Queue

What is a Queue?

A **Queue** is a data structure where:

First In, First Out (FIFO)

The **first item that goes in** is the **first item that comes out**.

Just like a real-life waiting line.

Real-Time Examples of a Queue

---

1. People waiting in a line (Queue at ATM / Bus Stop)

This is the best example.

Person1 → Person2 → Person3 → Person4
(front) (rear)

- Person1 came first → gets service first
- New people join at **rear**
- Service happens from **front**

This is exactly how a Queue works in programming.

---

2. Printer Queue

When you print multiple documents:

Doc1 → Doc2 → Doc3 → Doc4

- Doc1 prints first
- Doc4 prints last

First submitted job gets printed first (**FIFO**).

---

3. Call Center Helpline

Customers are placed in a queue:

Caller 1 → Caller 2 → Caller 3

- Caller1 gets agent first
- New callers join at **rear**

FIFO processing.

---

4. Ticket Booking Counter

People line up to buy tickets.

- First person gets ticket first
- Next person moves forward
- New people join the end of line

Exactly a queue behavior.

---

5. CPU Task Scheduling (First-Come, First-Served)

Some operating systems schedule tasks in a queue:

Task1 → Task2 → Task3

CPU processes tasks in the same order.

FIFO task execution.

---

6. Order Processing System

In online shopping warehouses:

Orders come like:

Order101 → Order102 → Order103

First order placed is packed first.

Why use a Queue?

- Processes items **in order received**
- Prevents skipping
- Simple & predictable
- Useful in scheduling, buffering, simulations, and more

Simple Definition

A **Queue** is a data structure that processes elements in the order they arrive — **First In, First Out (FIFO)**.

Types of Queues

1. Linear Queue (Simple Queue)

✔ FIFO → First In, First Out

Elements get added at **rear**, removed from **front**.

10 → 20 → 30 → 40

□ Real-life example:

- Line at a **bus stop**
- People first in line get the bus first

---

2. Circular Queue

A queue where the **last position connects back to the first**, forming a circle.

Useful when you want to **reuse empty spaces**.

rear wraps to front (circular)

Real-life example:

- **Round-robin scheduling**
- **Ride queue** that continuously loops

---

3. Priority Queue

Elements are removed based on **priority**, not order of arrival.

Higher priority = served first

Emergency patient → High priority
Normal patient → Low priority

Real-life example:

- **Hospital emergency room**
- **Airport runway** (small plane waits, emergency plane lands first)

---

4. Double-End Queue (Deque)

(Pronounced as "Deck")

You can **insert and delete** at **both front and rear**.

Real-life example:

- Editing history in browsers
  (you can move backwards AND forwards)

---

5. Input Restricted Queue

Only **rear insertion** allowed.
Deletion allowed at **both ends**.

Best for:

- Fast removals from both sides but controlled input.

---

6. Output Restricted Queue

Insertion allowed at **both ends**.
Deletion only from **front**.

Best for:

- Special scheduling where removal order must remain consistent.

---

7. Double-Ended Priority Queue

Supports:

- Highest priority removal
- Lowest priority removal

Used in:

- Systems needing both **min** and **max** priority operations

---

8. Circular Deque

A deque but implemented in circular form for efficient memory use.

Used in:

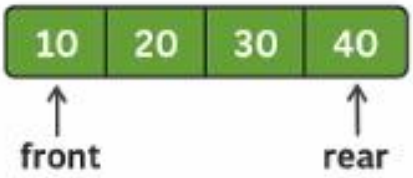
- CPU scheduling
- Task rotations

Summary Table

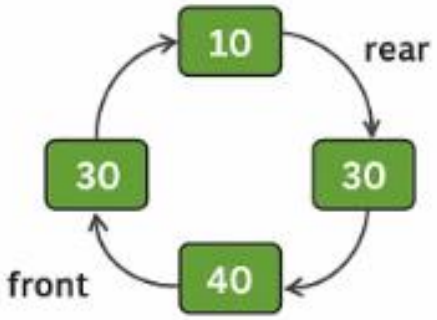
Queue Type	Can Insert	Can Remove	Real Example
Linear Queue	Rear	Front	Bus stop line
Circular Queue	Rear (wraps)	Front	Round-robin CPU
Priority Queue	Based on priority	Based on priority	Hospital ER
Deque	Both ends	Both ends	Browser history
Input Restricted Queue	Rear only	Both ends	Special scheduling
Output Restricted Queue	Both ends	Front only	Job queues
Double-Ended Priority Queue	Based on priority	Both ends	Stock market systems

---

Linear Queue

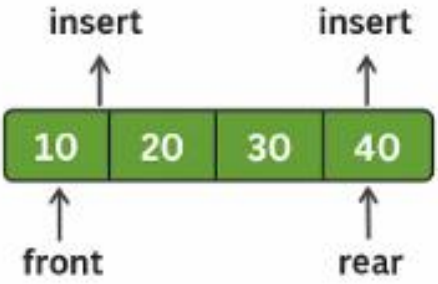


Circular Queue



Priority Queue

Deque



Priority Queue

Deque