

## Problem 8-1

(a) Base Case :-

$$BEST[i, i] = K[i], 1 \leq i \leq n.$$

$$\text{So, } BEST[1, 1] = K[1].$$

$$BEST[2, 2] = K[2].$$

$$BEST[n, n] = K[n]$$

Recursive formulation :-

$$BEST[i, j] = \min_{l=i}^j [2 \cdot BEST[i, l-1] + K[l] + 2 \cdot BEST[l+1, j]]$$

$BEST[i, j]$  represents the least cost of binary tree form  $K[i, \dots, j]$ .  
So; we are considering  $l$  as root where  $i \leq l \leq j$ . The cost of left subtree will be  $BEST[i, l-1]$  and the cost of right subtree will be  $BEST[l+1, j]$ . Then we will take the minimum cost of all such binary trees and store it in  $BEST[i, j]$ .

ALGORITHM ( $K[1, \dots, n]$ )

```
for P=1 to n  
  for i=1 to n-P+1  
    j = i + P - 1.  
    if (P >= 1):  
      BEST[i, 1] = K[i]
```

else:  
 $BEST[i, j] = \infty$

for l=i to j:

cont. ->

$$\text{BEST}[i, j] = \min (\text{BEST}[i, j], \text{BEST}[i, l-1] + K[l] + \text{BEST}[l+1, j])$$

Since there are 3 nested loops.

So the runtime is  $O(n^3)$

(ii) Base Case. For 1 to  $n$

$$\text{BEST}[i, j] = K[i], \quad 1 \leq i \leq n. \quad | \quad \text{if } i > j \\ \text{BEST}[i, i] = K[i]$$

Recursive formulation

$\text{BEST}[i, j]$  will give the least cost BT where first node traversal corresponds to  $K[i, \dots, j]$

$$\text{BEST}[i, j] = \min (2 \cdot \text{BEST}[i, l] + 2 \cdot \text{BEST}[l+1, j] + K[l])$$

For calculation of  $\text{BEST}[i, j]$  we are considering  $K[i]$  as the root node as this is a pre-order traversal. Now  $\text{BEST}[i, l]$  represents the min cost of the left subtree and  $\text{BEST}[l+1, j]$  represents the min cost of the right sub-tree.

Algorithm ( $K[1, \dots, m]$ )

for  $P=1$  to  $m$ .

for  $i=1$  to  $m-P+1$ .

$$j = i + P - 1$$

if ( $P=1$ )

$$\text{BEST}[i, i] = K[i]$$

ELSE:

$$\text{BEST}[i, j] = \infty$$

for  $l=0$  to  $j$ :

$$\text{BEST}[i, j] = \min (\text{BEST}[i, j], 2 \cdot \text{BEST}[i, l] + 2 \cdot \text{BEST}[l+1, j] + K[l])$$

Since there are 3 nested loops,  
the runtime of the algorithm is  $O(n^3)$

(a) Base Case  $\rightarrow$

$$BEST[i, i] = K[i]$$

(b) Base Case  $\rightarrow$

$$BEST[i, i] = K[i], \quad 1 \leq i \leq n. \quad | \text{ if } i > j$$

$\rightarrow$  Recursive formulation  $\rightarrow$  As  $K[i, \dots, j]$  is the partnode.  $BEST[i, j] = 0$

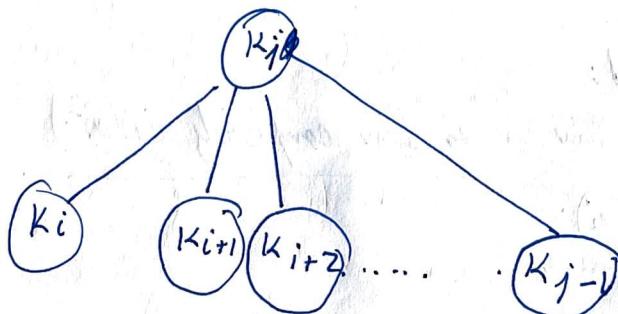
As traversal,  $K[j]$  will be the root.

Now we can calculate the least cost of Subtree by calculating l.

$$BEST[i, j] = \min_{\substack{c=1 \\ i \leq c \leq j}} \left( 2 \sum_{r=c}^j BEST[c, r] \right) + K[j]$$

Since this is a cost of n-nodes arbitrary tree so the optimal solution for  $BEST[i, j]$  will be obtained when  $K[j]$  will be the root and all the other nodes.

$K[i], K[i+1], \dots, K[j-1]$  are its direct children.



ALGORITHM. ( $K[1, \dots, m]$ )

for  $P = 1$  to  $n$

for  $i = 1$  to  $n-P+1$ .

$$j = i + P - 1.$$

if ( $P == 1$ ):

$$\text{BEST}[i:i] = K[i]$$

else:

$$\text{BEST}[i:i] = \text{BEST}[i, j-1] + K[j-1] + K[j]$$

→ Since there are 3 nested loops.

