**Experiment-7**

**House Robber**

**CODE:**

class Solution {

public:

    int rob(vector<int>& nums) {

  int rob = 0;

        int norob = 0;

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

            int newRob = norob + nums[i];

            int newNoRob = max(norob, rob);

            rob = newRob;

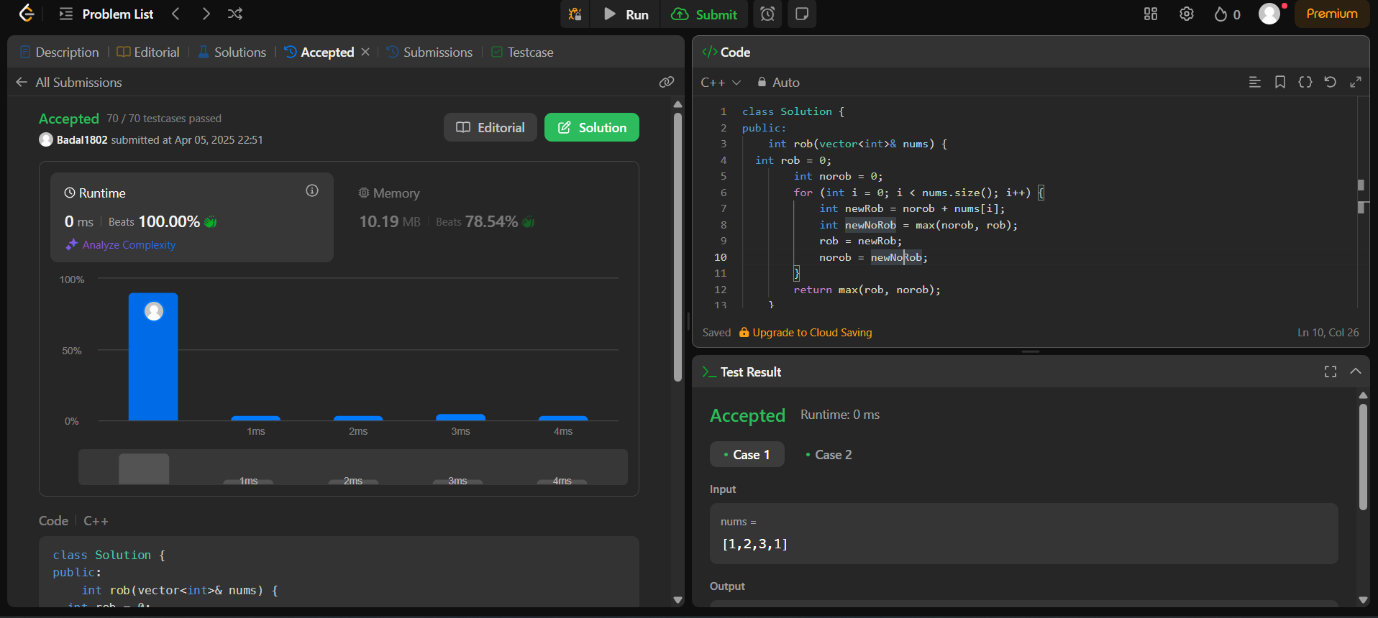
            norob = newNoRob;

        }

        return max(rob, norob);

    }

};

**OUTPUT:**

**Jump Game**

**CODE:**

class Solution {

public:

    bool canJump(vector<int>& nums) {

    int reach = 0;

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

            if (i > reach) return false;

            reach = max(reach, i + nums[i]);

