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

Big O notation is just a way for us to talk about how fast or slow an algorithm is, especially when we're dealing with a lot of data.

It helps us figure out:

* How much time an algorithm might take.
* How much memory it might use.
* And how it behaves when the amount of data gets bigger and bigger.

Examples

O (1) – Constant time

O(n) – Linear time

O (log n) – Logarithmic time

O(n²) – Quadratic time

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

a. Linear Search

Best Case: O (1)-first element matches

Average: O(n)

Worst Case: O(n)-last or no element matches

b. Binary Search

Best Case: O (1)-middle element matches

Average: O (log n)

Worst: O (log n)-element not found or at end

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

a. Linear Search: checks each element one by one

Time complexity: O(n)

-works on unsorted arrays

b. Binary Search: repeatedly divides the search space in half for faster performance which follows divide and conquer method.

Time complexity: O (log n)

-this requires a sorted array

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

For an e-commerce platform, binary search is better because:

* It is much faster.
* It helps users get results quickly, even if there are thousands of products which is major drawback in linear search as it needs to look through the whole list which makes it slower.

But to use binary search, the list of products must be sorted (like by product name or ID).

If the product list keeps changing and is not sorted, then we may temporarily use linear search, but it won’t be fast.