CSE 3500: Problem Set 4
Due by 11:59 PM on Monday, Oct 30.

Please note:

- *Students are permitted to discuss general concepts and questions concerning the homework assignments, but sharing written solutions with others or using solutions provided by others, in part or in whole, is prohibited.*
- *Whenever a question asks you to give an algorithm for a problem, be sure to also prove its correctness and analyze its time complexity.*
- *If you consult an outside resource (e.g., web page, book, or research paper) to arrive at your solution, be sure to cite that resource.*

Suggested reading: Chapter 5 from textbook.

Homework questions:

Question 1. (10 points) An array $A[1 \dots n]$ is said to have a *majority element* if more than half of its entries are the same. Suppose you are given array $A[1 \dots n]$ of n objects. These objects are not necessarily from some ordered domain, and so there can be no comparisons of the form "is $A[i] > A[j]$?". (The objects may be GIF files, for example.) However, you can answer questions of the form "is $A[i] = A[j]$?" in constant time. Give an $O(n \lg n)$-time algorithm to determine if the array A has a majority element.

Question 2. (10 points) Suppose we have k sorted arrays, each with n elements. We we want to merge these k arrays into a single sorted array with kn elements.
(a) One way to solve this problem is to use the merge procedure of merge-sort to first merge the first two arrays, then merge in the third array, and then the fourth array, and so on until all k arrays have been merged. What is the time complexity of this algorithm in terms of n and k?
(b) Use divide and conquer to give a more efficient solution for this problem.

Question 3. (10 points) Solve the following recurrences and give a Θ bound for each . You may use either the recursion tree technique or the master theorem to solve these recurrences. You may also assume that $T(1)$ is $\Theta(1)$ in each case.

a) $T(n) = 2T(n/2) + n^4$.
b) $T(n) = T(7n/10) + n$.
c) $T(n) = 16T(n/4) + n^2$
d) $T(n) = T(n-1) + 2$

Question 4. (10 points) Suppose you are given an infinite array $A[\cdot]$ in which the first n cells contain integers in sorted order and all the remaining cells are filled with ∞. You have not been given the value of n. Describe a $O(\log n)$-time algorithm that takes as input an integer x and finds a position in the array containing x, if such a position exists.

Question 5. (10 points) Exercise 2 from Chapter 5, pages 246 of the textbook.