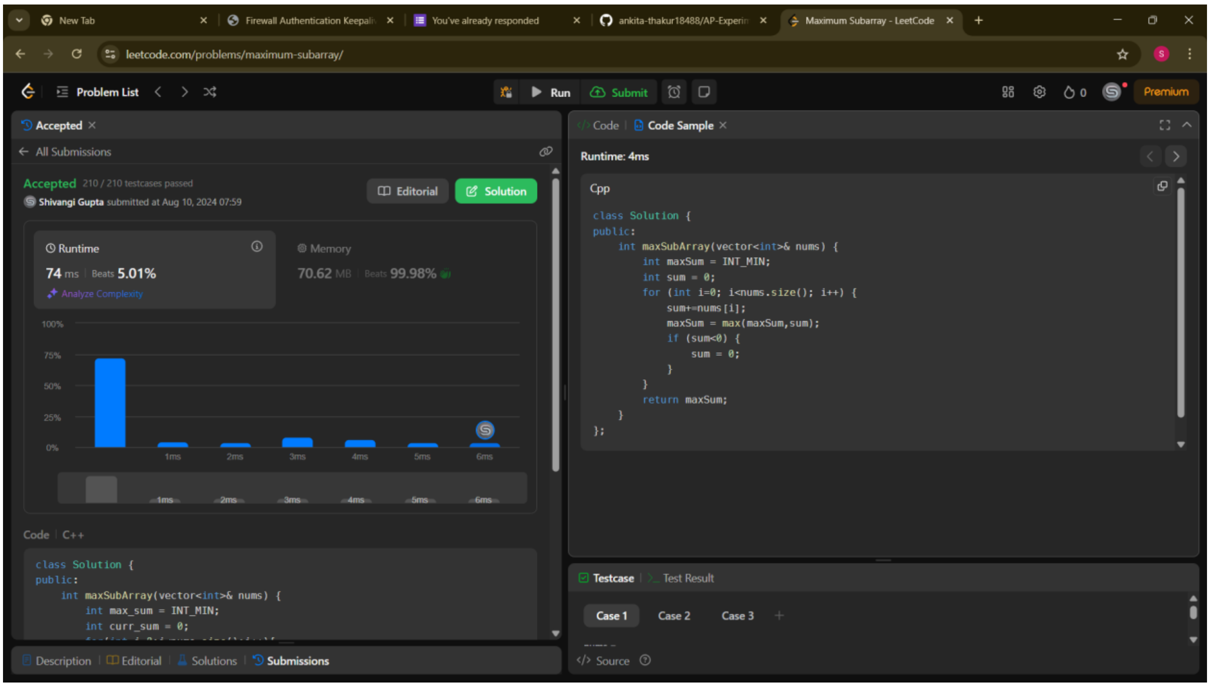
AP Assignment-7

Name: Shivangi Gupta UID: 22BCS15008 Section: 601-A

Q1. Maximum array https://leetcode.com/problems/maximum-subarray/description/

CODE:

```
//kadane's Algo

class Solution {

public:

    int maxSubArray(vector<int>& nums) {

        int max_sum = INT_MIN;

        int curr_sum = 0;

        for(int i=0;i<nums.size();i++){

            curr_sum += nums[i];

            if(curr_sum > max_sum){

                max_sum = curr_sum;

            }

            if(curr_sum<0){

                curr_sum = 0;

            }

        }

        return max_sum;

    }

};
```



Accepted 210 / 210 testcases passed

Shivangi Gupta submitted at Aug 10, 2024 07:59

Runtime

74 ms | Beats 5.01%

Memory

70.62 MB | Beats 99.98%

Code | C++

```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int max_sum = INT_MIN;
        int curr_sum = 0;
```

