Pairwise Sequence alignment
Basic Algorithms
Review Previous Lessons
Dynamic Programming

- Dynamic Programming is an algorithm design technique, often used for optimization problems: often minimizing or maximizing.
- Like divide and conquer, DP solves problems by combining solutions to subproblems.
- Unlike divide and conquer, subproblems are not independent.
  - Subproblems may share subsubproblems,
  - However, solution to one subproblem may not affect the solutions to other subproblems of the same problem. (More on this later.)
- DP reduces computation by
  - Solving subproblems in a bottom-up fashion.
  - Storing solution to a subproblem the first time it is solved.
  - Looking up the solution when subproblem is encountered again.
- Key: determine structure of optimal solutions
Steps in Dynamic Programming

2. Define value of optimal solution recursively.
3. Compute optimal solution values either top-down with caching (memoization) or bottom-up in a table.
4. Construct an optimal solution from computed values.
The Central Dogma of Molecular Biology

DNA sequence
AATTCATGAAAATCGTATACTGGTCTGGTACCGG
CAACACTGAGAAAATGGCAGAGCTCATCGCTAAA
GGTATCATCGAATCTGGTAAAGACGTCAACACCA
TCAACGTGTCTGACGTTAACATCGATGAACTGCT
GAACGAAGATATCCTGATCCTGGGTTGCTCTGCC
ATGGGCGATGAAGTTCTCGAGGAAAGCGAATTTG

> Protein sequence
MKIVYWSGTGNTEKMAELIAKGIIES
GKDVTINVSDVNDLNNEDILILGC
SAMGDEVLEESEFEPFIEEISTKISG
KKVALFGSYGWGDGKWMRDFEER
MNGYGCVVTPLIVQNEPDEAEQD
CIEFGKKIANI
Cystic Fibrosis and the CFTR Protein

- CFTR (Cystic Fibrosis Transmembrane conductance Regulator) protein is acting in the cell membrane of epithelial cells that secrete mucus.
- These cells line the airways of the nose, lungs, the stomach wall, etc.
Cystic Fibrosis and CFTR Gene:

Chromosome 7

Sequences of nucleotides in CFTR gene:
- ATC
- ATT

Amino acid sequence of CFTR protein:
- Isoleucine 506
- Isoleucine 507
- Phenylalanine 508
- Glycine 509
- Valine 510

Deleted in many patients with cystic fibrosis
Cystic Fibrosis and the CFTR Protein

• **CFTR (Cystic Fibrosis Transmembrane conductance Regulator)** protein is acting in the cell membrane of epithelial cells that secrete mucus.
  • These cells line the airways of the nose, lungs, the stomach wall, etc.
ATP binding proteins are present on cell membrane and act as transport channel.

In 1989 biologists found similarity between the cystic fibrosis gene and ATP binding proteins.

A plausible function for cystic fibrosis gene, given the fact that CF involves sweat secretion with abnormally high sodium level.
Identification of the Cystic Fibrosis Gene: Cloning and Characterization of Complementary DNA

John R. Riordan; Johanna M. Rommens; Bat-sheva Kerem; Noa Alon; Richard Rozmahel; Zbyszko Grzelczak; Julian Zielenski; Si Lok; Natasa Plavsic; Jia-Ling Chou; Mitchell L. Drumm; Michael C. Iannuzzi; Francis S. Collins; Lap-Chee Tsui


<table>
<thead>
<tr>
<th>CFTR 122-136</th>
<th>Y L G I G L C L L F I V R T L</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLNP 198-212</td>
<td>Y L I I T L V L S F I L R R L</td>
</tr>
<tr>
<td>CFTR 307-319</td>
<td>S S A F F F S G F F V V F</td>
</tr>
<tr>
<td>COX 89-101</td>
<td>S E V F F F A G F F W A F</td>
</tr>
<tr>
<td>CFTR 701-713</td>
<td>I L N P I N S I R K F S I</td>
</tr>
<tr>
<td>NaCh 111-123</td>
<td>I L T P F N P I R K L A I</td>
</tr>
<tr>
<td>CFTR 1425-1442</td>
<td>D S I Q K L N E R S L F R Q A I S</td>
</tr>
<tr>
<td>raf 578-595</td>
<td>D S I K K L R D E R P L F P Q I L S</td>
</tr>
</tbody>
</table>
Outline

