Dynamic Programming

Design and Analysis of Algorithms (CST 226-2)

Lecture 03

Longest Common Subsequence (LCS)

- Longest Common Subsequence (LCS) problem is one of the example problem that can be solved using Dynamic Programming.
- LCS Problem Statement: Given two sequences, find the length of longest subsequence present in both of them.
- A subsequence is a sequence that appears in the same relative order, but not necessarily contiguous.

- example,

“abc”, “abg”, “bdf”, “aeg”, ‘”acefg”, .. etc are subsequences of “abcdefg”. So a string of length n has 2^n different possible subsequences.

- The naive solution for this problem is to generate all subsequences of both given sequences and find the longest matching subsequence.

- Each subsequence corresponds to a subset of the indices. A string of length n has 2^n different possible subsequences. This solution is exponential in terms of time complexity.

- Examples:

1. LCS for input Sequences “ABCDGH” and “AEDFHR” is “ADH” of length 3.
2. LCS for input Sequences “AGGTAB” and “GXTXAYB” is “GTAB” of length 4.

- Applications : Bioinformatics (DNA string comparison).

    Check differences between two files

- DNA string: a sequence of symbols A,C,G,T.

    S=ACCGGTCGAGCTTCGAAT

- Subsequence (of X): is X with some symbols left out.

    Z=CGTC is a subsequence of X=ACGCTAC.

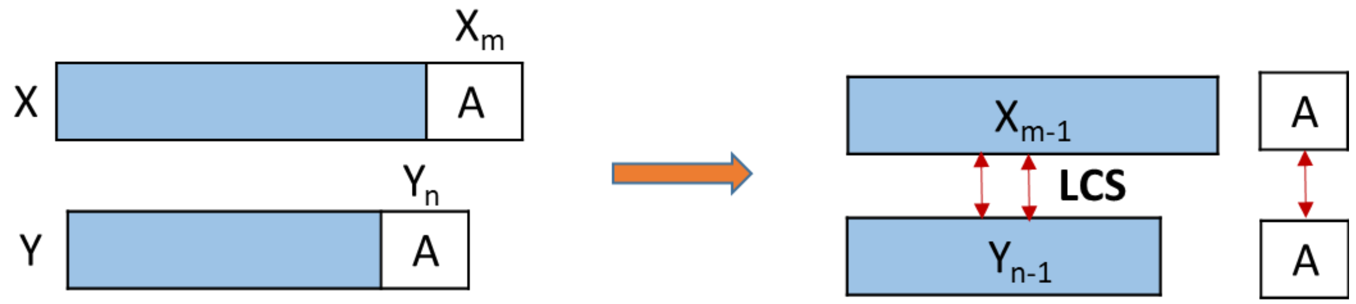
LCS - Optimal Structure Property

- Let X=<x_1, x_2,…, x_m> (= X_m) and Y=<y_1, y_2,…,y_n> (= Y_n) and Z=<z_1, z_2,…, z_k> (= Z_k) be any LCS of X and Y,

i) if $x_m = y_n$, then $z_k = x_m = y_n$, and $Z_{k\text{-}1}$ is the LCS of $X_{m\text{-}1}$ and $Y_{n\text{-}1}$.

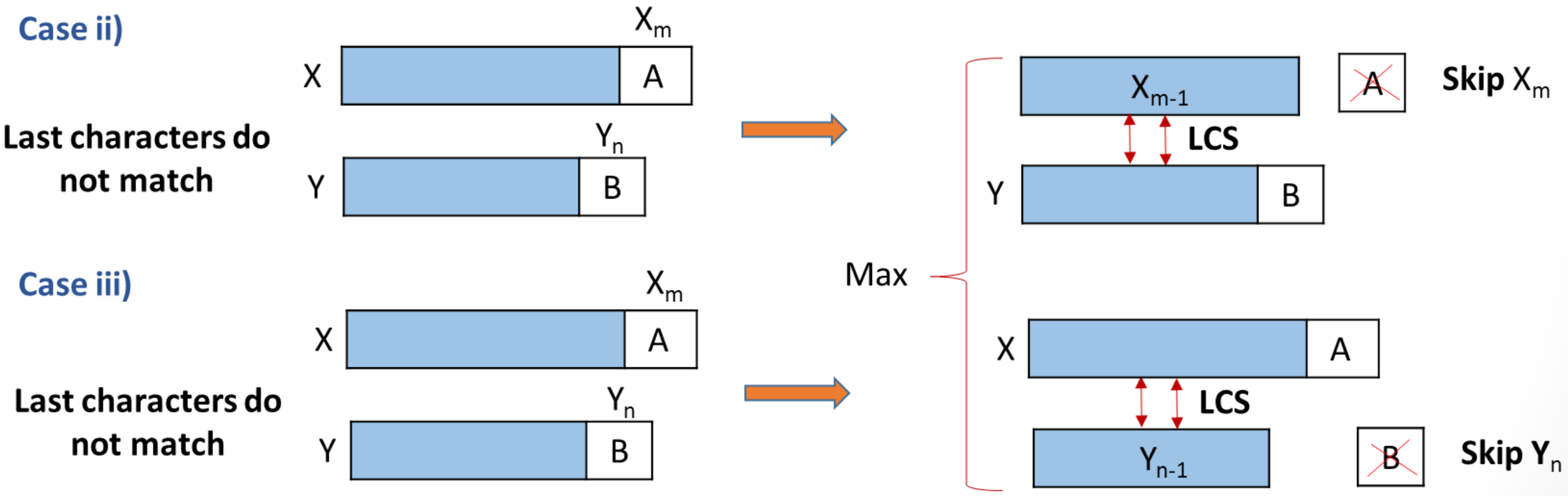
Case i)

