COSC 240, Fall 2018: Problem Set $\#2$

Assigned: Monday, 9/24/17.
Due: Monday, 10/1, at the beginning of class (hand in hard copy).
Lectures Covered: 5 to 7.
Academic Integrity: You can work with other people in the class but you must write up your own answers in your own words. You can also use the textbook and talk to the professor. You may not use any other resources (e.g., material found online) or talk to people outside the class about these problems. See the syllabus for details on the academic integrity policy for problem sets.

Problems

1. Describe with pseudocode a divide-and-conquer algorithm that finds the largest value in an array A of numbers. In more detail, your algorithm should take as arguments an array A, and two integers p, r, such that $1 \leq p \leq r \leq length(A) = n$, and return the largest value in $A[p...r]$.

2. Describe the step complexity of your solution to problem 1 with a recurrence. Solve the recurrence using the master method.

3. Consider the problem in which you are provided an array of bits and the goal is to count the number of 0 bits in the array. Describe with pseudocode a randomized divide-and-conquer algorithm that satisfies the following two properties:

   (a) For the worst-case inputs and random coin flip outcomes, it requires $\Omega(n^2)$ steps.

   (b) Its randomized step complexity is $\Theta(n)$.

   To receive any credit, your algorithm must be a pure divide-and-conquer solution (fully recursive; no loops).

   (Hint: My solution flipped a biased coin at the beginning of the initial call—i.e., when the subproblem size is the same as the size of the whole array—and then passed the outcome of this one flip to all recursive calls that followed.)

4. Analyze the randomized step complexity of your solution to the above problem.

5. Consider the *pseudo-sorting* problem where you are provided an array A containing $n > 1$ unique values (i.e., no two value are equal), and you must return an array B that contains a permutation of these values reordered such that the $k = \lfloor n/2 \rfloor$ smallest values in A are in the first k positions of B. The order of the values in the first k positions does not matter. All that matters is that the k smallest values occupy these first k positions.

   Describe in pseudocode the fastest possible solution you can come up with to this problem. Then provide a big-Θ time complexity analysis of your solution. Two notes:

   - Your solution is allowed to make use of a **median-search**(A) subroutine that returns the median value from A and requires only $O(length(A))$ steps.

- Your solution does *not* have to use a divide-and-conquer strategy. (In fact, it is much easier if you do not use divide-and-conquer.)

6. Argue clearly and concisely why the comparison-based sorting algorithm lower bound does not apply to this pseudo-sorting problem.