• Manhattan Tourist Problem
• Longest Paths in Graphs
• Sequence Alignment
• Longest Common Subsequence Problem
We seek the following similarities between sequences:

• Find similar proteins
  – Allows to predict function & structure

• Locate similar subsequences in DNA
  – Allows to identify (e.g) regulatory elements

• Locate DNA sequences that might overlap
  – Helps in sequence assembly
Sequence Modifications

• Three types of changes
  – Substitution (point mutation)
  – Insertion
  – Deletion

  \[
  \text{Indel (replication slippage)}
  \]
Choosing Alignments

There are many possible alignments

For example, compare:

\[-\text{GCGC} - \text{ATGGATTGAGCGA}\]
\[\text{TGCGCCATTTGAT} - \text{GACC} - \text{A}\]

to

\[-----\text{GCGCATGGATTGAGCGA}\]
\[\text{TGCGCC}-----\text{ATTGATGACCA}-----\]

Which one is better?
Another example

Given two sequences:

X: TGCATAT
Y: ATCCGAT

Question:

How can X be transformed into Y?

Or,

How did Y evolve from X?
One possible transformation

**Alignment:**

```
- TGC-ATAT
ATCCGAT--
```

5 operations

- delete T
- delete A
- insert A
- G → C
- insert G
Another possible transformation

Another possible transformation

Which one is better?
In order to align two sequences we need a quantitative model to evaluate similarity between sequences.

How do we quantitate sequence similarity?
Scoring Similarity

• Assume independent mutation model
  – Each site considered separately

• Score at each site
  – Positive if the same
  – Negative if different

• Sum to make final score
  – Can be positive or negative
  – Significance depends on sequence length
Pairwise Alignment - Identity

Human Hemoglobin (HH) vs Sperm Whale Myoglobin (SWM):

(HH) VLSPADKTNVKAAWGGKVEAHAGYEG

(SWM) VLSGEGWQLVLHVWAKVEADVAGHG

• Percent Identity: 36.000 (% only)
D and E are similar:

1. structure is similar.
2. both are acidic and hydrophilic
3. On the DNA level, their codons differ by only one letter

Percent Similarity: 40.000 (| and .)
Percent Identity: 36.000 (| only)
Pairwise Alignment – Gap insertion

(HH) VLSPADKTNVKAAWGKVGAH–AGYEG

(SWM) VLSEGEWQLVLHVWAKVEADVAGH–G

• Gaps: 2
• Percent Similarity: 54.167
• Percent Identity: 45.833 (12/26)
Pairwise Alignment - Scoring

• The final score of the alignment is the sum of the positive scores and penalty scores:

  + Number of Identities
  + Number of Similarities
  - Number of gap insertions

Alignment score
Pairwise Alignment - Scoring

(HH) VLSPADKTNVKAAWGKVGAGH−AGYEG
| | | . | | | | | | | | | | | (SWM) VLSEGEWQLVLHVWAKVEADVAGH−G

Final score:

(V,V) + (L,L) + (S,S) + (D,E) + ...
- (penalty for gap insertion)*(number of gaps)

We are interested in both the score and the alignment trace.
Optimum Alignment

The score of an alignment is a measure of its quality

**Optimum alignment problem:** Given a pair of sequences $X$ and $Y$, find an alignment with maximum score

The similarity between $X$ and $Y$, denoted $\text{sim}(X, Y)$, is the maximum score of an alignment of $X$ and $Y$
Aligning DNA Sequences

V = ATCTGATG  \( n = 8 \)
W = TGCATAC  \( m = 7 \)

4 matches
1 mismatches
3 deletions
2 insertions
Computing Optimal Score

- How can we compute the optimal score?
  - If $|s| = n$ and $|t| = m$, the number $A(m,n)$ of possible “legal” alignments is large!

