FINANCIAL FORECASTING

**1. Understanding Recursive Algorithms**

**Concept of Recursion**:

* **Definition**: Recursion is a programming technique where a function calls itself directly or indirectly to solve a problem. A recursive function typically has two parts: a base case that stops the recursion and a recursive case that reduces the problem towards the base case.
* **Simplification**: Recursion can simplify complex problems by breaking them down into smaller, more manageable subproblems that are similar to the original problem. This approach is particularly useful for problems that have a natural hierarchical structure, such as tree traversal, factorial computation, and certain mathematical sequences.

**4. Analysis**

**Time Complexity**:

* The time complexity of the recursive algorithm calculateFutureValue is O(n), where n is the number of years. This is because the function makes a single recursive call for each year until the base case is reached.

**Optimizing the Recursive Solution**:

* **Memoization**: One way to optimize a recursive solution is by using memoization, which involves storing the results of expensive function calls and reusing them when the same inputs occur again. In this case, however, memoization is not particularly necessary since each year's calculation depends directly on the previous year's result, leading to a straightforward linear recursion.
* **Iterative Approach**: Another optimization approach is to convert the recursion into an iterative solution, which can be more efficient in terms of space complexity by avoiding the overhead of recursive function calls.

**Advantages of the Iterative Approach**:

* The iterative version has the same time complexity of O(n) but with a space complexity of O(1), as it doesn't involve the overhead of maintaining a call stack for recursion.

**Conclusion**:

* Recursion is a powerful tool for breaking down complex problems and can lead to elegant solutions. However, it's essential to be aware of its potential drawbacks, such as excessive memory usage and stack overflow for deep recursions. In many cases, iterative solutions can achieve the same results with better space efficiency.