CSE 427 Comp Bio

Sequence Alignment
Sequence Alignment

What

Why

A Dynamic Programming Algorithm
Sequence Alignment

Goal: position characters in two strings to “best” line up identical/similar ones with one another

We can do this via Dynamic Programming
What is an alignment?

Compare two strings to see how “similar” they are
E.g., maximize the # of identical chars that line up

ATGTTAT vs
ATCGTAC

<table>
<thead>
<tr>
<th>A</th>
<th>T</th>
<th>-</th>
<th>G</th>
<th>T</th>
<th>T</th>
<th>A</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>T</td>
<td>C</td>
<td>G</td>
<td>T</td>
<td>-</td>
<td>A</td>
<td>C</td>
</tr>
</tbody>
</table>
What is an alignment?

Compare two strings to see how “similar” they are. E.g., maximize the # of identical chars that line up.

ATGTTAT vs ATCGTAC

<table>
<thead>
<tr>
<th>A</th>
<th>T</th>
<th>-</th>
<th>G</th>
<th>T</th>
<th>T</th>
<th>A</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>T</td>
<td>C</td>
<td>G</td>
<td>T</td>
<td>-</td>
<td>A</td>
<td>C</td>
</tr>
</tbody>
</table>

matches

mismatches
Sequence Alignment: Why

Biology
Among most widely used comp. tools in biology
DNA sequencing & assembly
New sequence always compared to data bases

Similar sequences often have similar origin and/or function
Recognizable similarity after $10^8 - 10^9$ yr

Other
spell check/correct, diff, svn/git/..., plagiarism, ...
### Taxonomy Report

<table>
<thead>
<tr>
<th>Kingdom</th>
<th>Hits</th>
<th>Organisms</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>64</td>
<td>16</td>
</tr>
<tr>
<td>Eukaryota</td>
<td>62</td>
<td>14</td>
</tr>
<tr>
<td>Fungi/Metazoa</td>
<td>57</td>
<td>11</td>
</tr>
<tr>
<td>Bilateria</td>
<td>38</td>
<td>7</td>
</tr>
<tr>
<td>Coelomata</td>
<td>36</td>
<td>6</td>
</tr>
<tr>
<td>Tetrapoda</td>
<td>26</td>
<td>5</td>
</tr>
<tr>
<td>Eutheria</td>
<td>24</td>
<td>4</td>
</tr>
<tr>
<td>Homo sapiens</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>Murinae</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Rattus norvegicus</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Mus musculus</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Sus scrofa</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Xenopus laevis</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Drosophila melanogaster</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>Caenorhabditis elegans</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Ascomycota</td>
<td>19</td>
<td>4</td>
</tr>
<tr>
<td>Schizosaccharomyces pombe</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>Saccharomycetales</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>Saccharomyces</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Saccharomyces kluyveri</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Candida albicans</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Arabidopsis thaliana</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Apicomplexa</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Plasmodium falciparum</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Toxoplasma gondii</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

### Alternate demo:

- find “BLAST” button about ½ way down page, under “Sequences”, just above big grey box with the amino sequence of this protein
- click “go” button
- after a minute or 2 you should see the 1st of 10 pages of “hits” – matches to similar proteins in other species
- you might find it interesting to look at the species descriptions and the “identity” column (generally above 50%, even in species as distant from us as fungus -- extremely unlikely by chance on a 1071 letter sequence over a 20 letter alphabet)
- Also click any of the colored “alignment” bars to see the actual alignment of the human XPO1 protein to its relative in the other species – in 3-row groups (query 1st, the match 3rd, with identical letters highlighted in between)
Terminology

**string** ordered list of letters

**prefix** consecutive letters from front

**suffix** consecutive letters from back

**substring** consecutive letters from anywhere

**subsequence** any ordered, nonconsecutive letters, i.e. AAA, TAG

T A T A A G
An alignment of strings $S$, $T$ is a pair of strings $S'$, $T'$ with dash characters “$-$” inserted, so that

1. $|S'| = |T'|$, and
2. Removing dashes leaves $S$, $T$

Consecutive dashes are called “a gap.”

