CSE 521 Algorithms

Sequence Alignment
CSE421 Algorithms

Sequence Alignment
CSEP 590 A
Computational Biology
Autumn 2013

Lecture 2
Sequence Alignment
Tonight

Last week’s “quiz” & homework
Sequence alignment
Weekly “bio” interlude - DNA replication
More sequence alignment
“HW 0” Background Poll

In your own words, what is DNA? Its main role?
What is RNA? What is its main role in the cell?
How many amino acids are there? How many are used in proteins?
Did human beings, as we know them, develop from earlier species of animals?
What are stem cells?
What did Viterbi invent?
What is dynamic programming?
What is a likelihood ratio test?
What is the EM algorithm?
How would you find the maximum of \( f(x) = ax^3 + bx^2 + cx + d \) in the interval \(-10<x<25\)?

Don’t worry, we’ll talk about all this stuff before the course ends.
Evolution & Scientific Literacy

“Human beings, as we know them, developed from earlier species of animals”
(avoiding the now politically charged word “evolution”)

From 1985 to 2005, the % of Americans
rejecting: declined from 48% to 39%
accepting: also declined 45% to 40
uncertain: increased 7% to 21%

In a 2005 survey, the proportion of adults who accept evolution in 34 countries (US, Europe, Japan…), the United States ranked 33rd, just above/below Turkey.

Sequence Alignment

Part I
Motivation, dynamic programming, global alignment
Sequence Alignment

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

What
Why
A Dynamic Programming Algorithm
Sequence Similarity: What

G G A C C A
T A C T A A G
T C C A A G
Sequence Similarity: What

G G A C C A

T A C T A A G

T C C – A A G
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**

Recognizable similarity after $10^8 - 10^9$ yr
Sequence Similarity: Why

Bio
Most widely used comp. tools in biology
New sequence always compared to data bases
Similar sequences often have similar origin or function
Recognizable similarity after $10^8 - 10^9$ yr
DNA sequencing & assembly

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

root ........................................ 64 hits  16 orgs
  . Eukaryota ............................. 62 hits  14 orgs [cellular organisms]
    . . Fungi/Metazoa group ............ 57 hits  11 orgs
    . . . Bilatera .......................... 38 hits  7 orgs [Metazoa; Eumetazoa]
    . . . Coelomata ........................ 36 hits  6 orgs
    . . . . Tetrapoda ................. 26 hits  5 orgs [Vertebrata; Sarcopterygii]
    . . . . Eutheria .................... 24 hits  4 orgs [Amniota; Mammalia; Theria]
    . . . . Homo sapiens ................ 20 hits  1 orgs [Hominidae; Homo]
    . . . . Murinae ...................... 16 hits  2 orgs [Rodentia; Sciurognathi; Muridae]
    . . . . . Rattus norvegicus .......  10 hits  1 orgs [Rattus]
    . . . . . Mus musculus .............  1 hits  1 orgs [Mus]
    . . . . . Sus scrofa ..............  1 hits  1 orgs [Cetartiodactyla; Suina; Suidae; Sus]
    . . . . Xenophasia laevis ...........  2 hits  1 orgs [Amphibia; Xenopodinae; Xenopus]
    . . . . Drosophila melanogaster ... 10 hits  1 orgs [Protostomia; Drosophila]
    . . . . Caenorhabditis elegans ...... 19 hits  4 orgs [Fungi]
    . . . Ascomycota ...................... 10 hits  1 orgs [Saccharomyces]
    . . . . Schizosaccharomyces pombe ...  9 hits  3 orgs [Saccharomycotina; Saccharomycetes]
    . . . . Saccharomyces ................  8 hits  2 orgs [Saccharomyces]
    . . . . . Saccharomyces cerevisiae  17 hits  1 orgs
    . . . . . . Saccharomyces kluveri ...  1 hits  1 orgs
    . . . . . Candida albicans ..........  1 hits  1 orgs [mitospic Saccharomyces]
    . . . Arabidopsis thaliana ..........  2 hits  1 orgs [Viridiplantae; Brassicaceae]
    . . . Apicomplexa ....................  3 hits  2 orgs [Alveolata]
    . . . Plasmodium falciparum ......  2 hits  1 orgs [Haemosporida; Plasmodia]
    . . . Toxoplasma gondii ............  1 hits  1 orgs [Coccidia; Eimeriida; Sarcocystidae]
    . . synthetic construct ............  1 hits  1 orgs [other; artificial sequence]
    . . lymphocystis disease virus ....  1 hits  1 orgs [Viruses; dsDNA viruses, no RNA ...]

