CSE 331: Algorithm and Complexity — Summer 2020

Homework 3

Instructor: Xiangyu Guo — **Deadline: Jul/15/2020**

Your Name: ____________________ Your Student ID: ______________

Problems	1	2	3	4	Total
Max. Score	10	20	20	20	70
Your Score					

Problem 1 (10 points). For each of the following recurrences, use the master theorem to give the tight asymptotic upper bound.

1. $T(n) = T(n/2) + O(n)$.
2. $T(n) = 4T(n/2) + O(n\sqrt{n})$.
3. $T(n) = 9T(n/3) + O(n^2)$.
4. $T(n) = 5T(n/2) + O(n^2)$.

Problem 2 (20 points). We consider the following problem of counting *stronger inversions*. Given an array A of n positive integers, a pair $i, j \in \{1, 2, 3, \cdots, n\}$ of indices is called a *strong inversion* if $i < j$ and $A[i] > 2A[j]$. The goal of the problem is to count the number of strong inversions for a given array A.

Give an $O(n \log n)$-time divide-and-conquer algorithm to solve the problem. (**Hint:** modify the merge-and-count rountine used for counting inversions.)

Problem 3 (20 points) An array A of n integers is said to be *bi-monotone* if there is some $i \in [n]$ such that the sub-array $A[1 \ldots i]$ is strictly increasing and the sub-array $A[i \ldots n]$ is strictly decreasing. Formally, we have $A[1] < A[2] < A[3] < \cdots < A[i1] < A[i] > A[i+1] > A[i+2] > \cdots > A[n]$. We say i is the *peak index* of the bi-monotone array. For example, the array $[30, 60, 80, 100, 120, 50, 20]$ is a bi-monotone array and 5 is the peak index.

Given an array A of size n, which is promised to be bi-monotone, we need to output its peak index. Design an $O(\log n)$-time divide-and-conquer algorithm to solve the problem.

Problem 4 (20 points). Suppose you know the prices of one stock in sequence of days. You can buy one share of the stock in some day and sell it in another day later. The goal of the problem is to maximize the profit. Notice that you can only buy and sell the stock once. *If you can not make a profit, you do not need to buy and sell the stock.*

The daily price of a stock is given in an array $A[1 \ldots n]$. ($A[1]$ is the stock price on day 1 etc. n can be fairly large, say we have data for 30 years.) The task is to determine the largest possible profit during the period of the stock. Specifically, we need to find i, j such that (1) $1 \leq i \leq j \leq n$, and (2) $A[j] - A[i]$ is maximum.

For example, if the prices of the stock in a sequence of 7 days is 15, 30, 18, 45, 9, 40, 35. Then you can buy one share of the stock in day 5 and sell it in day 6 and then your profit is $40 - 9 = 31$.

Design an $O(n \log n)$-time divide-and-conquer algorithm to solve the problem.