EE3980 Algorithms

Homework 9. Coin Set Design

106061151 劉安得

1. Introduction

Currently in Taiwan, we have $1, $5, $10 and $50 NTD coins. Thus, any dollar amounts less than 100 must consist of a number of coins. This homework we need using greedy algorithm to calculate the average number of coins one needs to carry for Coins[] = {1, 5, 10, 50}, which represents the Taiwan's system, if the probabilities of carrying $1 to $99 coins are equal. And then using the same greedy algorithm to find the minimum average of coins, where Coins[] = {1, 5, 10, dd}, {1, 5, dd, 50}, {1, 5, dd, dd}, dd is the values of coins replacing $10 or $50.

- Greedy Algorithm
  - Feasible Solutions

    Arrange the inputs to satisfy some constraints
  - Optimal Solution

    Find feasible solution that minimize or maximize an objective function

The greedy method is an algorithm that takes one input at a time. If a particular input results in infeasible solution, then it is rejected; otherwise it is included. The input is selected according to some measure. The selection measure can be the objective functions or other functions that approximate the optimality. However, this method usually generates a suboptimal solution.

2. Approach

A. Keys of Implementation

- Find the minimum average

  To find the minimum average, I use for loop to try every possible value, and record the minimum average and the values of coins replacing.

  For example, {1, 5, 10, dd}, I use for loop to iterate dd from 11 to 99.

B. Greedy

```
// Given an int D, Ncoin, Coins[], use greedy algorithm to find the solution
// Input: D, NCoin, Coins[]
// Output: number of coins
Algorithm NCoinGreedy(D, NCoin, Coins[])
{
   num := 0;
   for i := NCoin to 1 step -1 do {    // select max value of coins
      num := D/Coins[i];
      D := D%Coins[i];
   }
   return num;
}
```

I. Correctness

- For Taiwan’s system, Coins[] = {1, 5, 10, 50}, the algorithm find the optimal solution.

  Because the optimal solution only can be at most four $1, one $5, four $10, and this algorithm will select the largest value of coins, and the solution will match the condition coincidentally. Therefore, for Taiwan’s system, this algorithm produces optimal solution.

- However, for arbitrary Coins[], the algorithm may not find the optimal solution.

  For example, when Coins[] = {1, 5, 10, 11}, D = 20

  The optimal solution is 2 (2*10), however, the algorithm output is 6 (1*11+1*5+4*1).

II. Time Complexity

Suppose NCoin = 4

Statement	s/e	Freq.	Total steps
`Algorithm NCoinGreedy(D, NCoin, Coins[])`	0	-	0
`{`	0	-	0
`    num := 0;`	1	1	1
`    for i := NCoin to 1 step -1 do {`	1	NCoin+1	NCoin+1
`        num := D/Coins[i];`	1	NCoin	NCoin
`        D := D%Coins[i];`	1	NCoin	NCoin
`    }`	0	-	0
`    return num;`	1	1	1
`}`	0	-	0
Total			3*NCoin+3

T = 3*NCoin+3 = Θ(NCoin)

III. Space Complexity

one for each variable: *D*, *NCoin*, *i*, and *num*; four for *Coin[]*

S = 8 = Θ(1)

C. Main Function

```
// Driver function to solve Coin Set Design.
// Input: None
// Output: average number of coins, the values of coins replacing
Algorithm main(void)
{
    Coins[] := {1, 5, 10, 50};
    num = 0;
    for i := 1 to 99 do {       // try $1 to $99
        num := num + NCoinGreedy(i, 4, Coins);
    }
    aver := num / 99;           // calculate the average
    Coins[] := {1, 5, 10, dd};
    Find the minimum average;
    Coins[] := {1, 5, dd, 50};
    Find the minimum average;
    Coins[] := {1, 5, dd, dd};
    Find the minimum average;
    write(average number of coins, the values of coins replacing);
}
```

3. Result & Observations

A. Output Result

```
For coin set {1, 5, 10, 50} the average is 5.05051

Coin set {1, 5, 10, 28} has the minimum average of 4.54545

Coin set {1, 5, 14, 50} has the minimum average of 4.52525

Coin set {1, 5, 13, 42} has the minimum average of 4.24242
```

B. Conclusions/ Observations

The greedy algorithm is not appropriate for Coin Set Design, we can use Dynamic Programing to solve this problem, use an array to store the number of coins for each D, and for a new input D, we only need to check the value of index D-Coin[].