

## Financial Forecasting

```
using System;

using System.Collections.Generic;

using System.Diagnostics;

namespace FinancialForecasting
{
    public class FinancialForecaster
    {

        public decimal CalculateFutureValueRecursive(decimal currentValue, decimal
annualGrowthRate, int years)
        {

            if (years <= 0)
            {
                return currentValue;
            }

            decimal nextValue = currentValue * (1 + annualGrowthRate / 100m);
            return CalculateFutureValueRecursive(nextValue, annualGrowthRate, years - 1);
        }

        private Dictionary<(decimal, decimal, int), decimal> memoCache = new Dictionary<(decimal,
decimal, int), decimal>();

        public decimal CalculateFutureValueMemoized(decimal currentValue, decimal
annualGrowthRate, int years)
        {

            var key = (currentValue, annualGrowthRate, years);
```

```
if (memoCache.TryGetValue(key, out decimal cachedValue))
{
    return cachedValue;
}
```

```
if (years <= 0)
{
    memoCache[key] = currentValue;
    return currentValue;
}
```

```
decimal nextValue = currentValue * (1 + annualGrowthRate / 100m);
decimal result = CalculateFutureValueMemoized(nextValue, annualGrowthRate, years - 1);
memoCache[key] = result;
return result;
}
```

```
public decimal CalculateFutureValueIterative(decimal currentValue, decimal annualGrowthRate,
int years)
```

```
{
    decimal result = currentValue;
    for (int i = 0; i < years; i++)
    {
        result *= (1 + annualGrowthRate / 100m);
    }
    return result;
}
}
```

```
class Program
```

```
{
```

```
static void Main(string[] args)
{
    FinancialForecaster forecaster = new FinancialForecaster();

    decimal initialValue = 10000m;
    decimal growthRate = 5m;
    int years = 10;

    var stopwatch = Stopwatch.StartNew();

    decimal recursiveResult = forecaster.CalculateFutureValueRecursive(initialValue, growthRate,
years);
    stopwatch.Stop();

    Console.WriteLine($"Recursive Result: {recursiveResult:C2}");
    Console.WriteLine($"Time taken: {stopwatch.ElapsedTicks} ticks");

    stopwatch.Restart();

    decimal memoizedResult = forecaster.CalculateFutureValueMemoized(initialValue,
growthRate, years);
    stopwatch.Stop();

    Console.WriteLine($"Memoized Result: {memoizedResult:C2}");
    Console.WriteLine($"Time taken (first run): {stopwatch.ElapsedTicks} ticks");

    stopwatch.Restart();

    decimal iterativeResult = forecaster.CalculateFutureValueIterative(initialValue, growthRate,
years);
    stopwatch.Stop();

    Console.WriteLine($"Iterative Result: {iterativeResult:C2}");
    Console.WriteLine($"Time taken: {stopwatch.ElapsedTicks} ticks");

    Console.WriteLine("\nTesting memoization with repeated calculations...");
    stopwatch.Restart();
    for (int i = 0; i < 1000; i++)
```

```

    {
        forecaster.CalculateFutureValueMemoized(initialValue, growthRate, years);
    }

    stopwatch.Stop();

    Console.WriteLine($"1000 memoized calls: {stopwatch.ElapsedTicks} ticks");

    stopwatch.Restart();

    for (int i = 0; i < 1000; i++)
    {
        forecaster.CalculateFutureValueRecursive(initialValue, growthRate, years);
    }

    stopwatch.Stop();

    Console.WriteLine($"1000 recursive calls: {stopwatch.ElapsedTicks} ticks");
}
}
}

```

The screenshot shows the Visual Studio Code interface with the terminal output of a .NET application. The application is running in a PowerShell window at the path `PS C:\Users\dell\FinancialForecasting>`. The output shows the results of three different calculation methods for a future value of ₹ 16,288.95 over 1000 iterations:

- Recursive:** Result: ₹ 16,288.95, Time taken: 14227 ticks.
- Memoized:** Result: ₹ 16,288.95, Time taken (first run): 29135 ticks.
- Iterative:** Result: ₹ 16,288.95, Time taken: 2259 ticks.

The application also includes a test for memoization with repeated calculations, showing that 1000 memoized calls took 20225 ticks and 1000 recursive calls took 12902 ticks.

```

PS C:\Users\dell\FinancialForecasting> dotnet run
Recursive Result: ₹ 16,288.95
Recursive Result: ₹ 16,288.95
Time taken: 14227 ticks

Memoized Result: ₹ 16,288.95
Time taken (first run): 29135 ticks

Iterative Result: ₹ 16,288.95
Time taken: 2259 ticks

Testing memoization with repeated calculations...
1000 memoized calls: 20225 ticks
1000 recursive calls: 12902 ticks
PS C:\Users\dell\FinancialForecasting>

```