Outline: Sequence Alignment

- What
- Why (applications)
  - Comparative genomics
  - DNA sequencing
- A simple algorithm
- Complexity analysis
- A better algorithm:
  - “Dynamic programming”
Sequence Alignment: What

- **Definition**
  - An arrangement of two or several biological sequences (e.g. protein or DNA sequences) highlighting their similarity
  - The sequences are padded with gaps (usually denoted by dashes) so that columns contain identical or similar characters from the sequences involved

- Example – pairwise alignment

```
T A C T A A G
T C C A A T
```
Sequence Alignment: Why

- The most basic sequence analysis task
  - First aligning the sequences (or parts of them) and
  - Then deciding whether that alignment is more likely to have occurred because the sequences are related, or just by chance

- Similar sequences often have similar origin or function

- New sequence always compared to existing sequences (e.g. using BLAST)

Sequence Alignment

- Example: gene HBB
  - Product: hemoglobin
  - Sickle-cell anaemia causing gene
  - Protein sequence (146 aa)

```
MVHTPEEKS AVTALKGVKN VDEVGGEALG RLLVYVPWTQ RFFESFGDLS TPDAVMGNPK VKAHKVIQV AFSIDLAHLD NLKXTFATLS ELHCDKLHVD PENFRLLGKV LIQVLASHGFG KEPTPPVQAA YQKVVAVVAN ALAKYH
```

- BLAST (Basic Local Alignment Search Tool)
  - The most popular alignment tool
  - Try it! Pick any protein, e.g. hemoglobin, insulin, exportin,… BLAST to find distant relatives.
Sequence Alignment: Why

- The most basic sequence analysis task
  - First aligning the sequences (or parts of them) and
  - Then deciding whether that alignment is more likely to have occurred because the sequences are related, or just by chance

- Similar sequences often have similar origin or function

- New sequence always compared to existing sequences (e.g. using BLAST)

- Alignment algorithm is also used when sequencing DNA (reading DNA sequence)

How Do We Read DNA?

- We replicate it
How Do We Read DNA?

- We replicate it
- We shred it

Reading Short DNA

- Use replication machinery with colored bases
- Take pictures of massively parallel reaction
- 10 million reads of 30 per day & $1000
Example

The genome is:
TTATGGTCGGTGAGTGTGACTGGTGTTGTCTAA

The reads are:
GGTCGGTGAG
TGAGTGTGAC
TGGTGTTGTC
TGACTGGTTT
AATGGTCGGT
GAGTGTGACT
AAAAAAAAAAA

Example

The genome is:
TTATGGTCGGTGAGTGTGACTGGTGTTGTCTAA

The reads are:
GGTCGGTGAG
TGAGTGTGAC
TGGTGTTGTC
TGACTGGTTT
AATGGTCGGT
GAGTGTGACT
AAAAAAAAAAA
Example

The genome is:
TTATGGTCGGTGAGTGTGACTGGTGTTGTCTAA
     TGAGTGTGAC

The reads are:
GGTCGGTGAG
TGAGTGTGAC
TGGTGGTGC
TGACGTGGTTT
AATGGTCCGGT
GAGTGTGACT
AAAAAAAAAA

Example

The genome is:
TTATGGTCGGTGAGTGTGACTGGTGTTGTCTAA
     TGGTGGTGC

The reads are:
GGTCGGTGAG
TGAGTGTGAC
TGGTGGTGC
TGACGTGGTTT
AATGGTCCGGT
GAGTGTGACT
AAAAAAAAAA
Example

The genome is:

TTATGGT CGGTGAGTGACTGGT GTGTTGTCTAA

\[\text{TGACTGGTTT}\]

The reads are:

GGTCGGTGAG
TGAGTGAC
TGGTTGTGC
TGACTGGTTT
AATGGTCGCT
GAGTGTGACT

AAAAAAAAAAA

Example

The genome is:

TTATGGTCG GTGAGTGACTGGT GTGTTGTCTAA

\[\text{AATGGTCGCT}\]

The reads are:

GGTCG GTGAG
TGAGTGAC
TGGTTGTGC
TGACTGGTTT
AATGGTCGCT
GAGTGTGACT

AAAAAAAAAAA
Example

**The genome is:**
TTATGGTCGGTGAATGTGACTGGTGTTGTTGCTAA
GGTCGCTGAG
TGAGTGTGAC
TGGTTGTGTC
TGACTGTTTT
AATGGTCGTT
GAGTGTGACT

**Assembled genome sequence is:**
AATGGTCGTTGAGTGTGACTGGTTGTTGCT

Computational Problem

- **Goal:** Align reads to genome

- **Input:**
  Many reads: $l$-long strings $S_1, \ldots, S_m$
  Approximate reference genome: string $R$

- **Output:**
  $x_1, \ldots, x_m$ along $R$ where reads match, resp.

