Lecture 23.1

Topics:

1. Stack Implementation – Linked List Based
2. Queue — Introduction

---

1. Stack Implementation – Linked List Based

Again, a **`Last-In First-Out (LIFO)`** stack is generally depicted as in **Figure 1**.

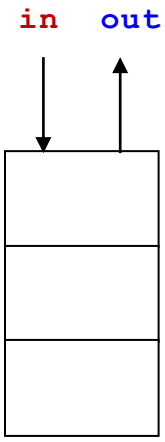

Figure 1 A stack representation

2.1 Stack Implementation — Linked List Base

Again, there are two basic operations that one can do with a stack:

(1) Pushing,

(2) Popping, and

(3) GetTop

With linked list based stack –

- There is no issue of size limitation.
- The operations are happening at the first node of the list.
- This understanding will give us the following implementation:

```
pop() → removeFirst()
push() → insertFirst()
top() → getFirst()
isEmpty() → isEmpty()
```

An implementation will be presented in class.

2. Queue ADT -- Introduction

In general, a queue is a data structure that stores and retrieves items in a **First-In-First-Out** (FIFO) order. Note that a stack is a **Last-In-First-Out** (LIFO) data structure.

2.1 Definition

A queue is a data structure that holds a sequence of objects (or, set of data values) and provides access to its objects in **FIFO** order. A queue has two ends, which are called the **rear** and the **front** of the queue. These two ends will provide current locations of the first and last objects from which operations can take place in the queue.

Figure 1 depicts a queue with its rear and front indicated.

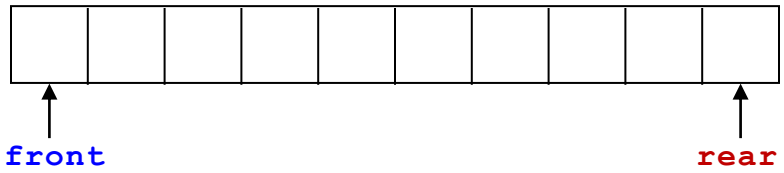

Figure 2 General queue

2.2 Applications

There are many applications of queue data structures.

a. Computer Operating System
b. Communication Software
c. Airline Checking Counter
d. Store Checking Counter

2.3 Queue Operations

When an object is added to a queue, it will be added to the rear. When an element is removed from the queue, it is removed from the front. The two primary queue operations are enqueuing and dequeuing.

- **To enqueue means to insert or add an object at the rear of a queue.**
- **To dequeue means to remove or delete an object at the front of a queue.**

There are several algorithms for implementing these operations based on the structures and behaviors such as linked lists, arrays, fixed-end, moving-end, etc.

The linked list approach offers dynamic queue with no size limit. The array-based approach does inherit the size limitation but in general faster than the linked list approach.

In the array-based approach, the basic considerations are with fixed ends and moving ends. The fixed end approach presents inefficiency while a moving end approach offers a better choice. In forming one of these moving-end queues, one must consider the case where the ends need to be wrapped around as the queue is not full and being added with new object while the ends are moving around.

Thus, the consideration of circular queue comes into the picture. The following implementation will be for a moving end and circular array approach.