→ So the runtime of the algorithm is  $O(n^3)$

Basically for  $\text{BEST}[i:j]$

$$\boxed{\text{BEST}[i:j] = K[j] + 2[K[i:j] + K[i+1:j] + \dots + K[j-1]]}$$

$$\text{So } \text{BEST}[i, j-1] = K[i-1] + 2[K[i] + K[i+1] + \dots + K[j-2]].$$

$$\text{So } \text{BEST}[i:j] \Rightarrow \text{BEST}[i, j-1] = K[i] + K[j-1]$$

$$\Rightarrow \boxed{\text{BEST}[i, 1] = \text{BEST}[i, j-1] + K[i-1] + K[j]}$$

This way we find the least score arbitrary tree s.t its postorder is  $K[i:j]$

$$\Rightarrow \text{Runtime} = O(n^3).$$

We see 3 loops nested:

Each of them run in time  $n$  so time complexity is  $O(n^3)$ .

Space Complexity is  $O(n^3)$ .

(d) Base Case:

FOR.  $i > j$

$$\text{BEST}[i, j] = 0$$

FOR.  $i = j \& l = 0$

$$\text{BEST}[i, i] = K[i]$$

FOR.  $l > j-i$

$$\text{BEST}[i, j] = 0.$$

Recursive Formulation:

$$\text{BEST}[i, j, l] = \begin{cases} 0; & i > j \text{ or } l > j-i. \\ K[i]; & i = j. \\ \min_{i \leq n < j} \left( \text{BEST}[n+1, j, l-1] + l \cdot \min_{0 \leq y \leq n-i+1} (\text{BEST}[i, n, y]) \right) \end{cases}$$

For all values  $K[i \dots j]$ ,  $K[j]$  will be the root. (Postorder traversal).

We calculate the best score for the leftmost subtree and the more towards the rightmost.

For each root of subtree, we find the best score with  $y$  children where  $0 \leq y \leq n-i+1$  giving  $\text{BEST}[i, n, y]$  multiplied by  $l$ . and adding to  $\text{BEST}[n+1, j, l-1]$  we get the  $\text{BEST}$  score for the tree after minimizing for the  $n$ ,  $i \leq n < j$ .

→ Runtime  $O(n^4)$ .

There are a total of 4 loops.

Space Complexity =  $O(n^3)$

as  $\text{BEST}[i, j, k]$  take 3 dimensions in total.

## Problem\* 8-2.

(a) Cost function will be such that if both characters are same, the cost is  $\textcircled{1}$ , otherwise it will be -1.

$$c(a,a) = \textcircled{1}, a \neq a$$

$$c(a,a) = -1.$$

$$\text{So, } c(m,y) = \begin{cases} \textcircled{1} & m=y \\ -1 & m \neq y \end{cases}$$

Since in the MCS problem, we have to maximize cost of long sub-sequence which is  $\textcircled{1}$  and will be found only when the sequences are LCS.

If the characters do match, it will increase the max. subsequence. will correspond to the ~~max~~ longest common Sub-sequence.

(b) Base Case. Case. :

$$\text{BEST}[i,j] = 0 \text{ if } i=0 \text{ or } j=0.$$

Recursive formulation -

$$\text{BEST}[i,j] = \max (c[x_i, y_j] + \text{BEST}[i-1, j-1], \text{BEST}[i, j-1], \text{BEST}[i-1, j])$$

So, we will have 3 options in the cost.

① First we can take the characters  $x_i$  and  $y_j$  and add their cost to  $\text{BEST}[i-1, j-1]$ .

② The other two options would be to skip either  $x_i$  or  $y_j$ .

Algorithm  $(X[1, \dots, m], Y[1, \dots, m])$

```
for i = 0 to m  
for j = 0 to m  
if i == 0 || j == 0:  
    BEST[i, j] = 0.
```

else:  
 $BEST[i, j] = \max \{ C[x_i, y_j] + BEST[i-1, j-1],$   
 $BEST[i, j-1], BEST[i-1, j] \}$

Since there are 2 nested loops. so the run time of the algorithm will  
be  $\boxed{O(mn)}$   $\Rightarrow O(n^2)$ . (quadratic time).

(c) Dark Case:

$\hookrightarrow BEST[i, j, l] = 0$ , if  $i=0, j=0$  or  $l=0$ .  $BEST[i, j, l]$  the best  
possible cost of sub-sequence of length  $l$  at  $x_i$  &  $y_j$ . Dimension [i,j,l].

