1.Time Complexity

Linked Lists:

Insert at index: O(n)

Delete at index: O(n)

Get size: O(n)

Is empty: O(1)

Reverse: O(n)

Append: O(n) or O(1)

Prepend: O(1)

Merge: O(n)

Get middle: O(n)

Index of: O(n)

Split at index: O(n)

Dynamic Arrays:

Insert at index: O(n)

Delete at index: O(n)

Get size: O(1)

Is empty: O(1)

Reverse: O(n)

Append: Amortized O(1)

Prepend: O(n)

Get middle: O(1)

Index of: O(n)

Split at index: O(n)

Space complexity :

The space complexity of most methods is O(1) for both linked lists and dynamic arrays, since they usually require just a certain amount of extra space for variables, pointers, etc. But sometimes, dynamic arrays need to be resized, which adds extra space complexity (O(n)) to the resizing process.

Advantages and disadvantages :

Linked Lists:

- Advantages:
  - Effective insertion and deletion in O(1) time at the start (prepend).
  - There's no need to resize, thus there's no extra space complexity brought on by resizing.
- Disadvantages:
  - Inefficient random access; O(n) time complexity results from having to traverse from the head in order to access elements by index.
  - Additional RAM used to store references and hyperlinks.

Dynamic Arrays (DynamicArray):

- Advantages:
  - Effective random access; O(1) time can be spent retrieving elements via an index.
  - Without frequent resizing, dynamic resizing enables effective append operations (amortized O(1) time).
- Disadvantages:
  - Shifting elements cause expensive insertion and deletion in the center of the array, with O(n) time complexity (worst case).
  - Sometimes, resizing procedures (amortized O(n)) might cause performance to deteriorate.

Overall Comparison:

- When memory allocation needs to be optimized or when there are frequent insertions and deletions made at the beginning of the list, linked lists are a good option.
- When efficient random access is required, dynamic arrays are the better option, especially if the data structure's size is known or can be approximated beforehand.

- Depending on the particular needs of the application, such as memory limitations and the frequency of various actions, linked lists or dynamic arrays should be used.
- With this comparison, developers may make well-informed decisions based on their unique use cases and performance requirements by highlighting the trade-offs between dynamic arrays and linked lists.