Dynamic-Programming Strategies for Analyzing Biomolecular Sequences

Kun-Mao Chao
Department of Life Science
National Yang-Ming University, Taiwan
2002 in Singapore

E-mail: kmchao@ym.edu.tw
WWW: http://www.ym.edu.tw/~kmchao
Dynamic Programming

• Dynamic programming is a class of solution methods for solving sequential decision problems with a compositional cost structure.

• Richard Bellman was one of the principal founders of this approach.
Two key ingredients

- Two key ingredients for an optimization problem to be suitable for a dynamic-programming solution:
  
  1. optimal substructures
  2. overlapping subproblems

Each substructure is optimal.
(Principle of optimality)

Subproblems are dependent.
(otherwise, a divide-and-conquer approach is the choice.)
Three basic components

• The development of a dynamic-programming algorithm has three basic components:
  – The recurrence relation (for defining the value of an optimal solution);
  – The tabular computation (for computing the value of an optimal solution);
  – The traceback (for delivering an optimal solution).
Fibonacci numbers

The Fibonacci numbers are defined by the following recurrence:

\[ F_0 = 0, \]
\[ F_1 = 1, \]
\[ F_i = F_{i-1} + F_{i-2} \quad \text{for} \quad i > 1. \]
How to compute $F_{10}$?
Tabular computation

- The tabular computation can avoid recomputation.

<table>
<thead>
<tr>
<th>$F_0$</th>
<th>$F_1$</th>
<th>$F_2$</th>
<th>$F_3$</th>
<th>$F_4$</th>
<th>$F_5$</th>
<th>$F_6$</th>
<th>$F_7$</th>
<th>$F_8$</th>
<th>$F_9$</th>
<th>$F_{10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>13</td>
<td>21</td>
<td>34</td>
<td>55</td>
</tr>
</tbody>
</table>
Maximum-sum interval

• Given a sequence of real numbers $a_1a_2...a_n$, find a consecutive subsequence with the maximum sum.

\[9 \ -3 \ 1 \ 7 \ -15 \ 2 \ 3 \ -4 \ 2 \ -7 \ 6 \ -2 \ 8 \ 4 \ -9\]

For each position, we can compute the maximum-sum interval starting at that position in $O(n)$ time. Therefore, a naive algorithm runs in $O(n^2)$ time.
\(O\)-notation: an asymptotic upper bound

- \(f(n) = O(g(n))\) iff there exist two positive constant \(c\) and \(n_0\) such that \(0 \leq f(n) \leq cg(n)\) for all \(n \geq n_0\)
How functions grow?

<table>
<thead>
<tr>
<th>function ( n )</th>
<th>( 30n )</th>
<th>( 92n \log n )</th>
<th>( 26n^2 )</th>
<th>( 0.68n^3 )</th>
<th>( 2^n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.003 sec.</td>
<td>0.003 sec.</td>
<td>0.0026 sec.</td>
<td>0.00068 sec.</td>
<td>( 4 \times 10^{16} ) yr.</td>
</tr>
<tr>
<td>100,000</td>
<td>3.0 sec.</td>
<td>2.6 min.</td>
<td>3.0 days</td>
<td>22 yr.</td>
<td></td>
</tr>
</tbody>
</table>

(Assume one million operations per second.)

For large data sets, algorithms with a complexity greater than \( O(n \log n) \) are often impractical!
Maximum-sum interval (The recurrence relation)

- Define $S(i)$ to be the maximum sum of the intervals ending at position $i$.

$$S(i) \leftarrow a_i + \max\left\{ S(i-1), 0 \right\}$$

If $S(i-1) < 0$, concatenating $a_i$ with its previous interval gives less sum than $a_i$ itself.
# Maximum-sum interval (Tabular computation)

The maximum sum

<table>
<thead>
<tr>
<th></th>
<th>9</th>
<th>-3</th>
<th>1</th>
<th>7</th>
<th>-15</th>
<th>2</th>
<th>3</th>
<th>-4</th>
<th>2</th>
<th>-7</th>
<th>6</th>
<th>-2</th>
<th>8</th>
<th>4</th>
<th>-9</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S(i)$</td>
<td>9</td>
<td>6</td>
<td>7</td>
<td>14</td>
<td>-1</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>-4</td>
<td>6</td>
<td>4</td>
<td>12</td>
<td>16</td>
<td>7</td>
</tr>
</tbody>
</table>

The maximum sum is indicated by a circle.
Maximum-sum interval
(Traceback)

The maximum-sum interval: 6 -2 8 4
Defining scores for alignment columns

- *infocon* [Stojanovic *et al.*, 1999]
  - Each column is assigned a score that measures its information content, based on the frequencies of the letters both within the column and within the alignment.

\[
\begin{array}{cccccccc}
C & G & G & A & T & C & A & T & - & G & G & A \\
\end{array}
\]
Defining scores (cont’d)

• *phylogen* [Stojanovic *et al*., 1999]
  
  – columns are scored based on the evolutionary relationships among the sequences implied by a supplied phylogenetic tree.

\[
\begin{align*}
\text{Score } &= 1 \\
\text{Score } &= 2
\end{align*}
\]
Two fundamental problems we recently solved (joint work with Lin and Jiang)

• Given a sequence of real numbers of length $n$ and an upper bound $U$, find a consecutive subsequence of length at most $U$ with the maximum sum --- an $O(n)$-time algorithm.

$U = 3$

9 -3 1 7 -15 2 3 -4 2 -7 6 -2 $\boxed{8}$ $\boxed{4}$ -9
Two fundamental problems we recently solved (joint work with Lin and Jiang)

• Given a sequence of real numbers of length $n$ and a lower bound $L$, find a consecutive subsequence of length at least $L$ with the maximum average. --- an $O(n \log L)$-time algorithm.

$L = 4$

3 2 14 6 6 2 10 2 6 6 14 2 1
Another example

Given a sequence as follows:
2, 6.6, 6.6, 3, 7, 6, 7, 2

and $L = 2$, the highest-average interval is the squared area, which has the average value $20/3$.

2, 6.6, 6.6, 3, [7, 6, 7, 2]
C+G rich regions

- Our method can be used to locate a region of length at least $L$ with the highest C+G ratio in $O(n \log L)$ time.

ATGACTCGAGCTCGTCA

00101011011011010

Search for an interval of length at least $L$ with the highest average.
Length-unconstrained version

- Maximum-average interval

3 2 14 6 6 2 10 2 6 6 14 2 1

The maximum element is the answer. It can be done in $O(n)$ time.
A naive algorithm

- A simple shift algorithm can compute the highest-average interval of a fixed length in $O(n)$ time

- Try $L, L+1, L+2, \ldots, n$. In total, $O(n^2)$. 
A pigeonhole principle

• Notice that the length of an optimal interval is bounded by $2L$, we immediately have an $O(nL)$-time algorithm.

We can bisect a region of length $\geq 2L$ into two segments, where each of them is of length $\geq L$. 
Future Development

• Best $k$ (nonintersecting) subsequences?
• Max-average with both upper and lower length bounds
• General (gapped) local alignment with length upper bound.
• Measurement of goodness?
Longest increasing subsequence (LIS)

- The longest increasing subsequence is to find a longest increasing subsequence of a given sequence of distinct integers $a_1a_2\ldots a_n$.

  e.g. 9 2 5 3 7 11 8 10 13 6

  2 3 7  
  5 7 10 13

  9 7 11  
  3 5 11 13

are increasing subsequences.

We want to find a longest one.

are not increasing subsequences.
A naive approach for LIS

• Let $L[i]$ be the length of a longest increasing subsequence ending at position $i$.

$$L[i] = 1 + \max_{j=0..i-1} \{ L[j] \mid a_j < a_i \}$$

(use a dummy $a_0 = \text{minimum}$, and $L[0]=0$)

9  2  5  3  7  11  8  10  13  6

$L[i]$  1  1  2  2  3  4  ?

