DSA-7

Recursion is a programming technique where a function calls itself to solve smaller instances of the same problem

**Base case**: One or more simple conditions under which the function returns directly, without further recursion.

**Recursive case**: The part where the function calls itself with “smaller” or simpler inputs, working toward the base case.

//FinancialForecast.java

package DSA7;

public class FinancialForecast {

    public static double futureValue(double[] rates, double principal, int n){

        if(n <= 0){

            return principal;

        }

        double prev = futureValue(rates, principal, n-1);

        return prev \* (1.0 + rates[n-1]);

    }

    public static void main(String[] args){

        double[] rates = {0.05, 0.03, -0.02, 0.04};

        double principal = 1000.0;

        int n = rates.length;

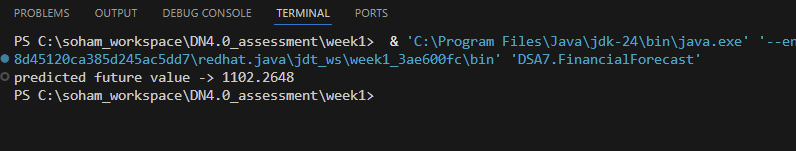
        double result = futureValue(rates, principal, n);

        System.out.println("predicted future value -> " + result);

    }

}

//Output



Time Complexity of the code is O(n)

Space Complexity of the code is O(n)

Optimizations include tail-recursion (where available), straight iteration (best for O(1) space), or DP/memoization if you need a table of results and avoid excessive computation