  (**How many possible alignments?)

  (**Does the problem comply with the “Optimal Substructure Property?”)
Computing Optimal Score

- How can we compute the optimal score?
  - If $|s| = n$ and $|t| = m$, the number $A(m,n)$ of possible “legal” alignments is large!

  (**How many possible alignments?)

  (**Does the problem comply with the “Optimal Substructure Property?”)

- we perform dynamic programming to compute the optimal score efficiently.
Manhattan Tourist Problem (MTP)

Imagine seeking a path (from source to sink) to travel (only eastward and southward) with the most number of attractions (*) in the Manhattan grid.
Manhattan Tourist Problem (MTP)

Imagine seeking a path (from source to sink) to travel (only eastward and southward) with the most number of attractions (*) in the Manhattan grid.
Manhattan Tourist Problem (MTP)

Imagine seeking a path (from source to sink) to travel (only eastward and southward) with the most number of attractions (*) in the Manhattan grid.
Manhattan Tourist Problem: Formulation

**Goal:** Find the longest path in a weighted grid.

**Input:** A weighted grid $G$ with two distinct vertices, one labeled “source” and the other labeled “sink.”

**Output:** A “heaviest” path in $G$ from “source” to “sink.”
MTP with weighted edges

source

j coordinate

i coordinate

sink
MTP: Greedy Algorithm Is Not Optimal

promising start, but leads to bad choices!
MTP: Dynamic Programming

- Calculate optimal path score for each vertex in the graph
- Each vertex’s score is the maximum of the prior vertices score plus the weight of the respective edge in between
MTP: Dynamic Programming
(cont’d)

![Diagram showing dynamic programming concept]

Source: S

1. \( S_{0,2} = 3 \)
2. \( S_{1,1} = 4 \)
3. \( S_{2,0} = 8 \)
MTP: Dynamic Programming (cont’d)

source

\[ S_{3,0} = 8 \]
\[ S_{1,2} = 13 \]
\[ S_{2,1} = 9 \]
MTP: Dynamic Programming (cont’d)

source

S_{1,3} = 8

S_{2,2} = 12

S_{3,1} = 9

greedy alg. fails!
MTP: Dynamic Programming

(cont’d)

\[ S_{2,3} = 15 \]

\[ S_{3,2} = 9 \]
MTP: Dynamic Programming

(cont’d)

(source)

Don't!
MTP: Recurrence

Computing the score for a point \((i,j)\) by the recurrence relation:

\[
S_{i,j} = \max \begin{cases} 
S_{i-1,j} + \text{weight of the edge between } (i-1, j) \text{ and } (i, j) \\
S_{i,j-1} + \text{weight of the edge between } (i, j-1) \text{ and } (i, j)
\end{cases}
\]

The running time ???

\((n = \# \text{ of rows}, \, m = \# \text{ of columns})\)
MTP: Recurrence

Computing the score for a point \((i,j)\) by the recurrence relation:

\[
\begin{align*}
  s_{i,j} &= \max \left\{ s_{i-1,j} + \text{weight of the edge between } (i-1, j) \text{ and } (i, j), \\
  s_{i,j-1} + \text{weight of the edge between } (i, j-1) \text{ and } (i, j) \right\}
\end{align*}
\]

The running time is \(n \times m\) for a \(n\) by \(m\) grid

\((n = \# \text{ of rows}, \ m = \# \text{ of columns})\)
Adding Diagonal Edges to the Grid

What about diagonals?

