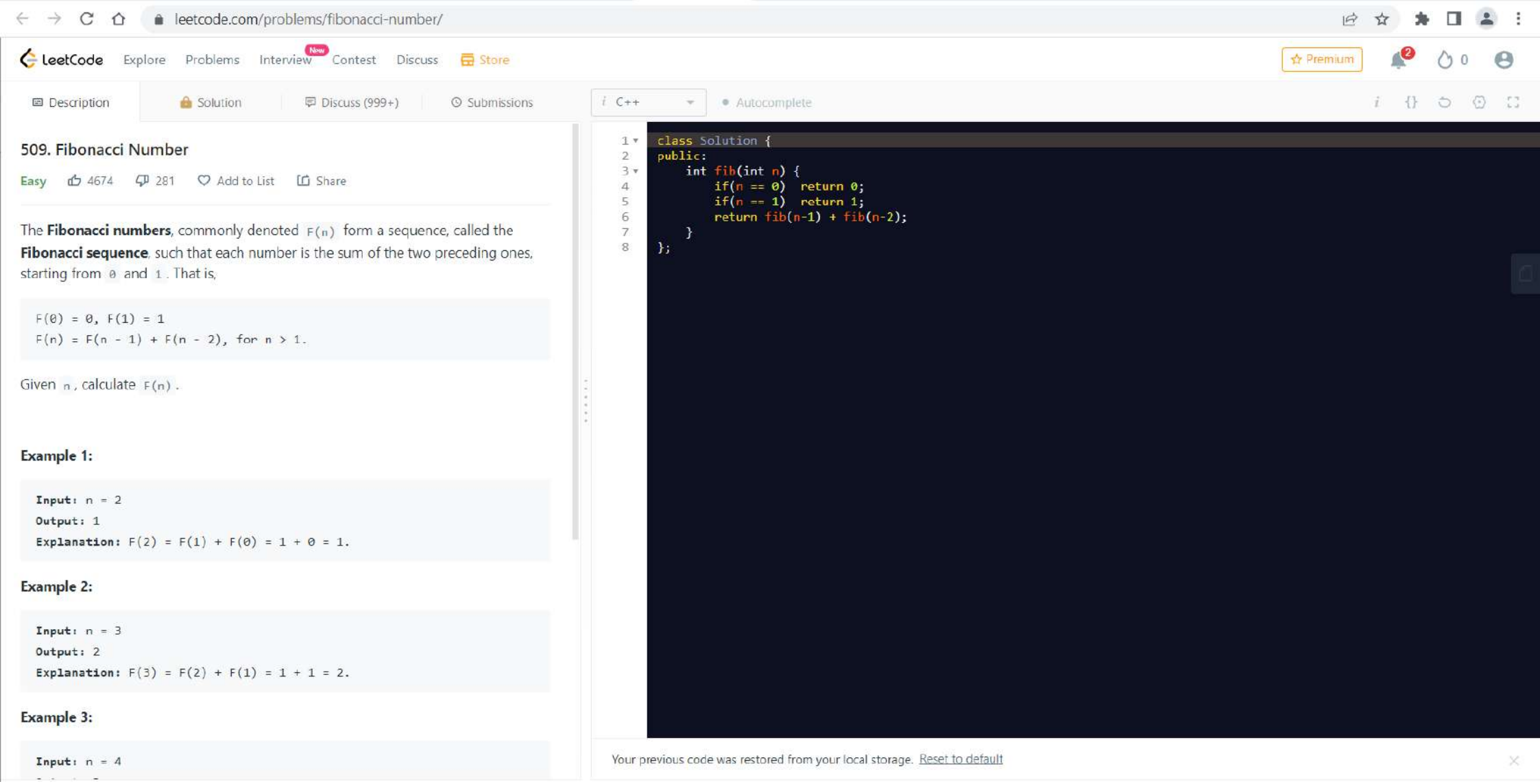
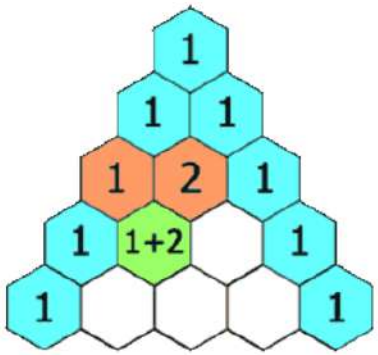

