CSci 231 Homework 4

Selection and Heaps

CLRS Chapter 6 and 9

Write and justify your answers on this sheet in the space provided.[1]

1. (CLRS 9.3-5) Suppose that you have a "black-box" worst-case linear-time median subroutine. Give a simple, linear-time algorithm that solves the selection problem for an arbitrary order statistic.

[1]Collaboration is allowed and encouraged, if it is constructive and helps you study better. Remember, exams will be individual. Write up the solutions on your own and list the names of the collaborators.

2. (CLRS 9.3-7) Describe an $O(n)$ algorithm that, given a set S of n distinct numbers and a positive integer $k \leq n$, determines the k numbers in S that are closest to the median of S.

3. 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. Your algorithm should use $O(1)$ additional space.

4. (CLRS 6.1-1) What are the minimum and maximum number of elements in a heap of height h? Note: the height of a heap is the number of edges on the longest root-to-leaf path.

5. (CLRS 6.1-4) Where in a min-heap might the largest element reside, assuming that all elements are distinct?

6. (CLRS 6.1-5) Is an array that is in sorted order a min-heap?

7. (CLRS 6.2-4) What is the effect of calling MIN-HEAPIFY(A, i) for $i > size[A]/2$?

8. (CLRS 6.5-3) Write pseudocode for the procedures HEAP-EXTRACT-MIN, HEAP-DECREASE-KEY and HEAP-INSERT that implement a min-priority queue with a min-heap.

9. (CLRS 6.5-8) Give an $O(n \lg k)$-time algorithm to merge k sorted lists into one sorted list, where n is the total number of elements in all the input lists. (*Hint: use a min-heap for k-way merging.*)

10. (CLRS 9.3-6) Give an $O(n \lg k)$ algorithm to find the $k-1$ elements in a set that partition the set into (approx.) k equal-sized sets $A_1, A_2, \ldots A_k$ such that all elements in A_i are smaller than all elements in A_{i+1}. Assume k is a power of 2.

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. Find 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.