(Note that this is a definition for a general alignment, not optimal.)
Scoring an arbitrary alignment

Define a score for pairs of aligned chars, e.g.

\[ \sigma(x, y) = \begin{cases} 
\text{match} & 2 \\
\text{mismatch} & -1 
\end{cases} \]

(Toy scores for examples in slides)

Apply that per column, then add.

\[
\begin{array}{cccccccc}
\text{a} & \text{c} & - & - & \text{g} & \text{c} & \text{t} & \text{g} \\
- & \text{c} & \text{a} & \text{t} & \text{g} & \text{t} & - & - \\
\end{array}
\]

\[-1 & +2 & -1 & -1 & +2 & -1 & -1 & -1\]

Total Score = -2
More Realistic Scores: BLOSUM 62
(the “σ” scores)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>R</th>
<th>N</th>
<th>D</th>
<th>C</th>
<th>Q</th>
<th>E</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>L</th>
<th>K</th>
<th>M</th>
<th>F</th>
<th>P</th>
<th>S</th>
<th>T</th>
<th>W</th>
<th>Y</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>-1</td>
<td>-2</td>
<td>-2</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>-2</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-2</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-2</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>-3</td>
</tr>
<tr>
<td>R</td>
<td>-1</td>
<td>5</td>
<td>0</td>
<td>-2</td>
<td>-3</td>
<td>1</td>
<td>0</td>
<td>-2</td>
<td>0</td>
<td>-3</td>
<td>-2</td>
<td>1</td>
<td>0</td>
<td>-3</td>
<td>-3</td>
<td>-2</td>
<td>-3</td>
<td>-1</td>
<td>-1</td>
<td>-3</td>
</tr>
<tr>
<td>N</td>
<td>-2</td>
<td>0</td>
<td>6</td>
<td>1</td>
<td>-3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-3</td>
<td>-3</td>
<td>0</td>
<td>-2</td>
<td>-3</td>
<td>-2</td>
<td>-3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-4</td>
</tr>
<tr>
<td>D</td>
<td>-2</td>
<td>-2</td>
<td>1</td>
<td>6</td>
<td>-3</td>
<td>0</td>
<td>2</td>
<td>-1</td>
<td>-1</td>
<td>-3</td>
<td>-4</td>
<td>-1</td>
<td>-3</td>
<td>-3</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>-3</td>
<td>-3</td>
<td>-3</td>
<td>9</td>
<td>-3</td>
<td>-4</td>
<td>-3</td>
<td>-3</td>
<td>-1</td>
<td>-1</td>
<td>-3</td>
<td>-1</td>
<td>-2</td>
<td>-3</td>
<td>-1</td>
<td>-1</td>
<td>-2</td>
<td>-2</td>
<td>-1</td>
</tr>
<tr>
<td>Q</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-3</td>
<td>5</td>
<td>2</td>
<td>-2</td>
<td>0</td>
<td>-3</td>
<td>-2</td>
<td>1</td>
<td>0</td>
<td>-3</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>-2</td>
<td>-1</td>
</tr>
<tr>
<td>E</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>-4</td>
<td>2</td>
<td>5</td>
<td>-2</td>
<td>0</td>
<td>-3</td>
<td>-3</td>
<td>1</td>
<td>-2</td>
<td>-3</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-3</td>
<td>-2</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>-2</td>
<td>0</td>
<td>-1</td>
<td>-3</td>
<td>-2</td>
<td>-6</td>
<td>-2</td>
<td>-4</td>
<td>-3</td>
<td>-4</td>
<td>-2</td>
<td>-3</td>
<td>-3</td>
<td>-2</td>
<td>0</td>
<td>-2</td>
<td>-2</td>
<td>-3</td>
<td>-3</td>
</tr>
<tr>
<td>H</td>
