DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

CU
CHANDIGARH UNIVERSITY

WORKSHEET- 6

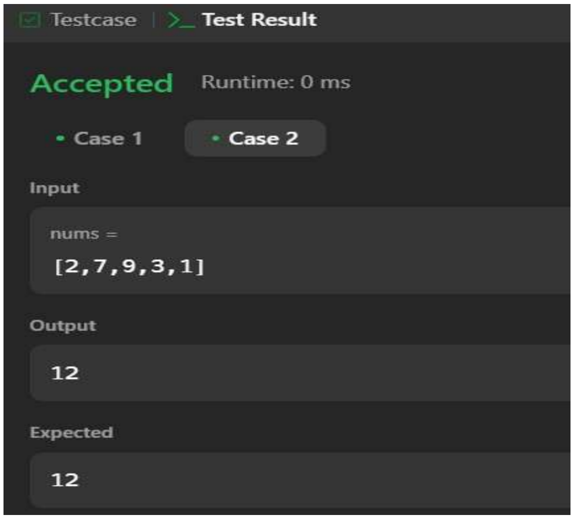
Student Name: Reeva
UID: 22BCS16744
Branch: BE-CSE
Section/Group: 22BCS_NTPP-
Semester: 6th
Date of Performance: 20/02/2025
Subject Name: AP LAB - II
Subject Code: 22CSP-351

1. **Aim:** Given an integer array nums representing the amount of money of each house, return the maximum amount of money you can rob tonight without alerting the police1

2. **Source Code:**

```
def rob(self, nums: List[int]) ->int:       if not nums:

    return 0

if len(nums) == 1:

    return nums[0]



      prev, curr = 0, 0

      for num in nums:

         prev, curr = curr, max(curr, prev + num)



      return curr
```

3. Screenshots of outputs:



Accepted Runtime: 0 ms

Input

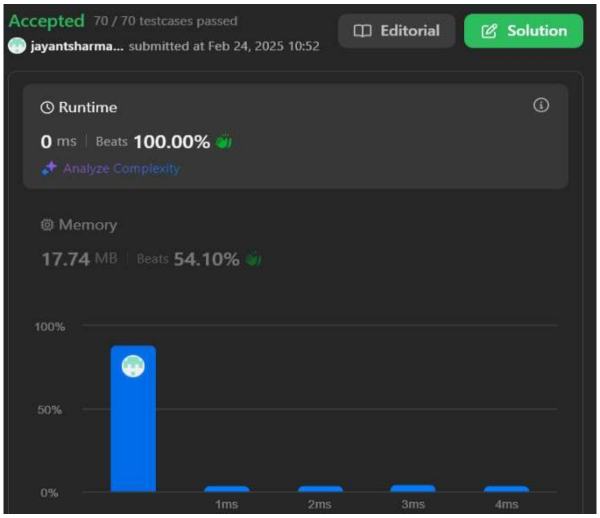
```
nums =
[2,7,9,3,1]
```

Output

```
12
```

Expected

```
12
```



Accepted 70 / 70 testcases passed

jayantsharma... submitted at Feb 24, 2025 10:52

Runtime

0 ms | Beats 100.00%

Memory

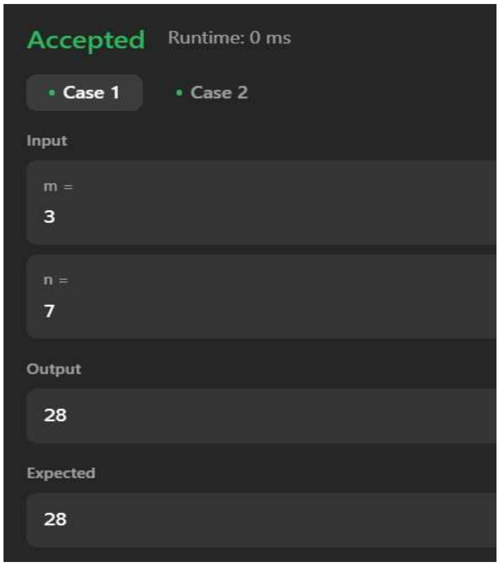
17.74 MB | Beats 54.10%

2. **Aim:** Given the two integers m and n, return the number of possible unique paths that the robot can take to reach the bottom-right corner.

