Total Number of Pages in Section A (including top page): 12

EXAMINATION SCRIPT

STUDENT NO.						
1	8	0	5	1	0	6

DEPARTMENT:	CSE
BANGLADESH UNIVERISTY OF ENGINEERING AND TECHNOLOGY	

COURSE NO.	CSE203	DATE	31/08/2021
COURSE TITLE	Data Structures and Algorithms I		

SECTION A

Declaration on the Online Course Conduct by Undergraduate Student of BUET for COVID-19 Situation

On my honour, I bearing Student No. 5201805106 hereby declare that,

I shall not misuse, in any form or method, the course materials including Lecture Notes, Reading Materials, Audio and Video Records of the lectures of this course. I shall not adopt any unfair means during the Final Examination and shall not receive any help or offer/provide help to anyone. I shall preserve hard copy and soft copies of the answer scripts and will not expose the same to any person/party/media. I agree to accept any punitive measure taken by BUET Authority if at any time during or after the completion of the course it is revealed/violated otherwise

Signature: SK. Sabit Bin Mosaddek Date: 31.8.21

Instructions

1. Clearly enter your Student ID, Course Number, Course Title, and Date in the space provided. *Complete the declaration exactly as below with your signature and date.* You can also insert the scanned image of your handwritten declaration in this box.
2. Declaration: I shall not misuse, in any form or method, the course materials including Lecture Notes, Reading Materials, Audio and Video Records of the lectures of this course. I shall not adopt any unfair means during the Final Examination and shall not receive any help or offer/ provide help to anyone. I shall preserve hard copy and soft copies of the answer scripts and will not expose the same to any person/party/media. I agree to accept any punitive measure taken by BUET Authority if at any time during or after the completion of the course it is revealed/ violated otherwise.
3. Do not put your name or any other form of identification except the Student No. anywhere in the answer script.
4. Use offset/normal white paper of A4 size for writing the answer. Use only one side of the paper for writing. On each page, clearly write your Student ID and Page numbers.
5. After completing the exam, **before scanning**, please write the total number of pages in this Section (including the top page) on the top right corner of the top page.

Answer to the ques. no. 01

(a)

To calculate edit distance, we can use dynamic programming which can solve the problem in polynomial time.

here, dp[i, j] = Required cost to equate first i character of first string and first j character of second string

The recurrence will be then,

$$dp[,i,j] = \begin{cases} 2*(i+j) & \text{, if } i==0 \text{ or } j==0 \\ dp[i-1][j-1] & \text{. if } s[i] == t[j] \\ \min(dp[i-1][j]+2, dp[i][j-1]+2, & \\ \quad dp[i-1][j-1]+1) & \end{cases}$$

Here, the time complexity will be $O(nm)$ where n and m are the lengths of the strings.

```
~~solve (s, t, i, j):~~
    ~~if i==0 or j==0~~
            ~~return 2×(i+j)~~
   ~~if s[i]== t[j]~~
            ~~return solv~~
```

1805106 A-03

```
solve (s, t, n, m)
    for i = 0 to n :
        dp[i][0] = 2*i
    for j = 0 to m:
        dp[0][j] = 2*j

    for i = 1 to n :
        for j = 1 to m :
            if s[i] == t[j]
                dp[i][j] = dp[i-1][j-1]
            else
                dp[i][j] = min (dp[i-1][j]+2, dp[i][j-1]+2,
                                dp[i-1][j-1]+1)

    Return dp[n][m]
```

s\t			A	T	C	C	G	A
	i\j	0	1	2	3	4	5	6
	0	0	2	4	6	8	10	12
T	1	2	1	2	4	6	8	10
G	2	4	3	2	3	5	6	8
C	3	6	5	4	2	3	5	7
A	4	8	6	6	4	3	4	5
T	5	10	8	6	6	5	4	5
A	6	12	10	8	7	7	6	4
T	7	14	12	10	9	8	8	6

Ans : 6

(b) This can be solved using greedy algorithm. We first sort the data based on the ~~price~~ and profit and then we can search for each task an appropriate slot which is as date as possible.

```
{
solve (A, n)
 Input : an array of length n having task details
 Output : maximum profit
 ans = 0
 sort ( ~~An~~ A.begin(), A.end(), decreasing profit() ~~compareprofit()~~ );

 let ~~[illegible]~~ slot is an array of length max deadline.
 for ~~i~~x in slot : x = 0

 for i ~~i~~= 1 to n :
      int j = A[i].deadline

      ~~whi~~
      while ~~slot[j]~~ j>0 and slot [j] ==0 :
           j--
      if j>0
           slot [j] = 1
           ans += A[i].profit

 return ans
```

here if we sort the given tasks.
we will get,

T6, T2, T4, T1, T9, T8, T10, T7, T3, T5

first assign slot[5] to T6, ans = 15

then, slot[3] to T2, ans = 29

slot [6] to T4, ans = 39

slot [7] to T1, ans = 48

~~slo~~

slot [2] to T9, ans = 56

slot [1] to T8, ans = 59

slot [4] to T10, ans = 65

T7 can not be taken. ans = 65

T3 ~~is~~ also can not be taken, ans = 65

slot[8] to T5, ans = 68

Answer = 68.

