Assignment 1

Analysis of Algorithm

Spring 2025

Submission Deadline: **Tuesday, 11th March, 2025 (During Lecture)**

Note: This assignment should be handwritten on A4 pages, with a printed cover page stating students' names and Roll Numbers, etc.

Question 1

Given the following recurrence relations, find the running time (in big-O notation) using the recursion tree method, substitution method and the Master Theorem.

1. $T(n) = 4T(n/2) + n^3$
2. $T(n) = 8T(n/4) + n^2$
3. $T(n) = 2T(n/4) + \sqrt{n}$
4. $T(n) = 3T(n/4) + n \log n$
5. $T(n) = T(9n/10) + n$
6. $T(n) = T(n - 1) + n$
7. $T(n) = 2T(n/2) + n / \log n$
8. $T(n) = T(n/2) + T(n/4) + T(n/8) + n$
9. $T(n) = T(n - 1) + 1/n$

Question 2

Suppose you are choosing between the following 3 algorithms:

- **Algorithm A** solves the problem of size n by dividing it into 5 subproblems of size n/2, recursively solving each subproblem, and then combining the solutions in linear time.
- **Algorithm B** solves the problem of size n by recursively solving two subproblems of size n −1 and then combining the solutions in constant time
- **Algorithm C** solves the problem of size n by dividing it into nine subproblems of size n/3, recursively solving each subproblem, and then combining the solutions in $O(n^2)$ time.

What is the time complexity of each algorithm, and which would you choose and why?

Question 3

Given a sorted array of distinct integers ***A*[1...*n*]**, you want to find out whether there is an index ***i*** for which ***A*[*i*] = *i***. Give a divide and conquer algorithm that runs in ***O*(log *n*)** time to find out an index ***i*** if it exists.

Question 4

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 an inversion of A. Give an algorithm that determines the number of inversions in any permutation on n elements in **O(n log n)** worst-case time. (Hint. Modify merge sort.)

Question 5

Describe a **O(n log n)**-time algorithm that, given n integers stored in an array **A[1 ... n]** and another integer **z**, determines whether or not there exist **1 ≤i; j ≤ n** such that **A[i] + A[j] = z** .