Analysis of Algorithms
CS 375, Fall 2018

Homework 1

Due **AT THE BEGINNING OF CLASS** Wednesday, September 12

- From your textbook (Levitin), please read Chapter 1, Sections 2.1–2.3, and Appendix A (formulas for logarithms and sums).
- Unless otherwise specified, exercises will be from Levitin and will be named on HW assignments by the Section and exercise number in the book. For instance, Exercise 1.2.9 below refers to Levitin, Section 1.2, exercise 9.
- *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.

1. Exercise 1.1.5
2. [**NOTE: Restricted collaboration**] Exercises 1.2.1 and 1.2.2. *Explain the full thought process by which you arrived at your answer!* Or, if you aren't able to find a full answer, just explain the thought process as far as you get with your reasoning.

   Credit for these two exercises (unlike many other exercises) will not depend much on whether your answer is correct—instead, these are about thinking through a problem and clearly expressing the design process for your solution. For example, one might say "First, we thought about sending [blah blah blah], but we then realized [blah blah blah]. Then, to address that, we thought [blah blah blah], but that didn't work because of [blah blah blah]. Because of that, we . . . ". (Please try to avoid using the word "blah" in your answer!)

   NOTE: For these exercises, please do not collaborate with anyone (other than the other person with whom you're working on this assignment, if you choose to do so)—that is, for Exercises 1.2.1 and 1.2.2, please do not ask friends / classmates / TAs / Professors / etc. for assistance. Don't worry if you don't find correct solutions to these problems on your own, that's absolutely fine—all that matters is that you put thought into them and clearly express your full thought process.

   If you have any questions about what's being asked, though, please feel free to ask your prof.! These are classic puzzles—I hope you have some fun with them!
3. Exercise 1.2.9
4. Exercise 1.3.6
5. Exercise 1.4.2
6. Exercises 2.1.3 and 2.1.7

7. The following Python implementation of the binary search algorithm takes too long—it is much slower than binary search should be with a sorted list L! Why does it take so long? What is its actual time complexity?

```
def binarySearch(x,L):
    if L == []:
        return False
    else:
        midValue = L[len(L)/2]
        if x < midValue:
            return binarySearch(x,L[0:len(L)/2])
        elif x > midValue:
            return binarySearch(x,L[len(L)/2+1:len(L)])
        else:
            return True
```