CSCI 270 - Spring 2023 - HW 5

Due: February 22, 2023

Graded Problems

1. [20 points] For the following recurrence equations, solve for $T(n)$ if it can be found using the master method (make sure to show which case applies and why). Else, indicate that the master method is not applicable and explain why.

   (a) $T(n) = 8T(n/2) + n \log n - 2023n$

   (b) $T(n) = 2T(n/2) + n^3(\log n)^3$

   (c) $T(n) = 4T(n/2) + n^2(\log n)^2$

   (d) $T(n) = 3T(n/3) - n \log n$

2. [10 points] Consider the divide and conquer solution described in the discuss section to find the closest pair of points in a 2D plane. Assume that we did not have a driver routine to sort the points. So our recursive function did not receive the points in sorted orders of their X and Y coordinates and the sorting had to be done for each subproblem (at every level). What would be the worst-case complexity of this algorithm assuming that the rest of the algorithm remains the same?

3. [10 points] Solve Kleinberg and Tardos, Chapter 5, Exercise 3.

4. [10 points] You are given with two integers a and b, and a variation of Fibonacci series, with $f(0) = a$ and $f(1) = b$. Recall that the Fibonacci sequence is $f(n) = f(n-1) + f(n-2)$. Devise an efficient algorithm to find the n^{th} Fibonacci number with $O(\log n)$ time complexity and prove its time complexity using recurrence relation.
   (Hint: You can represent the calculation of Fibonacci series using matrix multiplication as follows

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} f(n-1) \\ f(n-2) \end{bmatrix} = \begin{bmatrix} f(n-1) + f(n-2) \\ f(n-1) \end{bmatrix} = \begin{bmatrix} f(n) \\ f(n-1) \end{bmatrix}$$

   You can repetitively multiply the resultant matrix with $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ to get subsequent Fibonacci numbers.)

5. [10 points] You are given a **sorted** array consisting of $k+1$ values. Only one of the values appears once, and the rest of the k values appear twice. That is, the size of the array is $2k+1$. Design an efficient Divide and Conquer algorithm for finding which value appears only once. Partial credit (at most 6 points) will be given for non-Divide and Conquer algorithms. Discuss the runtime for your algorithm. Here are some example inputs to the problem:

   1, 1, 2, 2, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8

   10, 10, 17, 17, 18, 18, 19, 19, 21, 21, 23

   1, 3, 3, 5, 5, 7, 7, 8, 8, 9, 9, 10, 10