CS 170 Homework 7

Due **3/13/2022, at 10:00 pm (grace period until 11:59pm)**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write "none".

2 Copper Pipes

Bubbles has a copper pipe of length n inches and an array of nonnegative integers that contains prices of all integer-length pieces of size at most 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 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.

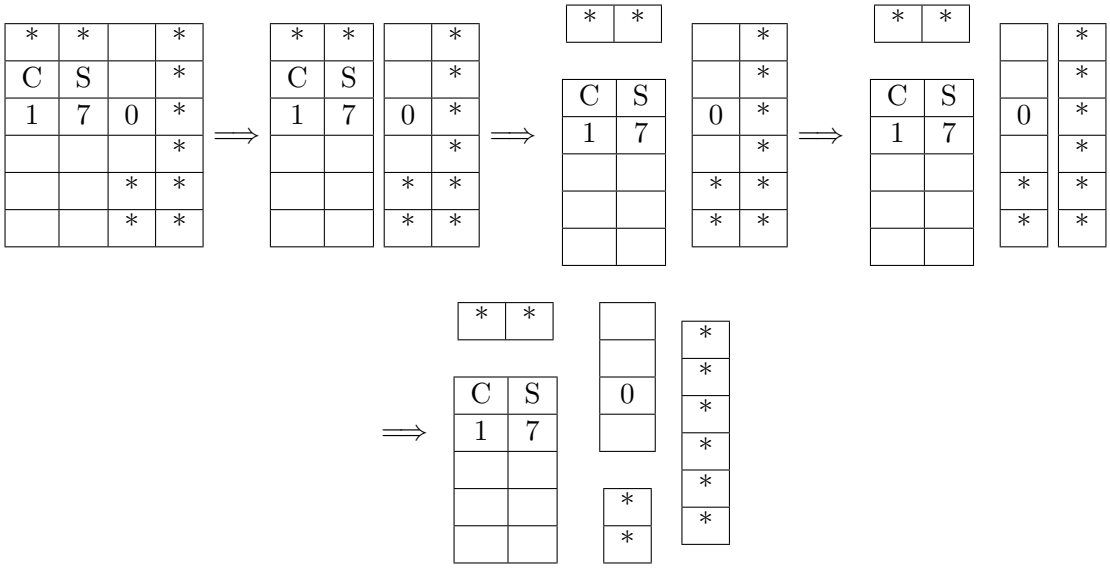
Your task is to, given a bit string s, determine how many ways one can interpret s. The mapping from symbols to bit strings of the code will be given to you as a dictionary d (e.g., in the example, $d = \{A : 1, B : 01, C : 101\}$); you may assume that you can access each symbol in the dictionary in 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 and provide a runtime analysis.

4 Paper Cutting

One morning, you spilled coffee on your CS 170 homework paper, which is an $m \times n$ rectangular grid of squares. Some of the squares have coffee stains on them, and you cannot use paper with a stain on it. You would like to cut the paper into pieces so as to separate all the squares with coffee stains from all the clean squares. You want to do this so that you can save as much as your work as possible.

For example, shown below is a 6×4 piece of paper with stains on squares marked *. As shown in the picture, one can separate the stains out in exactly four cuts.



(Each *cut* is either horizontal or vertical, and of one piece of paper at a time.)

Design a DP based algorithm to find the smallest number of cuts needed to separate all the stains out. Formally, the problem is as follows:

Input: Dimensions of the paper $m \times n$ and an array $A[i, j]$ such that $A[i, j] = 1$ if and only if the ij^{th} square has a stain.

Goal: Find the minimum number of cuts needed to separate the stains out.

(a) Define your subproblem.

(b) Write down the recurrence relation for your subproblems. A fully correct recurrence relation will always have the base cases specified.

(c) What is the time complexity of solving the above mentioned recurrence? Provide a justification.

5 Cards in a Line

You may have seen the idea of minimax decision making in two-player games in artificial intelligence classes. In this problem, we will see that for simple games, we can efficiently compute the minimax strategy using dynamic programming.

Alice and Bob are playing a game where there are n cards in a line, and the ith card is worth $p_i \geq 0$ points. Alice goes first, and Alice and Bob take turns either picking up the first or last card remaining in the line. For example, Alice can take the 1st or nth card on her first turn. If she takes the 1st, on Bob's first turn, he can take the 2nd or nth card.

(a) Alice decides to use a greedy strategy: "on my turn, I will take whichever of the first and last card is worth more points". Show a small counterexample ($n \leq 5$) where Alice will lose if she plays this greedy strategy, assuming Alice goes first and Bob plays optimally, but she could have won if she had played optimally.

(b) After seeing your counterexample, Alice decides to use dynamic programming to play optimally, assuming Bob also plays optimally. Write a dynamic program that computes how many points Alice ends up with, and state its runtime.

(Hint: Let $s(i, j)$ denote Alice's score at the end of the game if the game consists only of cards i to j and Alice goes first. To write a recurrence for $s(i, j)$, first think about what Bob should do on his turn. Then use that to decide what Alice should do on her turn).