CS 325 - Homework Assignment 3

Problem 1: *(2 points)* **Rod Cutting**: (from the text CLRS) 15.1-2

Problem 2: *(3 points)* **Modified Rod Cutting**: (from the text CLRS) 15.1-3

Problem 3: *(6 points)* **Product Sum**

Given a list of n integers, $v_1, \ldots, v_n$, the product-sum is the largest sum that can be formed by multiplying adjacent elements in the list. Each element can be matched with at most one of its neighbors.

For example, given the list 4, 3, 2, 8 the product sum is 28 = (4 × 3) + (2 × 8), and given the list 2, 2, 1, 3, 2, 1, 2, 2, 1, 2 the product sum is 19 = (2 × 2) + 1 + (3 × 2) + 1 + (2 × 2) + 1 + 2.

a) Compute the product-sum of 2, 1, 3, 5, 1, 4, 2.

b) Give the dynamic programming optimization formula OPT[j] for computing the product-sum of the first j elements.

c) What would be the asymptotic running time of a dynamic programming algorithm implemented using the formula in part b).

Problem 4: *(5 points)* **Making Change**: Given coins of denominations (value) $1 = v_1 < v_2 < \ldots < v_n$, we wish to make change for an amount A using as few coins as possible. Assume that v_i's and A are integers. Since $v_1 = 1$ there will always be a solution.

Formally, an algorithm for this problem should take as input:

- An array V where V[i] is the value of the coin of the i^{th} denomination.
- A value A which is the amount of change we are asked to make

The algorithm should return an array C where C[i] is the number of coins of value V[i] to return as change and m the minimum number of coins it took. You must return exact change so

$$\sum_{i=1}^{n} V[i] \cdot C[i] = A$$

The objective is to minimize the number of coins returned or:

$$\mathrm{m} = \min \sum_{i=1}^{n} C[i]$$

a) Describe and give pseudocode for a dynamic programming algorithm to find the minimum number of coins to make change for A.

b) What is the theoretical running time of your algorithm?

Problem 5: *(10 points)* **Making Change Implementation**

Submit a copy of all your files including the txt files and a README file that explains how to compile and run your code in a ZIP file to TEACH. We will only test execution with an input file named amount.txt.

You may use any language you choose to implement your DP change algorithm. The program should read input from a file named "amount.txt". The file contains lists of denominations (V) followed on the next line by the amount A.

Example amount.txt:

```
1 2 5
10
1 3 7 12
29
1 2 4 8
15
```

In the above example the first line contains the denominations V=(1, 2, 5) and the next line contains the amount A = 10 for which we need change. There are three different denomination sets and amounts in the above example. A denomination set will be on a single line and will always start with the 1 "coin".

The results should be written to a file named change.txt and should contain the denomination set, the amount A, the change result array and the minimum number of coins used.

Example change.txt:

```
1 2 5
10
0 0 2
2
1 3 7 12
29
0 1 2 1
4
1 2 4 8
15
1 1 1 1
4
```

In the above example, to make 29 cents change from the denomination set (1, 3, 7, 12) you need 0: 1 cent coin, 1: 3 cent coin, 2: 7 cent coins and 1: 12 cent coin for a total of 4 coins.

Problem 6: *(4 points)* **Making Change Experimental Running time**

a) Collect experimental running time data for your algorithm in Problem 4. Explain in detail how you collected the running times.

b) On three separate graphs plot the running time as a function of A, running time as a function of n and running time as a function of nA. Fit trend lines to the data. How do these results compare to your theoretical running time? *(Note: n is the number of denominations in the denomination set and A is the amount to make change)*