**Exercise 2:**

Explain Big O notation and how it helps in analyzing algorithms.

Big O notation describes the **worst-case time or space complexity** of an algorithm as input size grows.  
It helps compare the **efficiency** of algorithms regardless of hardware or programming language.  
By using Big O, developers can choose algorithms that scale better with large inputs.

Describe the best, average, and worst-case scenarios for search operations.

The **best case** occurs when the target element is found immediately (e.g., first position).  
The **average case** assumes the element is somewhere in the middle or found after several checks.  
The **worst case** happens when the element is not present or found at the last position, requiring maximum comparisons.

Compare the time complexity of linear and binary search algorithms.

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

Discuss which algorithm is more suitable for your platform and why.

For an inventory management system, **Binary Search** is more suitable as it offers faster lookup times (O(log n)) for large datasets.  
It requires the product list to be sorted, which can be maintained efficiently during updates.  
This ensures quick access to product details, improving overall system performance.

Output:

