CSCI-GA.1170-001 Homework 8

Ankit Sati

TOTAL POINTS

59.5 / 51

QUESTION 1

1 Optimal Trees 27 / 21

✓ **+ 1 pts** *Base case for (a)*

✓ **+ 4 pts** *Recursive formulation for best*

✓ **+ 2 pts** *Running time and justification*

✓ **+ 1 pts** *Base case for (b)*

✓ **+ 2 pts** *Running time and justification for (b)*

✓ **+ 1 pts** *base case for c*

✓ **+ 4 pts** *recursive formulation for (c)*

✓ **+ 2 pts** *running time for (c)*

✓ **+ 7 pts** *partial extra credit*

✓ **+ 3 pts** *recursive formula partially correct for (b)*

QUESTION 2

2 More with LCS 16 / 9

✓ **- 0 pts** *Correct*

✓ **+ 1 pts** *Correct dimension of Best in part (c)*

✓ **+ 1 pts** *Correct definition of Best in part (c)*

✓ **+ 2 pts** *Correct base case in part (c)*

✓ **+ 1 pts** *Partial credits for running time in part (c), since you have missed the minimum function.*

✓ **+ 2 pts** *Recursive formula only partially correct in part (c)*

QUESTION 3

3 Enumerating LCS 16.5 / 21

Part a

✓ **- 2 pts** *The proof is insufficient*

Part b

✓ **- 0 pts** *Correct*

Part c

✓ **- 0 pts** *Correct*

Part d

✓ **- 0.5 pts** *Incorrect or Missing for $$X[i] = Y [j]$$, $$Count[i - 1, j-1] + Count[i, j - 1] + Count[i - 1, j]$$, if $$c[i, j - 1] = c[i - 1, j] = c[i, j]$$*

✓ **- 0.5 pts** *Incorrect or Missing for $$X[i] = Y [j]$$, $$Count[i - 1, j-1] + Count[i, j - 1]$$, if $$c[i - 1, j] < c[i, j - 1] = c[i, j]$$*

✓ **- 0.5 pts** *Incorrect or Missing for $$X[i] = Y [j]$$, $$Count[i - 1, j-1] + Count[i - 1, j]$$, if $$c[i, j - 1] < c[i - 1, j] = c[i, j]$$*

Part e

✓ **- 0.5 pts** *Incorrect or Missing proof for $$Count[i, j] = Ω(n)$$*

✓ **- 0.5 pts** *Missing or Incorrect Running Time: $$O(mn^2)$$*

Problem 8-1

(a) Base Case :-

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

So, $BEST[1,1] = K[1]$.

