Name: Georges Hatem

CS 590 Homework 8 Creativity Exercises

Due Date: March 27, 2022

<u>Problem 10.5.15:</u>

Case 1:

Given the values of denominations, D1 = 25, D2 = 10, D3 = 5, and D4 = 1, a greedy algorithm will pick the denominations one by one in non-increasing order and divide the given amount until all the denomination values are used.

The greedy algorithm is as follows:

Algorithm GreedyCoinsChange (int value):

- Let D [] be the array containing the denomination values.
- Sort the array D in non-increasing order.
- Let n be the variable that denotes the size of the array D.
- Set the value of the variable count to 0.
- Start the loop from i=0 to n-1.
    - If value <= 0,
        - Exit from the loop.
    - Otherwise,

- Set count = count + int (value / D [i])
- Set value = value % D[i].

- Return count.

The above algorithm runs in **O (n)** time, where n is the number of denominations.
The trace of the above algorithm for the given values is as follows:

D [] = {25, 10, 5, 1}, n = 4, count = 0

Value = 40

Iteration 1: i=0, value=40, count = 0
Value > 0, jump to else case.

Count = count + value/ D [0] = 0 + 40/25 = 1

Value = value % 25 = 40 % 25 = 15

Iteration 2: i=1, value = 15, count = 1
Value > 0, jump to else case.

Count = count + value/ D [1] = 1 + 15/10 = 2

Value = value % 10 = 15 % 10 = 5

Iteration 3: i=2, value = 5, count = 2
Value > 0, jump to else case.

Count = count + value/ D [2] = 2 + 5/5 = 3

Value = value % 5 = 5 % 5 = 0

Iteration 3: i=3, value = 0, count = 3
Value = 0, exit the loop.

Return count = 3

Therefore, the minimum coins required is equal to 3.

Case 2:
For an added value of denomination, D5 = 20, the greedy algorithm does not provide an optimal solution for all the amounts.

For example, consider the case when value = 40.

The greedy algorithm results in 3 coins (25+10+5), but the optimal is 2 coins (20+20).

Therefore, a dynamic programming solution is required to look for all the cases and then find the optimal answer. The dynamic programming algorithm is as follows:

Algorithm DPCoinsChange (int value):

- Let D [] be the array containing the denomination values.
- Let n be the variable that denotes the size of the array D.
- Create an array table [] of size value + 1 to store the calculated results.
- Start a loop from i=0 to value.
    - Set the value of table [i] = infinity
    - Increase the value of the variable i by 1.
- Set the value of table[0] = 0;
- Start the loop from i=1 to value.
    - Start the loop from j=0 to n-1.
        - If D [j] <= i,

            - Create the variable res = table [i – D [j]]
            - Set the value of table [i] = res+1, if the value of res + 1 < table [i].
        - Increase the value of the variable j by 1.
    - Increase the value of the variable i by 1.
- Return table [value].

Since the above algorithm stores all the different cases to sum up to the given amount and then computes the minimum number of coins, it works for all the amounts.