<td>-2</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>-3</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>8</td>
<td>-3</td>
<td>-3</td>
<td>-1</td>
<td>-2</td>
<td>-1</td>
<td>-2</td>
<td>-1</td>
<td>-1</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
</tr>
<tr>
<td>I</td>
<td>-1</td>
<td>-3</td>
<td>-3</td>
<td>-3</td>
<td>-1</td>
<td>-3</td>
<td>-3</td>
<td>-4</td>
<td>-3</td>
<td>4</td>
<td>2</td>
<td>-3</td>
<td>1</td>
<td>0</td>
<td>-3</td>
<td>2</td>
<td>-1</td>
<td>3</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>L</td>
<td>-1</td>
<td>-2</td>
<td>-3</td>
<td>-4</td>
<td>-1</td>
<td>-2</td>
<td>-3</td>
<td>-4</td>
<td>-3</td>
<td>2</td>
<td>4</td>
<td>-2</td>
<td>2</td>
<td>0</td>
<td>-3</td>
<td>-2</td>
<td>-1</td>
<td>-1</td>
<td>-2</td>
<td>-1</td>
</tr>
<tr>
<td>K</td>
<td>-1</td>
<td>2</td>
<td>0</td>
<td>-1</td>
<td>-3</td>
<td>1</td>
<td>1</td>
<td>-2</td>
<td>-1</td>
<td>-3</td>
<td>-2</td>
<td>5</td>
<td>-1</td>
<td>-3</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>-3</td>
<td>-2</td>
<td>-2</td>
</tr>
<tr>
<td>M</td>
<td>-1</td>
<td>-1</td>
<td>-2</td>
<td>-3</td>
<td>-1</td>
<td>0</td>
<td>-2</td>
<td>-3</td>
<td>-2</td>
<td>1</td>
<td>2</td>
<td>-1</td>
<td>5</td>
<td>0</td>
<td>-2</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>F</td>
<td>-2</td>
<td>-3</td>
<td>-3</td>
<td>-3</td>
<td>-2</td>
<td>-3</td>
<td>-3</td>
<td>-3</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>-3</td>
<td>0</td>
<td>6</td>
<td>-4</td>
<td>2</td>
<td>-2</td>
<td>1</td>
<td>3</td>
<td>-1</td>
</tr>
<tr>
<td>P</td>
<td>-1</td>
<td>-2</td>
<td>-2</td>
<td>-1</td>
<td>-3</td>
<td>-1</td>
<td>-1</td>
<td>-2</td>
<td>-2</td>
<td>-3</td>
<td>-3</td>
<td>-1</td>
<td>-2</td>
<td>-4</td>
<td>7</td>
<td>0</td>
<td>-1</td>
<td>-4</td>
<td>-3</td>
<td>-2</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-2</td>
<td>-2</td>
<td>0</td>
<td>-1</td>
<td>-2</td>
<td>-1</td>
<td>4</td>
<td>1</td>
<td>-3</td>
<td>-2</td>
<td>-2</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-2</td>
<td>-2</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-2</td>
<td>-1</td>
<td>1</td>
<td>5</td>
<td>-2</td>
<td>-2</td>
<td>0</td>
</tr>
<tr>
<td>W</td>
<td>-3</td>
<td>-3</td>
<td>-4</td>
<td>-4</td>
<td>-2</td>
<td>-2</td>
<td>-3</td>
<td>-2</td>
<td>-1</td>
<td>-1</td>
<td>-3</td>
<td>-3</td>
<td>1</td>
<td>3</td>
<td>-3</td>
<td>-1</td>
<td>1</td>
<td>11</td>
<td>2</td>
<td>-3</td>
</tr>
<tr>
<td>Y</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>-3</td>
<td>-2</td>
<td>-2</td>
<td>-3</td>
<td>-3</td>
<td>2</td>
<td>-1</td>
<td>-1</td>
<td>-2</td>
<td>-1</td>
<td>3</td>
<td>-3</td>
<td>-2</td>
<td>-2</td>
<td>2</td>
<td>7</td>
<td>-1</td>
</tr>
<tr>
<td>V</td>
<td>0</td>
<td>-3</td>
<td>-3</td>
<td>-3</td>
<td>-1</td>
<td>-2</td>
<td>-2</td>
<td>-3</td>
<td>-3</td>
<td>3</td>
<td>1</td>
<td>-2</td>
<td>1</td>
<td>-1</td>
<td>-2</td>
<td>1</td>
<td>0</td>
<td>-3</td>
<td>-1</td>
<td>4</td>
</tr>
</tbody>
</table>
Optimal Alignment:
A Simple Algorithm

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

S = agct   A = ct
T = wxyz   B = xz
-agc-t    a-gc-t
w--xyz    -w-xyz
Analysis

Assume $|S| = |T| = n$

Cost of evaluating one alignment: $\geq n$

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

Total time: $\geq n \binom{2n}{n} > 2^{2n}$, for $n > 3$

E.g., for $n = 20$, time is $> 2^{40}$ operations
Polynomial vs Exponential Growth

$2^2n$, $2^{n/10}$, $1000n^2$, $1000000n$
Fibonacci Numbers
(recursion)

```c
fibr(n) {
    if (n <= 1) {
        return 1;
    } else {
        return fibr(n-1) + fibr(n-2);
    }
}
```

Simple recursion, but many repeated subproblems!!

⇒

Time = \( \Omega(1.61^n) \)
Call tree - start

```
F (6)
  /   \
F (5)  F (4)
  /  \
F (4) F (3)
  /  \
F (3) F (2)
  /  \
F (2) F (1)
  /  \
F (1) F (0)
  /  \
1  1  1  0
```
many duplicates ⇒ exponential time!
Fibonacci, II
(dynamic programming)

```c
int fibd[n];
fibd[0] = 1;
fibd[1] = 1;
for(i=2; i<=n; i++) {
    fibd[i] = fibd[i-1] + fibd[i-2];
}
return fibd[n];
```

Avoid repeated subproblems by tabulating their solutions
⇒
Time = O(n)
(in this case)
Can we use Dynamic Programming?

1. Can we decompose into subproblems?
   E.g., can we align smaller substrings (say, prefix/suffix in this case), then combine them somehow?

2. Do we have **optimal substructure**?
   I.e., is optimal solution to a subproblem independent of context? E.g., is appending two optimal alignments also be optimal? Perhaps, but some changes at the interface might be needed?
Optimal Substructure
(In More Detail)

Optimal alignment *ends* in 1 of 3 ways:

- last chars of S & T aligned with each other
- last char of S aligned with dash in T
- last char of T aligned with dash in S

(assume $\sigma(-, -) < 0$, so never align dash with dash)

*In each case, the rest of S & T should be optimally aligned to each other*
Optimal Alignment in $O(n^2)$ 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 $0 \leq i \leq n$, $0 \leq j \leq m$. 
Base Cases

\( V(i,0): \) first \( i \) chars of \( S \) all match dashes

\[
V(i,0) = \sum_{k=1}^{i} \sigma(S[k],-) 
\]

\( V(0,j): \) first \( j \) chars of \( T \) all match dashes

\[
V(0,j) = \sum_{k=1}^{j} \sigma(-,T[k]) 
\]
General Case

Opt align of $S[1], \ldots, S[i]$ vs $T[1], \ldots, T[j]$: 

$$V(i,j) = \max \begin{cases} 
V(i-1,j-1) + \sigma(S[i], T[j]) \\
V(i-1,j) + \sigma(S[i], -) \\
V(i,j-1) + \sigma(-, T[j]) 
\end{cases}$$

for all $1 \leq i \leq n$, $1 \leq j \leq m$. 

