**ADVANCED PROGRAMMING-II**

**ASSIGNMENT-07**

**Name :**Sanjusha Singh

**UID:** 22BCS13130

**Q1 Climbing Stairs:**

**CODE:**

class Solution {

    public int climbStairs(int n) {

       if (n<=3 ){

        return n;

       }

       int sum = 0;

       int first = 2 ;

       int sec = 3;

       for(int i=4 ; i<=n ; i++){

        sum = first + sec;

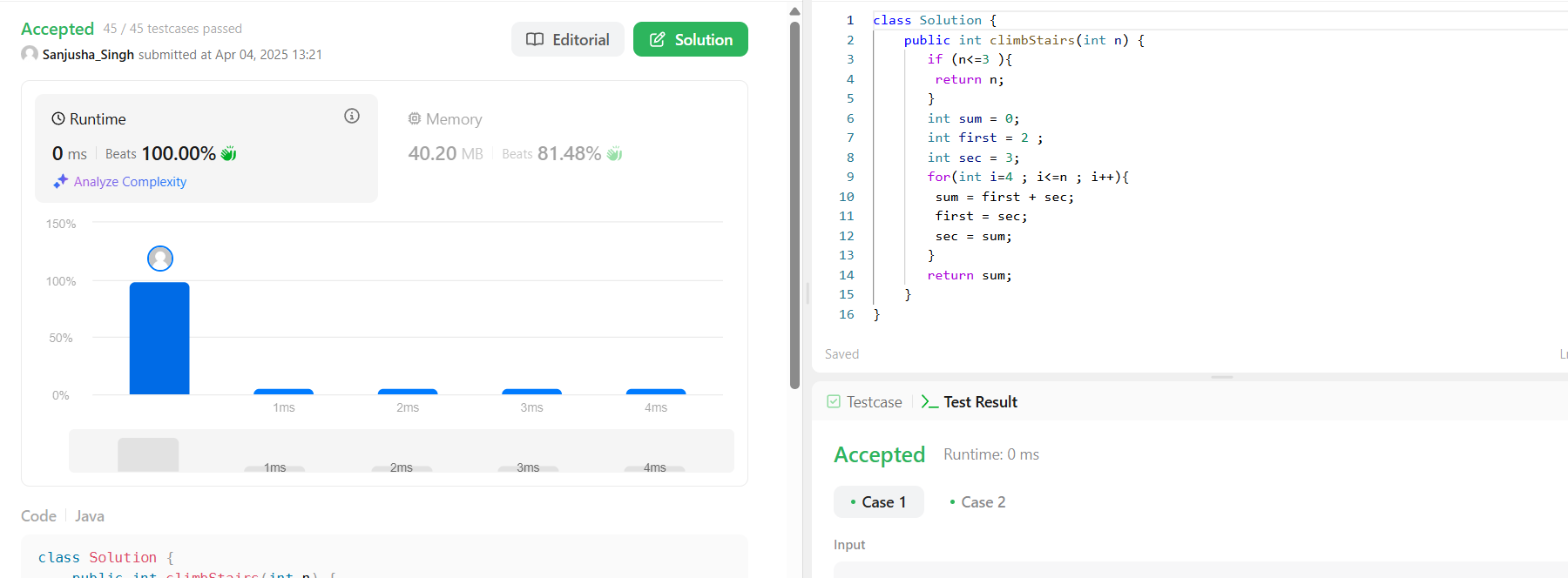
        first = sec;

        sec = sum;

       }

       return sum;

    }



**Q2 Maximum Subarray :**

**CODE:**

class Solution {

    public int maxSubArray(int[] nums) {

        int res = nums[0];

        int total = 0;

        for (int n : nums) {

            if (total < 0) {

                total = 0;

            }

            total += n;

            res = Math.max(res, total);

