

Experiment 6

Name: Saurav Saha

UID: 22BCS16589

Branch: BE-CSE

Section/Group: 22BCS_IOT-637/B

Semester: 6th

Subject Code: 22CSP-351

Subject Name: AP Lab-II

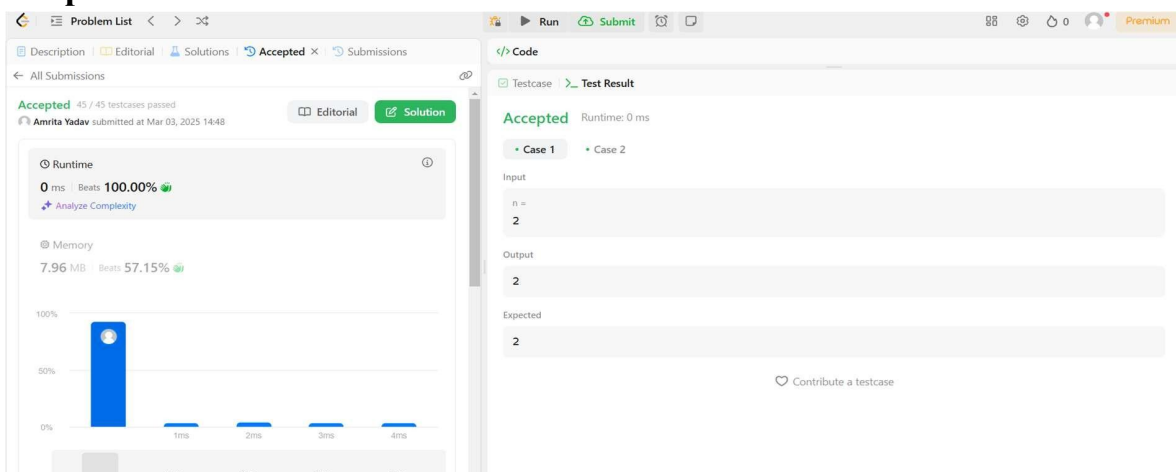
A. Climbing Stairs

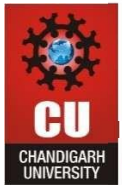
1. Aim: You are climbing a staircase. It takes n steps to reach the top. Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

2. Code

```
class Solution
{ public:
    int climbStairs(int n)
    { if (n <= 2) return n;
      int oneStepBefore = 2;
      int twoStepsBefore = 1;
      int allWays = 0;
      for (int i = 3; i <= n; i++) {
          allWays = oneStepBefore + twoStepsBefore;
          twoStepsBefore = oneStepBefore;
          oneStepBefore = allWays;
      }
      return allWays;
    }
};
```

