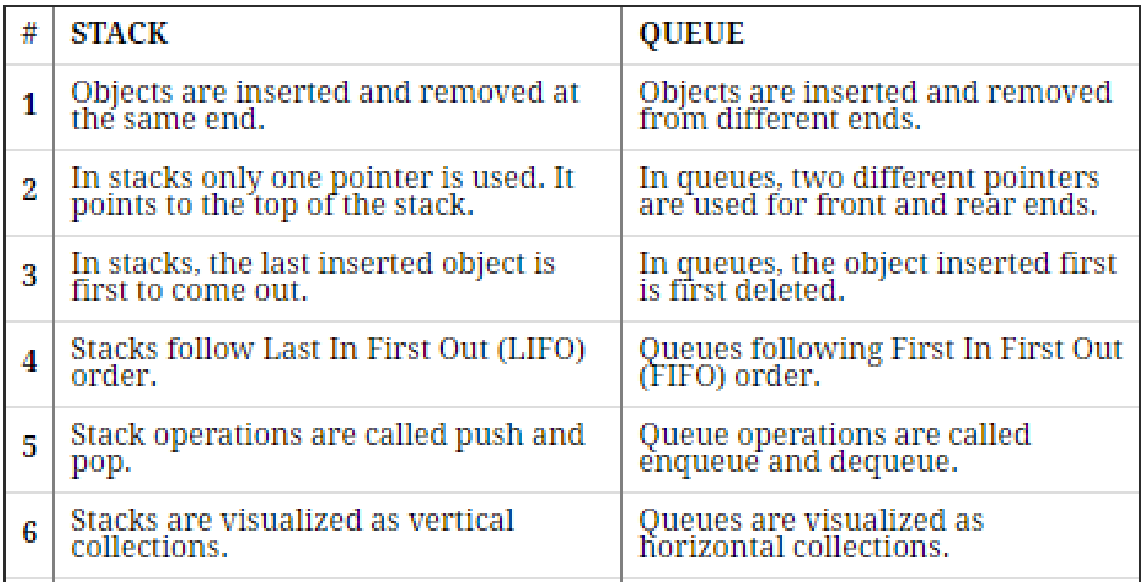
DATA STRUCTURES

UNIT 2 & 3

<u>4 - MARKS</u>

1. Differentiate stack and queue?



#	STACK	QUEUE
1	Objects are inserted and removed at the same end.	Objects are inserted and removed from different ends.
2	In stacks only one pointer is used. It points to the top of the stack.	In queues, two different pointers are used for front and rear ends.
3	In stacks, the last inserted object is first to come out.	In queues, the object inserted first is first deleted.
4	Stacks follow Last In First Out (LIFO) order.	Queues following First In First Out (FIFO) order.
5	Stack operations are called push and pop.	Queue operations are called enqueue and dequeue.
6	Stacks are visualized as vertical collections.	Queues are visualized as horizontal collections.

2. Explain about the types of Queue ADT.

Queue is a linear data structure that follows the First-In-First-Out (FIFO) principle. It means that the elements added first to the queue are the first to be removed. In a queue, the elements are inserted at one end, which is called the rear end and the elements are removed at the other end, which is called the front end.

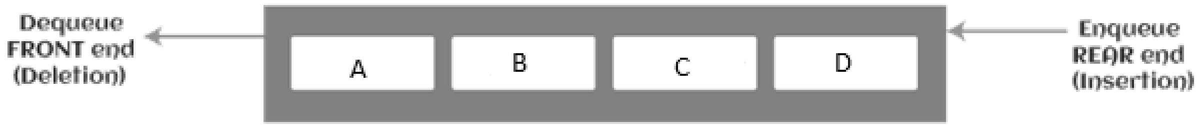
Types of Queue :

1. Simple Queue or Linear Queue
2. Circular Queue
3. Priority Queue
4. Double Ended Queue (or Deque)

1. Simple Queue or Linear Queue

A simple queue is a linear data structure that follows the First-In-First-Out (FIFO) principle. It allows elements to be added to the end of the queue (rear) and removed from the front. The two main operations performed on a simple queue are enqueue (add an element to the rear end) and dequeue (remove an element from the front end).

A simple queue can be implemented using an array or a linked list and is used in many applications, such as managing tasks in an operating system, implementing communication protocols, job scheduling etc…

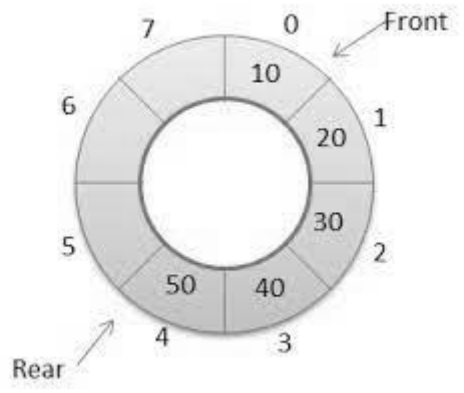


2. Circular Queue

A circular queue is a type of queue data structure that uses a circular buffer to store elements. In a circular queue, the front and rear pointers "wrap around" the buffer after reaching the end, allowing the reuse of space that has been freed up by dequeue operations.

The drawback that occurs in a linear queue is overcome by using the circular queue. If the empty space is available in a circular queue, the new element can be added in an empty space by simply incrementing the value of rear. The main advantage of using the circular queue is better memory utilization.

The main operations performed on a circular queue are enqueue and dequeue, similar to a simple queue. However, in a circular queue, when an element is enqueued and the rear pointer reaches the end of the buffer, it wraps around to the beginning, allowing the reuse of memory. Similarly, when an element is dequeued and the front pointer reaches the end of the buffer, it also wraps around to the beginning.



3. Priority Queue

In a priority queue each element is assigned a priority and the elements are served in the order of priority. This means that the highest priority element is served first and the lowest priority element is served last. Suppose some elements occur with the same priority, they will be arranged according to the FIFO principle. The main advantage of using a priority queue is that it enables efficient processing of elements based on their priority. This makes it useful in many scheduling processes in an operating system, implementing shortest-path algorithms in graph theory, or implementing event-driven simulations.

There are two types of priority queue -

Ascending priority queue

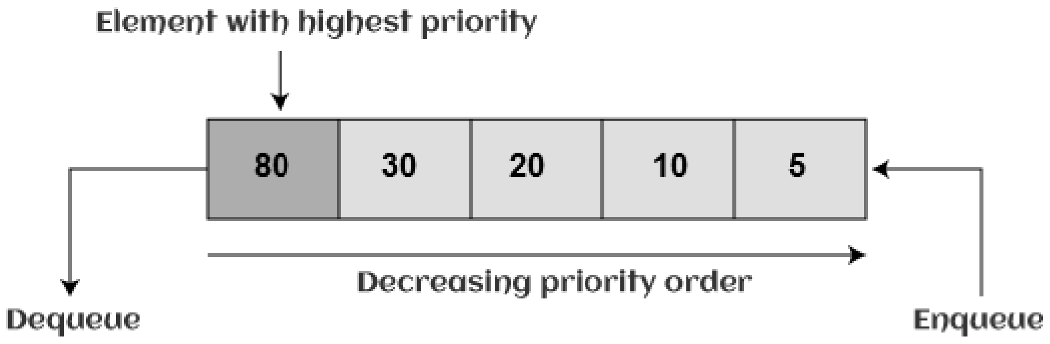
In ascending priority queue, elements can be inserted in arbitrary order, but only smallest element can be deleted first.

Suppose an array with elements 7, 5, and 3 in the same order, so, insertion can be done with the same sequence, but the order of deleting the elements is 3, 5, 7.

Descending priority queue

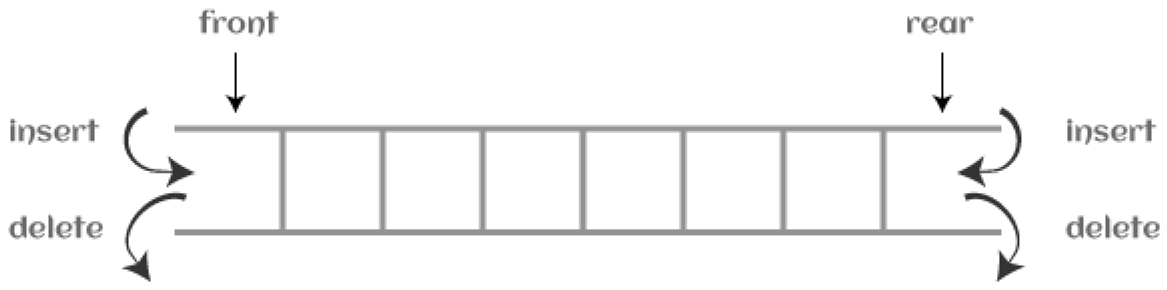
In descending priority queue, elements can be inserted in arbitrary order, but only the largest element can be deleted first.

Suppose an array with elements 7, 3, and 5 in the same order, so, insertion can be done with the same sequence, but the order of deleting the elements is 7, 5, 3.



4. Deque (Double Ended Queue)

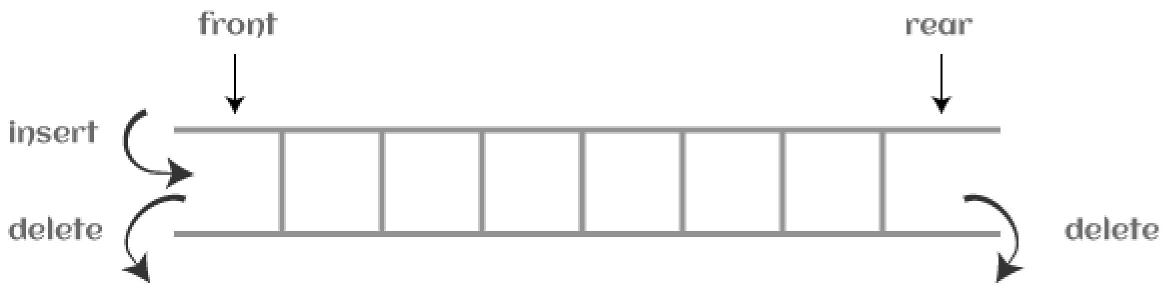
Deque (Double Ended Queue) is a linear data structure where elements can be inserted and removed from both the end of queue whether the front or the rear end of the queue. It means that we can insert and delete elements from both front and rear ends of the queue. Some applications of deque include searching algorithms, computer memory management and graph algorithms.