Last characters match



ii) if $x_m \neq y_n$, then $z_k \neq x_m$ implies Z is the LCS of $X_{m\text{-}1}$ and Y_n.

iii) if $x_m \neq y_n$, then $z_k \neq y_n$ implies Z is the LCS of X_m and $Y_{n\text{-}1}$.



8/30/2017

Longest Common Subsequences

$$c[m,n] = \begin{cases} 0 & \text{if } m=0, \text{ or } n=0 \\ c[m-1,n-1]+1 & \text{if } m,n > 0 \text{ and } x_m = y_n, \\ \max\{c[m-1,n], c[m,n-1]\} & \text{if } m,n > 0 \text{ and } x_m \neq y_n, \end{cases}$$

This gives a recursive algorithm to solve the problem.

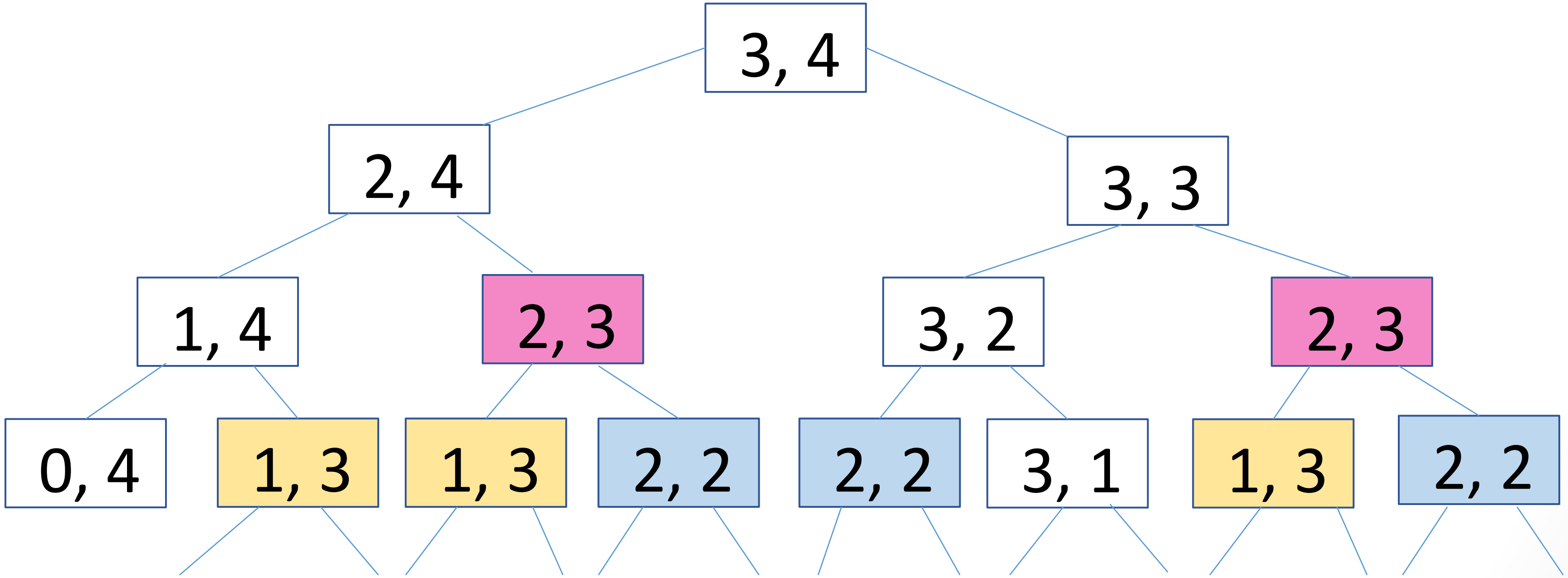
Longest Common Subsequences - Recursive

```
int  lcs( String X, String Y, int m, int n )
{
   if (m == 0 || n == 0)
     return 0;
   if (X[m] == Y[n])
     return 1 + lcs(X, Y, m-1, n-1);
   else
     return max(lcs(X, Y, m, n-1), lcs(X, Y, m-1, n));
}
```

LCS – Overlapping Substructure Property

- A recursive solution contains a “small” number of distinct sub-problems repeated many times.

If the lengths of strings X = 3 and Y = 4 then we get the following recursion tree.



It can be seen from the recursion tree that sub-problems are repeatedly solved.

