**WEEK1\_ALGORITHMS\_DATA\_STRUCTURES**

**EXERCISE-7**

**Understand Recursive Algorithms**

**Recursion:**

* **Concept**: A recursive algorithm solves a problem by solving smaller instances of the same problem and combining their results. It typically involves a base case to terminate the recursion and a recursive case that reduces the problem size.
* **Simplification**: Recursion can simplify code for problems that can be broken down into similar subproblems, like calculating factorials, Fibonacci sequences, or forecasting based on historical data.

**Example**: Computing the Fibonacci sequence using recursion simplifies the problem into subproblems of calculating smaller Fibonacci numbers.

**Analysis**

**Time Complexity:**

* **Recursive Time Complexity**: O(n), where n is the number of years. Each recursive call decreases the number of years by 1 until it reaches the base case.
* **Space Complexity**: O(n) due to the call stack growth with each recursive call.

**Optimization:**

* **Memoization**: Store results of subproblems to avoid redundant calculations. This is particularly useful if the problem can be broken down into overlapping subproblems.
* **Iteration**: Convert the recursive approach to an iterative one if the problem size is large to reduce stack space and potentially improve performance.