**ESO207A: Data Structures and Algorithms**

Homework *2* **Due: September 13, 2017, end of class.**

**Instructions.**

1. Start each problem on a new sheet. For each problem, Write your name, Roll No., the problem number, the date and the names of any students with whom you collaborated.
2. For questions in which algorithms are asked for, your answer should take the form of a short write-up. The first paragraph should summarize the problem you are solving and what your results are (time/space complexity as appropriate). The body of the essay should provide the following:
   (a) A clear description of the algorithm in English and/or pseudo-code, where, helpful.
   (b) At least one worked example or diagram to show more precisely how your algorithm works.
   (c) A proof/argument of the correctness of the algorithm.
   (d) An analysis of the running time of the algorithm.

   Remember, your goal is to communicate. *Full marks will be given only to correct solutions which are described clearly.* Convoluted and unclear descriptions will receive *low marks.*

---

**Problem 1. Heaps: Running Median.** (30)

The median of a set of $n$ numbers is the $(n+1)/2$th number, for $n$ odd, and it is the $\lfloor (n+1)/2 \rfloor$th ranked member and the $\lceil (n+1)/2 \rceil$ th ranked member (i.e., two medians), for $n$ even. The following operations on a dynamic set $S$ of elements have to be supported efficiently: INSERT($x$) and MEDIAN. The INSERT($x$) operation inserts a new value $x$ into $S$ and is supported in $O(\log n)$ time, where, $n = |S|$. The MEDIAN operation is supported in $O(1)$ time and returns the $(n+1)/2$th ranked member of $S$, if $n$ is odd, or returns the $\lfloor (n+1)/2 \rfloor$th and $\lceil (n+1)/2 \rceil$th ranked member of $S$, if $S$ is even. The rank 1 member is the smallest member of $S$ and rank $n$ member is the largest member of $S$, etc.. Give a correctness and complexity analysis of your algorithm.

(*Hint:* Keep the smaller half of the numbers (ranks $1 \ldots \lfloor (n+1)/2 \rfloor$) in one heap and the remaining higher ranked numbers in a second heap.)

**Problem 2. Stack** (30)

Given an array $A[1,\ldots,n]$ containing positive integers, for each $1 \le i \le n$, let $B[i]$ denote the largest number of consecutive indices $i - B[i] + 1, i - B[i] + 2, \ldots, i$ such that $A[i - B[i] + j] \le A[i]$, for $1 \le j \le B[i]$. The problem is to compute the $B[1,\ldots,n]$ array, given $A[1,\ldots,n]$. For example, given $A[6] = \{9, 5, 6, 11, 2, 13\}$, the output $B[6] = \{1, 1, 2, 4, 1, 6\}$. Design an algorithm that makes a single pass over the array $A$ and may keep $O(n)$ space extra storage.

(*Hint:* Define $H(i)$ to be the index $j \le i$ that is closest to $i$ such that $A[j] > A[i]$. If $H(i)$ is computed, then, the answer $B(i) = i - H(i)$. Assume, $H(1) = 0$, and for convenience, $A[0] = \infty$. Maintain the invariant: after processing $A[i]$, the top of stack contains $H(i)$.

**Problem 3. Queues: Breadth first search in a maze.** (40)

The input is a two dimensional $N \times N$ array $M[1 \ldots N, 1 \ldots, N]$ of tiles. A mouse is sitting initially at tile $M[1, 1]$. From tile numbered $[i, j]$, the mouse can navigate in *one step* to any of the four neighboring tiles at $[i + 1, j]$, $[i, j + 1]$, $[i - 1, j]$ and $[i, j - 1]$, provided, in each case, the mouse does not step out of the grid. Additionally, associated with each tile $M[i, j]$ there is a status flag $M[i, j].status$, which is 1 if the tile is "normal" and 0 if the tile is defective. If the tile $(i, j)$ is defective, then the mouse cannot navigate to it, that is, this tile cannot be reached from any of its neighbors. The exit from this maze is at position $[N, N]$. Given the input tile array $M$, design an algorithm that finds a shortest length path (in terms of number of steps) from the tile $[1, 1]$ to $[N, N]$ without stepping onto any defective tile. You algorithm should run in time linear in the size of the input. Give a correctness proof and a complexity analysis of your algorithm.

(*Note:* This is an application of the graph search method called breadth-first-search.