Longest Common Subsequence

- *Longest common subsequence* (*LCS*) problem:

- Given two sequences x[1..m] and y[1..n], find the longest subsequence which occurs in both
- Ex: x = {A B C B D A B }, y = {B D C A B A}
  {B C} and {A A} are both subsequences of both

What is the LCS?
- Brute-force algorithm: For every subsequence of x, check if it's a subsequence of y

  How many subsequences of x are there?
  What will be the running time of the brute-force alg?

LCS Algorithm

- Brute-force algorithm: 2^m subsequences of x to check against n elements of y: $O(n\ 2^m)$
- We can do better: for now, let's only worry about the problem of finding the *length* of LCS
- When finished we will see how to backtrack from this solution back to the actual LCS
- Notice LCS problem has optimal substructure
- Subproblems: LCS of pairs of *prefixes* of x and y

LCS recursive solution

$$c[i,j] = \begin{cases} c[i-1,j-1]+1 & \text{if } x[i]=y[j], \\ \max(c[i,j-1],c[i-1,j]) & \text{otherwise} \end{cases}$$

- We start with $i = j = 0$ (empty substrings of x and y)
- Since X_0 and Y_0 are empty strings, their LCS is always empty (i.e. $c[0,0] = 0$)
- LCS of empty string and any other string is empty, so for every i and j: $c[0, j] = c[i,0] = 0$

LCS Example (2)

ABCB
BDCAB

	j	0	1	2	3	4	5
i		Yj	B	D	C	A	B
0	Xi	0	0	0	0	0	0
1	A	0	0				
2	B	0					
3	C	0					
4	B	0					

if (X_i == Y_j)
c[i,j] = c[i-1,j-1] + 1
else c[i,j] = max(c[i-1,j], c[i,j-1])

ABCB
BDCAB

LCS Example (15)

	j	0	1	2	3	4	5
i		Yj	B	D	C	A	B
0	Xi	0	0	0	0	0	0
1	A	0	0	0	0	1	1
2	B	0	1	1	1	1	2
3	C	0	1	1	2	2	2
4	B	0	1	1	2	2	3

if ($X_i == Y_j$)
c[i,j] = c[i-1,j-1] + 1
else c[i,j] = max(c[i-1,j], c[i,j-1])

LCS Algorithm Running Time

- LCS algorithm calculates the values of each entry of the array c[m,n]
- So what is the running time?

O(m.n)

since each c[i,j] is calculated in constant time, and there are m.n elements in the array

Finding LCS (2)

	j	0	1	2	3	4	5
i		Yj	B	D	C	A	B
0	Xi	0	0	0	0	0	0
1	A	0	0	0	0	1	1
2	B	0	1	1	1	1	2
3	C	0	1	1	2	2	2
4	B	0	1	1	2	2	3

LCS (reversed order): B C B

LCS (straight order):

Optimal Substructure of LCS

$$c[i,j] = \begin{cases} c[i-1, j-1] + 1 & \text{if } x[i] = y[j], \\ \max(c[i, j-1], c[i-1, j]) & \text{otherwise} \end{cases}$$

- Observation 1: Optimal substructure
  A simple recursive algorithm will suffice
  Draw sample recursion tree from c[3,4]
  What will be the depth of the tree?

- Observation 2: Overlapping subproblems

Find some places where we solve the same subproblem more than once