CS 170 HW 7

Due on 2018-10-14, at 9:59 pm

1 (★) Study Group

List the names and SIDs of the members in your study group.

2 (★★) Copper Pipes

Bubbles has a copper pipe of length n inches and an array of nonnegative integers that contains prices of all pieces of size smaller than n. He wants to find the maximum value he can make by cutting up the pipe and selling the pieces. For example, if length of the pipe is 8 and the values of different pieces are given as following, then the maximum obtainable value is 22 (by cutting in two pieces of lengths 2 and 6).

length	1	2	3	4	5	6	7	8
price	1	5	8	9	10	17	17	20

Give a dynamic programming algorithm so Bubbles can find the maximum obtainable value given any pipe length and set of prices. Clearly describe your algorithm, prove its correctness and runtime.

3 (★★★) A Dice Game

Consider the following 2-player game played with a 6-sided die. On your turn, you can decide either to roll the die or to pass. If you roll the die and get a 1, your turn immediately ends and you get 1 point. If you instead get some other number, it gets added to a running total and your turn continues (i.e. you can again decide whether to roll or pass). If you pass, then you get either 1 point or the running total number of points, whichever is larger, and it becomes your opponent's turn. For example, if you roll 3, 4, 1 you get only 1 point, but if you roll 3, 4, 2 and then decide to pass you get 9 points. The first player to get to N points wins, for some positive N.

Alice and Bob are playing the above game. Let $W(x, y, z)$ be the probability that Alice wins given that it is currently Alice's turn, Alice's score (in the bank) is x, Bob's score is y and Alice's running total is z.

(a) Give a recursive formula for the winning probability $W(x, y, z)$.

(b) Based on the recursive formula you gave in the previous part, design an $O(N^3)$ dynamic programming algorithm to compute $W(x, y, z)$. Briefly describe your algorithm, prove its correctness and runtime.

4 (★★★) Road Trip

Suppose you want to drive from San Francisco to New York City on I-80. Your car holds C gallons of gas and gets m miles to the gallon. You are handed a list of the n gas stations that are on I-80 and the price that they sell gas. Let d_i be the distance of the i^{th} gas station from SF, and let c_i be the cost of gasoline at the i^{th} station. Furthermore, you can assume that for any two stations i and j, the distance $|d_i - d_j|$ between them is divisible by m. You start out with an empty tank at station 1. Your final destination is gas station n. You may not run out of gas between stations but you need not fill up when you stop at a station, for example, you might to decide to purchase only 1 gallon at a given station.

Find a polynomial-time dynamic programming algorithm to output the minimum gas bill to cross the country. Clearly describe your algorithm and prove its correctness. Analyze the running time of your algorithm in terms of n and C. You do not need to find the most efficient algorithm, as long as your solution's running time is polynomial in n and C.

5 (★★★★) Non-Prefix Code

As we have learned in lecture, the Huffman code satisfies the *Prefix Property*, which states that the bit string representing each symbol is not a prefix of the bit string representing any other symbol. One nice property of such codes is that, given a bit string, there is at most one way to decode it back to a sequence of symbols. However, this is not true anymore once we are working with codes that do not satisfy the Prefix Property. For example, consider the code that maps A to 1, B to 01 and C to 101. A bit string 101 can be interpreted in two ways: as C or as AB.

Your task is to, given a bit string s, determine how many ways one can interpret s. The mapping from symbols to bit strings of the code will be given to you as a dictionary d (e.g., in the example, $d = \{A : 1, B : 01, C : 101\}$); you may assume that you can access each symbol in the dictionary in constant time. Your algorithm should run in time at most $O(nm\ell)$ where n is the length of the input bit string s, m is the number of symbols, and ℓ is an upper bound on the length of the bit strings representing symbols.

Clearly describe your algorithm, prove its correctness and runtime.

6 (★) A HeLPful Introduction

Find necessary and sufficient conditions on real numbers a and b under which the linear program

$$\begin{aligned} \max\ & x + y \\ & ax + by \leq 1 \\ & x, y \geq 0 \end{aligned}$$

(a) Is infeasible.

(b) Is unbounded.

(c) Has a unique optimal solution.

7 (★★) Spaceship

A spaceship uses some *oxidizer* units that produce oxygen for three different compartments. However, these units have some failure probabilities.

Because of differing requirements for the three compartments, the units needed for each have somewhat different characteristics.

A decision must now be made on just *how many* units to provide for each compartment, taking into account design limitations on the *total* amount of *space*, *weight* and *cost* that can be allocated to these units for the entire ship. Specifically, the total space for all units in the spaceship should not exceed 500 cubic inches, the total weight should not exceed 200 lbs and the total cost should not exceed $400,000$ dollars.

The following table summarizes the characteristics of units for each compartment and also the total limitation:

	Space (cu in.)	Weight (lb)	Cost ($)	Probability of failure
Units for compartment 1	40	15	30,000	0.30
Units for compartment 2	50	20	35,000	0.40
Units for compartment 3	30	10	25,000	0.20
Limitation	500	200	400,000	

The objective is to *minimize the probability* of all units failing in all three compartments, subject to the above limitations and the further restriction that each compartment have a probability of no more than 0.05 that all its units fail.

Formulate the *integer programming model* for this problem. An integer programming model is the same as a linear programming model with the added functionality that variables can be forced to be integers.

Side note: Integer programming is often intractable, so we use a linear program as a heuristic. We can take out the integrality constraints and change our model into a linear program. We then round the solution and make sure none of constraints have been violated (you should think about why this won't always give us the optimum)