Experiment-7

Student Name: Manish Lalwani **UID:** 22BCS16288
Branch: BE-CSE **Section/Group:** KPIT-901/B
Semester: 6th **Date of Performance:** 13/03/25
Subject Name: Advanced Programming Lab - 2 **Subject Code:** 22CSP-351

1. **Aim:** Greedy algorithm.
   1. Problem: 53. Maximum Subarray.
   2. Problem: 70. Climbing Stairs.

2. **Objective:**
   1. Find the contiguous subarray with the largest sum using Kadane's Algorithm in O(n) time complexity.
   2. Find the number of distinct ways to climb a staircase with n steps, where each step can be 1 or 2 at a time.

3. **Implementation/Code:**

1.)

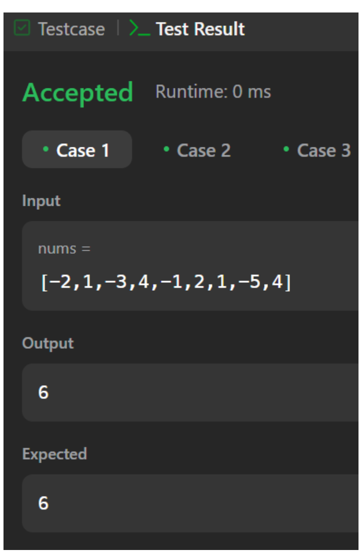
```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int maxSum = nums[0], currSum = 0;

        for (int num : nums) {
            currSum = max(num, currSum + num);
            maxSum = max(maxSum, currSum);
        }

        return maxSum;
    }
};
```

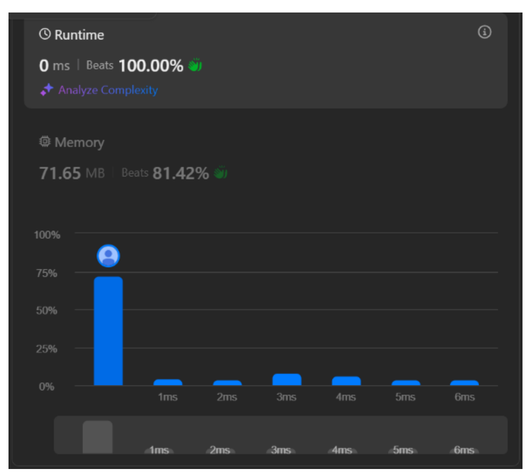
2.)

```
class Solution {
public:
    int climbStairs(int n) {
        if (n <= 2) return n;

        int prev1 = 1, prev2 = 2, curr;
        for (int i = 3; i <= n; i++) {
            curr = prev1 + prev2;
            prev1 = prev2;
            prev2 = curr;
        }

        return prev2;
    }
};
```

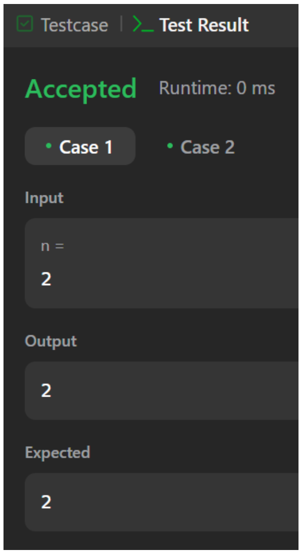
4. Output:

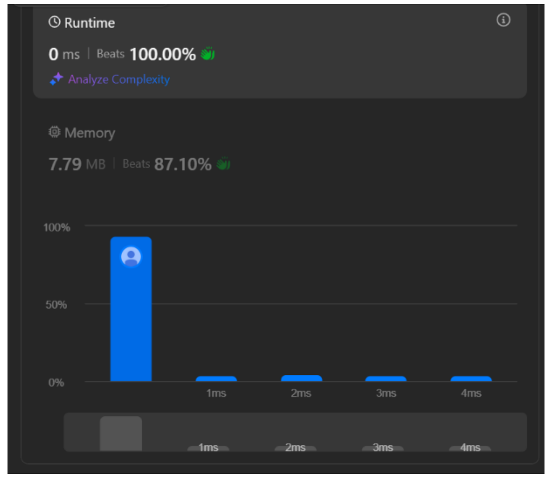
1.





2.





5. Time Complexity:

1. O(n)
2. O (n)

6. Space Complexity:

1. O (1)
2. O(1)

7. Learning Outcome:

1. Kadane's Algorithm efficiently finds the maximum subarray sum in linear time.
2. It follows a greedy approach, maintaining a running sum and updating the maximum sum encountered.
3. Helps understand dynamic programming principles, particularly prefix sums and state transitions.Learn how to optimize space complexity (O (1) in Jump Game, O(n) in Coin Change) while maintaining efficient time complexity.
4. The problem follows the Fibonacci sequence pattern: dp[i] = dp[i-1] + dp[i-2].

5. Dynamic Programming (DP) Optimization: We reduce space complexity by using only two variables instead of an array.
6. The approach efficiently computes results in O(n) time with constant space, making it scalable.