Opt align of $S_1 \ldots S_{i-1}$ & $T_1 \ldots T_{j-1}$
Calculating One Entry

\[ V(i,j) = \max \left\{ \begin{array}{l} V(i-1,j-1) + \sigma(S[i],T[j]) \\ V(i-1,j) + \sigma(S[i], - ) \\ V(i,j-1) + \sigma(-, T[j]) \end{array} \right\} \]
### Example

<table>
<thead>
<tr>
<th>i</th>
<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></td>
<td></td>
<td>c</td>
<td>a</td>
<td>t</td>
<td>g</td>
<td>t</td>
<td></td>
</tr>
<tr>
<td>0</td>
<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>
<td></td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td>-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>g</td>
<td>-3</td>
<td></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>
<td></td>
</tr>
<tr>
<td>5</td>
<td>t</td>
<td>-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>g</td>
<td>-6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mismatch = -1  
Match = 2

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

<table>
<thead>
<tr>
<th></th>
<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>t</td>
<td>g</td>
<td>t</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>
<td></td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>-1</td>
<td></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>
<td></td>
</tr>
<tr>
<td>3</td>
<td>g</td>
<td>-3</td>
<td></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>
<td></td>
</tr>
<tr>
<td>5</td>
<td>t</td>
<td>-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>g</td>
<td>-6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Mismatch** = -1

**Match**  = 2

Score(-,a) = -1
### Example

<table>
<thead>
<tr>
<th></th>
<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>t</td>
<td>g</td>
<td>t</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>
<td></td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>-1</td>
<td></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>
<td></td>
</tr>
<tr>
<td>3</td>
<td>g</td>
<td>-3</td>
<td></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>
<td></td>
</tr>
<tr>
<td>5</td>
<td>t</td>
<td>-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>g</td>
<td>-6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mismatch = -1  
Match = 2

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

<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>t</td>
<td>g</td>
<td>t</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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>g</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>t</td>
<td>-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>g</td>
<td>-6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mismatch = -1
Match = 2

\[
\sigma(a,a) = +2 \\
\sigma(-,a) = -1 \\
\sigma(a,-) = -1
\]
Example

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>←T</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td></td>
<td>c</td>
<td>a</td>
<td>t</td>
<td>g</td>
<td>t</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>
<td></td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td></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>
<td></td>
</tr>
<tr>
<td>3</td>
<td>g</td>
<td>-3</td>
<td></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>
<td></td>
</tr>
<tr>
<td>5</td>
<td>t</td>
<td>-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>g</td>
<td>-6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mismatch = -1
Match = 2

Time = O(mn)
### Example

<table>
<thead>
<tr>
<th></th>
<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></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>
<td>-2</td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td>-2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-2</td>
</tr>
<tr>
<td>3</td>
<td>g</td>
<td>-3</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>c</td>
<td>-4</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>t</td>
<td>-5</td>
<td>-2</td>
<td>-2</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>g</td>
<td>-6</td>
<td>-3</td>
<td>-3</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Mismatch = -1
Match = 2

$\text{Mismatch} = -1$
\text{Match} = 2
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>-1</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>-2</td>
</tr>
<tr>
<td>2</td>
<td>-2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-2</td>
</tr>
<tr>
<td>3</td>
<td>-3</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>-4</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>-5</td>
<td>-2</td>
<td>-2</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>-6</td>
<td>-3</td>
<td>-3</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Ex: what are the 3 alignments? C.f. slide 12.
Finding Alignments: Trace Back

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

NB: trace back follows max terms (pink boxes; ngbr+σ), not max neighbors (white boxes). E.g., TB from yellow cell is only diagonal (ngbr=-1, term=1), not to the equally-good horizontal neighbor (term=-2).
Complexity Notes

Time = $O(mn)$, (value and alignment)

Space = $O(mn)$

Easy to get value in Time = $O(mn)$ and Space = $O(\min(m,n))$

Possible to get value \textit{and alignment} in Time = $O(mn)$ and Space = $O(\min(m,n))$
Sequence Alignment

Part II
Local alignments & gaps
Variations

Local Alignment

Preceding gives *global* alignment, i.e. full length of both strings;
Might well miss strong similarity of *part* of strings amidst dissimilar flanks

Gap Penalties

10 adjacent dashes cost 10 x one dash?

