Data Structures and Algorithms (DSA): Queues

Introduction to Queues

A queue is a data structure that can contain multiple items. It operates on the FIFO (First-In-First-Out) principle, similar to people standing in line. The first person in line is the first to be served.

Basic Queue Operations

1. Enqueue: Add a new item to the queue.

2. Dequeue: Remove and return the first (front) item from the queue.

3. Peek: Return the first item in the queue without removing it.

4. isEmpty: Check if the queue is empty.

5. Size: Find the number of items in the queue.

Practical Uses of Queues

Queues are commonly used in:

- Task scheduling for office printers.

- Processing e-ticket requests.

- Implementing breadth-first search (BFS) in graph algorithms.

Implementing Queues Using Arrays

Using arrays for queue implementation can have the following benefits and drawbacks:

Benefits:

- Memory Efficiency: Array elements do not store pointers like linked lists.

- Ease of Implementation: Fewer lines of code required.

Drawbacks:

- Fixed Size: Arrays have a fixed memory allocation.

- Inefficiency in Removal: Removing the first item causes a shift, impacting performance.

Python Example Using Lists as Arrays

```
queue = []


# Enqueue items

queue.append('A')

queue.append('B')

queue.append('C')

print("Queue:", queue)


# Dequeue item

item = queue.pop(0)

print("Dequeued:", item)


# Peek

front_item = queue[0]

print("Peek:", front_item)


# isEmpty

is_empty = not bool(queue)

print("isEmpty:", is_empty)


# Size

print("Size:", len(queue))
```