3. Output:





4. Link: <https://leetcode.com/problems/climbing-stairs/submissions/1561256026/>

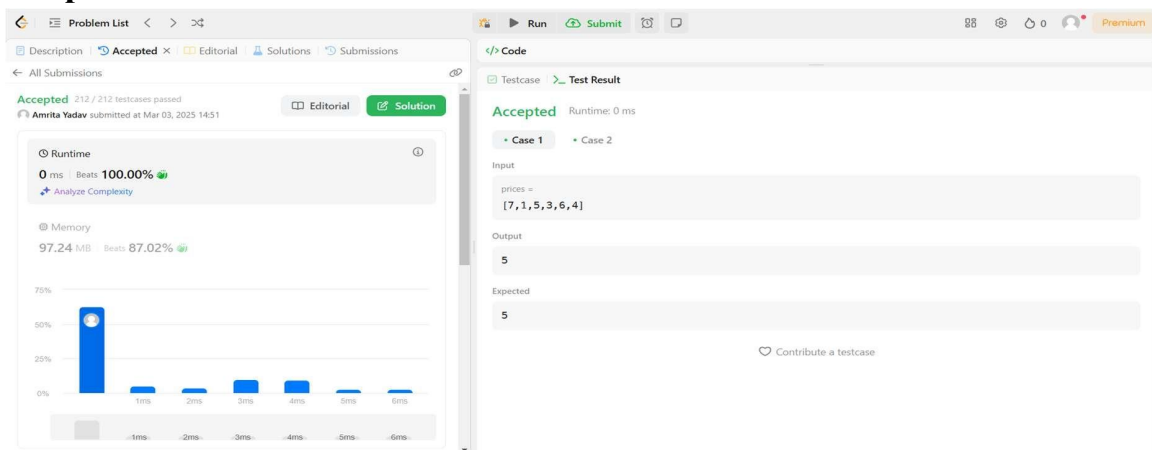
B. Best Time to Buy and Sell a Stock

1. **Aim:** You are given an array prices where prices[i] is the price of a given stock on the ith day. You want to maximize your profit by choosing a single day to buy one stock and choosing a different day in the future to sell that stock. Return the maximum profit you can achieve from this transaction. If you cannot achieve any profit, return 0.

2. Code

```
class Solution
{ public:
    int maxProfit(vector<int>& prices)
    { int minPrice = INT_MAX;
      int maxProfit = 0;
      for (int price : prices)
      { if (price < minPrice)
        {
            minPrice = price;
        } else if (price - minPrice > maxProfit)
        { maxProfit = price - minPrice;
        }
      }
      return maxProfit;
    }
};
```

3. Output:





4. Link: <https://leetcode.com/problems/best-time-to-buy-and-sell-stock/submissions/1561258258/>

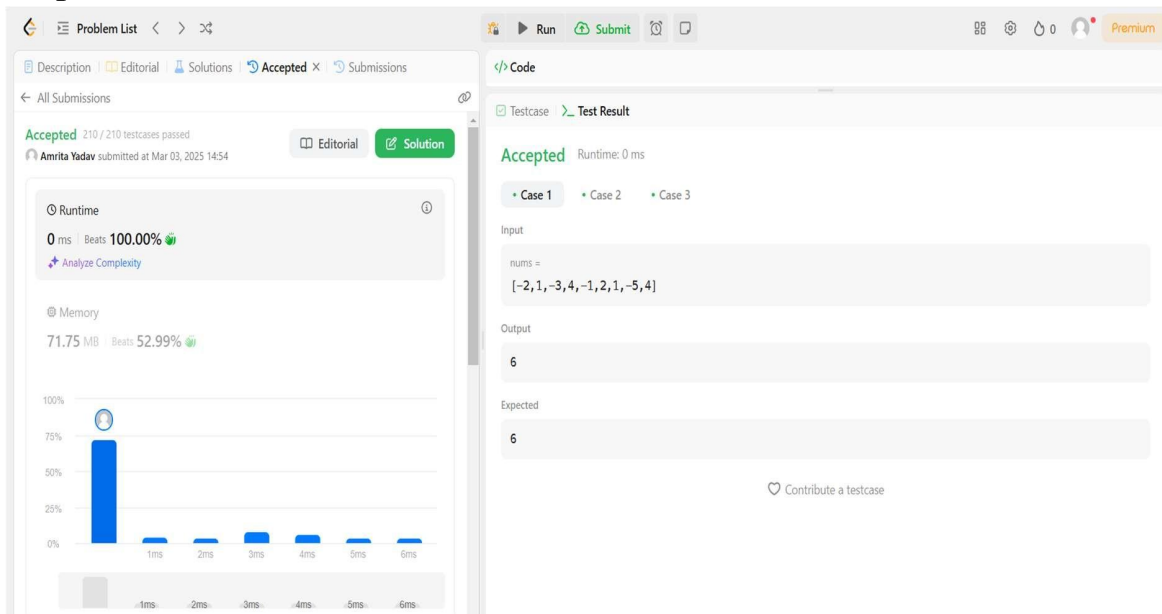
C. Maximum Subarray

1. **Aim:** Given an integer array nums, find the subarray with the largest sum, and return its sum.

2. Code

```
class Solution
{ public:
    int maxSubArray(vector<int>& nums)
    { int maxSoFar = nums[0];
      int maxEndingHere = nums[0];
      for (int i = 1; i < nums.size(); i++) {
          maxEndingHere = max(nums[i], maxEndingHere + nums[i]);
          maxSoFar = max(maxSoFar, maxEndingHere);
      }
      return maxSoFar;
    }
};
```

3. Output:



4. Link: <https://leetcode.com/problems/maximum-subarray/submissions/1561260616/>