### ****ANSWERS OF TWO QUESTIONS Q. Explain Big O notation and how it helps in analyzing algorithms.****

**Ans. Big O notation** describes the upper bound of an algorithm’s time or space complexity in terms of the input size **n**.  
It helps us understand how efficiently an algorithm performs as the input grows, by focusing on the **growth rate** rather than actual execution time.  
This is essential in evaluating algorithms for performance and scalability.

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

#### Ans : ****Linear Search:****

**Best Case:** O(1) – Element is the first item in the array.

**Average Case:** O(n) – Element is somewhere in the middle.

**Worst Case:** O(n) – Element is at the end or not present.

#### ****Binary Search (on a sorted array):****

**Best Case:** O(1) – Element is exactly in the middle.

**Average Case:** O(log n) – Array is halved each iteration.

**Worst Case:** O(log n) – Element not found, continues dividing.