Minimal Number of Coins for Change

Gets minimal number of coins with value v1,v2, … ,vn to make change for amount of money with value t.

**Analysis:**

In order to solve this problem, we need to define a function f(t) to represent minimal number of coins for target t. As we have n coins, we can make n choices for value t: we can pick coin with value v1 to my coins set so that remain value is t – v1, for this case the minimal number of coins is f(t – v1) + 1. Same case if we pick v2 from coins set we can get minimal number is f(t-v2) + 1.

In this way, we divide this problem into a n sub-problems: f(t-v1), f(t-v2),….,f(t-vn), so we can conduct a formal equation based on this method:

For this kind of equation, we can implement it by recursion quickly. But I find some overlap case so I decide to use iteration.

**Example:**

Let me show an example first.

Suppose we have coins of value 1, 5, 10, 15, 25. Our target is 16. I will explain my approach by a table.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 1 | 0 | 1 | 2 | 3 | 4 | 5 | 2 | 3 | 4 | 5 | 6 | 2 | 3 | 4 | 5 | 6 | 2 |
| 5 | 0 |  |  |  |  | 1 | 2 | 3 | 4 | 5 | 2 | 2 | 3 | 4 | 5 | 2 | 2 |
| 10 | 0 |  |  |  |  |  |  |  |  |  | 1 | 2 | 3 | 4 | 5 | 2 | 2 |
| 15 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 | 2 |
| 25 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

So based on this table, we found the number of minimal coins for 16 is 2. Actually each highlighted number represents the minimal number of coins for that value.

Although we show the calculation process in a 2-D table, we actually can implement it in 1-D array. Because we only need to record the minimal number for each value which is the highlighted numbers.

**Solution:**

Let me show my code in c++:

int GetMinCount(int total,vector<int> coins)

{

int\* counts = new int[total + 1];

counts[0] = 0;

int minCoins[total+1] = {-1};

const int MAX = 0x7FFFFFFF;

for(int i = 1; i <= total; ++ i)

{

int count = MAX;

for(int j = 0; j < coins.size(); ++ j)

{

if(i - coins[j] >= 0 && count > counts[i - coins[j]])

{

count = counts[i - coins[j]];

minCoins[i] = coins[j];

}

}

if(count < MAX)

counts[i] = count + 1;

else

counts[i] = MAX;

}

int minCount = counts[total];

delete[] counts;

int S = total;

while (S > 0)

{

cout<<minCoins[S];

S = S - minCoins[S];

}

cout<<endl;

return minCount;

}

The time complexity is as we have two loops and space complexity is O(n as we have a new array to store the minimal value.