Diagram
A naive approach for LIS

\[ L[i] = 1 + \max_{j = 0..i-1} \{ L[j] \mid a_j < a_i \} \]

The subsequence 2, 3, 7, 8, 10, 13 is a longest increasing subsequence.

This method runs in \( O(n^2) \) time.
Binary search

- Given an ordered sequence $x_1 x_2 \ldots x_n$, where $x_1 < x_2 < \ldots < x_n$, and a number $y$, a binary search finds the largest $x_i$ such that $x_i < y$ in $O(\log n)$ time.
Binary search

• How many steps would a binary search reduce the problem size to 1?

\[ n \quad n/2 \quad n/4 \quad n/8 \quad n/16 \quad \ldots \quad 1 \]

How many steps? \( O(\log n) \) steps.

\[
\frac{n}{2^s} = 1
\]

\[ \Rightarrow s = \log_2 n \]
An $O(n \log n)$ method for LIS

- Define $BestEnd[k]$ to be the smallest number of an increasing subsequence of length $k$. 

\[
\begin{array}{cccccccccc}
9 & 2 & 5 & 3 & 7 & 11 & 8 & 10 & 13 & 6 \\
9 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\
5 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
7 & 7 & 7 & 7 & 7 & 7 & 7 & 7 & 7 & 7 \\
11 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\
10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\
13 & & & & & & & & &
\end{array}
\]

$\rightarrow$ $BestEnd[1]$

$\rightarrow$ $BestEnd[2]$

$\rightarrow$ $BestEnd[3]$

$\rightarrow$ $BestEnd[4]$

$\rightarrow$ $BestEnd[5]$

$\rightarrow$ $BestEnd[6]$
An $O(n \log n)$ method for LIS

- Define $BestEnd[k]$ to be the smallest number of an increasing subsequence of length $k$.

For each position, we perform a binary search to update $BestEnd$. Therefore, the running time is $O(n \log n)$.
Longest Common Subsequence (LCS)

• A subsequence of a sequence $S$ is obtained by deleting zero or more symbols from $S$. For example, the following are all subsequences of “president”: pred, sdn, predent.

• The longest common subsequence problem is to find a maximum-length common subsequence between two sequences.
LCS

For instance,

Sequence 1: president
Sequence 2: providence
Its LCS is priden.
LCS

Another example:

Sequence 1: algorithm
Sequence 2: alignment

One of its LCS is algm.
How to compute LCS?

- Let \( A = a_1a_2\ldots a_m \) and \( B = b_1b_2\ldots b_n \).
- \( \text{len}(i, j) \): the length of an LCS between \( a_1a_2\ldots a_i \) and \( b_1b_2\ldots b_j \).
- With proper initializations, \( \text{len}(i, j) \) can be computed as follows.

\[
\text{len}(i, j) = \begin{cases} 
0 & \text{if } i = 0 \text{ or } j = 0, \\
\text{len}(i-1, j-1) + 1 & \text{if } i, j > 0 \text{ and } a_i = b_j, \\
\max(\text{len}(i, j-1), \text{len}(i-1, j)) & \text{if } i, j > 0 \text{ and } a_i \neq b_j.
\end{cases}
\]
procedure \( \text{LCS-Length}(A, B) \)

