7. Reach a given score

Easy Accuracy: 76.47% Submissions: 2898 Points: 2

Consider a game where a player can score **3** or **5** or **10** points in a move. Given a total score **n**, find the number of distinct combinations to reach the given score.

Example 1:

```
Input:
n = 8
Output: 1
Explanation:when n = 8,{3,5} and {5,3}
are the two possible permutations but
these represent the same combination.
Hence output is 1.
```

Example 2:

```
Input:
n = 20
Output: 4
Explanation:When n = 20, {10,10},
{5,5,5,5},{10,5,5} and {3,3,3,3,3,5}
are different possible permutations.
Hence output will be 4.
```

Your Task:

Complete **count()** function which takes N as an argument and returns the **number of ways/combinations** to reach the given score.

Expected Time Complexity: O(N).

Expected Auxiliary Space: O(N).

Constraints:

1 ≤ n ≤ 1000

```
public static int count(int n)
{
   //Your code here
}
```

10. Count ways to reach the n'th stair

Medium Accuracy: 42.67% Submissions: 58239 Points: 4

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 (**order does matter**).

Example 1:

```
Input:
n = 4
Output: 5
Explanation:
You can reach 4th stair in 5 ways.
Way 1: Climb 2 stairs at a time.
Way 2: Climb 1 stair at a time.
Way 3: Climb 2 stairs, then 1 stair
and then 1 stair.
Way 4: Climb 1 stair, then 2 stairs
then 1 stair.
Way 5: Climb 1 stair, then 1 stair and
then 2 stairs.
```

Example 2:

```
Input:
n = 10
Output: 89
Explanation:
There are 89 ways to reach the 10th stair.
```

Your Task:

Complete the function **countWays()** which takes the top stair number m as input parameters and returns the answer **% 10^9+7**.

Expected Time Complexity : O(n)

Expected Auxiliary Space: O(1)

Constraints:

$1 \leq n \leq 10_4$

```
class Solution
{
   //Function to count number of ways to reach the nth stair.
   int countWays(int n)
   {

       // your code here
   }
}
```

11. Count ways to N'th Stair(Order does not matter)

Medium Accuracy: 51.45% Submissions: 29132 Points: 4

There are **N** stairs, and 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 (**order does not matter**).

Note: Order does not matter means for n=4 {1 2 1},{2 1 1},{1 1 2} are considered same.

Example 1:

```
Input:
N = 4
Output: 3
Explanation: You can reach 4th stair in
3 ways.
3 possible ways are:
1, 1, 1, 1
1, 1, 2
2, 2
```

Example 2:

```
Input:
N = 5
Output: 3
Explanation:
You may reach the 5th stair in 3 ways.
The 3 possible ways are:
1, 1, 1, 1, 1
1, 1, 1, 2
1, 2, 2
```

Your Task:

Your task is to complete the function **countWays()** which takes single argument(N) and returns the answer.

Expected Time Complexity: O(N)

Expected Auxiliary Space: O(N)

Constraints:

1 <= N <= 10^6

```
class Solution
{
    //Function to count number of ways to reach the nth stair
    //when order does not matter.
    Long countWays(int m)
    {
        // your code here
    }
}
```

16. Unique BST's

Medium Accuracy: 44.17% Submissions: 41964 Points: 4

Given an integer. Find how many **structurally unique binary search trees** are there that stores the values from 1 to that integer (inclusive).

Example 1:

```
Input:
N = 2
Output: 2
Explanation:for N = 2, there are 2 unique
BSTs
     1             2
     \            /
     2        1
```

Example 2:

```
Input:
N = 3
Output: 5
Explanation: for N = 3, there are 5
possible BSTs
  1           3    3    2    1
     \        /     /   /  \      \
     3      2     1      1  3    2
     /      /       \                 \
   3     1        2                 3
```

Your Task:
You don't need to read input or print anything. Your task is to complete the function **numTrees()** which takes the integer N as input and returns the total number of Binary Search Trees possible with keys [1.....N] inclusive. Since the answer can be very large, return the **answer modulo 1e9 + 7**.
Expected Time Complexity: O(N2).
Expected Auxiliary Space: O(N).
Constraints:
1<=N<=1000

```
class Solution
{
    //Function to return the total number of possible unique BST.
    static int numTrees(int N)
    {
        // Your code goes here
```

```
   }
}
```

18. Max sum subarray by removing at most one element

Medium Accuracy: 46.3% Submissions: 14110 Points: 4

You are given array **A** of size **n**. You need to find the maximum-sum sub-array with the condition that you are allowed to skip at most one element.

Example 1:

```
Input:
n = 5
A[] = {1,2,3,-4,5}
Output: 11
Explanation: We can get maximum sum
subarray by skipping -4.
```

Example 2:

```
Input:
n = 8
A[] = {-2,-3,4,-1,-2,1,5,-3}
Output: 9
Explanation: We can get maximum sum
subarray by skipping -2 as [4,-1,1,5]
sums to 9, which is the maximum
achievable sum.
```

Your Task:
Your task is to complete the function **maxSumSubarray** that take array and size as parameters and returns the maximum sum.
Expected Time Complexity: O(N).

Expected Auxiliary Space: O(N).

Constraints:

1 <= n <= 100

-10_3 <= A_i<= 10_3

```
class Solution
{
    //Function to return maximum sum subarray by removing at most one element.
    public static int maxSumSubarray(int A[], int n)
    {
     //add code here.
    }
}
```