LINKED LIST

A linked list is the most sought-after data structure when it comes to handling dynamic data elements. A linked list consists of a data element known as a node. And each node consists of two fields: one field has data, and in the second field, the node has an address that keeps a reference to the next node.

- A linked list is a linear data structure that stores a collection of data elements dynamically.
- Nodes represent those data elements, and links or pointers connect each node.
- Each node consists of two fields, the information stored in a linked list and a pointer that stores the address of its next node.
- The last node contains null in its second field because it will point to no node.
- A linked list can grow and shrink its size, as per the requirement.
- It does not waste memory space.Ā

Representation of a Linked List

This representation of a linked list depicts that each node consists of two fields. The first field consists of data, and the second field consists of pointers that point to another node.

Types of Linked Lists

The linked list mainly has three types, they are:

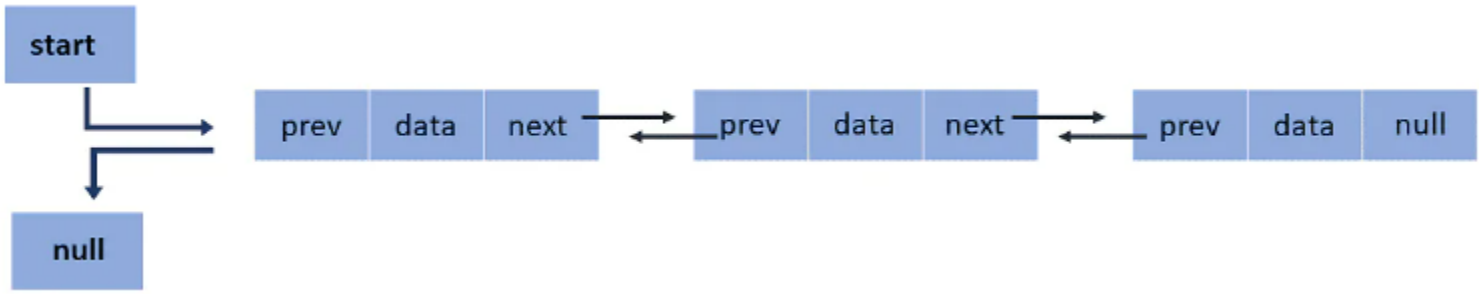
1. Singly Linked List
2. Doubly Linked List
3. Circular Linked List

Singly Linked List

A singly linked list is the most common type of linked list. Each node has data and an address field that contains a reference to the next node.

Doubly Linked List

There are two pointer storage blocks in the doubly linked list. The first pointer block in each node stores the address of the previous node. Hence, in the doubly linked inventory, there are three fields that are the previous pointers, that contain a reference to the previous node. Then there is the data, and last you have the next pointer, which points to the next node. Thus, you can go in both directions (backward and forward).



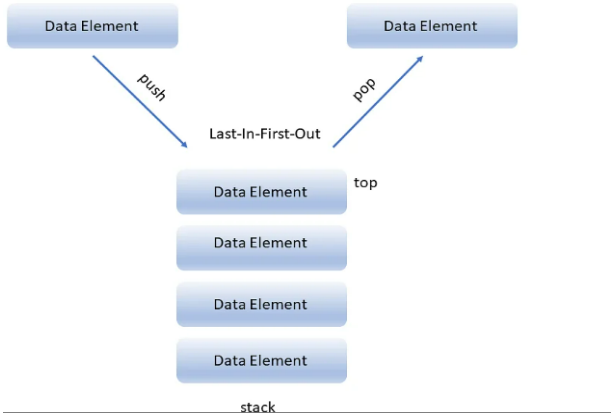
Circular Linked List

The circular linked list is extremely similar to the singly linked list. The only difference is that the last node is connected with the first node, forming a circular loop in the circular linked list.

Stack:

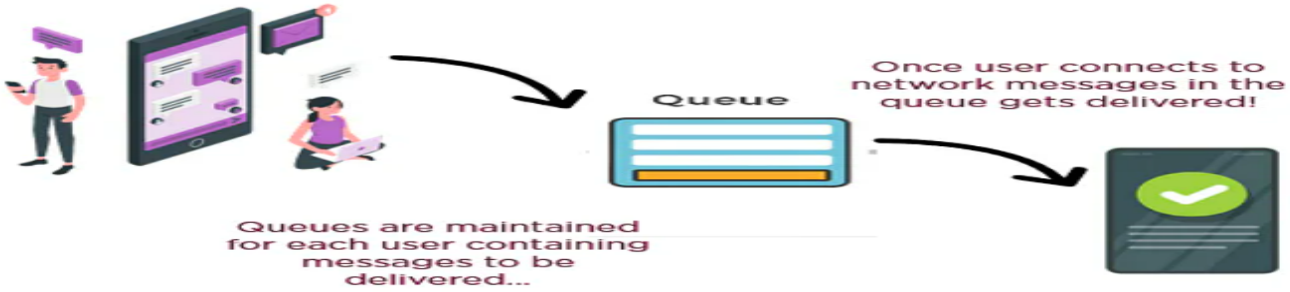
Stacks in Data Structures is a linear type of data structure that follows the LIFO (Last-In-First-Out) principle and allows insertion and deletion operations from one end of the stack data structure, that is top.The stack data structure is a linear data structure that accompanies a principle known as LIFO (Last In First Out) or FILO (First In Last Out). Real-life examples of a stack are a deck of cards, piles of books, piles of money, and many more.

Stack Representation in Data Structures



Queue:

Queue in data structures is a linear collection of different data types which follow a specific order while performing various operations. It can only be modified by the addition of data entities at one end or the removal of data entities at another.



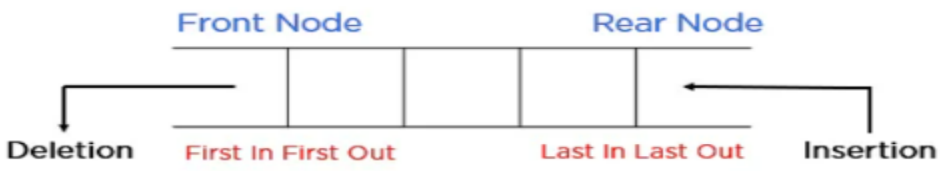
Queue Representation

Before dealing with the representation of a queue, examine the real-life example of the queue to understand it better. The movie ticket counter is an excellent example of a queue where the customer that came first will be served first. Also, the barricades of the movie ticket counter stop in-between disruption to attain different operations at different

ends.

The queue in the data structure acts the same as the movie ticket counter. Both the ends of this abstract data structure remain open. Further, the insertion and deletion processes also operate analogously to the wait-up line for tickets.

The following diagram tries to explain queue representation as a data structure



Queue Representation

: