**Assignment- 7**

**Student Name: Avin Kumar UID: 22BCS10600 Section/Group: IOT-614 B Semester: 6th**

**Subject Name: Advanced Programming Subject Code:** **22CSP-351**

**Lab-2**

**Dynamic Programming (Basic Problems)**

[**70. Climbing Stairs**](https://leetcode.com/problems/climbing-stairs/):

**Code:**

class Solution {

    public int climbStairs(int n) {

        if(n == 0 || n == 1){

        return 1;

        }

        int a = 1;

        int b = 1;

        for(int i = 2; i <= n; i++){

            int temp = b;

            b = a + b;

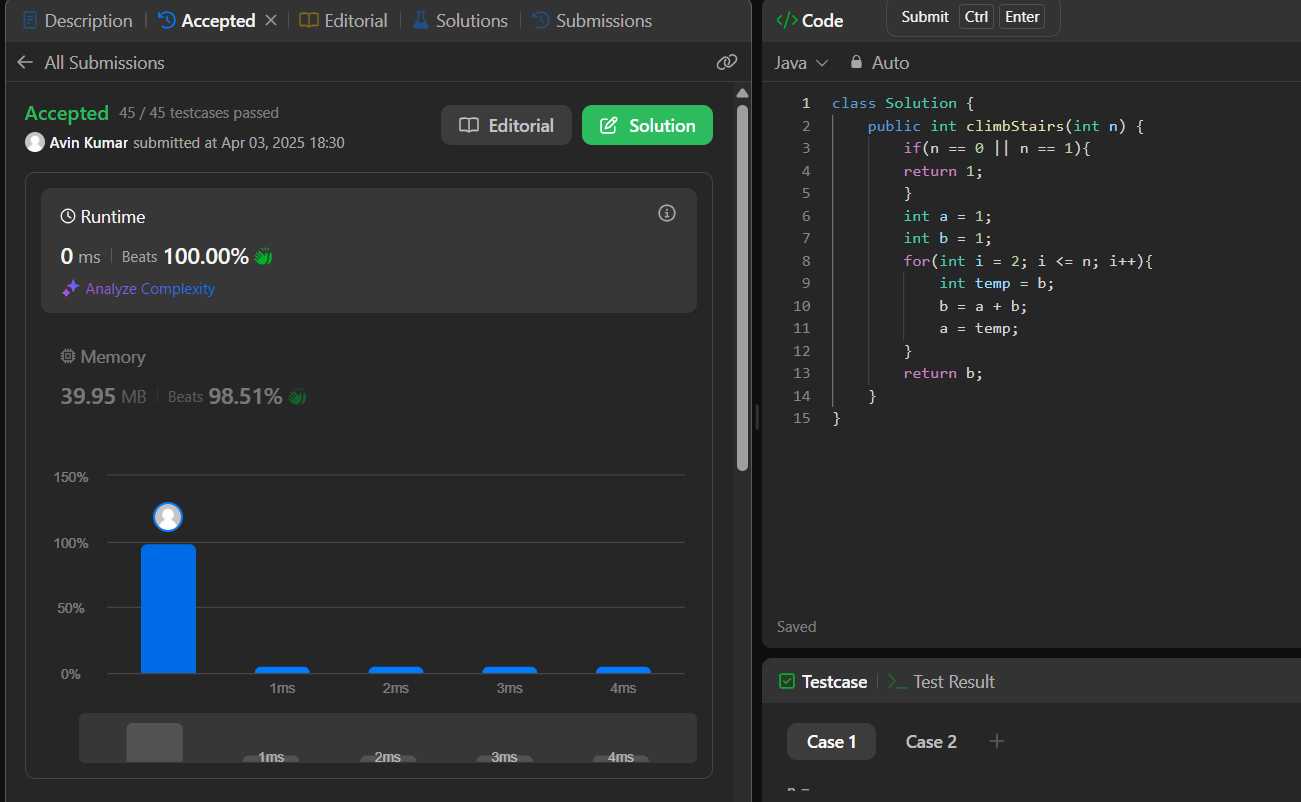
            a = temp;

        }

        return b;

    }

}



[**121. Best Time to Buy and Sell Stock**](https://leetcode.com/problems/best-time-to-buy-and-sell-stock/):