Source Code

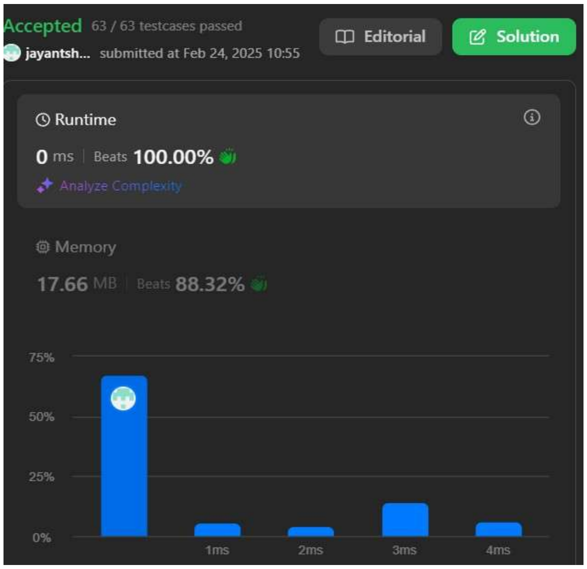
```
class Solution:
  def uniquePaths(self, m: int, n: int) -> int:
      dp = [[1] * n for _ in range(m)]

      for i in range(1, m):
          for j in range(1, n):
              dp[i][j] = dp[i - 1][j] + dp[i][j - 1]

      return dp[-1][-1]
```

Screenshots of outputs:



Accepted Runtime: 0 ms

```
Input
m =
3
n =
7
Output
28
Expected
28
```



Accepted 63 / 63 testcases passed

jayantsh... submitted at Feb 24, 2025 10:55

Runtime
0 ms | Beats 100.00%

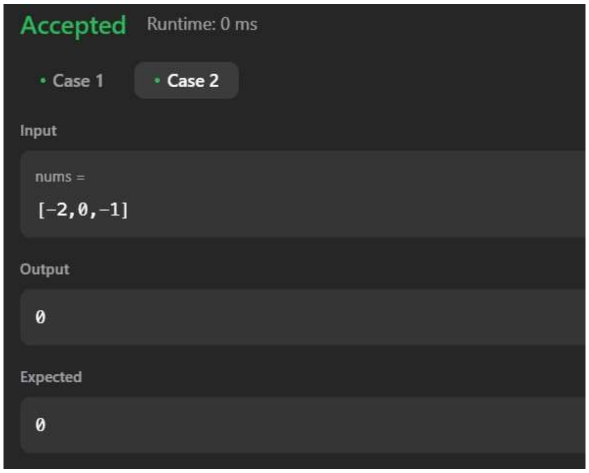
Memory
17.66 MB | Beats 88.32%

3. Aim: Given an integer array nums, find a subarray that has the largest product, and return the product.

Source Code:

```
 class Solution:
  def maxProduct(self, nums: List[int]) ->
 int:
     max_prod = min_prod = result =
 nums[0]

        for num in nums[1:]:
            temp_max = max(num, max_prod * num, min_prod * num)
            min_prod = min(num, max_prod * num, min_prod * num)
            max_prod = temp_max
            result = max(result, max_prod)

        return result
```

Screenshots of outputs:



Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

nums =

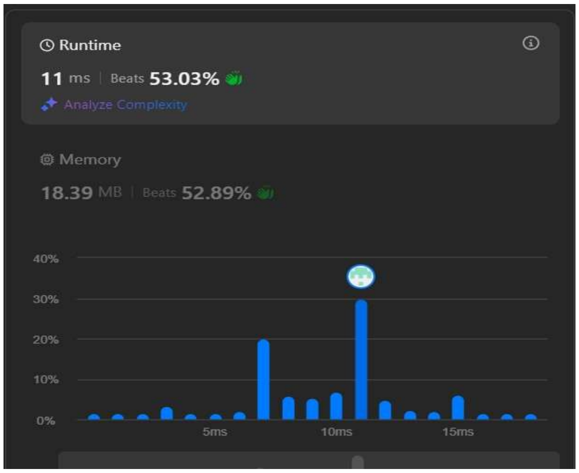
```
[-2,0,-1]
```

Output

```
0
```

Expected

```
0
```



Runtime

11 ms | Beats 53.03%

Analyze Complexity

Memory

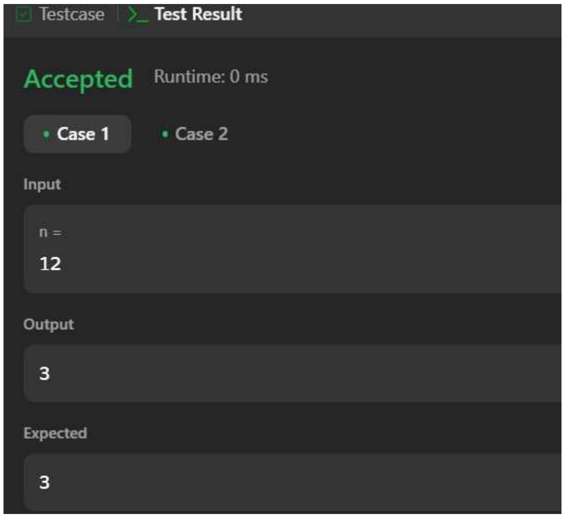
18.39 MB | Beats 52.89%

4. **Aim:** Given an integer n, return *the least number of perfect square numbers that sum to* n.

Source Code:

```
class Solution:
   def numSquares(self, n: int) -> int:
      dp = [float('inf')] * (n + 1)
        dp[0] = 0

      for i in range(1, n + 1):
     for j in range(1, int(math.sqrt(i)) + 1):
          dp[i] = min(dp[i], dp[i - j * j] + 1)

      return dp[n]
```

4. Screenshots of outputs:



Accepted Runtime: 0 ms

Input

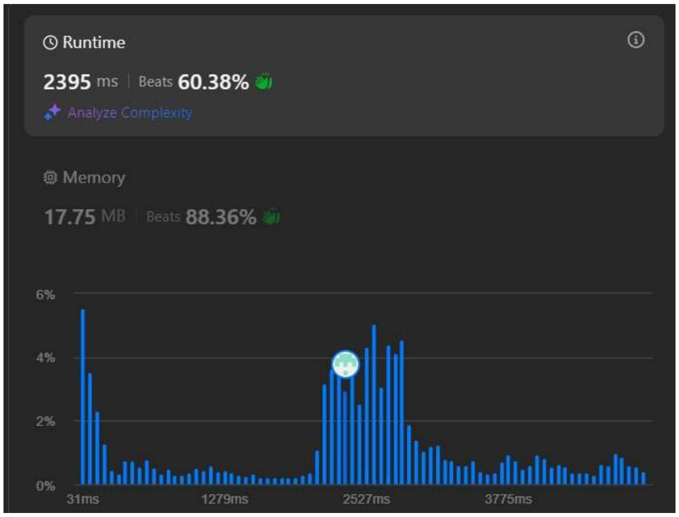
n =
12

Output

3

Expected

3



Runtime

2395 ms | Beats 60.38%

Memory

17.75 MB | Beats 88.36%

Learning Outcomes:

- Binary Tree Construction:* Efficiently reconstruct a binary tree from inorder and postorder traversals using recursion and hash maps.
- Dynamic Programming (DP) Concepts:* Applying DP to solve problems like the minimum number of perfect squares summing to n, unique paths in a grid, and house robber problem.
- State Transition Optimization:* Utilizing DP and variable swapping to optimize space complexity, as seen in rob() and maxProduct().
- Mathematical Approaches:* Understanding how mathematical properties like square numbers and factorial-based paths contribute to problem-solving.
- Algorithmic Thinking:* Developing problem-solving skills with recursive tree construction, iterative DP, and greedy strategies for maximizing results.