**Question:**

**Explain Big O notation and how it helps in analyzing algorithms.** Describe the best, average, and worst-case scenarios for search operations.

**Answer:**

Big O notation describes the upper bound of an algorithm's runtime or space complexity, giving an idea of the worst-case scenario. It helps in understanding the scalability of an algorithm by providing a way to compare how the runtime grows as the input size increases.

* **Best-case scenario:** The situation where the algorithm performs the fewest possible steps. For example, in a search algorithm, the best case occurs when the target element is found at the beginning of the list.
* **Average-case scenario:** The expected number of steps an algorithm will take, averaged over all possible inputs of the same size.
* **Worst-case scenario:** The maximum number of steps the algorithm will take. For a search algorithm, this is when the target element is at the end of the list or not present.

**Question:**

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

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

**Answer:**

**Time Complexity:**

* **Linear Search:** O(n), where n is the number of products. In the worst case, the search needs to examine every product.
* **Binary Search:** O(log n), where n is the number of products. The search space is halved with each step, making it significantly faster than linear search for large datasets.

**Suitability:**

* **Linear Search** is simple to implement and doesn't require the data to be sorted. It's useful when dealing with small datasets or when data changes frequently, making it impractical to maintain a sorted array.
* **Binary Search** is much more efficient for large datasets but requires the data to be sorted. The overhead of maintaining a sorted list (especially with frequent updates) can offset the benefits in some cases.