

Experiment – 3.3

Student Name: Avinash Jena

UID: 20BCS2690

Branch: CSE

Section/Group: 707/B

Subject Name: Competitive Coding II

Subject Code: 20CSP-351

Aim: Best Time to Buy and Sell Stock

Objective:

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.

Constraints:

$1 \leq \text{prices.length} \leq 105$

$0 \leq \text{prices}[i] \leq 104$

Code:

```
class Solution {  
    public int maxProfit(int[] prices) {  
        int lsf = Integer.MAX_VALUE;  
        int op = 0;  
        int pist = 0;  
  
        for(int i = 0; i < prices.length; i++){  
            if(prices[i] < lsf){  
                lsf = prices[i];  
            }  
            pist = prices[i] - lsf;  
            if(op < pist){  
                op = pist;  
            }  
        }  
        return op;  
    }  
}
```

Output:

```
i Java ▾ | • Auto

1  class Solution {
2      public int maxProfit(int[] prices) {
3          int lsf = Integer.MAX_VALUE;
4          int op = 0;
5          int pist = 0;
6
7          for(int i = 0; i < prices.length; i++){
8              if(prices[i] < lsf){
9                  lsf = prices[i];
10             }
11             pist = prices[i] - lsf;
12             if(op < pist){
13                 op = pist;
14             }
15         }
16         return op;
17     }
18 }
```

Testcase	Result
Accepted Runtime: 0 ms	
• Case 1 • Case 2	
Input	
prices = [7,1,5,3,6,4]	
Output	
5	
Expected	
5	

Aim: Climbing Stairs

Objective:

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

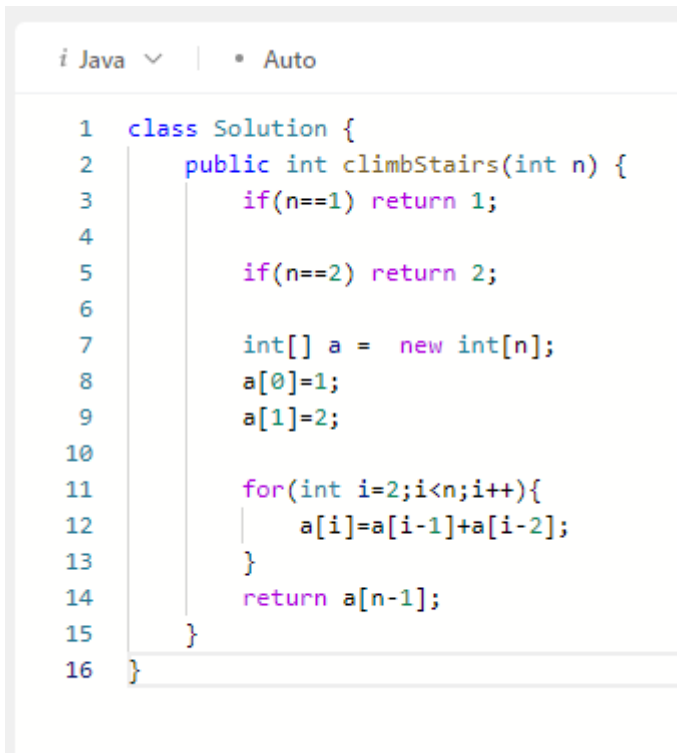
Constraints:

$1 \leq n \leq 45$

Code:

```
class Solution {  
    public int climbStairs(int n) {  
        if(n==1) return 1;  
  
        if(n==2) return 2;  
  
        int[] a = new int[n];  
        a[0]=1;  
        a[1]=2;  
  
        for(int i=2;i<n;i++){  
            a[i]=a[i-1]+a[i-2];  
        }  
        return a[n-1];  
    }  
}
```

Output:



```
i Java | • Auto  
  
1  class Solution {  
2      public int climbStairs(int n) {  
3          if(n==1) return 1;  
4  
5          if(n==2) return 2;  
6  
7          int[] a = new int[n];  
8          a[0]=1;  
9          a[1]=2;  
10  
11         for(int i=2;i<n;i++){  
12             a[i]=a[i-1]+a[i-2];  
13         }  
14         return a[n-1];  
15     }  
16 }
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Testcase

Result

Accepted Runtime: 0 ms

• Case 1

• Case 2

Input

n =
3

Output

3

Expected

3