vector $v_j$ such that $\langle v_j, w_i \rangle > 1$. This is impossible. 60
2.5 $\arccos(\alpha) \leq \frac{\pi}{2} - \frac{2\pi\alpha}{7}$ for $0 \leq \alpha \leq 1$. 61
2.6 The region where all $z_k$’s must fall to ensure that $(i, j) \in R$, when $M_{i,j} = 0$. 61
2.7 The region where all $z_k$’s must fall to ensure that $(i, j) \in R$, when $M_{i,j} = 1$. 61

3.1 A protocol for equality using shared randomness. 65
3.2 A protocol for equality using private randomness and an error correcting code C. 66
3.3 Public-coin protocol for greater than. 66
3.4 When $X, Y$ are disjoint, typically half of $Y$ will be eliminated by $R_i$. 67
3.5 Public-coin protocol for $k$-disjointness. 67

4.1 The average of two distinct points on a sphere cannot lie on the sphere—they must lie in the interior. 77
4.2 A cylinder intersection. Watch an animation. 79
4.3 Figure 4.2 viewed from above. 79
4.4 The intersection of two cylinders. 79
4.5 A cylinder intersection. Watch an animation. 80
4.6 Figure 4.5 viewed from above. 80
4.7 Figure 4.5 viewed from the left. 80
4.8 Figure 4.5 viewed from the right. 80
4.9 A monochromatic corner. 81
4.10 A rainbow-corner. 81
4.11 A rainbow-corner induced by two smaller rainbow-corners. 82

5.1 A dense graph with no 3-cycles. 87
5.2 The set $A'$ contains many sets that do not intersect each other too much. 97
5.3 There will be a set in $B$ that intersects many of the sets $X_1, \ldots, X_t$. 97
5.4 Four examples showing projections of the set \( \{+1, -1\}^5 \) onto different 2-dimensional vector spaces. The circles are centered at the origin and have radius \( \sqrt{2} \).

6.1 The entropy of a bit with \( p(1) = \epsilon \).
6.2 A finite set in \( \mathbb{R}^3 \) projected onto the 3 planes.
6.3 If the vertices near a fixed vertex do not form a tree, then the graph contains a short cycle.
6.5 The divergence between two distributions on bits \( p, q \).
6.6 Pinsker’s Inequality for two bits. The difference between the divergence and its lower bound is shown.
6.7 The solid line plots the divergence of an \( \epsilon \)-biased bit from a bit with bias 2/3. The dashed line plots \( 2 \ln 2 (\epsilon - 2/3)^2 \).
6.8 An example of an input to pointer chasing, with \( n = 8, k = 5 \).

7.1 A compression algorithm for deterministic protocols.
7.2 The correlated sampling procedure.
7.3 An illustration of the sampling procedure. \((M_4, \rho_4)\) is selected in this case and \( M_A = M_B \). Note that \( \rho_5 < q(M_5) \) but \( \rho_5 > p(M_5) \).
7.5 The sampling procedure of Theorem 7.6. Here \( T \) is 3 and the sampled point is the 3’rd point of \( S_T \).
7.4 Compressing a single round of communication to its internal information.
7.6 Sampling from \( p \) when the sender knows only one distribution. Here Alice chooses \( M_7 \), and it likely takes the players four rounds to compute \( M_7 \).
7.7 Compressing a single round of communication to its internal information.
7.8 Finding the correct path \( m \). In this case the \( m \) is found after 3 mistakes have been fixed.
7.9 Compressing protocols to their internal information.
8.1 A circuit computing the parity \( x_1 \oplus x_2 \oplus x_3 \). This circuit has size 15 and depth 4.
8.2 A schematic description of \( G_X \) and \( H_Y \).
8.3 A formula \( F \) and the tree \( T_i \) that corresponds to the input gates of \( x_i \). Here shaded input gates correspond to \( x_i \).
8.4 A refutation of \( F \). In each step, two clauses are combined to give a new clause that must be true. The final step produces an empty clause, which represents a contradiction.
9.1 A branching program computing \( x_1 \land x_2 \land x_3 \).
9.2 An example of the hard inputs $H, J$ for the matching problem. The graph $G$ has many large induced matchings. Alice gets a random subgraph $H$ of $G$. Bob gets a graph $J$ which matches all the vertices not included in some matching of $G$. The maximum matching includes the edges touching $S$ and all the edges contained in a random induced matching.

9.3 The bipartite graph defined by the branching program can be partitioned into cliques. The $Q_j$'s and $R_j$'s in the claim are not necessarily consecutive blocks, and some of the $Q_j$'s may be of size 1.

10.1 A heap. Here the minimum (3) is deleted from the heap, and a new number (2) is added.

10.2 Maintaining numbers in a binary search tree with $S = \{2, 3, 7, 8, 9, 10, 11, 12, 16\}$.

10.3 An example of a van Emde Boas tree.

10.4 Maintaining a partition of the universe into sets using the union-find data structure. Each cell is associated with an element of the universe, and stores a pointer to the cell corresponding to its parent, as well as the height of the subtree rooted at the element. The result of merging two sets is shown.

