



Experiment3.3

Student Name: Aman Gokul

Branch: BE-CSE

Semester: 6th

Subject Name: Competitive Coding-II

UID: 20BCS5449

Section/Group: 607/A

Date of Performance: 03/05/2023

Subject Code: 20CSP-351

1. Aim:

To implement the concept of greedy algorithm approaches.

2. Objective:

- The objective is to build problem solving capability and to learn the basic concepts of greedy algorithm approaches in data structures.
- The implementation of Candy which shows and brushes up the concept of greedy algorithm and various other concepts as well as algorithms of data structures. □ The implementation of Can place flowers problem on LeetCode.

3. LeetCode code and output:

- **PROBLEM**
- **CLIMBING STAIRS**

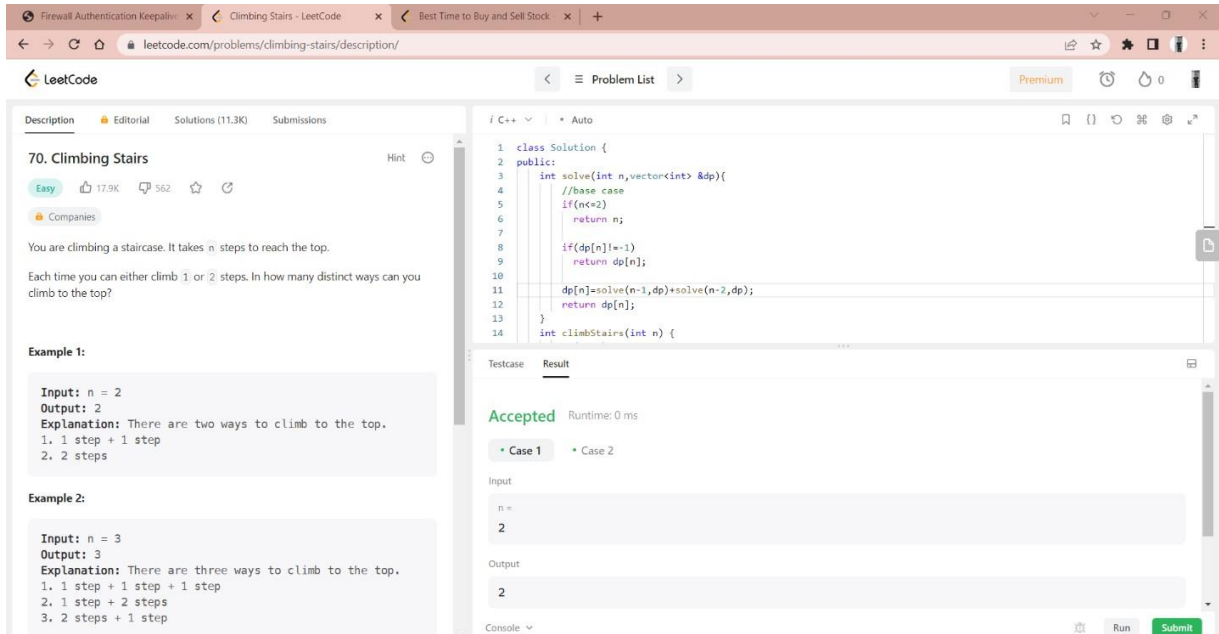
```
class Solution { public:
    int solve(int n,vector<int> &dp){
        //base case
        if(n<=2)
            return n;

        if(dp[n]!=-1)
            return dp[n];

        dp[n]=solve(n-
1,dp)+solve(n-2,dp);        return dp[n];
    }
    int climbStairs(int n) {
        if(n<=2)            return n;
        vector<int>         dp(n+1);
        for(int             i=0;i<=n;i++)
            dp[i]=-1;
    }
};
```

```

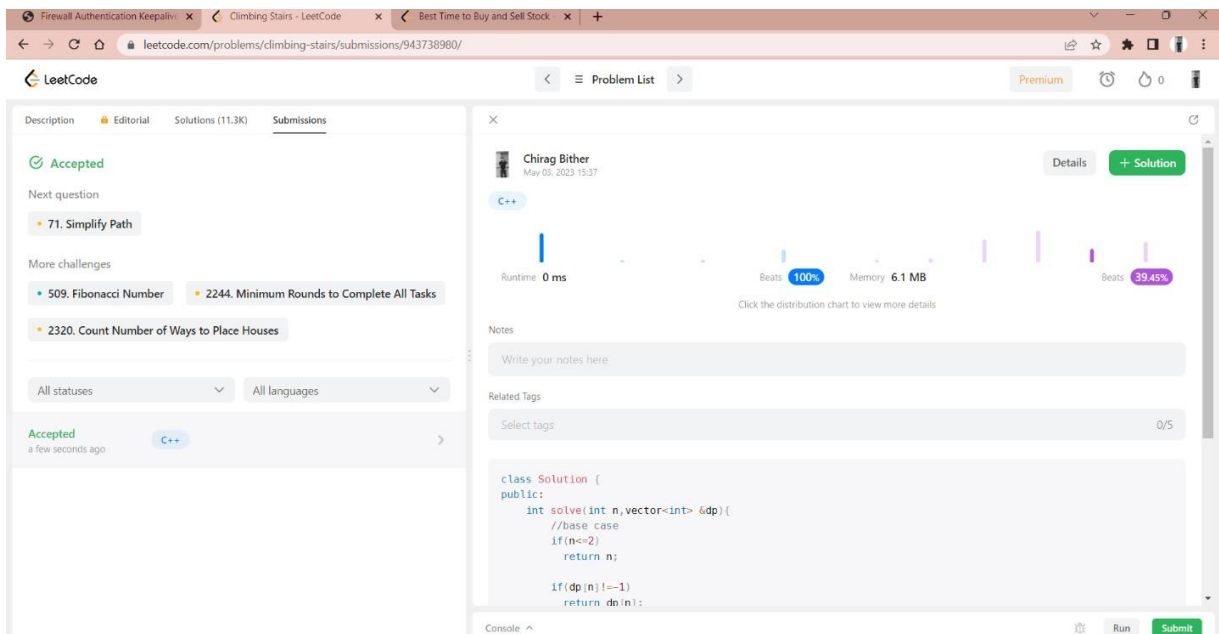
        return
    solve(n,dp);
}
};
    
```



