CS 310: Algorithms – Recurrences, Divide and Conquer *Mudassir Shabbir, February 04, 2026*

Student ID 1 & 2:*(Do this in pairs)*

---

1 Solve the following recurrence relations:

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

2. $T(n) = 5T(n/4) + n$

3. $T(n) = 2T(n-1) + 1$

4. $T(n) = 9T(n/3) + n^2$

2 Majority Element Problem

An array $A[1 \dots n]$ is said to have a *majority element* if more than half of its entries are the same. Given an array, the task is to design an efficient algorithm to determine whether the array has a majority element, and, if so, to find that element. The elements of the array are not necessarily from an ordered domain (e.g., integers), so comparisons of the form "$A[i] > A[j]$" are not allowed. You may only answer questions of the form "$A[i] = A[j]$" in constant time. (Think of the array elements as GIF files, for instance.)

Part 1: Show how to solve this problem in $O(n \log n)$ time.[1]

[1] *Hint:* Split the array A into two arrays A_1 and A_2 of half the size. Does knowing the majority elements of A_1 and A_2 help you determine the majority element of A? If so, use a divide-and-conquer approach.

Part 2: Can you give a linear-time algorithm?[2]

[2] *Hint:* Consider the following divide-and-conquer strategy:

- Pair up the elements of A arbitrarily to form $n/2$ pairs.
- For each pair: *If the two elements are different, discard both. If they are the same, keep exactly one of them.*

Show that after this procedure there are at most $n/2$ elements left, and that the remaining elements have a majority element if and only if the original array A does. Prove time complexity.