EXPERIMENT 7

NAME: Aditya Kumar Bharti **UID: 22BET10276**

Branch: BE – IT **SECTION/GROUP: 22BET_IOT – 703 (B)**

SEMESTER: 6th **SUBJECT CODE: 22ITP – 351**

Problem 1

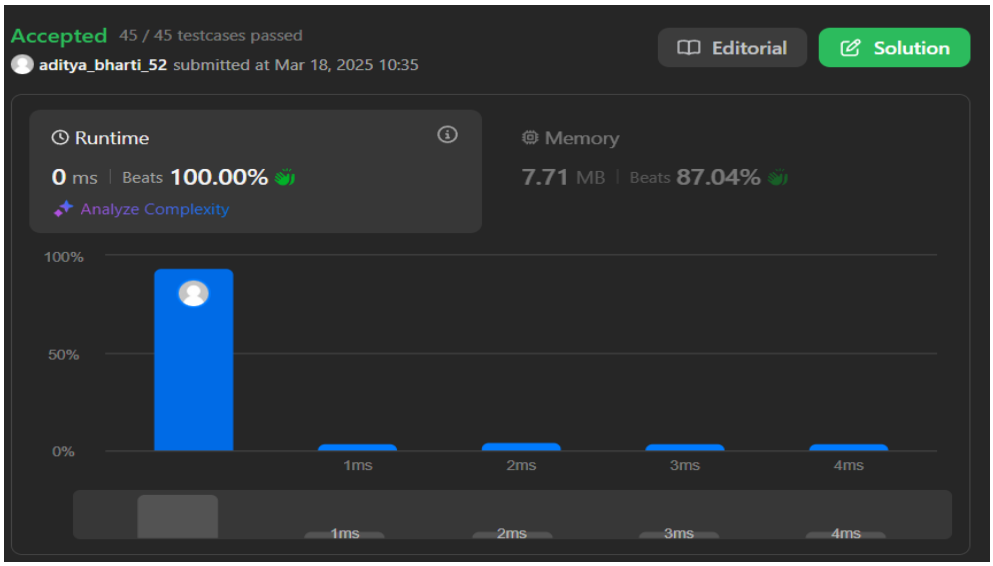
AIM: Climbing Stairs

CODE:

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

        int first = 1, second = 2, current;
        for (int i = 3; i <= n; ++i) {
            current = first + second;
            first = second;
            second = current;
        }
        return current;
    }
};
```

OUTPUT:



EXPERIMENT 7

Problem 2

AIM: Maximum Subarray

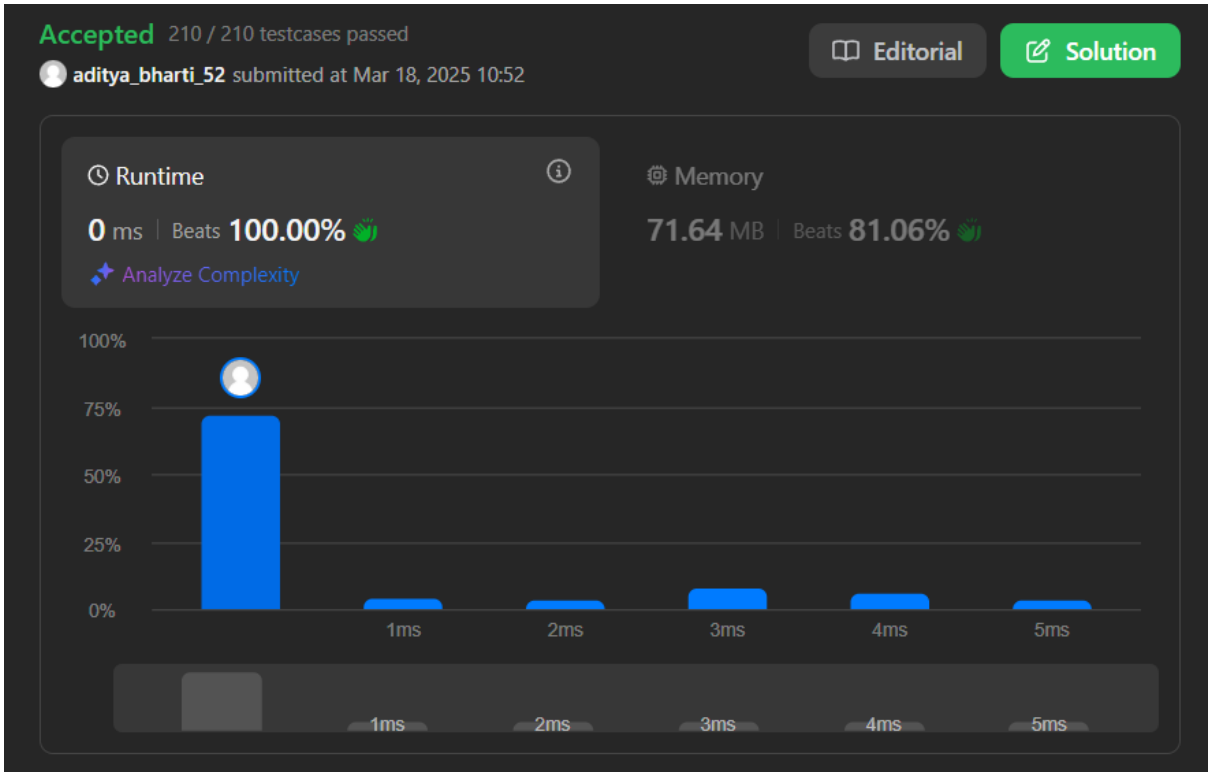
CODE:

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

        for (int i = 1; i < nums.size(); ++i) {
            currentSum = max(nums[i], currentSum + nums[i]);
            maxSum = max(maxSum, currentSum);
        }

        return maxSum;
    }
};
```

OUTPUT:



EXPERIMENT 7

Problem 3

AIM: House Robber

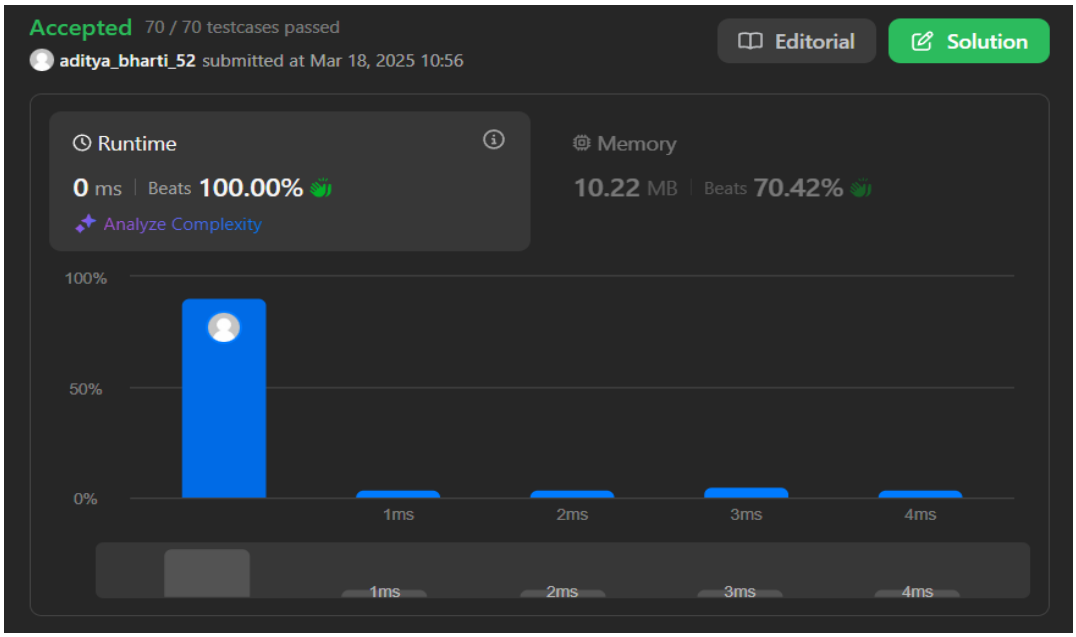
CODE:

```
class Solution {
public:
    int rob(vector<int>& nums) {
        int n = nums.size();
        if (n == 0) return 0;
        if (n == 1) return nums[0];

        int prev1 = 0, prev2 = 0;
        for (int num : nums) {
            int temp = max(prev1, prev2 + num);
            prev2 = prev1;
            prev1 = temp;
        }

        return prev1;
    }
};
```

