Dynamic Programming | Set 7 (Coin Change)

Given a value N, if we want to make change for N cents, and we have infinite supply of each of S = { S1, S2, .. , Sm} valued coins, how many ways can we make the change? The order of coins doesn't matter.

For example, for N = 4 and S = {1,2,3}, there are four solutions: {1,1,1,1},{1,1,2},{2,2},{1,3}. So output should be 4. For N = 10 and S = {2, 5, 3, 6}, there are five solutions: {2,2,2,2,2}, {2,2,3,3}, {2,2,6}, {2,3,5} and {5,5}. So the output should be 5.

1) Optimal Substructure
To count total number solutions, we can divide all set solutions in two sets.
1) Solutions that do not contain mth coin (or Sm).
2) Solutions that contain at least one Sm.
Let count(S[], m, n) be the function to count the number of solutions, then it can be written as sum of

count(S[], m-1, n) and count(S[], m, n-Sm).

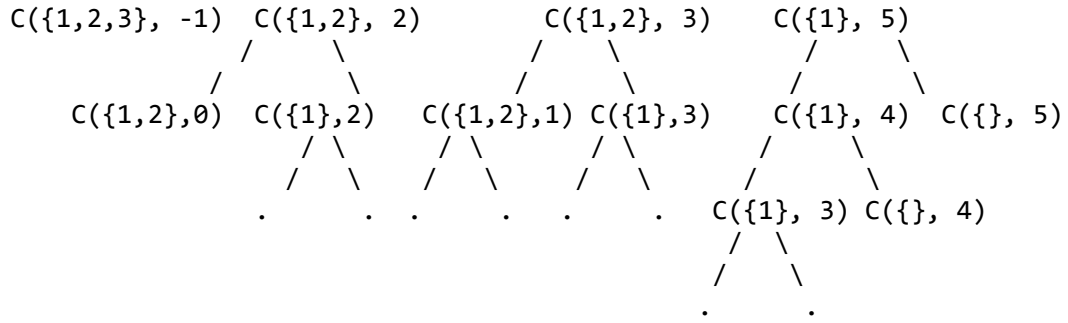
Therefore, the problem has optimal substructure property as the problem can be solved using solutions to subproblems.

2) Overlapping Subproblems
Following is a simple recursive implementation of the Coin Change problem. The implementation simply follows the recursive structure mentioned above.

```
#include<stdio.h>

// Returns the count of ways we can sum  S[0...m-1] coins to get sum n
int count( int S[], int m, int n )
{
    // If n is 0 then there is 1 solution (do not include any coin)
    if (n == 0)
        return 1;

    // If n is less than 0 then no solution exists
    if (n < 0)
        return 0;

    // If there are no coins and n is greater than 0, then no solution exist
    if (m <=0 && n >= 1)
        return 0;

    // count is sum of solutions (i) including S[m-1] (ii) excluding S[m-1]
    return count( S, m - 1, n ) + count( S, m, n-S[m-1] );
}

// Driver program to test above function
int main()
{
    int i, j;
    int arr[] = {1, 2, 3};
    int m = sizeof(arr)/sizeof(arr[0]);
    printf("%d ", count(arr, m, 4));
    getchar();
    return 0;
}
```

It should be noted that the above function computes the same subproblems again and again. See the following recursion tree for S = {1, 2, 3} and n = 5.
The function C({1}, 3) is called two times. If we draw the complete tree, then we can see that there are many subproblems being called more than once.

```
C() --> count()
                                C({1,2,3}, 5)
                             /                \
                           /                    \
                C({1,2,3}, 2)                  C({1,2}, 5)
              /      \                         /          \
            /          \                     /              \
```



Since same suproblems are called again, this problem has Overlapping Subprolems property. So the Coin Change 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.

Dynamic Programming Solution

```
#include<stdio.h>

int count( int S[], int m, int n )
{
    int i, j, x, y;

    // We need n+1 rows as the table is consturcted in bottom up manner using
    // the base case 0 value case (n = 0)
    int table[n+1][m];

    // Fill the enteries for 0 value case (n = 0)
    for (i=0; i<m; i++)
        table[0][i] = 1;

    // Fill rest of the table enteries in bottom up manner
    for (i = 1; i < n+1; i++)
    {
        for (j = 0; j < m; j++)
        {
            // Count of solutions including S[j]
            x = (i-S[j] >= 0)? table[i - S[j]][j]: 0;

            // Count of solutions excluding S[j]
            y = (j >= 1)? table[i][j-1]: 0;

            // total count
            table[i][j] = x + y;
        }
    }
    return table[n][m-1];
}

// Driver program to test above function
int main()
{
    int arr[] = {1, 2, 3};
```

```
    int m = sizeof(arr)/sizeof(arr[0]);
    int n = 4;
    printf(" %d ", count(arr, m, n));
    return 0;
}
```