- **Complications:**
  - Errors
  - Differences
Sequence Alignment

- What
- Why (applications)
  - Comparative genomics
  - DNA sequencing
- A simple algorithm
- Complexity analysis
- A better algorithm:
  - “Dynamic programming”

Sequence Alignment: Key Issues

- What sorts of alignment should be considered
- The scoring system used to rank alignments
- The algorithm used to find optimal (or good) scoring alignments
- The statistical methods used to evaluate the significance of an alignment score
Terminology (CS, not necessarily Bio)

- **String**: ordered list of letters
  - TATAAG
- **Prefix**: consecutive letters from front
  - empty, T, TA, TAT, ...
- **Suffix**: ... from end
  - empty, G, AG, AAG, ...
- **Substring**: ... from ends or middle
  - empty, TAT, AA, ...
- **Subsequence**: ordered, nonconsecutive
  - TT, AAA, TAG, ...

Sequence Alignment

\[
\begin{align*}
S &= a \ c \ b \ c \ d \ b \\
T &= c \ a \ d \ b \ d
\end{align*}
\]

\[
\begin{align*}
S' &= a \ c \ - \ - \ b \ c \ d \ b \\
T' &= c \ a \ d \ b \ - \ - \ d \ -
\end{align*}
\]

**Definition:** An *alignment* of strings $S$, $T$ is a pair of strings $S'$, $T'$ (with spaces) s.t.

1. $|S'| = |T'|$, and ($|S| = \text{"length of } S\text{"}$)
2. removing all spaces leaves $S$, $T$
Alignment Scoring

\[
\begin{align*}
S &= a \ c \ b \ c \ d \ b \\
T &= c \ a \ d \ b \ d
\end{align*}
\]

\[
\begin{align*}
S' &= a \ c \ - \ - \ b \ c \ d \ b \\
T' &= - \ c \ a \ d \ b \ - \ d \ -
\end{align*}
\]

Mismatch = -1
Match = 2

\[
\text{Value} = 3 \times 2 + 5 \times (-1) = +1
\]

- The score of aligning (characters or spaces) \(x\) & \(y\) is \(\sigma(x,y)\).
- Value of an alignment \(\sum_{i=1}^{\mid S'\mid} \sigma(S'[i],T'[i])\)
- An optimal alignment: one of max value

Optimal Alignment: A Simple Algorithm

for all subseqs \(A\) of \(S\), \(B\) of \(T\) s.t. \(\mid A\mid = \mid B\mid\) do
align \(A[i]\) with \(B[i]\), \(1 \leq i \leq \mid A\mid\)
align all other chars to spaces
compute its value
retain the max
end
output the retained alignment

Example

\[
\begin{align*}
S = abcd & \rightarrow A = cd \\
T = wxyz & \rightarrow B = xz \\
-abc-d & \rightarrow -abc-d \\
w--xyz & \rightarrow -w-xyz
\end{align*}
\]
Outline: Sequence Alignment

- What
- Why (applications)
  - Comparative genomics
  - DNA sequencing
- A simple algorithm
- Complexity analysis
- A better algorithm:
  - “Dynamic programming”

Complexity Analysis

- Assume $|S| = |T| = n$
- Cost of evaluating one alignment: $\geq n$

- How many alignments are there: $\binom{2n}{n}$
  - pick $n$ chars of $S$, $T$ together
  - say $k$ of them are in $S$
  - match these $k$ to the $k$ unpicked chars of $T$

- Total time: $\geq n \binom{2n}{n} > 2^{2n}$, for $n > 3$
- E.g., for $n = 20$, time is $> 2^{40}$ operations
Outline: Sequence Alignment

