Algorithms

Assignment #1 (Deadline 17/11/2022)

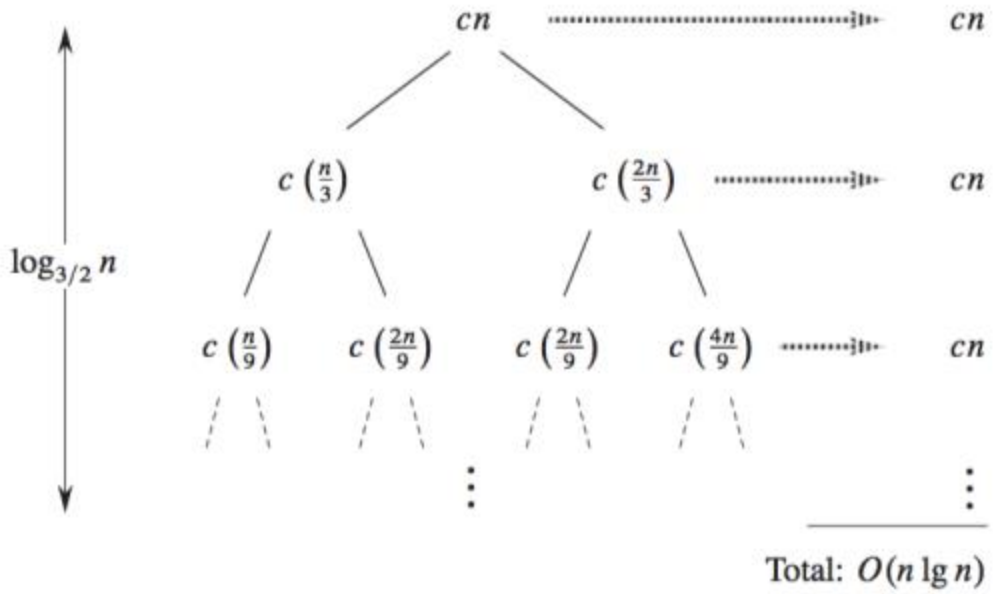
Guidelines:

- **This assignment is a group of 4 from the same Lab.**
- **Please submit your solutions via Google Classroom**
- **You should submit a zip file (name this file with your ids A1_ID1_ID2_ID3_ID4.zip), put all files (codes) for each problem (name code file p1 , p2 , .. etc)**

Problem 1:

Let c represent the constant factor in the O(n) term.

- Write Recurrence relation.
- Verify that upper bound is O(n log n) using substitution method.



Problem 2:

Construct a recursion tree for the recurrence T(n) = 3T(n/4) + cn2 and solve it using the substitution method.

Problem 3:

Solve the following recurrence relation:
T(0) = c1 , T(1) = c2 , T(n) = 2kT(n/2k) + (2k − 1)c3

Divide and Conquer Problems:
Solve all the following problems using divide and conquer strategy.

Problem 4:
Given an array of n elements A[0: n-1] that may contain positive and negative numbers.

- Find the **contiguous** subarray of numbers which has the largest sum.
- Write the recurrence relation and solve it.

Problem 5:
Find the multiplication matrix of 2 square matrices A and B of size n*n each. Write the recurrence relation and solve it.

Problem 6:
We are given an array of n points in the plane, and the problem is to find out the closest pair of points in the array. This problem arises in a number of applications. For example, in air-traffic control, you may want to monitor planes that come too close together, since this may indicate a possible collision. Recall the following formula for distance between two points p and q.

Euclidean distance d(p , q) = sqrt[(qx – px)^2 + (qy – py)^2]

The Brute force solution is O(n^2), compute the distance between each pair and return the smallest.

Find the smallest distance in O(nLogn) time using Divide and Conquer strategy.

Problem 7:
Consider an unsorted array with N number of elements and given number k value in range from 1 to N; **we have to find the kth largest element in the best possible way which take time complexity O(n)**

Example:

Input:
K = 3
A[] = {14, 5, 6, 12, 14, 8, 10, 6, 25}
Output: Kth largest element = 12

Input:
K = 5
A[] = {18, 15, 3, 1, 2, 6, 2, 18, 16}
Output: Kth largest element = 3

Problem 8:
Given an array of integers. **Find the Inversion Count in the array**.

Inversion Count: For an array, inversion count indicates how far (or close) the array is from being sorted. If array is already sorted then the inversion count is 0. If an array is sorted in the reverse order then the inversion count is the maximum.
Formally, two elements a[i] and a[j] form an inversion if a[i] > a[j] and i < j.

Examples:

Input: N = 5, arr[] = {2, 4, 1, 3, 5}
Output: 3
Explanation: The sequence 2, 4, 1, 3, 5
has three inversions (2, 1), (4, 1), (4, 3).

Input: N = 5
arr[] = {2, 3, 4, 5, 6}
Output: 0
Explanation: As the sequence is already
sorted so there is no inversion count.

Input: N = 3, arr[] = {10, 10, 10}
Output: 0
Explanation: As all the elements of array
are same, so there is no inversion count.