Runtime: 4ms

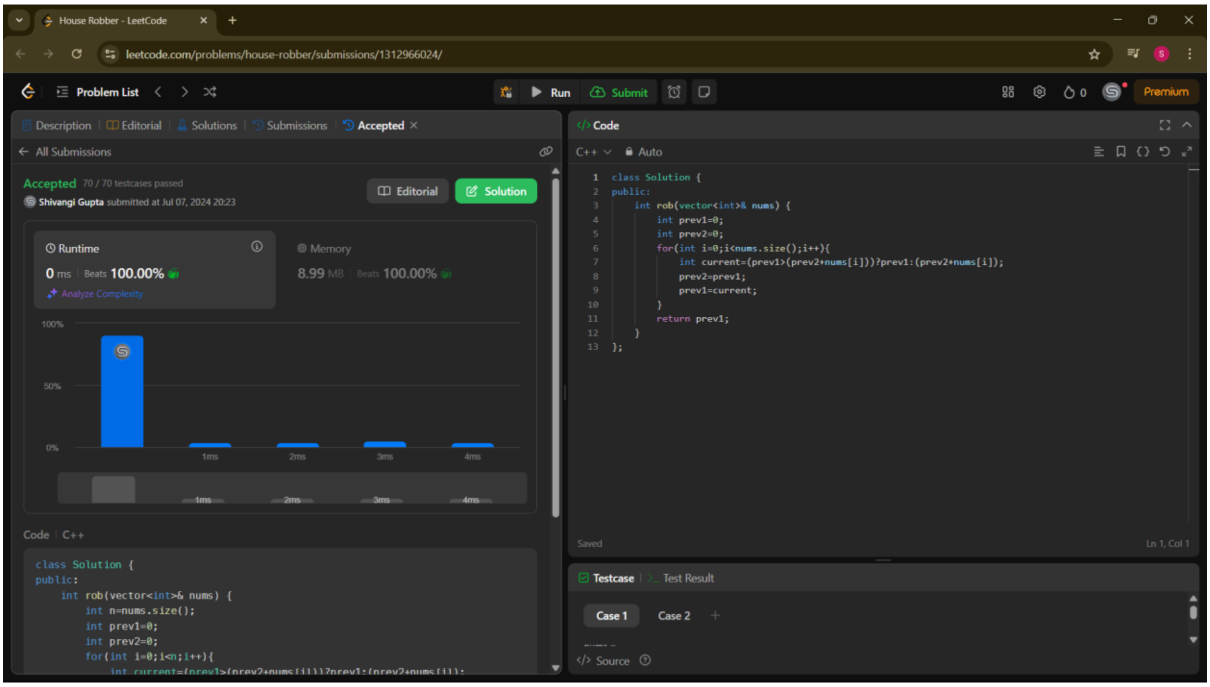
Cpp

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

Q2. House Robber https://leetcode.com/problems/house-robber/description/

CODE:

```
class Solution {
public:
    int rob(vector<int>& nums) {
        int prev1=0;
        int prev2=0;
        for(int i=0;i<nums.size();i++){
            int current=(prev1>(prev2+nums[i]))?prev1:(prev2+nums[i]);
            prev2=prev1;
            prev1=current;
        }
        return prev1;
    }
};
```



Accepted 70 / 70 testcases passed

Shivangi Gupta submitted at Jul 07, 2024 20:23

Runtime 0 ms | Beats 100.00%

Memory 8.99 MB | Beats 100.00%

Code | C++

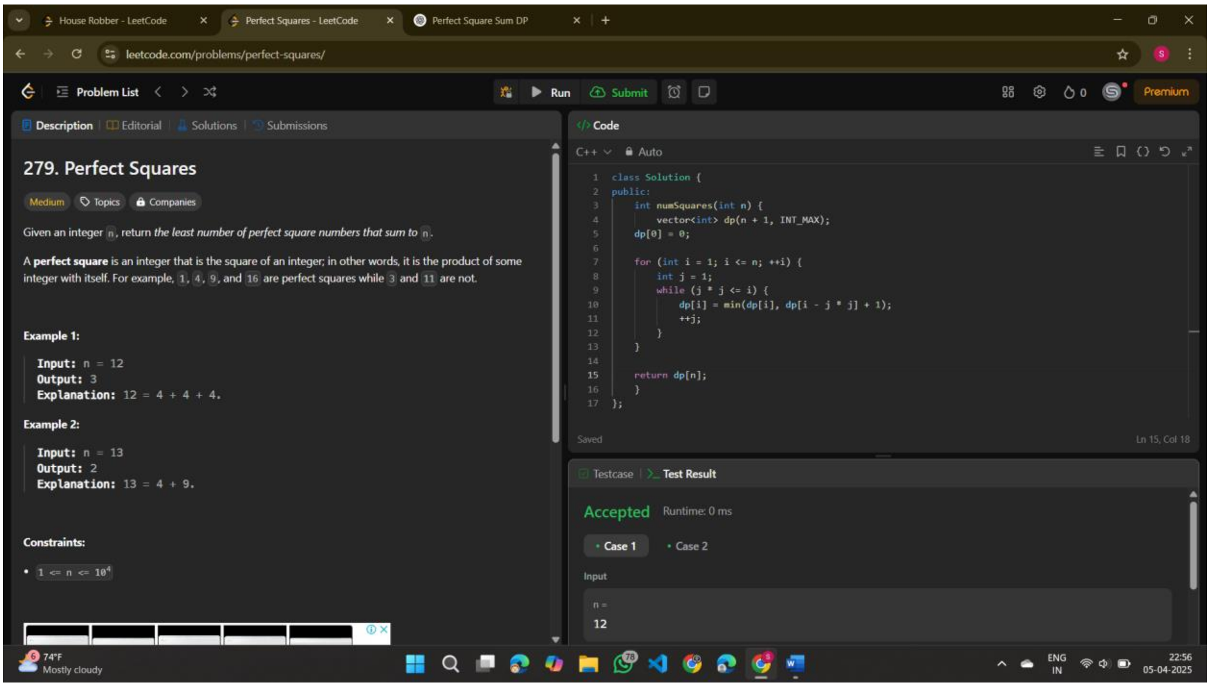
```
class Solution {
public:
    int rob(vector<int>& nums) {
        int n=nums.size();
        int prev1=0;
        int prev2=0;
        for(int i=0;i<n;i++){
            int current=(prev1>(prev2+nums[i]))?prev1:(prev2+nums[i]);
```

```
class Solution {
public:
    int rob(vector<int>& nums) {
        int prev1=0;
        int prev2=0;
        for(int i=0;i<nums.size();i++){
            int current=(prev1>(prev2+nums[i]))?prev1:(prev2+nums[i]);
            prev2=prev1;
            prev1=current;
        }
        return prev1;
    }
};
```

Q3. Perfect Squares https://leetcode.com/problems/perfect-squares/

Code:

```
class Solution {

public:

    int numSquares(int n) {

        vector<int> dp(n + 1, INT_MAX);

    dp[0] = 0;



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

        int j = 1;

        while (j * j <= i) {

            dp[i] = min(dp[i], dp[i - j * j] + 1);

            ++j;

        }

    }



    return dp[n];

    }

};
```



279. Perfect Squares

Given an integer `n`, return *the least number of perfect square numbers that sum to* `n`.

A **perfect square** is an integer that is the square of an integer; in other words, it is the product of some integer with itself. For example, `1`, `4`, `9`, and `16` are perfect squares while `3` and `11` are not.

Example 1:

```
Input: n = 12
Output: 3
Explanation: 12 = 4 + 4 + 4.
```

Example 2:

```
Input: n = 13
Output: 2
Explanation: 13 = 4 + 9.
```

Constraints:

- `1 <= n <= 10^4`

```
class Solution {
public:
    int numSquares(int n) {
        vector<int> dp(n + 1, INT_MAX);
    dp[0] = 0;

    for (int i = 1; i <= n; ++i) {
        int j = 1;
        while (j * j <= i) {
            dp[i] = min(dp[i], dp[i - j * j] + 1);
            ++j;
        }
    }

    return dp[n];
    }
};
```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

