CSE 421
Intro to Algorithms
Winter 2001

Sequence Alignment

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

Sequence Similarity: What

G G A C C A

T A C T A A G

|:|:|:|:

T C C - A A T

Sequence Similarity: Why

● Diff
● RCS
● Molecular Bio

Similar sequences often have similar origin
or function

Similarity often recognizable after $10^8$ – $10^9$ years

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

a c b c d b a c -- b c d b
c a d b d -- c a d b -- d --

Defn: An alignment of strings S, T is a pair of strings S', T' (with spaces) s.t.
(1) |S'| = |T'|, and (|S| = “length of S”)
(2) removing all spaces leaves S, T
6.8: "Min_Edit_Distance"

<table>
<thead>
<tr>
<th>Original State</th>
<th>Edit Operation</th>
<th>Modified State</th>
</tr>
</thead>
<tbody>
<tr>
<td>a b b c</td>
<td>delete a</td>
<td>b b c</td>
</tr>
<tr>
<td>b b c</td>
<td>insert a</td>
<td>b a b c</td>
</tr>
<tr>
<td>b a b c</td>
<td>replace c</td>
<td>b a b b</td>
</tr>
<tr>
<td>b a b b</td>
<td>delete c</td>
<td>-</td>
</tr>
</tbody>
</table>

- a b b c
gen 3
- b a b b
gen 2

Costs:
- delete 3
- insert 2
- replace 2

Alignment Scoring

- a c b d b
- c a d b d
- -1 2 -1 2 -1

Value = 3*2 + 5*(-1) = +1

- The score of aligning (characters or spaces) x & y is $\sigma(x, y)$.
- Value of an alignment $= \sum_i \sigma(S[i], T[i])$
- An optimal alignment: one of max value

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

S = abcd
A = cd
T = wxyz
B = xz
-abc-d
da-bc-d
w--xyz
-y-xyz

Analysis

- Assume |S| = |T| = n
- Cost of evaluating one alignment: $\geq n$
- How many alignments are there: $\geq \left(\begin{array}{c}2n \\ n \end{array}\right)$
  - 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: $\geq n \left(\begin{array}{c}2n \\ n \end{array}\right) > 2^n$, for $n > 3$
- E.g., for $n = 20$, time is $> 2^{40}$ operations

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).

Polynomial vs Exponential Growth

- $2^n$
- $2^{n/10}$
- $1000n^2$

- $10^2$
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, \(\sigma(-,-) \geq 0\))
- 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:

\[
V(i,j) = \text{value of optimal alignment of } S[1], \ldots, S[i] \text{ with } T[1], \ldots, T[j]
\]
for all \(0 \leq i \leq n, 0 \leq j \leq m\).

General Case

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

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

Base Cases

- \(V(i, 0): \text{first } i \text{ chars of } S; \text{all match spaces}\)
  \[V(i, 0) = \sum_{k=1}^{i} \sigma(S[k], -)\]
- \(V(0, j): \text{first } j \text{ chars of } T; \text{all match spaces}\)
  \[V(0, j) = \sum_{k=1}^{j} \sigma(-, T[k])\]

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></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>a</td>
<td>-1</td>
<td>-2</td>
<td>-3</td>
<td>-4</td>
<td>-5</td>
</tr>
<tr>
<td>1</td>
<td>b</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>
</tr>
<tr>
<td>3</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>
</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>

Calculating One Entry

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

Mismatch = -1
Match = 2

Time = \(O(mn)\)
### Finding Alignments: Trace Back

<table>
<thead>
<tr>
<th>i</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>0</td>
<td>0</td>
<td>-3</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>0</td>
<td>-1</td>
<td>-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td>-2</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>b</td>
<td>-3</td>
<td>0</td>
<td>-1</td>
<td>2</td>
<td>1</td>
<td></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></td>
</tr>
<tr>
<td>5</td>
<td>d</td>
<td>-5</td>
<td>-2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>b</td>
<td>-6</td>
<td>-3</td>
<td>0</td>
<td>3</td>
<td>-2</td>
<td></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.