Time Complexity:

Method	Linked List	Dynamic Array
Insertion at Index	O(n)	O(n) (on average)
Deletion at Index	O(n)	O(n) (on average)
Access by Index	O(n)	O(1)
Append	O(1)	O(1) (amortized)
Prepend	O(1)	O(n) (worst case)
Search	O(n)	O(n)
Rotation	O(n)	O(n)
Reversal	O(n)	O(n)
Merging	O(m + n)	O(m + n)
Interleaving	O(m + n)	O(m + n)
Middle Element	O(n)	O(1)
Index of Element	O(n)	O(n)

Space Complexity:

- Linked List: O(n)
- Dynamic Array: O(n)

Advantages and Disadvantages:

Linked Lists:

- Advantages:
    - Dynamic size: No need to specify initial size.
    - Efficient insertion and deletion at arbitrary positions.
- Disadvantages:
    - Poor random access: Accessing elements by index is O(n).
    - Higher space overhead: Requires additional memory for pointers

Dynamic Arrays:

- Advantages:
    - Efficient random access: Accessing elements by index is O(1).
    - Better cache locality: Elements are contiguous in memory.
- Disadvantages:

- Costly insertions and deletions: May require resizing and shifting elements.
- Fixed capacity: May need to allocate more memory when capacity is reached.

Conclusion:

Both linked lists and dynamic arrays have their strengths and weaknesses, making them suitable for different use cases.

- Use linked lists when:
  - Dynamic size is essential.
  - Frequent insertions and deletions at arbitrary positions are expected.
  - Random access is not a primary concern.
- Use dynamic arrays when:
  - Random access is important.
  - Predominantly read-heavy operations are involved.
  - Overhead from pointers is a concern.

In summary, the choice between linked lists and dynamic arrays depends on the specific requirements of the application and the trade-offs between time complexity, space complexity, and desired operations.