Time Complexity: O(mn)

Following is a simplified version of method 2. The auxiliary space required here is O(n) only.

```
int count( int S[], int m, int n )
{
    // table[i] will be storing the number of solutions for
    // value i. We need n+1 rows as the table is consturcted
    // in bottom up manner using the base case (n = 0)
    int table[n+1];

    // Initialize all table values as 0
    memset(table, 0, sizeof(table));

    // Base case (If given value is 0)
    table[0] = 1;

    // Pick all coins one by one and update the table[] values
    // after the index greater than or equal to the value of the
    // picked coin
    for(int i=0; i<m; i++)
        for(int j=S[i]; j<=n; j++)
            table[j] += table[j-S[i]];

    return table[n];
}
```

Thanks to Rohan Laishram for suggesting this space optimized version.

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

References:
http://www.algorithmist.com/index.php/Coin_Change

Related Topics:

- Linearity of Expectation
- Iterative Tower of Hanoi
- Count possible ways to construct buildings
- Build Lowest Number by Removing n digits from a given number
- Set Cover Problem | Set 1 (Greedy Approximate Algorithm)
- Find number of days between two given dates

- How to print maximum number of A's using given four keys
- Write an iterative O(Log y) function for pow(x, y)

Tags: Dynamic Programming

Writing code in comment? Please use **ideone.com** and share the link here.

Neha • 12 hours ago

How to print the combinations in iterative DP Solution Method 1 ?

• Reply • Share ›

Saurabh • a month ago

http://ideone.com/cA9ShC

This is a simple code without sub structures.

• Reply • Share ›

Techie Me • 2 months ago

Wonderful explanation.. Really appreciate your effort... Here is a Java version of the application and a much deeper dive into the approach.

http://techieme.in/minimum-num...

• Reply • Share ›

Mission Peace • 2 months ago

https://www.youtube.com/watch?...

1 • Reply • Share ›

Mission Peace • 2 months ago

https://www.youtube.com/watch?...

1 • Reply • Share ›

Abhi • 2 months ago

GeeksforGeeks we can use this simple method

int count(int arr[],int s,int m) //s is the number of coin in array;
//m is the number which we want to get //from the given coins

```
{

int dp[m+1];

dp[0]=0;

int i;

int j;

for(i=1;i<=m;i++)

{

dp[i]=0;

for(j=0;j<s;j++) {="" if(i="">=arr[j] && dp[i]<dp[i-arr[j]]+1) {="" dp[i]="dp[i-arr[j]]+1;" }="" }="" }="" return="" dp[m];="" }="">
```

• Reply • Share ›

RJ • 3 months ago

I am not able to understand simplified version of method 2, can someone help with more detail.

• Reply • Share ›

Nik • 3 months ago

in first solution if we do recursion with memoization wont that work?

• Reply • Share ›

Anurag Singh ➚ Nik • 3 months ago

Will work. That will be lazy memoization.
If at all we go for memorization/DP, iterative approach (eager) will be considered better than recursive approach.

• Reply • Share ›

Vib • 3 months ago

@Geeks... Shouldn't i & ii be swapped in the comment ? in recursive version
// count is sum of solutions (i) including S[m-1] (ii) excluding S[m-1]
return count(S, m - 1, n) + count(S, m, n-S[m-1]);

• Reply • Share ›

guest • 3 months ago

2nd solution ...awesome (Y)

• Reply • Share ›

Tejwinder • 3 months ago

for first solution:

for first solution:

if (m <=0 && n >= 1)
return 0;
this may fail when m =0 and n=s[m]

it should be if (m<0) return 0;

please correct me if i am wrong.

• Reply • Share ›

Serin ➚ Tejwinder • 3 months ago

m = 0 means s contains no element => n = s[m] will cause IndexOutOfBoundException. Your case is invalid.

• Reply • Share ›

Tejwinder ➚ Serin • 3 months ago

ok

• Reply • Share ›

Ankit • 4 months ago

And can someone please explain the space optimized version as well. I am not able to see how it is working.

• Reply • Share ›

Abhijeet Sachdev ➚ Ankit • 24 days ago

Hey, see this:
http://www.geeksforgeeks.org/c...

• Reply • Share ›

Ankit • 4 months ago

And can someone please explain the space optimized version as well. I am not able to see how it is working.

• Reply • Share ›

Ankit • 4 months ago

What is table[0][i] representing in first solution ?

