PROJECT TITLE

AMINI PROJECTREPORT

Submittedby

PULKIT SHARMA [Reg No: RA2011003010136]

for the course 18CSC204J Design and Analysis of Algorithms

Under the guidance of

Dr. Sindhuja M

(Assistant Professor, Department of Computing Technologies)

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M.Nagar, Kattankulathur, ChengalpattuDistrict

JUNE2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

S.R.M. NAGAR, KATTANKULATHUR -603 203

KANCHEEPURAM DISTRICT

BONAFIDE CERTIFICATE

Register No_

Certified to be the bonafide record of work done by_____of _____

_____, B.Tech Degree course in the Practical ***18CSC205J –***

18CSC204J Design and Analysis of Algorithms *during the academic year 2022-2023*

Lab Incharge

Date: Year Co-ordinator

*Submitted for University Examination held in_*

_____, ***SRM INSTITUTE OF SCIENCE AND***

TECHNOLOGY, Kattankulathur.

Date: Examiner-1 Examiner-2

PROJECT REPORT
ON

Coin change problem : Greedy algorithm

Submitted in partial fulfillment of the requirement for the IV semester

of

BACHELOR OF TECHNOLOGY IN
DESIGN AND ANALYSIS OF ALGORITHM

Submitted By:

PULKIT SHARMA (RA2011003010136)

Under the supervision of
Dr. M Sindhuja
(Asst. Professor)

DEPARTMENT OF COMPUTING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

SESSION - 2022

CERTIFICATE

This is to certify that the project entitled "**Coin change problem By Greedy algorithm** " carried out by "**PULKIT SHARMA**" under my supervision at Department of Computing and Technology, SRM Institute of Technology, Kattankulathur, Chennai.

The work is original, as it has not been submitted earlier either in part or full for any purpose before.

Dr. M Sindhuja

(Asst. Professor)

DECLARATION

I, hereby declare that the work presented in this dissertation entitled "Coin change problem : Greedy algorithm " has been done by me, and this dissertation embodies my own work.

PULKIT SHARMA

Approved By: Dr. M Sindhuja

ACKNOWLEDGEMENTS

I thanks Dr. M SINDHUJA (Asst. Professor) who have been the great inspiration and who have provided sufficient background knowledge and understanding of this subject.

Our humble prostration goes to her, for providing all the necessary resources and environment ,which have aided me to complete this project successfully.

PREFACE

This report is an introduction to Coin change problem : Greedy algorithm in C programming.Anybody, who doesn't know even the basics of Coin change problem in C ,will be certainly able to understand and gain the great knowledge from this report. The core theme of the report focuses on the development of Coin change problem : Greedy algorithm in C language.

The report also contains the strategy of making Coin change problem which serve a good idea to make a Coin change problem program in C language to the programmer.

The most of the idea of making this problem and report is taken from " Let Us C- by Yashwant Kanetkar'' , Schaum's Outline-C(TMH publications), and Internet (Wikipedia ,Google ,etc.) .

.

CONTENTS

ABSTRACT

The Coin Change problem is to represent a given amount V with fewest number of coins m. As a variation of knapsack problem, it is known to be NP-hard problem. Most of the time, Greedy alogrithm (time complexity O(m), space complexity O(1)), irrespective of real money system, doesn't give optimal solution. Dynamic algorithm (time complexity O(mV), space complexity O(V)) gives optimal solution but is still expensive as amount V can be very large. In this paper, we have presented a suboptimal solution for the coin change problem which is much better than the greedy algorithm and has accuracy comparable to

dynamic solution. Moreover, comparison of different algorithms has been stated in this paper. Proposed algorithm has a time complexity of O(m2f) and space complexity of O(1), where f is the maximum number of times a coin can be used to make amount V. It is, most of the time, more efficient as compared to dynamic algorithm and uses no memoization, this is a significant advantage over dynamic approach.

Introduction

WHAT IS COIN CHANGE PROBLEM BY GREEDY ALGORITHM?

Greedy algorithm greedily selects the best choice at each step and hopes that these choices will lead us to the optimal solution of the problem. Of course, the greedy algorithm doesn't always give us the optimal solution, but in many problems it does. For example, in the coin change problem of the Coin Change chapter, we saw that selecting the coin with the maximum value was not leading us to the optimal solution. But think of the case when the denomination of the coins are 1¢, 5¢, 10¢ and 20¢. In this case, if we select the coin with maximum value at each step, it will lead to the optimal solution of the problem.

