Homework Assignment No. 2

CSCI 4470/6470 Algorithms, CS@UGA, Fall 2019

Due Monday September 16, 2019

Five questions with total 140 points.

Homework submission should be in hardcopy. If for a reason an email submission is necessary, it needs an approval from this instructor. Submission needs to be received by 5:00pm on the due day.

1. (**40 points**) Consider that duplicate elements may occur in a list to be sorted by QUICK SORT.

   (a) Modify function PARTITION so that elements of the same value as a pivot are placed in consecutive positions after function PARTITION is applied on any sublist as the input to PARTITION. Make sure the modified PARTITION still has the time complexity $O(n)$ on sublists of length n.

   (b) Based on the above claim, QUICK SORT can be modified accordingly so that elements of the same value as a pivot do not need to be considered in the subsequent levels of recursive calls. Show that if there are k distinct elements in the input list of n elements, the modified QUICK SORT has only k levels of recursive calls.

   (c) Argue that the worst case running time of the modified QUICK SORT can still be $\geq ckn$, for some constant $c > 0$, even if there are only k distinct elements in the input list of n elements.

2. (**20 points**) Given the same scenario described in **Q**1, i.e., the input list to be sorted by the modified QUICK SORT has only k distinct elements out of total n elements. Which of the following upper bounds is the best for the *average case time complexity* of the modified QUICK SORT algorithm? Explain. **Your explanation can be based on an**

intuitive analysis on the recursive tree or a more rigorous computation of the expected time with probability theory.

(a) $O(n \log_2 n)$ (b) $O(k \log_2 n)$ (c) $O(n \log_2 k)$ (d) $O(k \log_2 k)$

3. (**20 points**) Use the decision tree method to prove that searching an $n \times n$ matrix for a key has the lower bound $\Omega(\log_2 n)$ using the comparison-based model, regardless the elements in the matrix are sorted or not (and how they are sorted). Give a formal proof.

4. (**40 points**) Consider the problem to sort a list of n binary bits so that all bits of value 0 come before those of value 1.

   (a) Design a comparison-based algorithm for this sorting problem. What is the number of comparisons your algorithm uses?

   (b) Use the decision tree method to show a (non-trivial) lower bound for the problem.

5. (**20 points**) Consider the algorithm SELECT that finds the k^{th} smallest element in a given set. Argue that if, instead of making groups of 5 elements, we make groups of 3 elements, the analysis of the algorithm does not lead to the conclusion of the linear time complexity.

The answers must be word-processed or typed. You may substitute formulae and figures with hand-writings. Your submitted algorithms should be in the pseudo-code, not in any specific programming language. Answers deviating from these requirements will be returned without grading.

The answers must be the student's own work. Idea sharing and referencing to others' work (including those online) are not allowed. Plagiarism and other forms of academic dishonesty will be handled within the guidelines of the Student Handbook and reported to the University.