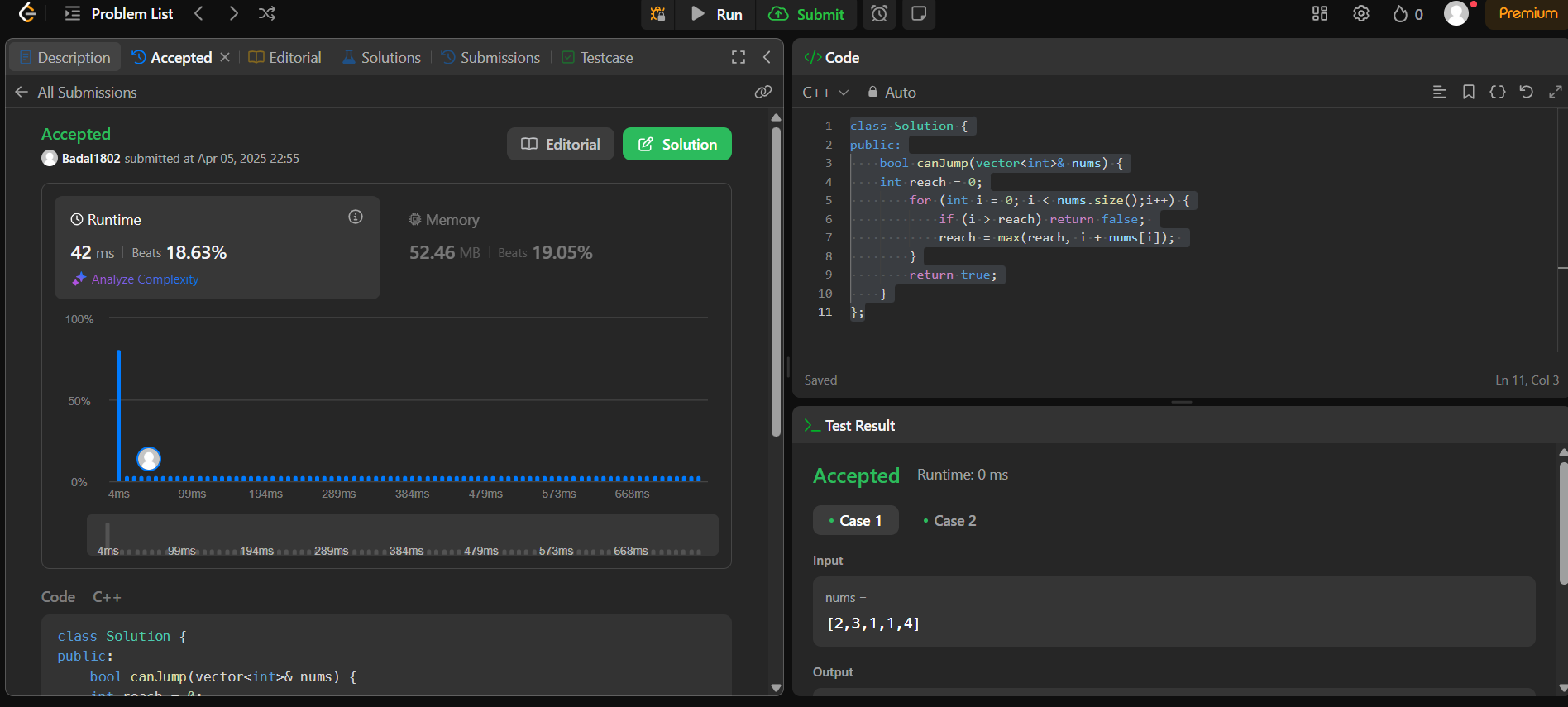
        }

        return true;

    }

};

**OUTPUT:**

****

**Maximum Product Subarray**

**CODE:**

class Solution {

public:

    int maxProduct(vector<int>& nums) {

        int maxi = INT\_MIN;

        int prod=1;

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

        {

          prod\*=nums[i];

          maxi=max(prod,maxi);

          if(prod==0)

           prod=1;

        }

        prod=1;

        for(int i=nums.size()-1;i>=0;i--)

        {

          prod\*=nums[i];

          maxi=max(prod,maxi);

          if(prod==0)

           prod=1;

