**Understand Recursive Algorithms**

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

=>Recursion: Recursion is a programming technique where a function calls itself directly or indirectly to solve a problem. Each recursive call works on a smaller instance of the same problem, and the solution to the overall problem is built up from the solutions to these smaller instances.

Base Case and Recursive Case: A recursive algorithm must have at least one base case that does not make a recursive call, to prevent infinite recursion. The recursive case reduces the problem size and makes the recursive call.

Advantages of Recursion:

1.Simplicity: Recursive solutions can be more intuitive and easier to implement for problems that have a natural recursive structure, such as tree traversal, factorial computation, and Fibonacci sequence.

2.Divide-and-Conquer: Many algorithms (e.g., quicksort, mergesort) benefit from the divide-and-conquer strategy, which is naturally implemented using recursion.

**Analysis**

Q.Discuss the time complexity of your recursive algorithm?

=>The time complexity of the recursive algorithm is O(n), where n is the number of periods. This is because the function makes a recursive call for each period, resulting in n recursive calls in total.

Q.Explain how to optimize the recursive solution to avoid excessive computation?

=>Optimizing the Recursive Solution:

1.Memoization: To avoid redundant computations in recursive algorithms, memoization can be used. Memoization involves storing the results of expensive function calls and reusing the stored results when the same inputs occur again.

2.Iterative Approach: Converting the recursive solution to an iterative one can improve performance and avoid the overhead of recursive calls.