National University of Computer and Emerging Sciences

Assignment -02
Design and Analysis of Algorithms

Course Code: CS-2009

Due:18-Feb-2025 (11:59 PM)

Course Instructor	Usama Hassan Alvi
Course TA	Talha Azhar
Assignment -02	Divide and conquer Algo design problems
Section	BS(CS) 6A
Semester	Spring 2025
Marks	70

Department of Computer Science

FAST-NU, Lahore, Pakistan

Instruction/Notes:

- Attempt any seven questions.
- Plagiarism in any form will not be tolerated.
- Only handwritten,hard copy assignments will be accepted.
- You are also required to submit a pdf file of your hand written assignment on google classroom.
- **deadline 18-Feb-2025 (11:59 PM)**

Question #1 :

You are given two sorted arrays A[] and B[] of size "N" each. Your task is to design an in-place algorithm using divide and conquer technique to determine the median of both the arrays in O(log N). Median is the value located at the central position of sorted data.

Sample input and output:

Ar1[] = {5,12,15,26,38}
Ar2[] = {3,13,17,30.45}
Median: 16
Reason: overall sorted data 3,5,12,13,15,17,26,30,38,45 (total size = 10, even so median = (5^{th} term + 6^{th} term)/2
Sample input and output#2
Ar1[] = {35,45,50,60}
Ar2[] = {10,15,20,25}
Median: 30
Reason: (overall sorted data) 10,15,20,25,35,45,50,60 Median => (4^{th} term + 5^{th} term)/2

Question #2:

Given an array of integers Arr[] of length "N". Your task is to design an algorithm using divide and conquer technique to determine the majority element in the array. A majority element is an element that appears more than half (N/2) times in the array. Your algorithm must return the majority element if it exists in an array, otherwise return -1.

Note: You are not allowed to sort the array.

Sample input and output:

Input: Ar1[] = {48, 92, 35, 48, 48, 35, 48, 17, 48}
Majority Element: 48 (total size= 9), occurrences of the value 48 in the array = 5 (more than half)
Sample input and output#2:
Ar1[] = {48, 92, 35, 48, 92, 35, 48, 35, 48}
Majority Element: -1 (there is no element having occurrences more than half times)

Question #3

Q1) Given an array of n integers, an inversion is defined as a pair (i, j) where i < j and A[i] > A[j]. The task is to determine the total number of such inversions in the array. This count represents how far the array is from being sorted. If the array is already sorted in ascending order, the inversion count is 0, whereas if it is sorted in descending order, the count is maximized.

For example, consider the array [8, 4, 2, 1]. The valid inversion pairs are (8,4), (8,2), (8,1), (4,2), (4,1), and (2,1), leading to an inversion count of 6. Similarly, for the array [3, 1, 2], the inversion count is 2, as the valid pairs are (3,1) and (3,2). An efficient approach using O(n log n) complexity is required to compute the inversion count efficiently.

Question #4:

Find the Kth Smallest Element in Two Sorted Arrays

Given two sorted arrays **arr1** and **arr2**, and an integer **k**, find the **k-th smallest element** in the combined sorted array formed by merging **arr1** and **arr2**. The total length of the merged array is **n + m**, where **n** and **m** are the lengths of **arr1** and **arr2**, respectively.

Your goal is to find the **k-th smallest element** in **O(n + m)** time using a **Divide and Conquer** approach. You are not allowed to merge the arrays and then sort them, which would take **O((n + m) log(n + m))** time.

Example:

Input:

arr1 = [2, 3, 4, 10]

arr2 = [1, 5, 6, 8, 9]

k = 4

output:

4

Question #5:

The subarray sum problem, approached via the divide and conquer technique, seeks to find the maximum sum of a contiguous subarray within an array of integers. Through recursive division of the problem into smaller subarrays and subsequent computation of the maximum subarray sum for each segment and across midpoints. Now, consider the following problem:

Given an array of positive integers `nums` and a positive integer `target`, the task is to return the minimal length of a subarray whose sum is greater than or equal to `target`. If no such subarray exists, return 0 instead.

Example 1:

Input: target = 7, nums = [2,3,1,2,4,3]

Output: 2

Explanation: The subarray [4,3] has the minimal length under the problem constraint.

Example 2:

Input: target = 4, nums = [1,4,4]

Output: 1

Example 3:

Input: target = 11, nums = [1,1,1,1,1,1,1,1]

Output: 0

Question #6:

Consider an array of distinct integers Arr[] of size "N" was sorted in ascending order. The array has been rotated clockwise "K" number of times. Given such an array, your task is to find the value of "K". Design an algorithm using divide and conquer technique to get the value of "K" in O(log N).

Sample input and output:

Input: Ar1[] = {78, 85, 92, 25, 34, 48, 56, 62} // after "K" times clockwise rotation
Output: K = 3
Reason: Since original array was sorted so initial contents were Ar1[] = {25, 34, 48, 56, 62, 78, 85, 92}

Question #7:

Assume that you are given an unsorted array of integers Arr[] of size "N" and two integers "x" and "y". It is guaranteed that the values of both the integers are from the original array. Your task is to find the distance between "x" and "y". The distance between two elements of the array is the number of elements that lie between them in sorted order. Design an efficient algorithm using the divide and conquer approach to get the distance between two elements of the array.

Sample input and output:

Ar1[] = {30, 55, 40, 20, 50, 45, 10, 60, 25, 58}, x = 20, y = 50
Output: 4 => since after sorting the data we have (10, 20, **25, 30, 40, 45,** 50, 55, 58, 60**)**

Question #8:

Consider an unsorted array of integers Arr[] of size “N” and an integer “y”. Your task is to determine whether there exists any pair of indices whose absolute difference is equal to “y”. Design an algorithm using divide and conquer approach to solve this problem in O(N ∗ log N).

Sample input and output:

Input: Ar1[] = {3, 7, 2, 1, 4, 10} , y = 1
Since we are talking about Abs Difference so |3-4| = 1 and also |2-1| = 1 so your algorithm must return true

Question #9:

Given two sorted arrays “A[]”, and “B[]” of size “M” and “N” respectively and integer value “K”. Your task is to find the element that would be at Kth position of the final sorted array. Design an efficient in-place algorithm using divide and conquer approach to find the element at kth position of the final sorted array.

Sample input and output:

Input: Ar1[] = {20, 30, 60, 70, 90} , Ar2[] = {10, 40, 80, 100}, K = 5
Output: 60 the final sorted array would be like {10,20,30,40,**60**, 70, 80, 90, 100}