OUTPUT:



EXPERIMENT 7

Problem 4

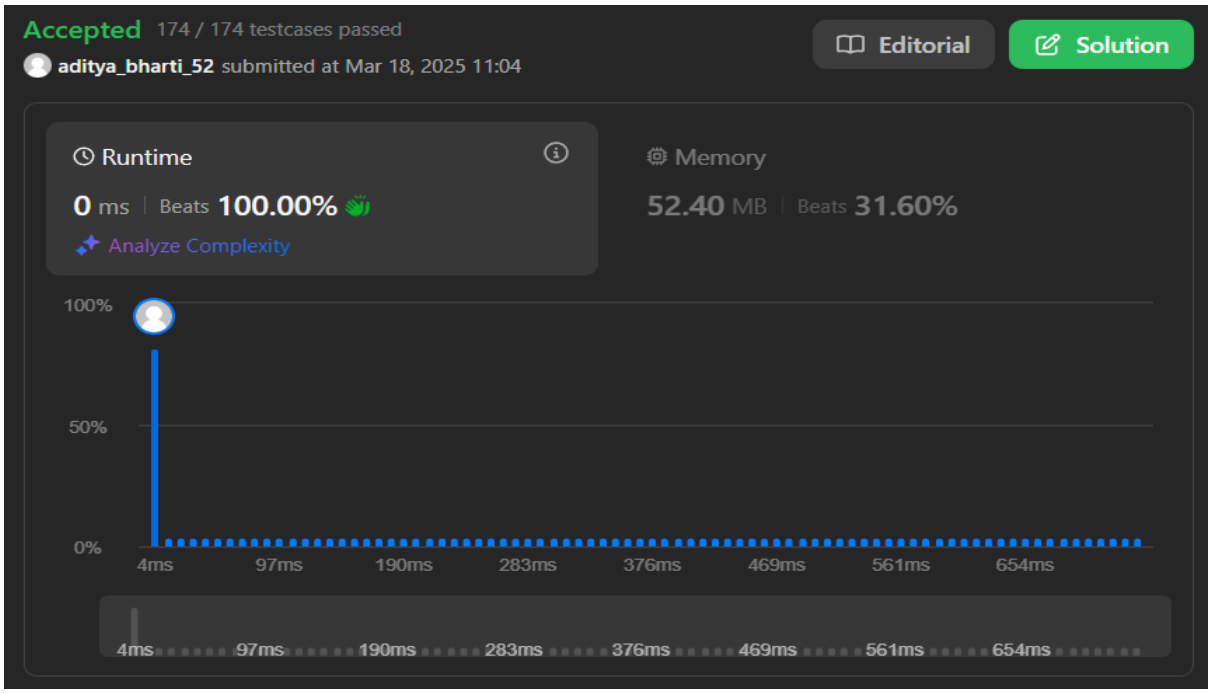
AIM: Jump Game

CODE:

```
class Solution {
public:
    bool canJump(vector<int>& nums) {
        int maxReach = 0;
        int n = nums.size();

        for (int i = 0; i < n; ++i) {
            if (i > maxReach) return false;
            maxReach = max(maxReach, i + nums[i]);
            if (maxReach >= n - 1) return true;
        }
        return false;
    }
};
```

