← Notes

Change Making Problem

11 Dynamic Programming

The change making problem is an optimization problem that asks "**What is the minimum number of coins I need to make up a specific total?**"

The input to the Change Making Problem is a sequence of positive integers `[d1, d2, d3 ... dn]` and `T`, where `di` represents a coin denomination and `T` is the target amount. Assuming an unlimited supply of coins of each denomination, we need to find the number of coins `N` required to form the given amount. An extra effort would be to find the exact coins to build up the amount.

> The above problem represents an optimal sub-structure, which means that the problem can be broken down into smaller parts. Suppose there is an optimal solution for amount `T` and if we break the target amount into two parts `m` and `T-m`, then there will be optimal solution for making amount `m` using some portion from the optimal solution for amount `T` and the remaining coins from the solution will be the optimal solution for making amount `T-m`.

Let C[m] be the minimum number of coins of denominations d1,d2,...,dk needed to make change for m amount. In the optimal solution to making change for `m` amount, there must exist some first coin `di`, where `di < m`. Furthermore, the remaining coins in the solution must themselves be the optimal solution to making change for `m- di`.

Thus, if `di` is the first coin in the optimal solution to making change for m amount, then `C[m]=1 + C[m - di]` i.e. one di coin plus `C[m - di]` coins to optimally make change for `m - di` amount. We don't know which coin `di` is the first coin; however, we may check all n such possibilities (subject to the constraint that `di < m`), and the value of the optimal solution must correspond to the minimum value of `1 + C[m - di]`, by definition.

Furthermore, when making change for 0, the value of the optimal solution is clearly 0 coins. We thus have the following recurrence.

```
C[p]    =     0                                if p = 0
               min(i: di < p) {1 + C[p - di]}   if p > 0
```

Below is the code given for the above algorithm

```
#include <iostream>
#define N 4
#define C 17
using namespace std;

// In this example we take the amount as 17, and a total of
// 4 denominations of coins

int main()
{
    // contains the coin denominations
    int coins[N]={1,2,5,10};

    // C[i] contains the minimum number of coins required
    // to form the sum i
    int amount[C+1]={0};

    for(int amt = 1; amt <= C ;amt++)
    {
        amount[amt] = INT_MAX;
        int temp = INT_MAX;
        for(int c = 0; c < N;c++)
        {
            // if the value of the coin is less than the amount
                    if(coins[c] <= amt)
            {
                // What is the other number of coins that will b
                // if coins[c] is used in the solution for amoun
                int temp_amt = amount[amt-coins[c]] + 1;

                // choose the minimum number of coins that will
                // for the amount i

                if(temp_amt < temp)
                {
                    temp = temp_amt;
                    amount[amt] = temp;
                }
            }
```

```
        }
    }
    cout << amount[C] << endl;
    return 0;
}
```


AUTHOR

Sachin Gupta
Co-founder & CEO at Hack...
Bangalore, Karnataka, India
4 notes

Write Note
My Notes
Drafts

TRENDING NOTES

Strings And String Functions
written by Vinay Singh

Segment Tree and Lazy Propagation
written by Akash Sharma

Number Theory - II
written by Tanmay Chaudhari

Matrix exponentiation
written by Mike Koltsov

Graph Theory - Part II
written by Pawel Kacprzak

more ...

Blog
Engineering Blog
Updates & Releases
Team
Careers
In the Press

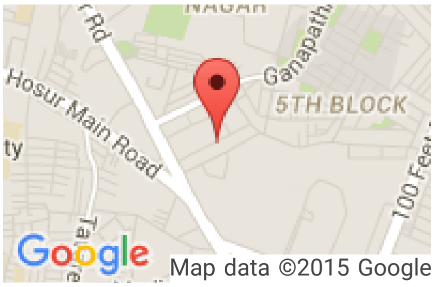
API
Chrome Extension
CodeTable
HackerEarth Academy
Developer Profile
Resume
Campus Ambassadors
Get Me Hired
Privacy
Terms of Service

AMA
Code Monk
Judge Environment
Solution Guide
Problem Setter Guide
Practice Problems
HackerEarth Challenges
College Challenges

RECRUIT

Developer Sourcing
Lateral Hiring
Campus Hiring
FAQs
Customers
Annual Report

REACH US



IIIrd Floor, Salarpuria Business Center,
4th B Cross Road, 5th A Block,
Koramangala Industrial Layout,
Bangalore, Karnataka 560095, India.

contact@hackerearth.com
+91-80-4155-4695
+1-650-461-4192