Also as stated earlier, the fraction knapsack can also be solved using greedy strategy i.e., by taking the items with the highest valueweightvalueweight ratio first.
Thus, checking if the greedy algorithm will lead us to the optimal solution or not is our next task and it depends on the following two properties:

- **Optimal substructure** → If the optimal solutions of the sub-problems lead to the optimal solution of the problem, then the problem is said to exhibit the optimal substructure property.
- **Greedy choice property** → The optimal solution at each step is leading to the optimal solution globally, this property is called greedy choice property.

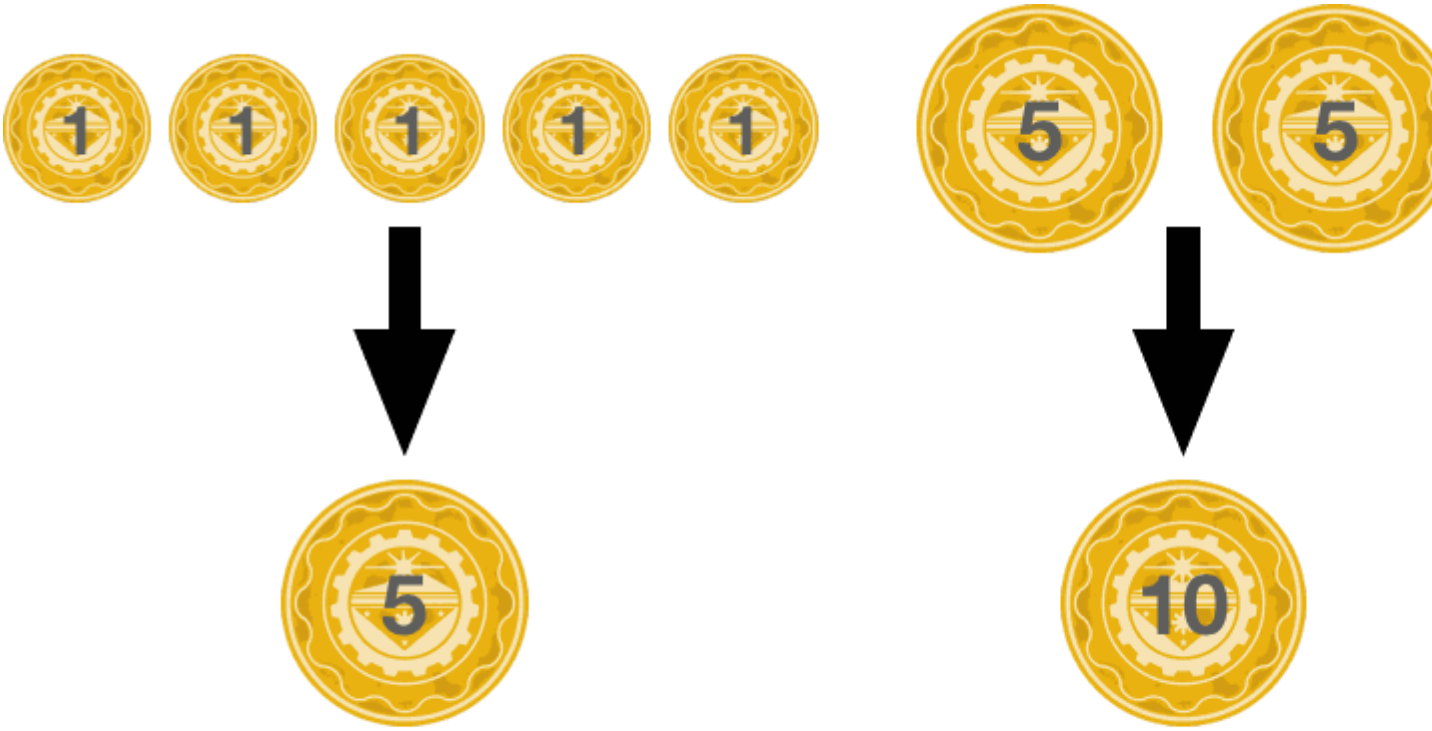
Implementation of the greedy algorithm is an easy task because we just have to choose the best option at each step and so is its analysis in comparison to other algorithms like divide and conquer but checking if making the greedy choice at each step will lead to the optimal solution or not might be tricky in some cases. For example, let's take the case of the coin change problem with the denomination of 1¢, 5¢, 10¢ and 20¢. As stated earlier, this is the special case where we can use the greedy algorithm instead of the dynamic programming to get the optimal solution, but how do we check this or know if it is true or not?