1. \( \text{for } i \leftarrow 0 \text{ to } m \) \( \text{do } \)
   \( \text{len}(i, 0) = 0 \)
2. \( \text{for } j \leftarrow 1 \text{ to } n \) \( \text{do } \)
   \( \text{len}(0, j) = 0 \)
3. \( \text{for } i \leftarrow 1 \text{ to } m \) \( \text{do } \)
4. \( \text{for } j \leftarrow 1 \text{ to } n \) \( \text{do } \)
5. \( \text{if } a_i = b_j \) \( \text{then } \)
   \( \begin{align*}
   \text{len}(i, j) &= \text{len}(i-1, j-1) + 1 \\
   \text{prev}(i, j) &= "\downarrow" 
\end{align*} \)
6. \( \text{else if } \text{len}(i-1, j) \geq \text{len}(i, j-1) \)
7. \( \text{then } \)
   \( \begin{align*}
   \text{len}(i, j) &= \text{len}(i-1, j) \\
   \text{prev}(i, j) &= "\uparrow" 
\end{align*} \)
8. \( \text{else } \)
   \( \begin{align*}
   \text{len}(i, j) &= \text{len}(i, j-1) \\
   \text{prev}(i, j) &= "\leftarrow" 
\end{align*} \)
9. \( \text{return } \text{len} \text{ and } \text{prev} \)
<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>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>p</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>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>r</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>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>o</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>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>v</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>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>p</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>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>p</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>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>r</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>r</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>e</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>s</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>e</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>i</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>i</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>d</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>d</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>e</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>e</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>n</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>n</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>t</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>t</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>t</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>
procedure $Output-LCS(A, prev, i, j)$

1. if $i = 0$ or $j = 0$ then return

2. if $prev(i, j) =$ "then $\begin{cases} Output - LCS(A, prev, i - 1, j - 1) \\ print a_i \end{cases}$

3. else if $prev(i, j) =$ " then $Output-LCS(A, prev, i-1, j)$

4. else $Output-LCS(A, prev, i, j-1)$
<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>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>j</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>p</td>
<td>r</td>
<td>o</td>
<td>v</td>
<td>i</td>
<td>d</td>
<td>e</td>
<td>n</td>
<td>c</td>
<td>e</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>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>p</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</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>2</td>
<td>r</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>e</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>s</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>i</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>d</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>e</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>n</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>t</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

Output: *priden*
Dot Matrix

Sequence A : CTTAACT

Sequence B : CGGATCAT

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

Sequence A: CTAAACT
Sequence B: CGGATCAT

An alignment of A and B:

C---TTAAACT  ←  Sequence A
CGGATCA---T  ←  Sequence B
Pairwise Alignment

Sequence A: CTTAACT
Sequence B: CGGATCAT

An alignment of A and B:
Alignment Graph

Sequence A: CTTAACT

Sequence B: CGGATCAT

C G G A T C A T

C---TTAACT
CGGATCA--T
A simple scoring scheme

• Match: +8 \( w(x, y) = 8 \), if \( x = y \)
• Mismatch: -5 \( w(x, y) = -5 \), if \( x \neq y \)
• Each gap symbol: -3 \( w(-, x) = w(x, -) = -3 \)

\[
\begin{align*}
\text{C} & \quad - \quad - \quad \text{T} \quad T \quad T \quad A \quad A \quad C \quad T \\
\text{C} \quad \text{G} \quad \text{G} \quad \text{A} \quad \text{T} \quad \text{C} \quad \text{A} & \quad - \quad - \quad \text{T} \\
+8 & \quad -3 \quad -3 \quad -3 \quad +8 \quad -5 \quad +8 \quad -3 \quad -3 \quad +8 \quad = \quad +12
\end{align*}
\]
An optimal alignment
-- the alignment of maximum score

• Let $A=a_1a_2\ldots a_m$ and $B=b_1b_2\ldots b_n$.

• $S_{i,j}$: the score of an optimal alignment between $a_1a_2\ldots a_i$ and $b_1b_2\ldots b_j$

• With proper initializations, $S_{i,j}$ can be computed as follows.

\[
S_{i,j} = \max\left\{ 
S_{i-1,j} + w(a_i, -), 
S_{i,j-1} + w(-, b_j), 
S_{i-1,j-1} + w(a_i, b_j)
\right\}
\]
Computing $S_{i,j}$
## Initializations

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>G</th>
<th>G</th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>A</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-3</td>
<td>-6</td>
<td>-9</td>
<td>-12</td>
<td>-15</td>
<td>-18</td>
<td>-21</td>
<td>-24</td>
</tr>
<tr>
<td>-3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- C     G     G     A     T     C     A     T

TTAAC
\[ S_{3,5} = ? \]

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>G</th>
<th>G</th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>A</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-3</td>
<td>-6</td>
<td>-9</td>
<td>-12</td>
<td>-15</td>
<td>-18</td>
<td>-21</td>
<td>-24</td>
</tr>
<tr>
<td>-3</td>
<td>8</td>
<td>5</td>
<td>2</td>
<td>-1</td>
<td>-4</td>
<td>-7</td>
<td>-10</td>
<td>-13</td>
</tr>
<tr>
<td>-6</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>-3</td>
<td>7</td>
<td>4</td>
<td>1</td>
<td>-2</td>
</tr>
<tr>
<td>-9</td>
<td>2</td>
<td>0</td>
<td>-2</td>
<td>-5</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The optimal score for the alignment is \( S_{3,5} = 14 \).
C T T A A C − T
C G G A T C A T
8 − 5 −5 +8 −5 +8 −3 +8 = 14

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>G</th>
<th>G</th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>A</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>0</td>
<td>-3</td>
<td>-6</td>
<td>-9</td>
<td>-12</td>
<td>-15</td>
<td>-18</td>
<td>-21</td>
</tr>
<tr>
<td>T</td>
<td>-3</td>
<td>8</td>
<td>5</td>
<td>2</td>
<td>-1</td>
<td>-4</td>
<td>-7</td>
<td>-10</td>
</tr>
<tr>
<td>T</td>
<td>-6</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>-3</td>
<td>7</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>-9</td>
<td>2</td>
<td>0</td>
<td>-2</td>
<td>-5</td>
<td>5</td>
<td>-1</td>
<td>-4</td>
</tr>
<tr>
<td>A</td>
<td>-12</td>
<td>-1</td>
<td>-3</td>
<td>-5</td>
<td>6</td>
<td>3</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>A</td>
<td>-15</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>3</td>
<td>1</td>
<td>-2</td>
<td>8</td>
</tr>
<tr>
<td>C</td>
<td>-18</td>
<td>-7</td>
<td>-9</td>
<td>-11</td>
<td>0</td>
<td>-2</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>T</td>
<td>-21</td>
<td>-10</td>
<td>-12</td>
<td>-14</td>
<td>-3</td>
<td>8</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>
Global Alignment vs. Local Alignment

- **global alignment:**

- **local alignment:**
An optimal local alignment

- $S_{i,j}$: the score of an optimal local alignment ending at $a_i$ and $b_j$
- With proper initializations, $S_{i,j}$ can be computed as follows.

$$s_{i,j} = \max \begin{cases} 0 \\ s_{i-1,j} + w(a_i, -) \\ s_{i,j-1} + w(-, b_j) \\ s_{i-1,j-1} + w(a_i, b_j) \end{cases}$$
**local alignment**

- **Match:** 8
- **Mismatch:** -5
- **Gap symbol:** -3

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>G</th>
<th>G</th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>A</th>
<th>T</th>
</tr>
</thead>
<tbody>
<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>C</td>
<td>0</td>
<td>8</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>5</td>
<td>3</td>
<td>?</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**local alignment**

- **Match:** 8
- **Mismatch:** -5
- **Gap symbol:** -3

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>G</th>
<th>G</th>
<th>A</th>
<th>T</th>
<th>C</th>
<th>A</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</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>T</td>
<td>0</td>
<td>8</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>5</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>5</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>8</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td>T</td>
<td>0</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>13</td>
<td>10</td>
<td>8</td>
</tr>
</tbody>
</table>

The best score is 18.
The best score is 18.

The alignment is as follows:

**A - C - T**

**A T C A T**

8–3+8–3+8 = 18
Affine gap penalties

- Match: +8 \( w(x, y) = 8 \), if \( x = y \)
- Mismatch: -5 \( w(x, y) = -5 \), if \( x \neq y \)
- Each gap symbol: -3 \( w(-,x)=w(x,-)=-3 \)
- Each gap is charged an extra gap-open penalty: -4.

\[
\begin{array}{cccccccccccc}
\text{C} & \text{---} & \text{T} & \text{T} & \text{A} & \text{A} & \text{C} & \text{T} \\
\text{C} & \text{G} & \text{G} & \text{A} & \text{T} & \text{C} & \text{A} & \text{---} & \text{T} \\
\end{array}
\]

\[
\begin{array}{cccccccccccc}
+8 & -3 & -3 & -3 & +8 & -5 & +8 & -3 & -3 & +8 \\
\end{array}
\]

Alignment score: 12 – 4 – 4 = 4
Affine gap penalties

• A gap of length $k$ is penalized $x + k \cdot y$.

Three cases for alignment endings:

1. \ldots \text{x} \ldots \text{x} an aligned pair
2. \ldots \text{x} \ldots \text{−} a deletion
3. \ldots \text{−} \ldots \text{x} an insertion
Affine gap penalties

- Let $D(i, j)$ denote the maximum score of any alignment between $a_1a_2...a_i$ and $b_1b_2...b_j$ ending with a deletion.
- Let $I(i, j)$ denote the maximum score of any alignment between $a_1a_2...a_i$ and $b_1b_2...b_j$ ending with an insertion.
- Let $S(i, j)$ denote the maximum score of any alignment between $a_1a_2...a_i$ and $b_1b_2...b_j$. 
Affine gap penalties

\[ D(i, j) = \max \begin{cases} 
D(i - 1, j) - y \\
S(i - 1, j) - x - y 
\end{cases} \]

\[ I(i, j) = \max \begin{cases} 
I(i, j - 1) - y \\
S(i, j - 1) - x - y 
\end{cases} \]

\[ S(i, j) = \max \begin{cases} 
S(i - 1, j - 1) + w(a_i, b_j) \\
D(i, j) \\
I(i, j) 
\end{cases} \]
Affine gap penalties
$k$ best local alignments

- **Smith-Waterman**
  (Smith and Waterman, 1981; Waterman and Eggert, 1987)

- **FASTA**
  (Wilbur and Lipman, 1983; Lipman and Pearson, 1985)

- **BLAST**
  (Altschul et al., 1990; Altschul et al., 1997)
$k$ best local alignments

- **Smith-Waterman**
  (Smith and Waterman, 1981; Waterman and Eggert, 1987)
  - linear-space version: sim (Huang and Miller, 1991)
  - linear-space variants: sim2 (Chao et al., 1995); sim3 (Chao et al., 1997)

- **FASTA**
  (Wilbur and Lipman, 1983; Lipman and Pearson, 1985)
  - linear-space band alignment (Chao et al., 1992)

- **BLAST**
  (Altschul et al., 1990; Altschul et al., 1997)
  - restricted affine gap penalties (Chao, 1999)
FASTA

1) Find runs of identities, and identify regions with the highest density of identities.

2) Re-score using PAM matrix, and keep top scoring segments.

3) Eliminate segments that are unlikely to be part of the alignment.

4) Optimize the alignment in a band.
FASTA

Step 1: Find runes of identities, and identify regions with the highest density of identities.
FASTA

Step 2: Re-score using PAM matrix, and keep top scoring segments.
FASTA

Step 3: Eliminate segments that are unlikely to be part of the alignment.
FASTA

Step 4: Optimize the alignment in a band.
BLAST

1) Build the hash table for Sequence A.
2) Scan Sequence B for hits.
3) Extend hits.
**BLAST**

Step 1: Build the hash table for Sequence A. (3-tuple example)

For DNA sequences:

Seq. A = AGATCGAT

<table>
<thead>
<tr>
<th>12345678</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA</td>
</tr>
<tr>
<td>AAC</td>
</tr>
<tr>
<td>..</td>
</tr>
<tr>
<td>AGA</td>
</tr>
<tr>
<td>..</td>
</tr>
<tr>
<td>ATC</td>
</tr>
<tr>
<td>..</td>
</tr>
<tr>
<td>CGA</td>
</tr>
<tr>
<td>..</td>
</tr>
<tr>
<td>GAT</td>
</tr>
<tr>
<td>..</td>
</tr>
<tr>
<td>TCG</td>
</tr>
<tr>
<td>..</td>
</tr>
<tr>
<td>TTT</td>
</tr>
</tbody>
</table>

For protein sequences:

Seq. A = ELVIS

Add $xyz$ to the hash table if $\text{Score}(xyz, ELV) \geq T$;
Add $xyz$ to the hash table if $\text{Score}(xyz, LVI) \geq T$;
Add $xyz$ to the hash table if $\text{Score}(xyz, VIS) \geq T$;
BLAST

Step 2: Scan sequence B for hits.
Step 2: Scan sequence B for hits.

---

Step 3: Extend hits.

Terminate if the score of the extension fades away.

BLAST 2.0 saves the time spent in extension, and considers gapped alignments.
Remarks

• Filtering is based on the observation that a good alignment usually includes short identical or very similar fragments.
• The idea of filtration was used in both FASTA and BLAST.
Linear-space ideas
Hirschberg, 1975; Myers and Miller, 1988
Two subproblems
½ original problem size

Diagram:

- Two subproblems
- Half of the original problem size
- m/2 and m/4
- 3m/4
Four subproblems

$\frac{1}{4}$ original problem size
Time and Space Complexity

- Space: $O(M+N)$
- Time:
  
  $$O(MN) \times (1 + \frac{1}{2} + \frac{1}{4} + \ldots) = O(MN)$$
Band Alignment
(Joint work with W. Pearson and W. Miller)

Sequence A

Sequence B
Band Alignment in Linear Space

The remaining subproblems are no longer only half of the original problem. In worst case, this could cause an additional $\log n$ factor in time.
Band Alignment in Linear Space
Multiple sequence alignment (MSA)

- The multiple sequence alignment problem is to simultaneously align more than two sequences.

| Seq1: GCTC | GC–TC |
| Seq2: AC   | A----C |
| Seq3: GATC | G–ATC |
How to score an MSA?

- *Sum-of-Pairs (SP-score)*

\[
\text{Score} \left( \begin{array}{c} 
\text{GC-TC} \\
\text{A---C} \\
\text{G-ATC} 
\end{array} \right) + \text{Score} \left( \begin{array}{c} 
\text{GC-TC} \\
\text{G-ATC} \\
\text{A---C} 
\end{array} \right) + \text{Score} \left( \begin{array}{c} 
\text{GC-TC} \\
\text{G-ATC} \\
\text{A---C} 
\end{array} \right)
\]
MSA for three sequences

• an $O(n^3)$ algorithm
General MSA

- For $k$ sequences of length $n$: $O(n^k)$
- NP-Complete (Wang and Jiang)
- The exact multiple alignment algorithms for many sequences are not feasible.
- Some approximation algorithms are given. (e.g., $2^{-l/k}$ for any fixed $l$ by Bafna et al.)
Progressive alignment

- A heuristic approach proposed by Feng and Doolittle.
- It iteratively merges the most similar pairs.
- “Once a gap, always a gap”

The time for progressive alignment in most cases is roughly the order of the time for computing all pairwise alignment, i.e., $O(k^2n^2)$.
Concluding remarks

• Three essential components of the dynamic-programming approach:
  – the recurrence relation
  – the tabular computation
  – the traceback

• The dynamic-programming approach has been used in a vast number of computational problems in bioinformatics.