Homework Assignment No. 2

CSCI 4470/6470 Algorithms, CS@UGA, Spring 2018

Due Monday February 12, 2018

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.

Five questions with 120 points for graduate students and 110 points for undergraduates.

1. **(30 points)** In the HeapSort algorithm, the subroutine Build-Max-Heap builds the maximum heap with subroutine Max-Heapify, which is called $\frac{n}{2}$ times, from the non-leaf node of the highest index to the one with the lowest index (i.e., the root).

   (1) Show an example to justify that the algorithm would not work if the order of these calls to Max-Heapify were to be reversed.

   (2) To estimate a better time upper bound for subroutine Build-Max-Heap, we assume the height of node of index $i$ in the heap to be $h_i$. (Height of a node is defined as the minimum number of edges connecting the node to a leaf.) Then we know the time complexity for each call Max-Heapify($A, i$) is $\leq c \cdot h_i$.

   Prove that in the heap, there are roughly $\frac{n}{2}$ nodes with height 1, $\frac{n}{2^2}$ nodes with height 2, ..., $\frac{n}{2^k}$ nodes of height $k - 1$, where $k \leq \lceil \log_2 n \rceil$.

   (3) **(Graduate students only, bonus for undergraduate)** Use the estimation in (2) for heights of nodes in the heap to argue that the total time complexity for Build-Max-Heap is $O(n)$, instead of $O(n \log_2 n)$ shown in the lecture note.
2. **(20 points)** Consider that duplicate elements may occur in a list to be sorted by QUICK Sort.

   (1) Modify the pseudo code of function PARTITION so that all elements the same as the pivot will not be considered in subsequent recursive calls.

   (2) Argue that the worst case running time of the modified QUICK Sort is $O(kn)$ if there are only $k$ distinct elements in the input list.

3. **(20 points)** Use decision tree to prove that searching a sorted list with the comparison based model has the lower bound $\Omega(\log_2 n)$.

4. **(20 points)** Consider the SELECT that finds the $k$th smallest element in a given set. Show 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.

5. **(30 points)** Consider the Assembly Line problem where there are three assembly lines (instead of two).

   (1) Define an objective function for this problem.

   (2) Formulate a recursive solution for the objective function.

   (3) What would be the time complexity to compute this function?