University of Washington
Department of Computer Science and Engineering
CSE 417, Winter 2020

February 27, 2020

Homework 8, Due Wednesday, March 4, 2020

On all problems provide justification of your answers. Provide a clear explanation of why your algorithm solves the problem, as well as a justification of the run time. Since this assignment is from the dynamic programming section - your algorithms should use dynamic programming!

Problem 1 (10 points) Homework optimization:

Based on problem 20, Page 329 from the text, with a simplification: Suppose it is nearing the end of the quarter and you are taking n courses, each with a final project that still has to be done. Each project will be graded on the following scale: It will be assigned an integer number on a scale of 1 to g, higher numbers being better grades. Your goal is to maximize your average grade on the n projects.

You have a total of H hours in which to work on the n projects cumulatively, and you want to decide how to divide up this time. For simplicity, assume H is a positive integer and you'll spend an integer number of hours on each project. To figure out how best to divide up your time, you have come up with a set of functions $\{f_i : i = 1, 2, \ldots, n\}$ for each of your n courses; if you spend $h \leq H$ hours on the project for course i, you'll get a grade of $f_i(h)$. (You may assume that the functions f_i are non-decreasing.)

The problem is: Given these functions $\{f_i\}$, decide how many hours to spend on each project (in integer values only) so that your average grade, as computed according to the f_i, is as large as possible. In order to be efficient, the running time of your algorithm should be polynomial in n, g, and H; none of these quantities should appear as an exponent in your running time.

Now to simplify the problem - you only signed up for $n = 4$ courses, so you just need to consider functions f_1, f_2, f_3, and f_4. Give an algorithm to determine how much time to spend on each assignment that runs in $O(H^2)$ time.

Problem 2 (10 points) Strict Subset Sum:

The strict subset sum problem is: Given a set of values $\{s_1, \ldots, s_n\}$, and an integer K, is there a subset of the items that sum to exactly K. Design an algorithm that solves the strict subset sum, and finds a set that sums to K with as large a number of items as possible. Your algorithm should have runtime polynomial in n and K.

Problem 3 (10 points) Counting solutions to the subset sum:

The subset sum counting problem is: Given a set of values $S = \{s_1, \ldots, s_n\}$, and an integer K, determine the number of subsets of S that sum to exactly K. Design an algorithm that solves the subset sum counting problem. Your algorithm should have runtime $O(nK)$.

Problem 4 (20 points) Programming: Electoral College Ties:

Determine how many different ways the Electoral College can result in a 269-269 tie in the 2020 US Presidential election.

How the electoral college works: Each US state plus the District of Columbia has a given number of delegates based on its population. An election is held in each state and the winner of that election receives all of the delegates for that state. The person receiving the largest number of delegates is then the president of the US. (This method has the possibility that the person elected president is not necessarily the person winning the most votes nationally.)

For this problem, you are given a list of the number of votes each state has in the electoral college, and you are asked to compute the number of ways that these votes can be allocated to reach a 269-269 tie. We are assuming that there are only two candidates, and that states allocate all of their votes to one candidate or the other.

Obviously, use dynamic programming. There are lots of different ways of reaching a tie - so many that you will need to use 64-bit integers (e.g., long ints).

The data is available here so you can copy the arrays into your program.

a.) How many ways are there for the electoral college to result in a 269-269 tie.

b.) Find a group of states that can reach exactly 269 votes.

c.) Provide your algorithmic code.

d.) What is the runtime of your algorithm (as a function of the number of states, and of the number of electoral votes).

e.) Provide a justification for items a, b, and d.