Experiment 6 A

Student Name: PARDEEP SINGH **UID: 22BCS16692**

Branch: CSE **Section/Group: Ntpp 602-A**

Semester: 6TH **Date of Performance:20/02/25**

Subject Name: AP Lab-2 **Subject Code: 22CSH-352**

1. TITLE:

Climbing Stairs

2. AIM:

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?

3. Algorithm

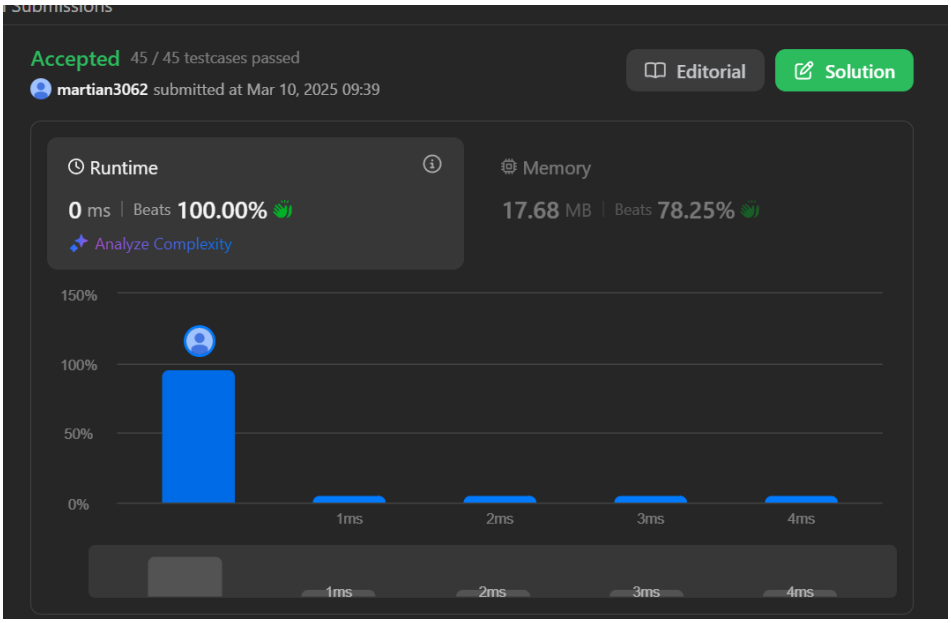
- If n is 1, return 1 (base case).
- Initialize two variables: prev (previous step count) as 1 and prev2 (second previous step count) as 0.
- Loop through the range from 1 to n, calculating the current step count as curi = prev + prev2, and update prev and prev2.
- Return prev as the total number of ways to climb the stairs.

Implemetation/Code

```
class Solution:
    def climbStairs(self, n: int) -> int:
        if n == 1:
            return 1
        prev = 1
        prev2 = 0
```

```
for i in range(1, n + 1):
    curi = prev + prev2
    prev2 = prev
    prev = curi
return prev
```

Output



Time Complexity : O(n)

Space Complexity : O(1)

Learning Outcomes:-

- Understand how dynamic programming can optimize the calculation of stair climbing problems.
- Learn how to reduce space complexity by storing only the last two computed values.

Experiment 6 B

Student Name: PARDEEP SINGH **UID: 22BCS16692**

Branch: CSE **Section/Group: Ntpp 602-A**

Semester: 6TH **Date of Performance:20/02/25**

Subject Name: AP Lab-2 **Subject Code: 22CSH-352**

1. TITLE:

Best time to buy and Sell Stock

2. AIM:

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.

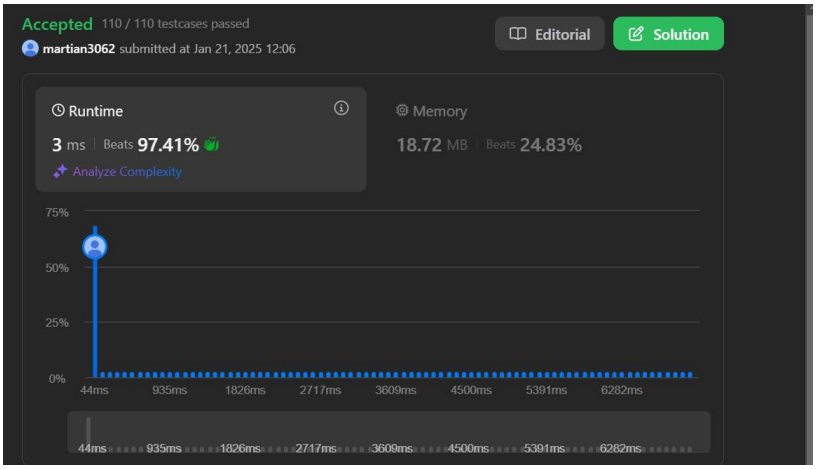
3. Algorithm

1. Initialize l = 0 (left pointer) and maxP = 0 (to store maximum profit).
2. Iterate through the prices list with the right pointer r from index 1.
3. If prices[r] > prices[l], calculate the profit and update maxP if it's higher than the current max.
4. If prices[r] <= prices[l], update l to the current index r to start a new potential profit calculation.

Implemetation/Code:

```
class Solution:
    def maxProfit(self, prices: List[int]) -> int:
        l = 0
        maxP = 0
        for r in range(1, len(prices)):
            if prices[l] < prices[r]:
                prof = prices[r] - prices[l]
                maxP = max(maxP, prof)
            else:
                l = r
        return maxP
```

Output



Time Complexity : O(n)

Space Complexity : O(1)

Learning Outcomes:-

- Concept of the "two-pointer" technique to find the maximum profit in a stock buy-sell problem.
- Learn how to update state variables efficiently within a loop to track the highest profit.