



Intelligent Medical Systems Department

Data Structures – Lecture (3)
Second Stage

Lecturer Name

Asst.Prof. Mehdi Ebady Manna



جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

كلية العلوم

قسم الأنظمة الطبية الذكية

Intelligent Medical Systems Department

Lecture: (3)

Queues

Subject: Data structure

Class: second

Lecturer: Prof.Dr. Mehdi Ebady Manaa

Programmer:- Fatima Hussein Jawad



A queue : is a linear data structure that follows the First In, First Out (FIFO) principle, meaning that the first item that comes in is the first item that comes out. A queue can be thought of as a waiting line (queue) where people join at the end and leave at the beginning.

➤ **Basic operations in Queue:**

1. Enqueue: Add an item to the end of the Queue.
2. Dequeue: Remove the first item from the Queue.
3. Peek/Front: See the item at the front of the Queue without removing it.
4. isEmpty: Checks if the Queue is empty.
5. isFull: Checks if the Queue is full (if it has a specified capacity).

➤ **Types of Queue:**

1. **Simple Queue:** Allows items to be inserted from the end and removed from the beginning.
2. **Circular Queue:** Circular queue where the last location is linked to the first location, allowing space to be reused.
3. **Priority Queue:** Priority queue, where items are processed based on priority rather than chronological order.



4. **Deque (Double-Ended Queue):** Items can be inserted and removed from both ends.

➤ Here are the basic operations of a Queue in Python, demonstrated with simple examples:

1. **Enqueue (Adding an item to the queue)** This operation adds an item to the end of the queue.

```
queue = []  
queue.append(1) # Enqueue operation  
queue.append(2)  
queue.append(3)  
print("Queue after enqueue operations:", queue)  
  
# Output: Queue after enqueue operations: [1, 2, 3]
```

2. **Dequeue (Removing an item from the queue)**

This operation removes the item from the front of the queue (the first item that was added).

```
queue = []  
queue.append(1) # Enqueue operation  
queue.append(2)  
queue.append(3)
```



```
first_item = queue.pop(0)
print("Dequeued item:", first_item)  # Dequeue operation
print("Queue after dequeue:", queue)
# Output: Dequeued item: 1
# Output: Queue after dequeue: [2, 3]
```

3. Peek (Looking at the front item)

This operation lets you view the front item of the queue without removing it.

```
queue = [1,2,3]
front_item = queue[0] if queue else None  # Peek operation
print("Front item:", front_item)
# Output: Front item: 1
```

4. isEmpty (Check if the queue is empty)

This operation checks whether the queue is empty or not.

```
queue = [1,2,3]
is_empty = len(queue) == 0  # isEmpty operation
print("Is the queue empty?", is_empty)
# Output: Is the queue empty? False
```



5. Queue Size (Get the current size of the queue)

This operation returns the number of items in the queue.

```
queue = [1,2,3]
```

```
queue_size = len(queue)           # Size of the queue
```

```
print("Queue size:", queue_size)
```

Output: Queue size: 3

➤ Summary of operations:

Enqueue: Add to the end of the queue (**append**).

Dequeue: Remove from the front of the queue (**pop(0)**).

Peek: View the front item without removing it (**queue[0]**).

isEmpty: Check if the queue is empty (**len(queue) == 0**).

Size: Get the current size of the queue (**len(queue)**).

➤ The difference between array , stack and queue:

Array : provides random access to elements and allows data to be stored in sequential form.

Stack: Commonly used for backtracking algorithms, parsing expressions, and managing function calls.

Queue: Often used in scheduling tasks, managing processes, and handling requests in order.



In summary, a Stack allows for reverse order processing (LIFO), while a Queue processes elements in the order they were added (FIFO).

➤ **Arithmetic operations on the Queue :**

1. Code for Addition Between Two Arrays .

```
queue1 = [10, 20, 30]           # Two arrays representing queues
queue2 = [5, 15, 25]
result = []                     # List to store the result of the addition
for i in range(len(queue1)):    # Add elements from both arrays
    result.append(queue1[i] + queue2[i])
print("Sum result:", result)
```

2. Code for Subtraction Between Two Arrays

```
queue1 = [50, 40, 30]          # Two arrays representing queues
queue2 = [10, 20, 5]
result = []                     # List to store the result of the subtraction
for i in range(len(queue1)):    # Subtract elements from both arrays
    result.append(queue1[i] - queue2[i])
print("Subtraction result:", result)
```



3. Code for Division Between Two Arrays

```
queue1 = [100, 50, 40]           # Two arrays representing queues
queue2 = [2, 5, 8]
result = []                       # List to store the result of the division
for i in range(len(queue1)):      # Divide elements from both arrays
    if queue2[i] != 0:            # Ensure not dividing by zero
        result.append(queue1[i] / queue2[i])
    else:
        result.append(None)      # Handle division by zero case
print("Division result:", result)
```

4. Code for Checking Odd and Even Numbers in an Array

```
queue = [15, 22, 35, 44]         # An array representing a queue
for num in queue:                # Check for odd and even numbers
    if num % 2 == 0:
        print(num, "is even")
    else:
        print(num, "is odd")
```

Here are the previous operations on a queue implemented in Python without using any external libraries ..



1. Calculate Size:

```
queue = [1, 2, 3, 4, 5]
def size(queue):
    return len(queue)
print(size(queue))  # Output: 5
```

2. Search for an Element in the Queue:

```
queue = [1, 2, 3, 4, 5]
def search(queue, element):
    return element in queue
print(search(queue, 3))  # Output: True
print(search(queue, 6))  # Output: False
```

3. Find Maximum or Minimum:

```
queue = [1, 2, 3, 4, 5]
def find_max(queue):
    return max(queue)
def find_min(queue):
    return min(queue)
print(find_max(queue))  # Output: 5
print(find_min(queue))  # Output: 1
```




4. Reverse the Queue:

```
queue = [1, 2, 3, 4, 5]
def reverse_queue(queue):
    return queue[::-1]
reversed_queue = reverse_queue(queue)
print(reversed_queue) # Output: [5, 4, 3, 2, 1]
```

5. Merge Two Queues:

```
queue = [1, 2, 3, 4, 5]
queue2 = [6, 7, 8]
def merge_queues(queue1, queue2):
    return queue1 + queue2
merged_queue = merge_queues(queue, queue2)
print(merged_queue) # Output: [1, 2, 3, 4, 5, 6, 7, 8]
```

6. Rotate the Queue:

```
queue = [1, 2, 3, 4, 5]
def rotate_queue(queue):
    first_element = queue.pop(0) # Remove the first element
    queue.append(first_element) # Add it to the end of the list
    return queue
rotated_queue = rotate_queue(queue)
print(rotated_queue) # Output: [2, 3, 4, 5, 1]
```