Experiment-6

Student Name: MEGHA SHREE   **UID:** 22BCS10381
Branch: BE-CSE   **Section/Group:** KPIT-901/B
Semester: 6th   **Date of Performance:** 06/03/25
Subject Name: Advanced Programming Lab - 2   **Subject Code:** 22CSP-351

1. **Aim:** Dynamic Programming.
   1. Problem: 55. Jump Game.
   2. Problem: 322. Coin Change.

2. **Objective:**
   1. This program checks whether you can reach the last index of the given array nums, where each element represents the maximum number of steps you can jump forward.
   2. This program finds the minimum number of coins needed to make up a given amount using coins of different denominations. If it is not possible, it returns -1.

3. **Implementation/Code:**

1.)
```cpp
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;
  }
};
```

2.)

```
class Solution {
public:
    int coinChange(vector<int>& coins, int amount) {
        vector<int> dp(amount + 1, amount + 1);
        dp[0] = 0;

        for (int i = 1; i <= amount; i++) {
            for (int coin : coins) {
                if (i >= coin) {
                    dp[i] = min(dp[i], dp[i - coin] + 1);
                }
            }
        }

        return (dp[amount] == amount + 1) ? -1 : dp[amount];
    }
};
```

4. Output:

1.

2.
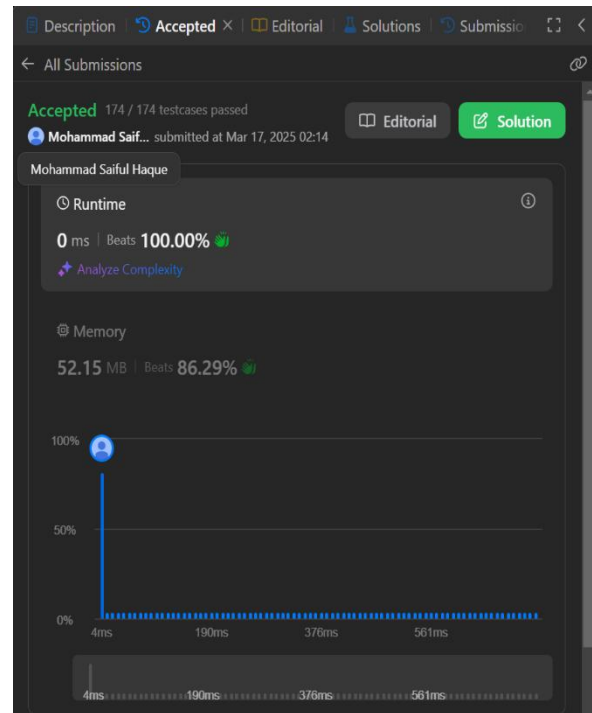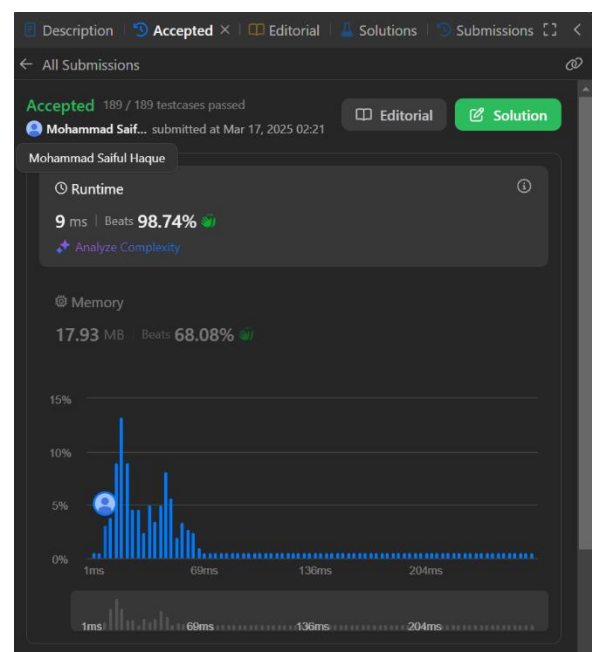
5. **Time Complexity:**
   1. O(n)
   2. O (n * m)

6. **Space Complexity:**
   1. O (1)
   2. O(n)

7. **Learning Outcome:**
   1. Understand greedy algorithms and dynamic programming (DP) and their applications in solving optimization problems.
   2. Learn how to track reachability in arrays using a greedy approach and how to use a bottom-up DP approach to compute the minimum number of coins.
   3. Gain experience in handling edge cases, such as unreachable indices in the Jump Game and impossible amounts in the Coin Change problem.
   4. Improve problem-solving skills by breaking down complex problems into smaller sub-problems and optimizing solutions using efficient conditions and early exits.
   5. Learn how to optimize space complexity (O (1) in Jump Game, O(n) in Coin Change) while maintaining efficient time complexity.
   6. Gain insight into real-world applications, such as pathfinding, resource allocation, and financial computations, by applying DP and greedy strategies.