**FINANCIAL FORECASTING**

Recursion is a method where a function repeatedly calls itself to tackle a problem by breaking it down into simpler, smaller versions of the same problem until a base case is reached. This technique is particularly effective for problems that can be modelled with a hierarchical structure, such as navigating directories or handling nested data. It simplifies complex issues by dividing them into more manageable tasks.

To enhance the efficiency of recursive solutions and avoid redundant calculations, consider these strategies:

1. **Dynamic Programming**: Solve a problem by breaking it into smaller, simpler problems, solving each only once, and keeping track of the results.
2. **Iterative Solutions**: Convert recursive approaches into loops to avoid the overhead associated with multiple function calls.
3. **Memorization**: Store results of previous calculations to quickly retrieve them when needed, reducing redundant processing.
4. **Pruning**: Eliminate unnecessary branches or paths in the problem space to reduce the computational effort.
5. **Alternative Algorithms**: Use methods that require less depth in recursion to prevent stack overflow and optimize resource use.
6. **Lazy Evaluation**: Delay the execution of certain computations until absolutely necessary to improve performance.
7. **Tail Recursion Optimization**: Adjust the recursion to ensure that the recursive call is the last action in the function, which can help minimize stack usage.

Applying these methods can help create more efficient recursive solutions and mitigate common issues such as excessive memory consumption and stack overflow.

**Time Complexity**: The time complexity of a recursive function is often O(n) if it involves n recursive calls to arrive at the final result.