**1269. Number of Ways to Stay in the Same Place After Some Steps: -**

Hard Accepted: 56.4K Submissions: 118.7K Acceptance Rate: 47.5%

You have a pointer at index 0 in an array of size arrLen. At each step, you can move 1 position to the left, 1 position to the right in the array, or stay in the same place (The pointer should not be placed outside the array at any time).

Given two integers steps and arrLen, return the number of ways such that your pointer is still at index 0 after **exactly** steps steps. Since the answer may be too large, return it **modulo** 109 + 7.

**Example 1:**

**Input:** steps = 3, arrLen = 2

**Output:** 4

**Explanation:** There are 4 differents ways to stay at index 0 after 3 steps.

Right, Left, Stay

Stay, Right, Left

Right, Stay, Left

Stay, Stay, Stay

**Example 2:**

**Input:** steps = 2, arrLen = 4

**Output:** 2

**Explanation:** There are 2 differents ways to stay at index 0 after 2 steps

Right, Left

Stay, Stay

**Example 3:**

**Input:** steps = 4, arrLen = 2

**Output:** 8

**Constraints:**

* 1 <= steps <= 500
* 1 <= arrLen <= 106

**Code: -**

class Solution {

public:

    int mod = 1e9 + 7;

    vector<vector<int>> dp;

    int helper(int steps, int ind, int n){

        // base case

        if(steps < 0)

            return 0;

        else if(steps == 0){

            if(ind == 0)

                return 1;

            else

                return 0;

        }

        // dp found case

        if(dp[ind][steps] != -1)

            return dp[ind][steps];

        // recursive case

        int ans = helper(steps-1, ind, n) % mod;

        if(0 <= ind - 1)

            ans = ((ans % mod) + (helper(steps-1, ind-1, n) % mod)) % mod;

        if(ind + 1 < n)

            ans = ((ans % mod) + (helper(steps-1, ind+1, n) % mod)) % mod;

        // return from current state

        return dp[ind][steps] = ans;

    }

    int numWays(int steps, int len) {

        len = min(steps, len);

        dp = vector<vector<int>> (len, vector<int>(steps+1, -1));

        return helper(steps, 0, len);

    }

};

**T.C: - O(steps2)**

**S.C: - O(steps2)**