Practice Problem Set 9

Problem 1

Suppose the price of the different size cuts for the rod-cutting problem are as follows: $p_1 = 1, p_2 = 3, p_3 = 3, p_4 = 9, p_5 = 12, p_6 = 12, p_7 = 14, p_8 = 17$ Determine the price per "foot" for each of the above pieces. For example, a piece of size 1 is worth $\$1/foot$. A piece of size 2 is worth $\$1.5/foot$. What is the most valuable piece size (per foot?). Explain why cutting a rod "greedily" using the most valuable piece size does not always give the optimal solution.

Problem 2

The rod cutting problem from class finds the maximum profit obtainable over all ways of cutting a rod of length n into pieces and selling those pieces. In this question we would like to count the number of different *ways* of cutting the rod. Note that the order of the pieces doesn't matter. For example, a rod of length 5 can be cut into pieces in 7 different ways:

$$\{5\}, \{4, 1\}, \{3, 2\}, \{3, 1, 1\}, \{2, 2, 1\}, \{2, 1, 1, 1\}, \{1, 1, 1, 1, 1\}$$

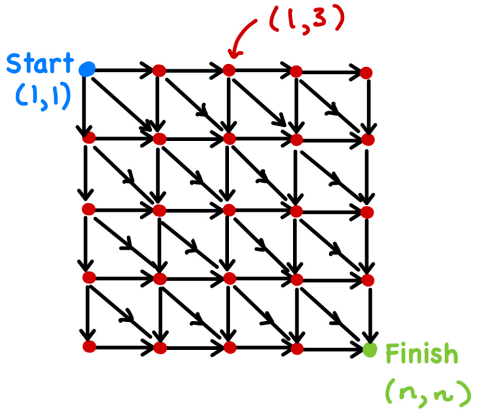
Provide a dynamic programming solution for determining the number of different ways we can cut a rod of length n into pieces (where the order of the pieces doesn't matter). What is the runtime of your algorithm?

Problem 3

- Suppose the rod-cutting problem has a price list that only consists of certain piece-sizes. In the current version of rod-cutting, all piece-sizes have an associated cost. Suppose instead you are given a list p of prices that correspond to certain sizes: (ex. $p_1 = 1, p_4 = 6, p_7 = 12$) where some of the piece sizes may be even larger than your log. How would you use the rod-cutting algorithm from class to solve this problem? Do not write a new dynamic programming solution.
- A certain river runs north-south. A set of n cities is built along the east riverbank. A set of m cities is built along the west riverbank. The terrain is not flat, and each city has an associated altitude. Bridges can be built between cities on opposite sides of the river as long the two cities have the exact same altitude. The goal is to build as many bridges as possible, in such a way that each city is connected to at most one other city and certainly bridges can't "cross over" each other. Explain which dynamic programming problem from class can be used to model this problem, and justify your answer. What is the runtime of finding a solution via this dynamic programming algorithm and why?

Problem 4

A city map is an $n \times n$ grid consisting of a set of one-way roads connected by toll booths. An example is shown below for $n = 5$. A driver must get from position $(1, 1)$ to position (n, n). Each dot in the diagram represents a toll booth. The toll at position (i, j) is exactly $t(i, j)$ dollars. Describe a dynamic programming algorithm that finds the minimum total toll over all routes from start to the finish location. You must describe the table, the algorithm and justify the runtime. Describe the algorithm to output this least-expensive route.



Problem 5

- Fill out the dynamic programming table for the longest common subsequence between the two sequences shown below. Show how to reconstruct the solution using the table.

  `YTUAWAXTT`
  `UWYTRAXTAAX`

- Fill out the dynamic programming table (both r and s) for the rod cutting problem where $n = 8$ and the price list is $p_1 = 1, p_2 = 3, p_3 = 3, p_4 = 9, p_5 = 12, p_6 = 12, p_7 = 14, p_8 = 17$. Show how to reconstruct the solution from this table.

Problem 6

Suppose in year 1, a woman invests 100 in her bank account. In year 2 she deposits an additional 400. For every year after the bank adds 50% interest, and immediately afterwards she withdraws exactly *half* of what her balance was two years earlier.

Use a dynamic programming algorithm to determine her balance in year n. What is the runtime of your algorithm?

Provide a recursive solution. What is the runtime recurrence for this algorithm? Which solution is asymptotically faster?

Problem 7

Suppose a group of n people are lined up in a row in increasing order of age. The heights of the people vary. A photographer would like to select a set of people who can be lined up in such a way that their ages are increasing and their heights are also increasing. He would like to line to have as many people in it as possible. Your job

is to solve this problem with dynamic programming. The input is an array $H[1, \ldots, n]$ representing heights the n people, where the people are already arranged by age. For example $H[i]$ is the height of the ith person, and person i is younger than person $i+1$. The output must be the longest line possible that satisfies the photographers requirements.