K.R. MANGALAM UNIVERSITY

THE COMPLETE WORLD OF EDUCATION

Data Structure

ENCS205

School of Engineering & Technology (SOET)
K.R. MANGALAM University

UNIT-2

Session 27: Queue ADT and Operations

Recap

A stack is a simple data structure that operates on the Last In, First Out (LIFO) principle, resembling a stack of objects. It's efficient, with O(1) time complexity for push and pop operations.

Advantages:

- Easy implementation
- Efficient push and pop operations
- Effective memory management

Common Operations:

Push: Add item to the top

Pop: Remove item from the top

Peek: View top item without removal

isEmpty, Size

Recap

Applications:

- Function calls
- Expression evaluation
- Backtracking
- Undo mechanisms
- Browser history

Limitations:

- Fixed capacity (in traditional implementations)
- Limited access (only top item accessible)

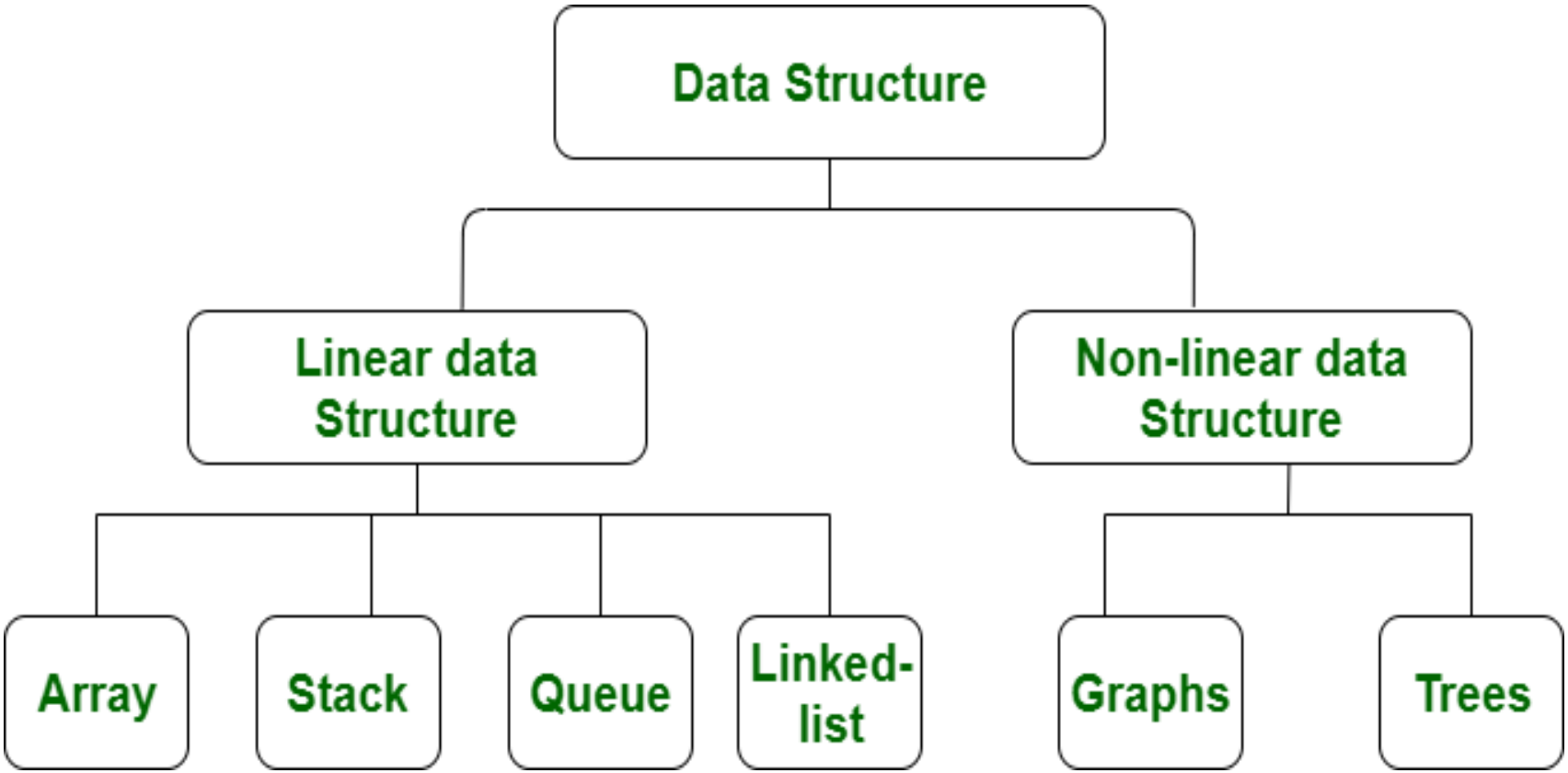
Session 27 Outlook

- Basic Introduction of Queues
- Types of Queues
- Operations on Queues
- Representation of Queues

Objective

- Students should be able to recall Queues terminology.
- Student should be able to Understand Queues structure vs. other data structures. Differentiate implementations and expression representations.
- Apply queue structures for process scheduling problems.
- Students should Evaluate and optimize queue performance

Data Structure



https://www.geeksforgeeks.org/data-structure-meaning/

Structure of Queues?

Fig 1 : QUEUE

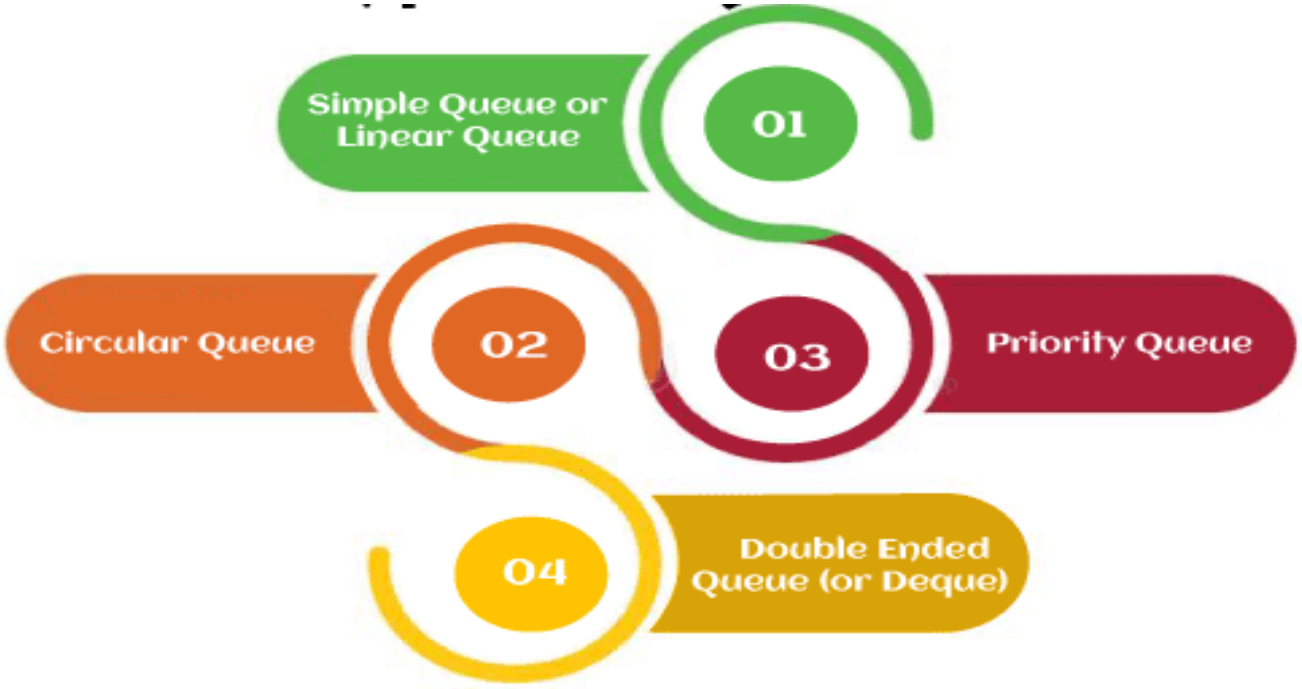
https://www.scaler.com/topics/data-structures/queue-in-data-structure/

Queues?

- An ordered list which enables insert operations to be performed at one end called **REAR** and delete operations to be performed at another end called **FRONT**.
- Referred to be as First In First Out list.