- What
- Why (applications)
  - Comparative genomics
  - DNA sequencing
- A simple algorithm
- Complexity analysis
- A better algorithm:
  - “Dynamic programming”
Alignment Scoring

<table>
<thead>
<tr>
<th>S</th>
<th>a</th>
<th>c</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>c</td>
<td>a</td>
<td>d</td>
<td>b</td>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S'</th>
<th>a</th>
<th>c</th>
<th>-</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>T'</td>
<td>-</td>
<td>c</td>
<td>a</td>
<td>d</td>
<td>b</td>
<td>-</td>
<td>d</td>
</tr>
</tbody>
</table>

Value = $3 \times 2 + 5 \times (-1) = +1$

- The score of aligning (characters or spaces) $x$ & $y$ is $\sigma(x,y)$: e.g. $\sigma(a, -) = -1$, $\sigma(c, c) = 2$.
- Value of an alignment $\sum_{i=1}^{||S||} \sigma(S'[i], T'[i])$
- An optimal alignment: one of max value
- A simple algorithm: complexity $> 2^{2^n}$

Mismatch = -1
Match = 2

Needleman-Wunsch Algorithm

- Align by “Dynamic programming”
- Key idea: Build up an optimal alignment using previous solutions for optimal alignments of smaller subsequences.

- Optimal alignment between $S$ & $T$ ends in 1 of 3 ways:
  - last chars of $S$ & $T$ aligned with each other
  - last char of $S$ aligned with space in $T$
  - last char of $T$ aligned with space in $S$
  (never align space with space; $\sigma(-, -) < 0$)
- In each case, the rest of $S$ & $T$ should be optimally aligned to each other:

  Opt align of $S_1...S_{n-1}$ & $T_1...T_{m-1}$

  Opt align of $S_1...S_n$ & $T_1...T_m$

  Opt align of $S_1...S_n$ & $T_1...T_{m-1}$
Optimal Alignment in O(n²) via “Dynamic Programming”

- Input: S, T, |S| = n, |T| = m
- Output: value of optimal alignment
- Easier to solve a “harder” problem:
  \[ V(i,j) = \text{value of optimal alignment of } S[1], \ldots, S[i] \text{ with } T[1], \ldots, T[j] \]
  for all \( 0 \leq i \leq n, 0 \leq j \leq m \).

General Case

Optimal align of \( S[1], \ldots, S[i] \) vs \( T[1], \ldots, T[j] \):

\[
\begin{align*}
V(i,j) &= \max \left\{ V(i-1,j-1) + \sigma(S[i],T[j]), \right. \\
&\quad V(i-1,j) + \sigma(S[i], - ), \right. \\
&\quad \left. \left. V(i,j-1) + \sigma(-, T[j]) \right\} \\
&\text{for all } 1 \leq i \leq n, 1 \leq j \leq m.
\end{align*}
\]
Calculating One Entry

\[ V(i,j) = \max \left\{ V(i-1,j-1) + \sigma(S[i], T[j]) \right\} \]

Base Cases

- \( V(i,0) \): first \( i \) chars of \( S \) all match spaces
  \[ V(i,0) = \sum_{k=1}^{i} \sigma(S[k], -) \]

- \( V(0,j) \): first \( j \) chars of \( T \) all match spaces
  \[ V(0, j) = \sum_{k=1}^{j} \sigma(-, T[k]) \]
### Example

- $V(i,j) = \text{value of optimal alignment of } S[1], ..., S[i] \text{ with } T[1], ..., T[j]$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td></td>
<td>c</td>
<td>a</td>
<td>d</td>
<td>b</td>
<td>d</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-2</td>
<td>-3</td>
<td>-4</td>
<td>-5</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td>-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>b</td>
<td>-3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>c</td>
<td>-4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>d</td>
<td>-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>b</td>
<td>-6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Mismatch = -1**

**Match = 2**

### Example

- $V(i,j) = \text{value of optimal alignment of } S[1], ..., S[i] \text{ with } T[1], ..., T[j]$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td></td>
<td>c</td>
<td>a</td>
<td>d</td>
<td>b</td>
<td>d</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-2</td>
<td>-3</td>
<td>-4</td>
<td>-5</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td>-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>b</td>
<td>-3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>c</td>
<td>-4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>d</td>
<td>-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>b</td>
<td>-6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Mismatch = -1**

