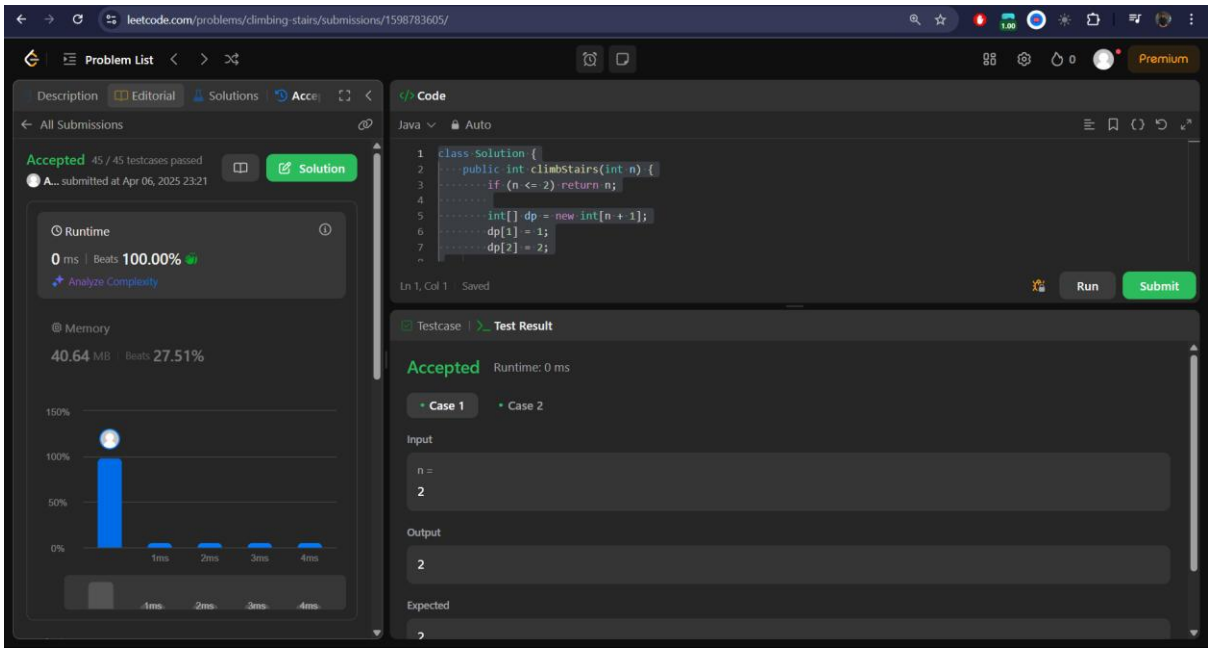
ASSIGNMENT 6

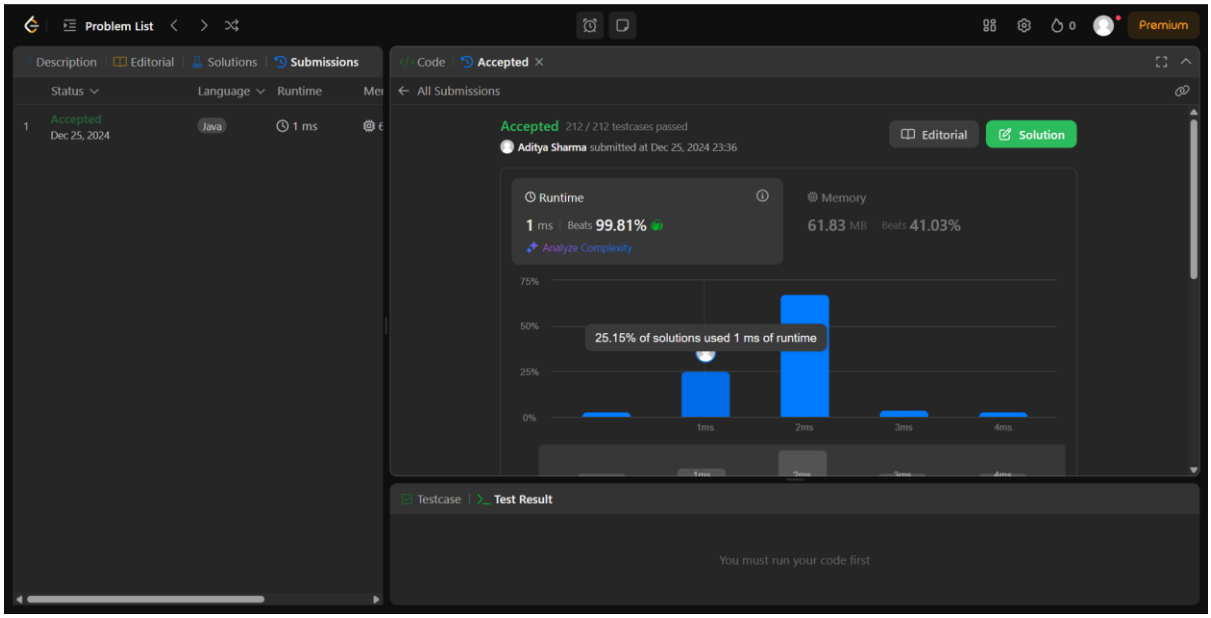
70. Climbing Stairs

```java
class Solution {
    public int climbStairs(int n) {
        if (n <= 2) return n;

        int[] dp = new int[n + 1];
        dp[1] = 1;
        dp[2] = 2;

        for (int i = 3; i <= n; i++) {
            dp[i] = dp[i - 1] + dp[i - 2];
        }

        return dp[n];
    }
}
```



