Design and Analysis of Algorithms

CSCI 3650 Spring 2021 Assignment 4

Type: Problem Set Due Date: Mon. 5^{th} Apr.

You have to solve the following problems and submit them. Although preferred, you are not required to type set the answers. If you choose to write your answers down, make sure that your writing is legible though. In either case, you should submit a single document in `pdf` format using CANVAS.

Make sure that the work you are submitting is your own. Try to write down your answers in a clear manner. Understandability of the solution is as desirable as correctness. Show all intermediate steps and justify any auxiliary claims that you make.

This homework is based on the **Divide and Conquer Paradigm** for algorithm design and this topic is discussed in Chapter 5 of the textbook.

1. Using the Master Theorem discussed in class, solve the following recurrence relations asymptotically. Assume $T(1) = 1$ in all cases. (25 Pts.)

   (a) $T(n) = T(9n/10) + n$

   (b) $T(n) = 16T(n/4) + n^2$

   (c) $T(n) = 7T(n/3) + n^2$.

   (d) $T(n) = 7T(n/2) + n^2$.

   (e) $T(n) = 2T(n/4) + \sqrt{n}\log^2 n$.

2. Sometimes, you might be interested in solving a recurrence relation exactly (i.e., not just asymptotically). Note that when you are interested in solving a recurrence relation exactly, you cannot use the Master Theorem. You can often do so by using the method of repeated substitutions (and observing the pattern and algebraically simplifying). You should have learnt this technique in discrete math courses. You can refresh your memory by solving the following two problems.

   - T(1)=3 and for all $n \geq 2$. $T(n) = T(n-1) + 2n - 3$. (10 Pts.)
   - T(1)=1 and for all $n \geq 2$, a power of 2, $T(n) = 2T(n/2) + 6n - 1$. (15 Pts.)

3. You are given an array A of n distinct integers (indexed 1 through n) which are not arranged in any particular order. You want to determine the smallest and the largest integer in the array. Observe that the problem is trivial if $n = 2$. For the sake of simplicity, you can assume that n is a power of two. (25 Pts.)

(a) Develop an algorithm based on the *divide and conquer* principle to do so. Present your idea in simple English. (5 Pts.)

(b) Now express the same idea in the form of a pseudocode. Specifically, write a recursive function Min-Max(A,i,j) that returns the smallest and the largest integer (the output will be a *struct* type as it consists of two elements) in the portion of the array starting from index i to j using the divide and conquer principle. Note that We can solve the original problem, by simply calling your function with parameters of $A, 1, n$. (7 Pts.)

(c) Develop a recurrence relation for the running time of your algorithm. State both the recurrence relation and the base condition for it. Assume that the comparison of two integers from the array is the **only** time consuming operation. Time spent doing anything else can be ignored. (5 Pts.)

(d) Solve the recurrence relation exactly (i.e., not just asymptotically) and determine the running time (i.e., number of comparisons) needed to solve a problem instance of size n. (8 Pts.)

4. Given an array $A[1..n]$ of natural numbers, find values of i and j with $1 \leq i \leq j \leq n$ such that

$$\sum_{k=i}^{j} A[k]$$

is maximized.

(a) Present a brute force algorithm for this problem which runs in time $O(n^3)$. (5 Pts.)

(b) With little bit of thinking, it is not too difficult to come up with an algorithm that runs in time $O(n^2)$. Present such an algorithm. [**Hint:** Use an auxiliary array to maintain partial sums] (8 Pts.)

(c) Devise an algorithm based on **divide and conquer technique** for solving this problem so that your algorithm runs in time $O(n \log n)$. (12 Pts.)