BLAST Demo

Try it!
pick any protein, e.g. hemoglobin, insulin, exportin,… BLAST to find distant relatives.
**Terminology**

**String:** ordered list of letters  
*Example:* TATAAG

**Prefix:** consecutive letters from front  
*Examples:* empty, T, TA, TAT, ...

**Suffix:** … from end  
*Examples:* empty, G, AG, AAG, ...

**Substring:** … from ends or middle  
*Examples:* empty, TAT, AA, ...

**Subsequence:** ordered, nonconsecutive  
*Examples:* TT, AAA, TAG, ...
Sequence Alignment

\[
\begin{align*}
a & c & b & c & d & b \\
\underline{c} & a & d & b & d \\
\end{align*}
\]

\[
\begin{align*}
a & c & -- & b & c & d & b \\
\underline{c} & a & d & b & -- & d & -- \\
\end{align*}
\]

**Defn:** An *alignment* of strings S, T is a pair of strings S’, T’ (with dashes) s.t.

1. \(|S’| = |T’|\), and \((|S| = “length of S”\))
2. removing all dashes leaves S, T
Alignment Scoring

The *score* of aligning (characters or dashes) x & y is $\sigma(x,y)$.

**Value** of an alignment

$$\sum_{i=1}^{\vert S' \vert} \sigma(S'[i],T'[i])$$

An *optimal alignment*: one of max value

(Assume $\sigma(-,-) < 0$)
Optimal Alignment: A Simple Algorithm

for all subseqs $A$ of $S$, $B$ of $T$ s.t. $|A| = |B|$ do
  align $A[i]$ with $B[i]$, $1 \leq i \leq |A|$,
  align all other chars to spaces,
  compute its value,
  retain the max,
end
output the retained alignment
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: $\geq n \binom{2n}{n} > 2^{2n}$, for $n > 3$

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

How does run time grow as a function of problem size?

\[ n^2 \text{ or } 100 \, n^2 + 100 \, n + 100 \text{ vs } 2^{2n} \]

**Defn:** \( f(n) = O(g(n)) \) iff there is a constant \( c \) s.t. \[ |f(n)| \leq cg(n) \] for all sufficiently large \( n \).

- \( 100 \, n^2 + 100 \, n + 100 = O(n^2) \) \[ \text{[e.g. } c = 101] \]
- \( n^2 = O(2^{2n}) \)
- \( 2^{2n} \text{ is } not\: O(n^2) \)
Big-O Example

\[ f(n) = O(g(n)) = O(g'(n)) \]
Utility of Asymptotics

“All things being equal,” smaller asymptotic growth rate is better

All things are never equal

Even so, big-O bounds often let you quickly pick most promising candidates among competing algorithms

Poly time algs often practical; non-poly algs seldom are.

(Yes, there are exceptions.)
Fibonacci Numbers
(recursion)

fibr(n) {
    if (n <= 1) {
        return 1;
    } else {
        return fibr(n-1) + fibr(n-2);
    }
}
Call tree - start

F (6)
  /   \
F (5)   F (4)
  /   \
F (4)   F (3)
  /   \
F (2)   F (2)
  /   \
F (3)   F (1)
  /  \  /  \
F (2) F (1) F (0)
    /  \  \
   1   1 0
Full call tree

many duplicates $\Rightarrow$ exponential time!
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];
Alignment by 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 \textcolor{blue}{\textit{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

( never align dash with dash; \( \sigma(\sim, \sim) < 0 \) )

In each case, the \textcolor{blue}{\textit{rest}} of S & T should be \textcolor{blue}{\textit{optimally}} aligned to each other
Optimal Substructure

Optimal alignment \textit{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
  ( never align dash with dash; \( \sigma(-, -) < 0 \) )

In each case, the \textit{rest} of S & T should be \textit{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

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

\[ \begin{aligned}
V(i-1,j) + \sigma(S[i], -) \\
V(i,j-1) + \sigma(-, T[j])
\end{aligned} \]
### 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>d</td>
<td>b</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></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>
<td></td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>c</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>d</td>
<td></td>
<td></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>
<td></td>
</tr>
</tbody>
</table>

Mismatch = -1  
Match = 2

Score(c, -) = -1
Example

Mismatch = -1
Match = 2

<table>
<thead>
<tr>
<th></th>
<th>j</th>
<th>i</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></td>
<td>c</td>
<td>a</td>
<td>d</td>
<td>b</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>i</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>←T</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>
<td></td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>-1</td>
<td></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>
<td></td>
</tr>
<tr>
<td>3</td>
<td>b</td>
<td>-3</td>
<td></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>
<td></td>
</tr>
<tr>
<td>5</td>
<td>d</td>
<td>-5</td>
<td></td>
<td></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>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Mismatch = -1
Match   = 2

