Linked lists are a fundamental data structure that consists of a series of nodes. Each node contains data and a reference (or pointer) to the next node in the list. Unlike arrays, linked lists allow for dynamic resizing and efficient insertion/deletion of elements at any position.

There are two main types of linked lists: singly linked lists and doubly linked lists. Singly linked lists only have a pointer to the next node, while doubly linked lists have pointers to both the previous and next nodes.

Here's a summary of the key operations on linked lists:

- **Insertion:** Adding a new node to the list. In singly linked lists, insertion at the beginning or end is O(1), but random insertion is O(n). In doubly linked lists, insertion at any position is O(1).
- **Deletion:** Removing a node from the list. Deletion can be O(1) in some cases (e.g., removing the head/tail in a singly linked list with head/tail pointers), but it can also be O(n) in other cases (e.g., removing a node in the middle of a singly linked list).
- **Searching:** Finding a specific node in the list. Searching is always O(n) in both singly and doubly linked lists.
- **Traversal:** Visiting each node in the list. Traversal is straightforward and takes O(n) time in both singly and doubly linked lists.

Overall, linked lists are a versatile data structure that can be useful in various applications. Their dynamic nature and efficient insertion/deletion capabilities make them a good choice when frequent modifications to the data are needed.