**Understanding Recursive Algorithms**

**✅ What is Recursion?**

Recursion is a **programming technique** where a function (or method) calls **itself** to solve a problem.

Rather than solving a big problem all at once, recursion breaks the problem down into **smaller subproblems** of the same type and solves each of them step by step — until it reaches a **base case**, which is a condition that stops the recursion.

**🔁 How It Works**

Every recursive function needs two things:

1. **Base Case** – the condition under which the function stops calling itself.
2. **Recursive Case** – the part where the function calls itself with a simpler or smaller input.

**Real-Life Example: Climbing Stairs**

Suppose you want to climb **n steps** of stairs.

* You know how to climb **1 step**. That’s your base case.
* To reach the **nth step**, you can first reach **(n-1)** and then take 1 step.

So, the process becomes:

climb(n) = climb(n - 1) + 1 step

This is how recursion thinks — solving from the bottom up.

**🧩 Advantages of Recursion**

* Helps simplify complex problems that can be divided into similar subproblems.
* Useful in mathematical calculations (like factorial, Fibonacci, etc.), tree/graph traversal, and dynamic programming.
* Makes code more elegant and readable in many cases.