

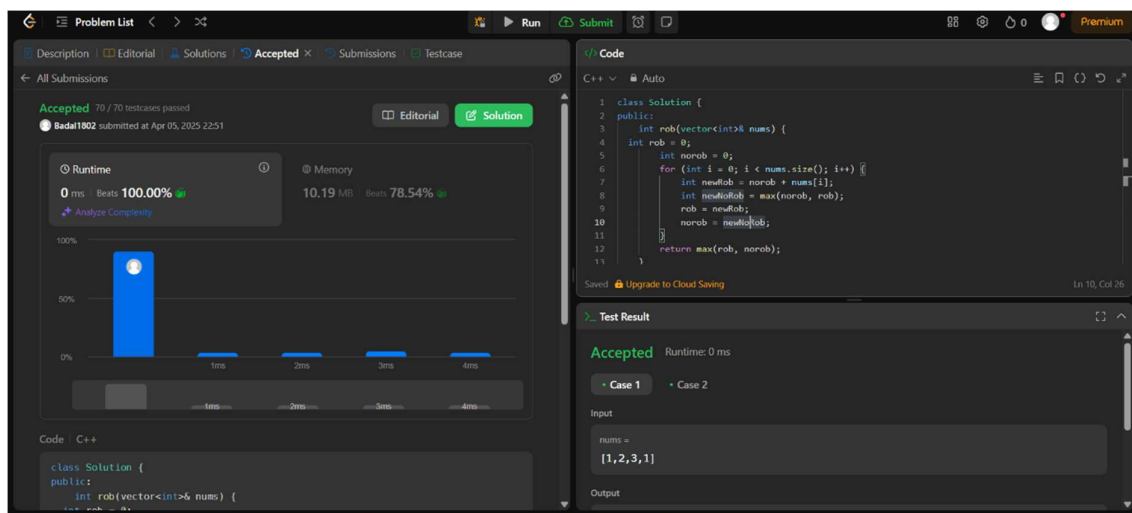
Experiment-7

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:

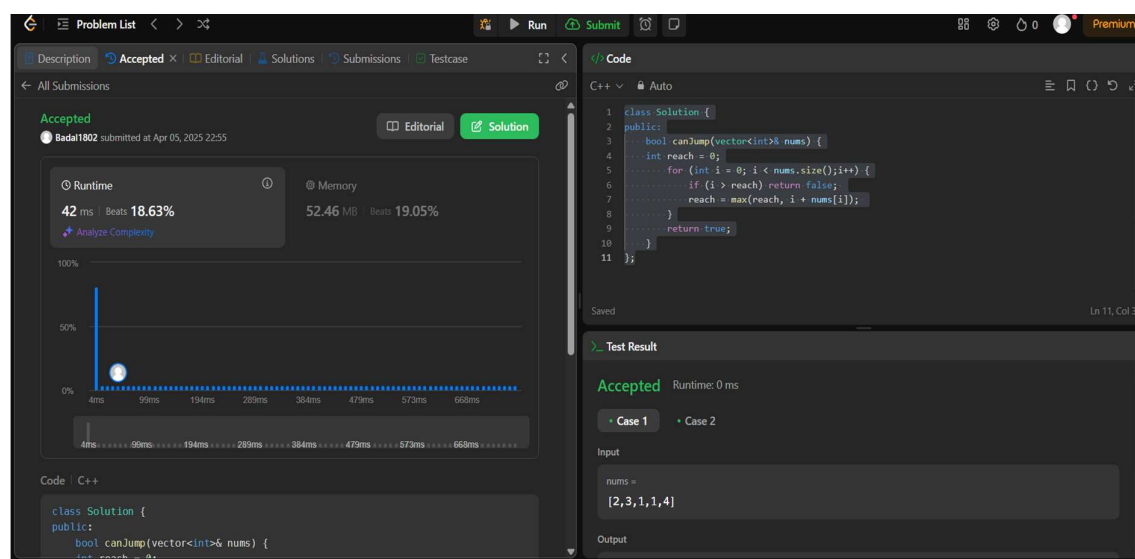


Jump Game

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:

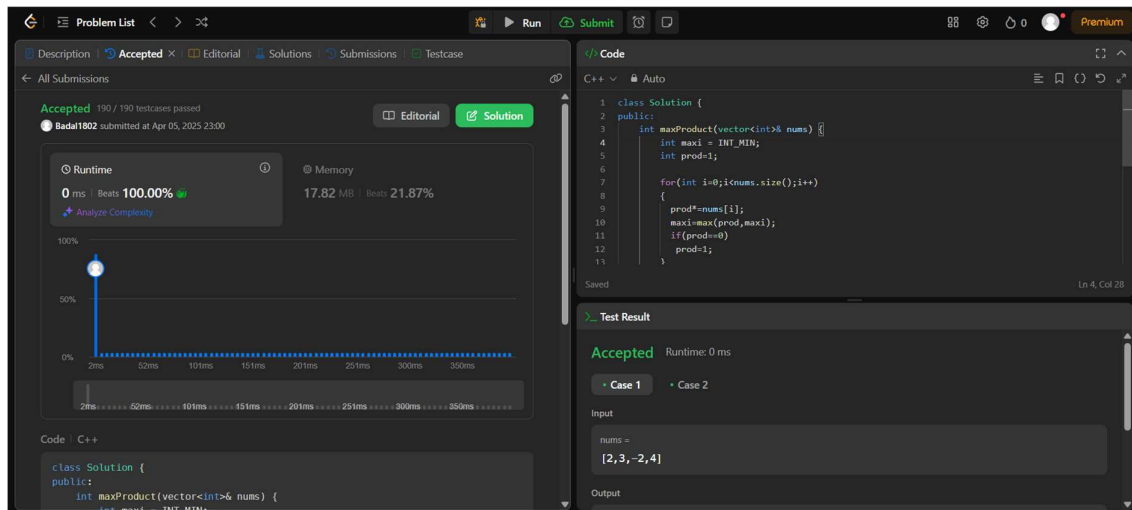


Maximum Product Subarray

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:

The screenshot shows the LeetCode interface for problem 279, "Perfect Squares". The problem description states: "Given an integer n , return the least number of perfect square numbers that sum to n ." It defines a perfect square as an integer that is the square of another integer. Examples provided are: for $n=12$, the output is 3 (since $12 = 4 + 4 + 4$); for $n=13$, the output is 2 (since $13 = 4 + 9$). The constraints are $1 \leq n \leq 10^4$.

The solution is implemented in C++ using a dynamic programming approach. The code defines a class `Solution` with a public method `numSquares` that takes an integer `n` and returns the minimum number of perfect squares that sum to `n`. The implementation uses a vector `dp` of size `n + 1` initialized with `INT_MAX`, and `dp[0]` is set to 0. A nested loop iterates over `i` from 1 to `n`, and for each `i`, it iterates over `j` from 1 to `i`. The value of `dp[i]` is updated to be the minimum of its current value and `dp[i - j * j] + 1`.

The test results section shows a single test case where the output is 3, which matches the expected result.

```
1 class Solution {
2 public:
3     int numSquares(int n) {
4         vector<int> dp(n + 1, INT_MAX);
5         dp[0] = 0;
6         for (int i = 1; i <= n; ++i) {
7             for (int j = 1; j * j <= i; ++j) {
8                 dp[i] = min(dp[i], dp[i - j * j] + 1);
9             }
10        }
11        return dp[n];
12    }
13 }
```

Output: 3
Expected: 3