Sequence Alignment

Longest Common Subsequence

Given two strings, x of length m and y of length n, find a common non-contiguous subsequence that is as long as possible.

What is the complexity of the naive algorithm?
How can we make this efficient?

Longest Common Subsequence

Given two strings, x of length m and y of length n, find a common non-contiguous subsequence that is as long as possible.

What is the complexity of the naive algorithm?

How can we make this efficient?

Longest Common Subsequence

Given two strings, x of length m and y of length n, find a common non-contiguous subsequence that is as long as possible.

What is the complexity of the naive algorithm?
How can we make this efficient?

Recursive Approach

$LCS(i,j)$ means length of LCS only considering up to x_i and y_j

$$LCS(i,j) = \begin{cases} 0, & \text{if } i = 0 \text{ or } j = 0 \\ LCS(i-1,j-1)+1, & \text{if } x_i = y_j \\ max(LCS(i-1,j), LCS(i,j-1)) & \text{otherwise} \end{cases}$$

Optimal Substructure

Prove global optimum uses optimal solutions of subproblems:

Let z be an $LCS(i,j)$ of length k
What if subproblem of optimal solution were not optimal?
Three cases:

1. If $x_i = y_j$, then $z_k = x_i = y_j$ and $LCS(i-1, j-1) = z_0..z_{k-1}$.
   Not including z_k makes LCS suboptimal: contradiction!
   If $z_0..z_{k-1}$ were not LCS, z could be longer, so not optimal: contradiction!
2. If $x_i \neq y_j$ and $z_k \neq x_i$ then z is $LCS(i-1, j)$
   If longer exists, z would not be LCS: contradiction!
3. If $x_i \neq y_j$ and $z_k \neq y_j$ then z is $LCS(i, j-1)$
   Similar case to 2

Optimal Substructure

Prove global optimum uses optimal solutions of subproblems:

Let z be an $LCS(i,j)$ of length k
What if subproblem of optimal solution were not optimal?
Three cases:

1. If $x_i = y_j$, then $z_k = x_i = y_j$ and $LCS(i-1, j-1) = z_0..z_{k-1}$.
   Not including z_k makes LCS suboptimal: contradiction!
   If $z_0..z_{k-1}$ were not LCS, z could be longer, so not optimal: contradiction!
2. If $x_i \neq y_j$ and $z_k \neq x_i$ then z is $LCS(i-1, j)$
   If longer exists, z would not be LCS: contradiction!
3. If $x_i \neq y_j$ and $z_k \neq y_j$ then z is $LCS(i, j-1)$
   Similar case to 2

Optimal Substructure

Prove global optimum uses optimal solutions of subproblems:

Let z be an $LCS(i,j)$ of length k
What if subproblem of optimal solution were not optimal?
Three cases:

1. If $x_i = y_j$, then $z_k = x_i = y_j$ and $LCS(i-1, j-1) = z_0..z_{k-1}$.
   Not including z_k makes LCS suboptimal: contradiction!
   If $z_0..z_{k-1}$ were not LCS, z could be longer, so not optimal: contradiction!
2. If $x_i \neq y_j$ and $z_k \neq x_i$ then z is $LCS(i-1, j)$
   If longer exists, z would not be LCS: contradiction!
3. If $x_i \neq y_j$ and $z_k \neq y_j$ then z is $LCS(i, j-1)$
   Similar case to 2

Proof by Induction

Proof structure:

- Prove base case is true ($i, j = 1$)
- Assume true for $i - 1$ and $j - 1$, prove true for i and j

If $i, j = 1$, then x_{i-1} and y_{j-1} are empty strings, so trivially true

We showed the second part of the proof already

Therefore, for any i, j, the subproblem of an optimal solution is optimal

Summary of Dynamic Programming

1. optimal substructure: global optimum uses optimal solutions of subproblems
2. ordering of subproblems: solve 'smallest' first, build larger solutions from smaller
3. 'overlapping' subproblems: polynomial number of subproblems, used multiple times
4. independent subproblems: optimal solution of one subproblem doesn't affect optimality of another

- top-down: memoization
- bottom-up: compute table, then recover solution