1.Use examples to explain the sorting algorithms.
ANS:

A Sorting Algorithm is used to rearrange a given array or list elements according to a comparison operator on the elements. The comparison operator is used to decide the new order of element in the respective data structure. For example: **The below list of characters is sorted in increasing order of their ASCII values**.

2.What are the benefits of stacks?
ANS:

Advantages of Stack

- Stack is easy to learn and implement

for beginners.

- Stacks are used to solving problems that work on recursion.

- It allows you to control how memory is allocated and deallocated.

3.What is the difference between a Stack and a queue?
ANS:
The primary difference between Stack and Queue Data Structures is that Stack follows LIFO while Queue follows FIFO data structure type. LIFO refers to Last In First Out. It means that when we put data in a Stack, it processes the last entry first. Conversely, FIFO refers to First In First Out. It means that when we put

data in Queue, it processes the first entry first. Let us understand further the differences between them.

Stack Data Structure:-

You can refer to Stack as a linear form of data structure. In this, a user can delete and insert elements from only one side of a list. It is known as the **top**. The stack data structure implements and follows the Last In, First Out (LIFO) principle. It implies that the element that is inserted last comes out first.

The process of inserting an element in the stack is known as the **push** operation. Here, the deletion of elements is known as the **pop** operation. A user can feasibly keep track of the last function/element of the list in a stack using a pointer (top).

Queue Data Structure:-

You can also refer to Queue as a linear form of data structure. In this, a user can insert elements from only one side of the list. It is known as the rear. Also, one can delete these elements from another side- known as the front. This type of data structure implements and follows the First In, First Out (FIFO) principle. It implies that the first element that inserts into a list will come out first.

The process of inserting an element into a queue is known as the **enqueue** operation. Here, the process of deleting an element is known as the **dequeue** operation. A user can always hold two pointers in a queue. The **front** pointer points to the first inserted element that is still in the list. The second pointer is the **rear** one that points to the last inserted element in the list.

4.What are the different forms of queues?
ANS:

1.Simple Queue

A simple queue is the most basic queue. In this queue, **the enqueue operation takes place at the rear, while the dequeue operation takes place at the front:**

Its applications are process scheduling, disk scheduling, memory management, IO buffer, pipes, call center phone systems, and interrupt handling.

2.Circular Queue

A circular queue **permits better memory utilization than a simple queue** when the queue has a fixed size.

In this queue, the last node points to the first node and creates a circular connection. Thus, it allows us to insert an item at the first node of the queue when the last node is full and the first node is free.

It’s also called a ring buffer:

It’s used to switch on and off the lights of the traffic signal systems. Apart from that, it can be also used in place of a simple queue in all the applications mentioned above.

3.Priority Queue

A priority queue is a special kind of queue in which each item has a predefined priority of service. In this queue, the enqueue operation takes place at the rear in the order of arrival of the items, while the dequeue operation takes place at the front based on the priority of the items.

That is to say that **an item with a high priority will be dequeued before an item with a low priority**.

In the case, when two or more items have the same priority, then they'll be dequeued in the order of their arrival. Hence, it may or may not strictly follow the FIFO rule:

It’s used in interrupt handling, Prim’s algorithm, Dijkstra’s algorithm, A* search algorithm, heap sort, and Huffman code generation.

4. Double-Ended Queue (Deque)

A deque is also a special type of queue. In this queue, the **enqueue and dequeue operations take place at both front and rear**. That means, we can insert an item at both the ends and can remove an item from both the ends. Thus, it may or may not adhere to the FIFO order:

It’s used to save browsing history, perform undo operations, implement A-Steal job scheduling algorithm, or implement a stack or implement a simple queue.

Further, it has two special cases: **input-restricted deque** and **output-restricted deque**.

In the first case, the enqueue operation takes place only at the rear, but the dequeue operation takes place at both rear and front:

An input-restricted queue is useful when we need to remove an item from the rear of the queue.

In the second case, the enqueue operation takes place at both rear and front, but the dequeue operation takes place only at the front:

An output-restricted queue is handy when we need to insert an item at the front of the queue.

5. Conclusion

In this article, we've learned four kinds of queues with their applications.

If you have a few years of experience in Computer Science or research, and you're interested in sharing that experience with the community, have a look at our **Contribution Guidelines**.

5.Why should I use stack or queue Data Structures instead of array or lists, and when should i use them?
ANS:
The stack can contain elements of different data types. The array contains elements of the same data type. There are limited number of operations can be performed on a stack: push, pop, peek, etc. It is rich in methods or operations that can be perform on it like sorting, traversing, reverse, push, pop, etc.

Queues are widely used as **waiting lists for a single shared resource** like printer, disk, CPU. Queues are used in asynchronous transfer of data (where data is not being transferred at the same rate between two processes) for eg. pipes, file IO, sockets.

6.What is the significance of Stack being a recursive data structure?

ANS:

Programming involves recursive thinking, and **it can help us to write shorter and more efficient code when used appropriately**. While iterative functions can usually do the same job, recursive functions are simpler to read and understand. Recursive functions are especially powerful in traversing tree structures.