CSE 6140 / CX 4140

Computational Science & Engineering Algorithms

Homework 2

Please type in all answers.

1. (15 points) You are managing a consulting team. You need to plan their schedule for the year. For each week, you can either assign them a low-stress job or a high-stress job.

You are given arrays l and h. $l[i]$ is the revenue you make by assigning a low stress job in week i. $h[i]$ is the revenue you make by assigning a high stress job in week i.

In order for the team to take on a high-stress job in week i, it's required that they do no job (of either type) in week $i-1$; they need a full week of prep time for the high-stress job. On the other hand, it is okay for them to take a low-stress job in week i even if they have done a job (or either type) in week $i-1$.

Write pseudocode of an algorithm that finds the maximum revenue you can make for the year.

Hint: Use dynamic programming. For week i, assume that you know the maximum revenue achieved up to weeks $i-1$ and $i-2$. Let's say these are $OPT(i-1)$ and $OPT(i-2)$ respectively. In week i you can either choose a low stress job (revenue $l[i]$) or high-stress job (revenue $h[i]$). You want to find a recurrence that relates $OPT(i)$ to $OPT(i-1)$ and $OPT(i-2)$.

2. (10 points) Consider a modification to the rod-cutting problem we discussed in class. In addition to a price p_i for each rod, each cut now incurs a fixed cost of c. The revenue associated with a solution is now the sum of the prices of the pieces minus the costs of making the cuts. Give a dynamic-programming algorithm to solve this modified problem. (Hint: modify the Bottom-Up-Cut-Rod algorithm we discussed in class)

3. (15 points) You are given an integer array coins representing coins of different denominations and an integer amount representing a total amount of money.

Write pseudocode of a bottom-up dynamic programming algorithm that returns the fewest number of coins that you need to make up that amount. If that amount of money cannot be made up by any combination of the coins, return -1.

You may assume that you have an infinite number of each kind of coin.

4. (20 points) A number of languages (including Chinese and Japanese) are written without spaces between words. You are given text in such a language, and you are required to design an algorithm to infer likely boundaries between consecutive words in the text. If English were written without spaces, the analogous problem would consist of taking a string like "meetateight" and deciding that the best segmentation is "meet at eight" (and not "me et at eight", or "meet ate aight", or any of a huge number of possibilities).

To solve the problem, you are given a function Quality that takes any string of letters and returns a number that indicates the quality of the word formed by the string. A high number indicates that the string resembles a word in the language (e.g. 'meet'), whereas a low number means that the string does not resemble a word (e.g. 'eeta')

The total quality of a segmentation is determined by adding up the qualities of each of its words.

Write pseudocode of a dynamic programming algorithm that take a string y and computes a segmentation of maximum total quality. What is the running time of your algorithm? (You can treat the call to Quality as a single computational step, O(1)).

5. (15 points) A thief robbing a store wants to take the most valuable load that can be carried in a knapsack capable of carrying at most W pounds of loot. The thief can choose to take any subset of n items in the store. The i-th item is worth v_i dollars and weighs w_i pounds, where v_i and w_i are integers. Which items should the thief take and what is their total value? Write pseudocode to solve the problem.

This problem is called the $0-1$ *knapsack problem*.

6. In the *fractional knapsack problem*, the setup is the same as the previous question, but the thief can take fractions of items, rather than having to make a binary (0-1) choice for each item. You can think of an item in the 0-1 knapsack problem as being like a gold ingot and an item in the fractional knapsack problem as more like gold dust.

a. (5 points) Describe (in words) a greedy algorithm for solving the fractional knapsack problem. (Hint: consider the value per pound of each item).

To prove that the greedy algorithm is correct, you have to show that the problem exhibits the greedy-choice and optimal-substructure properties.

b. (10 points) Prove that the fractional knapsack problem has the greedy-choice property (i.e. prove that the greedy choice you made in part a is always part of an optimal solution).

c. (10 points) Prove that the fractional knapsack problem has optimal substructure.