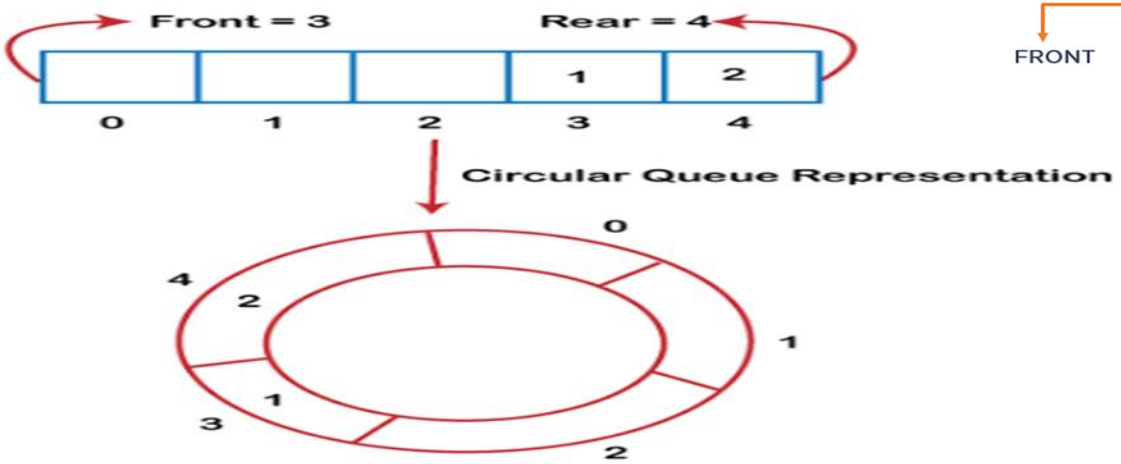
https://www.javatpoint.com/data-structure-queue

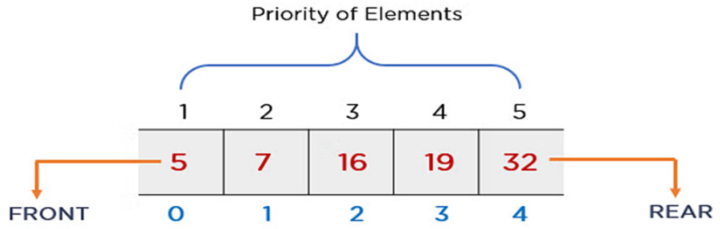
Types

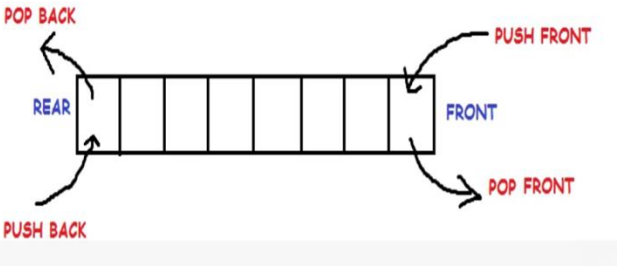


https://www.javatpoint.com/data-structure-queue

Types







https://www.javatpint.com/data-structure-queue

Operations

Enqueue (Insert): Adds an element to the rear of the queue.

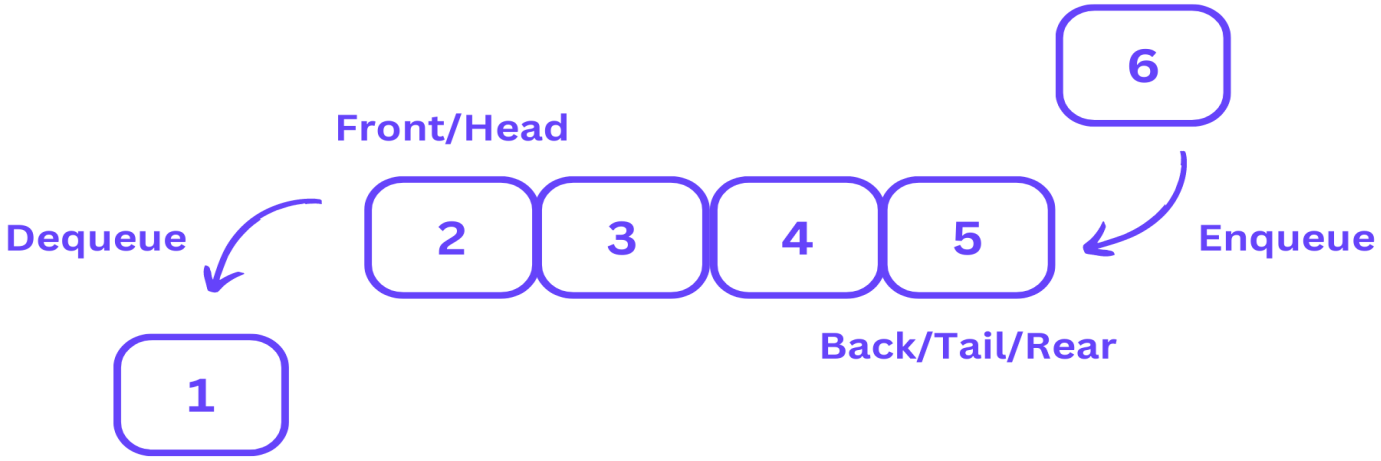
Dequeue (Delete): Removes and returns the element from the front of the queue.

Peek: Returns the element at the front of the queue without removing it.

isEmpty: Checks if the queue is empty.

isFull: Checks if the queue is full.

