Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110
(An Autonomous Institution, Affiliated to Anna University, Chennai)
UCS3412 – COMPETITIVE PROGRAMMING LABORATORY

Batch : 2024-2028
Academic Year: 2025-2026 (Even)

Assignment 4 : Algorithms using Divide and Conquer

1. DAA QUESTIONS

Question 1: Finding the Maximum Value using Divide and Conquer [CO3, K3]

A data processing system needs to determine the highest value from a list of recorded sensor readings. To improve performance, the system applies a Divide and Conquer approach rather than scanning the list sequentially. Apply the Divide and Conquer algorithm to find the maximum element in the given list and determine the time complexity.

Sample Input:
Array = {15, 42, 7, 29, 88, 54}

Sample Output:
Maximum element = 88

Question 2: Counting Inversions using Merge Sort [CO3, K3]

In a ranking system, an inversion represents a pair of values that are out of order. Counting inversions helps measure how unsorted a list is. Apply the Divide and Conquer approach using Merge Sort to count the number of inversions in the given ranking list and determine the time complexity.

Sample Input:
Array = {7, 5, 3, 1}

Sample Output:
Number of Inversions = 6

Question 3: Finding the Kth Largest Element Using Quicksort Technique [CO3, K3]

Given an unsorted array of n integers and a positive integer k, find the kth largest element in the array using a Quicksort-based method (Quickselect).

Hint: In every iteration, check if the current pivot is placed in k^{th} position. If not, move the search to one of the subarrays based on the value of the pivot element, and discard the other half.

Sample Input:
Array = {7, 5, 3, 1, 12, -16, 4, -2} k = 3

Sample Output:
5

Question 4: Finding the Closest Pair of Points using Divide and Conquer [CO3, K3]

A GPS-based mapping system needs to find the two closest locations among multiple coordinate points to optimize travel routes. Apply the Divide and Conquer algorithm to determine the closest pair of points and compute the minimum distance.

Sample Input:
Points = {(2,3), (12,30), (40,50), (5,1), (12,10)}

Sample Output:
Closest Pair = (2,3) and (5,1)
Minimum Distance = 3.61

2. COMPETITIVE PROGRAMMING QUESTIONS

Question 1: You are given an array of k linked-lists lists, each linked-list is sorted in ascending order. Using divide and conquer, Merge all the linked-lists into one sorted linked-list and return it.

https://leetcode.com/problems/merge-k-sorted-lists/description

Question 2: You are given an integer array nums with no duplicates. A maximum binary tree can be built recursively from nums using the following algorithm:

- Create a root node whose value is the maximum value in nums.
- Recursively build the left subtree on the subarray prefix to the left of the maximum value.
- Recursively build the right subtree on the subarray suffix to the right of the maximum value.
- Return the maximum binary tree built from nums.

https://leetcode.com/problems/maximum-binary-tree/description

Question 3: The skyline problem

https://leetcode.com/problems/the-skyline-problem/description

Hint:

Think of the skyline as a sequence of critical points.
Can you break the list of buildings into two halves and solve the skyline for each half separately?

If you split the buildings into two parts:

- compute the skyline of the left half
- compute the skyline of the right half

How would you merge two skylines into one?

Each skyline can be represented as a list of (x, height) points. While merging:

- keep two pointers (one for each skyline),
- track the current height from left skyline and right skyline,
- at each step, the visible height is max(leftHeight, rightHeight).

When merging, only add a new point to the resulting skyline when the height changes.No change in height ⇒ no new critical point. What happens when:

- one skyline ends before the other?
- both skylines have the same x-coordinate?
- heights drop to zero?

Make sure your merge logic handles these cases.

Additional Practice Problems:
(Optional for lab record but included for tests and exams)

Question 1: Sorting Employee Salaries using Merge Sort [CO3, K3]

A company wants to sort employee salary records in ascending order for payroll analysis. Due to large datasets, an efficient sorting method is required. Apply the Merge Sort algorithm using Divide and Conquer to sort the given salary list. Show intermediate steps and determine the time complexity.

Sample Input:
Salaries = {45000, 32000, 60000, 50000, 38000}

Sample Output:
Sorted Salaries = {32000, 38000, 45000, 50000, 60000}

Question 2: Computing Power using Divide and Conquer (Fast Exponentiation)
[CO3, K3]

A scientific calculator needs to compute large exponential values efficiently. Instead of multiplying repeatedly, the calculator applies a Divide and Conquer technique to compute powers faster. Apply the Divide and Conquer algorithm to compute x^n efficiently and determine the time complexity.

Sample Input:
x = 2, n = 10

Sample Output:
Result = 1024

Question 3: Sorting Product Prices using Quick Sort [CO3, K3]

An online store needs to sort product prices quickly for display and filtering. Implement the Quick Sort algorithm using Divide and Conquer to sort the price list. Show pivot selection and partition steps. Determine the time complexity for best and worst cases.

Sample Input:
Prices = {1200, 500, 1500, 800, 300}

Sample Output:
Sorted Prices = {300, 500, 800, 1200, 1500}