Data Structures

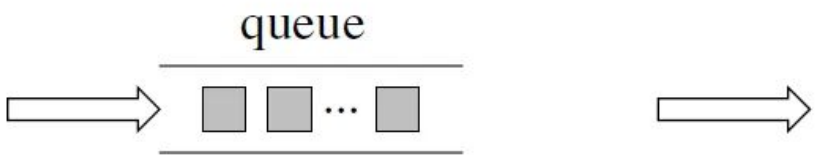
Queue Data Structure

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

What is a queue?

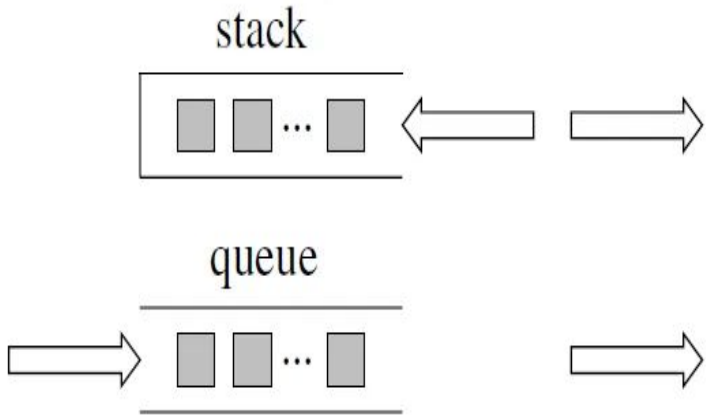
- In English, queue is a **line of items** awaiting their turn. Examples include:
  - A queue of customers in a restaurant waiting to be served
  - A queue of patients in a hospital
  - A queue of customers who have called a helpline and are waiting for a response
- Let's say we are in restaurant and there are 5 people awaiting
  - Who should be served first? The first person who arrived!
  - We call this: **FIFO = First in, First out**



Img src

FIFO vs FILO

- Keep the difference between stack processing and queue processing in mind



Img src

Queue ADT

- We need to design a data structure that follows FIFO
- **enqueue**(value): **Add** to the **end** (**rear**) of the queue
- **dequeue**(): **Delete** from the **front** of the queue
- Useful functionalities:
  - isEmpty(), isFull(), clear(), frontQueue(), rearQueue()
- Any implementation that satisfies FIFO = Queue

Img src

Queue Tracing

- Let’s trace the following operations
- enqueue(5)

Front

5				

Queue Tracing

- Let’s trace the following operations
- enqueue(5)
- enqueue(7)

Front

5	7			

Queue Tracing

- Let’s trace the following operations
- enqueue(5)
- enqueue(7)
- enqueue(8)

Front

5	7	8		

Queue Tracing

- Let’s trace the following operations
- enqueue(5)
- enqueue(7)
- enqueue(8)
- dequeue() ⇒ 5

Front

7	8			

Queue Tracing

- Let’s trace the following operations
- enqueue(5)
- enqueue(7)
- enqueue(8)
- dequeue() ⇒ 5
- enqueue(6)

Front

7	8	6		

Queue Tracing

- Let’s trace the following operations
- enqueue(5)
- enqueue(7)
- enqueue(8)
- dequeue() ⇒ 5
- enqueue(6)
- dequeue() ⇒ 7