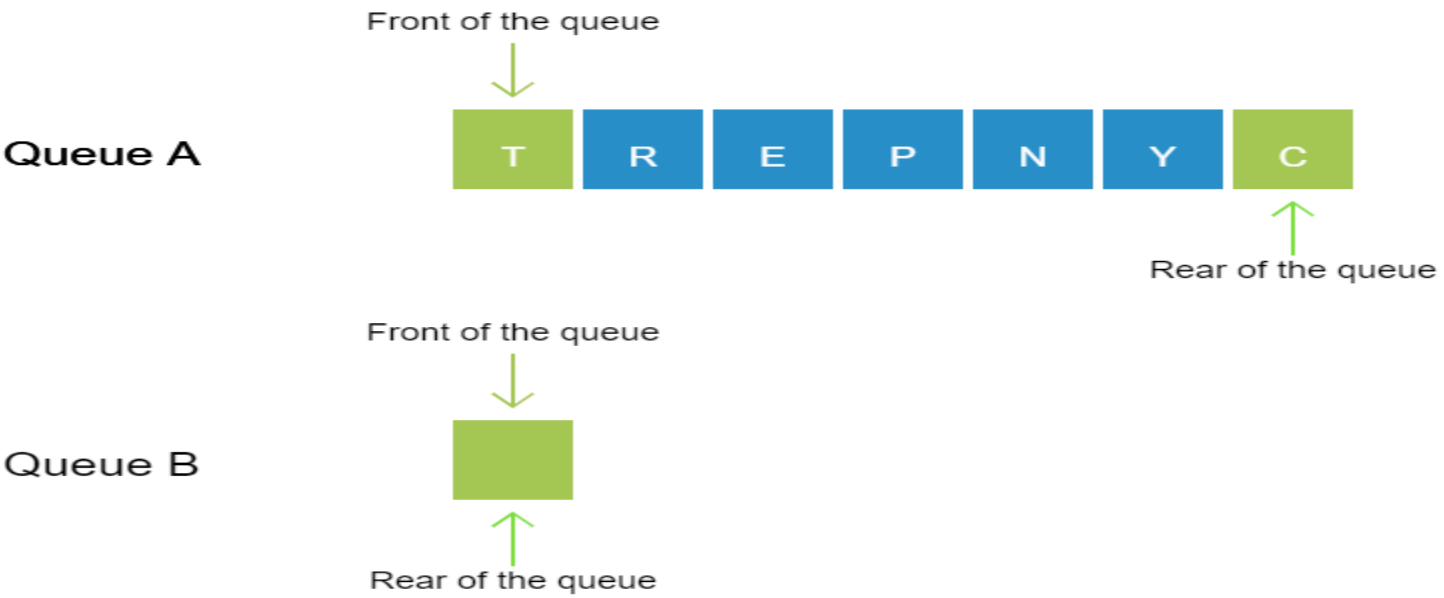
Operations

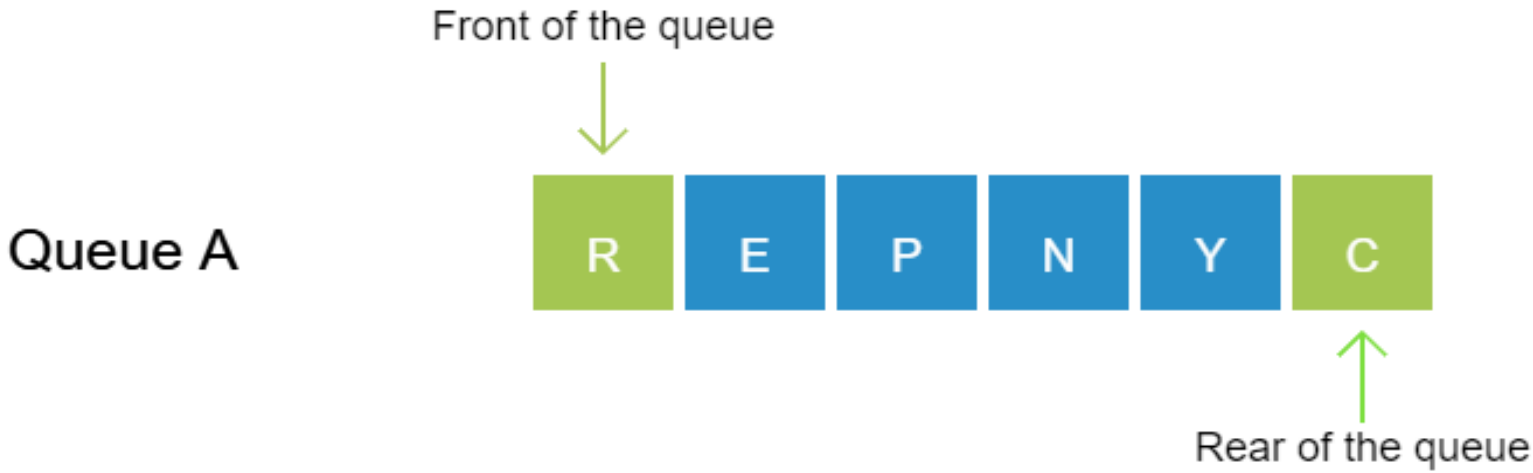


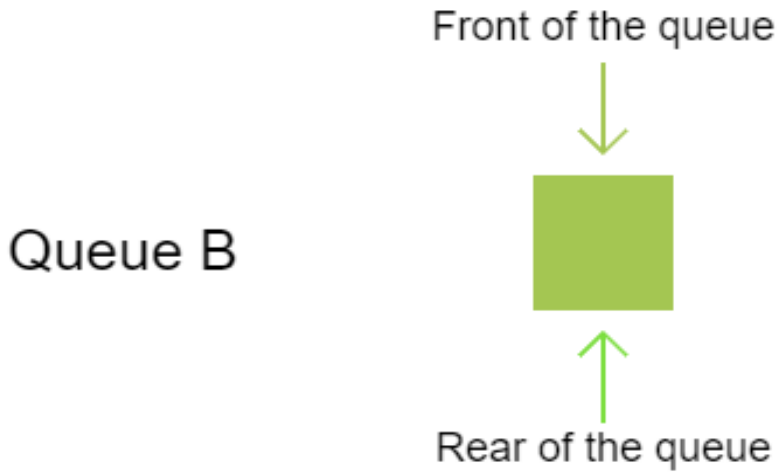
https://learnloner.com/queues-in-data-structures-and-algorithms-dsa/

Working of Queue

Observe: The letters in the word "ENCRYPT" are stored in Queue A in a random manner. Observe the operations performed to display the letters in their correct order in Queue B so as to form the actual word.

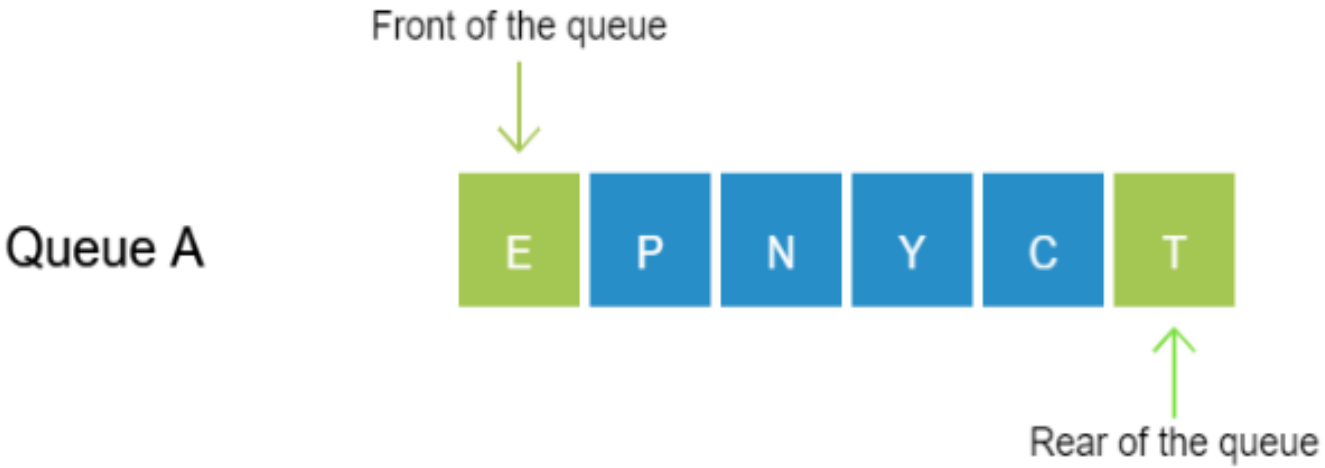


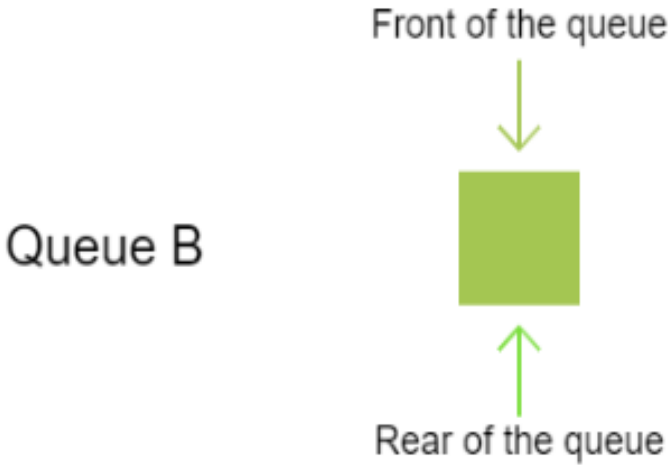




Observations

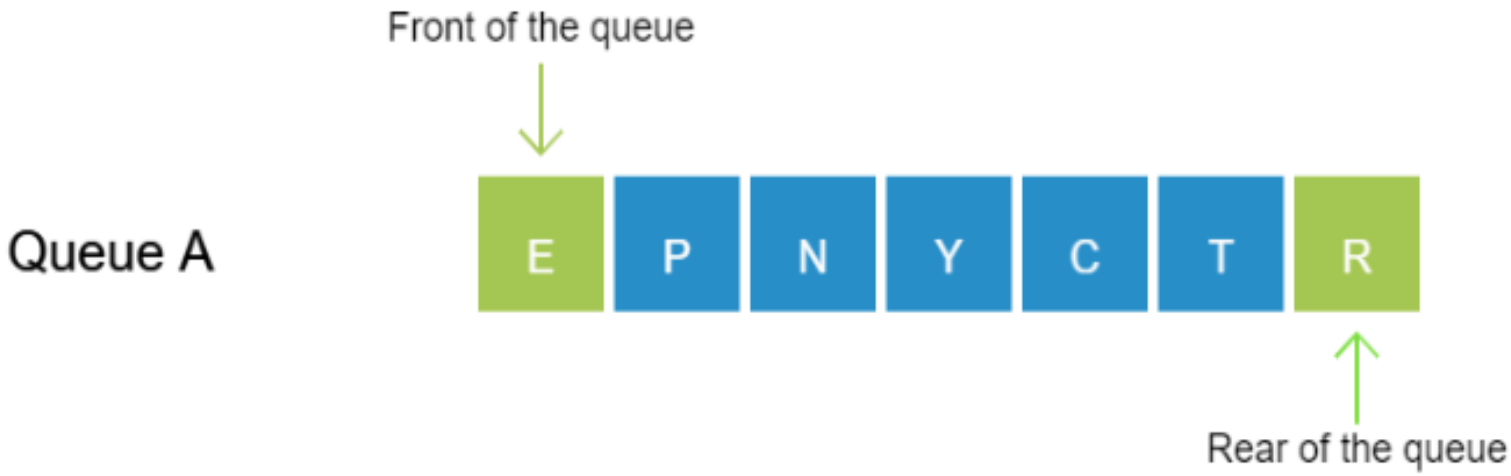
The letter pointed by the Front pointer (i.e T) is not the required one so it is Dequeued from **Queue A**

