Compare the following aspects of linked lists and dynamic arrays:

o Time complexity of each method

o Space complexity of each method

o Advantages and disadvantages of each data structure

**Time Complexity:**

**Linked Lists:**

* Access: O(n)
* Search: O(n)
* Insertion (at beginning): O(1)
* Insertion (at end, if tail is known): O(1)
* Deletion (at beginning): O(1)
* Deletion (at end, if tail is known): O(n)

Dynamic array

* Access: O(1)
* Search: O(n)
* Insertion (at end, if space available): O(1) amortized
* Insertion (at beginning or middle, shifting required): O(n)
* Deletion (at end): O(1)
* Deletion (in middle, shifting required): O(n)

**Space Complexity:**

**Linked Lists:**

* Space complexity for n elements: O(n) (each node consumes extra space)

**Dynamic Arrays:**

* Space complexity for n elements: O(n) (but with potential for overallocation)

**Advantages and Disadvantages:**

**Linked Lists:**

* Advantages:
  + Dynamic size: Easy to expand or shrink.
  + Efficient insertion and deletion at any position (with pointers to the nodes).
* Disadvantages:
  + Poor performance for random access.
  + More memory overhead due to pointers.

**Dynamic Arrays:**

* Advantages:
  + Efficient random access (constant time).
  + Better cache locality which can result in faster traversal.
* Disadvantages:
  + Costly insertion and deletion in the middle due to shifting.
  + Fixed capacity, may need resizing which can be costly.

**Choosing Between Them:**

* Use linked lists when:
  + Frequent insertion and deletion operations are expected.
  + Memory allocation and deallocation time is not a concern.
  + There is no need for random access.
* Use dynamic arrays when:
  + Frequent random access is required.
  + The number of elements is known in advance or can be estimated to prevent frequent resizing.
  + Memory overhead needs to be minimized.