Let's suppose that we have to make the change of a number n using these coins. We can write n as $5x+y$, where x and y are whole numbers. It means that we can write any value as multiple of 5 + some remainder. For, example 4 can be written as $5*0+4$, 7 can be written as $5*1+2$, etc.

$5*4 + 2 = 22$

Now, the value of y will range from 0 to 4 (if it becomes greater than or equal to 5, then it will be covered in the $5x$ part) and we can check that any value between 0 to 4 can be made only by using all coins of value 1. So, we know that the optimal solution for the part y will contain coins of value 1 only. Therefore, we will consider for the optimal solution of the $5x$ part. As the problem exhibits optimal substructure, so the optimal solution to both the subproblems will lead to the optimal solution to the problem.

Since $5x$ is a multiple of 5, so it can be made using the values 5, 10 and 20 (as all three are multiples of 5). Also in 5, 10 and 20, the higher value is multiple of the lower ones. For example, 20 is multiple of 5 and 10 both and 10 is multiple of 5. So, we can replace the multiple occurrences of the smaller coins with the coins having higher value and hence, can reduce the total number of coins. For example, if 5 is occurring more than once, it can be replaced by 10 and if 10 is occurring more than once it can be replaced by 20. In other words, we can choose the coins with higher value first to reduce the total number of coins.



USEDCODINGIN/ ALGORITHMSTRATEGY

Let's start by having the values of the coins in an array in reverse sorted order i.e., `coins = [20, 10, 5, 1]`.

Now if we have to make a value of n using these coins, then we will check for the first element in the array (greedy choice) and if it is greater than n, we will move to the next element, otherwise take it. Now after taking one coin with value `coins[i]`, the total value which we have to make will become n-coins[i].

Make 50

Now to make 50-20 = 30

Now to make 30-20 = 10

TARGET: 19

TARGET: 19 – 10 =9


```
i = 0
while (n)
  if coins[i] > n
    i++
```

`if coins [i] > n` → We are starting from the 0th element (element with the largest value) and checking if we can use this coin or not. If the value of this coin is greater than the value to be made, then we are moving to the next coin - `i++`.
If the value of the coin is not greater than the value to be made, then we can take this coin. So, we will take it. Let's just print the value right here to indicate we have taken it, otherwise, we can also append these value in an array and return it.

```
while (n)
  ...
  else
    print coins[i]
```

Now, the value to be made is reduced by coins[i] i.e., `n-coins[i]`.

```
COIN-CHANGE-GREEDY(n)
  coins = [20, 10, 5, 1]
  i = 0

  while (n)
    if coins[i] > n
      i++
    else
      print coins[i]
      n = n-coins[i]
```

Time Complexity

What will the time complexity of the implementation? First of all, we are sorting the array of coins of size n, hence complexity with O(nlogn). While loop, the worst case is O(amount). If all we have is the coin with 1-denomination. Overall complexity for coin change problem becomes O(n log n) + O(amount).
Will this algorithm work for all sort of denominations? The answer is no. It will not give any solution if there is no coin with denomination 1. So be careful while applying this algorithm.

PROGRAM FOR COIN CHANGE PROBLEM:

```
#include <stdio.h>

int coins[] = { 1,5,10,25,100 };

int findMaxCoin(int amount, int size){
for(int i=0; i<size; i++){
if(amount < coins[i]) return i-1;
}
return -1;
}
int findMinimumCoinsForAmount(int amount, int change[]){

int numOfCoins = sizeof(coins)/sizeof(coins[0]);
int count = 0;
while(amount){
int k = findMaxCoin(amount, numOfCoins);
if(k == -1)
printf("No viable solution");
else{
amount-= coins[k];
change[count++] = coins[k];
}
}
return count;
}

int main(void) {
int change[10]; // This needs to be dynamic
int amount = 34;
int count = findMinimumCoinsForAmount(amount, change);

printf("\n Number of coins for change of %d : %d", amount, count);
printf("\n Coins : ");
for(int i=0; i<count; i++){
printf("%d ", change[i]);
}
return 0;
}
```

OUTPUT-

```
Number of coins for change of 34 : 6
Coins : 25 5 1 1 1 1
```

CONCLUSION

I implemented coin change problem successfully with the help of c language and it is very entertaining. I also take some help from google and the assigned book yashwant kanetkar and some wikipedia on google. Also we've studied a greedy algorithm to find the least number of coins for making the change of a given amount of money and analyzed its time complexity. Further, we've also illustrated the working of the discussed algorithm with a real-life example and discussed its limitation.

REFERENCE

1. GOOGLE
2. WIKIPEDIA
3. LET US "C" by Yashwant Kanetkar