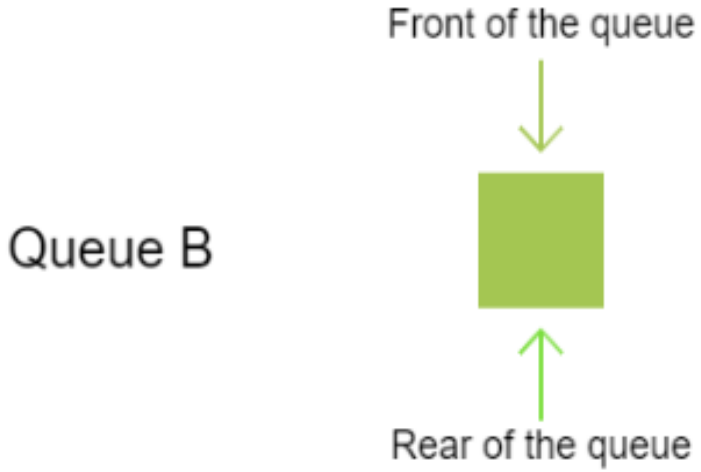




Observations

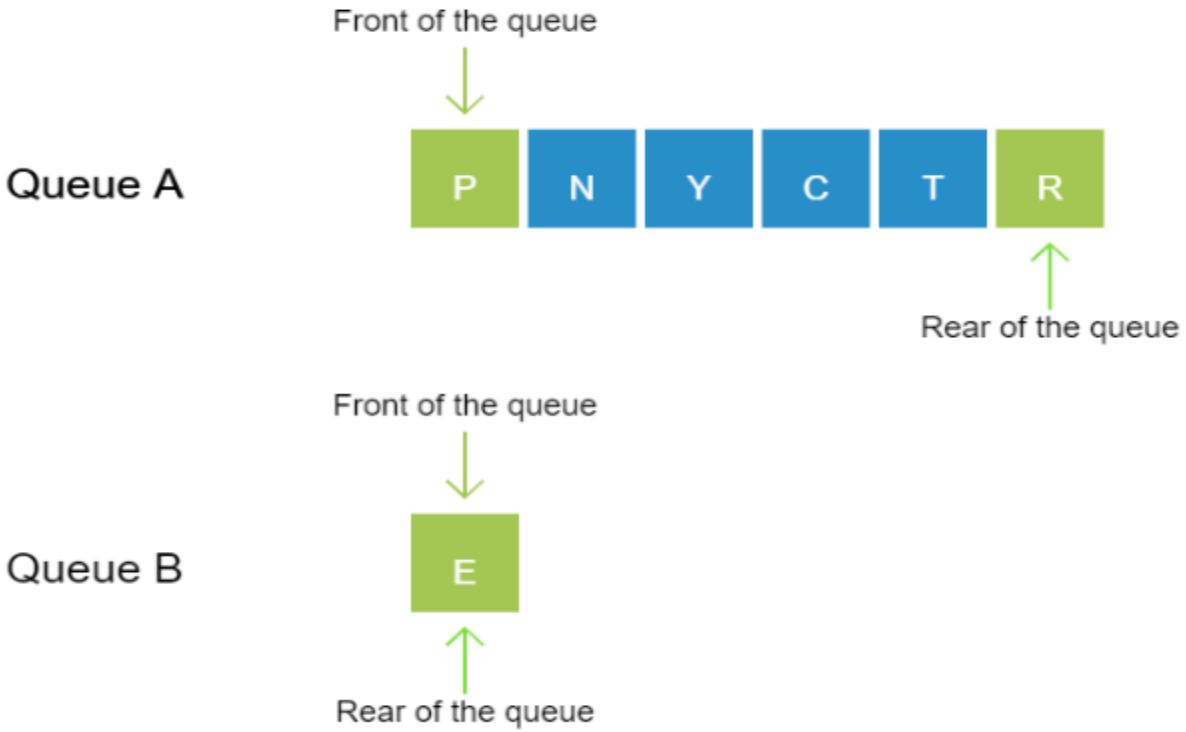
The letter pointed by the Front pointer (i.e R) is not the required one so it is Dequeued from **Queue A**





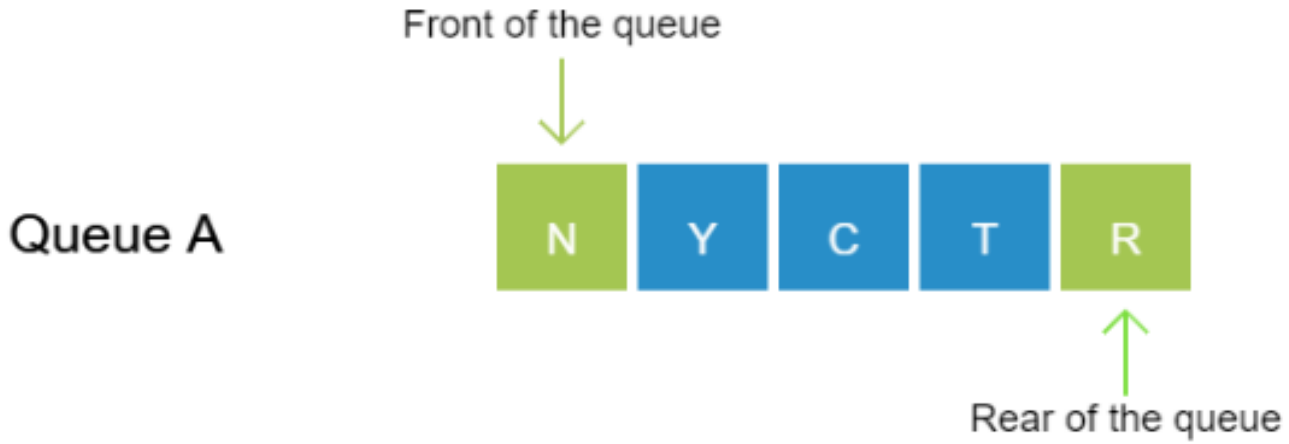
Observations

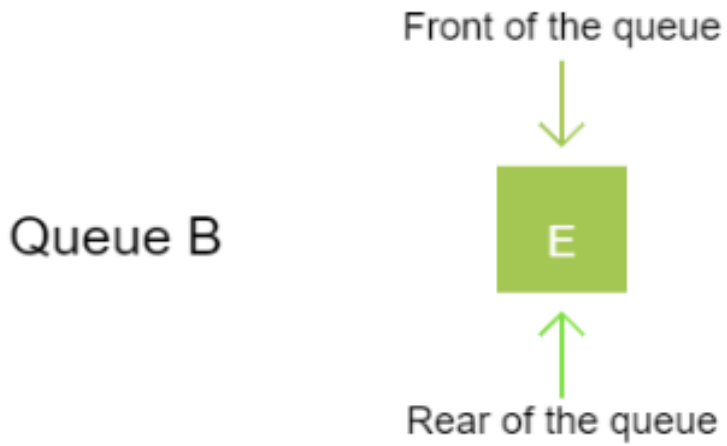
The letter Dequeued from queue A (i.e R) is Enqueued to **Queue A** as it is not the required one



