National University of Computer and Emerging Sciences

Design and Analysis of Algorithms (CS2009)

Date: Feb 29, 2024

Course Instructor(s)

Dr. Saira Karim
Dr. Aasim Qureshi
Ms. Abeeda Akram
Mr. Usama Hassan Alvi
Mr. Sajid Ali Kazim
Mr. Iteza Muzaffar

Sessional-I Exam

Total Time: 1 Hours
Total Marks: 20
Total Questions: 3

Semester: SP-2024
Campus: Lahore
Dept: Computer Science

_____	_____	_____	_____
Student Name	Roll No	Section	Student Signature

_____	_____
Vetted by	Vetter Signature

Instruction/Notes: Attempt all questions. Answer in the space provided. **Do not attach rough sheets with this exam**. Do not use pencil or red ink to answer the questions**.** In case of confusion or ambiguity make a reasonable assumption.

CLO #1: Design algorithms using different algorithms design techniques i.e. Brute Force, Divide and Conquer, Dynamic Programming, Greedy Algorithms and apply them to solve problems in the domain of the program

Question1: **[10 marks]**

You are given a sorted array A of n integers, a search key v, and a positive integer k < n. Your goal is to find the k nearest values to the search key present in the array. Note that the search key may or may not be present in the array. Devise an efficient algorithm that returns the subarray (starting and ending index) containing the k nearest values. Also compute the time complexity of your algorithm. Please note that less effeicient solutions will be awarded lesser marks.

Input: A = { 2, 3, 5, 6, 9, 15, 23}, v=6 and k=4
Output: Subarray {3, 5, 6, 9}
Input: A= {6, 8, 9, 12, 23}, v = 4 and k=3
Output: Subarray {6, 8, 9}

CLO 2: Analyze the time and space complexity of different algorithms by using standard asymptotic notations for recursive and non-recursive algorithms.

Question2: **[5 marks]**

Derive the recurrence of the given recursive algorithm and solve the recurrence using tree method.

```
//Assume that "left" and "right" were initially pointing to the 1st and last index of the array.
data_partition(A[], left, right, N)
{
  if(left < right) {
        k = 2*((right-left+1)/7)
        v1 = data_partition(A, left, k, k)
        v2 = data_partition(A, k+1, right, N-k)
        v3 = update(A, left, right)
        return v1+v2+v3
  }
  return A[right]
}
update(A[], left, right){
        i = left
        r = 0
        for(i=left; i<=right; i*=2){
                for(m=left; m<=i; m++){
                        A[m] = A[i]*A[m]
                        r = A[m]
                }
        }
        return r
}
```

CLO 4: Implement the algorithms, compare the implementations empirically, and apply fundamental algorithms knowledge to solve practical problems related to the program.

Question3: **[5 marks]**

Suppose that you somehow know that the number of inversions in an array of size N is 10 where 10 much smaller than N. Which of the following sorting algorithms should you use to sort this array completely: Merge Sort, Selection Sort, Insertion Sort. Give an appropriate reason for your choice.

National University of Computer and Emerging Sciences, Lahore Campus

Course:	Design & Analysis of Algorithms	Course Code:	CS-2009
Program:	BS (Computer Science)	Semester:	Fall 2022
Duration:	60 Minutes	Total Marks:	21
Paper Date:	21-Mar-22	Weight	N/A
Section:	N/A	Page(s):	8
Exam:	Midterm	~~Roll No~~.	*Solution*

Instruction/Notes:

Weightage of the exam is Section Specific (i.e. for each section the weightage would be as per the announcement done in that regard).

Do NOT un-staple your exam, otherwise it might be cancelled.

Ample space is provided for rough work; NO EXTRA sheets will be provided.

Question 1: **3 Marks**

Arrange the function $(1.5)^n, n^{100}, (log\ n)^3, \sqrt{n}\ log\ n, 10^n, (n!)^2$, and $n^{99} + n^{98}$ in a list so that each function is big-*O* of the next function.

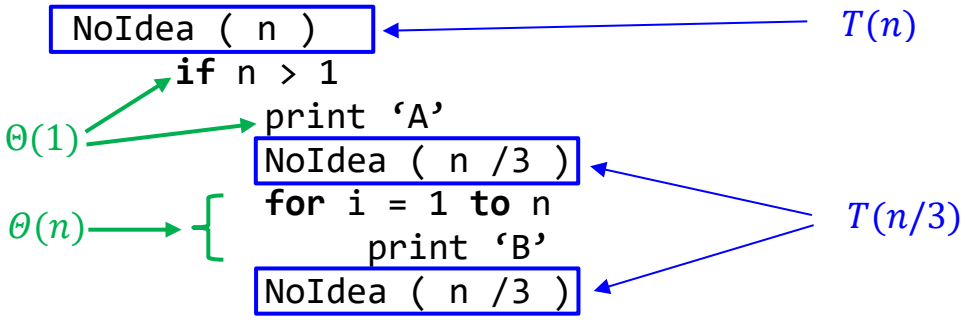
Answer:

$$(log\ n)^3 = O(\sqrt{n}\ log\ n) = O(n^{99} + n^{98}) = O(n^{100}) = O((1.5)^n) = O(10^n) = O((n!)^2)$$

Solution

Question 2: **4 Marks**

Consider the following algorithm



Write down the time-complexity of the above algorithm in recursive form (*only write the recursive equation, you are not required to solve the recurrence*).