OUTPUT:



EXPERIMENT 7

Problem 5

AIM: Unique Paths

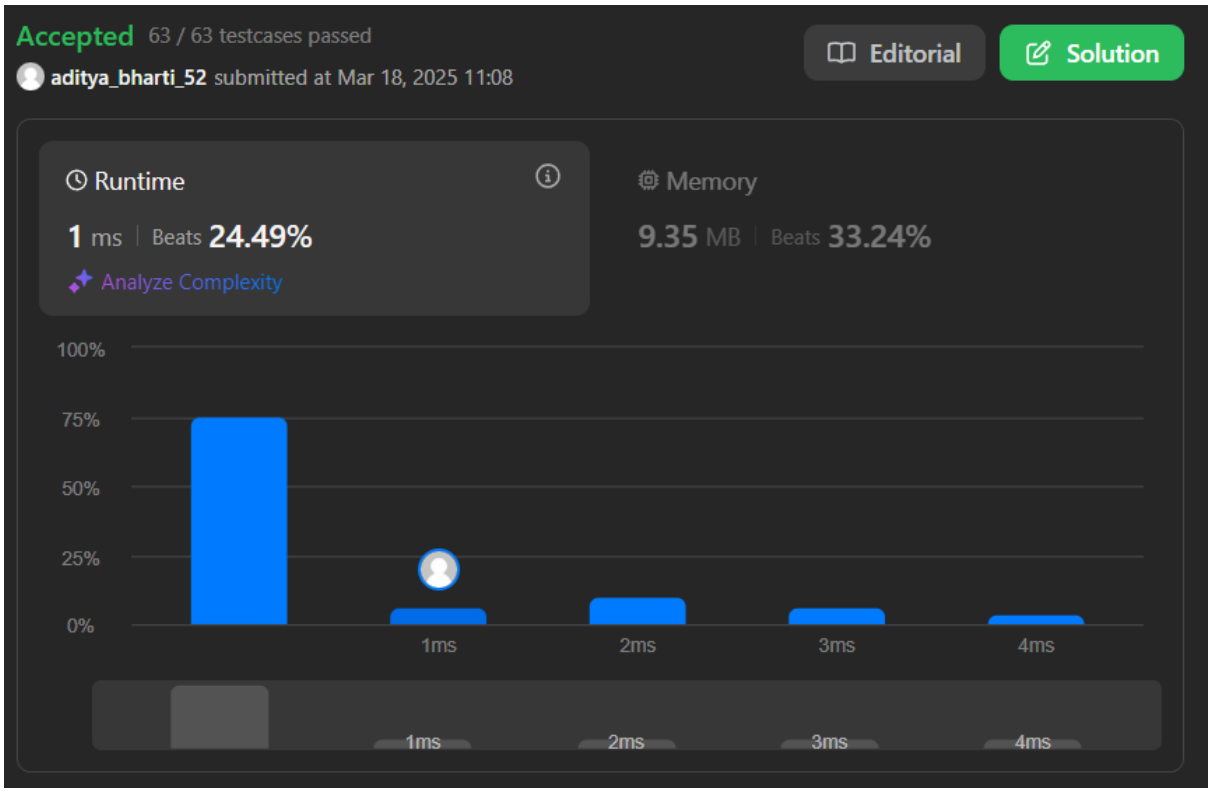
CODE:

```
class Solution {
public:
    int uniquePaths(int m, int n) {
        vector<vector<int>> dp(m, vector<int>(n, 1));

        for (int i = 1; i < m; ++i) {
            for (int j = 1; j < n; ++j) {
                dp[i][j] = dp[i - 1][j] + dp[i][j - 1];
            }
        }

        return dp[m - 1][n - 1];
    }
};
```

OUTPUT:



EXPERIMENT 7

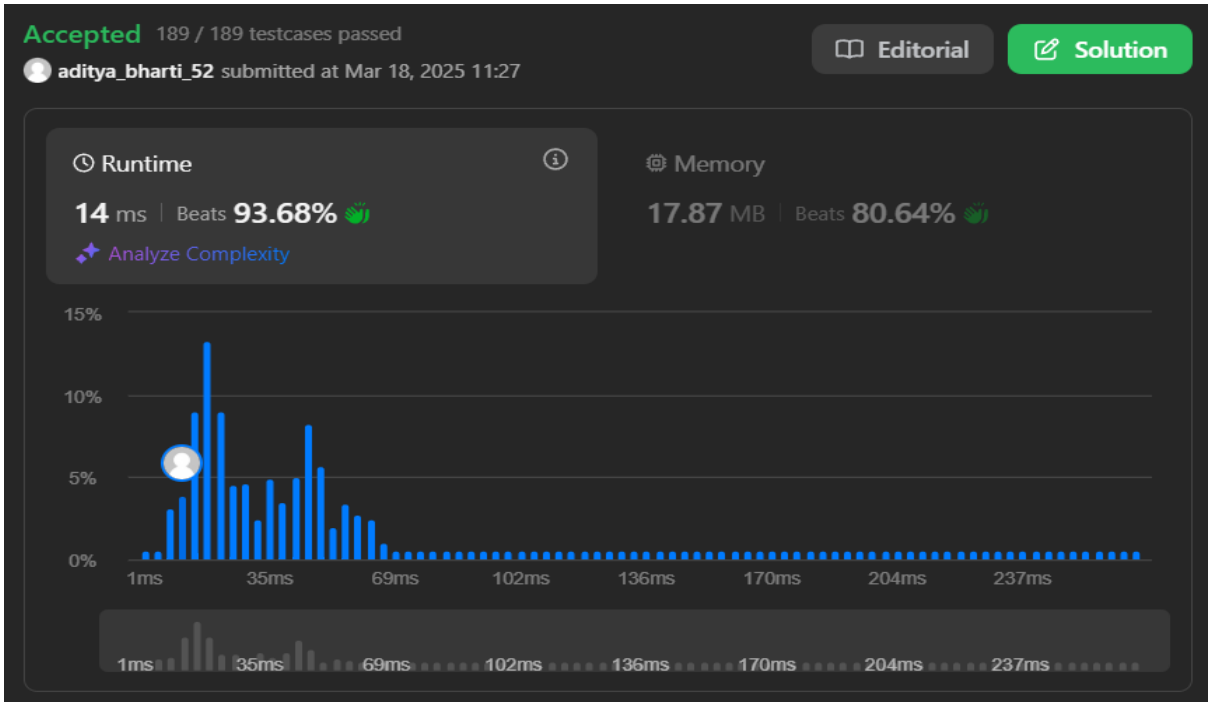
Problem 6

AIM: Coin Change

CODE:

```
class Solution {
public:
    int coinChange(vector<int>& coins, int amount) {
        vector<int> dp(amount + 1, INT_MAX);
        dp[0] = 0;
        for (int coin : coins) {
            for (int i = coin; i <= amount; ++i) {
                if (dp[i - coin] != INT_MAX) {
                    dp[i] = min(dp[i], dp[i - coin] + 1);
                }
            }
        }
        return dp[amount] == INT_MAX ? -1 : dp[amount];
    }
};
```

OUTPUT:



EXPERIMENT 7

Problem 7

AIM: Longest Increasing Subsequence

CODE:

```
class Solution {
public:
    int lengthOfLIS(vector<int>& nums) {
        if (nums.empty()) return 0;
        vector<int> dp(nums.size(), 1);
        int maxLength = 1;
        for (int i = 1; i < nums.size(); ++i) {
            for (int j = 0; j < i; ++j) {
                if (nums[i] > nums[j]) {
                    dp[i] = max(dp[i], dp[j] + 1);
                }
            }
            maxLength = max(maxLength, dp[i]);
        }
        return maxLength;
    }
};
```

OUTPUT: