Assignment-4 Part A

(Total Points: 40)
(Extra Credit: 20)

Instructions

Submission

You can either write your answer using Word/LaTex or take a picture/scan of your hand-written (neat and tidy) solution, then put your solution in one PDF file. Submit your PDF online using Canvas. All submission must be posted before the deadline.

Problem 1 (Max Points: 10)

Consider the problem of making change for n cents using the fewest number of coins. Assume that each coin's value is an integer.

a) Describe a greedy algorithm to make change consisting of quarters, dimes, nickels, and pennies.

b) Give a set of coin denominations for which the greedy algorithm does not yield an optimal solution. Your set should include a penny so that there is a solution for every value of n.

Problem 2 (Max Points: 10)

Given a rod of length n inches and an array of prices ($p_{i,}$ for i = 1…n) of all pieces of size smaller than or equal to n, that is p_i is the price for a piece of length i inches. The Rod Cutting problem is to determine the maximum value obtainable by cutting up the rod and selling the pieces.

a) Describe a greedy algorithm to determine a set of piece sizes to cut up a rod to maximize the value when selling the pieces. For example, given p = [1, 3, 4, 5], the optimal solution would be [2, 2]. That is, a 4-inch rod is divided into two 2-inch pieces worth 3 each, for a total of 6.

a) Give a set of prices for which the greedy algorithm does not yield an optimal solution. Show the solution your algorithm yields along with an optimal solution.

Problem 3 (Max Points: 10)

The Fibonacci numbers are defined by recurrence $F_0 = 0$, $F_1 = 1$ and $F_i = F_{i-1} + F_{i-2}$. Give an O(n) time dynamic-programming algorithm to compute the nth Fibonacci number i.e. F_n. Draw the sub-problem graph. How many vertices and edges are in the graph?

Problem 4 (Max Points: 10)

Give a dynamic-programming solution to the 0-1 knapsack problem that runs in O(nW) time where n is the number of items and W is the maximum weight of items that can be put in the knapsack.

Provide a sample run of your algorithm for 4 items (w_i, v_i) as (1,3), (2,4), (3,5), (4,8) where w_i is the weight of i-th item and v_i is the value of i-th item and also the maximum weight that can be put in the knapsack(W) is 6. Give the maximum value of items that can be put in the knapsack.

Problem 5 (Max Points: 10) [Extra Credit]

You are climbing a staircase. It takes *n* (> 0) steps to reach to the top.
Each time you can either climb 1, 2 or 3 steps. In how many distinct ways can you climb to the top?

Example:
Number of stairs: 3
Number of ways: 4
Distinct ways to climb to the top: {(1,1,1), (1, 2), (2, 1), (3)}

Note: your algorithm only needs to return the number of distinct ways and not the actual ways to climb to the top.

Problem 6 (Max Points: 10) [Extra Credit]

Describe an efficient algorithm that, given a set {x_1, x_2, …., x_n} of points on the real line, determines the size of the smallest set of unit-length closed intervals that contains all of the given points. Also give the time complexity of your algorithm.

For example: Given points {0.8, 4.3, 1.7, 5.4}, the size of the smallest set of unit-length closed intervals to cover them is 3.

Run your algorithm on the following set of points:
{0.8, 2.3, 3.1, 1.7, 3.6, 4.0, 4.2, 5.5, 5.2, 1.0, 3.9, 4.7} and determine the size of the smallest set of unit-length closed intervals that contains all of the given points.