Q1. Explain Big O notation and how it helps in analyzing algorithms.

Ans.

Big O notation expresses **how the run-time of an algorithm increases** with the input size (**n**).It helps us compare the **efficiency** of different algorithms **regardless of hardware**.

Q2. Describe the best, average, and worst-case scenarios for search operations.

Ans.

|  |  |  |  |
| --- | --- | --- | --- |
| **Search Algorithm** | **Best Case** | **Average Case** | **Worst Case** |
| Linear Search | O(1) | O(n/2) → O(n) | O(n) |
| Binary Search | O(1) | O(log n) | O(log n) |

Q3. Compare the time complexity of linear and binary search algorithms.

Ans.

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm** | **Time Complexity** | **Space Complexity** | **Suitable When?** |
| Linear Search | O(n) | O(1) | Dataset is **unsorted or small** |
| Binary Search | O(log n) | O(1) | Dataset is **sorted & large** |

Q4. Discuss which algorithm is more suitable for your platform and why.

Ans.

Use **Linear Search** when:

* You don’t want to maintain sorted data.
* Data is small or rarely searched.

Use **Binary Search** when:

* Performance is critical.
* The product list is **sorted alphabetically by product name**.
* Searches happen frequently.