**Code:**

class Solution {

    public int maxProfit(int[] prices) {

        int buy = prices[0];

        int profit = 0;

        for (int i = 1; i < prices.length; i++) {

            if (prices[i] < buy) {

                buy = prices[i];

            } else if (prices[i] - buy > profit) {

                profit = prices[i] - buy;

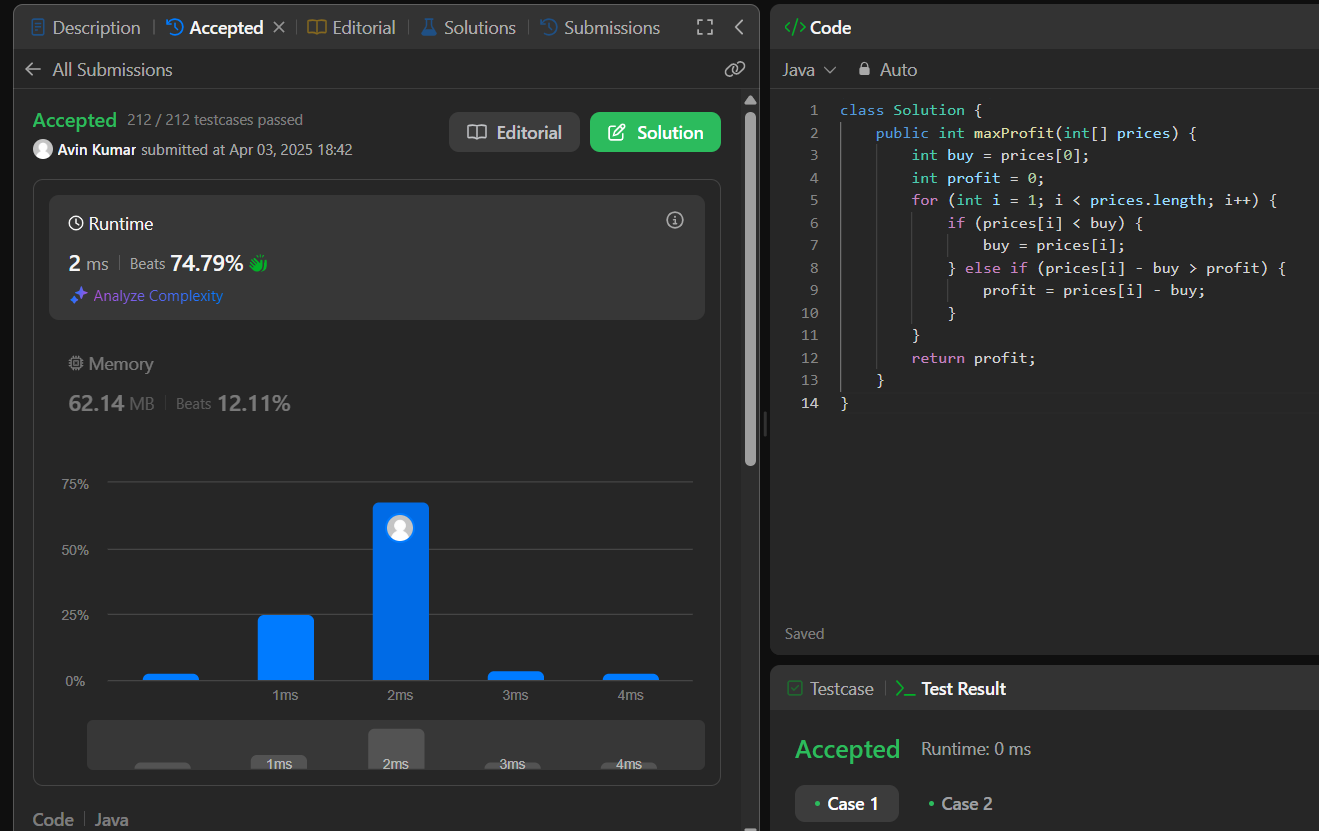
            }

        }

        return profit;

    }

}



[**53. Maximum Subarray**](https://leetcode.com/problems/maximum-subarray/):

**Code:**

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;

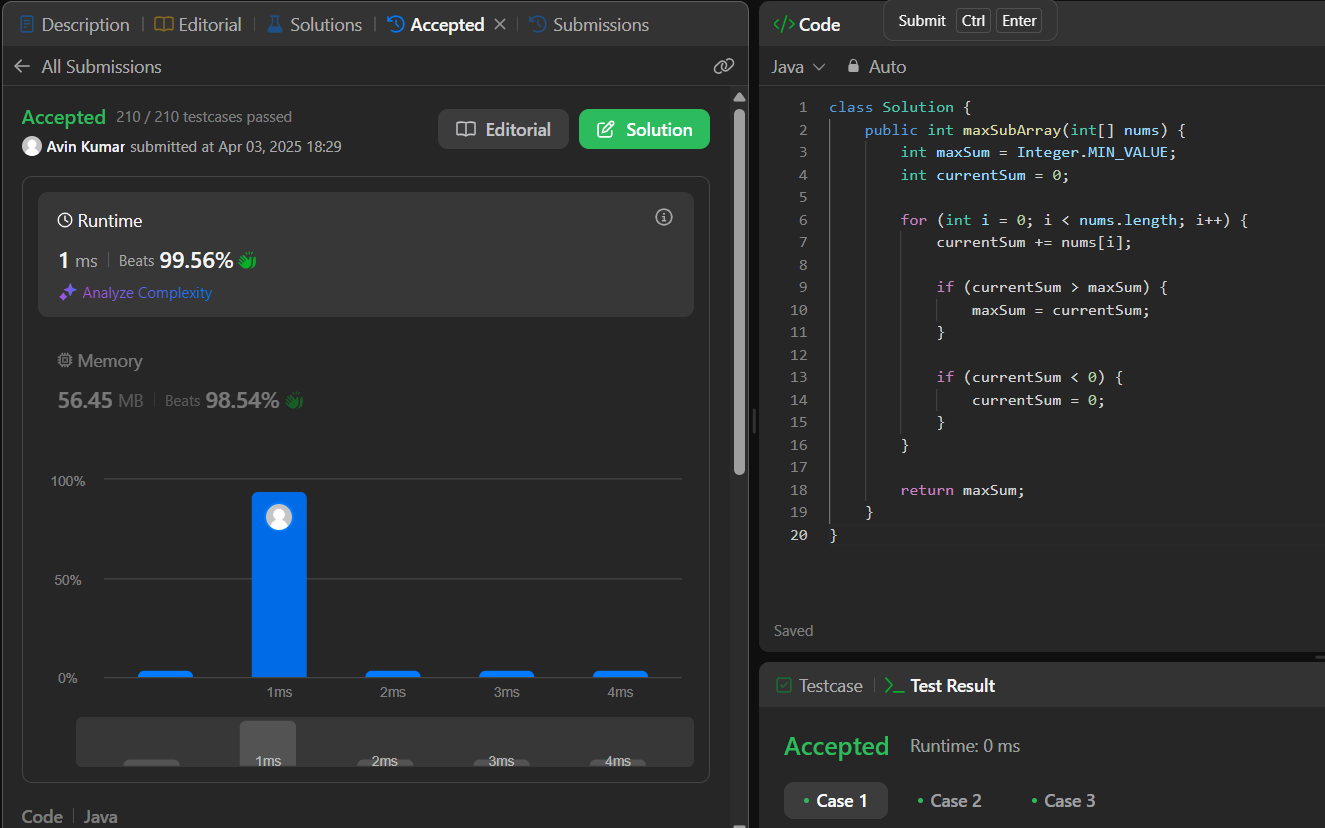
            }

        }

        return maxSum;

    }

}



[**198. House Robber**](https://leetcode.com/problems/house-robber/):

**Code:**

class Solution {

    private int rec(int idx, int[] dp,int[] nums)

    {

        if(idx==0) return nums[0];

        if(idx==1) return nums[1];

        if(dp[idx]!=-1) return dp[idx];

        for(int i = idx-2;i>=0;i--)

        {

            int val = nums[idx] + rec(i,dp,nums) ;

            dp[idx] = Math.max(dp[idx],val);

        }

        return dp[idx];

    }

    public int rob(int[] nums) {

        int n = nums.length;

        if(n==1) return nums[0];

        if(n==2) return Math.max(nums[0],nums[1]);

        int[] dp = new int[n];

        for(int i =0 ;i<n;i++) dp[i]=-1;

        int a = rec(n-1,dp,nums);

        int b = rec(n-2,dp,nums);

        return Math.max(a,b);

    }

}

