**Understand Recursive Algorithms**

**Concept of Recursion:**

* **Definition:** Recursion is a technique where a function calls itself to solve smaller instances of the same problem until it reaches a base case.
* **Simplification:** Recursion can simplify problems by breaking them down into smaller, more manageable sub-problems. It is particularly useful for problems that have a naturally recursive structure, such as the Fibonacci sequence, tree traversal, and factorial computation.

**Example of Recursion:**

* **Factorial Calculation:** The factorial of a number nnn is defined as n!=n×(n−1)!n! = n \times (n-1)!n!=n×(n−1)! with the base case 0!=10! = 10!=1.

java

Copy code

public int factorial(int n) {

if (n == 0) {

return 1; // Base case

} else {

return n \* factorial(n - 1); // Recursive call

}

}

**Analysis**

**Time Complexity:**

* The time complexity of the recursive algorithm is O(n), where nnn is the number of years. This is because each recursive call reduces the problem size by one, leading to nnn recursive calls.

**Optimizing the Recursive Solution:**

* **Memoization:** To avoid excessive computation and repeated calculation of the same values, memoization can be used. Memoization stores the results of expensive function calls and reuses the stored results when the same inputs occur again.

**Discussion:**

* **Time Complexity with Memoization:** The optimized solution with memoization has a time complexity of O(n) for the first computation and O(1) for subsequent computations for the same number of years. This significantly reduces redundant calculations.
* **Space Complexity:** The space complexity is O(n) due to the storage of intermediate results in the memo map.

**Advantages of Recursive Approach:**

* **Readability:** The recursive solution is often more readable and closer to the natural description of the problem.
* **Simplicity:** For problems with a natural recursive structure, recursion can simplify the implementation.

**Disadvantages:**

* **Overhead:** Recursive calls have overhead due to function call stack maintenance, which can be costly for deep recursion.
* **Stack Limitations:** Deep recursion can lead to stack overflow if not optimized properly.