CS 535 Homework 9
Due: April 19 (Th), in class.

Undergrads, please answer problems 1 and 2. Problem 3 is a bonus. Graduate students, please answer all of them.

1. C-4.19

2. An array $A[1 \ldots n]$ has a majority element if more than half of its entries are the same. Given such an array $A$, our goal is to design an efficient algorithm that will determine if $A$ has a majority element, and if so, to find that element.
   a. Suppose we simply use the following brute force method: For $i = 1$ to $n$, compute the number of times $A[i]$ appears in $A$. If the number is greater than $n/2$, return the number. Otherwise, once the for loop has ended and no number has been returned, return that $A$ has no majority element. What is the running time of this algorithm?
   b. As always, we want to do better than brute force. Suppose we take a divide-and-conquer approach. Split $A$ into two smaller arrays $A_1$ and $A_2$ whose sizes are half that of $A$. Determine, if any exists, the majority elements in $A_1$ and $A_2$.
      (i) How does the majority element of $A$ (if it exists) relate to the majority elements of $A_1$ and $A_2$ (if they exist)?
      (ii) Given your answer in (i), create a divide-and-conquer algorithm for our problem. What is the running time of your algorithm?

3. Let $T(n)$ denote the worst case running time of an algorithm when its input has size $n$. In divide and conquer algorithms, $T(n)$ is often expressed using a recursion. Hence, expressing $T(n)$ in terms of the big-Oh notation requires a bit of work. Now the “formula” for $T(n)$ can often be divided into two parts: the recursive part and the non-recursive part. To determine the growth rate of $T(n)$ then, we usually (1) draw the recursion tree out, (2) determine the non-recursive work that is done at each level, and then (3) add them all up. The final step often involves analyzing a geometric sum of the form $1 + a + a^2 + \ldots + a^r$. Recall that if $a < 1$, this sum is $O(1)$; if $a = 1$, this sum is equal to $O(r)$; if $a > 1$, this sum is $O(a^r)$.

   Using steps 1, 2, and 3, determine the growth rate of $T(n)$ using the big-Oh notation for each of the cases below.
   a. $T(n) = 3T(n/3) + O(n)$
   b. $T(n) = T(n/2) + O(n^2)$
   c. $T(n) = 8T(n/4) + O(n^2)$