Recursive Formulation -

$$BEST[i, j, l] = \max_{\substack{1 \leq k \leq l \\ 1 \leq i \leq m \\ 1 \leq j \leq n}} (C[x_i, y_j] + BEST[i-1, j-1, l-1],  
BEST[i, j-1, l], BEST[i-1, j, l]).$$

Similarly we have 3 choices here.

$\Rightarrow$  If we are choosing  $x_i$  and  $y_j$ ; then we have a subsequence of length  
 $l-1$ . So we need  $BEST[i-1, j-1, l-1]$ .

$\Rightarrow$  Otherwise if we are either skipping  $x_i$  or  $y_j$ ; we have to find the  
subsequence of length  $l$  only.

A LGORITHM. ( $X[1, \dots, m]$ ,  $Y[1, \dots, n, k]$ ).

for  $i = 0$  to  $m$ :

    for  $j = 0$  to  $n$ :

        for  $l = 0$  to  $k$ :

            if  $i = 0$  or  $j = 0$  or  $l = 0$ :

$BEST[i, j, l] = 0$

            else:

$BEST[i, j, l] = \max(C[x_i, y_j] + \cancel{m})$

$BEST[i-1, j-1, l-1], BEST[i, j-1, l], BEST[i-1, j, l])$

$m = \cancel{\infty}$

for  $l = 1$  to  $k$ :

$m = \max(m, BEST[m, m, l] - l^2)$

return  $m$ .

As we can see, Run-time =  $O(mn\lambda)$  for 3 nested loops.

Also Space Complexity =  $O(mn\lambda)$ .

Problem 8-3.

(a) Let  $a$  be some string in the set  $\text{LCS}(x' \parallel a, y' \parallel a)$ .

so  $a \in \text{LCS}(x' \parallel a, y' \parallel a)$

Now, since all string  $a$  will contain  $b$  as the ending character which is common.

So,

$a \in \text{LCS}(x', y') \parallel a$ .

$\therefore \text{LCS}(x' \parallel a, y' \parallel a) \leq \text{LCS}(x', y') \parallel a$ .

Now let  $y$  be one string in the set  $\text{LCS}(x, y') \parallel a$ .

so  $y \in \text{LCS}(x, y') \parallel a$ .

Now, since all string  $y$  need to have a character  $b$  is also appended at the end.

so,  $y \in \text{LCS}(x' \parallel a, y' \parallel a)$

$\therefore \text{LCS}(x, y') \parallel a \leq \text{LCS}(x' \parallel a, y' \parallel a)$

Now, since  $\text{LCS}(x' \parallel a, y' \parallel a) \leq \text{LCS}(x', y') \parallel a$

and  $\text{LCS}(x', y') \parallel a \leq \text{LCS}(x, y') \parallel a$ .

Now,  $\boxed{\text{LCS}(x' \parallel a, y' \parallel a) = \text{LCS}(x', y') \parallel a}$

$\Rightarrow$  we will get all the sub-sequences ending with  $a$  from  $\text{LCS}(x' \parallel a, y' \parallel a)$  which is exactly what we needed.

ALGORITHM. ( $X[1, \dots, m]$ ,  $Y[1, \dots, n, k]$ ).

for  $i = 0$  to  $m$ :

    for  $j = 0$  to  $n$ :

        for  $l = 0$  to  $k$ :

            if  $i = 0$  or  $j = 0$  or  $l = 0$

$$BEST[i, j, l] = 0$$

else.

$$BEST[i, j, l] = \max. (C[x_i, y_j] + \cancel{BEST})$$

$$BEST[i-1, j-1, l-1], BEST[i, j-1, l], BEST[i-1, j, l])$$

$$m = -\infty$$

for  $l = 1$  to  $k$ :

$$m = \max(m, BEST[m, m, l] - l^2)$$

return  $m$  as Cost.

As we can see, Run-time =  $O(mn^k)$  for 3 nested loops.

Also Space Complexity =  $O(mn^k)$ .

(b) Recursion Formula.

$$\left. \begin{array}{l} \text{if } x_i = y_j: \\ L[i,j] = L[i-1,j-1] \cup x_i \end{array} \right\} : \begin{array}{l} \text{if } i=0 \text{ or } j=0 \\ \text{or } x_i = y_j. \end{array}$$

else:

$$\text{if } c[i-1,j] > c[i,j-1]$$

$$L[i,j] = L[i-1,j]$$

$$\cdot \text{ else. if } c[i-1,j] < c[i,j-1]$$

$$L[i,j] = L[i,j-1]$$

$$\text{else. } L[i,j] = L[i-1,j] \cup L[i,j-1]. \quad (\text{for all other cases})$$

So, if  $x_i$  and  $y_j$  are same. Then the set  $L[i,j]$  will have  $x_i$  appended to  $L[i-1,j-1]$ .

otherwise if  $c[i-1,j]$  is greater then  $L[i,j]$  will be the same as  $L[i-1,j]$ .

if  $c[i-1,j]$  is less than  $c[i,j-1]$

then  $L[i,j]$  will be same as  $L[i,j-1]$ .

and finally if  $c[i-1,j]$  equal  $c[i,j-1]$

then  $L[i,j]$  needs to include both  $L[i-1,j]$  and  $L[i,j-1]$ .

