**Recursive Algorithms**

**Recursion** is when a function calls itself to solve a smaller part of the original problem. It typically includes:

* A **base case** (to stop recursion),
* A **recursive case** (where the function calls itself).

The Base Case is to Stop the recursion Process and hence the required output is formed.

The recursive case calls the function itself, to iterate them with different arguments.

**For example:** Factorial using recursion

int factorial(int n) {

if (n == 0) return 1; // Base case

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

}

Here the n value is reduced by one in every recursive call until the n value is 0.

When the n value becomes 0 i.e. the **Base Case** then the recusive calls are stopped.

**Method to calculate the future value**

FutureValue(n) = PastValue × (1 + growthRate)^n

The past value is multiplied with the growth rate of the company, the growth rate is taken by the past years performances.

**Time complexity of your recursive** **algorithm**

The Time complexity of the code is O(n)

As the code runs for no of years to estimate the Future Values.

**Optimize the recursive solution**

To optimize the Recursive Solution we can use **Memoization** Technique.

**Memoization** is a powerful optimization technique used in **recursive algorithms** to **avoid redundant computations** by **storing results** of expensive function calls and **reusing** them when the same inputs occur again.