# Report on Comparison of Linked Lists and Dynamic Arrays

Introduction :

Linked lists and dynamic arrays are fundamental data structures in computer science. Each of it has it’s unique characteristics, advantages, and disadvantages. Understanding their time and space complexities, as well as their pros and cons, is crucial for selecting the appropriate data structure.

Time Complexity Comparison :

1. Linked Lists :

1. Access (get)

Time Complexity : O(n)

Explanation : To access an element, traversal from the head node to the target node is required, which takes linear time in the worst case.

2. Insertion

- At Head: O(1)

- At Tail: O(n) if a singly linked list without a tail pointer, O(1) with a tail pointer

- At Middle: O(n)

- Explanation : Inserting at the head is constant time as it involves only adjusting pointers. Inserting at the tail requires traversal unless a tail pointer is maintained. Inserting in the middle requires traversal to the insertion point.

3. Deletion

- Time Complexity : O(n)

- Explanation : To delete a node, traversal to find the node (or its predecessor) is needed, which takes linear time in the worst case.

4. Search

- Time Complexity : O(n)

- Explanation : Searching for a specific value requires traversal through the list until the value is found or the end is reached.

2. Dynamic Arrays

1. Access (get)

- Time Complexity : O(1)

- Explanation : Direct access via index is possible in constant time due to contiguous memory allocation.

2. Insertion

- At End (Amortized) : O(1)

- At Beginning/Middle : O(n)

- Explanation : Inserting at the end is usually constant time, but occasionally O(n) when resizing is needed (amortized to O(1)). Inserting at the beginning or middle requires shifting elements, resulting in linear time complexity.

3. Deletion

- At End : O(1)

- At Beginning/Middle : O(n)

- Explanation : Deleting the last element is constant time. Deleting an element from the beginning or middle requires shifting elements to fill the gap, resulting in linear time complexity.

4. Search

- Time Complexity : O(n)

- Explanation : Linear search is required for an unsorted dynamic array, taking linear time in the worst case.

Space Complexity Comparison

Linked Lists

- Space Complexity : O(n)

- Explanation : Each element requires space for the data and a pointer to the next element. Therefore, the total space is linear in the number of elements.

Dynamic Arrays

- Space Complexity : O(n)

- Explanation : Space is allocated for the elements, and an additional amount for resizing is reserved. The reserved space typically results in some wasted memory but remains linear with respect to the number of elements.

Advantages and Disadvantages

Linked Lists

Advantages :

1. Dynamic Size : Easily grows and shrinks as elements are added or removed.

2. Efficient Insertions/Deletions : Adding or removing elements is efficient if the pointer to the node is known, especially at the head or tail.

Disadvantages :

1. Memory Overhead : Extra memory is required for storing pointers.

2. Sequential Access : Accessing elements requires traversal, resulting in linear time complexity.

3. Cache Performance : Poor cache performance due to non-contiguous memory allocation.

Dynamic Arrays

Advantages :

1. Fast Access : Direct access to elements via index in constant time.

2. Cache Performance : Better cache performance due to contiguous memory allocation.

3. Amortized Insertion at End : Efficient insertion at the end (amortized constant time).

Disadvantages :

1. Resizing Overhead : Resizing the array when capacity is reached involves copying all elements, leading to potential performance hits.

2. Insertion/Deletion Overhead : Inefficient insertions and deletions at arbitrary positions due to the need to shift elements.

Conclusion :

The choice between linked lists and dynamic arrays depends on the specific requirements of the application:

-The “Linked Lists” are used when frequent insertions and deletions are required, especially at the beginning or middle of the list, and where excess memory is not a critical concern.

- The “Dynamic Arrays” are used when fast access to elements and good cache performance are needed, and insertions/deletions are primarily at the end of the array.

Understanding the difference between them helps to identify the suitable data structure for applicastion.