Analysis of Algorithms
CS 375, Spring 2019

Homework 7

Due **AT THE BEGINNING OF CLASS** Monday, March 11

- From your textbook (CLRS), please read Chapter 4, pages 65–67 and 88–97.
- *A general note:* When writing up your homework, please write neatly and **explain your answers clearly**, giving all details needed to make your answers easy to understand. Graders may not award credit to incomplete or illegible solutions. Clear communication *is* the point, on every assignment.

Exercises

1. Solve the following recurrences **showing your work, using the recursion tree method** and give the Θ class of the solution.

   (a) $T(n) = 3T(n-1)$ for $n > 1$; $T(1) = 4$.

   (b) $T(n) = T(n/2) + n$ for $n > 1$; $T(1) = 1$. (Assume n is of the form 2^k for this exercise.)

2. An algorithm to compute 2^n would be defined by the following specifications:

   ```
   // Input: n, a non-negative integer (i.e., 0 or greater)
   // Output: 2^n (i.e., 2 to the n'th power)
   ```

   (a) Write a recursive algorithm for computing 2^n for any non-negative integer n **that is based on the formula** $2^n = 2^{n-1} + 2^{n-1}$. (This may not be the most natural way to think of an algorithm to compute 2^n, but please be sure to use it for this exercise!)

   (b) Set up a recurrence for the number of additions made by the algorithm on input n and solve it **using the recursion tree method**.

   (c) What is the Θ complexity class of this algorithm? If the above recurrence is helpful in this, explain why; if not, set up a different recurrence for the runtime of this algorithm and solve it **using a method of your choice** to get the complexity class.

   (d) Is this a good algorithm for computing 2^n? As always, be sure to explain your answer!

3. Prof. Nigma at the Portland Institute of Technology (motto: "We don't think about acronyms!") hired you to analyze the algorithm given here in pseudocode, but as usual, Prof. Nigma neglected to explain what the algorithm does.

   ```
   def Q($A[0..n-1]$)
   // Input: Array $A[0..n-1]$ of $n$ real numbers, for $n \geq 1$
       if $n = 1$ return $A[0]$
       else $temp = Q(A[0..n-2])$
           if $temp \leq A[n-1]$ return $temp$
           else return $A[n-1]$
   ```

(a) What does this algorithm compute? Give an English description of what value it returns on a given input array. (You do not need to give examples as part of your answer, but as always, you are welcome to include examples along with the English description, if it would make your answer clearer.)

(b) Set up a recurrence for an exact count of the number of comparison operations $\leq$ performed by the algorithm, and solve it using the recursion tree method, showing your work.

(c) What is the Θ complexity class of this algorithm? If the above recurrence is helpful in this, explain why; if not, set up a different recurrence for the runtime of this algorithm and solve it **using a method of your choice** to get the complexity class.