• Reply • Share ›

Serin ➚ Ankit • 3 months ago

That represents the number of ways we can get a sum of 0, not using the coins after i-th coin in S. Since the sum is 0, there is only 1 way: take no coin at all, thus table[0][i] is 1 regardless of i

1 • Reply • Share ›

Guest • 4 months ago

// whats wrong in this code I am getting duplicates ...

```
def int count(s, n, aux) {
if (n == 0) {
return 1;
}

if (n < 0) {
return -1;
}

if (aux[n] != null)
return aux[n]

int ways = 0;

for (def i=0; i<s.size(); i++)="" {="" int="" x="count(s," n="" -="" s[i],="" aux);="" if="" (x="" !="-1)" {="" ways="" +="x;" aux[n-s[i]]="x;" }="" return="" ways;="" }="">
```

• Reply • Share ›

Guest • 4 months ago

```
def int count(s, n, aux) {
if (n == 0) {
return 1;
}

if (n < 0) {
return -1;
}

if (aux[n] != null)
return aux[n]

int ways = 0;

for (def i=0; i<s.size(); i++)="" {="" int="" x="count(s," n="" -="" s[i],="" aux);="" if="" (x="" !="-1)" {="" ways="" +="x;" aux[n-s[i]]="x;" }="" return="" ways;="" }="">
```

• Reply • Share ›

ds_beginnner • 4 months ago

I think I am getting duplicates in the below approach or is there anything wrong in here?

```
def int count(s, n, aux) {
if (n == 0) {
```

```
return 1;
}

if (n < 0) {
return -1;
}

// if (aux[n] != null)
// return aux[n]

int ways = 0;

for (def i=0; i<s.size(); i++)="" {="" int="" x="count(s," n="" -="" s[i],="" aux);="" if="" (x="" !="-1)" {="" ways="" +="x;" aux[n-s[i]]="x;" }="" }="" return="" ways;="" }="">
```

• Reply • Share ›

Guest • 4 months ago

for n = 4 and set as {1,2,3} by the above algorithm how can I get the {2,2} condition

• Reply • Share ›

kake • 5 months ago

when i put 4 for the n but it showings double 2 but showing 4 and when i try using arr[] = {1,4,5} and the n= 13 (itu supposed to be 4,4,4 and 1 but it showing 8 in your program, so what its means?

• Reply • Share ›

sandeep • 5 months ago

return count(S, m - 1, n) + count(S, m, n-S[m-1]); . I think this is wrong it should be like return count(S, m - 1, n) + count(S, m-1, n-S[m-1]);. Also the comment is totally wrong.

• Reply • Share ›

jaig ➚ sandeep • 4 months ago

the previous is correct.
count(S,m,n-S[m-1]) is correct coz we can include S[m-1] in order to solve for n-S[m-1]

1 • Reply • Share ›

neelabhsingh • 5 months ago

@GeeksForGeeks
// count is sum of solutions (i) including S[m-1] (ii) excluding S[m-1]
return count(S, m - 1, n) + count(S, m, n-S[m-1]);

//I think comment is in reverse order, please correct it.
(i) excluding S[m-1] (i!) including S[m-1] in sum

• Reply • Share ›

sameer hussain • 5 months ago

what we have to do if n is long??

• Reply • Share ›

Kenneth • 6 months ago

My DP solution with only O(N) space complexity:

http://ideone.com/x27uqM

1 • Reply • Share ›

Sid • 6 months ago

Space optimized version (O(n)) doesnot give correct answer . Ex - denominations 1,5 finding ways to get 6 -> Space optimized way will consider both (1,5), (5,1) as different cases .. leading to value 3 while the answer should be 2.

• Reply • Share ›

sandeep ➦ Sid • 5 months ago

bro i think the code is correct and it deosnot give 3 ,it gives out 2 .

• Reply • Share ›

bhopu • 6 months ago

#include<stdio.h>

sum(int *arr,int *temp,int st,int ind,int m,int n,int s){
int i;
if(s==n){
int j;
printf("\n");
for(j=0;j<ind;j++) printf("%d="" ",temp[j]);="" return="" 0;="" }="" if(s="">n)
return 0;

for(i=st;i<m;i++){ temp[ind]="arr[i];" sum(arr,temp,i,ind+1,m,n,s+arr[i]);="" }="" }="" main(){="" int="" arr[]="{2," 5,="" 3,="" 6};="" int="" temp[10];="" sum(arr,temp,0,0,4,10,0);="" }="">

• Reply • Share ›

Ravi • 7 months ago

2nd solutions is vey nice.

• Reply • Share ›

Kaushal Yagnik • 7 months ago

@GeeksforGeeks

How can we find the number of times each coin has been used to make that sum?

2 • Reply • Share ›

Priyal Rathi • 7 months ago

Another recursive solution:
for i=start to end
include arr[i] in the sum and num of ways in which we can get N-arr[i] from the array starting from ith index (this is done to avoid same combination of coins in different order)

Link: http://ideone.com/HEPe9X

1 • Reply • Share ›

Ritesh Malav • 8 months ago

how do we solve the problem if n is very large (n < 10^18)

1 • Reply • Share ›

sk ↗ Ritesh Malav • a month ago

Did you ever check how much it takes when you execute a for loop for 10^18 . If not then check do some operation inside for loop.

