1. **Understand Recursive Algorithms:**

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

Recursion is a programming technique where a function calls itself directly or indirectly to solve a problem. This self-referential nature allows certain problems to be solved in a more elegant and straightforward way.

How Recursion Works

Base Case: This is the condition under which the recursion terminates. Without a base case, the function would call itself indefinitely, leading to a stack overflow.

Recursive Case: This defines how the problem should be divided into smaller subproblems and how the function should call itself with these subproblems.

1. **Analysis:**
   1. Discuss the time complexity of your recursive algorithm.

Time Complexity with Iteration: O(n), Space Complexity with Iteration: O(1)

* 1. Explain how to optimize the recursive solution to avoid excessive computation.

Soln: Memoization: Useful for optimizing recursive solutions by avoiding redundant calculations. Space complexity depends on the number of unique subproblem solutions.

Dynamic Programming (Tabulation): Avoids recursion by solving problems iteratively and storing intermediate results in a table. Generally uses more space but has lower function call overhead.