Problem Set 2

Data Structures and Algorithms, Fall 2019

Due: September 19, in class.

From CLRS

Exercise 10.1-6, 10.2-7, 2.3-7. Problem 2-4. Exercise 4.3-6, 4.4-5, 4.4-9, 4.5-4. Problem 4-2.

Additional Problem One

[This is NOT a bonus problem, and you ARE required to solve it.]

Design a MAXSTACK data structure that can store comparable elements and supports the stack operations `push(x)`, `pop()`, as well as the `max()` operation, which returns the maximum value currently stored in the data structure. All operations should run in $O(1)$ time. You should give a brief overview of your data structure, then provide pseudocode for each of the three operations. You should also briefly argue why your implementation is correct, and analyze the space complexity of your implementation.

Additional Problem Two

[This is NOT a bonus problem, and you ARE required to solve it.]

An array $A[1 \cdots n]$ is said to have a *majority element* if more than half of its entries are the same. Given an array, the task is to design an algorithm to tell whether the array has a majority element, and, if so, to find that element. The elements of the array are not necessarily from some ordered domain, and so there can be no comparisons of the form "is $A[i] > A[j]$?". (Think of the array elements as JPEG images, say.) However, you can answer questions of the form "is $A[i] = A[j]$?" in constant time.

(a) Show how to solve this problem in $O(n \lg n)$ time.
(b) Can you give a linear-time algorithm? *(Hint: You might find the following process helpful. First, pair up the elements of A arbitrarily to get $n/2$ pairs. Then, look at each pair: if the two elements are different, discard both of them; if they are the same, keep just one of them.)*