CS 312: Algorithms **Fall 2018**

Homework 7

Your Name: ____________________

Collaborators and sources: ______________________________________

Instructions. You may work in groups, but you must write solutions yourself. List collaborators on your submission.

If you are asked to design an algorithm, please provide: (a) either pseudocode or a precise English description of the algorithm, (b) an explanation of the intuition for the algorithm, (c) a proof of correctness, (d) the running time of your algorithm and (e) justification for your running time analysis.

Submission instructions. This assignment is due by noon on Thursday, November 8 in Gradescope (as a pdf file). Please review the course policies on the course home page about Gradescope submissions.

1. **(5 points) Proof by Induction for Recurrences.** *(Work independently)* Prove the second case of the Master Theorem by induction. Suppose that

$$T(n) \leq aT(n/b) + cn^d$$

and $\log_b a = d$ with the base case that $T(b) \leq c$. Prove that $T(n)$ is $O(n^d \log_b n)$.

2. **(10 points) Independent Set.** K&T, Chapter 6, Exercise 1. *(Work independently at least through counterexamples.)*

3. **(10 points) Longest Increasing Subsequence.** In the *longest increasing subsequence problem*, you are given as input an unsorted array A of length n, e.g,

$$A = 5, 2, 10, 3, -1, 6, 8, 9, 3$$

The goal is to find the longest strictly increasing subsequence of A. The subsequence need not be contiguous. For example, the boxed numbers below indicate the longest increasing subsequence in our example:

$$A = 5, \boxed{2}, 10, \boxed{3}, -1, \boxed{6}, \boxed{8}, \boxed{9}, 3$$

To approach this problem, it is first helpful to define a "helper" function LIS(j) to compute the length of the longest increasing subsequence that ends at index j (and *includes* item $A[j]$). Here are examples for $j = 3$ and $j = 5$:

$$5, \boxed{2}, \boxed{10} \qquad\qquad 5, 2, 10, 3, \boxed{\text{-1}}$$

Therefore LIS(3) should return 2, and LIS(5) should return 1.

   (a) Write a recursive algorithm for LIS(j)

   (b) Translate this recursive algorithm into a recurrence. Define OPT(j) to be the length of the longest increasing subsequence ending at index j, and write a recurrence for OPT(j).

   (c) Use this recurrence to write an iterative algorithm to compute the value of OPT(j) and store it in the array entry $M[j]$ for all j.

   (d) Use the computed optimal values to find the value of the overall longest increasing subsequence (ending at any j).

4. **(0 points).** How long did it take you to complete this assignment?