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

$\vdots$

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

Recursive formulation :-

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

$BEST[i,j]$ represents the least cost of binary tree form $K[i \ldots\ldots j]$. So, we are considering l as root where $i \le l \le j$. 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.....,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] =∞
        for l=i to j:
```

Cont.→.

$BEST[i,j] = \min(BEST[i,j],$

$2\,BEST[i, l-1] + K[l] + 2\,BEST[l+1, j])$

Since there are 3 nested loops.
So the runtime is $O(n^3)$

(b) Base Case. For 1 to n

$BEST[i,j] = K[i], \quad 1 \le i \le n.$ | if $i > j$ $BEST[i,j] = 0$

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

Recursive formulation

$BEST[i,j]$ will give the least cost BT whose post order traversal corresponds to $K[i \ldots, j]$

$$BEST[i,j] = \min_{i \le l \le j} (2\,BEST[i,l] + 2\,BEST[l+1,j] + K[l])$$

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

Algorithm $(K[1, \ldots, n])$

for $P = 1$ to n.

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

$j = i + P - 1$.

if $(P == 1)$

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

ELSE:

$BEST[i,j] = \infty$

for $l = 0$ to j :

$BEST[i,j] = \min(BEST[i,j],$

$2\,BEST[i,l] + 2\,BEST[l+1,j] + K[j])$ cost

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

(a) Base Case →
~~BEST[i,i] = K[i]~~

(c)
Base Case. →

$$BEST[i,i] = K[i], \quad 1 \le i \le n. \qquad \text{if } i > j \quad BEST[i,j] = 0$$

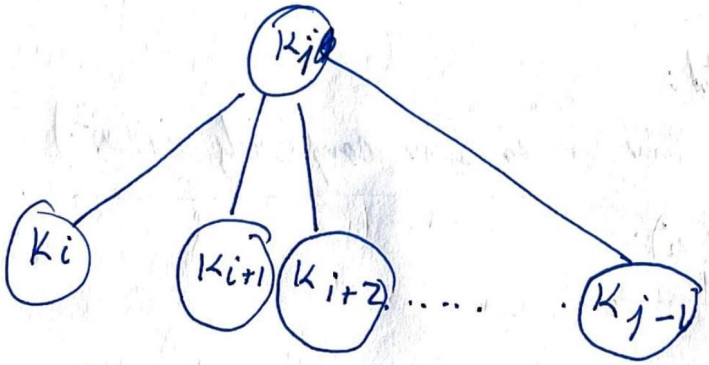
→ Recursive Formulation → . As K[i.....,j] is the postorder.
By traversal, K[j] will be the root.
Now we can calculate the least cost of subtree by calculating l.

~~Recursive Formulation~~

$$BEST[i,j] = \left(2 \sum_{x=1}^{l} BEST[i,x]\right) + K[j] \quad \min_{i \le k \le j}$$

Since this is a case of n-nodes arbitary 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]..... , K[j-2] are its direct children.



ALGORITHM (K[1,.....,n])

for P = 1 to n

for i = 1 to n-P+1.

j = i + p - 1.

if (p == 1):

BEST [i,i] = K[i]

else:

BEST [i,j] = BEST [i, j-1] + K[j-1] + K[j]

→ Since there are three nested loops.

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

Basically for BEST [i,j]

$$BEST[i,j] = K[j] + 2.[K[i] + K[i+1] + \ldots + K[j-1]).$$

So $BEST[i,j-1] = K[j-1] + 2[K[i] + K[i+1] + \ldots K[j-2])$.

So $BEST[i,j] \Rightarrow BEST[i,j-1] = K[i] + K[j-1]$

$$\Rightarrow BEST[i,j] = BEST[i,j-1] + K[j-1] + K[j]$$

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

⇒ Runtime = $O(n)^3$.

We see 3 loops nested:
each of them runs in time n so time complexity is $O(n^3)$.

Space Complexity is $O(n^2)$.

(d) Base Cases:

FOR. $i > j$

BEST $[i,j] = 0$

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

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

FOR $l > j - i$

BEST $[i,j] = 0$.

Recursive formulation:

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

For all values $K[i \ldots j]$, $K[j]$ will be the root. (Post order traversal).

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

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

→ Runtime $O(n^4)$.

There are a total of 4 loops.

Space Complexity = $O(n^3)$

as $BEST[i,j,k]$ takes 3 dimensions in total

1 Optimal Trees 27 / 21

✓ **+ 1 pts** *Base case for (a)*

✓ **+ 4 pts** *Recursive formulation for best*

✓ **+ 2 pts** *Running time and justification*

✓ **+ 1 pts** *Base case for (b)*

✓ **+ 2 pts** *Running time and justification for (b)*

✓ **+ 1 pts** *base case for c*

✓ **+ 4 pts** *recursive formulation for (c)*

✓ **+ 2 pts** *running time for (c)*

✓ **+ 7 pts** *partial extra credit*

✓ **+ 3 pts** *recursive formula partially correct for (b)*

Problem 8-2.

(a) Cost function will be such that if both characters are same, then cost is 1, otherwise it will be -1.

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

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

Since in the MCS problem, we have to maximize cost of some sub-sequence which be 1 and will be found only when the subsequence is a LCS.

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

(b) Base Case. Case.:

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

Recursive formulation -.

$$BEST[i,j] = \max\left(c[x_i, y_j] + BEST[i-1, j-1],\ BEST[i, j-1],\ BEST[i-1, j]\right)$$

So, we will have 3 options in the end.

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

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

Algorithm (X[1, ... m], Y[1, ... n])

for i = 0 to m

for j = 0 to n

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 algorith will be $\boxed{O(mn)} \approx O(m^2) \approx O(n^2)$. (quadratic time).

(c) Base Case.

↳. BEST [i, j, l] = 0, if i = 0, j = 0 or l = 0. BEST [i, j, k] the best possible cost of sub-sequence of length l at x_i & y_j. 3 dimensions [i, j, l.]

Recursive Formulation -.

BEST [i, j, l] = max (C [x_i, y_j] + BEST [i-1, j-1, l-1],

$1 \le l \le k$.

$1 \le i \le m$

$1 \le j \le n$.

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

Similarly we have 3 choices here.

⇒ 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].

⇒ Otherwise if we are either skipping x_i or y_j we have to find the subsequence of lenght l only.

ALGORITHM.(X[1, ..., m], Y[1, ... 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] + BEST[i-1, j-1, l-1], BEST[i, j-1, l], BEST[i-1, j, l])

m = -∞

for l = 1 to k:

m = max(m, BEST[m, n, l] - k²)

return maxCost.

As we can see, Run-time = O(mnk) for 3 nested loops.

Also Space Complexity = O(mnk).

2 More with LCS 16 / 9

✓ **- 0 pts** *Correct*

✓ **+ 1 pts** *Correct dimension of Best in part (c)*

✓ **+ 1 pts** *Correct definition of Best in part (c)*

✓ **+ 2 pts** *Correct base case in part (c)*

✓ **+ 1 pts** *Partial credits for running time in part (c), since you have missed the minimum function.*

✓ **+ 2 pts** *Recursive formula only partially correct in part (c)*

Problem 8-3.

(a) Let x be some string in the set $LCS(x' \| b, y' \| b)$.

So $x \in LCS(x' \| b, y' \| b)$

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

So.

$$x \in LCS(x', y') \| b.$$

$$\therefore LCS(x' \| b, y' \| b) \leq LCS(x', y') \| b.$$

Now let y be one string in the set $LCS(x', y') \| b$.

So $y \in LCS(x', y') \| b$.

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

So, $y \in LCS(x' \| b, y' \| b)$

$$\therefore LCS(x' y') \| b \leq LCS(x' \| b, y' \| b)$$

Now, since $LCS(x' \| b, y' \| b) \leq LCS(x', y') \| b$

and $LCS(x', y') \| b \leq LCS(x', y') \| b$.

Hence, $\boxed{LCS(x' \| b, y' \| b) = LCS(x', y') \| b.}$

$\Rightarrow$ We will get all the sub-sequence ending with b from $LCS(x' \| b, y' \| b)$ which is exactly what we needed.

ALGORITHM.(X[1, ..., m], Y[1, ... 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] + BEST[i-1, j-1, l-1], BEST[i, j-1, l], BEST[i-1, j, l])

m = -∞

for l = 1 to k:

m = max(m, BEST[m, n, l] - k^2)

return maxCost.

As we can see, Run-time = O(mnk) for 3 nested loops.

Also Space Complexity = O(mnk).

(b) Recursive Formula.

if $x_i = y_j$: } : if $i == 0$ OR $j == 0$
$L[i,j] = L[i-1, j-1] \| x_i$ } : if $x_i == y_j$.

else:

if $c[i-1, j] > c[i, j-1]$

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

else if $c[i-1, j] < c[i, j-1]$

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

else $L[i, j] = L[i-1, j] \cup L[i, j-1]$ (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 $n=1$.

$X_3 = 012$.

$Y_3 = .102$.

$LCS(X_3, Y_3) = \{0, 1, 2\}$

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

so $\lambda[3n, 3n] = \Omega(2^n)$ for $n=1$.

Induction Hypothesis.

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

Induction Step -

$X_{3(n+1)} = X_{3n+3} = X_{3n} X_3 = X_{3n} 012$.

$Y_{3(n+1)} = Y_{3n+3} = Y_{3n} Y_3 = Y_{3n} 102$

The LCS of X_{3n} and Y_{3n} is of length $2n$ for $n \neq 1$.

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

so, $\lambda[3(n+1), 3(n+1)] = 2 \times \Omega(2^n)$

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

So the induction justifies that

$\lambda[3n, 3n] = \Omega(2^n)$. This is justified from hypothesis

(e) When $X[i] = Y[j]$

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

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 $X[i] \neq Y[j]$:

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

$count[i, j] = count[i-1, j] + count[i, j-1]$

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

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

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

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

else:

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

Again we have 3 cases that arise.

If $c[i, j-1]$, $c[i,j]$ and $c[i-1, j]$ are same then $count[i,j]$ will be the sum of $count(i-1, j)$ and $count(i, j-1)$.

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

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