Score(-,c) = -1
Example

Mismatch = -1
Match = 2

\[
\begin{array}{ccccccc}
\text{i} & \text{j} & 0 & 1 & 2 & 3 & 4 & 5 \\
\hline
0 & 0 & -1 & -2 & -3 & -4 & -5 & \leftarrow T \\
1 & a & -1 & -1 & \textbf{1} & & & \\
2 & c & -2 & & & & & \\
3 & b & -3 & & & & & \\
4 & c & -4 & & & & & \\
5 & d & -5 & & & & & \\
6 & b & -6 & & & & & \\
\end{array}
\]

\[\sigma(a,a) = +2\]
\[\sigma(-,a) = -1\]
\[\sigma(a,-) = -1\]
\[\sigma(a,-) = -1\]
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>d</td>
<td>b</td>
<td>d</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>b</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>d</td>
<td>-5</td>
<td></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>
<td></td>
</tr>
</tbody>
</table>

Mismatch = -1
Match = 2

Time = O(mn)
### Example

![Dynamic Programming Table]

<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>-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>

Mismatch = -1
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>
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 = $O(\min(m,n))$ (KT section 6.7)
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 (DEKM 2.6)
Significance of Alignments

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

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

More on this next time; a taste today, for use in next HW
Significance of Alignments

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

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

Interesting stats problem; much is known
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 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 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
Better idea: permute y N times
Generating Random Permutations

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, a specific other element will be next-to-last with prob 1/(n-1), …; overall: 1/(n!)
Weekly Bio Interlude

DNA Replication
DNA Replication: Basics
1st ~10 nt’s added are called the primer
In simple model, DNA pol has 2 jobs: prime & extend
Priming is error-prone
So, specialized primase does the priming; pol specialized for fast, accurate extension
Still doesn’t solve the accuracy problem (hint: primase makes an RNA primer)
Issue 2: Rep Forks & Helices

“Replication Fork”: DNA double helix is progressively unwound by a DNA helicase, and both resulting single strands are duplicated.

DNA polymerase synthesizes new strand 5' -> 3' (reading its template strand 3' -> 5')

That means on one (the “leading”) strand, DNA pol is chasing/pushing the replication fork.

But on the other “lagging” strand, DNA pol is running away from it.
Lagging strand gets a series of “Okazaki fragments” of DNA (~200nt in eukaryotes) following each primer.

The RNA primers are later removed by a nuclease and DNA pol fills gaps (more accurate than primase; primed by DNA from adjacent Okazaki frag).

Fragments joined by ligase.
Issue 4: Coord of Leading/Lagging

Alberts et al., Mol. Biol. of the Cell, 3rd ed, p258
newly synthesized leading strand

newly synthesized lagging strand

parental DNA helix
Issue 5: Twirls & Tangles

Unwinding helix (~10 nucleotides per turn) would cause stress. *Topoisomerase I* cuts DNA backbone on one strand, allowing it to spin about the remaining bond, relieving stress.

*Topoisomerase II* can cut & rejoin both strands, after allowing another double strand to pass through the gap, de-tangling it.
Issue 6: Proofreading

Error rate of pol itself is $\sim 10^{-4}$, but overall rate is $10^{-9}$, due to proofreading & repair, e.g.

pol itself can back up & cut off a mismatched base if one happens to be inserted

priming the new strand is hard to do accurately, hence RNA primers, later removed & replaced

other enzymes scan helix for “bulges” caused by base mismatch, figure out which strand is original, cut away new (faulty) copy; DNA pol fills gap

which strand is original? Bacteria: “methylate” some A’s, eventually. Euks: strand nicking
Replication Summary

Speed: 50 (eukaryotes) to 500 (prokaryotes) bp/sec
Accuracy: 1 error per $10^9$ bp
Complex & highly optimized
Highly similar across all living cells

More info:
Alberts et al., *Mol. Biol. of the Cell*
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
Similarly fast DP algs often possible
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
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
Local Alignment

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

\[
S = abcxdex 
\]

\[
T = xxxcde 
\]

\[
A = cxde 
\]

\[
B = c - d e 
\]

value = 5
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 $0 \leq i \leq n$, $0 \leq j \leq m$:

$V(i,j) = \max$ value of opt (global) alignment of a suffix of $S[1], \ldots, S[i]$ 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 $1 \leq i \leq n, 1 \leq j \leq m$.
### Scoring Local Alignments

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

↑ | j
↓ s ↔ T
## Finding Local Alignments

![Alignment Table](alignment_table.png)

Again, arrows follow max.
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”
Sequence Evolution

“Nothing in Biology Makes Sense Except in the Light of Evolution” – Theodosius Dobzhansky, 1973

Changes happen at random
Deleterious/neutral/advantageous changes unlikely/possibly/likely spread widely in a population
Changes are less likely to be tolerated in positions involved in many/close interactions, e.g.
   - enzyme binding pocket
   - protein/protein interaction surface
   ...