### (c) Basis of induction

For  $n=1$ .

$$x_3 = 012.$$

$$y_3 = 102.$$

$$\text{LCS}(x_3, y_3) = \{0, 1, 2\}$$

$$\text{so } \lambda[3,3] = 3 > 2^1 = 2.$$

$$\text{so } \lambda[3m, 3m] = \Omega(2^m) \text{ for } m=1.$$

### Induction Hypothesis

Let  $\lambda[3m, 3m] = \Omega(2^m)$  be true for  $x \& y$  of length  $3m$

### Induction Step -

$$x_{3(m+1)} = x_{3m+3} = x_{3m} x_3 = x_{3m} 012.$$

$$y_{3(m+1)} = y_{3m+3} = y_{3m} y_3 = y_{3m} \cancel{102} 1$$

The LCS of  $x_{3m}$  and  $y_{3m}$  is of length  $2m$  for  $m \neq 1$ .

LCS always starts with 0 or 1. and always end with 2.

$$\text{so, } \lambda[3(m+1), 3(m+1)] = 2 \times \Omega(2^m)$$

$$\lambda[3(m+1), 3(m+1)] = \Omega(2^{m+2})$$

So the induction Justifies that

$$\boxed{\lambda[3m, 3m] = \Omega(2^m)}$$

This is justified from hypothesis

(e) When  $x[i] = y[j]$

$$\text{count}[i, j] = \text{count}[i-1, j-1]$$

If  $x_i$  and  $y_j$  are same, then the LCS will be  $x_i$  appended to LCS of  $x_{i-1}, y_{j-1}$ .

So that the count of LCS for  $x_i, y_j$  will be same as count of LCS for  $x_{i-1}, y_{j-1}$ .

when  $x[i] \neq y[j]$ :

$$\text{if } c[i, j-1] = c[i, j] = c[i-1, j]:$$

$$\text{count}[i, j] = \text{count}[i-1, j] + \text{count}[i, j-1]$$

$$\text{if } c[i, j] = c[i-1, j-1]:$$

$$\text{count}[i, j] = \text{count}[i, j] - \text{count}[i-1, j-1]$$

$$\text{else if } c[i-1, j] = c[i, j];$$

$$\text{count}[i, j] = \text{count}[i-1, j]$$

$$\text{else: } \text{count}[i, j] = \text{count}[i, j-1]$$

Again we have 3 cases that arise.

If  $c[i, j-1], c[i, j]$  and  $c[i-1, j]$  are same then  $\text{count}[i, j]$  will be the sum of  $\text{count}[i-1, j]$  and  $\text{count}[i, j-1]$ .

But if  $c[i, j]$  is also same as  $c[i-1, j-1]$  in this case we need to reduce  $\text{count}[i-1, j-1]$  from  $\text{count}[i, j]$ .

For the remaining two cases if  $c[i, j]$  equals  $c[i-1, j]$ ,  $\text{count}[i, j]$  will be equal to  $\text{count}[i-1, j]$ , otherwise in the last case it will be equal to  $\text{count}[i, j-1]$ .

ALGORITHM. ( $X[1, \dots, m], Y[1, \dots, n]$ ):

for  $i = 0$  to  $m$ :

    for  $j = 0$  to  $n$ :

        if  $i = 0$  or  $j = 0$ :

$\text{count}[i, j] = 0$

        else :

            if  $X[i] == Y[j]$ :

$\text{count}[i, j] = \text{count}[i-1, j-1]$

            else :

                if  $c[i, j] = c[i-1, j] = c[i, j-1]$ :

$\text{count}[i, j] = \text{count}[i, j-1] + \text{count}[i-1, j]$

                if  $c[i, j] = c[i-1, j-1]$ :

$\text{count}[i, j] = \text{count}[i, j] - \text{count}[i-1, j-1]$

                else if  $c[i-1, j] = c[i, j]$ :

$\text{count}[i, j] = \text{count}[i-1, j]$

                else :

$\text{count}[i, j] = \text{count}[i, j-1]$

Time complexity =  $O(m, n)$ .

(b) Comparison and addition/Subtraction takes  $\Theta(i)$  where  $i$  is the number of bits.

$$\text{Now, } m = 2^i$$

$$i = \log_2 m$$

so, the addition, comparison, etc. is taking  $\Theta(\log_2 n)$  instead of  $\Theta(1)$ .

So the time complexity will become.  $\Theta(mn \log_2 n)$