Algorithms: Design and Analysis, Part II

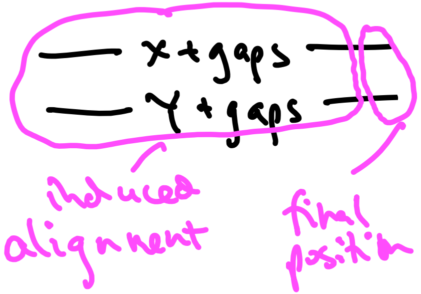
Dynamic Programming

An Algorithm for Sequence Alignment

The Subproblems

① x_m & y_n
② x_m & gap
③ y_n & gap

Optimal substructure: Let $X' = X - x_m$, $Y' = Y - y_n$.



If case ① holds, then induced alignment of X' & Y' is optimal.

If case ② holds, then induced alignment of X' & Y is optimal.

If case ③ holds, then induced alignment of X & Y' is optimal.

Relevant subproblems: have the form (X_i, Y_j), where

X_i = 1st i letters of X

Y_j = 1st j letters of Y

[Since only peel off letters from the right ends of the strings]

The Recurrence

Notation: P_{ij} = penalty of optimal alignment of X_i & Y_j.

Recurrence: for all $i = 1,2,3,\dots,m$ and $j = 1,2,3,\dots,n$:

$$P_{ij} = \min \begin{cases} (1)\ \alpha_{x_i y_j} + P_{(i-1),(j-1)} \\ (2)\ \alpha_{gap} + P_{(i-1),j} \\ (3)\ \alpha_{gap} + P_{i,(j-1)} \end{cases}$$

Correctness: optimal solution is one of these 3 candidates, and recurrence selects the best of these.

Base Cases

Question: what is the value of $P_{i,0}$ and $P_{0,i}$?

(A) 0

(B) $i \cdot \alpha_{gap}$

(C) $+\infty$

(D) undefined

The Algorithm

A = 2-D array.

$A[i,0] = A[0,i] = i \cdot \alpha_{gap} \quad \forall i \geq 0$

For $i = 1$ to m

  For $j = 1$ to n

$$A[i,j] = \min \begin{cases} (1)\ A[i-1,j-1] + \alpha_{x_i y_j} \\ (2)\ A[i-1,j] + \alpha_{gap} \\ (3)\ A[i,j-1] + \alpha_{gap} \end{cases}$$

all available for $O(1)$-time lookup!

Correctness:

[i.e., $A[i,j] = P_{ij}\ \forall i,j \geq 0$]

Follows from induction + correctness of recurrence.

Running Time:

$O(mn)$

[$O(1)$ work for each of $O(mn)$ subproblems]

Reconstructing A Solution

- trace back through filled-in table A, starting at A[m,n]
- when you reach subproblem A[i,j]:
  - if A[i,j] filled using case (1), match x_i & y_j and go to A[i-1,j-1]
  - if A[i,j] filled in using case (2), match x_i with a gap and go to A[i-1,j]
  - if A[i,j] filled in using case (3), match y_j with a gap and go to A[i,j-1]

[if i=0 or j=0, match remaining substring with gaps]

Running time is only O(m+n)!

Tim Roughgarden