• The score at point B is given by:

\[ s_B = \max \left\{ s_{A_1} + \text{weight of the edge } (A_1, B), \\
               s_{A_2} + \text{weight of the edge } (A_2, B), \\
               s_{A_3} + \text{weight of the edge } (A_3, B) \right\} \]
More generally, computing the score for point $x$ is given by the recurrence relation:

$$s_x = \max \left\{ s_y + \text{weight of vertex } (y, x) \mid y \in \text{Predecessors}(x) \right\}$$

- Predecessors ($x$) – set of vertices that have edges leading to $x$
Traveling in the Grid

• The only hitch is that one must decide on the order in which visit the vertices.

• By the time the vertex $x$ is analyzed, the values $s_y$ for all its predecessors $y$ should be computed – otherwise we are in trouble.

• We need to traverse the vertices in some order.

• Try to find such order for a directed acyclic graph.
Traversing the Manhattan Grid

- 3 different strategies:
  - a) Column by column
  - b) Row by row
  - c) Along diagonals
Next...

• How do you compute the optimal alignment between two sequences?
• How is this problem related to optimal paths in a grid?
• Let’s start with the simplest sequence alignment variant, the Longest Common Subsequence...
• How do you compute the optimal alignment between two sequences?
• How is this problem related to optimal paths in a grid?
• Let’s start with the simplest sequence alignment variant, the Longest Common Subsequence...
Longest Common Subsequence

- **Problem:** Given two sequences, \( X = \langle x_1, \ldots, x_m \rangle \) and \( Y = \langle y_1, \ldots, y_n \rangle \), find a common subsequence whose length is maximum.

<table>
<thead>
<tr>
<th>X:</th>
<th>springtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y:</td>
<td>printing</td>
</tr>
</tbody>
</table>

\[ \text{LCS}(X,Y) : \text{printing} \]
Longest Common Subsequence (LCS) – Alignment without Mismatches

• Given two sequences

  \[ v = v_1 \ v_2 \ldots v_m \text{ and } w = w_1 \ w_2 \ldots w_n \]

• The LCS of \( v \) and \( w \) is a sequence of positions in

  \[ v: \ l < i_1 < i_2 < \ldots < i_t < m \]

  and a sequence of positions in

  \[ w: \ l < j_1 < j_2 < \ldots < j_t < n \]

such that \( i_t \)-th letter of \( v \) equals to \( j_t \)-letter of \( w \) and \( t \) is maximal
LCS: Example

\[ \text{elements of } v \quad \text{elements of } w \]

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 2 & 3 & 3 & 4 & 5 \\
A & T & = & C & = & T & G & A & T & C \\
& = & T & G & C & A & T & = & A & = & C \\
\end{array}
\]

\[ i \text{ coords:} \quad 0 \quad 1 \quad 2 \quad 2 \quad 3 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \]

\[ j \text{ coords:} \quad 0 \quad 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 5 \quad 6 \quad 6 \quad 7 \]

\[(0,0) \rightarrow (1,0) \rightarrow (2,1) \rightarrow (2,2) \rightarrow (3,3) \rightarrow (3,4) \rightarrow (4,5) \rightarrow (5,5) \rightarrow (6,6) \rightarrow (7,6) \rightarrow (8,7)\]

Matches shown in red

positions in \(v\): 2 < 3 < 4 < 6 < 8
positions in \(w\): 1 < 3 < 5 < 6 < 7

Every common subsequence is a path in 2-D grid
LCS: Dynamic Programming

**Input:** A weighted graph \( G \) with two distinct vertices, one labeled “source” one labeled “sink”

**Output:** A longest path in \( G \) from “source” to “sink”

Solve using an LCS edit graph with diagonals replaced with +1 edges
LCS Problem as Manhattan Tourist Problem
Edit Graph for LCS Problem

Every path is a common subsequence.