leetcode.com/problems/fibonacci-number/
LeetCode
Explore
Problems
Interview
New
Contest
Discuss
Store
Premium
Description
Solution
Discuss (999+)
Submissions
C++
Autocomplete
509. Fibonacci Number
Easy
4674
281
Add to List
Share
The Fibonacci numbers, commonly denoted F(n) form a sequence, called the Fibonacci sequence, such that each number is the sum of the two preceding ones, starting from 0 and 1. That is,
F(0) = 0, F(1) = 1
F(n) = F(n - 1) + F(n - 2), for n > 1.
Given n, calculate F(n).
Example 1:
Input: n = 2
Output: 1
Explanation: F(2) = F(1) + F(0) = 1 + 0 = 1.
Example 2:
Input: n = 3
Output: 2
Explanation: F(3) = F(2) + F(1) = 1 + 1 = 2.
Example 3:
Input: n = 4
class Solution {
public:
int fib(int n) {
if(n == 0) return 0;
if(n == 1) return 1;
return fib(n-1) + fib(n-2);
}
};
Your previous code was restored from your local storage.
Reset to default

118. Pascal's Triangle

Easy 7240 237 Add to List Share

Given an integer `numRows`, return the first numRows of **Pascal's triangle**.

In **Pascal's triangle**, each number is the sum of the two numbers directly above it as shown:



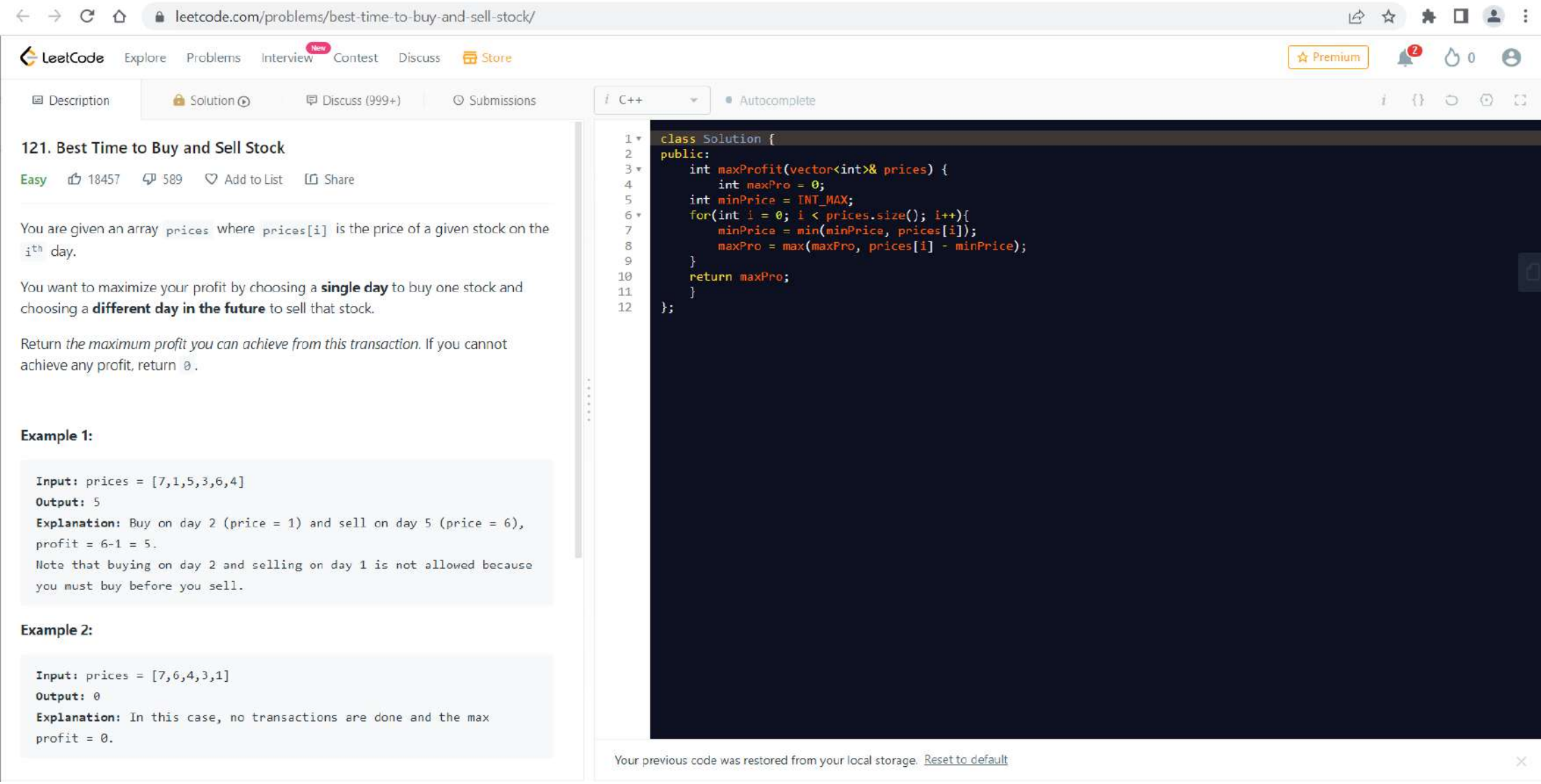
Example 1:

```
Input: numRows = 5
Output: [[1],[1,1],[1,2,1],[1,3,3,1],[1,4,6,4,1]]
```

Example 2:

```
Input: numRows = 1
Output: [[1]]
```

```cpp
class Solution {
public:
    vector<vector<int>> generate(int numRows) {
        vector<vector<int>> r(numRows);
        for (int i = 0; i < numRows; i++) {
            r[i].resize(i + 1);
            r[i][0] = r[i][i] = 1;
            for (int j = 1; j < i; j++)
                r[i][j] = r[i - 1][j - 1] + r[i - 1][j];
        }
        return r;
    }
};
```


leetcode.com/problems/best-time-to-buy-and-sell-stock/
LeetCode
Explore
Problems
Interview
New
Contest
Discuss
Store
Premium
Description
Solution
Discuss (999+)
Submissions
C++
Autocomplete
121. Best Time to Buy and Sell Stock
Easy
18457
589
Add to List
Share
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.
Example 1:
Input: prices = [7,1,5,3,6,4]
Output: 5
Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.
Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.
Example 2:
Input: prices = [7,6,4,3,1]
Output: 0
Explanation: In this case, no transactions are done and the max profit = 0.
class Solution {
public:
int maxProfit(vector<int>& prices) {
int maxPro = 0;
int minPrice = INT_MAX;
for(int i = 0; i < prices.size(); i++){
minPrice = min(minPrice, prices[i]);
maxPro = max(maxPro, prices[i] - minPrice);
}
return maxPro;
}
};
Your previous code was restored from your local storage.
Reset to default

1137. N-th Tribonacci Number

Easy 2063 118 Add to List Share

The Tribonacci sequence T_n is defined as follows:

$T_0 = 0$, $T_1 = 1$, $T_2 = 1$, and $T_{n+3} = T_n + T_{n+1} + T_{n+2}$ for $n >= 0$.

Given `n`, return the value of T_n.

Example 1:

```
Input: n = 4
Output: 4
Explanation:
T_3 = 0 + 1 + 1 = 2
T_4 = 1 + 1 + 2 = 4
```

Example 2:

```
Input: n = 25
Output: 1389537
```

Constraints:

- `0 <= n <= 37`
- The answer is guaranteed to fit within a 32-bit integer, ie. `answer <= 2^31 - 1`.

```cpp
class Solution {
public:
    int tribonacci(int n) {
         if (n < 2) return n;
        int a = 0, b = 1, c = 1, d = a + b + c;
        while (n-- > 2) {
            d = a + b + c, a = b, b = c, c = d;
        }
        return c;
    }
};
```

leetcode.com/problems/min-cost-climbing-stairs/

LeetCode Explore Problems Interview (New) Contest Discuss Store ☆ Premium 2 0

Description Solution Discuss (999+) Submissions C++ Autocomplete

746. Min Cost Climbing Stairs

Easy 7365 1193 Add to List Share

You are given an integer array `cost` where `cost[i]` is the cost of i^{th} step on a staircase. Once you pay the cost, you can either climb one or two steps.

You can either start from the step with index `0`, or the step with index `1`.

Return *the minimum cost to reach the top of the floor.*

Example 1:

```
Input: cost = [10,15,20]
Output: 15
Explanation: You will start at index 1.
- Pay 15 and climb two steps to reach the top.
The total cost is 15.
```

Example 2:

```
Input: cost = [1,100,1,1,1,100,1,1,100,1]
Output: 6
Explanation: You will start at index 0.
- Pay 1 and climb two steps to reach index 2.
- Pay 1 and climb two steps to reach index 4.
- Pay 1 and climb two steps to reach index 6.
- Pay 1 and climb one step to reach index 7.
- Pay 1 and climb two steps to reach index 9.
- Pay 1 and climb one step to reach the top.
```

```cpp
class Solution {
public:
    int minCostClimbingStairs(vector<int>& cost) {
         int n = cost.size();
        vector<int> dp(n + 1);
        for (int i = 2; i <= n; i++) {
            int jumpOneStep = dp[i - 1] + cost[i - 1];
            int jumpTwoStep = dp[i - 2] + cost[i - 2];
            dp[i] = min(jumpOneStep, jumpTwoStep);
        }
        return dp[n];
    }
};
```

Your previous code was restored from your local storage. Reset to default