Queue – Queue Operations

1. Introduction

Queue operations are the basic actions performed on a queue data structure.

All operations follow the **FIFO (First In, First Out)** principle and are performed using two pointers:

- **Front** – points to the first element
- **Rear** – points to the last element

Understanding these operations is essential before implementing queues using arrays or linked lists.

---

2. Main Queue Operations

A queue supports the following core operations:

1. **Enqueue**
2. **Dequeue**
3. **Front / Peek**
4. **isEmpty**
5. **isFull** (for array-based queue)

---

3. Enqueue Operation

What is Enqueue?

Enqueue is the operation of **adding an element to the rear of the queue**.

Logic (Plain English)

1. Check if the queue is full
2. If full, report queue overflow

3. Otherwise:
    - Increase the rear pointer
    - Insert the element at the rear position

Example

```
Enqueue 10
Enqueue 20
Enqueue 30
```

Queue:

```
Front → 10  20  30 ← Rear
```

---

4. Dequeue Operation

What is Dequeue?

Dequeue is the operation of **removing an element from the front of the queue**.

Logic (Plain English)

1. Check if the queue is empty
2. If empty, report queue underflow
3. Otherwise:
    - Remove the element at the front
    - Increase the front pointer

Example

```
Dequeue → removes 10
```

Queue becomes:

```
Front → 20  30 ← Rear
```

---

5. Front / Peek Operation

What is Peek?

Peek returns the **front element of the queue without removing it**.

Logic (Plain English)

1. Check if the queue is empty
2. If not empty, return the front element
3. Queue remains unchanged

---

6. isEmpty Operation

What is isEmpty?

isEmpty checks whether the queue contains **no elements**.

Condition

```
Front > Rear   OR   Front == -1
```

Used before dequeue and peek operations.

---

7. isFull Operation

What is isFull?

isFull checks whether the queue is **completely filled** (array implementation).

Condition

```
Rear == size − 1
```

Used before enqueue operations.

---

8. Queue Overflow

What is Queue Overflow?

Queue overflow occurs when:

- Trying to enqueue an element into a full queue

This happens in array-based queue implementations.

---

9. Queue Underflow

What is Queue Underflow?

Queue underflow occurs when:

- Trying to dequeue an element from an empty queue

This is an error condition.

---

10. Time Complexity of Queue Operations

Operation	Time Complexity
Enqueue	O(1)
Dequeue	O(1)
Peek	O(1)
isEmpty	O(1)
isFull	O(1)

All queue operations are executed in **constant time**.

---

11. Advantages of Queue Operations

- Efficient task processing
- Maintains order of execution

- Simple logic
- Ideal for scheduling systems

12. Limitations of Queue Operations

- No random access
- Fixed size in simple array queues
- Space wastage in linear queues

13. Real-World Applications

- CPU scheduling
- Printer job management
- Network packet handling
- Breadth-First Search (BFS)
- Customer service systems

14. Summary

- Queue operations follow FIFO principle
- Enqueue adds at rear
- Dequeue removes from front
- Peek views front element
- All operations run in O(1) time