Find minimum number of coins that make a given value

Given a value V, if we want to make change for V cents, and we have infinite supply of each of C = { C1, C2, .. , Cm} valued coins, what is the minimum number of coins to make the change?

Examples:

```
Input: coins[] = {25, 10, 5}, V = 30
Output: Minimum 2 coins required
We can use one coin of 25 cents and one of 5 cents

Input: coins[] = {9, 6, 5, 1}, V = 11
Output: Minimum 2 coins required
We can use one coin of 6 cents and 1 coin of 5 cents
```

Recommended: Please solve it on "*PRACTICE* " first, before moving on to the solution.

This problem is a variation of the problem discussed Coin Change Problem. Here instead of finding total number of possible solutions, we need to find the solution with minimum number of coins.

The minimum number of coins for a value V can be computed using below recursive formula.

```
If V == 0, then 0 coins required.
If V > 0
   minCoin(coins[0..m-1], V) = min {1 + minCoins(V-coin[i])}
                               where i varies from 0 to m-1
                               and coin[i] <= V
```

Below is recursive solution based on above recursive formula.

C++

```
// A Naive recursive C++ program to find minimum of coins
// to make a given change V
#include<bits/stdc++.h>
using namespace std;

// m is size of coins array (number of different coins)
int minCoins(int coins[], int m, int V)
{
   // base case
   if (V == 0) return 0;

   // Initialize result
   int res = INT_MAX;

   // Try every coin that has smaller value than V
   for (int i=0; i<m; i++)
   {
     if (coins[i] <= V)
     {
         int sub_res = minCoins(coins, m, V-coins[i]);

         // Check for INT_MAX to avoid overflow and see if
         // result can minimized
         if (sub_res != INT_MAX && sub_res + 1 < res)
            res = sub_res + 1;
     }
   }
   return res;
}

// Driver program to test above function
int main()
{
    int coins[] =  {9, 6, 5, 1};
    int m = sizeof(coins)/sizeof(coins[0]);
    int V = 11;
    cout << "Minimum coins required is "
         << minCoins(coins, m, V);
    return 0;
}
```

Run on IDE

Java

```
// A Naive recursive JAVA program to find minimum of coins
// to make a given change V
class coin
{
    // m is size of coins array (number of different coins)
    static int minCoins(int coins[], int m, int V)
    {
       // base case
       if (V == 0) return 0;

       // Initialize result
       int res = Integer.MAX_VALUE;

       // Try every coin that has smaller value than V
       for (int i=0; i<m; i++)
       {
         if (coins[i] <= V)
         {
             int sub_res = minCoins(coins, m, V-coins[i]);

             // Check for INT_MAX to avoid overflow and see if
             // result can minimized
```

```
                if (sub_res != Integer.MAX_VALUE && sub_res + 1 < res)
                   res = sub_res + 1;
            }
         }
         return res;
    }
    public static void main(String args[])
    {
       int coins[] =  {9, 6, 5, 1};
       int m = coins.length;
       int V = 11;
       System.out.println("Minimum coins required is "+ minCoins(coins, m, V) );
    }
}/* This code is contributed by Rajat Mishra */
```

Run on IDE

Output:

```
Minimum coins required is 2
```

The time complexity of above solution is exponential. If we draw the complete recursion tree, we can observer that many subproblems are solved again and again. For example, when we start from V = 11, we can reach 6 by subtracting one 5 times and by subtracting 5 one times. So the subproblem for 6 is called twice.

Since same suproblems are called again, this problem has Overlapping Subprolems property. So the min coins problem has both properties (see this and this) of a dynamic programming problem. Like other typical Dynamic Programming(DP) problems, recomputations of same subproblems can be avoided by constructing a temporary array table[][] in bottom up manner. Below is Dynamic Programming based solution.

C++

```
// A Dynamic Programming based C++ program to find minimum of coins
// to make a given change V
#include<bits/stdc++.h>
using namespace std;

// m is size of coins array (number of different coins)
int minCoins(int coins[], int m, int V)
{
    // table[i] will be storing the minimum number of coins
    // required for i value.  So table[V] will have result
    int table[V+1];

    // Base case (If given value V is 0)
    table[0] = 0;

    // Initialize all table values as Infinite
    for (int i=1; i<=V; i++)
        table[i] = INT_MAX;

    // Compute minimum coins required for all
    // values from 1 to V
    for (int i=1; i<=V; i++)
    {
```

```
        // Go through all coins smaller than i
        for (int j=0; j<m; j++)
          if (coins[j] <= i)
          {
              int sub_res = table[i-coins[j]];
              if (sub_res != INT_MAX && sub_res + 1 < table[i])
                  table[i] = sub_res + 1;
          }
    }
    return table[V];
}

// Driver program to test above function
int main()
{
    int coins[] =  {9, 6, 5, 1};
    int m = sizeof(coins)/sizeof(coins[0]);
    int V = 11;
    cout << "Minimum coins required is "
         << minCoins(coins, m, V);
    return 0;
}
```

Run on IDE

Output:

```
Minimum coins required is 2
```

Time complexity of the above solution is O(mV).

Find minimum number of coins (using Dynamic Programming) | GeeksforGeeks

Thanks to Goku for suggesting above solution in a comment here and thanks to Vignesh Mohan for suggesting this problem and initial solution.

Asked in: Accolite,Amazon,Morgan-Stanley,Oracle,Paytm,Snapdeal,Synopsys

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

GATE CS Corner Company Wise Coding Practice

Dynamic Programming

Recommended Posts:

Dynamic Programming | Set 7 (Coin Change)
Minimum number of squares whose sum equals to given number n
Collect maximum points in a grid using two traversals
Dynamic Programming | Set 10 (0-1 Knapsack Problem)
Dynamic Programming | Set 8 (Matrix Chain Multiplication)
Count ways to build street under given constraints
Maximum length subsequence with difference between adjacent elements as either 0 or 1
Unique paths in a Grid with Obstacles
Largest rectangular sub-matrix whose sum is 0
Coin game winner where every player has three choices

(Login to Rate and Mark)

3.2 Average Difficulty : **3.2/5.0**
Based on **105** vote(s)

Add to TODO List

Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments Share this post!