Observations

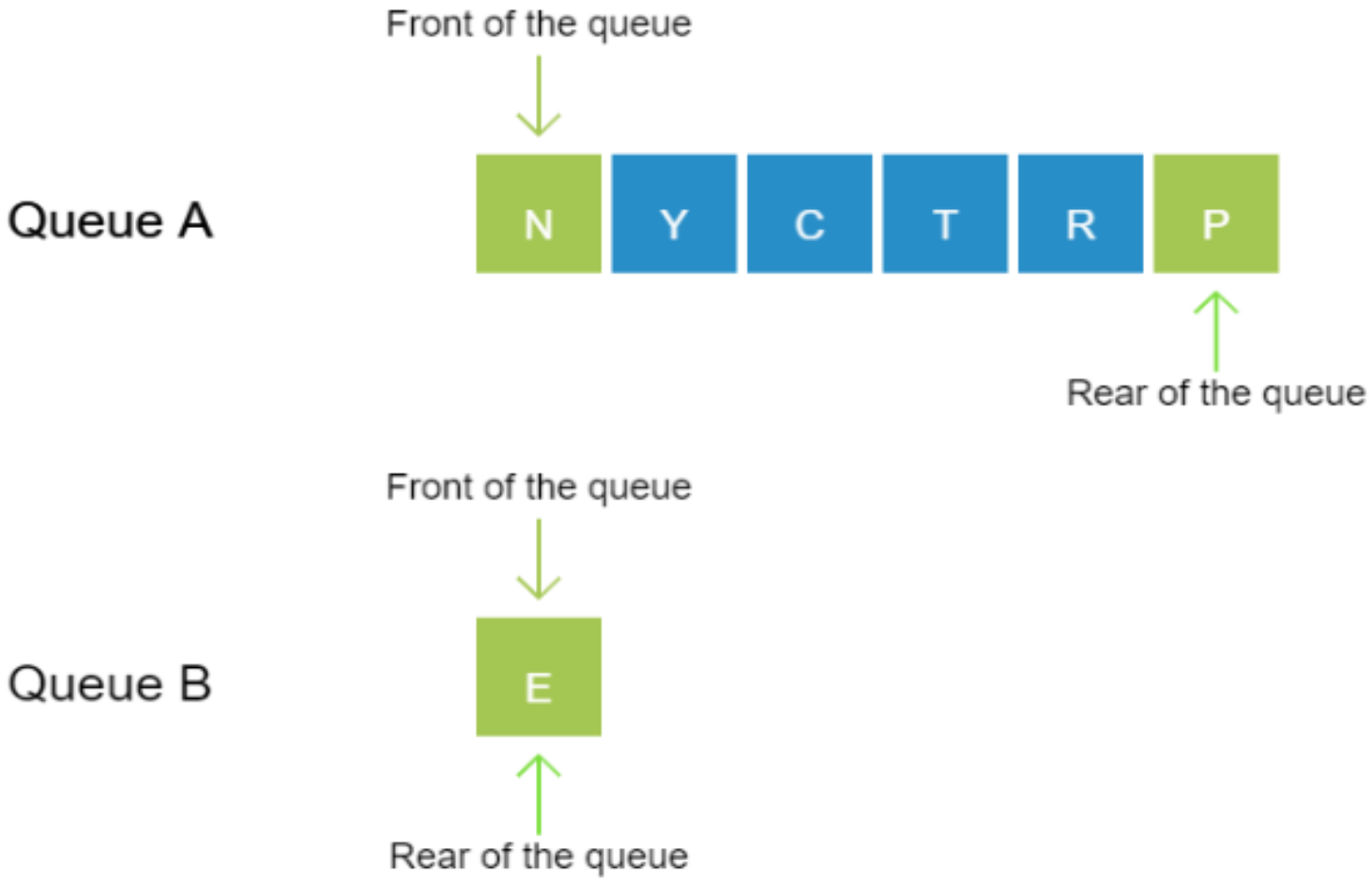
The letter pointed by the Front pointer (i.e E) is the required one so it is Dequeued from **Queue A** and Enqueued to **Queue B**





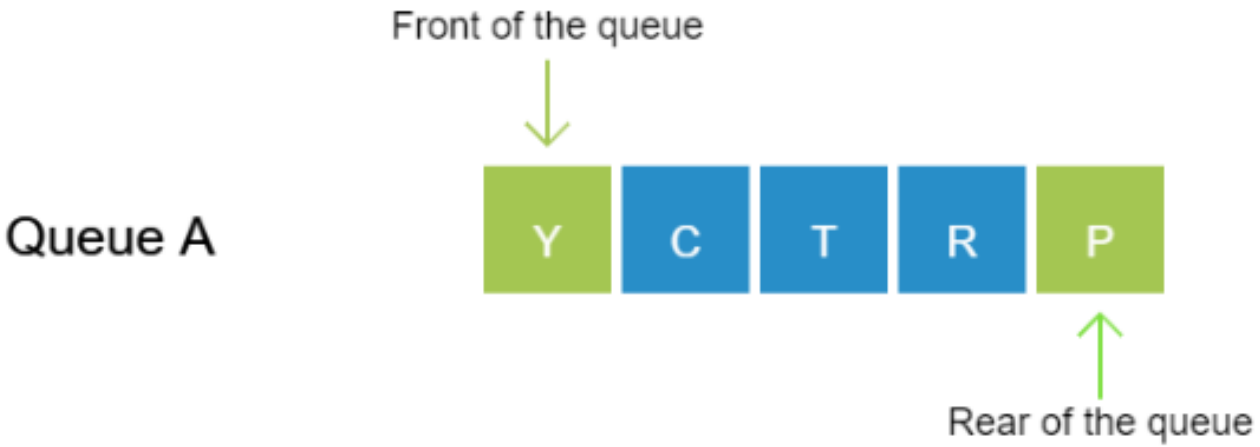
Observations

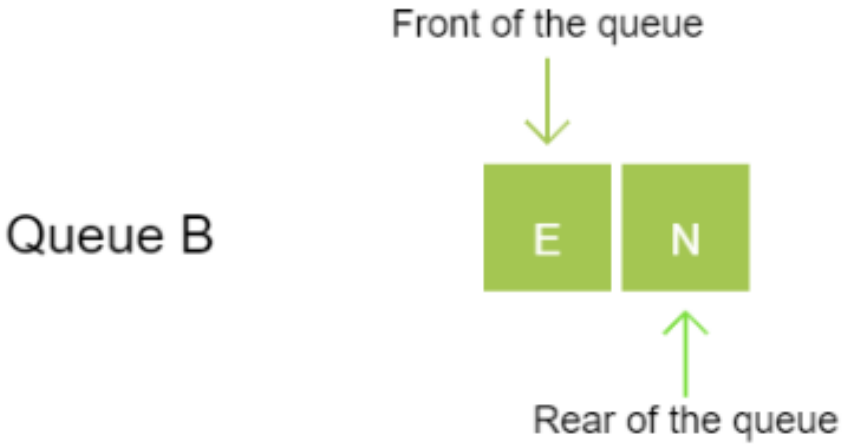
The letter pointed by the Front pointer (i.e P) is not the required one so it is Dequeued from **Queue A**



Observations

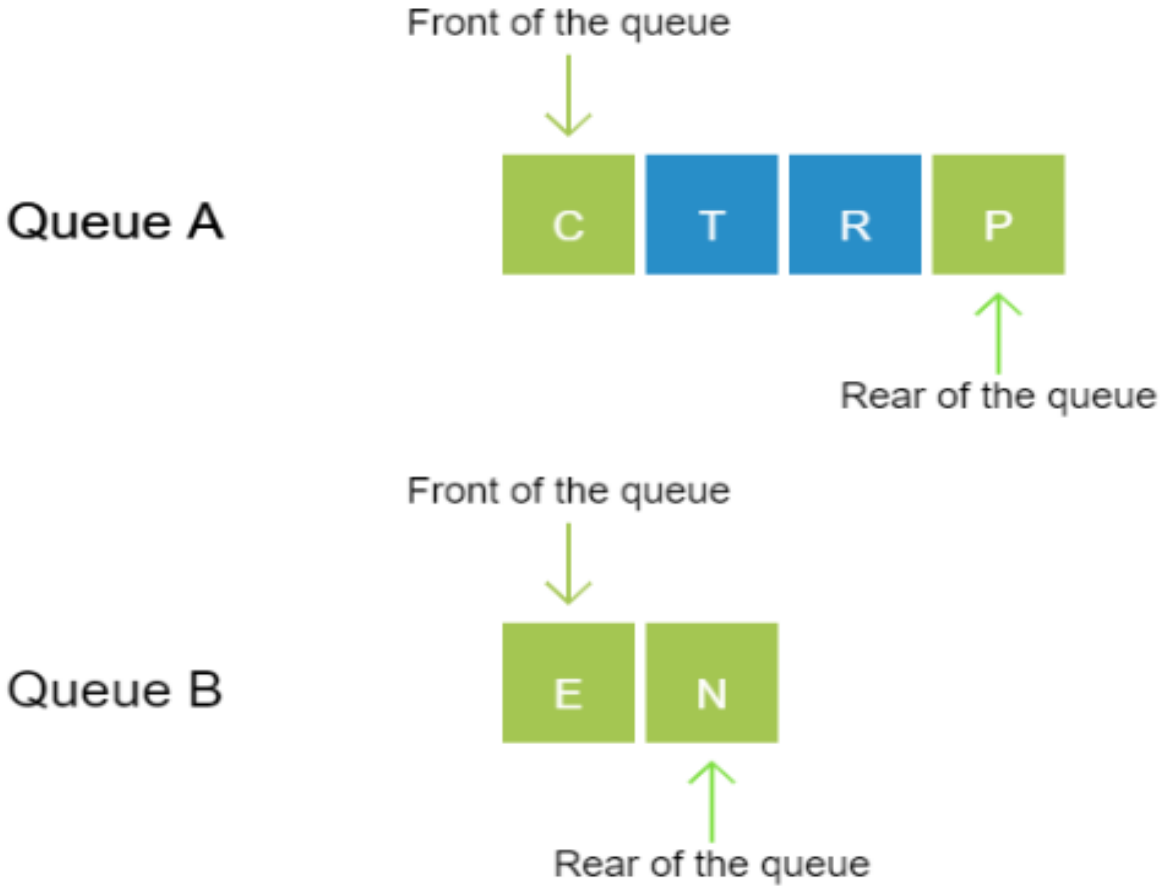
The letter Dequeued from queue A (i.e P) is Enqueued to **Queue A** as it is not the required one