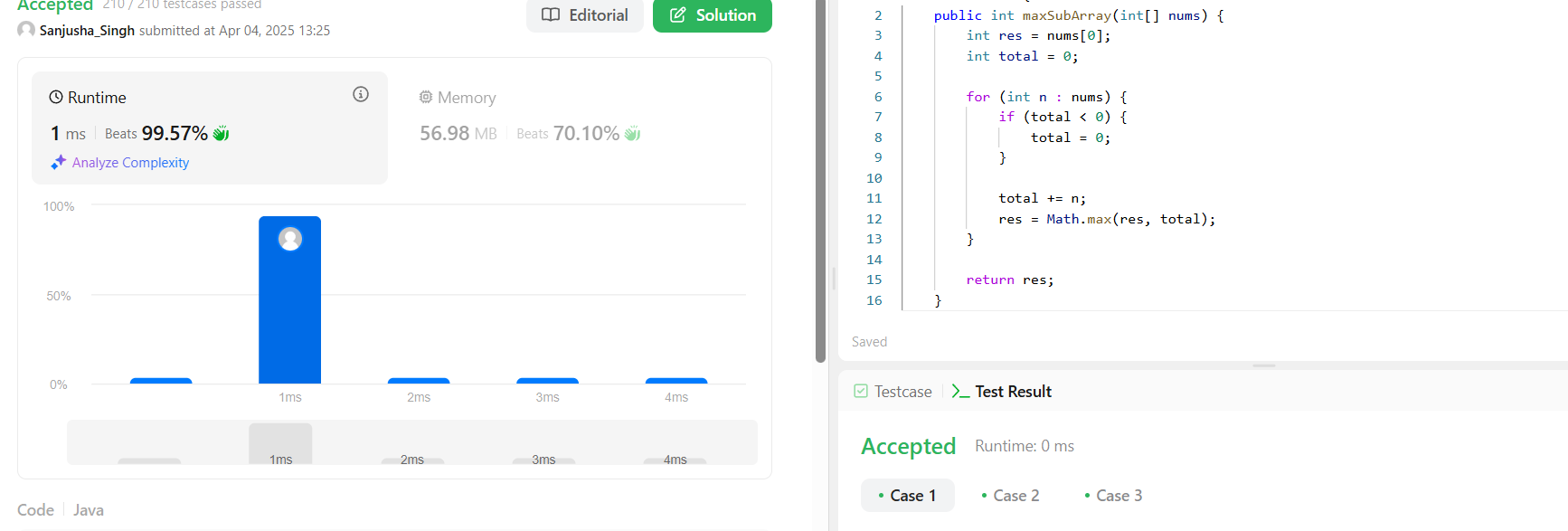
        }

        return res;

    }

}

**SCREENSHOT:**

****

**Q3 Jump Game:**

**CODE:**

class Solution {

    public boolean canJump(int[] nums) {

        int goal = nums.length - 1;

        for (int i = nums.length - 2; i >= 0; i--) {

            if (i + nums[i] >= goal) {

                goal = i;

