Algorithms[1] Lab 3
Due: Tuesday, 8 October 2019 (afternoon), *at the beginning of class*

Notes on Grading

It is just as important to write in a style that is easy to understand and convincing as it is to get the correct answer. So, your assignment will be evaluated based not only on the correctness of your final answer, but also on clarity, neatness and attention to details.

When you describe an algorithm, use high-level pseudocode if you feel it's necessary for the reader's understanding. Focus on clarity, and **don't forget to argue why the algorithm computes what it's supposed to compute (correctness), and to analyze its running time. You need to do this even if the problem does not specifically ask for it.**

Topics Covered for This Lab

- QuickSort
- Heaps and HeapSort

In Lab Exercises (COLLABORATION LEVEL 0)

The in-lab problems are meant to be be solved during the lab and to generate discussion and sharing of ideas. Your answers will not be graded, but you do need to work through these problems. You can turn in a single writeup, with all your answers stapled together. List the people in the group on the first page.

Quicksort

1. What is the running time of QUICKSORT when all elements of array A have the same value? Assume that you are using the Lomuto partitioning algorithm, described in the next problem.

[1]CSCI 2200, Laura Toma and Stephen Majercik, Bowdoin College

2. Below is pseudocode for QuickSort, more detailed than the pseudocode we looked at in class. As usual with recursive functions on arrays, we see the array indices p and r as arguments. QUICKSORT(A, p, r) sorts the part of the array A between p and r inclusive. Note that the partition algorithm is somewhat different from the one I presented in class. By the way, this one is called the Lomuto partition algorithm, named after its inventor.

```
QUICKSORT(A, p, r)
if p < r
    q=PARTITION(A, p, r)
    QUICKSORT(A, p, q − 1)
    QUICKSORT(A, q + 1, r)
```

```
PARTITION(A, p, r)
x = A[r]
i = p − 1
for j = p to r − 1 DO
    if A[j] ≤ x
        i = i + 1
        swap A[i] and A[j]
swap A[i + 1] and A[r]
return i + 1
```

Let $A = \{3, 6, 1, 5, 8, 2, 4, 1, 3\}$, and assume we call QUICKSORT(A, 0, A.length − 1), i.e. QUICKSORT(A, 0, 8), Show what happens during the first invocation of PARTITION. What is the value of q returned, and what are the two recursive calls made?

Heaps

1. (CLRS 6.1-3) Where in a min-heap might the largest element reside, assuming that all elements are distinct? Why?

2. (CLRS 6.1-5) Is an array that is in sorted order a min-heap? Why or why not?

3. (CLRS 6.5-2) Illustrate the operation of `insertItem(7)` on the heap A, assuming A is a min-heap:

   $$A = \{2, 5, 10, 6, 8, 100, 11, 9, 15, 9, 10, 200, 101\}$$

   Be sure to indicate all the swaps performed.

4. (CLRS 6.3-1) Illustrate the operation, step by step, of `buildMinHeap` on the array

$$A = \{5, 3, 17, 10, 84, 19, 6, 22, 9\}$$

Do it for both the $O(n \lg n)$ approach that repeatedly inserts items and bottom-up $O(n)$ approach. Are the heaps the same? If not, is it a problem?

5. (CLRS 6.1-7) Argue that the leaves in a heap of n elements are the nodes indexed by $\lfloor n/2 \rfloor + 1, ..., \lfloor n/2 \rfloor + 2, ..., n$. (Note: $\lfloor x \rfloor$ is the "floor" of x, which is the largest integer less than or equal to x.) Assume you are using the array implementation of a heap.

6. How would you implement a function that searches for a given element in a min-heap of size n, and how long would it take in the worst case? Keep in mind that you can use only the operations allowed on a heap. And consider two cases: 1) destructive, i.e. the heap does not need to exist afterwards, and 2) non-destructive, i.e. the heap must still exist afterwards. Does it matter in terms of the aymptotic time bound? Finally, suppose the heap was implemented as an array and you had access to the underlying array. Would that change your answer?

Homework Problems (COLLABORATION LEVEL 1)

1. Design an algorithm that finds the kth smallest element in a a set of n distinct integers in $O(n + k \lg n)$ time.

2. (C-4.9) Suppose we are given a sequence S of n elements, each of which is colored red or blue. Assuming S is represented as an array, give an $O(n)$ in-place method for ordering S so that all blue elements are listed before all the red elements.

3. (CLRS 6.5-9) Assume you have k sorted lists containing a total of n elements, and you want to merge them together in a single (sorted) list containing all n elements. For simplicity you may assume that the k lists contain the same number of elements.

   (a) Approach 1: Merge list 1 with list 2, then merge the result with list 3, then merge the result with list 4, and so on. What is the worst-case running time ?

   (b) Approach 2: Split the k lists into two halves, merge each one recursively, then use the standard 2-way merge procedure (from MergeSort) to combine the two halves. What is the worst-case running time ?

   (c) Give another approach (to merge the k lists) that uses a heap, and runs in $O(n \lg k)$-time.

4. (CLRS 7-3) Professor Stooge has proposed the following "elegant" sorting algorithm:

   STOOGE-SORT(A, i, j)
   if $A[i] > A[j]$
   swap $A[i] \leftrightarrow A[j]$
   if $i + 1 \geq j$
   return
   $k = \lfloor (j - i + 1)/3 \rfloor$
   STOOGE-SORT$(A, i, j - k)$
   STOOGE-SORT$(A, i + k, j)$
   STOOGE-SORT$(A, i, j - k)$

   Note 1: This pseudocode uses 1-based indexing for A.
   Note 2: $\lfloor x \rfloor$ is the "floor" of x, which is the largest integer less than or equal to x.

   (a) Correctness:

      i. Argue that STOOGE-SORT$(A, 1, A.length)$ correctly sorts any array of one element.

      ii. Argue that STOOGE-SORT$(A, 1, A.length)$ correctly sorts any array of two elements.

      iii. Would the algorithm work if the first line (that possibly swaps elements $A[i]$ and $A[j]$) was missing?

      iv. Now argue that STOOGE-SORT$(A, 1, A.length)$ correctly sorts any array of three elements.

      v. Assume that STOOGE-SORT sorts correctly any array of $2n/3$ elements, and argue that this implies that it sorts correctly any array of n elements (What is true after the first recursive call? After the second?)

   (b) Running time: Give a recurrence for the worst-case running time of STOOGE-SORT and a tight asymptotic (Θ-notation) bound on the worst-case running time.