Que1. Explain Big O notation and how it helps in analysing algorithms.

Ans: It is a mathematical notation used to define the complexity or performance of an algorithm. Specifically, it describes the worst-case scenario in terms of time or space used by any algorithms.

This helps us to understand how the algorithm behaves when the input size is increased:

* O(1): Constant time complexity. The operation does not depend on the input size.
* O(n): Linear time complexity. The operation scales linearly with the input size.
* O(log n): Logarithmic time complexity. The operation scales logarithmically with the input size.
* O(n^2): Quadratic time complexity. The operation scales quadratically with the input size.

Que2. Describe the best, average, and worst-case scenarios for search operations.

Ans: Linear Search:

* Best-case: the desired element is present at the zeroth index of the array.
* Average-case: the desired element is present at the middle (or somewhat close to it) of the array.
* Worst-case: the element is present at the last index of the list or not present in the array.

Binary Search:

* Best-case: the element is present at the middle index of the array.
* Average-case: the target element is located somewhere in the array, not necessarily at the middle, but over multiple searches.
* Worst-case: the target element is not present in the array

Que3. Compare the time complexity of linear search and binary search?

Ans: The time complexity of linear search is O(n) as it iterates over every element present in the array until it finds the element.

The time complexity of binary search is O(logn) because in every iteration it divides the array into two equal halves and then search for the values based on the conditions.

Que4. Discuss which algorithm is more suitable for your platform and why?

Ans: For an e-commerce platform where search operations are frequent and the data is large, binary search is generally more suitable because it has less time complexity then linear search.