University of Nevada, Las Vegas Las Vegas Computer Science 477/677 Fall 2019
Assignment 7 Due Monday November 4, 2019

Name:____

You are permitted to work in groups, get help from others, read books, and use the internet. But the handwriting on this document must be your own. Print out the document, staple, and fill in the answers. You may attach extra sheets, but only by stapling. Turn in the pages to the graduate assistant at the beginning of class, November 4.

1. Design a dynamic programming algorithm that inputs a sequence of integers, and finds that subsequence which has the largest total, subject to the condition that no two consecutive terms of the original sequence are in the subsequence. Identify the subproblems.

   For example, if the input sequence is 3,1,4,1,5,9,2,6,5,3,5, your algorithm will find the subsequence 3,4,9,6,5, whose terms add up to 27.

2. Work problem 5.11 on page 149 of your textbook. Your answer should consist of a sequence of pictures.

3. Work problem 6.1 on page 177 of your textbook.

   The book says to find a linear time algorithm. However, that is tricky. There is a simple $O(n^3)$-time algorithm, there is a not quite so simple $O(n^2)$-time algorithm, a much more sophisticated $O(n \log n)$-time algorithm, and finally a really clever $O(n)$-time dynamic programming algorithm. Do the best you can.

4. Work problem 6.4 on page 178 of the textbook. Although it might not seem like it, this problem is actually quite easy, but only after you successfully work the first problem in this assignment.

5. Work problem 6.11 on page 180 of your textbook. This is a well-known problem, called the LCS (longest common subsequence) problem. You can probably find solutions on the internet.

6. Work problem 6.20 on page 182 of your textbook. This is a well-known problem with an interesting history. The "obvious" (but still tricky) dynamic programming algorithm, which involves finding optimal binary search trees for each contiguous subsequence of the list, takes $O(n^3)$ time. In 1971, Donald Knuth found an $O(n^2)$-time algorithm for the problem. The algorithm was correct, but Knuth's proof of correctness was flawed.

   "Experts" believe there should be a faster algorithm, but no one has found one.