Dynamic Programming 3 - Longest Common Subsequence

▼ Problem overview

- Given two strings of different lengths (X[1...m] and Y[1...n]), find the longest common subsequence that has the longest possible length.

▼ Recursive solution to the problem

- We're not going to use a recursive function like L(m,n) (where it's the length of the LCS of X[1...m] and Y[1...n]). We need to do this in terms of smaller subsets.
- Let's write a function L(j,k), where the result is the length of the LCS of X[1...j] and Y[1...k] ($0 \leq j \leq m, 0 \leq k \leq n$).
- Algorithm:
  - $L(j,k) = 0$ if $j = 0$ or $k = 0$.
  - $L(j,k) = L(j-1, k-1) + 1$ if $j > 0, k > 0$, and $X[j] = Y[k]$
    - If the two charaters match, take the length of the previous LCS and add one (including the current character).
  - $L(j,k) = max\{\ L(j-1,k), L(j,k-1)\ \}$ if $j > 0, k > 0$ and $X[j] \neq Y[k]$.
    - We're going to throw away the last letter of X and the last letter of Y, figure out which one is the longest, and then use that.
    - This is the case where the LCS doesn't match!!!

▼ Dynamic Programming version

```
allocate array L[0…m][0…n];

for j = 0 to m
      for k = 0 to n
            if (j==0 || k==0)
                  L[j][k] = 0;
            else if (X[j]==Y[k])
                  L[j][k] = L[j–1][k–1] + 1;
            else
                  L[j][k] = max {L[j–1][k], L[j][k–1]};
```

Running time = $\theta(mn)$

- This tracks our solution pretty consistently.