Linked Lists vs. Dynamic Arrays; Comparison Report

Operation	Dynamic Array	Doubly Linked List
Inserts an element at the specified index	O(n)	O(n)
Inserts an element at the first	O(n)	O(1)
Inserts an element at the last	O(1)	O(1)
Deletes the element at the specified index	O(n)	O(n)
Removes the element from the first	O(n)	O(1)
Removes the element from the last	O(1)	O(1)
Returns the size	O(1)	O(1)
Checks if empty	O(1)	O(1)
Rotate by k positions to the right	O(n)	O(n)
Reverse	O(n)	O(n)
Prepend to beginning	O(n)	O(n)
Merge	O(n)	O(1)
Interleave	O(n)	O(n)
Return middle element	O(1)	O(n)
Index of first occurrence	O(n)	O(n)
Split	O(n)	O(n)
Resize (custom factor)	O(n)	-

Advantages and Disadvantages of Each Data Structure

Doubly Linked List

Advantages

- Dynamic size with efficient insertions and deletions at both ends.
- Can be traversed both forward and backward.
- No need for a contiguous memory allocation.

Disadvantages

- Sequential access only, O(n) time for accessing elements by index.
- Extra memory usage per node due to two pointers.
- Slightly higher overhead than singly linked lists due to additional pointers.

Dynamic Array

Advantages

- Random access by index (O(1)).
- Caching due to contiguous memory allocation.
- O(1) time complexity for append operations.

Disadvantages

- Fixed size until resized, which can lead to O(n) operations.
- Inefficient insertions and deletions at arbitrary positions (O(n)).
- Potentially significant memory overhead due to resizing (unused capacity).
- Need for contiguous memory allocation.

Conclusion

Linked Lists (singly and doubly) are preferable for applications where frequent insertions and deletions at the beginning or middle are required, and where space overhead due to pointers is not a major concern and contiguous memory is not available.

Dynamic Arrays are more suitable for applications where random access is needed and where append operations are frequent. They offer better cache performance and are simpler in terms of space management since they don't involve pointer overhead.