18CSC204J-DESIGN ALGORITHMS AND ANALYSIS

RECORD

SOLVING COIN CHANGE PROBLEM USING GREEDY ALGORITHM

TEAM MEMBERS	REG. NUMBER
1. NELAMALLI MEGHANA KIRAN	RA2011003011237
2. SUJITH KIRAN NELAMALLI	RA2011003011239
3. SAKETH P	RA2011003011233

LAB INCHARGE:

Abstract:

Automatic money exchange systems are very rarely found but are needed at certain times. The purpose of this project is to make it easier for people who want to exchange money from big to small nominal. The obstacle faced in this exchange system is the use of algorithms needed to run the system. Some methods that can be used to design and implement this system are using Greedy Algorithm and Brute force algorithms. By using this greedy algorithm, a problem with the shortest route search technique can be completed quickly, but the results generated by the greedy algorithm are not always optimal, while brute force can solve the problem of money exchange optimally but need a longer time. In this subject we are trying to solve this problem by making an application using python language to implement the greedy in solving this problem. After making the code algorithm and implementing the greedy, we conclude that our program can solve this coins exchange problem just in a second but sometimes the result is not optimal (with the smallest fraction), but if using brute force algorithm we can get the optimal total of money changes but the time will extremely increase up to 20 times higher depending on the amount of the total money that wants to be changed. In this experiment the user will input the money that they want to change, in the program we already have a predetermined amount and money that has been prepared to exchange money.

WHAT IS GREEDY ALGORITHM?

A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment. It doesn't worry whether the current best result will bring the overall optimal result.
The algorithm never reverses the earlier decision even if the voice is wrong. It works in a top-down approach.
This algorithm may not produce the best result for all the problems. It's because it always goes for the local best choice and produces the global best result.

Advantages of Greedy Approach:

- Greedy algorithms are very easy to describe.
- Greedy algorithms can perform better than other algorithms.

Drawbacks of Greedy Approach:

- The greedy algorithm doesn't always produce the optimal solution.

PROBLEM STATEMENT:

In the coin change problem, We are given an array of coins having different denominations and an integer sum representing the total money. You have to return the fewest coins that you will need to make up that sum if it’s not possible to construct that sum then return -1.

METHODOLOGY:

1. Sort the array of coins in decreasing order.
2. Initialize the result as empty.

3. Find the largest denomination that is smaller than the current amount.
4. Add found denomination to the result. Subtract value of found denomination from amount.
5. If the amount becomes 0, then print the result.
6. Else repeat steps 3 and 4 for the new value of V.

PROBLEM DESCRIPTION:

Let's understand what the problem is with a help of an example

According to the coin change problem, we are given a set of coins of various denominations. Consider the below array as the set of coins where each element is basically a denomination.

{1,2,5,10,20,50,100,500}

Our task is to use these coins to form a sum of money using the minimum (or optimal) number of coins. Also, we can assume that a particular denomination has an infinite number of coins. In other words, we can use a particular denomination as many times as we want.

As an example, if we have to achieve a sum of 93, we need a minimum of 5 coins as below:

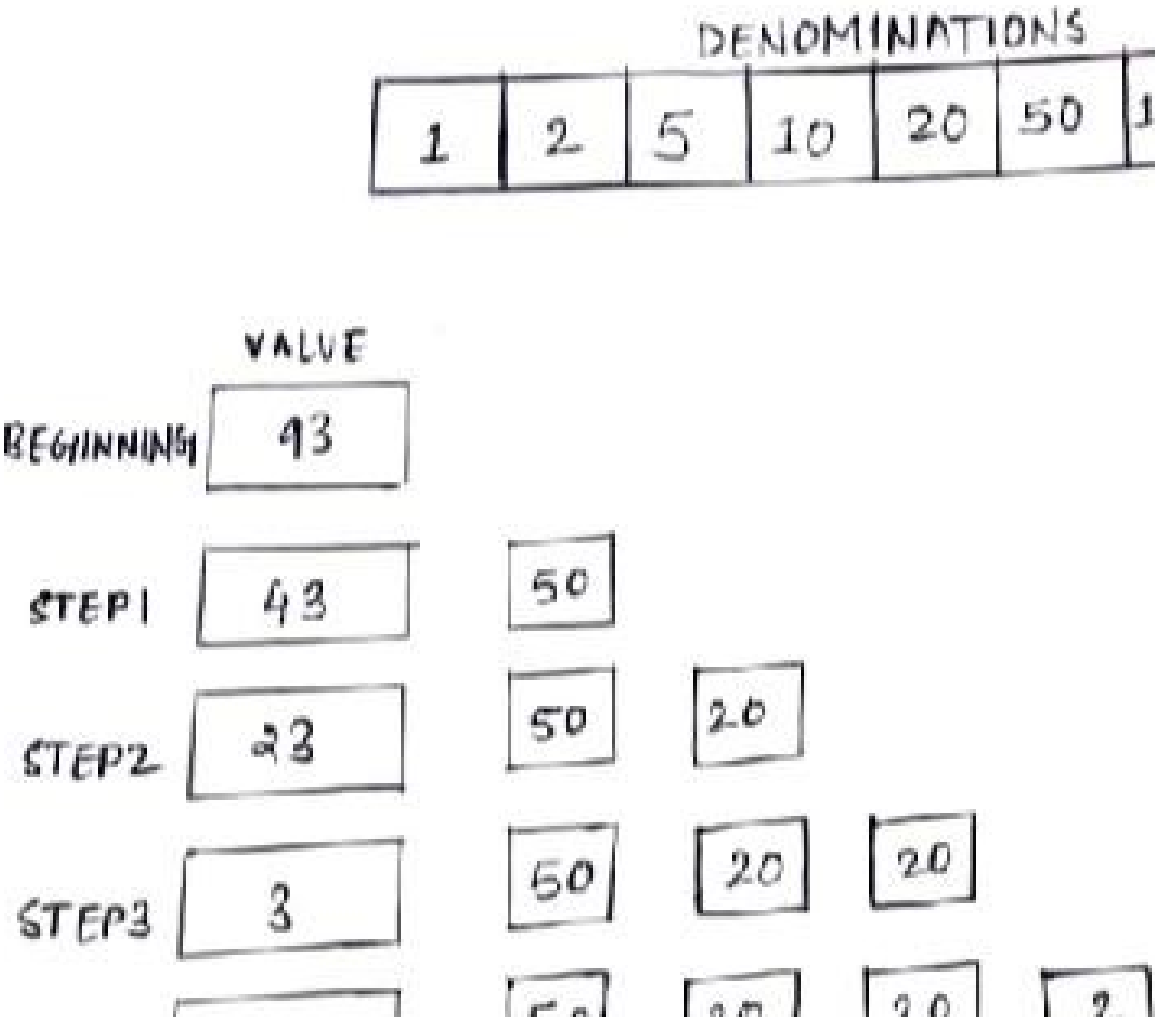
50202021

Going by the greedy approach, we first pull out a coin of denomination 50. We have 43 left to achieve. The next highest coin is 20 and therefore we pull that out. Then, we have 23 left. After that, we pull out another coin of

20 resulting in 3 left. We can then pull out 2 & 1 to finally reach 0 meaning we have reached a sum of 93.

As you can see, at each step, the algorithm makes the best possible choice. In this case, the highest denomination possible.

Below is an illustration that depicts the overall process:



As you can see, at each step we use the highest possible coin from the denominations. At last, we are able to reach the value of 93 just by using 5 coins.

Complete Code for The coin change problem:

```
# Recursive Python3 program for
# coin change problem.

def findMin(V):

	# All denominations of Indian Currency
	deno = [1, 2, 5, 10, 20, 50,
			100, 500,
	1000] n = len(deno)

	# Initialize Result
	ans = []

	# Traverse through all
	denomination i = n - 1
	while(i >= 0):

		# Find denominations
		while (V >= deno[i]):
			V -= deno[i]
			ans.append(deno[i])

		i -= 1

	# Print result
	for i in range(len(ans)):
		print(ans[i], end = " ")

# Driver Code
if __name__ == '__main__':
	n = 93
	print("Following is minimal number",
		"of change for", n, ": ", end = "")
	findMin(n)
```

Sample output: Following is minimal number of change for 93 : 50 20 20 2 1 >

Analyze the time complexity:

Greedy Algorithm
Number of Coins in the array: 9
The number of coins to be exchanged: n
Time complexity: 1

THANK YOU!