GeeksforGeeks

A computer science portal for geeks

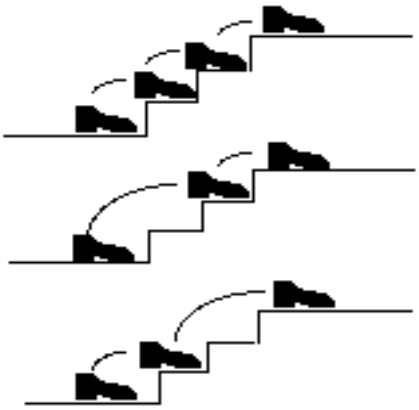
GeeksQuiz

- Home
- Algorithms
- DS
- GATE
- Interview Corner
- Q&A
- C
- C++
- Java
- Books
- Contribute
- Ask a Q
- About

Array
Bit Magic
C/C++
Articles
GFacts
Linked List
MCQ
Misc
Output
String
Tree
Graph

Count ways to reach the n’th stair

There are n stairs, a person standing at the bottom wants to reach the top. The person can climb either 1 stair or 2 stairs at a time. Count the number of ways, the person can reach the top.



Consider the example shown in diagram. The value of n is 3. There are 3 ways to reach the top. The diagram is taken from Easier Fibonacci puzzles

More Examples:

```
Input: n = 1
Output: 1
There is only one way to climb 1 stair

Input: n = 2
Output: 2
There are two ways: (1, 1) and (2)

Input: n = 4
Output: 5
(1, 1, 1, 1), (1, 1, 2), (2, 1, 1), (1, 2, 1), (2, 2)
```

We can easily find recursive nature in above problem. The person can reach n'th stair from either (n-1)'th stair or from (n-2)'th stair. Let the total number of ways to reach n't stair be 'ways(n)'. The value of 'ways(n)' can be written as following.

```
ways(n) = ways(n-1) + ways(n-2)
```

The above expression is actually the expression for Fibonacci numbers, but there is one thing to notice, the value of ways(n) is equal to fibonacci(n+1).

ways(1) = fib(2) = 1
ways(2) = fib(3) = 2
ways(3) = fib(4) = 3

So we can use function for fibonacci numbers to find the value of ways(n). Following is C++ implementation of the above idea.

```
// A C program to count number of ways to reach n't stair when
// a person can climb 1, 2, ..m stairs at a time.
#include<stdio.h>

// A simple recursive program to find n'th fibonacci number
int fib(int n)
{
   if (n <= 1)
      return n;
   return fib(n-1) + fib(n-2);
}

// Returns number of ways to reach s'th stair
int countWays(int s)
{
    return fib(s + 1);
}

// Driver program to test above functions
int main ()
{
  int s = 4;
  printf("Number of ways = %d", countWays(s));
  getchar();
  return 0;
}
```

Output:

```
Number of ways = 5
```

The time complexity of the above implementation is exponential (golden ratio raised to power n). It can be optimized to work in O(Logn) time using the previously discussed Fibonacci function optimizations.

Generalization of the above problem
How to count number of ways if the person can climb up to m stairs for a given value m? For example if m is 4, the person can climb 1 stair or 2 stairs or 3 stairs or 4 stairs at a time.

We can write the recurrence as following.

```
   ways(n, m) = ways(n-1, m) + ways(n-2, m) + ... ways(n-m, m)
```

Following is C++ implementation of above recurrence.

```
// A C program to count number of ways to reach n't stair when
// a person can climb either 1 or 2 stairs at a time
#include<stdio.h>

// A recursive function used by countWays
int countWaysUtil(int n, int m)
{
    if (n <= 1)
        return n;
    int res = 0;
```

```
    for (int i = 1; i<=m && i<=n; i++)
        res += countWaysUtil(n-i, m);
    return res;
}

// Returns number of ways to reach s'th stair
int countWays(int s, int m)
{
    return countWaysUtil(s+1, m);
}

// Driver program to test above functions
int main ()
{
    int s = 4, m = 2;
    printf("Nuber of ways = %d", countWays(s, m));
    return 0;
}
```

Output:

```
Number of ways = 5
```

The time complexity of above solution is exponential. It can be optimized to O(mn) by using dynamic programming. Following is dynamic programming based solution. We build a table res[] in bottom up manner.

```
// A C program to count number of ways to reach n't stair when
// a person can climb 1, 2, ..m stairs at a time
#include<stdio.h>

// A recursive function used by countWays
int countWaysUtil(int n, int m)
{
    int res[n];
    res[0] = 1; res[1] = 1;
    for (int i=2; i<n; i++)
    {
       res[i] = 0;
       for (int j=1; j<=m && j<=i; j++)
         res[i] += res[i-j];
    }
    return res[n-1];
}

// Returns number of ways to reach s'th stair
int countWays(int s, int m)
{
    return countWaysUtil(s+1, m);
}

// Driver program to test above functions
int main ()
```

```
{
    int s = 4, m = 2;
    printf("Nuber of ways = %d", countWays(s, m));
    return 0;
}
```

Output:

```
Number of ways = 5
```

This article is contributed by **Abhishek**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

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)

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

Bhagwat Singh • 9 months ago

There is a problem in second loop in DP solution :
As i think there should be j<=i in condition checking.

1 ▲ | ▼ • Reply • Share ›

Abhay Pandey ➦ Bhagwat Singh • 9 months ago

Thanx sir..

▲ | ▼ • Reply • Share ›

GeeksforGeeks Mod ➦ Bhagwat Singh • 9 months ago

Thanks for pointing this out. We have updated the condition.

▲ | ▼ • Reply • Share ›

wrestler ➦ GeeksforGeeks • a month ago

Hi, I see many problems which are not categorized like this article. So a user only come across this, if he/she has googled for similar problem. This is not good for the user, who is following each problem category wise for preparation.

▲ | ▼ • Reply • Share ›

Girish ➦ GeeksforGeeks • 6 months ago

I think res[1] = 2; instead of current res[1] = 1; since for a two floor scenario, there are two ways of climbing. res[0] = 1 is correct. In the DP solution, with n = 4, m = 2, i think we will get 1, 1, 2, 3 with current program whereas it should be 1, 2, 3, 5 which will give correct answer

▲ | ▼ • Reply • Share ›

Ekta Goel ➦ Girish • 6 months ago

res[1] specifies the number of ways to reach n'th stair where we have n as 1 i.e. we have to climb till 1'th stair when we can climb m steps at once(j). Now j is<=m and also it has to be less than i, thus the possible values are 0 and 1 and the only possible solution is (1,1). So, res[1]=1;

▲ | ▼ • Reply • Share ›

Vishnu • 9 months ago

Usefull for placement..

3 ▲ | ▼ • Reply • Share ›

sup • 9 months ago

quite easy !!!

▲ | ▼ • 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