Algorithms — CS 102 — Spring 2014
Instructor: Achlioptas
Homework 1

1. Suppose you are sent a sorted list of the distinct integer ID numbers of the students in the class. The IDs arrive in increasing order and your router dutifully puts them into a circular buffer, but then bursts into flames because you left it under a pile of dirty clothing on a hot day. You are thus left with an array containing an increasing sequence of integers, but one that starts somewhere in the middle of the array and wraps around. For example, $[68, 73, 75, 122, 16, 19, 33, 36, 38, 55, 59]$. Design an algorithm to find the last ID (the largest integer) in the list in $O(\log n)$ time. Prove that your algorithm is correct and that its running time is $O(\log n)$.

2. You are tallying votes from an election in which n people voted. If any one candidate gets more than half (at least $\lfloor n/2 \rfloor + 1$ votes), they win. Otherwise a runoff is needed. For privacy reasons you are not allowed to look at any one ballot, but you have a machine that can take any two ballots and answer the question: "are these two ballots for the same candidate, or no?" Design and analyze a divide and conquer algorithm that decides whether a runoff is needed that uses $O(n \log n)$ ballot equality tests.

   Hint: For this problem it is important to be mindful of floors and ceilings when you split the set of ballots in half.

   Extra Credit: Design an algorithm that uses $O(n)$ ballot equality tests.

3. Consider the following sorting algorithm: First sort the first two thirds of the elements in the array. Next sort the last two thirds of the elements of the array. Finally, sort the first two thirds again. (After each two-thirds sort, the relevant elements are stored in the same two thirds of the array were they resided before the sort; no extra storage is allocated.)

   (a) Give an informal but convincing explanation (not a rigorous proof by induction) of why this algorithm returns a fully sorted array.

   (b) Find a recurrence relation for the worst-case running time of this algorithm, then solve it by applying the Master Theorem.

4. Suppose you're given an array of size n which contains all but one of the integers from 0 to n. Assume $n = 2^k - 1$ for some integer k. Design a **divide and conquer algorithm** to find the missing number. Argue (informally) that your algorithm is correct and analyze its running time. Do **not** use any $O(n)$ sorting algorithm like Radix Sort.

   Hint: You can find the median of a list in $O(n)$.

Disclaimer

You have to justify everything. This means that you have to explain both why your algorithm is correct, i.e., why it produces the correct answer for every input, and why it runs in the time that you claim. Think of the person reading what you write as your boss who [having gotten you stuck with the problem] would like to earn praise for the group by showcasing your solution at the group's presentation to “the higher ups”[1]. This means that your boss *wants* your solution to be correct, but she is not willing to risk public humiliation in case you “missed” something. So, she will be reading what you write very carefully, but with good intent. Make her life easy by explaining things in a complete, lucid way that makes it clear that you’ve considered all the details. Explicitly stating all the details, describing data structures, and (God forbid!) presenting code-style-pseudocode is **not** the way to do this. Formulating principles, invariants and properties that are maintained by your algorithm **is** the way to do it[2]. And much harder.

[1]If you think that this is a negative way of representing professional reality, one that enforces cultural and class stereotypes, I strongly suggest you stay in college for as long as you can afford...

[2]The textbook itself, in my opinion, does a pretty good job at this. Don’t be afraid to use words, but do so in complete, well-thought sentences. Take time. Don’t “text” your answers.