Week 5: Lab

Collaboration level 0 (no restrictions). Open notes.

1. Consider a decision tree corresponding to a comparison-based sorting algorithm; and consider the shortest path from the root to a leaf. What does this path correspond to, in terms of running time?

2. What is the smallest possible depth of a leaf in a decision tree for a comparison-based sort of n elements?

3. Suppose that we were to rewrite the last for loop in Counting-sort as: for j=1 to A.length (instead of: for j=A.length down to 1). Would the algorithm work (i.e. sort properly)? What would change?

4. Assume you have n elements in the range $\{-20, -19, ..., ..., 19, 20\}$. How would you modify Counting-sort to sort this array?

5. Assume you have $n = 1,000$ integers in the range $\{0, .., 50\}$ and you need to sort them. Would you use Counting-sort or Quicksort, and why?

6. Describe one scenario when Counting-sort is more efficient than Quicksort.

7. Describe one scenario when Quicksort is more efficient than Counting-sort.

8. Describe one scenario when you cannot use Counting-sort.

9. Recall that in the (smart) SELECT() algorithm described in the notes, the input elements are divided into groups of 5. In this problem we'll look at what happens if the input is divided into groups of 7 element instead.

   (a) As before, the algorithm finds a "good" pivot before calling Partition: In this case, for each group of 7 elements it computes its median, and then finds the median of these medians. Denote it by x. Using the same argument as in the notes, find out how many elements in the input are guaranteed to be $< x$; and how many elements are guaranteed to be $> x$, respectively.

(b) Write the recurrence corresponding to this version of the Select() algorithm.

(c) Does this solve to $O(n)$ time?

(d) Based on this, does dividing the input into groups of 7 elements lead to a linear time SELECT() algorithm?

Note: Using same arguments, it can be shown that groups of size > 5 lead to a linear time algorithm, and groups of size < 5 do not lead to a linear algorithm.

10. Let A be a list of n (not necessarily distinct) integers. Describe an $O(n)$-algorithm to test whether any item occurs more than $\lceil n/2 \rceil$ times in A.

    (a) You may assume that the integers are in a small range, $K = O(n)$.

    (b) Come up with a general solution, without making any additional assumptions about the integers (in particular you may not assume that the range is small). Hint: use Select()

11. (CLRS 9-1) Given a set of n numbers, we wish to find the i largest in sorted order using a comparison-based algorithm. Spell out the algorithm that implements each of the following methods with the best asymptotic worst-case running time, and analyze the running times of the algorithms on terms of n and i.

    (a) Sort the numbers, and list the i largest.

    (b) Build a max-priority queue from the numbers, and call EXTRACT-MAX i times.

    (c) Use a SELECT algorithm to find the ith largest number, partition around that number, and sort the i largest numbers.

Note: For each problem we expect a justification of the answer; if you give an algorithm, we expect pseudocode, the main idea, justification of correctness, and analysis.