Comparison of Linked Lists and Dynamic Arrays:

Time complexity :

Insert at index(Linked Lists) :O(n)

Insert at index(Dynamic array) :O(n)

Delete at index(Linked Lists): O(n)

Delete at index(Dynamic array): O(n)

Get size(Linked Lists): ): O(n)

Get size(Dynamic array) :O(1)

Is empty(Linked Lists): O(1)

Is empty(Dynamic array) : O(1)

Rotate right(Linked Lists): O(n)

Rotate right(Dynamic array) : O(n)

Reverse(Linked Lists): O(n)

Reverse(Dynamic array) : O(n)

Append(Linked Lists):O(n)

Append(Dynamic array) :O(1)

Prepend(Linked Lists):O(1)

Prepend(Dynamic array) : O(1)

Merge(Linked Lists): O(n)

Merge(Dynamic array) :O(n+m)

Interleave(Linked Lists): O(n+m)

Interleave(Dynamic array) : O(n+m)

Get middle(Linked Lists): O(n)

Get middle(Dynamic array) :O(1)

Index of(Linked Lists): O(n)

Index of(Dynamic array) : O(n)

Split at index(Linked Lists) O(n)

Split at index(Dynamic array) : O(n)

Space complexity:

Linked Lists:

* Each node needs extra space for storing a pointer or a reference.
* Space complexity is O(n) for n elements, plus an additional O(n) for pointers.

Dynamic arrays:

\*Requires a contiguous memory block.

\*Space complexity is O(n) additional space, as arrays often double in size when their capacity is exceeded.

\*Resizing may require up to O(n) additional space, as arrays often double in size when their capacity is exceeded.

Advantages and Disadvantages:

Linked lists:

Advantages :

Random Access: Offers quick access(O(1)) to elements.

Memory Efficiency: Suitable for stable or gradually growing array sizes.

Compact Storage: Stores elements contiguously, enhancing cache performance.

Disadvantages:

Fixed Size: Initial allocation requirement, resizing can be costly (O(n)).

Insertions/Deletions: Requires shifting elements, resulting in O(n) time complexity.

Space Wastage: Potential for unused allocated memory, leading to space inefficiency.

Comparison Report:

In computer science, both linked lists and dynamic arrays serve as foundational data structures for storing collections of elements. Despite their distinct features, they offer various strengths and limitations. This comparative analysis explores their time complexity, and practical advantages and disadvantages.

Time complexity:

Linked Lists excel in swift insertion and deletion operations, especially at the beginning, with O(1) complexity. However, accessing elements sequentially entails O(n) time. Conversely, dynamic arrays provide O(1) access but face O(n) complexity for insertions and deletions due to element shifting.