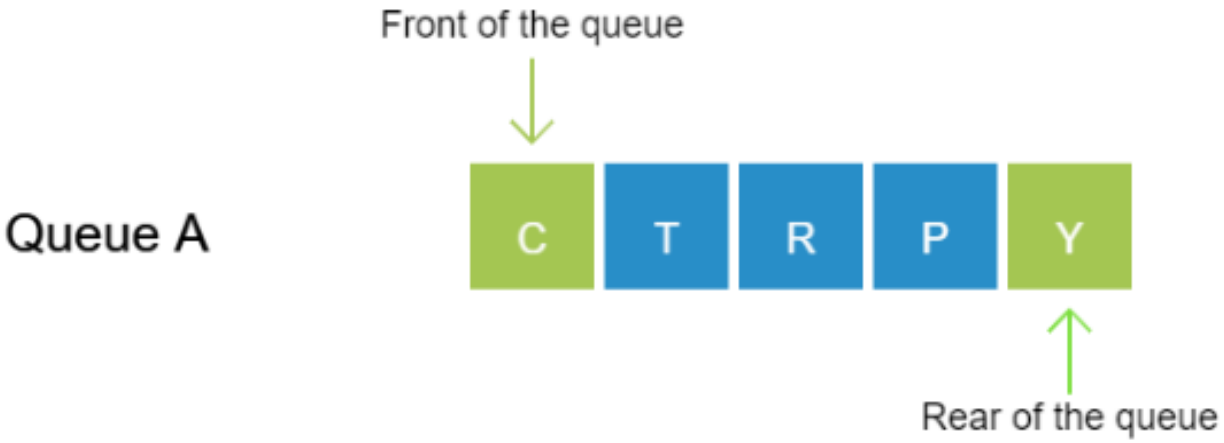
Observations

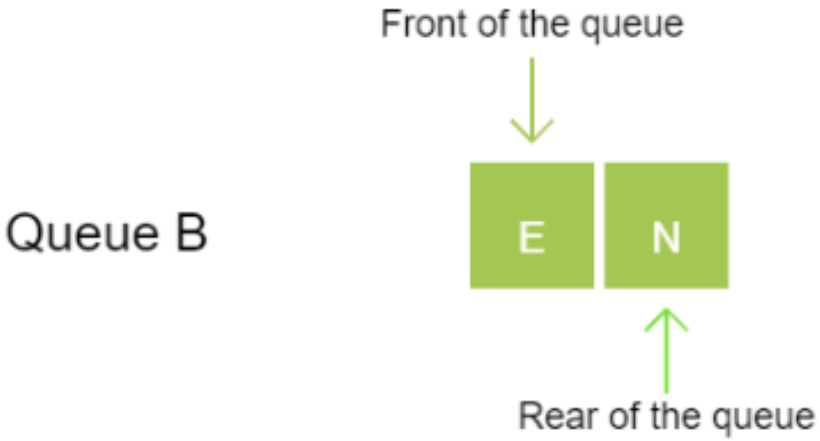
The letter pointed by the Front pointer (i.e N) is the required one so it is Dequeued from **Queue A** and Enqueued to **Queue B**



Observations

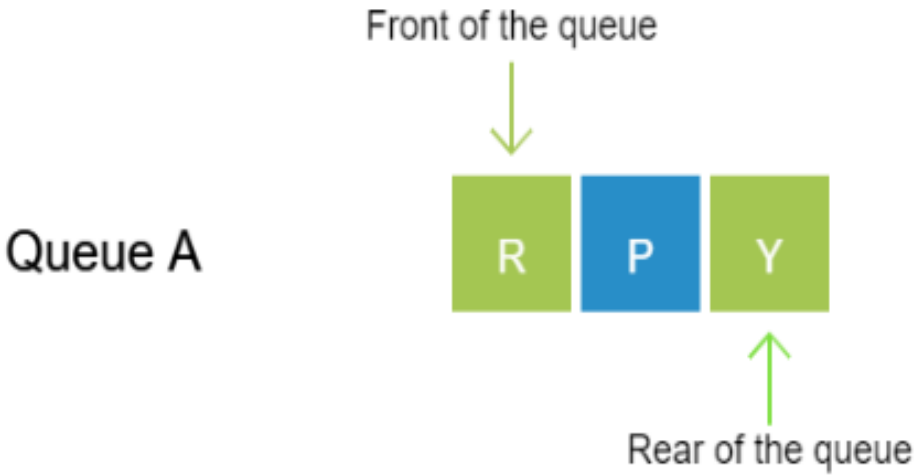
The letter pointed by the Front pointer (i.e Y) is not the required one so it is Dequeued from **Queue A**

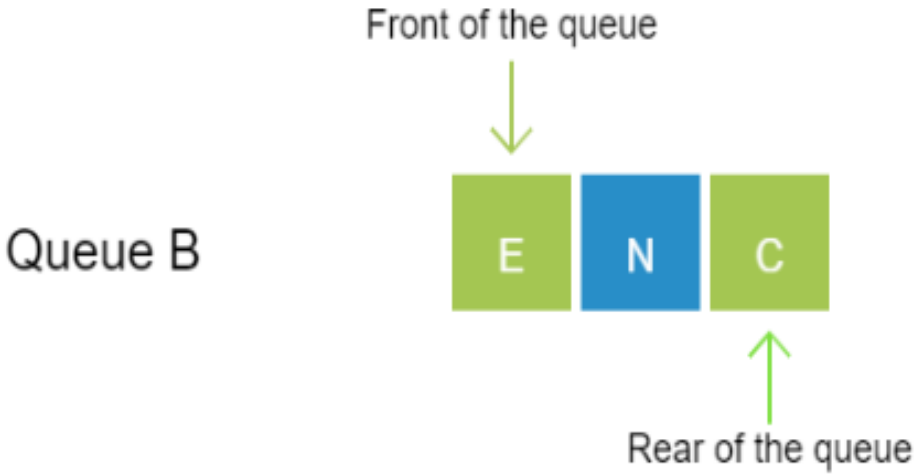




Observations

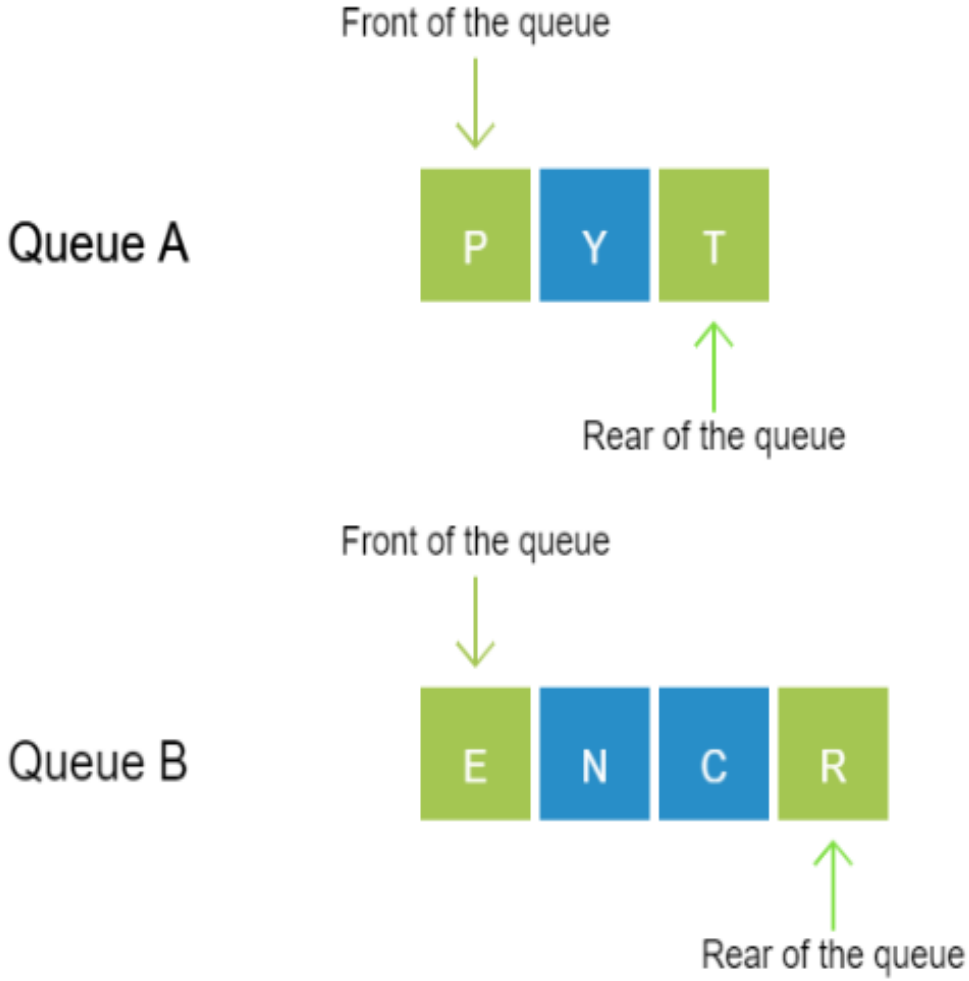
The letter Dequeued from queue A (i.e Y) is Enqueued to **Queue A** as it is not the required one





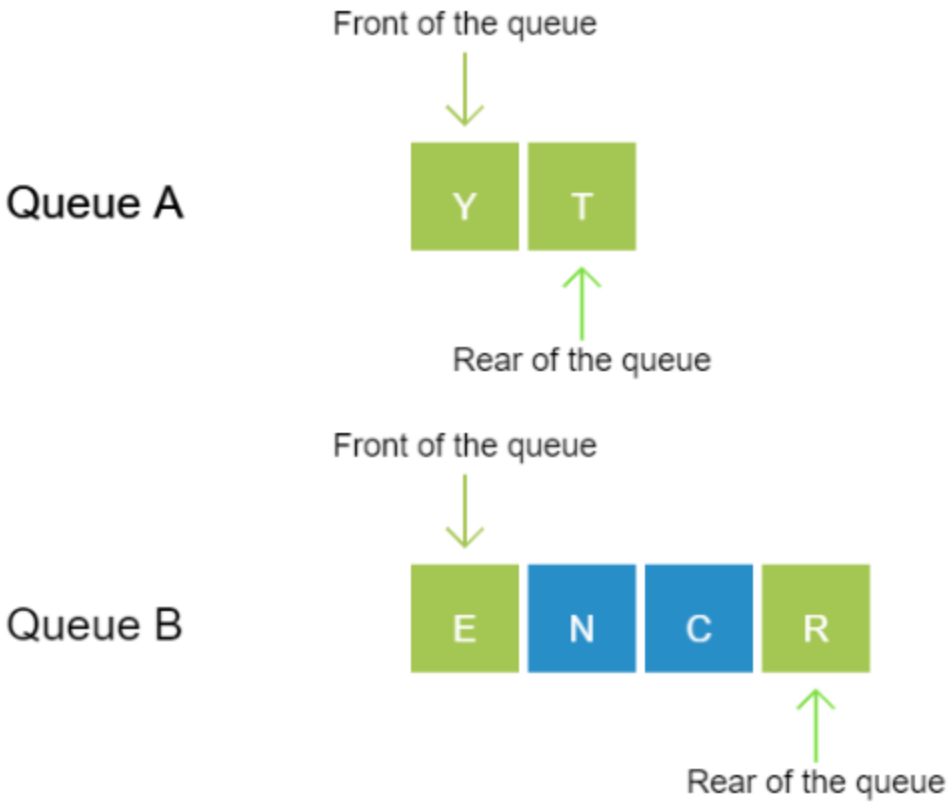
Observations

The letter pointed by the Front pointer (i.e T) is not the required one so it is Dequeued from **Queue A**



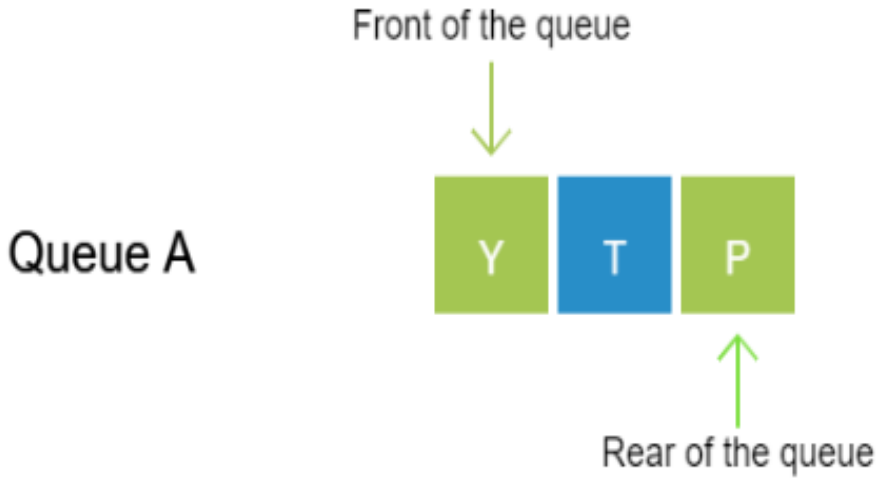
Observations

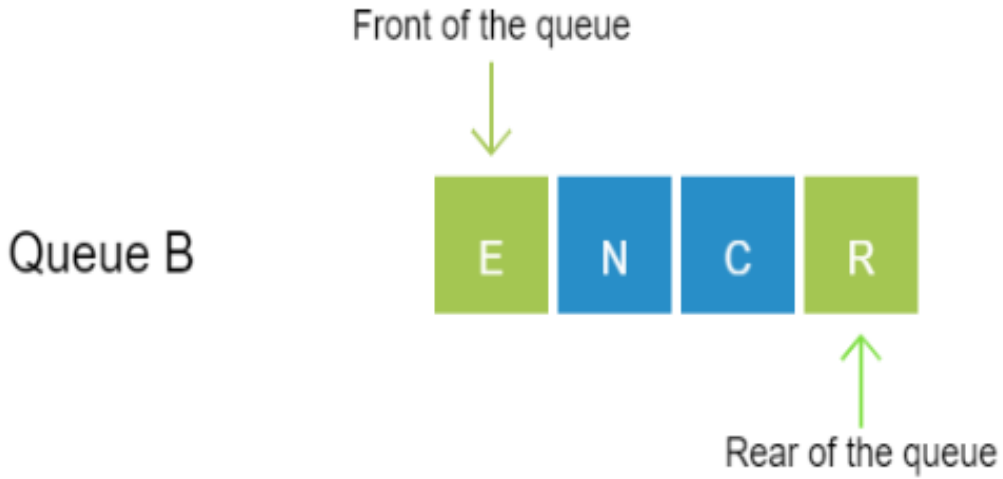
The letter pointed by the Front pointer (i.e R) is the required one so it is Dequeued from **Queue A** and Enqueued to **Queue B**



Observations

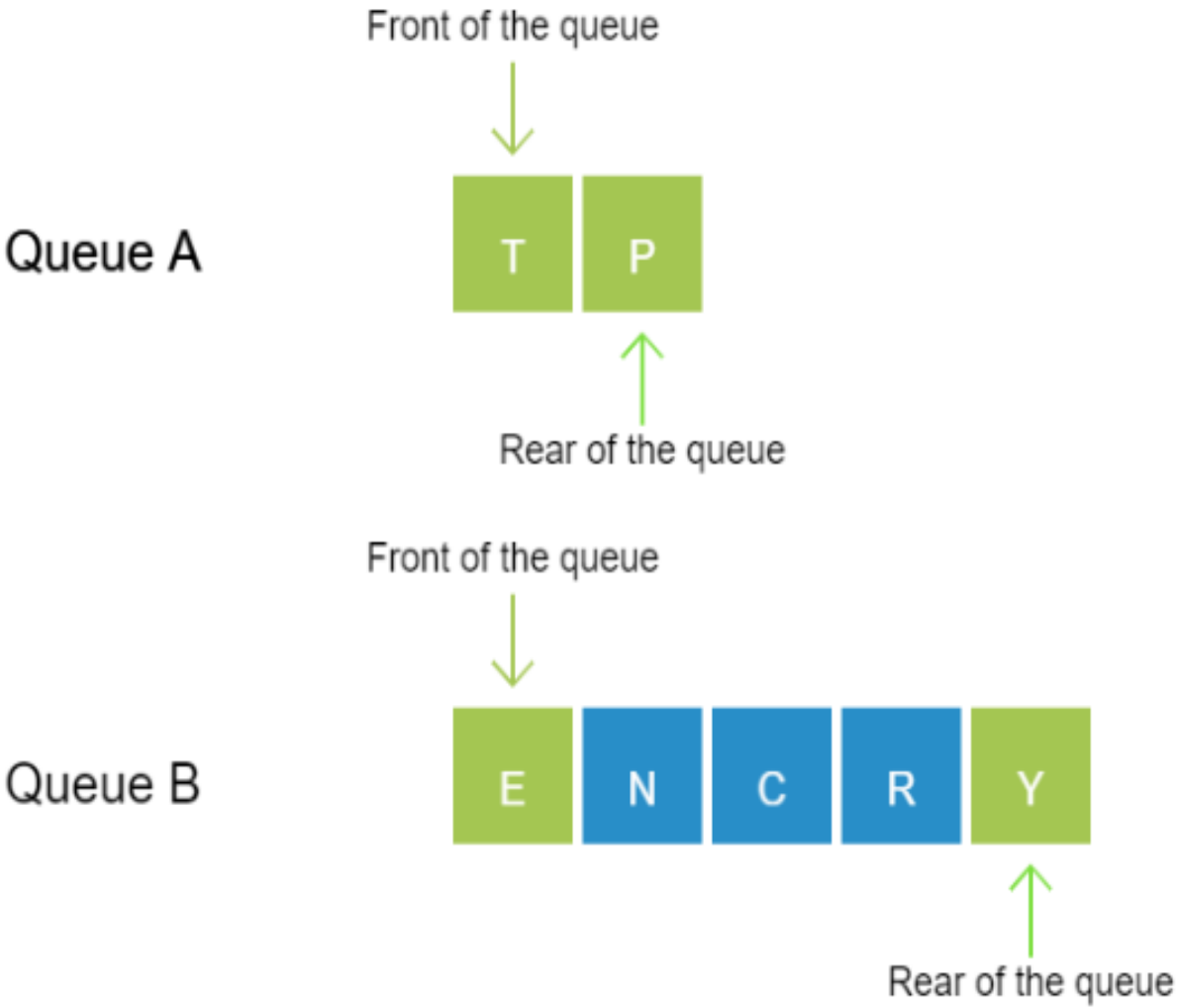
The letter pointed by the Front pointer (i.e P) is not the required one so it is Dequeued from **Queue A**





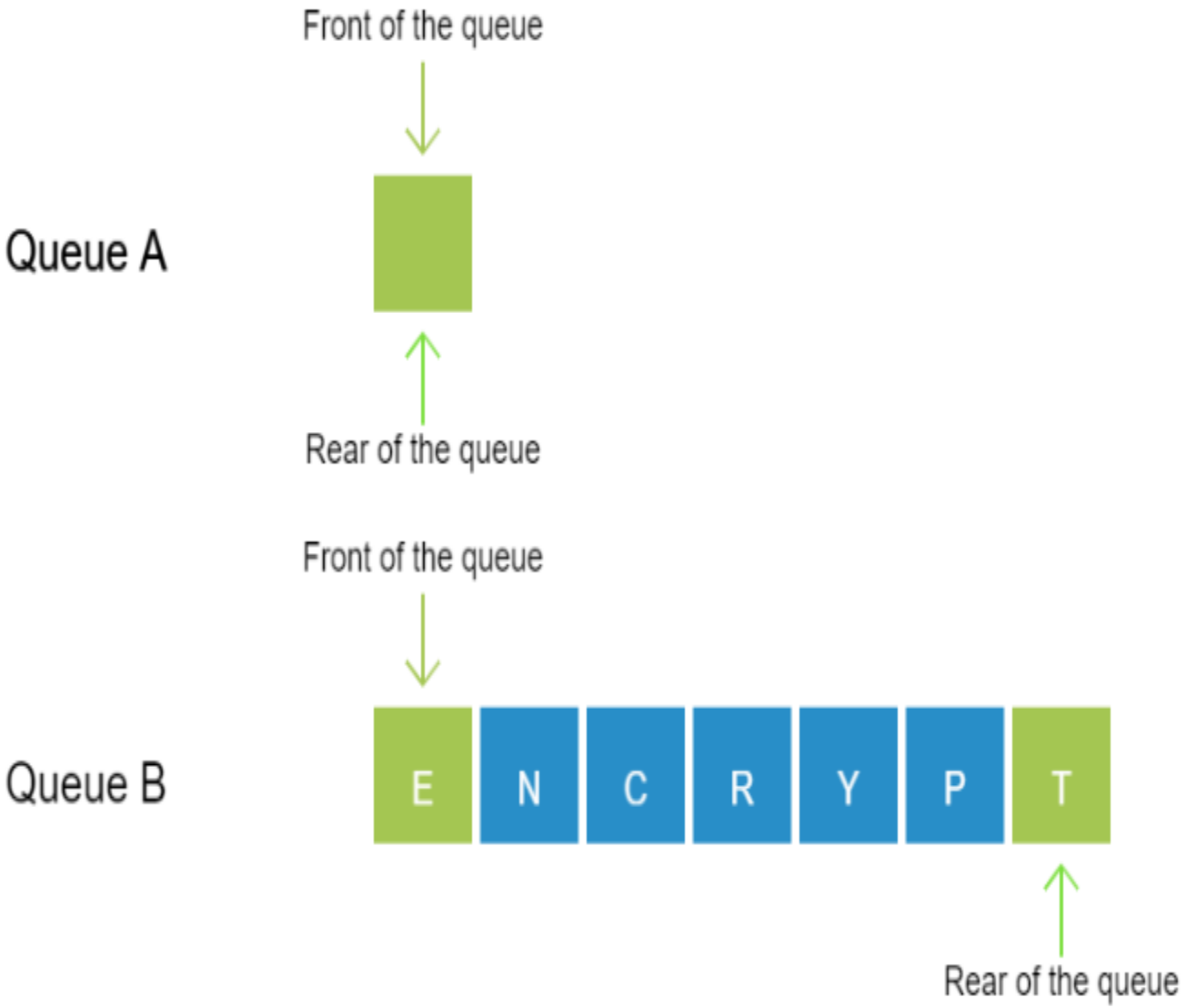
Observations

The letter Dequeued from queue A (i.e P) is Enqueued to **Queue A** as it is not the required one



Observations

The letter pointed by the Front pointer (i.e Y) is the required one so it is Dequeued from **Queue A** and Enqueued to **Queue B**


Front of the queue
Queue A
Rear of the queue
Front of the queue
Queue B
E
N
C
R
Y
P
T
Rear of the queue

Test Yourself

Which operation adds an element to the end of a Queue?

A. Enqueue
B. Dequeue
C. Peek
D. Push

Which operation removes an element from the front of a Queue?

A. Enqueue
B. Dequeue
C. Peek
D. Push

What happens when attempting to enqueue an element into a full Queue in an array-based implementation?

A. The element is added to the end of the Queue, expanding the size of the array.

B. The element is added to the end of the Queue, and the front element is removed.

C. The operation fails, as the Queue is full.

D. The element is added to the beginning of the Queue, shifting all elements one position to the right.

Which of the following data structures is typically used for the underlying implementation of Queues?

A. Arrays
B. Linked Lists
C. Stacks
D. Trees

What are the two primary operations performed on a Queue?

A. Insertion and Deletion
B. Searching and Sorting
C. Push and Pop
D. Enqueue and Dequeue

In an array-based implementation of a linear Queue, where is the front pointer initially positioned?

A. At the beginning of the array
B. At the end of the array
C. At a random position within the array
D. Not applicable, as arrays cannot be used to implement Queues

(Answers)

1. A. Enqueue
2. B. Dequeue
3. C. Peek
4. A. O(1) for both enqueue and dequeue
5. C. The operation fails, as the Queue is full.
6. B. Linked Lists
7. D. Enqueue and Dequeue
8. A. At the beginning of the array

Review

Basic Introduction of Queues: Queues are a fundamental data structure that follows the First In, First Out (FIFO) principle, similar to waiting in line. They manage data sequentially, with elements added to the rear and removed from the front.

Operations of Queues:

Enqueue: Insertion of an element at the rear.

Dequeue: Removal of an element from the front.

Peek: Viewing the front element without removal.

isEmpty, isFull, Size: Checking queue status.

Types of Queues

Implementations of Queues:

Array-based Queue: Simple, fixed size.

Linked List Queue: Dynamic size, efficient insertion/deletion.


THANK
YOU