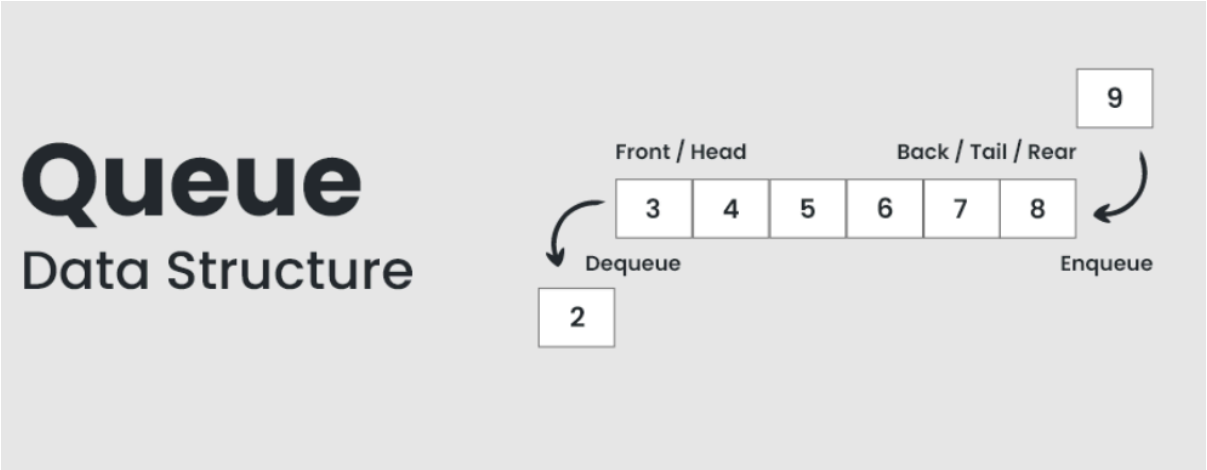
Queue Data Structure

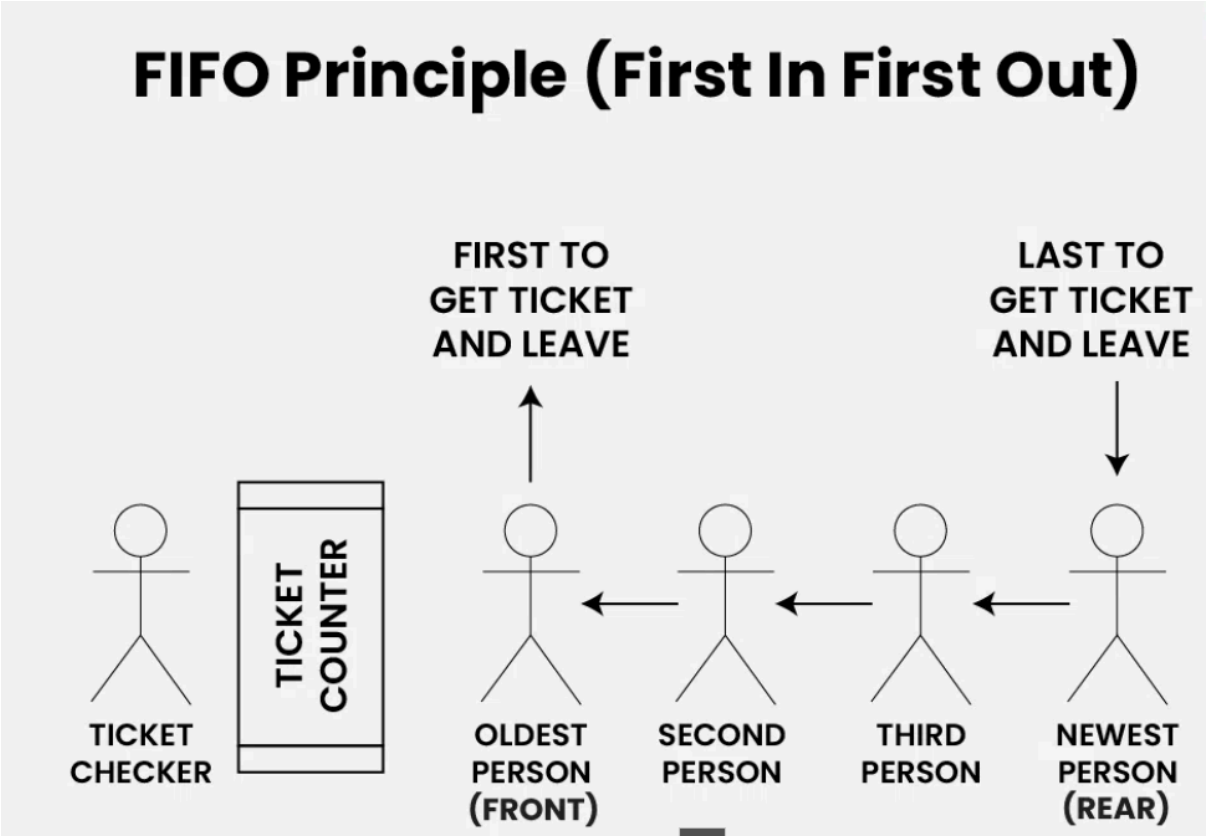
A **Queue Data Structure** is a fundamental concept in computer science used for storing and managing data in a specific order.