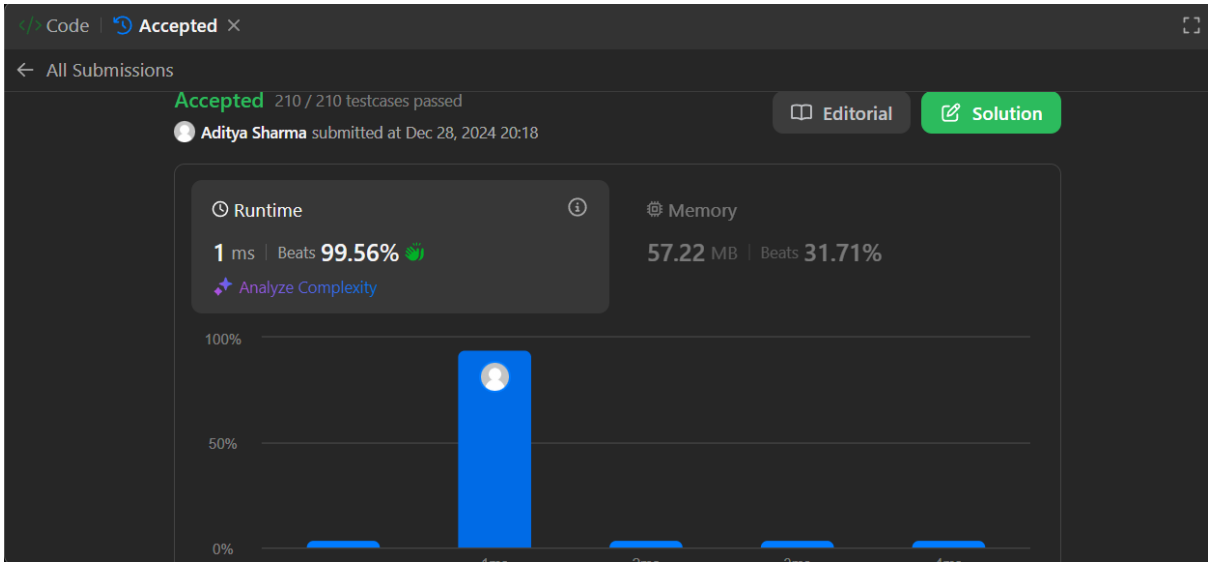
121. Best Time to Buy and Sell Stock

```
class Solution {
    public int maxProfit(int[] prices) {
            int lsf = Integer.MAX_VALUE;
        int op = 0;
        int pist = 0;

        for(int i = 0; i < prices.length; i++){
            if(prices[i] < lsf){
                lsf = prices[i];
            }
            pist = prices[i] - lsf;
            if(op < pist){
                op = pist;
            }
        }
        return op;
    }
}
```



53. Maximum Subarray

```java
class Solution {
    public int maxSubArray(int[] nums) {
        int maxSum = Integer.MIN_VALUE;
        int currentSum = 0;

        for (int i = 0; i < nums.length; i++) {
            currentSum += nums[i];

            if (currentSum > maxSum) {
                maxSum = currentSum;
            }

            if (currentSum < 0) {
                currentSum = 0;
            }
        }

        return maxSum;
    }
}
```



62. Unique Paths

```java
class Solution {
    public int uniquePaths(int m, int n) {
        // Create a 2D DP array filled with zeros
        int[][] dp = new int[m][n];

        // Initialize the rightmost column and bottom row to 1
        for (int i = 0; i < m; i++) {
            dp[i][n-1] = 1;
        }
        for (int j = 0; j < n; j++) {
            dp[m-1][j] = 1;
        }

        // Fill in the DP array bottom-up
        for (int i = m - 2; i >= 0; i--) {
            for (int j = n - 2; j >= 0; j--) {
                dp[i][j] = dp[i+1][j] + dp[i][j+1];
            }
        }

        // Return the result stored in the top-left corner
        return dp[0][0];
    }
}
```