📘 LeetCode 70: Climbing Stairs

🔗 0. Question with Explanation

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?

Example:

```
Input: n = 2
Output: 2
Explanation: 1 step + 1 step, or 2 steps

Input: n = 3
Output: 3
Explanation: 1+1+1, 1+2, 2+1
```

🔗 1. Definition and Purpose

- **Concept**: Dynamic Programming / Fibonacci Series
- **Why it exists**: Solve overlapping subproblems using memoization or iterative DP
- **Problem it solves**: Counting distinct sequences of steps using fixed step sizes

---

🔗 2. Syntax and Structure

```java
public int climbStairs(int n);
```

- Input: integer `n` (number of stairs)
- Output: number of distinct ways to climb to the top

🔗 3. Practical Examples

```
Input: 5
Output: 8
Explanation: Like Fibonacci(6): F(5) = F(4) + F(3)
```

🔗4. Internal Working

🔗Approach 1: Bottom-Up DP (Fibonacci)

- Time: O(n), Space: O(1)

```
public int climbStairs(int n) {
    if (n <= 2) return n;
    int one = 2, two = 1;
    for (int i = 3; i <= n; i++) {
        int temp = one + two;
        two = one;
        one = temp;
    }
    return one;
}
```

🔗Approach 2: Top-Down Recursion + Memoization

- Time: O(n), Space: O(n)

```
public int climbStairs(int n) {
    int[] memo = new int[n + 1];
    return climb(n, memo);
}

private int climb(int n, int[] memo) {
    if (n <= 2) return n;
    if (memo[n] != 0) return memo[n];
    memo[n] = climb(n - 1, memo) + climb(n - 2, memo);
    return memo[n];
}
```

🔗5. Best Practices

- Use bottom-up if you want constant space
- Use memoized recursion if you're building from base logic

🔗6. Related Concepts

- Fibonacci Numbers

- DP with choices (e.g., Coin Change, Decode Ways)

🔗7. Interview & Real-world Use

- Frequently asked in interviews
- Helps build intuition for other recursive+DP problems

🔗8. Common Errors & Debugging

- StackOverflow with plain recursion
- Off-by-one in array index

🔗9. Java Version Updates

- Works with all Java versions
- New Java versions offer `record` and pattern matching but not used here

---

🔗10. Practice and Application

- LeetCode 70: Climbing Stairs
- LeetCode 746: Min Cost Climbing Stairs
- HackerRank: Fibonacci Modified