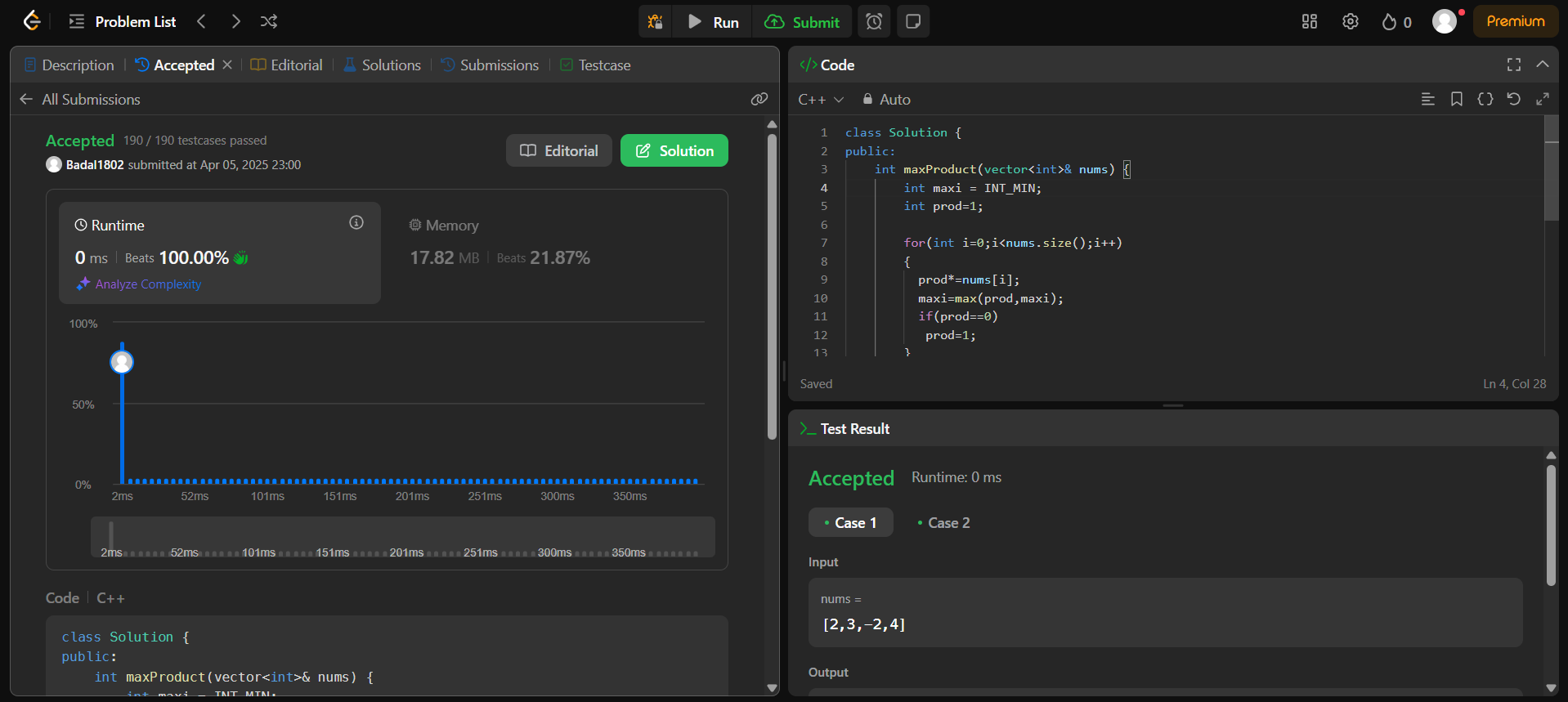
        }

        return maxi;

    }

};

**OUTPUT:**

****

**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) {

for (int j = 1; j \* j <= i; ++j){

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

}

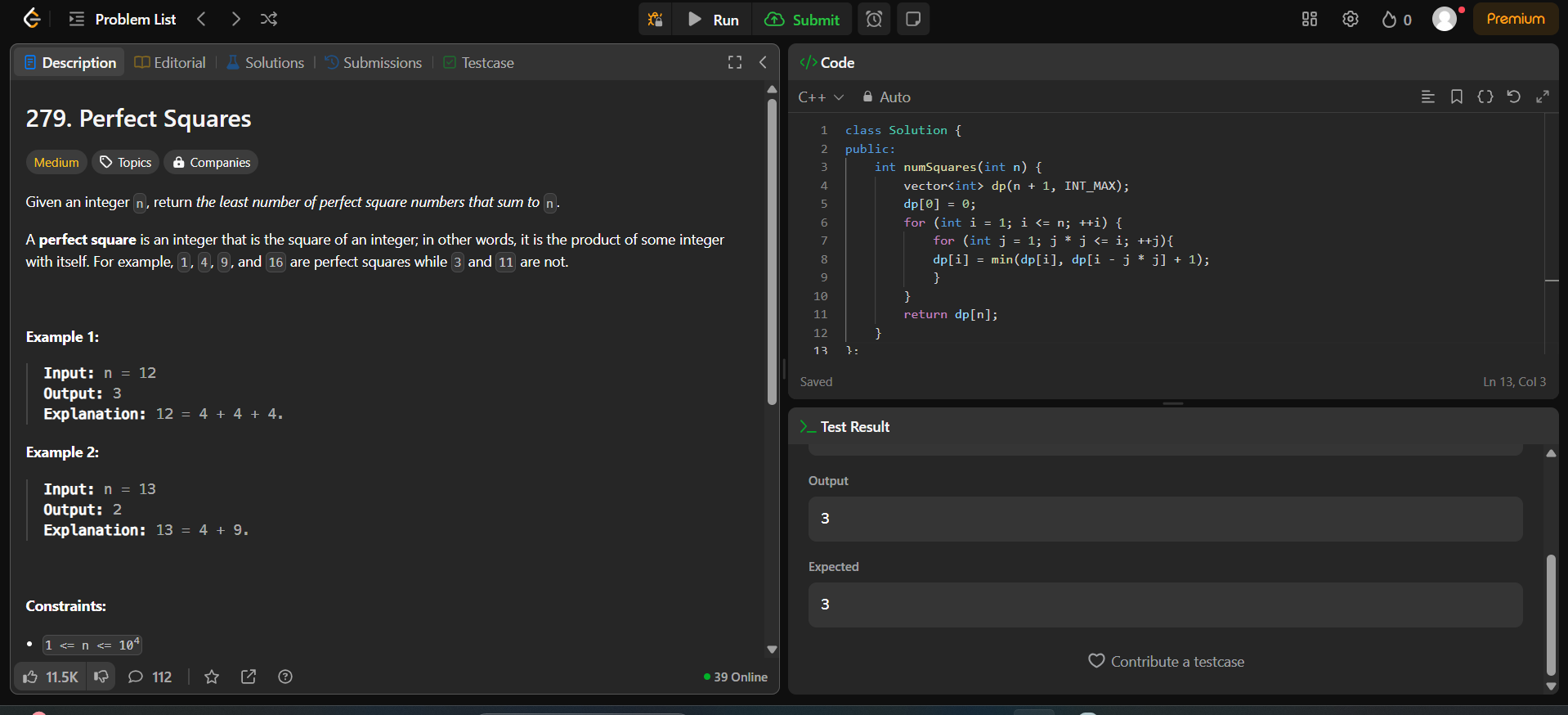
}

return dp[n];

}

};

**OUTPUT:**

****