double ended queue

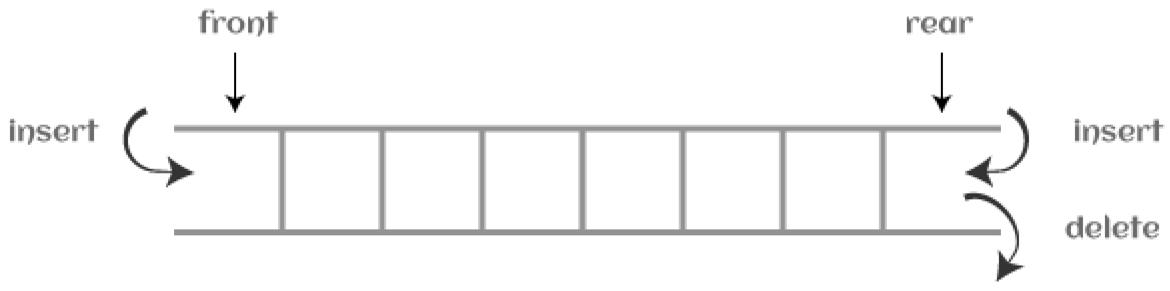
There are two types of deque –

Input restricted deque - As the name implies, in input restricted queue, insertion operation can be performed at only one end, while deletion can be performed from both ends.



input restricted double ended queue

Output restricted deque - As the name implies, in output restricted queue, deletion operation can be performed at only one end, while insertion can be performed from both ends.



Output restricted double ended queue

3. Compare Singly linked list and doubly linked list.

Singly Linked List	Doubly Linked List
It is a collection of nodes and each node is having one data field and next link field	It is a collection of nodes and each node is having one data field, one previous link field and one next link field
A Singly Linked List occupies less memory than the Doubly Linked List, as it has only 2 fields.	A Doubly Linked List occupies more memory than the Singly Linked List, as it has 3 fields.
The elements can be accessed using next link	The elements can be accessed using both previous link as well as next link
Accessing elements in a Singly Linked List is less efficient as only forward traversal is possible.	Accessing elements in a Doubly Linked List is more efficient as both forward and backward traversal is possible.
No extra field is required hence, Node takes less memory in Singly Linked List.	One field is required to store previous Link. Hence, node takes extra memory in Doubly Linked List.
The time complexity of inserting or deleting a node at a given position (if the pointer to that position is given) in a singly linked list is O(n).	The time complexity of inserting or deleting a node at a given position (if the pointer to that position is given) in a doubly linked list is O(1).
Singly linked list is preferred when we have memory limitation (we can't use much memory) and searching is not required.	Doubly linked list is preferred when we don't have memory limitation and searching is required (we need to perform search operation on the linked list).

4. What are the advantages of circular linked list?

Any node can be a starting point.

We can traverse the whole list by starting from any point. We just need to stop when the first visited node is visited again.

Useful for implementation of queue. Unlike this implementation, we don't need to maintain two pointers for front and rear if we use circular linked list.

We can maintain a pointer to the last inserted node and front can always be obtained as next of last.

Circular lists are useful in applications to repeatedly go around the list. For example, when multiple applications are running on a PC, it is common for the operating system to put the running applications on a list and then to cycle through them, giving each of them a slice of time to execute, and then making them wait while the CPU is given to another application.

It is convenient for the operating system to use a circular list so that when it reaches the end of the list it can cycle around to the front of the list.

5. List all the applications of Queues.

Checking strings of a language
Queuing theory simulation
Input / Output buffers
Graph searching
Jobs sent to a printer
Line in a ticket counter

6. How do you test an empty queue? Explain

To test for an empty queue, we have to check whether REAR=HEAD where REAR is a pointer pointing to the last node in a queue and HEAD is a pointer that points to the dummy header.

In the case of array implementation of queue, the condition to be checked for an empty queue is REAR=max-1, front=rear+1.

7. Give brief note on doubly linked list operations

The basic operations carried out in a doubly linked list include:

a. Creation of a list
b. Insertion of a node
c. Deletion of a node
d. Modification of a node
e. Traversal of the list

8. What are the disadvantages of linked list?

Searching a particular element in a list is difficult and time consuming.

A linked list will use more storage space than an array to store the same number of elements.