CSE 421
Intro to Algorithms
Summer 2004

Sequence Alignment

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

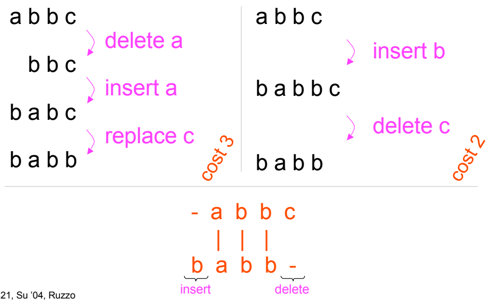
```
a c b c d b       a c – – b c d b
  /   \   |         |     |   |
c a d b d         – c a d b – d –
```

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”



a b b c
delete a
b b c
insert a
b a b c
replace c
b a b b
cost 3

a b b c
insert b
b a b b c
delete c
b a b b
cost 2

```
  - a b b c
    | | |
  b a b b -
```

insert

delete

Alignment Scoring

Mismatch = -1
Match = 2

```
a c b c d b      a  c  -  -  b  c  d  b
c a d b d        -  c  a  d  b  -  d  -
                 -1 2  -1 -1 2  -1 2  -1
                 Value = 3*2 + 5*(-1) = +1
```

- The *score* of aligning (characters or spaces) x & y is σ(x,y).
- *Value* of an alignment $= \sum_{i=1}^{|S'|} \sigma(S'[i], T'[i])$
- 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 ≤ i ≤ |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   a-bc-d
w--xyz   -w-xyz
```

Analysis

- Assume |S| = |T| = n
- Cost of evaluating one alignment: ≥ 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 *un*picked 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

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; σ(–, –) < 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) = value of optimal alignment of S[1], …, S[i] with T[1], …, T[j] for all $0 \le i \le n$, $0 \le j \le m$.

General Case

Opt align of S[1], …, S[i] vs T[1], …, T[j]:

$$\begin{bmatrix} \sim\sim\sim\sim & S[i] \\ \sim\sim\sim\sim & T[j] \end{bmatrix}, \quad \begin{bmatrix} \sim\sim\sim\sim & S[i] \\ \sim\sim\sim\sim & - \end{bmatrix}, \text{ or } \begin{bmatrix} \sim\sim\sim\sim & - \\ \sim\sim\sim\sim & T[j] \end{bmatrix}$$

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

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

for all $1 \le i \le n, \; 1 \le j \le m.$

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

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

Example

Mismatch = -1
Match = 2

j		0	1	2	3	4	5
i			c	a	d	b	d
0		0	-1	-2	-3	-4	-5
1	a	-1	-1	1			
2	c	-2	1				
3	b	-3					
4	c	-4					
5	d	-5					
6	b	-6					

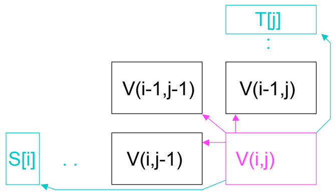
←T

↑S

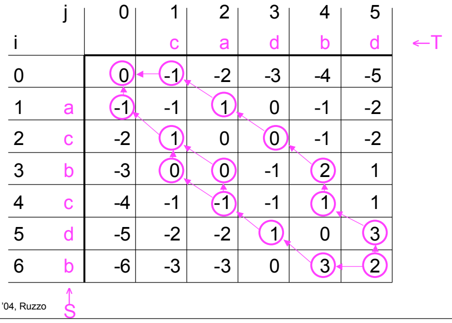
Time = O(mn)

Calculating One Entry

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



Finding Alignments: Trace Back



j		0	1	2	3	4	5
i			c	a	d	b	d
0		0	-1	-2	-3	-4	-5
1	a	-1	-1	1	0	-1	-2
2	c	-2	1	0	0	-1	-2
3	b	-3	0	0	-1	2	1
4	c	-4	-1	-1	-1	1	1
5	d	-5	-2	-2	1	0	3
6	b	-6	-3	-3	0	3	2

←T

↑S

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.