Ans. to the ques. no. 2

~~(b)~~
~~Merge Sort (A, n)~~
~~Input : Array of length n~~

(a)

```
MergeSort (A, L, R)
    if (L+1 ≤ R)
        if A[L] > A[R]
            swap (A[L], A[R])
    else
        $m_1 = \frac{2L+R}{3}$
        $m_2 = \frac{L+2R}{3}$
        MergeSort (A, L, $m_1$)
        MergeSort (A, $m_1$+1, $m_2$)
        MergeSort (A, $m_2$+1, R)

        Merge (A, L, $m_1$, $m_2$, R)
```

~~1805107~~ 1805106 A-07

```
Merge (A, L, $m_1$, $m_2$, R)
    K=0, i=L, j=$m_1$+1 P=$m_2$+1
    ~~let temp an integer array~~
    ~~while (i ≤ $m_1$ or j ≤ $m_2$ or P ≤ R~~
        ~~if i ≤ $m_1$ and j ≤ $m_2$ and~~
    let $L_1$, $L_2$, $L_3$ three integer array
    for i = L to $m_1$
        $L_1$[i-L] = A[i]
    for j = $m_1$+1 to $m_2$
        $L_2$[j-$m_1$-1] = A[j]
    for i = $m_2$+1 to R
        $L_3$[i-$m_2$-1] = A[i]
    $n_1$ = $m_1$ - L+1, $n_2$ = $m_2$ - $m_1$, $n_3$ = R - $m_2$
    $L_1$[$n_1$] = $L_2$[$n_2$] = $L_3$[$n_3$] = ∞
    i=0, j=0, K=0, P=L
    while P ≤ R
        if $L_1$[i] ≤ $L_2$[j] and $L_1$[i] ≤ $L_3$[k]
            ~~if $L_2$ $L_3$ A[P~~
            A[P] = $L_1$[i++]
        elseif $L_2$[j] ≤ $L_1$[i] and $L_2$[j] ≤ $L_3$[k]
            A[P] = $L_2$[j++]
```

```
else
    A[P] = L_3[k++]
p++
```

Runtime:

$$T(n) = 3T\left(\frac{n}{3}\right) + \Theta(n)$$

Applying Master theorem,

$\log_3 3 = 1$ and $f(n) = n^1$

$\therefore T(n) = n\log n$

Usual merge sort also runs in $n\log n$. Thus, this algorithm does not improve much.

In case of n being a power of 3, this algorithm may run faster. And the algorithm goes less deeper in recursion tree than the usual merge sort.

(b) Recursion tree can help us guessing correctly. Otherwise we may have to use a lot of trial and error in order to find a tight upper bound.

Like the mergesort; if we draw the tree then we can well guess that the depth of the tree is $\log n$ and each depth usually costs $O(n)$. Thus we can easily deduce that, our running time will be ~~arrow~~ around $n\log n$. Then, we can justify our assumption using substitation.

(c) In-place sorting is ~~a memo~~ memory efficient. It only requires constant ~~of~~ amount of additional memory to perform the sort. When we have a tight bound on memory, we should prefer in-place sorting. Mergesort and heapsort are not in-place sorting which is why they require O(n) additional memory. On the other hand, quicksort, insertion sort, selection sort uses constant amount of memory.

—

Ans. to the ques. no. 3

The given Array,

[10, 30, 50, 70, 90, 100, 80, 60, 40, 20]

```
Partition (A, p, q)
  pivot = (p + q) / 2
  i = p-1
  for j = p to q
      if j == pivot
          continue
      if A[j] ≤ A[pivot]
          i = i+1
          swap (A[j], A[i]
          if . i == pivot
              pivot = j
  swap (A[i+1], A[pivot])
```

Steps of partitioning: pivot = (0+9)/2 = 4

[10, 30, 50, 70, 90, 100, 80, 60, 40, 20]

↑ ↑ ↑

i j pivot

[10, 30, 50, 70, 90, 100, 80, 60, 40, 20]

↑ ↑ ↑

i j pivot

[10, 30, 50, 70, 90, 100, 80, 60, 40, 20]

↑ ↑ ↑

i j pivot

[10, 30, 50, 70, 90, 100, 80, 60, 40, 20]

↑ ↑ ↑

i j pivot

[10, 30, 50, 70, 90, 100, 80, 60, 40, 20]

↑ ↑ ↑

j pivot j

[∵ j==pivot is continue]

[10, 30, 50, 70, 90, 100, 80, 60, 40, 20]

↑ ↑ ↑

i pivot j

[10, 30, 50, 70, 80, 100, 90, 60, 40, 20]

↑ ↑ ↑

i pivot j

[10, 30, 50, 70, 80, 60, 90, 100, 40, 20]

↑ ↑ ↑

i pivot j

[10, 30, 50, 70, 80, 60, 40, 100, 90, 20]

↑ ↑ ↑

i pivot j

```
[10, 30, 50, 70, 80, 60, 40, 20, 90, 100]
                                ↑   ↑        ↑
                                i  pivot     j
```

The loop ends here. Then we swap A[i+1] with A[pivot]

```
[10, 30, 50, 70, 80, 60, 40, 20, 90, 100]
                                 ↑
                               pivot
```

This is final array with pivot at 8

~~Th~~ Taking the pivot at middle does not guarantee that it will never run in $\Theta(n^2)$. We can see that the given array ~~almost~~ partitioned into an imbalanced two pieces. where one side has 8 element and other has 1 element. Thus it is always possible that there exist a case where the array will always partitioned into n-1 and 0 element which will cost $\Theta(n^2)$ in worst case.