Introduction to Queues

⇒ What is Queue

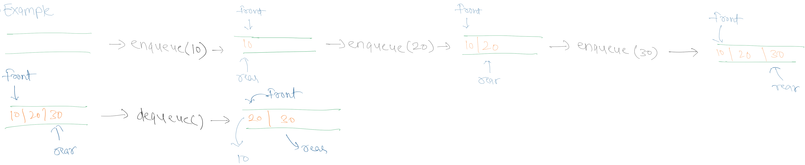
A list or collection with the restriction that insertion can be performed at one end (rear) and deletion can be performed at other end (front)

⇒ Stack as ADT

1. Enqueue(x) or push(x)
   ↳ Insert the element at the rear end of the queue
2. Dequeue() or pop()
   ↳ Delete the element from the front of the queue.
3. empty()
   ↳ returns true if queue is empty, false is not empty.
4. front() / peek()
   ↳ returns the element at the front end of the queue.

All of these above operations are performed in constant time, i.e their time complexity is O(1)

⇒ Example



⇒ Implementation in C++

[Done in class]

⇒ C++ library of using queue

C++ provides #include <queue> to use the already implemented queue data structure.

It provides similar operations as discussed above.

ex:-

queue<int> Q;

defines an queue of int data type.

⇒ Applications of queue

1. When we have shared resource, but we can handle only one request at a time.

   ex:- a single printer connected to multiple computers.

the software of the printer will maintain a queue to process the requests made by each computer, the order in which the queue will be maintained will be according to our need,

for example the software may follow first come - first serve approach, it may follow a priority based approach.

there are many scheduling algorithms available in OS, that uses different kind of queue.

⇒ Types of queue

1. Normal queue
2. Circular queue. [queue<int>]
3. Priority queue. [priority_queue<int>]

   } available in <queue>

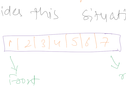
4. Double ended queue. [deque<int>]

   } available in <dequeue>

⇒ Circular queue

Limitations of a normal, linear queue

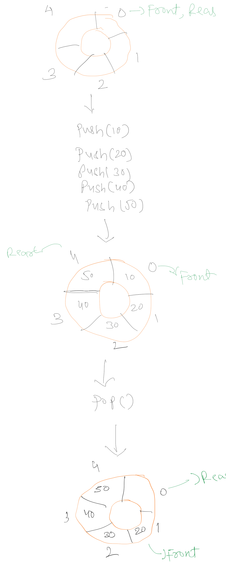
consider this situation



now if we remove the front element, the queue becomes.

Now if try to enqueue, the element wont get inserted because, the rear is at the end, and even if there is available space, the element is not getting inserted.

For overcoming this limitation, we introduce circular queues.

What is circular queue?

A circular queue is a special version of queue where the last element of the queue is connected to the first element of the queue.