CS340 Analysis of Algorithms
Fall 2023

Homework:	**5**	**Professor:**	**Dianna Xu**
Due Date:	**10/24/22 (Tuesday)**	**E-mail:**	`dxu@cs.brynmawr.edu`
Office:	**Park 203**	**URL:**	`http://cs.brynmawr.edu/cs340`

For divide-and-conquer full write-ups, all time analysis should include an explicit statement of the recurrence and what it solves to. All proof of correctness should be by structural induction.

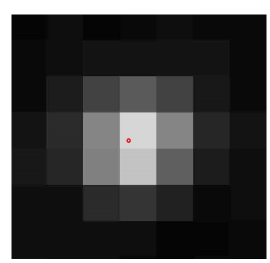
1. Solve the following recurrence relations; i.e. express each one as $T(n) = O(f(n))$ for the tightest possible function $f(n)$, and give a short justification (some are longer than others, but full inductions are not required). Unless otherwise stated, assume $T(1) = 1$.

1. $T(n) = 4T(n/2) + n^2$
2. $T(n) = 3T(n/4) + \sqrt{n}$
3. $T(n) = 7T(n/3) + n^3$
4. $T(n) = 2T(n/2) + nlogn$
5. $T(n) = 2T(\sqrt{n}) + 1$, where $T(2) = 1$

2. Full write-up. Haverford discovered in late August that some fake OneCards had been mixed in with the new ones they were planning to give the incoming freshmen. Security knew that strictly less than half of them were fake and that the fake cards had incorrect access codes (which may or may not be the same), while all the real cards had the same, correct, access code. They had a machine that could check if two OneCards had the same access code (but the machine couldn't say what the code was). Design an $O(nlogn)$ divide-and-conquer algorithm that would have helped Security to quickly find a single real OneCard to give a freshman arriving for an ID. Extra credit: find an $O(n)$ solution. It is ok if your extra credit solution is not divide and conquer.

3. Full write-up.

A typical representation of a grayscale image is an $n \times n$ 2D array, where each $[i, j]$ index stores a color value (for example, an integer between 0 and 255, with 0 reprensenting a black pixel and 255 white). In image processing, it is often the case that we would like to find those pixels that are brighter than all the surrounding pixels. This has applications in feature detection and many other filtering techniques. Design a divide-and-conquer algorithm that finds any one such pixel in $O(nlogn)$ time. You may limit the neighbors to the 4 edge neighbors. It is acceptable to interpret brighter than as "brighter than or equal to", or simply assume that all neighboring pixel values are distinct. Extra credit: find an $O(n)$ solution.



Please hand in your assignment electronically on Moodle.