Every diagonal edge adds an extra element to common subsequence

**LCS Problem:** Find a path with maximum number of diagonal edges
Computing LCS

Let \( v_i \) = prefix of \( v \) of length \( i \): \( v_1 \ldots v_i \)
and \( w_j \) = prefix of \( w \) of length \( j \): \( w_1 \ldots w_j \)

The length of LCS\((v_i, w_j)\) is computed by:

\[
s_{i,j} = \max \begin{cases} 
s_{i-1,j} \\
s_{i,j-1} \\
 s_{i-1,j-1} + 1 \text{ if } v_i = w_j 
\end{cases}
\]
Every Path in the Grid Corresponds to an Alignment

$V = \text{AT} - \text{GT}$

$W = \text{ATCG} -$
Comparison methods

• **Global alignment** – Finds the best alignment across the whole two sequences.

• **Local alignment** – Finds regions of similarity in parts of the sequences.
Global Alignment

• Algorithm of Needleman and Wunsch (1970)
• Finds the alignment of two complete sequences:

\[
\begin{align*}
\text{ADLGAVFALCDRYFQ} \\
\text{ADLGRTQN-CDRYYQ}
\end{align*}
\]

• Some global alignment programs “trim ends”
Local Alignment

• Algorithm of Smith and Waterman (1981).

• Makes an optimal alignment of the best segment of similarity between two sequences.

  ADLG        CDRYFQ
  ||||||       |||| |
  ADLG        CDRYYQ

• Can return a number of highly aligned segments.
Global Alignment: Algorithm

\[ S_{1..i} = \text{Prefix of length } i \text{ of } S \]
\[ T_{1..j} = \text{Prefix of length } j \text{ of } T \]

\[ C(i, j) = \text{Cost of optimum alignment of } S_{1..i} \text{ and } T_{1..j} \]

\[ w(a, b) = \begin{cases} 
+ \alpha & \text{if } a = b \\
- \beta & \text{if } a \neq b 
\end{cases} \]
**Theorem.** \( C(i,j) \) satisfies the following relationships:

**Initial conditions:**

\[
C(i,0) = -i \cdot \gamma \quad \quad \quad \quad \quad \quad C(0,j) = -j \cdot \gamma
\]

**Recurrence relation:** For \( 1 \leq i \leq n, \, 1 \leq j \leq m \):

\[
C(i, j) = \max \left\{ C(i-1, j-1) + w(S_i, T_j), \quad C(i-1, j) - \gamma, \quad C(i, j-1) - \gamma \right\}
\]
Example

Case 1: Line up $S_i$ with $T_j$

<table>
<thead>
<tr>
<th></th>
<th>i-1</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>S:</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>T:</td>
<td>C</td>
<td>-</td>
</tr>
</tbody>
</table>

Case 2: Line up $T_j$ with space

<table>
<thead>
<tr>
<th></th>
<th>i-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>S:</td>
<td>C</td>
</tr>
<tr>
<td>T:</td>
<td>C</td>
</tr>
</tbody>
</table>

Case 3: Line up $S_i$ with space

<table>
<thead>
<tr>
<th></th>
<th>i-1</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>S:</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>T:</td>
<td>C</td>
<td>-</td>
</tr>
</tbody>
</table>
Justification: Optimal Substructure Property Followed

\[ C(i-1,j-1) + w(S_i, T_j) \]

\[ C(i-1,j) - \gamma \]

\[ C(i,j-1) - \gamma \]
Computation Procedure

\[
C(i, j) = \max \left\{ \begin{array}{l}
C(i-1, j-1) + w(S_i, T_j), \\
C(i-1, j) - \gamma, \\
C(i, j-1) - \gamma
\end{array} \right. 
\]
<table>
<thead>
<tr>
<th>λ</th>
<th>C</th>
<th>T</th>
<th>C</th>
<th>G</th>
<th>C</th>
<th>A</th>
<th>G</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>λ</td>
<td>0</td>
<td>-5</td>
<td>-10</td>
<td>-15</td>
<td>-20</td>
<td>-25</td>
<td>-30</td>
<td>-35</td>
</tr>
<tr>
<td>C</td>
<td>-5</td>
<td>10</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>-10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>-15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>-20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>-25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>-30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>-35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

+10 for match, -2 for mismatch, -5 for space
Traceback can yield both optimum alignments