**COGNIZANT Deep\_Skilling**

**Data Structures and Algorithms**

Question 7: Financial Forecasting

Code:

using System;

class FinancialForecast

{

    public static double CalculateFutureValueRecursive(double presentValue, double growthRate, int years)

    {

        if (years == 0)

            return presentValue;

        return CalculateFutureValueRecursive(presentValue \* (1 + growthRate), growthRate, years - 1);

    }

    public static double CalculateFutureValueIterative(double presentValue, double growthRate, int years)

    {

        for (int i = 0; i < years; i++)

        {

            presentValue \*= (1 + growthRate);

        }

        return presentValue;

    }

    static void Main()

    {

        double presentValue = 1000;

        double growthRate = 0.05;

        int years = 6;

        double futureRecursive = CalculateFutureValueRecursive(presentValue, growthRate, years);

        Console.WriteLine($"[Recursive] Future value after {years} years: {futureRecursive:C2}");

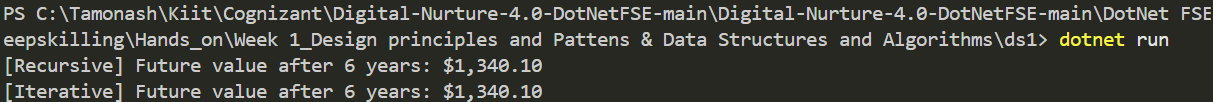
        double futureIterative = CalculateFutureValueIterative(presentValue, growthRate, years);

        Console.WriteLine($"[Iterative] Future value after {years} years: {futureIterative:C2}");

    }

}

**Output:**



**Understand Recursive Algorithms:**

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

Answer:

Recursion is a programming technique in which a function calls itself to solve smaller instances of a problem until it reaches a base case. This approach is particularly effective for problems that exhibit a repetitive or hierarchical structure, such as mathematical computations, tree traversal, and solving problems that can be broken down into similar sub-problems. In financial forecasting, recursion can simplify the calculation of future values by repeatedly applying a growth formula for each time period. By defining the base case (e.g., year 0 as the present value) and recursively applying the growth factor for each subsequent year, the recursive approach can mirror the logic of compound interest or exponential growth in a clear and concise way.

**Analysis Questions**

1. Discuss the time complexity of your recursive algorithm.

Answer:

The time complexity of the basic recursive algorithm for calculating future value is O(n), where n is the number of years for which the forecast is being made. This is because each recursive call performs a constant-time multiplication and then calls itself with a reduced input, repeating this process n times until the base case is reached. Since there is only one recursive call per level and no overlapping subproblems, the algorithm grows linearly with the number of years. This makes it efficient in terms of time for reasonably sized inputs, although each recursive call adds to the call stack, which may impact memory usage.

1. Explain how to optimize the recursive solution to avoid excessive computation.

Answer:

While the basic recursive solution is simple and elegant, it can lead to issues such as stack overflow or unnecessary overhead for large input sizes due to deep call stacks. To optimize the solution, it is often better to replace the recursive approach with an iterative version, which uses a loop instead of recursive calls and avoids the overhead of function calls. If recursion must be retained for conceptual clarity or educational purposes, tail recursion (where the recursive call is the final action) can be used in some languages that optimize tail calls. In more complex forecasting scenarios where values may depend on multiple subproblems (such as in dynamic models), memoization can be introduced to store and reuse results of previous computations, significantly reducing redundant processing.