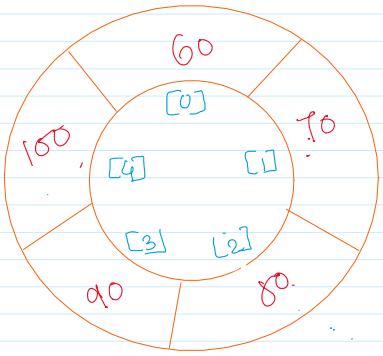
02 December 2025 14:35

arr[5]

front

rear



Queue Full Condition

front Rear

front == (Rear + 1) % SIZE

0 == (4 + 1) % 5

Enqueue:

1) Increment rear as
   Rear = (rear+1) % SIZE

1. Add element at rear position
2. If front == -1, make front = 0

Dequeue :

1) Increment front .

****

If front is 4 and rear is 0
to delete the rear position, we cannot increment front as front ++ Will be index 5 but we want to delete index 0;
So,
Front = (front +1) %SIZE

If deleting the last element in queue
If(front == rear)
Front = rear = -1

Deqeueue()

front++

front = (front + 1) % SIZE

front = (4 + 1) % 5 = 0

Enqueue()

1) Rear++; ×

Rear = (Rear + 1) % SIZE ✓

Rear = (-1 + 1) % 5 = 0

Rear = (0 + 1) % 5 = 1

Rear = (1 + 1) % 5 = 2

Rear = (2 + 1) % 5 = 3

Rear = (3 + 1) % 5 = 4

Rear = (4 + 1) % 5 = 0

Queue Empty condition
If(rear == -1) queue is empty

Queue Full condition :
When Queue is full,

Rear = 0 front 1
Rear = 1, front = 2
Rear = 2 front = 3
Rear = 3 front = 4
rear = 4, front = 0

This means,
Front == rear + 1

But when rear = 4 front =0
0 == 4+1 does not satisfy the above condition
So the queue full condition can be
Front == (rear+1) %SIZE
0 == (4+1) % 5
0 == 5%5
0==0

Option 2:

```
If rear == SIZE-1
    Rear = 0;
Else
    Rear++;
```

Double Ended Queue : (DEQUE)
In this queue, we can insert and delete the data from both the ends(rear and front)
Insertion has 2 functions :(insert_front and insert_rear)
Deletion has 2 functions : (delete_front and delete_rear)
This Queue is used on the basis of requirements.
It does not follow the FIFO functionality of the QUEUE.

Priority Queue : Insertion in the queue is done from the rear end.
Each element is given some priority. And the deletion is done based on the priority.
The element with the highest priority is deleted first.
Hence this queue also does not follow FIFO functionality.