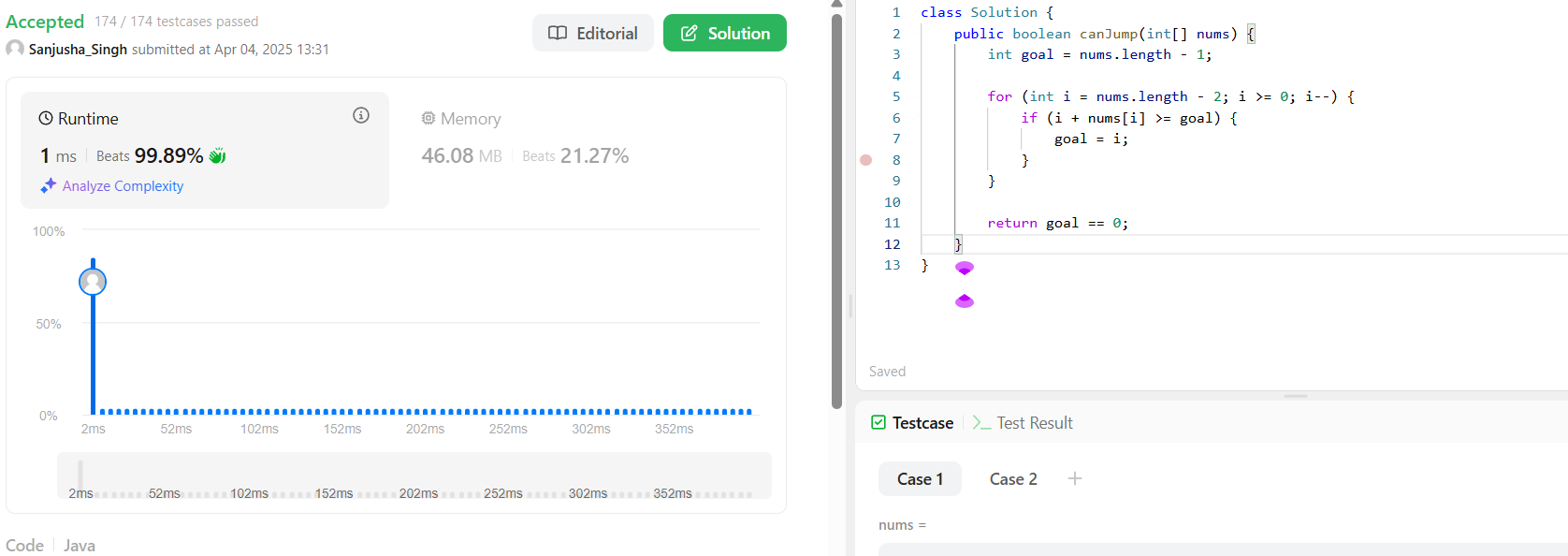
            }

        }

        return goal == 0;

    }

}

**SCREENSHOT: **

**Q4.Unique Paths :**

**CODE:**

class Solution {

    public int uniquePaths(int m, int n) {

        int[] aboveRow = new int[n];

        Arrays.fill(aboveRow, 1);

        for (int row = 1; row < m; row++) {

            int[] currentRow = new int[n];

            Arrays.fill(currentRow, 1);

            for (int col = 1; col < n; col++) {

                currentRow[col] = currentRow[col - 1] + aboveRow[col];

            }

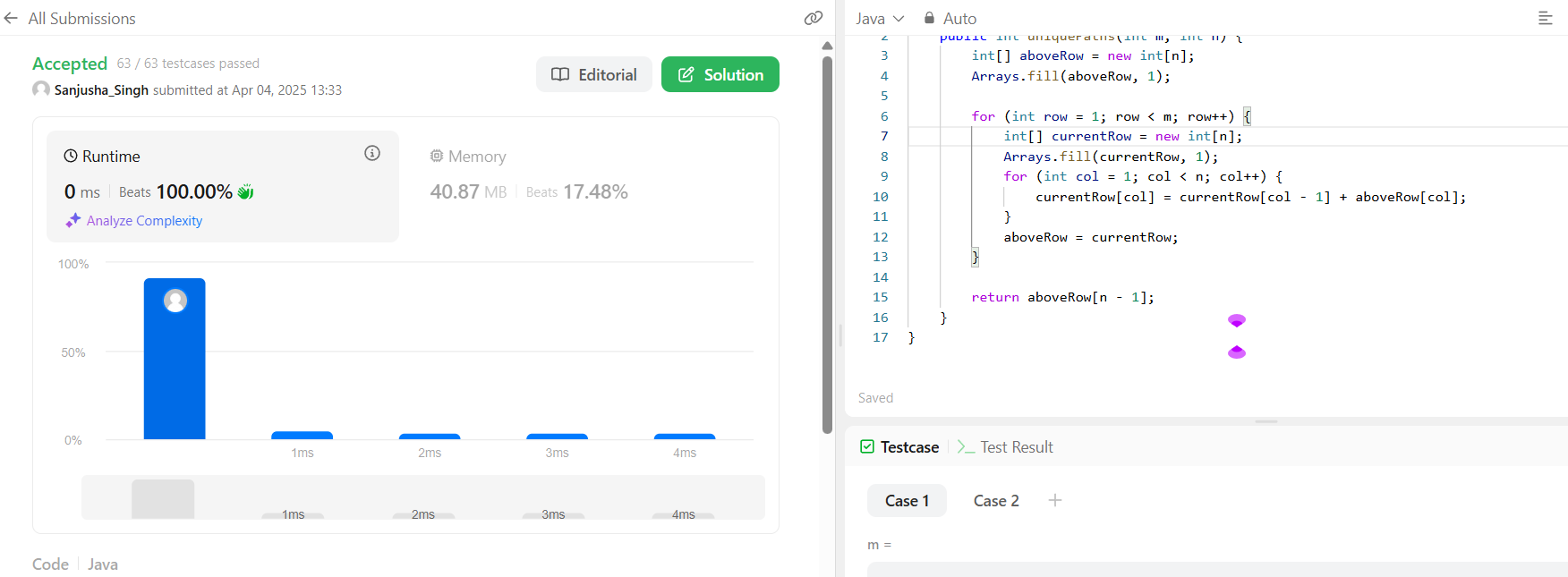
            aboveRow = currentRow;