```
ALGORITHM (X[1,...,m], Y[1...n]):
    for i=0 to m:
    for j=0 to n:
    if i=0 or j=0
        count [i,j] = 1.
    else:
        if X[i] == Y[j]:
            count[i,j] = count[i-1,j-1]
    else:
        if c[i,j] = c[i-1,j] = c[i,j-1]:
            count[i,j] = count[i,j-1] + count[i-1,j]
            if c[i,j] = c[i-1,j-1]:
                count[i,j] = count[i,j] - count[i-1,j-1]
        else if c[i-1,j] = c[i,j]:
            count[i,j] = count[i-1,j]
        else:
            count[i,j] = count[i,j-1]
```

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

(B) Comparison and addition / Subtraction takes $O(i)$ where i is the number of bits.

Now, $n = 2^i$

$i = \log_2 n$.

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

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

3 Enumerating LCS 16.5 / 21

Part a

✓ **- 2 pts** *The proof is insufficient*

Part b

✓ **- 0 pts** *Correct*

Part c

✓ **- 0 pts** *Correct*

Part d

✓ **- 0.5 pts** *Incorrect or Missing for $$X[i] = Y [j]$$,*

$$Count[i - 1, j-1] + Count[i, j - 1] + Count[i - 1, j]$$, if $$c[i, j - 1] = c[i - 1, j] = c[i, j]$$

✓ **- 0.5 pts** *Incorrect or Missing for $$X[i] = Y [j]$$,*

$$Count[i - 1, j-1] + Count[i, j - 1]$$, if $$c[i - 1, j] < c[i, j - 1] = c[i, j]$$

✓ **- 0.5 pts** *Incorrect or Missing for $$X[i] = Y [j]$$,*

$$Count[i - 1, j-1] + Count[i - 1, j]$$, if $$c[i, j - 1] < c[i - 1, j] = c[i, j]$$

Part e

✓ **- 0.5 pts** *Incorrect or Missing proof for $$Count[i, j] = Ω(n)$$*

✓ **- 0.5 pts** *Missing or Incorrect Running Time: $$O(mn^2)$$*