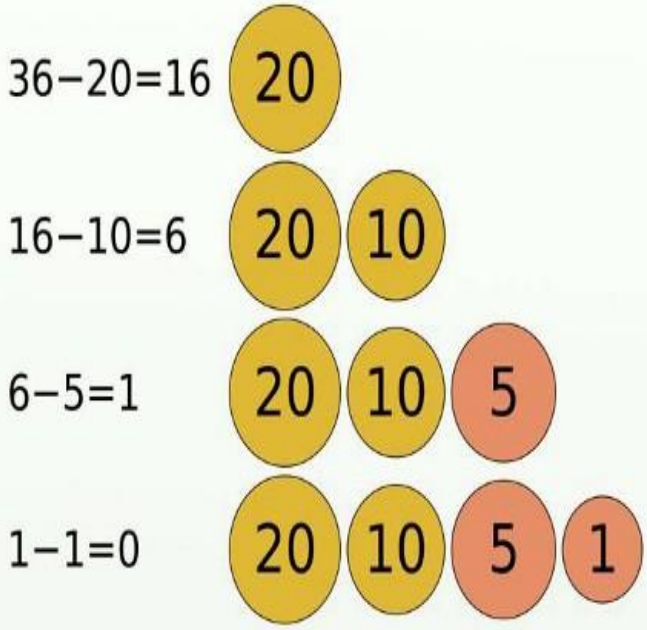
Greedy algorithm



https://en.wikipedia.org/wiki/File:Greedy_algorithm_36_cents.svg

CTA PRACTICAL 4

20MCED08

Practical 4

Solve Make-Change problem using Greedy approach.

Introduction

- Making Change problem is nothing but finding the minimum number of coins (of certain denominations) that add up to a given amount of money(Total money).
- Greedy method is used for obtaining optimum solution.(But not necessary that GREEDY will always give Optimum solution.

Greedy method:

For some type of coin system a greedy approach works. The values of coins will be like { 1,5,10,25,50}.

In our algorithm we always choose the biggest denomination, subtract the all possible values and going to the next denomination.

so,we will use greedy algorithm to give the least amount of coins.
suppose a customer wants change of 41 Rs

41–25=16 -> [25] so,take coin of 25.
16–10=6 ->[25][10]

6–5=1 ->[25][10][5]

1–1=0 ->[25][10][5][1]

so here,is the optimal solution of 41 Rs change is 25,10,5,1=41 Rs.

But always That is not the case.

The values of coins will be like { 4,10,25}.

41–25=16 ->[25]

16–10=6 ->[25][10]

6–4=2 ->[25][10][4]

2- ??

so here is the problem,or error message.

So here is the optimal solution is [25][4][4][4][4] =41 Rs.

1) Making Change Problem Using Greedy Method

```
def minCoin():
    rs = int(input("Enter the value of Rs(V)"))
    gl_rs = rs

    print("\nSelect the denominations from below")

    denomCh = int(input("Press 1 for Denom = [ 1,2, 5, 10, 20, 50, 100, 5
00, 1000](Sorted)\nPress 2 for Denom = [2,7,10]\nPress 3 for Denom = [100
0, 500, 100, 50,20,10,2,1](ReverseSorted)\n"))
    if denomCh == 1:
        denom  = [1, 2, 5, 10, 20, 50, 100, 500, 1000]
    elif denomCh == 2:
        denom  = [2,7,10]
    else:
        denom  = [1000, 500, 100, 50,20,10,2,1]
        denom.sort()
        # by default

    lenDenom = len(denom)
    res_list = []
    i = lenDenom - 1 # starting from last (maximum) element

    while(i >= 0):
        while (rs >= denom[i]):
            rs = rs - denom[i]
            res_list.append(denom[i])
            if(rs==0):
                flag = 1
            else:
                flag = 0

        i -= 1

    if(flag):
        resLen = len(res_list) # Length of res list
        print("\nResult....")
        print("\nTotal number of coin/notes required is =",resLen)
        print("Minimal number of change for rs : ",gl_rs)
        for i in range(resLen): # Printing the result
            print(res_list[i], end = " ")
    else:
        print("No solution found")


minCoin() # calling function
```

Output for rs = 93:

```
Enter the value of Rs(V)93

Select the denominations from below
Press 1 for Denom = [ 1,2, 5, 10, 20, 50, 100, 500, 1000](Sorted)
Press 2 for Denom = [2,7,10]
Press 3 for Denom = [1000, 500, 100, 50,20,10,2,1](ReverseSorted)
1

Result....

Total number of coin/notes required is = 5
Minimal number of change for rs :  93
50 20 20 2 1
```

Output for rs = 15:

```
Enter the value of Rs(V)15

Select the denominations from below
Press 1 for Denom = [ 1,2, 5, 10, 20, 50, 100, 500, 1000](Sorted)
Press 2 for Denom = [2,7,10]
Press 3 for Denom = [1000, 500, 100, 50,20,10,2,1](ReverseSorted)
2
No solution found
```

Output for rs = 56:

```
Enter the value of Rs(V)56

Select the denominations from below
Press 1 for Denom = [ 1,2, 5, 10, 20, 50, 100, 500, 1000](Sorted)
Press 2 for Denom = [2,7,10]
Press 3 for Denom = [1000, 500, 100, 50,20,10,2,1](ReverseSorted)
3

Result....

Total number of coin/notes required is = 4
Minimal number of change for rs :  56
50 2 2 2
```

Observation:

If a coin set of { 25-Rupees, 10-Rupees, 4-Rupees} coins.

The greedy algorithm would not be able to make change for 41 Rupeess, since after committing to use one 25-Rupees coin and one 10-Rupees coin it would be impossible to use 4-Rupees coins for the balance of 6 Rupeess, whereas a person or a more sophisticated algorithm could make change for 41 Rupeess with one 25-Rupees coin and four 4-Rupees coins.

Say we have {1,5,10,20,25} Rupeess, what if we wanted minimum coins for 40 Rupeess, the optimal choice will be two 20 Rupees coins, but the algorithm will choose coins 25,10,and 5, three coins.

So we can use dyanmic Algorithm to find the optimal solution for this kind of problem.