1. Definition of Queue

A **queue** is a **linear data structure** that follows the **FIFO (First In First Out)** principle.
This means the **element added first will be removed first**, just like people standing in a line.

Real-world Analogy:

- Think of a queue at a ticket counter: the person who comes first is served first.

Queue Operations:

Operation	Description
enqueue()	Add element to the back (rear)
dequeue()	Remove element from the front
peek()	View the front element
isEmpty()	Check if the queue is empty

2. Types of Queues

Type	Description
Simple Queue	Basic FIFO queue
Circular Queue	Connects rear to front (saves space)
Deque (Double-Ended Queue)	Can add/remove from both ends
Priority Queue	Elements removed based on priority, not position

3. Where Queue Can Be Applied

Real-world and Programming Applications:

Application	Description
Task scheduling	OS process scheduling, print queue
Breadth-First Search (BFS)	Tree or graph traversal
Level order traversal in trees	Nodes at the same level
Producer-Consumer problems	Thread-safe data exchange

Sliding Window Problems	Maintain current state in fixed-size window
Web requests handling	Load balancing of incoming requests
Cache design (LRU)	Queue + HashMap

4. How to Apply Queue
In Java (Using Queue interface with LinkedList)

```
Queue<Integer> queue = new LinkedList<>();

queue.offer(10);   // enqueue
queue.offer(20);
System.out.println(queue.peek());  // prints 10
System.out.println(queue.poll());  // removes and returns 10
```

5. How to Identify Queue-Based Problems
You can apply Queue when:

Pattern	Clue
BFS or Level Order Traversal	When you're exploring things **level by level** or **layer by layer**
First Come First Serve logic	Problems needing **order preservation**
Sliding Window / Moving Window	When you need **elements in order** of entry/removal in a fixed range
Producer-Consumer	Thread-safe data pipelines
Flood Fill, Shortest Path	When you need to **visit neighbors in order**

6. Benefits of Queue

Benefit	Description

Order Preservation	Maintains insertion order
Efficient for BFS/Level Traversal	Naturally fits graph/tree traversal
Decouples Producers and Consumers	Async processing in multi-threaded environments
Used in Buffers	Like IO buffers, print queues, etc.
Helps avoid recursion in BFS	Prevents stack overflow in wide graphs

7. Common Queue-Based Problems

Problem	Description
Level Order Traversal (Tree)	Traverse tree level by level
BFS in Graph	Find shortest path or traversal
Sliding Window Maximum	Maintain max of window using Deque
Rotting Oranges	BFS to spread infection
Course Schedule (Topo Sort)	Use queue in Kahn's algorithm
LRU Cache	Queue + HashMap for recent usage

8. Conclusion

If you need to process elements in the same order they arrive, or level-wise (BFS), or within a fixed-size window, consider using a Queue.

#	Problem Title	Type	Difficulty	Link
1	Binary Tree Level Order Traversal	Tree BFS	Medium	🔗 Link

2	Implement Queue using Stacks	Design	Easy	🔗 Link
3	Perfect Squares	BFS	Medium	🔗 Link
4	Sliding Window Maximum	Monotonic Deque	Hard	🔗 Link
5	Rotten Oranges	BFS Grid	Medium	🔗 Link
6	Course Schedule	BFS (Topo Sort)	Medium	🔗 Link
7	Walls and Gates	BFS Grid	Medium	🔗 Link
8	Number of Recent Calls	Queue Design	Easy	🔗 Link
9	Dota2 Senate	Queue Simulation	Medium	🔗 Link
10	Open the Lock	BFS	Medium	🔗 Link
11	Design Circular Queue	Design	Medium	🔗 Link
12	Moving Average from Data Stream	Queue Sliding Window	Easy	🔗 Link
13	Reveal Cards In Increasing	Simulation (Deque)	Medium	🔗 Link

	Order			
14	Find the Safest Path in a Grid	BFS + Priority Queue	Medium	🔗 Link
15	Time Needed to Inform All Employees	BFS Tree	Medium	🔗 Link