

Practice Assignment 1: Dynamic Programming

Attempt to solve the following four problems using **dynamic programming**. Your solutions (or partial solutions) should be typeset and saved as a pdf. Your solutions should be uploaded to Canvas (<https://canvas.oregonstate.edu>) by **Thursday January 19 at 10:00 AM**. Your solutions will be graded on effort alone: if you are unable to come up with a complete solution, then indicate your ideas toward generating a complete solution.

A complete solution to a problem will include the following elements:

- a recursive formulation of the solution to the problem
- an explanation *or* formal proof of why that formulation is correct
- pseudocode showing how to compute the solution in a bottom-up dynamic-programming way
- an analysis of the running time.

1. Suppose you are given an array $A[1..n]$ of integers, which may be positive, negative, or zero. Describe a linear-time (i.e. $O(n)$ -time) algorithm that finds the largest sum of elements in a contiguous, non-empty subarray $A[i..j]$. For example, given the array $[-6, 12, -7, 0, 14, -7, 5]$ as input, your algorithm should return the integer 19 (the sum of $[12, -7, 0, 14]$).

For the sake of analysis, assume that comparing, adding, or multiplying any pair of numbers takes $O(1)$ time.

2. String A is a *supersequence* of string B if string B can be obtained from string A by removing letters. For example, the strings BARNYARDSNACK, YUMMYBANANAS, and BWANWANWA are supersequences of the string BANANA.

Give a dynamic program for finding the length of the shortest string that is a supersequence of two input strings A and B .

3. Find the length (number of edges) of the longest path in a binary tree.
4. Suppose you are given an $n \times n$ bitmap, represented by a 2-dimensional array $M[1..n, 1..n]$ of 0s and 1s. A *solid block* in M is a subarray of the form $M[i..i', j..j']$ containing only 1s. Describe an algorithm to find the area of the maximum solid block in M in $O(n^3)$ time. If you can do that, try to design a faster algorithm that runs in $O(n^2)$ time.