Alignment With Gap Penalties

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

- `ab--ddc-d` 2 gaps in S’
- `a---ddcbd` 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

(NB: KT treats “gap” and “-” as synonyms)
Alignment With Gap Penalties

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

- ab--ddc–d 2 gaps in S’
- a---ddcbd 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)
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>MVRPLN</td>
<td>CVSQ</td>
<td>GKNGL</td>
<td>DLPWP</td>
<td>LRNEFYFQRMTTTSVEGKQNLVIMGKR</td>
</tr>
<tr>
<td>P00374</td>
<td>MVPGLN</td>
<td>CVSQ</td>
<td>GKNGL</td>
<td>DLPWP</td>
<td>LRNEFYFQRMTTTSVEGKQNLVIMGKK</td>
</tr>
<tr>
<td>P00378</td>
<td>VRSNSIAVCQ</td>
<td>NMGIG</td>
<td>KDGNL</td>
<td>PWPLRNE</td>
<td>KYFQRMTSTSHVEGKQNAVIMGKK</td>
</tr>
<tr>
<td>P17719</td>
<td>MLRFNL</td>
<td>VAVCENF</td>
<td>GIGRGL</td>
<td>DLPWP</td>
<td>IKSELKYFSRTTTRSDPTKQNAVVMGRK</td>
</tr>
<tr>
<td>P07807</td>
<td>MAGGKIPIVG</td>
<td>VACLQPEMG</td>
<td>IGFRRG</td>
<td>GLPWR</td>
<td>LPEMKYFQVTSLTKDPNKNNALIMGKR</td>
</tr>
</tbody>
</table>

Alignment:

```
P00375  TWFSIPEKRNPLKDRINIVLSRELKEP-----PRGAHFLAKSLDDALRLIEQPELASKVDM
P00374  TWFSIPEKRNPLKGRINLVLSRELKEP-----PQGAHFLSRLDDLALKLTEQPELANKVDM
P00378  TWFSIPEKRNPLKDRINIVLSRELKEA-----PKGAHYLSKSLDDLALLDSPELKSVKVDM
P17719  TYFGVPESKRPPLPDRLNIVLSTTLQESDL--PKG-VLLCPNLETAMKILEE---QNEVEN
P07807  TWESIPPKFRPLPNRMNVIISRSFKDDFVHDKERSIVQSNLSLANAIMNLESN-FKEHLER
```

```
P00375  VWIVGGSSVYQEAMNQPGHLRLFLVFTRIMPQEFESDTTFPEIDLGKYLLEPYPG------
P00374  VWIVGGSSVYKEAMNHGPGLKLFVFTRIMPQDFESDTTFPEIDLEKYKLLPEYPG------
P00378  VWIVGGTAVYKAAMEKPINHRLFLVFTRILHEFESDTTFPEIDYKDFKLLTEYPG------
P17719  IWIVGSGVYEEAMASPCHRRLYITKIMQKFDCDTFPPAIP-DSFREVAPPSD------
P07807  IYVIGGGEVQSFISITDHWLITKINPLDNATPADMTFILAKEEEVFSQPDAQLKEF
```

```
P00375  VLSEVQ---------EKEGIKYKFIEVEYKKD------
P00374  VLSDVQ---------EKEGIKYKFIEVEYKND------
P00378  VPADIO---------EEDGIQYKFEVYQKSVLAQ
P17719  MPLGVO---------EENGKFEYKILEKHS------
P07807  LPPKVELPETDCDQRYSLLEEKGYCFEFTLYNKR------
```

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

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

Note: no longer suffices to know just the \textit{score} of best subproblem(s) – \textit{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 = \( g + e^{*}(\text{gaplen}-1) \), \( g \geq e \geq 0 \)

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

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

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

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

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

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

Score = \( f(\text{gap length}) \)

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 gap model like affine.

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.
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
Seminars

CSE 590C
“Reading and Research in Computational Biology”
Mondays, 3:30-4:30ish, EEB 026
http://www.cs.washington.edu/590c

GENOME 521
“COMBI”
Wednesdays, 1:30-2:50 Foege S060