Front

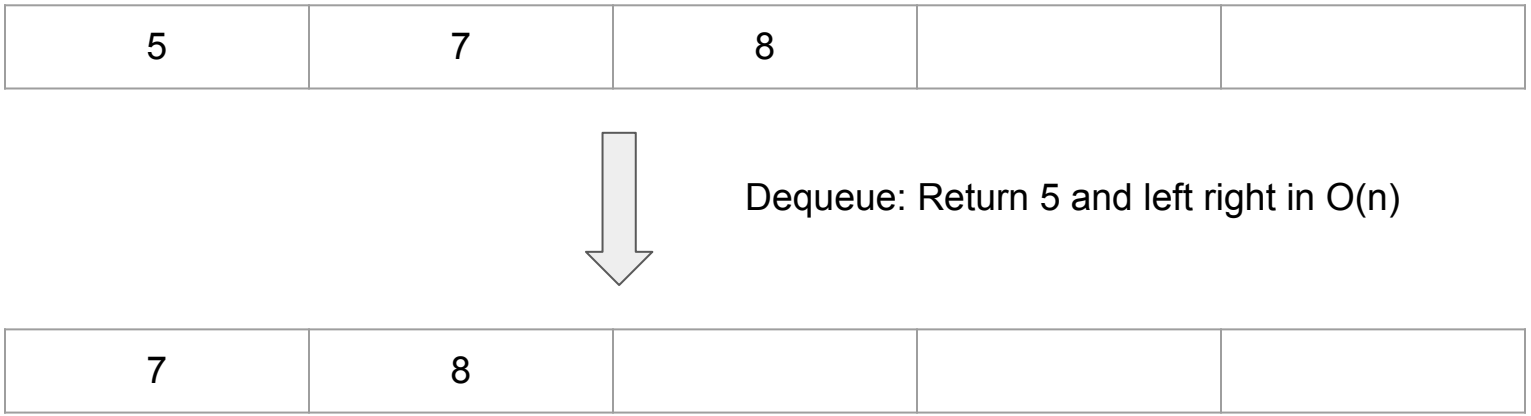
8	6			

Implementation

- Array-based Queue
  - Enqueuing elements is trivial, but dequeuing is very challenging!
  - Let's see a couple of approaches (direct implementation)
- Linked-list-based Queue

Array-based: Shift approach

- Assume we added 3 elements so far in the array: {5, 7, 8}
- Now, after we dequeue 5, this index = 0 is empty
- One way is to just shift the whole array to the left ⇒ **O(n) dequeue**!



Array-based: Front-Rear approach

- We'll use two **indices**; front and rear; which represent the start and end of the array respectively
  - When we **enqueue** an element, we add it to the rear
  - When dequeue an element, we **shift the front** index to the right⇒ **O(1)**
- Let’s add {5, 7, 8}

Array-based: Front-Rear approach

- Dequeue (front+=1)

Array-based: Front-Rear approach

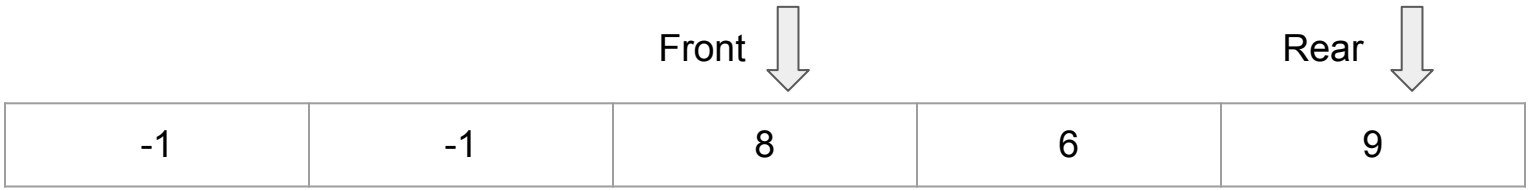
- Enqueue 6

Array-based: Front-Rear approach

- Enqueue 9

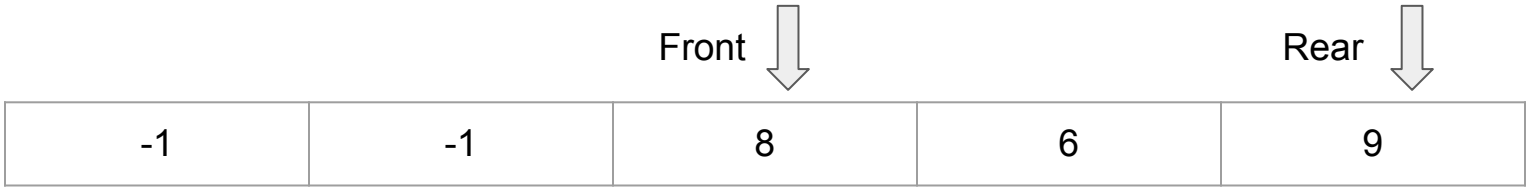
Array-based: Front-Rear approach

- Dequeue ⇒ 7



Array-based: Front-Rear approach

- Enqueue 3: ERROR Queue is full!
- However, there are empty slots at the beginning!
  - This is a critical **drawback** of this approach
  - How can we solve this? Think about it for 15 minutes!



"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."