Answer:

$$T(n) = \begin{cases} 2T\left(\frac{n}{3}\right) + \Theta(n) & \text{if } n > 1 \\ \Theta(1) & \text{otherwise} \end{cases}$$

Solution

Question 3: **5 Marks**

What would be the output of MyAlgo(A,1,10), where A is given as below:

$$A = \langle 3,7,2,6,1,8,5,4,0,9 \rangle$$

Note: Show complete working to claim for any **partial credit**.

```
MyAlgo(A,p,r)

if p < r
      q=DoSomething(A,p,r)
      MyAlgo(A,p,q − 1)
      MyAlgo(A,q,r − 1)


//--------------------------------------------------//

DoSomething(A,p,r)

x=A[r]
i=p-1
for j=p to r-1
     if A[j] ≤ x
          i=i+1
          exchange A[i] with A[j]
return i+1
```

Answer: $A = \langle 3,2,1,6,7,8,5,4,0,9 \rangle$

Working on the Next page

Solution

```
MyAlgo(A, 1, 10)                //Call 1
```
$A = \langle 3,7,2,6,1,8,5,4,0,9 \rangle$

```
    MyAlgo(A, 1, 9)          //Call 1.1
```
$A = \langle 3,7,2,6,1,8,5,4,0,9 \rangle$

```
        MyAlgo(A, 1, 0)    //Call 1.1.1
        MyAlgo(A, 1, 8)    //Call 1.1.2
```
$A = \langle 3,2,1,6,7,8,5,4,0,9 \rangle$

```
            MyAlgo(A, 1, 3) //Call 1.1.2.1
```
$A = \langle 3,2,1,6,7,8,5,4,0,9 \rangle$

```
                MyAlgo(A, 1, 0) //Call 1.1.2.1.1
                MyAlgo(A, 1, 2) //Call 1.1.2.1.2
```
$A = \langle 3,2,1,6,7,8,5,4,0,9 \rangle$

```
                    MyAlgo(A, 1, 0) //Call 1.1.2.1.2.1
                    MyAlgo(A, 1, 1) //Call 1.1.2.1.2.2
            MyAlgo(A, 4, 7) //Call 1.1.2.2
```
$A = \langle 3,2,1,6,7,8,5,4,0,9 \rangle$

```
                MyAlgo(A, 4, 3) //Call 1.1.2.2.1
                MyAlgo(A, 4, 6) //Call 1.1.2.2.2
```
$A = \langle 3,2,1,6,7,8,5,4,0,9 \rangle$

```
                    MyAlgo(A, 4, 5) //Call 1.1.2.2.2.1
```
$A = \langle 3,2,1,6,7,8,5,4,0,9 \rangle$

```
                      MyAlgo(A, 4, 4) //Call 1.1.2.2.2.1.1
                      MyAlgo(A, 5, 4) //Call 1.1.2.2.2.1.2
                    MyAlgo(A, 6, 5) //Call 1.1.2.2.2.2
    MyAlgo(A, 10, 9)         //Call 1.2
```

Solution

Question 4: **[1+6] Marks**

Let $A[1 \ldots n]$ be an array of n numbers such that

$$A[i] \leq A[i+2] \qquad \forall 1 \leq i \leq n-2$$

i) Write an instance of A for $n = 15$ (your array must not be already sorted).

$$A = \langle 2,1,4,3,6,5,8,7,10,9,12,11,14,13,15 \rangle$$

ii) Give an algorithm that sorts A in non-decreasing order i.e. the output of your algorithm should be the array A such that

$$A[i] \leq A[i+1] \qquad \forall 1 \leq i \leq n-1$$

Your algorithm must run in $\boldsymbol{O(n)}$ worst-case time.

Hint: Some modification to the Merge procedure could be helpful in this scenario.

Solution on the Next page

Solution

$n_1=\left\lfloor\frac{n}{2}\right\rfloor$

$n_2=\left\lfloor\frac{n}{2}\right\rfloor$

```
Let L[1…n1+1] and R[1…n2+1] be new arrays
i=1
j=1
for i=1 to n1
     L[i]=A[j]
     j=j+2
A[n1+1]=∞
i=1
j=2
for i=1 to n2
     R[i]=A[j]
     j=j+2
A[n2+1]=∞
i=1
j=1
for k=1 to n
     if L[i]≤R[j]
          A[k]=L[i]
          i=i+1
     else A[k]=R[j]
          j=j+1
```

Solution

Name: _____ Roll #: _____ Section: _____

National University of Computer and Emerging Sciences, Lahore Campus

Course:	**Design and Analysis of Algorithms**	**Course Code:**	**CS302**
Program:	**BS(Computer Science)**	**Semester:**	**Fall 2020**
Duration:	**90 Minutes**	**Total Marks:**	**40**
Paper Date:	**21-Oct-20**	**Weight**	**12.5%**
Section:	**ALL**	**Page(s):**	**5**
Exam:	**Midterm 1**		

Instruction/Notes: Attempt the examination on the question paper and write concise answers. You can use extra sheet for rough work. Do not attach extra sheets used for rough with the question paper. Don't fill the table titled Questions/Marks.

Question	1	2	3	4	5	Total
Marks	/ 5	/8	/5	/10	/12	/40

Q1) [5 marks] Selection Sort is an $O(n^2)$ algorithm that works by repeatedly swapping the next element in the array with the next minimum element. A pseudo-code is given below:

```
SelectionSort(A, n)
     FOR i ← 1 to n-1
          m ← i //assume i is the minimum index
          FOR j ← i+1 to n
               IF(A[j] < A[m])
                    m←j //update minimum index
     swap(A[i], A[m])   //swap min element with the ith element.
```

Is this algorithm, as described above, a stable sorting algorithm? Answer Yes or No. Then justify your answer in two lines.

Name: _____ Roll #: _____ Section: _____

Q2) [8 marks] You are implementing a class Set, where each set contains an array of unique ASCII characters. You wish to add the union and intersection methods to your Set class. What is the fastest asymptotic running time in which these functions can be performed between two such sets of size n each? First give the answer in terms of big-Oh, then explain your answer in a few lines.

Q3) [5 marks] The following line is a key part of the Merge Sort algorithm:

mid ← (left + right) /2

Suppose Merge Sort was applied to an array of size n. Then the above line will be executed approximately how many times? Encircle the correct answer below, then justify your answer in a few lines.

i) O(nlgn) times
ii) O(n) times
iii) O(lgn) times
iv) O(1) times.

Q4) Consider the following recursive algorithm:

```
StrangeSummation(A, p , r, sum) //sum is passed by reference
      IF(p<r){
            n ← r – p + 1
            StrangeSummation (A, p, p + n/3, sum);
            StrangeSummation (A, p + 2n/3, r, sum);

            FOR i← p to r
                  sum←sum+A[i];
      }
```

Name: _____ Roll #: _____ Section: _____

You may assume that $n=3^k$, where k= 0, 1, 2, …

i) [4 marks] Write the recurrence for the time function T(n).
ii) [6 marks] Solve your recurrence and derive a Big-Oh bound.