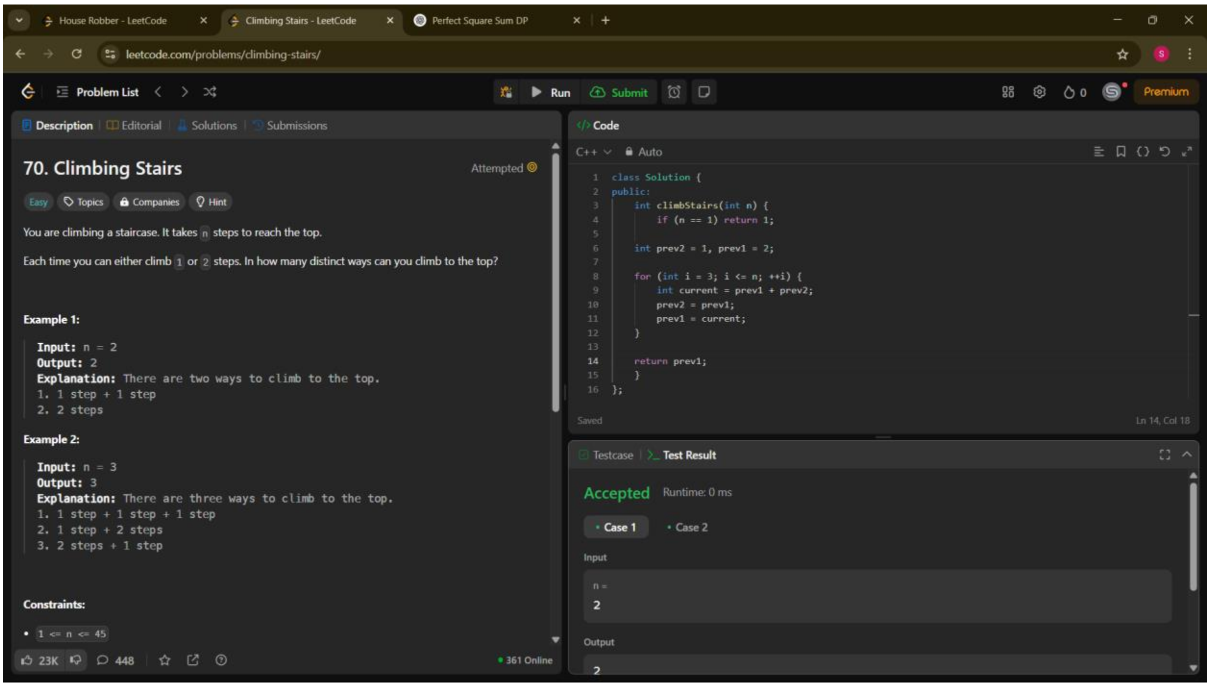
n =

12

Q4. Climbing Stairs https://leetcode.com/problems/climbing-stairs/

CODE:

```
class Solution {
public:
    int climbStairs(int n) {
        if (n == 1) return 1;

    int prev2 = 1, prev1 = 2;

    for (int i = 3; i <= n; ++i) {
        int current = prev1 + prev2;
        prev2 = prev1;
        prev1 = current;
    }

    return prev1;
    }
};
```



70. Climbing Stairs

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?

Example 1:

```
Input: n = 2
Output: 2
Explanation: There are two ways to climb to the top.
1. 1 step + 1 step
2. 2 steps
```

Example 2:

```
Input: n = 3
Output: 3
Explanation: There are three ways to climb to the top.
1. 1 step + 1 step + 1 step
2. 1 step + 2 steps
3. 2 steps + 1 step
```

Constraints:

- `1 <= n <= 45`

```
class Solution {
public:
    int climbStairs(int n) {
        if (n == 1) return 1;

    int prev2 = 1, prev1 = 2;

    for (int i = 3; i <= n; ++i) {
        int current = prev1 + prev2;
        prev2 = prev1;
        prev1 = current;
    }

    return prev1;
    }
};
```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

n =
2

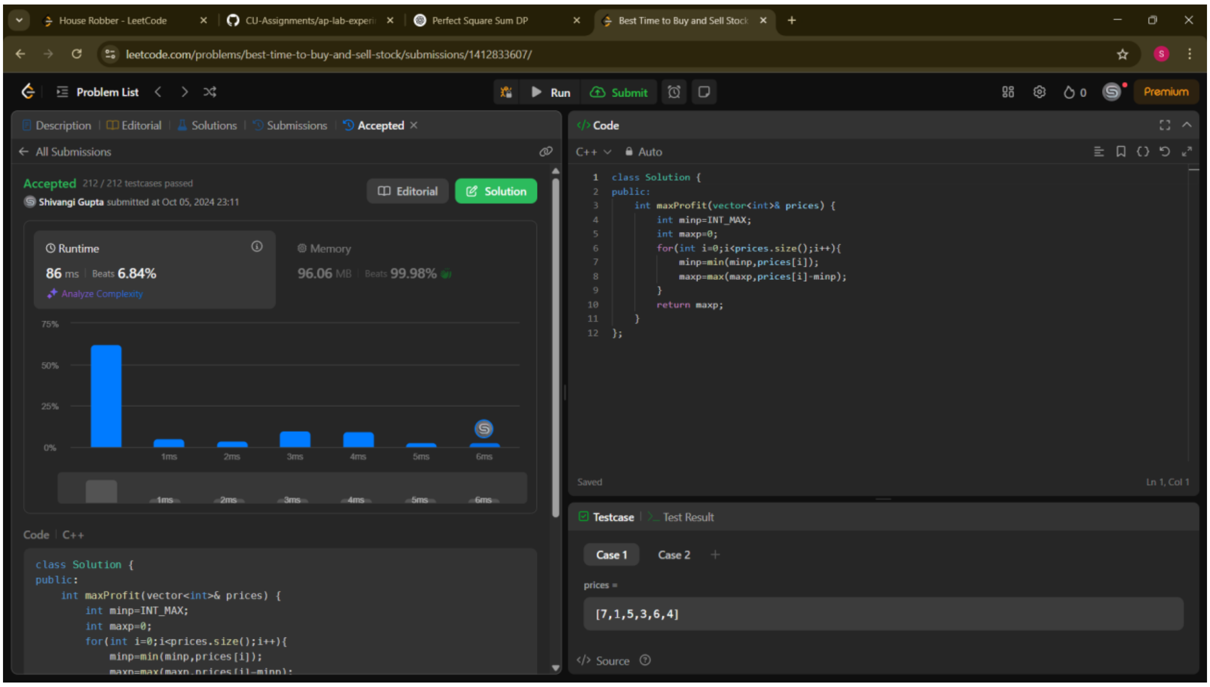
Output

2

Q5. Best time to buy and sell a stock https://leetcode.com/problems/best-time-to-buy-and-sell-stock/description/

CODE:

```
class Solution {

public:

   int maxProfit(vector<int>& prices) {

      int minp=INT_MAX;

      int maxp=0;

      for(int i=0;i<prices.size();i++){

         minp=min(minp,prices[i]);

         maxp=max(maxp,prices[i]-minp);

      }

      return maxp;

   }

};
```



Accepted 212 / 212 testcases passed

Shivangi Gupta submitted at Oct 05, 2024 23:11

Runtime 86 ms | Beats 6.84%

Memory 96.06 MB | Beats 99.98%

Code | C++

```
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        int minp=INT_MAX;
        int maxp=0;
        for(int i=0;i<prices.size();i++){
            minp=min(minp,prices[i]);
            maxp=max(maxp,prices[i]-minp);
```

```
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        int minp=INT_MAX;
        int maxp=0;
        for(int i=0;i<prices.size();i++){
            minp=min(minp,prices[i]);
            maxp=max(maxp,prices[i]-minp);
        }
        return maxp;
    }
};
```

prices =

```
[7,1,5,3,6,4]
```