**🔹 What is Big O Notation?**

**Big O Notation** describes **how the performance (time or space)** of an algorithm **scales** with the size of the input (**n**). It helps you understand how fast (or slow) an algorithm is, especially for **large inputs**.

**🔹 Common Big O Complexities:**

| **Big O** | **Name** | **Example** | **Meaning (for large n)** |
| --- | --- | --- | --- |
| O(1) | Constant Time | Accessing array by index | Always takes same time |
| O(log n) | Logarithmic Time | Binary Search | Time grows slowly |
| O(n) | Linear Time | Simple loop | Time grows in proportion to input |
| O(n log n) | Linearithmic Time | Merge Sort, Quick Sort | Faster than O(n²), slower than O(n) |
| O(n²) | Quadratic Time | Nested loops | Time grows fast |
| O(2ⁿ) | Exponential Time | Recursive Fibonacci | Time grows very quickly |

**🔹 Why is it useful?**

* Helps compare algorithms’ efficiency.
* Predicts performance before coding.
* Guides optimization for large data.