DSA Assignment-1

Compare the following aspects of linked lists and dynamic arrays:-

1)Time complexity of each method
Answer:-
Access/Search:
Linked List: O(n) - In worst-case scenarios, you have to traverse the list from the beginning to find an element.
Dynamic Array: O(1) - Accessing elements by index is constant time since arrays offer direct access to elements using their indices.

Insertion/Deletion:
Linked List: O(1) for insertion/deletion at the beginning (if the head is maintained), O(n) for insertion/deletion at the end or middle (as you may need to traverse to the specific position).
Dynamic Array: O(n) for insertion/deletion at arbitrary positions (as it may require shifting elements), though O(1) for insertion at the end if space is available.

2) Space complexity of each method
Answer:-
Linked List: O(n) - Each element in the linked list requires extra space for the pointer to the next node.
Dynamic Array: O(n) - The array needs to allocate contiguous memory space for its elements. However, it may also have some additional memory reserved for potential future growth.

3) Advantages and disadvantages of each data structure
Answer:-
Advantages and Disadvantages:
Linked Lists:
Advantages:-Efficient for insertion/deletion at the beginning (if head is maintained).
Dynamic memory allocation: Linked lists can grow or shrink in size dynamically during runtime.

No need for contiguous memory allocation: Linked lists can efficiently use scattered memory blocks.
Disadvantages:-Inefficient for random access due to traversal requirement.
Extra memory overhead for pointers.
Cache unfriendly: Since elements are scattered in memory, cache utilization may not be optimal.

Dynamic Arrays:
Advantages:-Efficient random access: Elements are stored in contiguous memory, facilitating efficient random access.
Cache friendly: Due to contiguous memory allocation, elements are usually stored closer to each other, improving cache utilization.
Better memory locality can lead to better performance in many scenarios.
Disadvantages:-Costly insertion/deletion in the middle: As it may require shifting elements.
Fixed initial memory allocation: Expanding beyond the initial allocation may require expensive resizing operations.
Not suitable for large deletions or insertions at arbitrary positions due to shifting overhead.