        }

        return aboveRow[n - 1];

    }

}

**SCREENSHOT:** 

**Q5 Maximum Product Subarray :**

**CODE:**

class Solution {

public int maxProduct(int[] nums) {

int res = Integer.MIN\_VALUE;

for (int n : nums) {

res = Math.max(res, n);

}

int curMax = 1, curMin = 1;

for (int n : nums) {

int temp = curMax \* n;

curMax = Math.max(temp, Math.max(curMin \* n, n));

curMin = Math.min(temp, Math.min(curMin \* n, n));

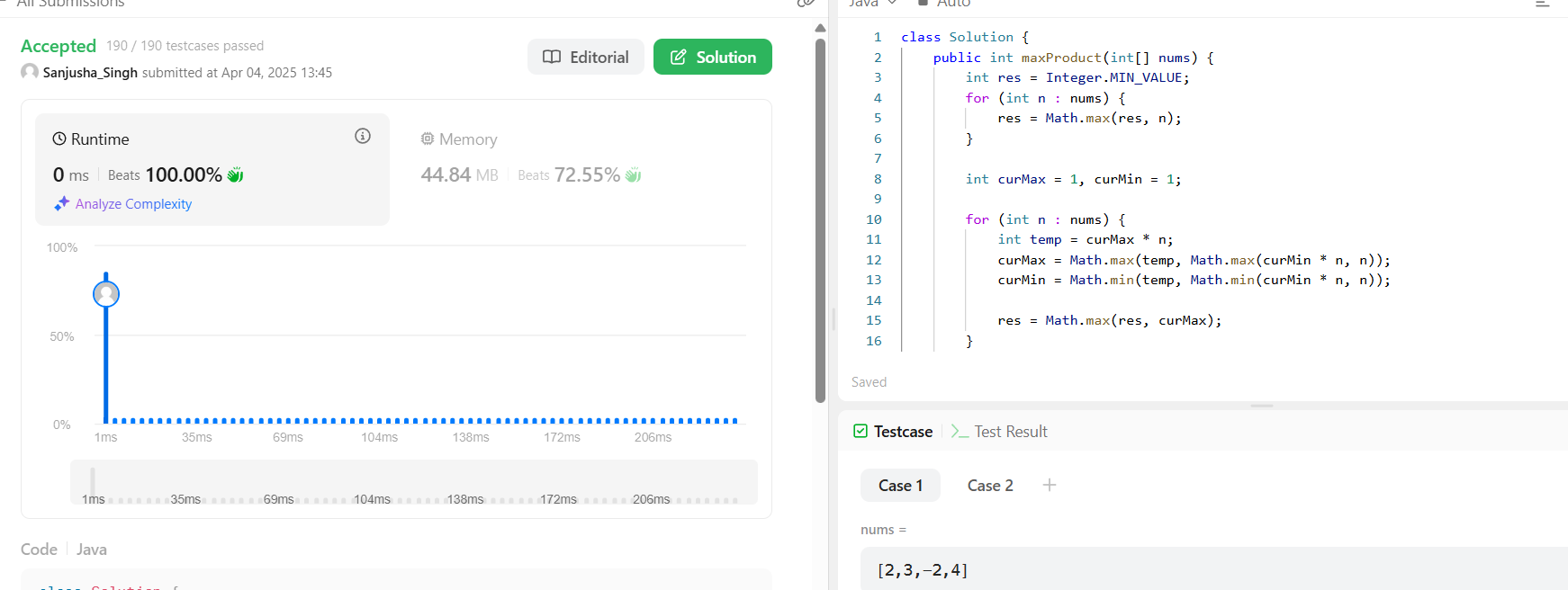
res = Math.max(res, curMax);

}

return res;

}

}**SCREENSHOT:**

****

**LEETCODE PROFILE LINK:** https://leetcode.com/u/Sanjusha\_Singh/

**GITHUB LINK: https**://github.com/SanjushaSingh20