University of Waterloo
CS240 - Fall 2015
Assignment 2

Due Date: Wednesday October 14 at 5pm

Please read `http://www.student.cs.uwaterloo.ca/~cs240/f15/guidelines.pdf` for guidelines on submission. All problems are written problems; submit your solutions electronically as a PDF with file name `a02wp.pdf` using MarkUs. We will also accept individual question files named `a02q1w.pdf`, `a02q2w.pdf`, ... ,`a02q5w.pdf` if you wish to submit questions as you complete them.

There are 65 possible marks available. The assignment will be marked out of 60.

Problem 1 [5+5+5=15 marks]

Consider a very keen teaching assistant. As students arrive during office hour she assigns a priority number to each student corresponding to their time of arrival and inserts the pair (time, name) into a priority queue, implemented as a min-heap. The next student is determined with a call to `deleteMin`. The student then has five minutes to ask as many questions as possible before being re-inserted in the queue with the current time as priority.

Occasionally students leave before their turn comes up. Describe how to perform a delete key operation in a heap under each of the following assumptions:

a) The input for the delete operation is the student name only, assumed to be unique.

b) The input for the delete operation is the priority value in the heap, again assumed to be unique.

c) The input to the delete operation is the index of the entry in the array where the key resides.

In each case the priority queue is implemented as a heap using an array. After the delete operation the array should still be a heap. For each implementation discuss the time complexity of the operation in the worst case. The running time of all of your methods should be $O(n)$, and at least one of the methods should have running time $o(n)$.

Problem 2 [10 marks]

A sorting algorithm is said to sort *in place* if only a constant number of elements of the input are ever stored outside the array. In class we showed that all comparison based sorting algorithms require $\Omega(n \log n)$ comparisons to sort an array of length n. But suppose are given an array $A[0 \ldots n-1]$ that contains a permutation of the first n non-negative integers. Allowing non comparison based algorithms, give an $O(n)$ in place algorithm to sort A. Analyze the the running time of your method. *Note:* For simplicity we are assuming A is filled with integer keys. Your algorithm must easily extend to work for an array A that is filled with (key,element) pairs, each integer key in the range $0 \ldots n-1$ occurring exactly once.

Problem 3 [10 marks]

You are given an unsorted array $A[0 \ldots n-1]$ filled with distinct integers. For a given k, $1 \leq k \leq n$, describe an in-place algorithm that rearranges the array so that $A[0 \ldots k-1]$ contains the k smallest integers in increasing order. If $k \in O(n/\lg n)$ your algorithm should have running time $O(n)$. Analyze the running time of your algorithm.

Problem 4 [6+7+7=20 marks]

You have found a treasure chest filled with n coins, some of which are genuine and some of which are counterfeit. There is at least one genuine coin and at least one counterfeit coin in the chest. All genuine coins weigh the same and all counterfeit coins weigh the same, but the counterfeit coins weigh less than the genuine coins. Your task is to separate the genuine coins from the counterfeit coins. To accomplish this you will compare the weight of pairs of subsets of the coins using a balance scale. The outcome of one weighing will determine that each subset of coins weighs the same, or that one or the other subset of coins weighs more.

a) [6 marks] Give a precise (not big-Omega) lower bound for the number of weighings required in the worst case to determine which coins are genuine and which are counterfeit.

b) [7 marks] Describe an algorithm called `FindGenuine` to determine the genuine coins when n = 4. Use the names C1, C2, C3, C4 for the four coins, and the function

$$CompareWeight\left(\{first\_subset; second\_subset\}\right),$$

which returns either "first weighs more", "second weighs more", or "both weigh the same". Your function should return the set of genuine coins.

Give an exact worst-case analysis of the number of weighings required by your algorithm. For full marks, this should match exactly the lower bound from Part (a) when $n = 4$.

c) [7 marks] Describe an algorithm to determine the genuine coins, for any n. Use the names C1, C2, ... , Cn and the *CompareWeight* subroutine from Part (b). Show that your algorithm is asymptotically optimal, meaning that the big-O cost should match the lower bound from Part (a).

Problem 5 [4+6=10 marks]

A *deterministic* algorithm is one whose execution depends only on the input. By contrast, the execution of a *randomized* algorithm depends also on some randomly-chosen numbers. A *Las Vegas* randomized algorithm always produces the correct answer, but has a running time which depends on the random numbers chosen (randomized quick-select and quick-sort are of this type). Informally, such algorithms are always correct, and probably fast. A *Monte Carlo* randomized algorithm has running time independent of the random numbers chosen, but may produce an incorrect answer. Informally, such algorithms are always fast, and probably correct.

Given an array A of length n, an element x is said to be *dominant* in A if x occurs at least $\lfloor n/2 \rfloor + 1$ times in A. That is, copies of x occupy more than half of the array.

a) [4 marks] Given an array A that contains a dominant element, describe an in-place Monte Carlo randomized algorithm to find the dominant element. Show that your algorithm has worst-case running time $O(1)$ and returns the correct answer with probability at least $1/2$.

b) [6 marks] Given an array A that contains a dominant element, describe an in-place Las Vegas randomized algorithm to find the dominant element. Show that your algorithm always returns the correct answer, and has expected-case running time $O(n)$.