Time Complexity

Linked Lists:

1. Access : O(n) - To access an element, you need to traverse the list from the head node to the desired index.
2. Insertion (at beginning): O(1) - Directly add a node at the beginning.
3. Insertion (at end): O(1) - If the tail pointer is maintained, otherwise O(n).
4. Insertion (at middle): O(n) - Need to traverse to the insertion point.
5. Deletion (at beginning): O(1) - Directly remove the head node.
6. Deletion (at end): O(n) - Need to traverse to the node before the last node.
7. Deletion (at middle): O(n) - Need to traverse to the node before the deletion point.
8. Search: O(n) - Need to traverse the list to find the element.

Arrays:

1. Access (by index): O(1) - Direct access by index.
2. Insertion (at beginning): O(n) - Need to shift all elements.
3. Insertion (at end): Amortized O(1) - Usually O(1), but O(n) when resizing is needed.
4. Insertion (at middle): O(n) - Need to shift elements.
5. Deletion (at beginning): O(n) - Need to shift all elements.
6. Deletion (at end): O(1) - Direct removal.
7. Deletion (at middle): O(n) - Need to shift elements.
8. Search: O(n) - Need to traverse the array to find the element.