**What is Big O Notation?**

Big O notation describes how the **runtime or space requirement** of an algorithm **grows as the input size increases**. It helps user analyze algorithm performance **independently of hardware**.

| **Notation** | **Meaning** | **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 | Bubble sort |

**Best, Average, and Worst Case for Search**

| **Search Type** | **Best Case** | **Average Case** | **Worst Case** |
| --- | --- | --- | --- |
| **Linear** | O(1) (first item) | O(n/2) → O(n) | O(n) |
| **Binary** | O(1) (mid item) | O(log n) | O(log n) |

**Analysis: Time Complexity & Suitability**

**🧠 Linear Search-**

* **Time Complexity**: O(n)
* **Pros**:
  + Works on **unsorted data**
  + Simple to implement
* **Cons**:
  + Slow for large datasets

**🧠 Binary Search-**

* **Time Complexity**: O(log n)
* **Pros**:
  + **Much faster** on large sorted datasets
* **Cons**:
  + Requires data to be **sorted**
  + Sorting adds a one-time cost: O(n log n)

**Which is Better for E-commerce Search?**

| **Criteria** | **Linear Search** | **Binary Search** |
| --- | --- | --- |
| Speed on large data | Slow | Fast |
| Requires sorting | No | Yes |
| Real-time filtering | Inefficient | Efficient w/ caching |

**Binary search is more suitable** for an optimized e-commerce platform when data is pre-sorted or cached.  
For small lists or simple filtering, linear search may be sufficient.