**1a.) Big O Notation**

Big O notation describes the upper bound of the time complexity of an algorithm, providing a way to express its efficiency in terms of input size nnn. It helps in analyzing how the performance of an algorithm scales with the size of the input.

* **O(1)**: Constant time complexity, meaning the execution time does not depend on the input size.
* **O(n)**: Linear time complexity, meaning the execution time grows linearly with the input size.
* **O(log n)**: Logarithmic time complexity, meaning the execution time grows logarithmically with the input size.

**1b.) Search Operations**

* **Linear Search**:
  + **Best Case**: O(1) - The element is found in the first position.
  + **Average Case**: O(n) - The element is found after scanning through half of the elements.
  + **Worst Case**: O(n) - The element is found at the last position or not found at all.
* **Binary Search**:
  + **Best Case**: O(1) - The element is found in the middle of the array.
  + **Average Case**: O(log n) - The search space is halved with each step.
  + **Worst Case**: O(log n) - The search space is reduced to one element.

**4a.) Time Complexity**

* **Linear Search**:
  + **Time Complexity**: O(n) in the worst and average cases.
  + **Space Complexity**: O(1) - No additional space is required beyond the input.
* **Binary Search**:
  + **Time Complexity**: O(log n) in the average and worst cases.
  + **Space Complexity**: O(1) - No additional space is required beyond the input.

**4b.) Suitability for the Platform**

**Binary Search**: More suitable for larger datasets where the array is sorted, as it provides faster search times (O(log n)) compared to linear search (O(n)).