1. **Understand Asymptotic Notation:**

**Big O Notation:** Big O notation is used to describe the complexity of an algorithm in terms of the size of the input (n).

**Time Complexity**: How the runtime of an algorithm grows with the size of the input.

**Space Complexity**: How the memory usage grows with the size of the input.

**Search operation time complexity**:

**Linear Search**:

* 1. Best Case: O(1) (target is the first element).
  2. Average Case: O(n) (target is somewhere in the middle).
  3. Worst Case: O(n) (target is not present, or it's the last element).

**Binary Search**:

* 1. Best Case: O(1) (target is the middle element).
  2. Average Case: O(logn) (logarithmic search tree).
  3. Worst Case: O(logn) (element not found in a sorted array).

**Comparing Time Complexity:**

1. **Linear Search**: O(n) time complexity, as every product in the array might need to be checked.
2. **Binary Search**:

* O(logn) time complexity, as it eliminates half of the search space at each step.
* Requires data to be sorted before searching. Time complexity for sorting O(nlogn).

**Choosing the Best Algorithm for the Platform:**

Binary search becomes efficient when the array is sorted, as sorting the array takes O(nlogn), and subsequent searches are O(logn).