**Queue Using Array Data Structure**

**Definition**

A queue is a linear data structure that follows the First In, First Out (FIFO) principle. This means that the first element added to the queue will be the first one to be removed. A queue can be implemented using various data structures, including arrays.

**Operations**

1. **Enqueue**: Add an element to the end of the queue.
2. **Dequeue**: Remove and return the element from the front of the queue.
3. **Front/Peek**: Return the front element without removing it.
4. **isEmpty**: Check if the queue is empty.
5. **isFull**: Check if the queue is full (relevant for fixed-size queues).

**Pros and Cons**

**Pros:**

* **Simplicity**: Easy to understand and implement.
* **Direct Memory Access**: As arrays provide direct access to elements, queue operations can be performed quickly.

**Cons:**

* **Fixed Size**: The size of the queue must be defined initially. This can lead to wasted space if the queue is not fully utilized or queue overflow if the queue exceeds its initial size.
* **Inefficient Dequeue**: If implemented naively, dequeuing an element from the front of the array can be inefficient as it requires shifting all subsequent elements from one position to the left.

**Applications**

* **Task Scheduling**: Used in operating systems for scheduling tasks (e.g., CPU scheduling).
* **Buffer Management**: Used in buffering data streams (e.g., IO Buffers).
* **Breadth-First Search**: Utilized in graph algorithms for traversing or searching through graph data structures.
* **Print Queue Management**: Used in managing print jobs in printers.
* **Handling Requests**: Used in servers to handle incoming requests in the order they are received.