Many others

Similarly fast DP algs often possible
Local Alignment: Motivations

“Interesting” (evolutionarily conserved, functionally related) segments may be a small part of the whole

“Active site” of a protein

Scattered genes or exons amidst “junk”, e.g. retroviral insertions, large deletions

Don’t have whole sequence

Global alignment might miss them if flanking junk outweighs similar regions
Optimal \textit{local alignment} of strings S & T: Find substrings A of S and B of T having max value global alignment

\begin{align*}
S &= \text{abcxdex} \quad A = \text{c x d e} \\
T &= \text{xxxxcde} \quad B = \text{c - d e} \quad \text{value = 5 (toy \sigma)}
\end{align*}
Local Alignment: “Obvious” Algorithm

for all substrings A of S and B of T:
    Align A & B via dynamic programming
    Retain pair with max value
end ;

Output the retained pair

Time: $O(n^2)$ choices for A, $O(m^2)$ for B, $O(nm)$ for DP, so $O(n^3m^3)$ total.

[Best possible? Lots of redundant work…]
Local Alignment in $O(nm)$ via Dynamic Programming

Input: $S$, $T$, $|S| = n$, $|T| = m$

Output: value of optimal local alignment

Better to solve a “harder” problem for all $0 \leq i \leq n$, $0 \leq j \leq m$:

$$V(i,j) = \max \text{ value of opt (global) alignment of a suffix of } S[1], \ldots, S[i] \text{ with a suffix of } T[1], \ldots, T[j]$$

Report best $i,j$
Base Cases

Assume $\sigma(x,-) < 0$, $\sigma(-,x) < 0$

$V(i,0)$: some suffix of first $i$ chars of $S$; all match dashes in $T$; best suffix is empty

$V(i,0) = 0$

$V(0,j)$: similar

$V(0,j) = 0$
General Case Recurrences

Opt suffix align \( S[1], \ldots, S[i] \) vs \( T[1], \ldots, T[j] \):

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

Opt align of suffix of 
\( S_1 \ldots S_{i-1} \) & 
\( T_1 \ldots T_{j-1} \)

opt suffix alignment has:
2, 1, 1, 0
chars of S/T
## Scoring Local Alignments

<table>
<thead>
<tr>
<th>i</th>
<th>j</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>c</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>x</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>d</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>e</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>x</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The alignment matrix shows the scoring of local alignments, with gaps penalized by a cost of 0.

The ΔT indicates the alignment at the bottom right.
### Finding Local Alignments

<table>
<thead>
<tr>
<th></th>
<th>j</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>c</td>
<td>d</td>
<td>e</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>b</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>c</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>d</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>e</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

Again, arrows follow max term (not max neighbor)

One alignment is: 
c-de
cxde

What’s the other?
Notes

Time and Space = $O(mn)$
Space $O(\min(m,n))$ possible with time $O(mn)$, but finding alignment is trickier

Local alignment: “Smith-Waterman”
Global alignment: “Needleman-Wunsch”
Significance of Alignment Scores

Is “42” a good score?

*Compared to what?*

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

More on this later; a taste now, for use in next HW
Overall Alignment Significance, II
Empirical (via randomization)

You just searched with x, found “good” score for x:y
Generate N random “y-like” sequences (say N = $10^3 - 10^6$)
Align x to each & score
If k of them have score than better or equal to that 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 y-like” seqs? Scores depend on:
Length, so use same length as y
Sequence composition, so uniform 1/20 or 1/4 is a bad idea; even background $p_i$ can be dangerous (if y unusual)
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: A specific element will be last with prob $1/n$; given that, another specific element will be next-to-last with prob $1/(n-1)$, …; overall: $1/(n!)$