- It follows the principle of "**First in, First out" (FIFO)**, where the first element added to the queue is the first one to be removed.
- It is used as a buffer in computer systems where we have speed mismatch between two devices that communicate with each other. For example, CPU and keyboard and two devices in a network
- Queue is also used in Operating System algorithms like CPU Scheduling and Memory Management, and many standard algorithms like Breadth First Search of Graph, Level Order Traversal of a Tree



Introduction to Queue Data Structure

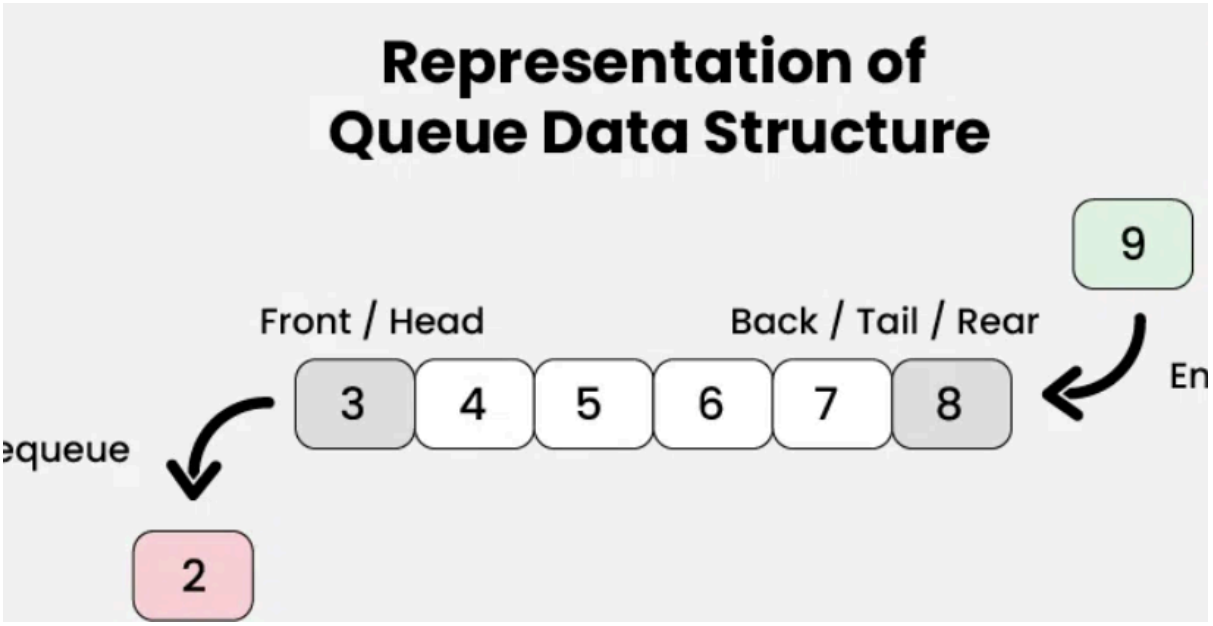
Queue is a linear data structure that follows **FIFO (First In First Out) Principle**, so the first element inserted is the first to be popped out.
FIFO Principle in Queue:
FIFO Principle states that the first element added to the Queue will be the first one to be removed or processed. So, Queue is like a line of people waiting to purchase tickets, where the first person in line is the first person served. (i.e. First Come First Serve).



Basic Terminologies of Queue

- **Front:** Position of the entry in a queue ready to be served, that is, the first entry that will be removed from the queue, is called the **front** of the queue. It is also referred as the **head** of the queue.
- **Rear:** Position of the last entry in the queue, that is, the one most recently added, is called the **rear** of the queue. It is also referred as the **tail** of the queue.
- **Size:** Size refers to the **current** number of elements in the queue.
- **Capacity:** Capacity refers to the **maximum** number of elements the queue can hold.

Representation of Queue



Queue Operations

1. **Enqueue**: Adds an element to the end (rear) of the queue. If the queue is full, an overflow error occurs.
2. **Dequeue**: Removes the element from the front of the queue. If the queue is empty, an underflow error occurs.
3. **Peek/Front**: Returns the element at the front without removing it.
4. **Size**: Returns the number of elements in the queue.
5. **isEmpty**: Returns `true` if the queue is empty, otherwise `false`.
6. **isFull**: Returns `true` if the queue is full, otherwise `false`.

Applications of Queue Data Structure

Application of queue is common. In a computer system, there may be queues of tasks waiting for the printer, for access to disk storage, or even in a time-sharing system, for use of the CPU. Within a single program, there may be multiple requests to be kept in a queue, or one task may create other tasks, which must be done in turn by keeping them in a queue.

- A Queue is always used as a buffer when we have a speed mismatch between a producer and consumer. For example keyboard and CPU.
- Queue can be used where we have a single resource and multiple consumers like a single CPU and multiple processes.
- In a network, a queue is used in devices such as a router/switch and mail queue.
- Queue can be used in various algorithm techniques like Breadth First Search, Topological Sort, etc.