1. Recursion Approach

Key Concept

- Recursion involves solving a problem by breaking it into smaller subproblems of the same type.
- The Fibonacci sequence is naturally recursive, as each term depends on the two preceding ones:
  F(n)=F(n−1)+F(n−2), with base cases:
  - F(0)=0
  - F(1)=1

Advantages

- Simple and intuitive to write.
- Mirrors the mathematical definition.

Disadvantages

- **Exponential Time Complexity (O(2^n)):** Repeatedly recalculates the same values.
- **Stack Overflow Risk:** Uses a lot of memory for deep recursion.

2. Dynamic Programming Approach

Key Concept

- Solves problems by breaking them into overlapping subproblems and storing results to avoid redundant calculations.
- Two forms:
  - **Tabulation (Bottom-Up):** Build solutions iteratively from smaller sub-problems.

Tabulation is a **Dynamic Programming** technique that solves problems in a **bottom-up manner**. Instead of using recursion, it builds the solution iteratively by solving smaller subproblems first and storing their results in a table (usually an array).

- **Linear Time Complexity O(n):** Much faster than recursion.
- Avoids stack overflow.

Disadvantages

- Higher memory usage for large arrays.

Iterative Approach

Key Concept

- Use a loop to calculate Fibonacci numbers iteratively, keeping track of only the last two numbers.

Advantages

- **Linear Time Complexity O(n):** Similar to dynamic programming.
- **Constant Space Complexity O(1):** Doesn’t require extra memory.

Disadvantages

- Slightly less intuitive compared to recursion for beginners.