CSE 525 Randomized Algorithms & Probabilistic Analysis **Winter 2008**

Lecture 1: January 7

Lecturer: James R. Lee *Scribe: Alice Neels and Alexei Czeskis*

1.1 Randomness in Computation

In this course we will deal mostly with random algorithms and random structures. Other topics that fall under "randomness in computation" include complexity (is randomness necessary for efficient computation?), derandomization of algorithms, imperfect sources of randomness, and pseudorandom generators.

1.1.1 Randomized Algorithm Definitions

A randomized algorithm is an algorithm that in addition to taking the usual input also takes a string of independent, uniformly distributed random bits. We may then view the (randomized) algorithm as a traversal of a decision tree, where we take the left branch at each stage if the next random bit is 0 and the right branch if it is 1. If the tree has height k, then each possible outcome has probability 2^{-k}.

The above is a strict model. In practice, we can say things like, "with probability 2/3 do..." because we can simulate any such probability with small error. We can also say "choose a random $x \in [0,1]$". The error of these approximations can be added to the error inherent in our algorithms as long as our algorithm does not depend in very precise ways on the probabilities.

Definition 1.1 (Decision Problem) *A decision problem is a function F such that for all possible inputs x, $F(x) = YES$ or NO.*

We now define two kinds of error:

Definition 1.2 (One-sided Error) *An algorithm A has one-sided error if for every input x*

- *If the answer is YES, then* $\Pr[A(x) = YES] = 1$
- *If the answer is NO, then* $\Pr[A(x) = YES] < 0.99$

By repeating the algorithm, we can drive the probability of failure down to be arbitrarily small, so the actual value of 0.99 is not important. Really we just need $1 - \epsilon$ for some small ϵ.

Definition 1.3 (Two-sided Error) *An algorithm A has two-sided error if for every input x*

- *If the answer is YES, then* $\Pr[A(x) = YES] \geq 2/3$ *(or $1/2 + \epsilon$)*
- *If the answer is NO, then* $\Pr[A(x) = YES] < 1/3$ *(or $1/2 - \epsilon$)*

To improve accuracy, run this several times and then take a majority vote of the answers. The error probability will decrease exponentially. We will prove this later. Again, we can aggregate errors here into the error bounds for our algorithms.

1.2 Example Algorithms

Now, some algorithms:

1.2.1 A Classical Example: Polynomial Identity Testing

Input: Two multivariate polynomials

$$Q(x_1, x_2, ..., x_n)$$

$$R(x_1, x_2, ..., x_n)$$

Output: If $Q \equiv R$ then output YES, otherwise NO.

If we were given both polynomials in expanded form we could just compare coefficients. However, we might get something like

$$Q(x) = \prod_{i=1}^{n-1} (x_i + x_{i+1})$$

which would take exponential time just to expand!

Instead, do something more clever:

1.2.1.1 Schwartz and Zippel(1979)

Assumption: given $a_1, a_2, ..., a_n \in \mathbb{R}^n$, we can efficiently evaluate $Q(a_1, ..., a_n)$ and $R(a_1, ..., a_n)$.

Algorithm: Let $S \subseteq \mathbb{R}$ be some finite set of values. Pick $r_1, ..., r_n \in S$ uniformly,independently at random. If $Q(r_1, ..., r_n) = R(r_1, ..., r_n)$ then output YES else NO.

It is useful to recognize that this is the same as zero-testing the polynomial $P = Q - R$.

1.2.1.2 Analysis

Note that the above algorithm (call it A) has one-sided error. If $P \equiv 0$ then A outputs YES with probability 1.

Claim: If P is not equivalent to 0, then

$$\Pr[P(r_1, ..., r_n) = 0] \leq \frac{d}{|S|}$$

where d is the degree of P.

Proof: By induction on n, the number of variables.

If $n = 1$, then there are at most d distinct roots (by the fundamental theorem of algebra). So

$$\Pr[P(r_1) = 0] = \Pr[r_1 \text{ is a root of } P] \leq \frac{d}{|S|}$$

If $n \geq 1$ then write

$$P(x_1, ..., x_n) = M(x_2, ..., x_n)x_1^k + N(x_1, ..., x_n)$$

where k is the degree of x_1 in P and the degree of x_1 in N is less than k.

First choose $r_2, ..., r_n$ uniformly, independently at random from S. Let $\mathcal{E}$ be the event that $M(r_2, ..., r_n) = 0$. Then

$$\Pr[\mathcal{E}] \leq \frac{d-k}{|S|}$$

by induction, because the degree of M is less than $d - k$.

What is $\Pr[P(r_1, ..., r_n) = 0|\neg\mathcal{E}]$? Define

$$P'(x_1) = P(x_1, r_2, ...r_n) = M(r_2, ..., r_n)x_1^k + N(x_1, r_2, ..., r_n)$$

Then $\Pr[P'(x_1) = 0] \leq \frac{k}{|S|}$ by the first case. Because the degree of x_1 in $N < k$, the two polynomials can never cancel, so

$$\Pr[P(r_1, ..., r_n) = 0|\neg\mathcal{E}] = P'(x_1) \leq \frac{k}{|S|}$$

Then

$$\begin{aligned}
\Pr[P(r_1, ..., r_n) = 0] &= \Pr[P(r_1, ..., r_n) = 0|\mathcal{E}] \Pr[\mathcal{E}] + \Pr[P(r_1, ..., r_n) = 0|\neg\mathcal{E}] \Pr[\neg\mathcal{E}] \\
&\leq \Pr[\mathcal{E}] + \Pr[P(r_1, ..., r_n) = 0|\neg\mathcal{E}] \\
&\leq \frac{d-k}{|S|} + \frac{k}{|S|} = \frac{d}{|S|}
\end{aligned}$$

■

We want to find x such that property(x) hold: choose x at random and check property(x). This has the following requirement:

1. Easy to check
2. Density of x's for which property(x) holds has to be large

If we choose $|S| > 2d$, then

- If $P \equiv 0$ then Pr[output YES] = 1
- If $P \not\equiv 0$ then Pr[output YES] $\leq \frac{d}{|S|} < \frac{1}{2}$

Running Time - Evaluate $P(r_1, r_2, \ldots, r_n)$ [linear time in size of P]

1.2.2 Perfect Matching

Input: Bipartite graph $G = (V_1, V_2, E)$.
Output: Does G have a perfect matching?
Classic solution: use flow (max flow). However, this can't be parallelized.

1.2.2.1 Solution using Randomization

Construct a Tutte matrix for G. Let $A_G(a_{ij})$ be an $n \times n$ matrix.

$$a_{ij} = \begin{cases} x_{ij} & \text{if } (i,j) \in E. \text{ Note: } x_{ij} \text{ is just a variable} \\ 0 & \text{otherwise} \end{cases}$$

Claim:

$$\begin{aligned} det(A_G) &\not\equiv \Leftrightarrow G \text{ has a perfect matching} \\ det(A_G) &= \sum_{\sigma \in S_n} sgn(\sigma) \cdot \prod_{i=1}^{n} a_{ij} \end{aligned}$$

In other words:

$$\begin{aligned} G \text{ no P.M.} &\Rightarrow det(A_G) \equiv 0 \\ G \text{ has P.M.} &\Rightarrow det(A_G) \not\equiv 0 \end{aligned}$$

We can use *SZ* algorithm to test whether $det(A_G) \equiv 0$. Furthermore, we can use the above to parallelize computation and find a perfect matching in $O(\log^2 n)$ time (Mulmuly-Vazirani[2]).