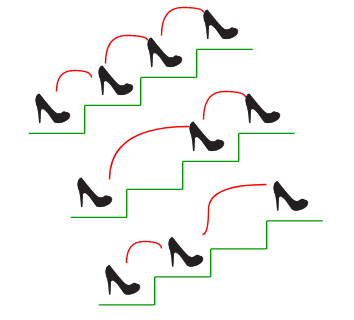
Count ways to reach the n’th stair

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.

[](https://media.geeksforgeeks.org/wp-content/uploads/nth-stair.png)

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](http://www.maths.surrey.ac.uk/hosted-sites/R.Knott/Fibonacci/fibpuzzles.html)

It’s always good to start off with some test cases. Let’s start with small cases and see if we can find some sort of pattern.

* N = 1, 1 way to climb: [1]
* N = 2, 2 ways to climb: [1, 1], [2]
* N = 3, 3 ways to climb: [1, 2], [1, 1, 1], [2, 1]
* N = 4, 5 ways to climb: [1, 1, 2], [2, 2], [1, 2, 1], [1, 1, 1, 1], [2, 1, 1]

Do you notice anything? When we look at N = 3, the number of ways to get to 3 steps is 3, and they’re based off N = 1 and N = 2. What’s the relationship?  
The only ways to get to N = 3, is to first get to N = 1, and then go up by 2 steps, or get to N = 2 and go up by 1 step. So f(3) = f(2) + f(1).

# A program to count the number of ways to reach n'th stair

# Recurssive program to find n'th fibonacci number

def fib(n):

    if n <= 1:

        return n

    return fib(n-1) + fib(n-2)

# returns no. of ways to reach s'th stair

def countWays(s):

    return fib(s + 1)

# Driver program

s = 4

print "Number of ways = ",

print countWays(s)

Non-recursive approach:-

def countWaysUtil(n,m):

    res = [0 for x in range(n)] # Creates list res witth all elements 0

    res[0],res[1] = 1,1

    for i in range(2,n):

        j = 1

        while j<=m and j<=i:

            res[i] = res[i] + res[i-j]

            j = j + 1

    return res[n-1]

# Returns number of ways to reach s'th stair

def countWays(s,m):

    return countWaysUtil(s+1, m)

# Driver Program

s,m = 4,2

print "Nmber of ways =",countWays(s,m)

# Cutting a Rod | DP-13

Given a rod of length n inches and an array of prices that contains prices of all pieces of size smaller than n. Determine the maximum value obtainable by cutting up the rod and selling the pieces. For example, if length of the rod is 8 and the values of different pieces are given as following, then the maximum obtainable value is 22 (by cutting in two pieces of lengths 2 and 6)

length | 1 2 3 4 5 6 7 8

--------------------------------------------

price | 1 5 8 9 10 17 17 20

And if the prices are as following, then the maximum obtainable value is 24 (by cutting in eight pieces of length 1)

length | 1 2 3 4 5 6 7 8

--------------------------------------------

price | 3 5 8 9 10 17 17 20

# A Dynamic Programming solution for Rod cutting problem

INT\_MIN = -32767

# Returns the best obtainable price for a rod of length n and

# price[] as prices of different pieces

def cutRod(price, n):

    val = [0 for x in range(n+1)]

    val[0] = 0

    # Build the table val[] in bottom up manner and return

    # the last entry from the table

    for i in range(1, n+1):

        max\_val = INT\_MIN

        for j in range(i):

             max\_val = max(max\_val, price[j] + val[i-j-1])

        val[i] = max\_val

    return val[n]

# Driver program to test above functions

arr = [1, 5, 8, 9, 10, 17, 17, 20]

size = len(arr)

print("Maximum Obtainable Value is " + str(cutRod(arr, size)))

Coin change problem:-

Go to Jenny’s lecture in youtube and take the solution.

# Min Cost Path | DP-6

Look at Tushar roy -coding makes simple problem and geek for geeks