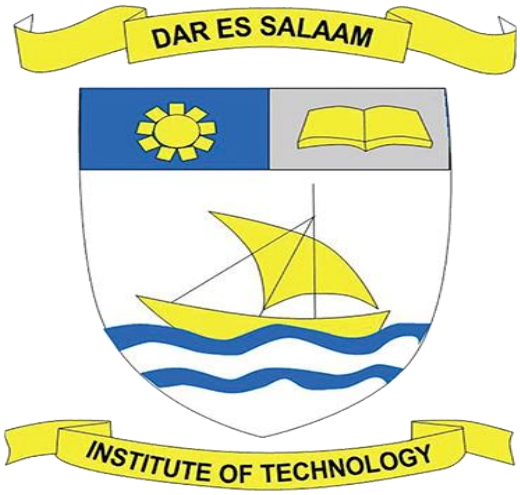
DAR ES SALAAM INSTITUTE OF TECHNOLOGY



GROUP ASSIGMENT 1

PROGRAMME: FUNDAMENTAL OF DATA STRUCTURE AND ALOGARITHMS FOR TECHNICHIANS

CLASS: OD22COE

NO.	GROUP MEMBERS	REGISTRATION NUMBER	
1	DANSTAN FELIX MWINUKA	220222448637	
2	DEOGRATIUS FESTO AKARO	220222481463	
3	EMMANUEL HURUMA	200220228460	

Enqueue and dequeue are two fundamental operations used in data structures, particularly in the context of queues.

Enqueue:

Enqueue is an operation that adds an element to the back or rear end of a queue. Imagine a queue as a line of people waiting for something, such as getting into a concert or buying tickets at a counter. When a new person arrives, they join the end of the line. This process is analogous to enqueueing.

Detailed Steps of Enqueue Operation;

1. Check if the Queue is Full: Before adding an element to the queue, it's essential to check if there's enough space in the queue to accommodate the new element. If the queue is full, you may encounter an overflow condition, depending on how the queue is implemented.

2. Insertion of Element: Once it's determined that there's space in the queue, the new element is inserted at the rear end. In implementations using arrays, this often involves incrementing a rear pointer/index and placing the element at that position. In linked list implementations, a new node is created and attached to the end of the linked list.

3. Update Queue Size: After successfully adding the element, the size of the queue is increased by one.

Dequeue:

Dequeue is an operation that removes an element from the front end of a queue. Continuing with the analogy of a queue of people waiting, when the service begins or the gate opens, the first person in the line gets served or enters first. This process is akin to dequeuing.

Detailed Steps of Dequeue Operation:

1. Check if the Queue is Empty: Before removing an element from the queue, it's crucial to ensure that the queue is not empty. Attempting to dequeue from an empty queue can lead to underflow conditions in some implementations.

2. Removal of Element: Once it's confirmed that there are elements in the queue, the element at the front end is removed. In array implementations, this often involves shifting the remaining elements to fill the empty space created by removing the front element. In linked list implementations, the node at the front is detached, and the head pointer is updated.

3. Update Queue Size: After successfully removing the element, the size of the queue is decreased by one.

Complexity

The time complexity of both enqueue and dequeue operations depends on the underlying data structure used to implement the queue. In general:

- For a queue implemented using an array, the time complexity of enqueue and dequeue operations is O(1), assuming no resizing is required.
- For a queue implemented using a linked list, both enqueue and dequeue operations have a time complexity of O(1), aSs adding or removing an element from the beginning or end of a linked list can be done in constant time.

Understanding enqueue and dequeue operations is fundamental to working with queues, which are widely used in various applications such as breadth-first search algorithms, task scheduling, and managing requests in computer systems.

1. Enqueue:
   - Enqueue is the process of adding an element to the back or rear end of a queue.
   - It is like getting in line at a store; you join the queue behind everyone else.
   - In terms of implementation, when you enqueue an element, you insert it at the end of the queue.
   - This operation increases the size of the queue by one.
   - Enqueue operation in a queue is often termed as "push" operation in some programming languages or contexts.

2. Dequeue:
   - Dequeue is the process of removing an element from the front end of a queue.
   - It's akin to being served at a counter; the person at the front of the line gets served first.
   - In terms of implementation, when you dequeue an element, you remove it from the front of the queue.
   - This operation decreases the size of the queue by one.
   - Dequeue operation in a queue is often termed as "pop" operation in some programming languages or contexts.

Queues follow the First-In-First-Out (FIFO) principle, meaning the element enqueued first will be dequeued first. These operations are crucial for managing data in scenarios where you need to process elements in the order they were added. They are widely used in computer science for tasks such as job scheduling, task management, and more.