

## Exercise 7: Financial Forecasting

### Scenario:

You are developing a financial forecasting tool that predicts future values based on past data.

### Steps:

#### 1. Understand Recursive Algorithms:

- Explain the concept of recursion and how it can simplify certain problems.

Recursion is a function which calls itself to solve smaller instances of a problem. It's useful for problems that can be broken into identical subproblems.

Examples:- Fibonacci Series, Factorial, Tree Traversal, Value Forecasting when the current value depends on the previous one.

#### 2. Setup:

- Create a method to calculate the future value using a recursive approach.

So, I'll be forecasting the future values using the Compound Interest formula:

$$FV = PV \times (1+r)^n$$

FV = Future Value

PV = Present Value

r = Rate of interest

n = Number of years

#### 3. Implementation:

- Implement a recursive algorithm to predict future values based on past growth rates.

using System;

Code:

```
public class Forecast
{
    // Recursive method to calculate future value
    public static double FutureValue(double pv, double rate, int n)
    {
        if (n == 0) return pv;
        return FutureValue(pv, rate, n - 1) * (1 + rate);
    }
}
```

```

// Optimized with memoization

public static double FutureValueMemo(double pv, double rate, int n,
double[] memo)
{
    if (n == 0) return pv;
    if (memo[n] != 0) return memo[n];
    memo[n] = FutureValueMemo(pv, rate, n - 1, memo) * (1 + rate);
    return memo[n];
}

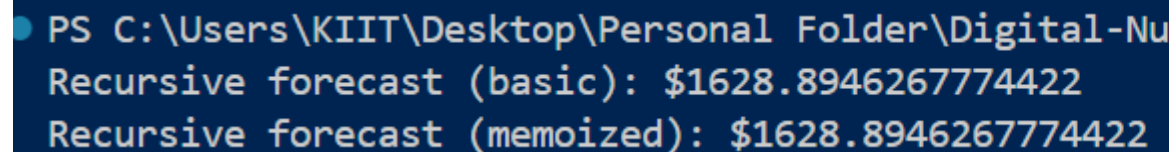
public static void Main(string[] args)
{
    double pv = 1000;    // Starting amount
    double rate = 0.05;   // 5% annual growth
    int years = 10;

    Console.WriteLine("Recursive forecast (basic): $" + FutureValue(pv, rate,
years));

    double[] memo = new double[years + 1];
    Console.WriteLine("Recursive forecast (memoized): $" +
FutureValueMemo(pv, rate, years, memo));
}
}

```

Output:



```

PS C:\Users\KIIT\Desktop\Personal Folder\Digital-Nu
Recursive forecast (basic): $1628.8946267774422
Recursive forecast (memoized): $1628.8946267774422

```

#### 4. Analysis:

- Discuss the time complexity of your recursive algorithm.  
 $O(n)$  Since basic Recursion so it'll be making only 1 call per year besides the memorized version also has the Time Complexity of  $O(n)$  but avoids re-computation in more complex recursive patterns.

- Explain how to optimize the recursive solution to avoid excessive computation.
- Memoization: Store previously computed values.
- Iteration: Convert to a loop if recursion is deep.
- Tail Recursion: Where supported, for efficient stack handling .