Name: _____ Roll #: _____ Section: _____

Q5) [12 marks] Given an array consisting of positive and negative integers, segregate them in linear time and constant space. Output should print all negative numbers followed by all positive numbers.
Example:

Input: {9, -3, 5, -2, -8, -6, 1, 3}

Output: {-3, -2, -8, -6, 5, 9, 1, 3}

Briefly explain the idea in a few lines and then write the pseudo code.

Name: _____ Roll #: _____ Section: _____

National University of Computer and Emerging Sciences, Lahore Campus

Course:	**Design and Analysis of Algorithms**	**Course Code:**	**CS302**
Program:	**BS(Computer Science)**	**Semester:**	**Fall 2020**
Duration:	**90 Minutes**	**Total Marks:**	**40**
Paper Date:	**21-Oct-20**	**Weight**	**12.5%**
Section:	**ALL**	**Page(s):**	**5**
Exam:	**Midterm 1**		

Instruction/Notes: Attempt the examination on the question paper and write concise answers. You can use extra sheet for rough work. Do not attach extra sheets used for rough with the question paper. Don't fill the table titled Questions/Marks.

Question	1	2	3	4	5	Total
Marks	/ 5	/8	/5	/10	/12	/40

Q1) [5 marks] Selection Sort is an $O(n^2)$ algorithm that works by repeatedly swapping the next element in the array with the next minimum element. A pseudo-code is given below:

```
SelectionSort(A, n)
      FOR i ← 1 to n-1
            m ← i //assume i is the minimum index
            FOR j ← i+1 to n
                  IF(A[j] < A[m])
                        m←j //update minimum index
      swap(A[i], A[m])   //swap min element with the ith element.
```

Is this algorithm, as described above, a stable sorting algorithm? Answer Yes or No. Then justify your answer in two lines.

Not stable:
If we run it on following array, 9a and 9b will not be in orignal order

Input: {2, 9a, 17, 18, 9b, 13, 4}
After first iteration of outer for loop: {2, 9a, 17, 18, 9b, 13, 4}
After second iteration of outer for loop: {2, 4, 17, 18, 9b, 13, 9a}
After third iteration of outer for loop: {2, 4, 9b, 18, 17, 13, 9a }
After fourth iteration of outer for loop: {2, 4, 9b, 9a, 17, 13, 18 }
After fifth iteration of outer for loop: {2, 4, 9b, 9a, 13, 17, 18 }
After sixth(last) iteration of outer for loop: {2, 4, 9b, 9a, 13, 17, 18 }

Name: _____ Roll #: _____ Section: _____

Q2) [8 marks] You are implementing a class Set, where each set contains an array of unique ASCII characters. You wish to add the union and intersection methods to your Set class. What is the fastest asymptotic running time in which these functions can be performed between two such sets of size n each? First give the answer in terms of big-Oh, then explain your answer in a few lines.

use count sort on both sets for sorting in linear time and then use merge routine for taking union and intersection.

OR

Use counting table of count sort for intersection (indices where we get a count of 2) and union (taking all indices where we get a positive count).

Q3) [5 marks] The following line is a key part of the Merge Sort algorithm:

mid ← (left + right) /2

Suppose Merge Sort was applied to an array of size n. Then the above line will be executed approximately how many times? Encircle the correct answer below, then justify your answer in a few lines.

i) O(nlgn) times
ii) O(n) times
iii) O(lgn) times
iv) O(1) times.

O(n) times

This line is executed once in each reccrsive call. Each reccrsive call is represented by a node in reccursion tree.

level	Number of nodes
0	1
1	2
2	$4 = 2^2$
3	$4 = 2^3$
. . .	
$K = \log_2 n$	$(2)^k$

$$\sum_{i=0}^{\log_2 n} 2^i = (2^{\log_2 n+1} - 1) / (2\text{-}1) = (2.n - 1) = \Theta(n)$$

Name: _____ Roll #: _____ Section: _____

Q4) Consider the following recursive algorithm:

```
StrangeSummation(A, p , r, sum) //sum is passed by reference
      IF(p<r){
            n ← r – p + 1
           StrangeSummation (A, p, p + n/3, sum);
           StrangeSummation (A, p + 2n/3, r, sum);

           FOR i← p to r
                 sum←sum+A[i];
       }
```

You may assume that $n=3^k$, where k= 0, 1, 2, …

i) [4 marks] Write the recurrence for the time function T(n).
ii) [6 marks] Solve your recurrence and derive a Big-Oh bound.

recurrence: $T(n) = 2T(n/3) + O(n)$

The reccurence tree will have $\log_3 n$ levels.

level	Number of computations
0	n
1	n/3 + n/3 = 2n/3
2	4n/9
3	8n/27
. . .	
K = $\log_3$ n	$n(2/3)^k$

Total computations at all levels of tree = $n\,(1 + 2/3 + (2/3)^2 + (2/3)^3 \ldots (2/3)^{\log_3 n})$

$$= n\sum_{i=0}^{\log_3 n} 2/3^i \leq n\sum_{i=0}^{\infty} 2/3^i = n\,(1/\,(1\text{-}2/3)) = O\,(n)$$

Q5) [12 marks] Given an array consisting of positive and negative integers, segregate them in linear time and constant space. Output should print all negative numbers followed by all positive numbers.
Example:

Input: {9, -3, 5, -2, -8, -6, 1, 3}

Output: {-3, -2, -8, -6, 5, 9, 1, 3}

Briefly explain the idea in few lines and then write pseudo code.

Solution:

Run partition function of quicksort with 0 as pivot. It is linear time and takes constant space.

National University of Computer and Emerging Sciences, Lahore Campus

Course:	Design & Analysis of Algorithms	Course Code:	CS-2009
Program:	BS (Computer/Data Science)	Semester:	Fall 2022
Duration:	60 Minutes	Total Marks:	27
Paper Date:	26-Spet-22	Section:	ALL
Exam:	Midterm 1	Page(s):	8
Name		Roll Number	

Instruction/Notes: Ample space is provided for rough work; NO EXTRA sheets will be provided.

