### Performance Basics: Big O Notation

**Big O Notation** helps us predict how an algorithm's time or space needs grow with input size (n). It's key for choosing efficient, scalable code.

|  |  |  |
| --- | --- | --- |
| **Search Case** | **Linear Search** | **Binary Search** |
| **Best** | O(1) | O(1) |
| **Average** | O(n) | O(log n) |
| **Worst** | O(n) | O(log n) |

### The Algorithms

We used a simplified product (P) with Id, N (name), and C (category), focusing on searching by N.

* **Linear Search (LSearch):** Checks each item sequentially. Simple but slow on large datasets.
* **Binary Search (BSearch):** Requires a **sorted** list. Divides the search area in half repeatedly. Much faster for larger data.

### The Verdict

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Best Case** | **Average Case** | **Worst Case** |
| **Linear Search** | O(1) | O(n) | O(n) |
| **Binary Search** | O(1) | O(log n) | O(log n) |

**Binary Search is significantly faster** than Linear Search, especially as data grows.

### E-commerce Recommendation

For a real-world platform:

* **Use Binary Search or indexed database searches** for speed.
* **Keep product names sorted** or rely on database indexing.

\