Understand Recursive Algorithms

What is Recursion?

Recursion is when a function calls itself to solve smaller subproblems. It's useful for problems that can be broken into repeated sub-tasks, such as predicting future values over time.

Why use it?

It simplifies code by reducing complex iterative logic into cleaner, base-case–driven logic.

Analysis -

Time Complexity

O(n) — because the function is called once for each year (n = number of years).

Problem -

For **large n**, this results in deep recursion, which can hit stack limits.

Optimization: Use Memoization or Iterative Approach.

Iterative Version (Optimized) -

```
public static double predictFutureValueIterative(double amount, double rate, int years)
{
   for (int i = 0; i < years; i++) {
      amount *= (1 + rate);
   }
```

```
    return amount;
}
```

Time Complexity: O(n)

Space Complexity: O(1) — avoids recursive call stack

Final Recommendation -

Use **recursion** for small n or educational purposes.

Use **iterative or dynamic programming (DP)** for production tools to avoid stack overflows and reduce memory use.