**Match = 2**

**Score(-,a) = -1**

**Score(c,-) = -1**
Example

Mismatch = -1
Match =  2

\[
\begin{array}{cccccc}
  & 0 & 1 & 2 & 3 & 4 & 5 \\
 0 & \_ & 0 & -1 & -2 & -3 & -4 & -5 \\
 1 & a & -1 \\
 2 & c & -2 \\
 3 & b & -3 \\
 4 & c & -4 \\
 5 & d & -5 \\
 6 & b & -6 \\
\end{array}
\]

\[
\begin{array}{cccccc}
  & 0 & 1 & 2 & 3 & 4 & 5 \\
 0 & \_ & 0 & -1 & -2 & -3 & -4 & -5 \\
 1 & a & -1 & 1 & 1 & 1 & 1 \\
 2 & c & -2 & \_ & \_ & \_ & \_ \\
 3 & b & -3 & \_ & \_ & \_ & \_ \\
 4 & c & -4 & \_ & \_ & \_ & \_ \\
 5 & d & -5 & \_ & \_ & \_ & \_ \\
 6 & b & -6 & \_ & \_ & \_ & \_ \\
\end{array}
\]

Score(\_,c) = -1
### Example

Mismatch = -1  
Match = 2

<table>
<thead>
<tr>
<th>j</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td></td>
<td>c</td>
<td>a</td>
<td>d</td>
<td>b</td>
<td>d</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-2</td>
<td>-3</td>
<td>-4</td>
<td>-5</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td>-2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>b</td>
<td>-3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>c</td>
<td>-4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>d</td>
<td>-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \Rightarrow T \]

### Example

Mismatch = -1  
Match = 2

<table>
<thead>
<tr>
<th>j</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td></td>
<td>c</td>
<td>a</td>
<td>d</td>
<td>b</td>
<td>d</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-2</td>
<td>-3</td>
<td>-4</td>
<td>-5</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td>-2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>b</td>
<td>-3</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>c</td>
<td>-4</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>d</td>
<td>-5</td>
<td>-2</td>
<td>-2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>b</td>
<td>-6</td>
<td>-3</td>
<td>-3</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

\[ \Rightarrow T \]

Time = \( O(mn) \)
### Finding Alignments: Trace Back

Arrows = (ties for) max in \( V(i,j) \); 3 LR-to-UL paths = 3 optimal alignments

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-2</td>
<td>-3</td>
<td>-4</td>
<td>-5</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td>-2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>b</td>
<td>-3</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>c</td>
<td>-4</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>d</td>
<td>-5</td>
<td>-2</td>
<td>-2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>b</td>
<td>-6</td>
<td>-3</td>
<td>-3</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

### Complexity Notes

- **Time** = \( \Theta(mn) \), (value and alignment)
- **Space** = \( \Theta(mn) \)
- Easy to get value in Time = \( \Theta(mn) \) and Space = \( \Theta(\min(m,n)) \)
- Possible to get value and alignment in Time = \( \Theta(mn) \) and Space = \( \Theta(\min(m,n)) \) but tricky.
Significance of Alignments

- Is “42” a good score?
  - Compared to what?

- Usual approach: compared to a specific “null model”, such as “random sequences”

Overall Alignment Significance, II

Empirical (via randomization)

- Generate N random sequences (say N = 10^3 - 10^6)
- Align x to each & score
- If k of them have better score than alignment of x to y, then the (empirical) probability of a chance alignment as good as observed x:y alignment is (k+1)/(N+1)
  - e.g., if 0 of 99 are better, you can say “estimated p < .01”

- How to generate “random” sequences?
  - Scores are often sensitive to sequence composition
  - So uniform 1/20 or 1/4 is a bad idea
  - Even background p_i can be dangerous
  - Better idea: permute y N times
Generating Random Permutations

```c
for (i = n-1; i > 0; i--){
j = random(0..i);
    swap X[i] <-> X[j];
}
```

All $n!$ permutations of the original data equally likely: Why? A specific element will be last with prob $1/n$; given that, a specific other element will be next-to-last with prob $1/(n-1)$, ...; overall: $1/(n!)$