**Understanding Recursive Algorithms**

**What is Recursion?**

Recursion is a programming technique where a method calls itself to solve a smaller subproblem until it reaches a base case. This approach is useful when a problem can be broken into repeating substructures, like in mathematics, searching, or forecasting.

**Structure of Recursive Function**

A recursive function generally has two parts:

**Base Case:** When the recursion ends (stopping condition)

**Recursive Case:** The function calls itself with a modified input

**Example Pattern:**

if (base\_case\_condition)

return base\_result;

else

return recursive\_call(modified\_input);

**Recursive Formula in Forecasting**

For financial forecasting, the future value can be calculated recursively:

FV(n) = FV(n - 1) × (1 + growthRate)

Where:

FV = Future Value

n = years

growthRate = expected annual growth

**C# Code Example**

public static double PredictFutureValue(double currentValue, double growthRate, int years)

{

if (years == 0)

return currentValue;

return PredictFutureValue(currentValue, growthRate, years - 1) \* (1 + growthRate);

}

**Benefits of Using Recursion**

Cleaner and shorter code

Easier to match mathematical formulas

Useful when a problem naturally breaks into subproblems

**When Not to Use Recursion**

If the depth of recursion is very high (can cause stack overflow)

When a simple loop (iteration) is more efficient