• Reply • Share ›

Daniel Lim Wee Soong • 8 months ago

Can anyone explain why does this work? Thanks

• Reply • Share ›

Robin ↗ Daniel Lim Wee Soong • 7 months ago

Daniel, this works by splitting the problem into two sub problems that can further be broken down until we reach the base case (asking how many ways to create a sum of 0, to which we always respond 1). The final solution can then be built up from the base cases in a way that avoids repetition.

To break the problem down, we notice that asking "How many ways can I make change for 11 cents using coins of sizes 1 and 5". Is the same as asking: "How many ways can I make change for 11 cents using no 5 cent coins" and "How many ways can I make change for 11 cents using at least one 5 cent coin", then summing the results.

Answering our first sub-question is relatively easy. There is only one way to make any amount of change with only 1's, and that's by using all 1's.

Answering the second sub-problem, where we use at least one 5 is slightly more subtle. Since we only know we'll have at least one 5 in the solution, we only need to check the number of ways to create the remaining 6 cents using 5's and 1's.

These sub-problems are state as count(S, m - 1, n), and count(S, m, n-S[m-1]), respectively, in the author's solution.

The DP solution for this calculates the results to those sub-problems from the bottom up, which avoids solving the subproblems repeatedly with the same arguments.

15 ▲ | ▼ • Reply • Share ›

Nilesh ➦ Robin • 5 months ago

this is awesome explanation. Thanks

▲ | ▼ • Reply • Share ›

sher • 8 months ago

very nice!

▲ | ▼ • Reply • Share ›

Guest • 9 months ago

Wat is the difference between sum of subsets and this problem?

▲ | ▼ • Reply • Share ›

Mayank ➦ Guest • 7 months ago

SOS doesn't include repeated elements.. if it is..then both are same

▲ | ▼ • Reply • Share ›

Paparao Veeragandham • 9 months ago

```
int CoinChange(int data[], int n , int sum)
{
int i , j ,count[sum+1] = {INT_MAX};
count[0] = 1;
for(i =1; i <=sum; i++)
for( j = 0; j<n;j++) {="" if(data[j]="=" i)="" count[i]="1;" elseif(="" i="" -="" data[j]="">=0)
count[i] = Min(count[i], count[i-data[j]] );
}
return count[sum];
}
```

▲ | ▼ • Reply • Share ›

Paparao Veeragandham ➦ Paparao Veeragandham • 4 months ago

```
if( data[j] == i)
count[i] = 1;
else if ( i - data[j] >= 0)
count[i] = min(count[i] , count[ i -data[j]] + 1);

return count[sum];
```

▲ | ▼ • Reply • Share ›

Vãîbhåv Joshî • 10 months ago

simple and elegant solution
let the coins are c1,c2,c3...cn and the amount is N
the recurrence relation is written as

C[i] = min(C[N-c1], C[N-c2]....C[N-cn]) + 1

https://ideone.com/yPXyXK

1 • Reply • Share ›

Monkey D. Luffy ➚ Vãîbhåv Joshî • 4 months ago

the question is how many ways you can make the change.It is not that what is the minimum no of coins to make the change.Please read the question before replying. btw thanks for the new question.

1 • Reply • Share ›

Rasool • 10 months ago

Nice. You made the 2-D array, without explaining why we need to make changes at every element's location. Thanks :/

5 • Reply • Share ›

RACHIT SAXENA • 10 months ago

second solution reduced copying of row's prev cell elements (y)

• Reply • Share ›

RACHIT SAXENA • 10 months ago

its all about hash map (y)

• Reply • Share ›

- Interview Experiences
- Advanced Data Structures
- Dynamic Programming
- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Mathematical Algorithms
- Recursion
- Geometric Algorithms

Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

Follow @GeeksforGeeks

Recent Comments

- lt_k

  i need help for coding this function in java...

  Java Programming Language · 1 hour ago

- Piyush

What is the purpose of else if (recStack[*i])...

Detect Cycle in a Directed Graph · 1 hour ago

- Andy Toh

  My compile-time solution, which agrees with the...

  Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 1 hour ago

- lucy

  because we first fill zero in first col and...

  Dynamic Programming | Set 29 (Longest Common Substring) · 2 hours ago

- lucy

  @GeeksforGeeks i don't n know what is this long...

  Dynamic Programming | Set 28 (Minimum insertions to form a palindrome) · 3 hours ago

- manish

  Because TAN is not a subsequence of RANT. ANT...

  Given two strings, find if first string is a subsequence of second · 3 hours ago