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