CSE 421 Intro Algorithms
Summer 2007

Sequence Alignment
Sequence Alignment

- What
- Why
- A Simple Algorithm
- Complexity Analysis
- A better Algorithm: “Dynamic Programming”
Sequence Similarity: What

GGACCA
TACTAAG
TCCAAT
Sequence Similarity: What

GGACCA

TACTAAG

| : | : | : | : |

TCC – AAT
Sequence Similarity: Why

- Most widely used comp. tools in biology
- New sequence always compared to sequence data bases

Similar sequences often have similar origin or function

- Selection operates on system level, but mutation occurs at the sequence level
- Recognizable similarity after $10^8 - 10^9$ yr
### Taxonomy Report

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

- **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**

\[
\begin{align*}
a & \ c & b & c & d & b \\
\uparrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
\ c & a & d & b & d \\
\end{align*}
\quad
\begin{align*}
a & \ c & \quad\quad \quad \quad \quad b & c & d & b \\
\quad & \quad & \quad & \quad & \quad & \quad \\
\quad & \quad & \quad & \quad & \quad & \quad \\
\quad & \quad & \quad & \quad & \quad & \quad \\
\end{align*}
\]

Defn: 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

\[ \begin{align*}
\text{a} & \quad \text{c} & \quad \text{b} & \quad \text{c} & \quad \text{d} & \quad \text{b} \\
\text{c} & \quad \text{a} & \quad \text{d} & \quad \text{b} & \quad \text{d} \\
\text{b} & \quad \text{c} & \quad \text{a} & \quad \text{d} & \quad \text{b} & \quad \text{d} \\
\text{Value} = 3*2 + 5*(-1) = +1
\end{align*} \]

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

Mismatch = -1
Match = 2
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

<table>
<thead>
<tr>
<th>S = abcd</th>
<th>A = cd</th>
</tr>
</thead>
<tbody>
<tr>
<td>T = wxyz</td>
<td>B = xz</td>
</tr>
<tr>
<td>-abc-d</td>
<td>a-bc-d</td>
</tr>
<tr>
<td>w--xyz</td>
<td>-w-xyz</td>
</tr>
</tbody>
</table>
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 \(T\)
• Total time: \(\geq n \binom{2n}{n} > 2^{2n}, \text{ for } n > 3\)
• E.g., for \(n = 20\), time is \(> 2^{40}\) operations—bad!
Candidate for Dynamic Programming?

- **Common Subproblems?**
  - Plausible: probably re-considering alignments of various small substrings unless we're careful.

- **Optimal Substructure?**
  - Plausible: left and right "halves" of an optimal alignment probably should be optimally aligned (though they obviously interact a bit at the interface).

  - (Both made rigorous below.)
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 space in T
  • last char of T aligned with space in S
  • ( never align space with space; $\sigma(\text{-}, \text{-}) < 0$ )

• 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:

$$V(i,j) = \text{value of optimal alignment of } S[1], \ldots, S[i] \text{ with } T[1], \ldots, T[j] \text{ for all } 0 \leq i \leq n, 0 \leq j \leq m.$$
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
  
  $$V(0,j) = \sum_{k=1}^{j} \sigma(-,T[k])$$
General Case

Opt align of $S[1], \ldots, S[i] \text{ vs } T[1], \ldots, T[j]$: $V(i,j) = \max \left\{ V(i-1,j-1) + \sigma(S[i],T[j]) \right\}$

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

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

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

<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>-6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mismatch = -1
Match = 2

Time = O(mn)
## Example

Mismatch = -1  
Match = 2

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

↑T
### Finding Alignments: Trace Back

<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>d</td>
<td>b</td>
<td>d</td>
<td>T</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>b</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>d</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>b</td>
<td>-6</td>
<td>-3</td>
<td>-3</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
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 and alignment in Time = O(mn) and Space = O(min(m,n)) but tricky.
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 spaces cost 10 x one space?

• Many others
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
Local Alignment

Optimal *local alignment* of strings S & T:
Find substrings A of S and B of T having max value global alignment

\[
S = abcxdex \\ A = c \ x \ d \ e \\
T = xxxcde \ \ B = c - d \ e \ \ value = 5
\]
The “Obvious” Local Alignment 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 $\sigma(x, -) \leq 0$, $\sigma(-, x) \leq 0$
- $V(i, 0)$: some suffix of first $i$ chars of $S$; all match spaces 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 $T[1], \ldots, T[j]$:

$$V(i,j) = \max \begin{cases} 
V(i-1,j-1) + \sigma(S[i], T[j]) \\
V(i-1,j) + \sigma(S[i], -) \\
V(i,j-1) + \sigma(-, T[j]) \\
0 
\end{cases},$$

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

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></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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<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>

S → T
## Finding Local Alignments

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

↑ S ← T
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”
Alignment With Gap Penalties

- **Gap**: maximal run of spaces in S’ or T’
  
  \[
  \text{ab----c-d} \\
  \text{a-ddddcbbd} \\
  \]
  2 gaps in S’, 1 in T’

- **Motivations, e.g.:**
  - mutation might insert/delete several or even many residues at once
  - matching cDNA (no introns) to genomic DNA (exons and introns)
Gap Penalties

- Score = f(gap length)
- Kinds, & best known alignment time
  - general: \( O(n^3) \)
  - convex: \( O(n^2 \log n) \)
  - affine: \( O(mn) \)
Global Alignment with Affine Gap Penalties

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

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

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

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

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

Gap penalty = \( g + s^*(\text{gap length}) \), \( g, s \geq 0 \)

\[ V(i,0) = E(i,0) = V(0,i) = F(0,i) = -g - i^*s \]

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

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

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

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

old gap \hspace{1cm} new gap
Summary

• In bio, similar sequences usually => same function (even after eons of divergent evolution)
• Surprisingly simple scoring model works well in practice: score each position separately & add, (possibly w/ fancier gap model like affine)
• Simple “dynamic programming” algorithms 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
• Many applications outside bio, too. (Spelling correction, spam detection, unix “diff”, CVS compression,…)}