Alignment With Gap Penalties

**Gap**: maximal run of dashes in S’ or T’

- `ag--ttc-t` 2 gaps in S’
- `a---ttcgt` 1 gap in T’

**Motivations, e.g.**:
- mutation might insert/delete several or even many residues at once
- matching mRNA (no introns) to genomic DNA (exons and introns)
- some parts of proteins less critical
A Protein Structure:
(Dihydrofolate Reductase)

http://www.rcsb.org/pdb/explore/jmol.do?structureId=5CC9&bionumber=1
Alignment of 5 Dihydrofolate reductase proteins

<table>
<thead>
<tr>
<th></th>
<th>mouse</th>
<th>human</th>
<th>chicken</th>
<th>fly</th>
<th>yeast</th>
</tr>
</thead>
<tbody>
<tr>
<td>P00375</td>
<td>MVRPLNCIVAVSQNMIGKNGDLPWPPLRNEFKYFQRMTTSSVEGKQNLVIMGK</td>
<td>MVRPLNCIVAVSQNMIGKNGDLPWPPLRNEFKYFQRMTTSSVEGKQNLVIMGK</td>
<td>MVRPLNCIVAVSQNMIGKNGDLPWPPLRNEFKYFQRMTTSSVEGKQNLVIMGK</td>
<td>MVRPLNCIVAVSQNMIGKNGDLPWPPLRNEFKYFQRMTTSSVEGKQNLVIMGK</td>
<td>MVRPLNCIVAVSQNMIGKNGDLPWPPLRNEFKYFQRMTTSSVEGKQNLVIMGK</td>
</tr>
<tr>
<td>P00374</td>
<td>MVGSLNCIVAVSQNMIGKNGDLPWPPLRNEFRYFQRMTTSSVEGKQNLVIMGK</td>
<td>MVGSLNCIVAVSQNMIGKNGDLPWPPLRNEFRYFQRMTTSSVEGKQNLVIMGK</td>
<td>MVGSLNCIVAVSQNMIGKNGDLPWPPLRNEFRYFQRMTTSSVEGKQNLVIMGK</td>
<td>MVGSLNCIVAVSQNMIGKNGDLPWPPLRNEFRYFQRMTTSSVEGKQNLVIMGK</td>
<td>MVGSLNCIVAVSQNMIGKNGDLPWPPLRNEFRYFQRMTTSSVEGKQNLVIMGK</td>
</tr>
<tr>
<td>P00378</td>
<td>VRSLSIVAVCQNMGIGKDGNLPWPPLRNEYKYFQRMTTSSVEGKQNLVIMGK</td>
<td>VRSLSIVAVCQNMGIGKDGNLPWPPLRNEYKYFQRMTTSSVEGKQNLVIMGK</td>
<td>VRSLSIVAVCQNMGIGKDGNLPWPPLRNEYKYFQRMTTSSVEGKQNLVIMGK</td>
<td>VRSLSIVAVCQNMGIGKDGNLPWPPLRNEYKYFQRMTTSSVEGKQNLVIMGK</td>
<td>VRSLSIVAVCQNMGIGKDGNLPWPPLRNEYKYFQRMTTSSVEGKQNLVIMGK</td>
</tr>
<tr>
<td>P17719</td>
<td>MLR-FNLIVAVCENCIGIRGDLPWR-IKSELKYFSRRTKRTSDPTKQNAVMGK</td>
<td>MLR-FNLIVAVCENCIGIRGDLPWR-IKSELKYFSRRTKRTSDPTKQNAVMGK</td>
<td>MLR-FNLIVAVCENCIGIRGDLPWR-IKSELKYFSRRTKRTSDPTKQNAVMGK</td>
<td>MLR-FNLIVAVCENCIGIRGDLPWR-IKSELKYFSRRTKRTSDPTKQNAVMGK</td>
<td>MLR-FNLIVAVCENCIGIRGDLPWR-IKSELKYFSRRTKRTSDPTKQNAVMGK</td>
</tr>
<tr>
<td>P07807</td>
<td>MAGGKIPIVGIVAACLQPEMGIGFRGGLPWR-LPSEMKYFQVTSLTKDPNKNALIMGK</td>
<td>MAGGKIPIVGIVAACLQPEMGIGFRGGLPWR-LPSEMKYFQVTSLTKDPNKNALIMGK</td>
<td>MAGGKIPIVGIVAACLQPEMGIGFRGGLPWR-LPSEMKYFQVTSLTKDPNKNALIMGK</td>
<td>MAGGKIPIVGIVAACLQPEMGIGFRGGLPWR-LPSEMKYFQVTSLTKDPNKNALIMGK</td>
<td>MAGGKIPIVGIVAACLQPEMGIGFRGGLPWR-LPSEMKYFQVTSLTKDPNKNALIMGK</td>
</tr>
</tbody>
</table>

CLUSTAL W (1.82) multiple sequence alignment

http://pir.georgetown.edu/cgi-bin/multialn.pl  2/11/2013
Topoisomerase I

http://www.rcsb.org/pdb/explore.do?structureId=1a36
Affine Gap Penalties

\[ \text{Gap penalty} = g + e \cdot (\text{gaplen} - 1), \quad g \geq e \geq 0 \]

Note: no longer suffices to know just the score of best subproblem(s) – state matters: do they end with ‘-’ or not.
Global Alignment with Affine Gap Penalties

\[ V(i,j) = \text{value of opt alignment of } S[1], \ldots, S[i] \text{ with } T[1], \ldots, T[j] \]

\[ G(i,j) = \ldots, \text{s.t. last pair matches } S[i] \text{ & } T[j] \]

\[ F(i,j) = \ldots, \text{s.t. last pair matches } S[i] \text{ & } - \]

\[ E(i,j) = \ldots, \text{s.t. last pair matches } - \text{ & } T[j] \]

**Time:** \( O(mn) \) [calculate all, \( O(1) \) each]
Affine Gap Algorithm

Gap penalty = g + e*(gaplen-1), g ≥ e ≥ 0

\[ V(i,0) = E(i,0) = V(0,i) = F(0,i) = -g-(i-1)^e \]

\[ V(i,j) = \max(G(i,j), F(i,j), E(i,j)) \]

\[ G(i,j) = V(i-1,j-1) + \sigma(S[i],T[j]) \]

\[ F(i,j) = \max( F(i-1,j)-e, V(i-1,j)-g ) \]

\[ E(i,j) = \max( E(i,j-1)-e, V(i,j-1)-g ) \]

Q. Why is the “V” case a “new gap” when V includes E & F?
Other Gap Penalties

Score = f(gap length)
Kinds, & best known alignment time

- **affine**
  \[ O(n^2) \]  
  [really, \( O(mn) \)]

- **convex**
  \[ O(n^2 \log n) \]

- **general**
  \[ O(n^3) \]
Summary: Alignment

Functionally similar proteins/DNA often have recognizably similar sequences even after eons of divergent evolution.

Ability to find/compare/experiment with “same” sequence in other organisms is a huge win.

Surprisingly simple scoring works well in practice: score positions separately & add, usually w/ fancier affine gap model.

Simple dynamic programming algorithms can find optimal alignments under these assumptions in poly time (product of sequence lengths).

This, and heuristic approximations to it like BLAST, are workhorse tools in molecular biology, and elsewhere.
Summary: Dynamic Programming

Keys to D.P. are to

a) Identify the subproblems (usually repeated/overlapping)
b) Solve them in a careful order so all small ones solved before they are needed by the bigger ones, and
c) Build table with solutions to the smaller ones so bigger ones just need to do table lookups (no recursion, despite recursive formulation implicit in (a))
d) Implicitly, optimal solution to whole problem devolves to optimal solutions to subproblems

A really important algorithm design paradigm