Longest Common Subsequences - Memoized

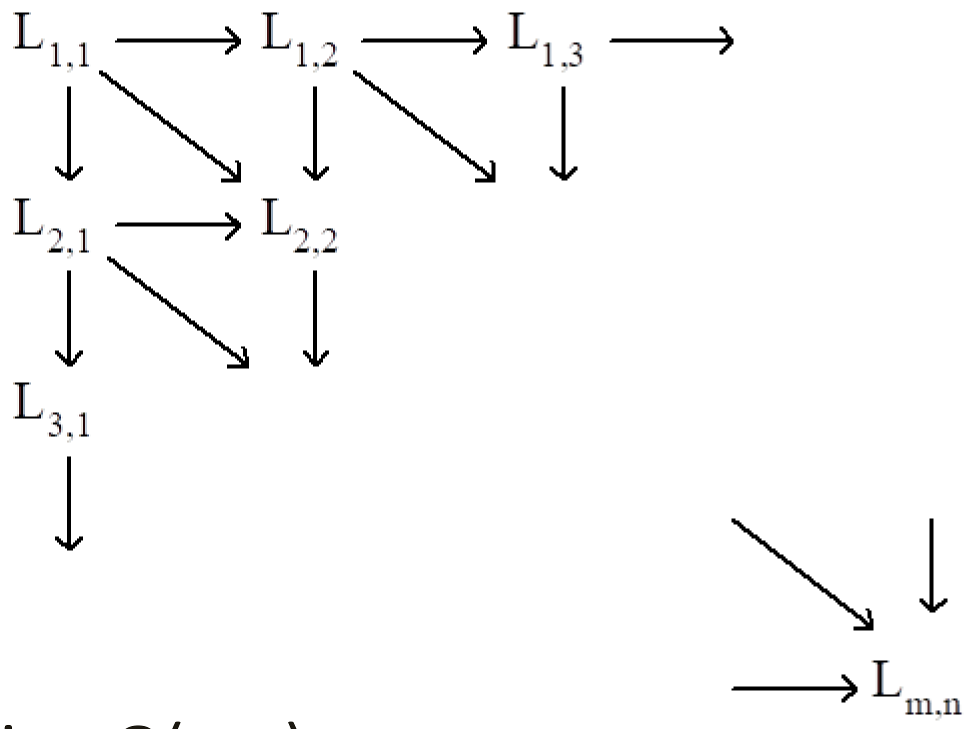
```
int  lcs( String X, String Y, int m, int n )
{
   if (m == 0 || n == 0)
     return 0;
   if (arr[n][m] != NULL) return arr[n][m];
   if (X[m] == Y[n])
      result =1 + lcs(X, Y, m-1, n-1);
    else
      result = max(lcs(X, Y, m, n-1), lcs(X, Y, m-1, n));
    arr[n][m] = result;
}
```

Longest Common Subsequences-Algorithm

```
LCS-LENGTH(X, Y)
m ← length[X]
n ← length[Y]
for i ← 1 to m
        do c[i, 0] ← 0
for j ← 0 to n
        do c[0, j] ← 0
for i ← 1 to m
        do for j ← 1 to n
                do if x_i = y_j
                      then c[i, j] ← c[i − 1, j − 1] + 1
                           b[i, j] ← "↖"
                      else if c[i − 1, j] ≥ c[i, j − 1]
                              then c[i, j] ← c[i − 1, j]
                                   b[i, j] ← "↑"
                              else c[i, j] ← c[i, j − 1]
                                   b[i, j] ← "←"
return c and b
```

Longest Common Subsequences

- The dynamic programming approach for solving the LCS problem:



- Time complexity: O(mn)

- Find the longest common subsequence for

X = A B C B D A B

Y = B D C A B A

- Find the longest common subsequence for

  X = A B C B D A B

  Y = B D C A B A

	j	0	1	2	3	4	5	6
i		y_i	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0	1	1	1	1	2	2
3	C	0	1	1	2	2	2	2
4	B	0	1	1	2	2	3	3
5	D	0	1	2	2	2	3	3
6	A	0	1	2	2	3	3	4
7	B	0	1	2	2	3	4	4

- Find the longest common subsequence for

  X = A B C B D A B

  Y = B D C A B A

	j	0	1	2	3	4	5	6
i		y_i	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0	1	1	1	1	2	2
3	C	0	1	1	2	2	2	2
4	B	0	1	1	2	2	3	3
5	D	0	1	2	2	2	3	3
6	A	0	1	2	2	3	3	4
7	B	0	1	2	2	3	4	4

	j	0	1	2	3	4	5	6
i		y_i	B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	1	1
2	B	0	1	1	1	1	2	2
3	C	0	1	1	2	2	2	2
4	B	0	1	1	2	2	3	3
5	D	0	1	2	2	2	3	3
6	A	0	1	2	2	3	3	4
7	B	0	1	2	2	3	4	4

Approach

If the last characters match :

- LCS[i][j] = LCS[i-1][j-1] + 1

If the last characters do not match :

- LCS[i][j] = max(LCS[i-1][j], LCS[i][j-1])

Exercise 01

1. Find the longest common subsequence for

   A = b a c a d,   B = a c c b a d c b