Exercise 7: Financial Forecasting

Steps1:. Understand Recursive Algorithms:

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

* Recursion is a technique where a function solves a problem by calling itself with a simpler version of the problem. It’s particularly useful for tasks that can be naturally divided into similar sub-tasks.

**Key Elements of Recursion**

1. **Base Case**: This stops the recursion when a simple, direct answer can be provided without further function calls. It prevents the function from calling itself indefinitely.
2. **Recursive Case**: This involves the function calling itself with modified arguments to solve a smaller version of the problem, gradually moving toward the base case.

**Benefits of Recursion**

1. **Simplifies Code**: Recursion can make code more intuitive and easier to write for problems that have a natural hierarchical structure, like tree traversal or divide-and-conquer algorithms.
2. **Breaks Down Problems**: It helps in breaking complex problems into smaller, manageable subproblems, each of which is easier to solve.

4. Analysis:

o Discuss the time complexity of your recursive algorithm.

The time complexity of the recursive algorithm for calculating the future value is O(n), where n is the number of periods.

**Explanation:**

1. **Recursive Calls**:
   * The recursive method futureValue makes a recursive call with n - 1 until n reaches 0.
   * This results in a total of n recursive calls (one for each period).
2. **Work Done Per Call**:
   * Each recursive call involves a constant amount of work: multiplying the result of the recursive call by (1 + r).

**Overall Time Complexity:**

* Since there are n recursive calls, and each call does a constant amount of work, the overall time complexity is O(n).

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

o Optimize the recursive solution and avoid excessive computation, you can use **memoization** or **dynamic programming** techniques. These methods help reduce redundant calculations by storing intermediate results.

**Optimizing Recursive Solution with Memoization**

**Memoization** involves storing the results of expensive function calls and reusing them when the same inputs occur again. For the future value calculation, since each recursive call is dependent only on its direct subproblem, memoization can save results for each period's future value.

Here’s how you can apply memoization to your recursive solution:

1. **Create a Cache**: Use a data structure like an array or a hash map to store the results of previously computed values.
2. **Check Cache Before Computation**: Before performing the recursive calculation, check if the result for the given period is already in the cache.
3. **Store Results in Cache**: After computing the result for a given period, store it in the cache for future reference.