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

Big O notation is a mathematical way to describe the efficiency of an algorithm, especially in terms of time (how fast it runs) or space (how much memory it uses) as the input size grows.

It helps compare algorithms independently of hardware or implementation.

It reveals the worst-case scenario (most common use).

It guides us to choose the right algorithm for large data sets.

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

When analyzing search algorithms (like Linear Search and Binary Search), it’s important to consider how they behave under different situations. This is where best-case, average-case, and worst-case analyses come in.

Let's say we are searching an element in an array and sorted array for Binary Search

**Linear Search**

Searches each element one by one from start to end

Best Case : Element is first in the array

Time Complexity : ( O(1) ) - constant

Average Case: Element is somewhere in the middle

Time Complexity : O(n/2)→ O(n)

Worst Case: Element is not found or is the last one

Time Complexity : O(n)

**Binary Search**

Repeatedly divides the array in half to search

Best Case: Element is exactly in the middle

Time Complexity: O(1)

Average Case: Element could be anywhere

Time Complexity: O(log n)

Worst Case:Element is not found

Time Complexity: O(log n)