Question	1	2	3	Total
Marks	/7	/10	/10	/27

Q1)

Consider the following sorting algorithm:

```
STOOGE-SORT(A, i, j)
1. if A[i] > A[j]
2.     then exchange A[i] ↔ A[j]
3. if i + 1 ≥ j
4.     then return
5. k ← ⌊(j − i + 1)/3⌋
6. STOOGE-SORT(A, i, j − k)      ▷ first two-thirds
7. STOOGE-SORT(A, i + k, j)      ▷ last two-thirds
8. STOOGE-SORT(A, i, j − k)      ▷ first two-thirds again
```

a) Give the recurrence for the worst-case running time of Stooge Sort. [2 Marks]

$T(n) = 3T(2n/3) + O(1)$

b) Calculate the running-time for Stooge Sort in Big Theta notation. [5 Marks]

Geometric series of 3

Height of tree = $\log_{3/2} n$

$3^{\log_{3/2} n}$

$= n^{\log_{3/2} 3}$

$= O(n^{2.7})$

Q2) Write a program that, given an array A[] of n numbers and another number x, determines whether or not there exist two elements in A[] whose sum is exactly x. [10 Marks]
A correct solution with $O(n^2)$ time complexity will get 2/10 Marks.
A correct solution with O(n lg n) or O(n) time complexity will get 10/10 Marks.
Input: *arr[] = {0, -1, 2, -3, 1}*
x= -2
Output: *Pair with a given sum -2 is (-3, 1)*
Valid pair exists
Explanation: *If we calculate the sum of the output,1 + (-3) = -2*
Input: *arr[] = {1, -2, 1, 0, 5}*
x = 0
Output: *No valid pair exists for 0*

Solution 1 (O(nlgn)

1. hasArrayTwoCandidates (A[], ar_size, sum)
2. Sort the array in non-decreasing order using randomized quick sort or merge sort.
3. Initialize two index variables to find the candidate elements in the sorted array.
    1. Initialize first to the leftmost index: l = 0
    2. Initialize second the rightmost index: r = ar_size-1
4. Loop while l < r.
    1. If (A[l] + A[r] == sum) then return 1
    2. Else if(A[l] + A[r] < sum) then l++
    3. Else r–
5. No candidates in the whole array – return 0

Solution 2 (O(nlgn)

Sort array using merge sort or quicksort
For (i: 1 to n)
    Bool = BinarySearch (A, x-A[i])
    If Bool is TRUE
        Return TRUE

Return FALSE

Q3) Following are two versions of quick sort partition function. These versions are O(n) time but not stable. Write pseudocode of stable version of partition function which runs in O(n) time. You can assume pivot is always the first element of the array. [5 Marks]

```
HOARE-PARTITION(A, p, r)
 1  x = A[p]
 2  i = p − 1
 3  j = r + 1
 4  while TRUE
 5      repeat
 6          j = j − 1
 7      until A[j] ≤ x
 8      repeat
 9          i = i + 1
10      until A[i] ≥ x
11      if i < j
12          exchange A[i] with A[j]
13      else return j
```

```
PARTITION(A, p, r)
1  x = A[r]
2  i = p − 1
3  for j = p to r − 1
4      if A[j] ≤ x
5          i = i + 1
6          exchange A[i] with A[j]
7  exchange A[i + 1] with A[r]
8  return i + 1
```

Q3 Solution

O(n) time and O(n) space stable partition function

```
j =1
 For (i: 1 to n)
        If (A[i] < pivot)
                B[j] = A[i]
                 j++
 For (i: 1 to n)
        If (A[i] > pivot)
                B[j] = A[i]
                 j++
```

Time Complexity = n + n = 2n = O(n)

Extra space for Rough work

National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	**Design and Analysis of Algorithms**	**Course Code:**	**CS 302**
	Program:	**CS**	**Semester:**	**Fall 2018**
	Duration:	**60 Minutes**	**Total Marks:**	**35**
	Paper Date:		**Weight**	
	Section:	**ALL**	**Page(s):**	**1**
	Exam Type:	**Sessional - I**		

Student : Name:_____ **Roll No.**_____ **Section:**_____

Instruction/Notes: Solve this exam on your answer sheets .

Problem 1 [15 pts]

Let A be an array of n distinct numbers. For indices i and j, if $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called an inversion of A (meaning that the numbers $A[i]$ and $A[j]$ appear in A in the inverted order).

a. List all the inversions in the array {2, 3, 8, 6, 1}

b. Give an algorithm that determines the number of inversions in an array A of size n in $O(nlgn)$ worst-case time.
Write the algorithm as a bulleted list of clearly specified statements, using English and symbols where necessary.

Problem 2 [10 pts]

Perform step-count analysis on the following pieces of code. Determine equations for the time functions $T(n)$ and express them in the big-Oh notation.

a.

```
for(int i=1;i<=n; i=i*2){
     for(int j=1; j<=i; j++){
                sum++;
     }
}
```

b.

```
for(int i=1;i<=n; i=i*2){
     for(int j=n; j>=1; j=j/2){
                sum++;
     }
}
```

Problem 3 [10 points]

a. If we modify the merge sort algorithm so that instead of dividing the array into two equal parts it divides it into three equal parts, sorts them separately, and then merges them to sort the original array, how will this effect the asymptotic running time of the algorithm?

b. Two arrays, A and B, both of size n, represent two mathematical sets of n numbers each. We wish to take the intersection of these sets. What is the fastest possible time in terms of big-Oh in which this can be done? Give a precise reason for your answer.

National University of Computer and Emerging Sciences, Lahore Campus

Course Name:	**Design and Analysis of Algorithms**	**Course Code:**	**CS302**
Degree Program:	**BS(CS)**	**Semester:**	**Spring 2020**
Exam Duration:	**60 Minutes**	**Total Marks:**	**40**
Paper Date:	**Feb 24, 2020**	**Weight**	**15**
Section:	**ALL**	**Page(s):**	**4**
Exam Type:	**Mid Term 1 Exam**		

Student : Name:_____ Roll No._____ Section:_____

Instruction/Notes: Attempt the examination on the question paper and write concise answers. You can use extra sheet for rough work. Do not attach extra sheets used for rough with the question paper. Do not use pencil and red ink. Don't fill the table titled Questions/Marks.

Question	1	2	3	4	5	Total
Marks	/15	/5	/5	/10	/5	/40

Q1) Suppose the higher management of FAST-NU is impressed by your problem solving skills and want you to add a new feature in flex that can answer the following query efficiently i.e. it should work in O(1): For a given class how many students earned grades between grade X and grade Y inclusive? Where grade X and Y can be any of these grades: {A+, A, A-, B+, B, B-, C+, C, C-, D+, D, F}. If you want you can preprocess the data beforehand so that you can answer the query within desired time bounds. **[15 Marks]**

Do you think you need to preprocess the data? If yes then explain your idea in 3-4 lines and then give the preprocessing steps (pseudo code) in details. Otherwise justify your answer.

Yes preprocessing is required. If for each grade g, we count the number of students having grade less or equal to g then we can answer this query in O(1) time. Below are steps
Make an array ***count*** of size 12 where grade F maps to index 0 and grade D maps to index 1 and so on
Define an O(1) function map that takes grade as input and returns the corresponding number e.g. for F it should return 0.
Initialize array count to zero
For i=1 to n
    count[map(student_grade)]++
For i=1 to 11
    count[i] = count[i] + count[i-1]

How much time will be required to preprocess the data if any? Give your answer in terms of n where n is number of students in the class.

O(n)

How will you answer the query in O(1)

Given grade X and Y max = maximum(map(X),map(Y)) and min = minimum(map(X),map(Y))
If(min>0)
    Total students = count[max] – count[min-1]
else
   Total students = count[max]

Q2) Suppose we are sorting an array of eight integers using Quick sort, and we have just finished the first partitioning with the array looking like this:

2 6 3 8 11 19 13 14 **[5 Marks]**

Which statement is correct?

a) The pivot could be either the 8 or the 11.
b) The pivot could be the 8, but it is not the 11.
c) The pivot is not the 8, but it could be the 11.
d) Neither the 8 nor the 11 is the pivot.

(a)

Q3) What is the asymptotic complexity of following algorithms when the input array is completely sorted in the desired order? **[5 Marks]**

1. Insertion sort
2. Quick sort(not randomized quick sort)
3. Merge Sort

1. **O(n)**
2. **$O(n^2)$**
3. **O(nlgn)**

Q4) Consider the following piece of code. Here, initially, | right - left | = n. Assume that n is large.

```
Function1(A, left, right)
          IF left < right
          {
                    k <– ((2*left)+right)/3

                    IF(rand()%2)
                    {
                              Function1(A, left, k)
                    }ELSE{
                              Function1(A, k, right)
                    }

                    Function2(A, left, right) //O(n) method
          }
```

i) Write a recurrence for the worst case and best case of Function1 **[5 Marks]**

Best Case:
T(n) = T(n/3) + O(n)
Worst Case:
T(n) = T(2n/3) +O(n)

ii) Solve the recurrence of worst case to find big-Oh bound (as tight as possible). Show complete working. **[5 Marks]**

T(n) = T(2n/3) +cn = T(4n/9) + c2n/3 + cn =

$$T(n) = T\left(\frac{2n}{3}\right) + cn = T\left(\frac{4n}{3}\right) + \frac{2cn}{3} + cn = T\left(\left(\frac{2}{3}\right)^{i} n\right) + cn\sum_{k=0}^{i-1}(2/3)^k$$

$$For\ \left(\frac{2}{3}\right)^{i} n = 1 \rightarrow i = log_{3/2}n$$

$$T(n) = T(1) + cn \sum_{k=0}^{log_{3/2}n-1} (2/3)^k \leq T(1) + cn\sum_{k=0}^{\infty}\left(\frac{2}{3}\right)^k = O(n)$$

Q5) Your friend makes the claim that the running time of their algorithm is both O(n) and O(nlgn). Are there any circumstances under which your friends claim may be true? **[5 Marks]**

If the running time of an algorithm is O(n) then it is also O(nlgn) (loose upper bound)

National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	Design and Analysis of Algorithms	Course Code:	CS2009
	Degree Program:	BSCS	Semester:	SPRING 2023
	Exam Date:	Monday, February 27, 2023	Total Marks:	24 + 12 = 36
	Section:	ALL	Page(s):	2
	Exam Type:	Mid-Term - I		

Student : Name:_____ **Roll No.**_____ **Section:**_____

Instruction/Notes: Attempt all questions. There are two question, don't forget to check the back side as well.

Question 1: [CLO - 2] [4+5+3+3+3+(4+2) = 24 Marks]

For each part in this question, you are required to analyze the given functions T(n) and answer the given question.

a) For the following function f(n), find a function g(n), such that f(n) = θ (g(n))

[Hint: For part a you have to provide the constants n_o, c_1, c_2]

$$\boldsymbol{T(n) = (1000)2^n + 4^n}$$

b) Solve the recurrence relation $\boldsymbol{T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{4n}{5}\right) + n}$, where $T(1) = 1$

[For part b you are required to use **recursion tree** method]

c) Solve the recurrence relation $\boldsymbol{T(n) = 25T\left(\frac{n}{5}\right) + n^2}$, where $T(1) = 1$

[For part c you can use any method]

d) Prove or disprove the following statement

$$2^{n+12} \; is \; O(2^n)$$

e) Prove or disprove the following statement

$$4^{12n} \; is \; O(2^n)$$

f) Consider the following sorting algorithm:

```
MySort(A, 1, 10)
    MergeSort(A, 1, 7)
    MergeSort(A, x, y)
    MergeSort(A, 1, 7)
```

i. What should be the minimum value of $y - x + 1$, so that MySort(A, 1, 10) correctly sorts the 100 elements array given to it as input.

ii. Based on your answer (for the minimum value of $y - x + 1$), what are exact values of x and y.

Question 2: [CLO - 1] [12 Marks]

Let A[1..n] be an array of *n* distinct numbers. If $i < j$ and $A[i] < A[j]$ then the pair (i, j) is called a compatibility of A. Design an algorithm that determines the number of compatibilities in any permutation on *n* elements in $O(n \log n)$ worst case time. (Hint: Modify merge sort.)

Roll Number: _____ Section: _____

CS 302 DESIGN AND ANALYSIS OF ALGORITHMS
MID 1 FALL 2016
Date: 19th Sept, 2016. Total Marks: 40 Time: 60 min

NOTE: Answer in the provided space. You can get extra sheets only for rough work. They will NOT be collected.

QUESTION 1 [4*5 = 20 marks] Following is a modified version of merge sort. It uses the same merge function as studied in class.

```
void mergeSort(int A, int p, int s){
    if(p<s){
        int one3rd=(s-p+1)/3;
        int q=p+one3rd;
        int r=p+2*one3rd;
        mergeSort(A,p,q);
        mergeSort(A,q+1,r);
        mergeSort(A,r+1,s);
        merge(A,p,q,r);
        merge(A,p,r,s);
    }
}
```

a) Draw a tree to show the division of the problem by this algorithm. Show the values of parameters p and s at each call. The first call in the main is: mergeSort(A, 0, 26)

b) Write down a recurrence relation T(n), with base case T(1) to describe the running time of this modified version of merge sort.

c) Solve the recurrence in (b) to get an asymptotic upper bound (big-Oh bound).

d) Based on the result in (c), if you were given a choice to use either regular merge sort or this version, which one will you choose and why? An answer without a reason will not get any points.

Roll Number: _____ Section: _____

QUESTION 2 [20 marks] Write an algorithm to take union of two sets of numbers, i.e. two arrays A, B of integers. Since these are sets, the numbers are distinct in each array, however, they could contain common elements. Make your algorithm as asymptotically efficient as possible.

Algorithm (in English):

C++ code

Design and Analysis of Algorithms

Sessional 1, Fall 2013

Date: September 23, 2013 **Marks: 45** **Time: 90 mins.**

Q1. [5+10] An array of n elements contains all but one of the integers from 1 to n+1.

i) Give the best algorithm you can for determining which number is missing if the array is sorted, and analyze its worst case asymptotic running time.

ii) Give the best algorithm you can for determining which number is missing if the array is not sorted, and analyze its worst case asymptotic running time.

Q2. [5 + 5] Analyze the running time of Quick Sort algorithm in the case where the pivot element always divides the array (at each step) into two potions of sizes 70% and 30% of the original.

i) Write a recurrence to describe the running time in this situation.

ii) Solve the recurrence to find an upper bound (Big O).

Q3. [5+5] Perform a step count analysis on the following and give tight bounds on each. Assume n is an exact power of 2.

i)
```
while(n >= 1){
        for(int j = n; j >= 1 ; j--){
                //do something in O(1)
        }
                n = n * 1/2;
}
```

ii)
```
for(int i=1; i <=n; i++){
        for(int j = i ; j >= 1 ; j = j/2){
                //do something in O(1)
        }
}
```

Q4. [5+5] A stable sort is one in which elements with same key values retain their relative order in the original array after sorting. For example, if elements x, and y, both have key equal to 5, and x appears before y in the array, then after sorting x still appears before y.

i) Is insertion sort (as implemented in class) a stable sort? (Don't write any code here, simply give a yes/no answer and explain why).

ii) How can we make sure that Merge Sort behaves like a stable sort? Where exactly in the code do we ensure this? (**Note:** You don't have to reproduce the code for Merge Sort, simply mention the relevant lines and how they ensure stability.)

Design & Analysis of Algorithms I

Mid 1, Spring 2014

Date: 27th Feb. 2014 Time: 90 mins.

Q1. (5+5)

Below are the pseudo codes for insertion sort and bubble sort. It is assumed that data is stored in an array A[1 … n]. Determine the loop invariant for the inner loops of both the sorts and prove their correctness.

Insertion Sort	Bubble Sort
for j = 2 **to** A.*length* *key* = A[j] i = j -1 **while** i > 0 **and** A[i] > *key* A[i+1] = A[i] i = i - 1 A[i+1] = *key*	**for** i = 1 to A.length-1 **for j** = 1 to A.length - j **if** (A[j] > A[j + 1]) temp = A[j] A[j]=A[j + 1] A[j + 1] = temp

Remarks: Looks good. No changes from me.

Modified Q2 (10)

Below is the pseudo code of count Sort. The indexes are 0-based for the C array, but 1-based for the arrays A and B. The below algorithm is stable.

If we however change the last for loop to go from 1 up to A.length, instead of A.length down to 1, it does not remain stable.

Your task is to change the Count sort code, so that with the new code, if we go from 1 to A.length in the last for loop, it still remains stable.
The modified algorithm must still be stable, and must still run in O(n+k) time.

```
COUNT-SORT (A, B , k)
//Let C[0..k] be a new array
for i = 0 to k
        C[i] = 0
for j = 1 to A.length
        C[A[j]] = C[A[j]]+1
for i = 1 to k
        C[i] = C[i] + C[i-1]
for j = A.length down to 1
        B[C[A[j]]] = A[j]
        C[A[j]] = C[A[j]] – 1
```

Design & Analysis of Algorithms I

Mid 1, Spring 2014

Date: 27th Feb. 2014 Time: 90 mins.

Modified Q3 (10)

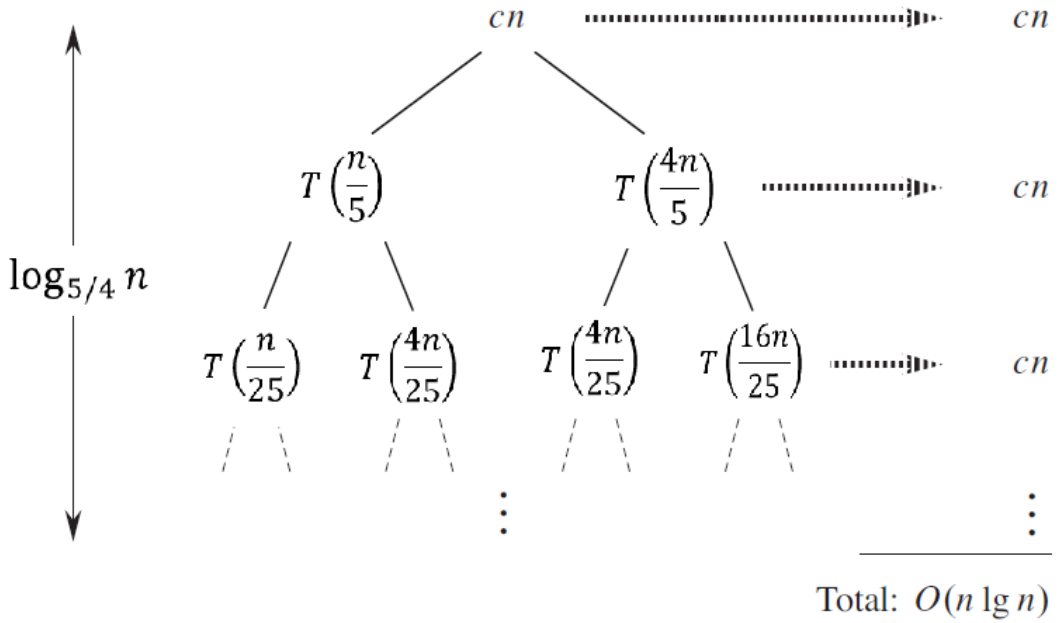
Let array A be an array consiting of only zeros and ones. (0's and 1's). Suggest an algorithm to sort the records in O(n) time and O(1) additional space.

Mid-1 (Solution)

Question 1:

a) As $\mathbf{4^n} \leq (1000)2^n + 4^n \leq (1001)\mathbf{4^n}$ for all $n \geq 1$, therefore, $(1000)2^n + 4^n = \mathbf{\Theta(4^n)}$. Here, $c_1 = 1, c_2 = 1001,$ and $n_0 = 1$.

b) Recursion Tree for $T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{4n}{5}\right) + n$ is given below:



Hence, the solution is $O(n \lg n)$.

c) We can solve the recurrence $T(n) = 25T\left(\frac{n}{5}\right) + n^2$, using the Master Method. Here, $a = 25, b = 5,$ and $f(n) = n^2$. As $n^{\log_5 25} = \Theta(n^2)$, the solution is $\mathbf{\Theta(n^2 \lg n)}$.

d) $2^{n+12} = O(2^n)$ as $2^{n+12} \leq c2^n$ for all $n \geq 1$, where c is a constant $\geq 2^{12} = 4096$.

e) $4^{12n} \neq O(2^n)$. To justify the answer, let's assume for the sake of contradiction that $4^{12n} = O(2^n)$. Then there exist positive constants c and n_0, such that $4^{12n} \leq c2^n$ for all $n \geq n_0$. Let's try to find the value of c: as $4^{12n} \leq c2^n$ it is same as $2^{24n} \leq c2^n$, or $2^{23n} \leq c$. Now, this is not possible because c is a constant and 2^{23n} is a variable having a positive growth rate. Hence, $4^{12n} \neq O(2^n)$.

f) The basic idea is that the last three elements must be included in the last call to sort the first seven elements (otherwise some element/s among the last three may not be included in any of the calls for MergeSort, and if any such element happens to have smaller value than the first seven elements, the final output would not be sorted).

i. Hence, the minimum value of $y - x + 1$ should be **6** to correctly sort the input.

ii. MergeSort(A, x, y) should be called as MergeSort(A, 5, 10).

Question 2:

```
CountAndSort(A, p, r)
1 if p < r
2     q=⌊p+r/2⌋
3     x=CountAndSort(A, p, q)
4     y=CountAndSort(A, q+1, r)
5     z=CountAndMerge(A, p, q, r)
6     return x+y+z
7 else return 0
```

```
CountAndMerge(A, p, q, r)
1 n1 = q - p + 1
2 n2 = r - q
3 let L[0…n1+1] and R[0…n2+1] be new arrays
4 for i = 1 to n1
5     L[i] = A[p + i – 1]
6 for j = 1 to n2
7     R[j] = A[q + j]
8 L[n1 + 1] = ∞
9 R[n2 + 1] = ∞
10 i = 1
11 j = 1
$$ count=0
12 for k = p to r
13    if L[i] <= R[j]
$$          count=count+(n2-j+1) //L[i] is compatible with all R[j…n2]
14          A[k] = L[i]
15          i = i + 1
16    else A[k] = R[j]
17          j = j + 1
$$ return count
```

The initial call would be `CountAndSort(A, 1, n).`

Design & Analysis of Algorithms - Spring 2013
Mid Term 1

February 26, 2013 **Time: 90 min**

Q1. (15)

Suppose you have an unsorted array A of colors *red*, *white* and *blue.* You want to sort this array so that all *reds* are before all *whites*, followed by all *blues*. Only operations available to you for this purpose are: equality comparison A[i] == c where c is one of the three colors, and swap(i, j) which swaps the colors at indices i and j in A. Write an algorithm to sort this array in O(n). First explain your algorithm in plain English and then code it.
(**Note: You cannot use an extra array in the solution.**)

Q2. (15)

You are given a very large array (you can assume it's of indefinite size); the first n entries of the array contain distinct integers in sorted order, after that all entries contain ∞. You DO NOT know the value of n. Devise an $O(lgn)$ time algorithm to search for an element key in this array.
(**Note**: The input to your program consists of a pointer to the beginning of the array, and the integer key.)

Q3. (5+5)

SelectionSort(A)

```
1. n = length[A]
2. for j = 1 to n − 1
3.       smallest = j
4.       for i = j + 1 to n
5.                if A[i ] < A[smallest]
6.                         smallest = i
7.       exchange (A[ j ], A[smallest])
```

a. State precisely a loop invariant for the **for** loop in lines 4-6, and prove that this loop invariant holds. Your proof should use the structure of the loop invariant proof presented in this chapter.

b. Using the termination condition of the loop invariant proved in part (a), state a loop invariant for the **for** loop in lines 2-7) that will allow you to prove that SelectionSort sorts the array correctly. You should use the structure of the loop invariant proof presented in this chapter.

Design & Analysis of Algorithms - Spring 2012
Mid Term 1

March 03, 2012 **Time: 90 min**

Q1. (15)
Devise an O(n) algorithm that determines the intersection of two sets of integers. The numbers in the sets may be positive and/or negative. The intersection of two sets is a set, possibly empty, that contains the common elements of the two sets.
First explain your algorithm in English and then write C++ code. Also derive the time complexity of code written in C++.

Q2. (5+5)
Find **maximum** and **minimum** items in a sorted list of n elements using **Divide and Conquer** strategy. First explain the algorithm in words and then write a recursive function to implement the algorithm. Also develop the recursive equation for the function and solve it.
(Note: The function should find out both the maximum and minimum function simultaneously. You should not write two functions --- one for minimum and the other for Maximum.)

Q3. (5+5)

(For sections A,B &C)

```
BUBBLESORT (A)
1 for i = 1 to A.length - 1
2 for j = A.length downto i-1
3 if A[ j ] < A[ j - 1]
4       exchange( A[ j ],  A[ j – 1] )
```

(For section D)

```
SelectionSort(A)
1. n = length[A]
2. for j = 1 to n − 1
3.      smallest = j
4.      for i = j + 1 to n
5.              if A[i ] < A[smallest]
6.                      smallest = i
7.      exchange (A[ j ], A[smallest])
```

a. State precisely a loop invariant for the **for** loop in lines 2–4 (lines 4-6), and prove that this loop invariant holds. Your proof should use the structure of the loop invariant proof presented in this chapter.

b. Using the termination condition of the loop invariant proved in part (a), state a loop invariant for the **for** loop in lines 1–4 (line 1-7) that will allow you to prove that BubbleSort(SelectionSort) sorts the array correctly. Your proof should use the structure of the loop invariant proof presented in this chapter.

National University of Computer and Emerging Sciences, Lahore Campus

Course:	Design & Analysis of Algorithms	Course Code:	CS-2009
Program:	BS (Computer/Data Science)	Semester:	Fall 2023
Duration:	60 Minutes	Total Marks:	19
Paper Date:	2-Oct-22	Section:	ALL
Exam:	Midterm 1	Page(s):	6
Name		Roll Number	

Instruction/Notes: Solve it on question paper

Question	1	2	3	4/5	Total
Marks	/4	/2	/8	/5	/19

Q1) Multiple Choice Questions [4 Marks]

a) Which of the following sort algorithms are stable?

i. Quick Sort
ii. Merge Sort
iii. Insertion Sort
iv. Count Sort

b) Which of the following sort algorithms are guaranteed to be O(n log n) even in the worst case?

i. Quick Sort
ii. Merge Sort
iii. Insertion Sort

c) Suppose we are sorting an array of eight integers using Quick sort, and we have just finished the first partitioning with the array looking like this:

2 5 0 6 8 13 9 10

Which statement is correct?

i. The pivot could be either the 6 or the 8.
ii. The pivot could be the 6, but it is not the 8.
iii. The pivot is not the 6, but it could be the 8.
iv. Neither the 6 nor the 8 is the pivot.

d) $f(n) = 30*2^n + 15*4^n + 3*16^n$

which of the following statements are true about f(N)

i. $f(n) = O(4^{2n})$
ii. $f(n) = O(8^{n+2})$
iii. $f(n) = O(2^{4n})$
iv. $f(n) = O(4^{n+4})$

Q2) What is asymptotic time complexity of following code. n is size of input. [2 Marks]

```
for(i=1; i<=n; i++)
{
   for(j=1; j<=n; j=j*2)
   {
      Arr[i] = Arr[i] + (Arr[j]*0.1)
    }
 }
```

Q3) [8 Marks] There are 2 sorted arrays **A** and **B** of size n each. Write an algorithm to find the median of the array obtained after merging the above 2 arrays (i.e. array of length 2n).

Input : ar1[] = {1, 12, 15, 26, 38}
ar2[] = {2, 13, 17, 30, 45}
Output : 16

Explanation :
After merging two arrays, we get
{1, 2, 12, 13, 15. 17, 26, 30, 38, 45}
Middle two elements are 15 and 17
Average of middle elements is (15 + 17)/2
which is equal to 16

(a) [4 Marks] Write an algorithm to solve the above problem that takes O(n) time.

(b) [4 Marks] Write an algorithm to solve the above problem that takes O(lg n).
Hint: You do not need to merge the two arrays. Use divide and conquer approach.

Question 4 is only for Sections BDS-(5A, 5B, 5C), BCS-(5C, 5D, 5E), BSR-5A

Q4) (a) Solve the following recurrence and write time complexity in asymptotic notation. Show all working [3 Marks]

$T(n) = 3T(n/3) + n^2$

Q4) **(b)** Consider the following algorithm

```
splitData(Arr[], left, right)
{
   if(left < right)
   {
      splitData(Arr, left, (left+right)/2)
      splitData(Arr, (left+right)/2 + 1, right)
      for(i=left; i<=right; i++)
      {
         for(j=1; j<=i; j++)
         {
            Arr[i] = Arr[i] + (Arr[j]*0.1)
         }
      }
   }
}
```

Give the recurrence for the worst-case running time of above algorithm. Only write the recurrence, you do not need to solve it [2 Marks]

Question 5 is only for Sections BCS-5A, BCS-5B

Q5) (a) [3 Marks] Prove that following statement is valid
$3n^2 + 29\ n = O\ (n^2)$
Prove by providing the value of c and n_0.

Q5) (b)[2 Marks]

i. How many comparisons will be made to sort the array arr={1,5,3,8,2} using counting sort?
   a) 5
   b) 7
   c) 9
   d) 0

ii. Which of the following sorting techniques is most efficient if the range of input data is significantly greater than a number of elements to be sorted?
   a) selection sort
   b) bubble sort
   c) counting sort
   d) merge sort