10.5 An example of the sets $S_j$ when $k = 2, r = 5$.

10.6 The butterfly graph with $d = 3, B = 2$.

10.7 The edges of $B$ when $k = 3, r = 5, j = 2$.

11.1 A cube in $\mathbb{R}^3$ can be defined by the 6 inequalities $-1 \leq x \leq 1, -1 \leq y \leq 1, -1 \leq z \leq 1$.

11.2 The anatomy of a polytope. The polytope has dimension 2. The six 1-dimensional faces are called facets. The six 0-dimensional faces are called vertices. The intersection between every two facets is a face—it is either a vertex or empty.

11.3 A projection of the cube.

11.4 Given a point $x$ in the polytope that puts weight on a directed cycle or a path that is not the chosen shortest path of the graph, one can always modify the point to obtain a point that only puts weight on the shortest path, and yet does not have a larger value of $L(x)$.

11.5 A polytope with 8 facets that has an extension with 6 facets. Watch an animation.

11.6 An octagon can be built with 3 reflections.

11.7 The permutahedron with $n = 4$, projected into 3-space. Watch an animation.

11.8 A separating polytope for the AND function. Watch an animation.

11.9 A separating polytope for the parity function: $x_1 + x_2 + x_3 \mod 2$. Watch an animation.
11.10 A numbering of the facets and vertices of the $1 \times 1 \times 1$ cube. The slack matrix corresponding to the vertices and the inequalities defining the facets:

11.11 The slack of the pair $S, T$ is the number of edges leaving $S$. In this case, the slack is 4.

11.12 An example of $A, W, F$ when $n = 6$, and $X, Y$ conditioned on $D$.

11.13 Two values of $C$ that intersect in only one element. The cut $X$ is consistent with $c$, and the matching $Y$ is consistent with $c'$.

11.14 If $c_1, c_2, c_3$ are three sets whose union is of size 5, and pairwise intersections are of size 2, there is no other set that can be added to the collection while keeping the pairwise intersection of size 2.

12.1 $G_{X,Y}$ for $n = 4$.

12.2 The graph $G$.

12.3 The graph $G_{X,Y,Z}$. 
Bibliography


Misha Gromov. In a search for a structure, part 1: On entropy.


## Index

Algorithms from polytopes, 229  

Boolean circuits, 165  
Boolean formulas, 171  
Branching programs, 185, 193  

Cauchy-Schwartz inequality, 18, 43, 105, 106, 158  

Chain rule, 113, 123  

Circuits with arbitrary gates, 173  

Coloring (distributed computing), 253  

Combinatorial rectangles, 31, 34, 36, 54  

Connectivity (data structure), 216, 218  

Convexity, 18  

Correlated sampling, 145  

Cutting planes (proof system), 179  

Cylinder intersections, 79  

Deterministic compression, 143  

Deterministic protocols, 23, 28  

Diameter (distributed computing), 255  

Dictionary, 197  

Direct sums, 45, 159  

Discrepancy, 85, 96  

Disjointness, 91, 96, 129  

Distributed computing, 253  

divergence, 120  

Dynamic data structures, 211  

Entropy, 109  

Equality function, 26, 36, 54  

Extension complexity of polytopes, 232, 240, 241  

Finite Fields, 18  

Fooling sets, 41  

Gap-hamming, 99  

General internal compression, 155  

Girth (distributed computing), 256  

Graphs, 15  

Greater Than, 127  

Indexing problem, 123  

Inner-product, 89  

Jensen’s inequality, 18, 87  

Karchmer-Wigderson games, 166  

KL-divergence, 120  

Krapchenko’s method, 42  

Matching problem, 168  

Maximum matching (streaming), 190
Memory size, 185
Moment estimation
(streaming), 188
Monochromatic Rectangles, 36
Monotone boolean circuits, 168
Montone boolean circuits, 170
Mutual information, 120

Nearest neighbor search, 202, 209
Nearly monochromatic rectangles, 72
Non-negative rank, 61

One-round compression, 147
Pinsker’s inequality, 125, 127, 133, 137, 158, 245, 250
Point-chasing, 134
Polynomials, 19
Polytopes, 223, 225
Predecessor search, 199, 207
Prefix sum (data structure), 211
Probabilistic inequalities, 17
Probability, 16
Proof systems, 175
Protocol, 25
Public versus private coins, 72

Ramsey theory, 80
Randomized protocols, 65
Rank, 51, 54
Rectangle covers, 43
Rectangles, 31
Resolution, 176
Richness, 39
Round Elimination, 127
Round elimination, 207

Set intersection (data structure), 205
Shearer inequality, 119
Simulation, 142
Slack matrix, 238
Sort statistics, 198
Span problem (data structure), 206
Static data structures, 205
Streaming algorithms, 188

The pigeonhole principle, 177
Union-find, 201