509. Fibonacci Number LeetCode

🧱 Fibonacci using Memoization (Top-Down DP)

💡 Problem

Compute the `n-th` Fibonacci number using **recursion with memoization** to avoid redundant calculations.

🧱 Approach — Recursive + DP (Top-Down)

We use a `dp` array to **store already computed results**, so that each Fibonacci value is computed only once.

⚙️ Code

```
class Solution {
public:
    int fibo(int n, vector<int>& dp) {
        if (n <= 1) return n;                // Base case: F(0)=0, F(1)=1
        if (dp[n - 1] != -1) return dp[n - 1]; // If already computed, return it

        // Store result before returning
        return dp[n - 1] = fibo(n - 1, dp) + fibo(n - 2, dp);
    }

    int fib(int n) {
        vector<int> dp(n, -1);  // Initialize DP array of size n with -1
        return fibo(n, dp);     // Compute F(n)
    }
};
```

🧠 Key Points

- **Base Case:**

  `fibo(0) = 0`, `fibo(1) = 1`

- **Recurrence Relation:**

  `fibo(n) = fibo(n-1) + fibo(n-2)`

- **Memoization:**
  - Store results in `dp` to prevent recomputation
  - Use `dp[n-1]` because vector size = `n` (indices `0` to `n-1`)

⏱️ Time Complexity

> O(n) — each Fibonacci number is computed once.

💾 Space Complexity

> O(n) — for the recursion stack + dp array.

✅ Example

For `n = 5`:

```
fibo(5)
 → fibo(4) + fibo(3)
 → (fibo(3) + fibo(2)) + (fibo(2) + fibo(1))
 → ...
Result = 5
```
