**Explain why data structures and algorithms are essential in handling large inventories.**

Big O notation expresses the upper bound of an algorithm’s time or space complexity, describing how performance scales with input size. It helps compare algorithms and identify bottlenecks by focusing on growth trends, ignoring constants and lower-order terms.

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

* **Best Case:** Minimum time needed, e.g., first element match in linear search (O(1)).
* **Average Case:** Expected performance over all inputs, typically O(n) for linear search.
* **Worst Case:** Maximum time taken, e.g., last element or not found (O(n)).

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

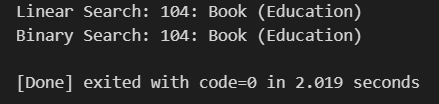
**Linear Search:** O(n) — checks each element one by one.

**Binary Search:** O(log n) — repeatedly divides the sorted array in half to find the target.

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

Binary search is more suitable for large, sorted datasets due to its logarithmic time. However, linear search is better for unsorted or small datasets where sorting isn't worth the overhead.

**Output:**

****