The screenshot shows the LeetCode interface for problem 70, "Climbing Stairs". The problem description states: "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. The code editor shows a C++ solution using dynamic programming. The test case results show "Accepted" with a runtime of 0 ms.

```

1 class Solution {
2 public:
3     int solve(int n, vector<int> &dp){
4         //base case
5         if(n<=2)
6             return n;
7
8         if(dp[n]!=-1)
9             return dp[n];
10
11        dp[n]=solve(n-1,dp)+solve(n-2,dp);
12        return dp[n];
13    }
14    int climbStairs(int n) {
    
```



The screenshot shows the LeetCode submission page for problem 70. The submission status is "Accepted". The next question is "71. Simplify Path". More challenges are listed: "509. Fibonacci Number", "2244. Minimum Rounds to Complete All Tasks", and "2320. Count Number of Ways to Place Houses". The submission details show the user "Chirag Bithar" with a runtime of 0 ms, memory of 6.1 MB, and a beats percentage of 100%. The code editor shows the same C++ solution as in the previous screenshot.

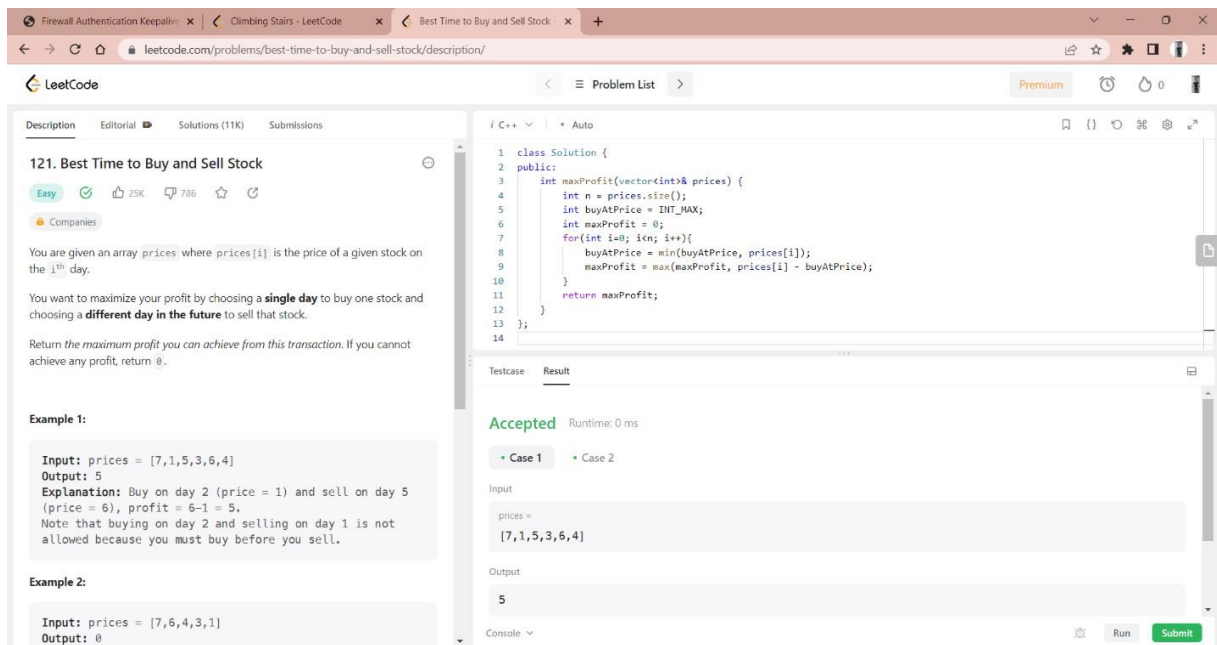
```

class Solution {
public:
    int solve(int n, vector<int> &dp){
        //base case
        if(n<=2)
            return n;

        if(dp[n]!=-1)
            return dp[n];
    
```

- **PROBLEM**
- **BEST TIME TO BUY AND SELL STOCK**

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



The screenshot shows the LeetCode interface for problem 121, "Best Time to Buy and Sell Stock". The problem description states: "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 shows input [7,1,5,3,6,4] and output 5, with an explanation that buying on day 2 (price 1) and selling on day 5 (price 6) yields a profit of 5. Example 2 shows input [7,6,4,3,1] and output 0. The code editor on the right shows the C++ solution:

```
1 class Solution {
2 public:
3     int maxProfit(vector<int>& prices) {
4         int n = prices.size();
5         int buyAtPrice = INT_MAX;
6         int maxProfit = 0;
7         for(int i=0; i<n; i++){
8             buyAtPrice = min(buyAtPrice, prices[i]);
9             maxProfit = max(maxProfit, prices[i] - buyAtPrice);
10        }
11        return maxProfit;
12    }
13 };
14
```

 The test case section shows "Accepted" status with a runtime of 0 ms. The input is [7,1,5,3,6,4] and the output is 5. The "Submit" button is visible at the bottom right.



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Firewall Authentication Keepalive x Climbing Stairs - LeetCode x Best Time to Buy and Sell Stock x +

leetcode.com/problems/best-time-to-buy-and-sell-stock/submissions/943739188/

LeetCode Problem List Premium

Description Editorial Solutions (11K) Submissions

Accepted

Next question

296. Best Meeting Point

More challenges

122. Best Time to Buy and Sell Stock II

123. Best Time to Buy and Sell Stock III

188. Best Time to Buy and Sell Stock IV

All statuses All languages

Accepted a few seconds ago C++

Accepted Jan 15, 2023 C++

Accepted Jan 13, 2023 C++

Chirag Bither May 03, 2023 15:37 Details + Solution

C++

Runtime 153 ms Beats 14.48% Memory 93.3 MB Beats 55.31%

Click the distribution chart to view more details

Notes

Write your notes here

Related Tags

Select tags 0/5

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

Console Run Submit