More Dynamic Programming: Longest Common Subsequence

Longest Common Subsequence

Problem Statement

- Given two sequences of symbols, determine the longest common subsequence between the two sequences
  - A subsequence is simply defined as the original sequence with zero or more items left out
  - The elements of the subsequence do not have to occur consecutively – but must maintain their ordering
  - (Same definition as for longest increasing subsequence)

  - Example:
    - hearty
    - hyena

    - LCS: hea

Longest Common Subsequence

Applications – Computational Biology

- DNA sequence matching
  - Determine the similarity of two different gene sequences
  - Each sequence is a string of DNA bases (A,C,G,T)
    - Used to predict evolutionary history of species (phylogenetic trees)
  - Many possible metrics for measuring similarity
    - One DNA sequence is a substring of another
    - Compute the number of changes needed to turn one string into the other (we will look at sequence alignment next class)
    - Compute the longest common subsequence

Longest Common Subsequence

Brute Force Search

- Consider all possible subsequences of one of the sequences
  - For length n sequence, there are Power Set = 2^n subsequences

Dynamic Programming Solution

- We need to define an optimal subproblem S
  - We've seen examples with one (S[n]) or two (S[m][n] variables)
  - What does each element in the matrix represent?

- And we need to define a recurrence relation that connects the solution to the larger subproblem to the solution of one or more smaller subproblems

Longest Common Subsequence

Dynamic Programming

- Given sequences $X = \{x_1, x_2, \ldots, x_m\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$
- S[j][k] := largest common subsequence that can be formed between the first j elements of X and the first k elements of sequence Y
- The overall solution will be S[m][n]
- The recurrence relation:
  - $S[j][k] = \begin{cases} 1 + S[j-1][k-1] \text{ if } x_j = y_k \\ \max(S[j-1][k], S[j][k-1]) \text{ if } x_j \neq y_k \end{cases}$
  - $S[j][k] = 0$ if $j = 0$ or $k = 0$

Longest Common Subsequence

Iterative Solution: "Bottom-Up"

- Initialize S[0][k] = S[j][0] = 0
- Start with S[1][1] and compute each next row

Complexity

- $O(mn)$, where m = |X| and n = |Y| are lengths of the two sequences

```
LONGEST-COMMON-SUBSEQUENCE(X,Y)
1. init S[j][0] to 0 for every j=0,…,|X|
   init S[0][k] to 0 for every k=0,…,|Y|
2. for j=1 to |X| do
3.     for k=1 to |Y| do
4.         S[j][k] = max{ S[j-1][k], S[j][k-1] }
5.         if (X[j] == Y[k]) then
6.             S[j][k] = S[j-1][k-1]+1
7. RETURN S[|X|][|Y|]
```

Longest Common Subsequence

Tracing back to determine the actual sequence

- We can store an additional matrix b[j][k] that indicates the optimal subproblem chosen for that entry
- However, this extra space is unnecessary
  - Each matrix element depends on only three other matrix elements, so the optimal subproblem chosen can be determined in O(1) time
    - If x[j] == y[k], include that letter and trace back diagonally in the matrix.
    - Else, trace back to horizontal / vertical neighbor (whichever has maximum value)
      - Ties can be broken arbitrarily, and thus lead to different equivalent answers
  - O(m+n) to trace back the complete sequence