CS 170 HW 7

Due 2019-10-16, at 10:00 pm

1 Study Group

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

2 Money Changing Redux

During discussion section, we saw a simple greedy algorithm to try to find change that adds up to a given number. We saw that the greedy algorithm didn’t find the optimal solution in all cases. In this problem, we will use our newly-found powers of computer science to fix this. Recall that in the money-changing problem, we were given a fixed set of positive integers called denominations \(x_1, x_2, \ldots, x_n\) (think of them as the integers 1, 5, 10, and 25). The problem you want to solve for these denominations is the following: Given an integer \(A\), express it as

\[ A = \sum_{i=1}^{n} a_i x_i \]

for some nonnegative integers \(a_1, \ldots, a_n \geq 0\). Find the way to do this using the minimum number of coins, e.g. so that \(\sum_{i=1}^{n} a_i\) is as small as possible.

(a) You might remember that we can represent any integer \(k\) in unary form by repeating \(k\) consecutive 1s (e.g., 3 is represented by 111 in unary). Assume you are given a positive integer \(A\) and a set of denominations \(x_1, x_2, \ldots, x_n\) in unary form. Give a fast algorithm to solve the money-changing problem.

(Please provide a 3-part solution)

(b) If you are given \(A\) and \(x_1, x_2, \ldots, x_n\) in binary, does your algorithm still run in polynomial time? Why or why not?

3 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.

Please provide a 3-part solution. Clearly describe your algorithm and prove its correctness. Analyze the running time of your algorithm in terms of \(n\) and \(C\).
4 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) Prove correctness of your algorithm by induction.

   (i) Show that the base case is correct.

   (ii) Assuming that \( W(x, y, z) \) is computed correctly for previous subproblems, show that it is correct for the current subproblem.

   (iii) Give the order where you can solve the subproblems. Show that for this order, evaluating the recurrence relation will use only subproblems that have already been computed.

(c) Find the runtime of your algorithm.

5 Propositional Parentheses

You are given a propositional logic formula using only \( \land \), \( \lor \), \( T \), and \( F \) that does not have parentheses. You want to find out how many different ways there are to correctly parenthesize the formula so that the resulting formula evaluates to true.

A formula \( A \) is correctly parenthesized if \( A = T \), \( A = F \), or \( A = (B \land C) \) or \( A = (B \lor C) \) where \( B \), \( C \) are correctly parenthesized formulas. For example, the formula \( T \lor F \lor T \land F \) can be correctly parenthesized in 5 ways:

\[
(T \lor (F \lor (T \land F))) \quad (T \lor ((F \lor T) \land F)) \quad ((T \lor F) \lor (T \land F)) \\
(((T \lor F) \lor T) \land F) \quad ((T \lor (F \lor T)) \land F)
\]

of which 3 evaluate to true: \( ((T \lor F) \lor (T \land F)) \), \( (T \lor ((F \lor T) \land F)) \), and \( (T \lor (F \lor (T \land F))) \).

(a) Give a dynamic programming algorithm to solve this problem.

(Please provide a 3-part solution)

(b) Briefly explain how you could use your algorithm to find the probability that, under a uniformly randomly chosen correct parenthesization, the formula evaluates to true.
6 Knightmare

Give an algorithm to find the number of ways you can place knights on an $N$ by $M$ ($M < N$) chessboard such that no two knights can attack each other (there can be any number of knights on the board, including zero knights). Clearly describe your algorithm and prove its correctness. The runtime should be $O(2^{3M} M \cdot N)$.

(Please provide a 3-part solution)