### **What is Big O Notation?**

Big O notation is a way to describe how the runtime (or memory usage) of an algorithm grows as the input size increases.

It helps in:

* Comparing algorithms regardless of hardware
* Predicting performance on large data
* Identifying slow parts of your code

| **Big O** | **Name** | **Example** |
| --- | --- | --- |
| O(1) | Constant time | Accessing an array element |
| O(n) | Linear time | Linear search |
| O(log n) | Logarithmic time | Binary search |
| O(n²) | Quadratic time | Nested loops (ex: bubble sort) |
| O(2ⁿ) | Exponential time | Recursive Fibonacci |

**2. BEST , AVERAGE AND WORST CASE OF SEARCH**

### 

### **Linear Search (searching one-by-one)**

* **Best Case: O(1) -** Element is at the beginning
* **Average Case: O(n) -** Element is somewhere in the middle
* **Worst Case: O(n) -** Element is at the end or not present

### **Binary Search (only works on sorted data)**

* **Best Case**: O(1) — Element is in the middle on the first try
* **Average Case**: O(log n) — We keep dividing the array
* **Worst Case**: O(log n) — Still very fast even for large inputs

**3. Compare the time complexity of linear and binary search algorithms.**

| **Cases** | **Linear Search** | **Binary Search** |
| --- | --- | --- |
| **Best Case** | **O(1) — found at first index** | **O(1) — found at middle** |
| **Average Case** | **O(n/2) —> O(n)** | **O(log n)** |
| **Worst Case** | **O(n)** | **O(log n)** |

### **Which is more suitable for an e-commerce platform (searching by product ID)?**

**Binary Search is more suitable.**

1. Product IDs are sorted — which is perfect for binary search.
2. Binary search is much faster than linear search for large data.
3. E-commerce platforms have thousands or millions of products — binary search handles this efficiently.
4. Search happens frequently on such platforms — speed matters.
5. Fewer comparisons — binary search takes only about 20 steps even for 1 million products.