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

Big O notation is used to describe the performance or complexity of an algorithm in terms of the size of the input data (denoted as n). It helps to analyze how the runtime of an algorithm scales as the input size increases. It focuses on the most significant factors, ignoring constant factors and lower-order terms, to provide a high-level understanding of the algorithm's efficiency.

1. **Describe the best, average, and worst-case scenarios for search operations.**
2. **Best Case:** The scenario where the algorithm performs the minimum number of operations. For search operations, this could be finding the item at the first position.

* Linear Search: O(1) if the target is the first element.
* Binary Search: O(1) if the target is the middle element initially.

1. **Average Case:** The expected number of operations for a typical input.

* Linear Search: O(n) where n is the number of elements, as it might need to check half of the elements on average.
* Binary Search: O(log n) due to repeatedly dividing the search interval in half.

1. **Worst Case:** The scenario where the algorithm performs the maximum number of operations. For search operations, this could mean searching through every element in the array or performing the maximum number of comparisons.

* Linear Search: O(n) when the target is not found or is the last element.
* Binary Search: O(log n) when the target is not found, requiring the maximum number of divisions.