1. Suppose a sorted array $A$ of $n$ numbers is given in which exactly one value appears once while the rest appear twice. Our goal is to find the value that appears only once. For example, if


the answer is 20. Design a $O(\log n)$ algorithm.

2. Suppose a sorted array $A$ of $n$ numbers is given containing terms of an arithmetic progression but with one term missing. Our goal is to find the missing term. For example, if

$$A = [5, 8, 11, 14, 20, 23, 26, 29, 32, 35],$$

the answer is 17. Design a $O(\log n)$ algorithm.

3. Suppose an array $A$ of $n$ numbers is given along with an integer $M$. Design an $O(n \log n)$ algorithm that checks whether there exist distinct indices $i$ and $j$ such that $A[i] + A[j] = M$.

4. Suppose an array $A$ of $n$ numbers is given along with an integer $M$. Assume that an algorithm which finds the median of an unsorted array in linear time is also given. Design a $O(n)$ algorithm that finds the largest subset $S \subseteq \{1, 2, \cdots, n\}$ satisfying $\sum_{i \in S} A[i] \leq M$.

5. Say that an array $A$ holding $n$ objects has a dominant object if at least $\lfloor n/2 \rfloor + 1$ entries of $A$ are identical. Our goal is to determine if $A$ has a dominant element and, if yes, identify it. Our only access to $A$ is by asking an oracle whether $A[i] = A[j]$ for any two $i, j \in [n]$.

   (a) Design an algorithm to solve this problem with $O(n \log n)$ questions. (Hint: first solve the problem when $n$ is a power of two.)

   (b) Design an algorithm to solve this problem with $O(n)$ questions. (Hint: Don’t divide. Pair up the elements arbitrarily and get rid of as many as you can, repeatedly.)

6. Suppose a problem $P$ can be solved in time $T_1(n) = n \log n$ using an algorithm $A_1$. Suppose also that $A_2$ is a divide-and-conquer algorithm for $P$ which for $n > 1$ generates two instances of $P$ of size $n/2$ in time $w(n) = c$, where $c > 4$ is a constant. Let $n$ be a power of 2.

   (a) Find the running time of $A_2$, assuming $T_2(1) = 1$, up to the correct leading constant.

   (b) Let $A_3$ be an algorithm that uses $A_2$ for $n > n_0$ but $A_1$ for $n \leq n_0$. Find the value of $n_0$ that minimizes the running time of $A_3$. 

See Next Page
Disclaimer

You have to justify everything. This means that you have to explain (prove) 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 would like to earn praise for the group by showcasing your solution at the group’s presentation to “the higher ups”. 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 have 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. And much harder.

1If you think that this is a negative way of putting things, enforcing cultural and class stereotypes, I strongly suggest you stay in college for as long as you can afford...

2The 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.