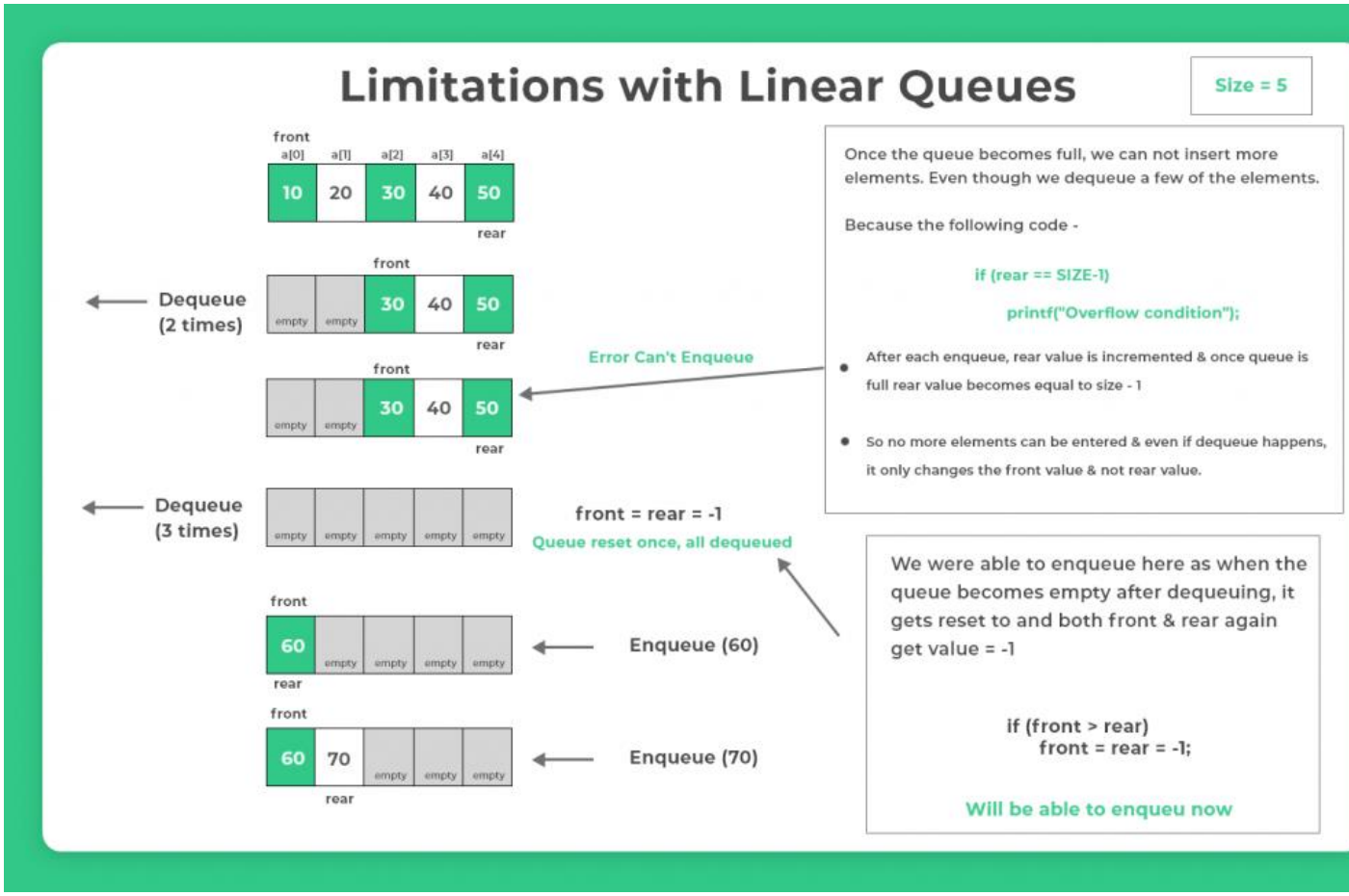
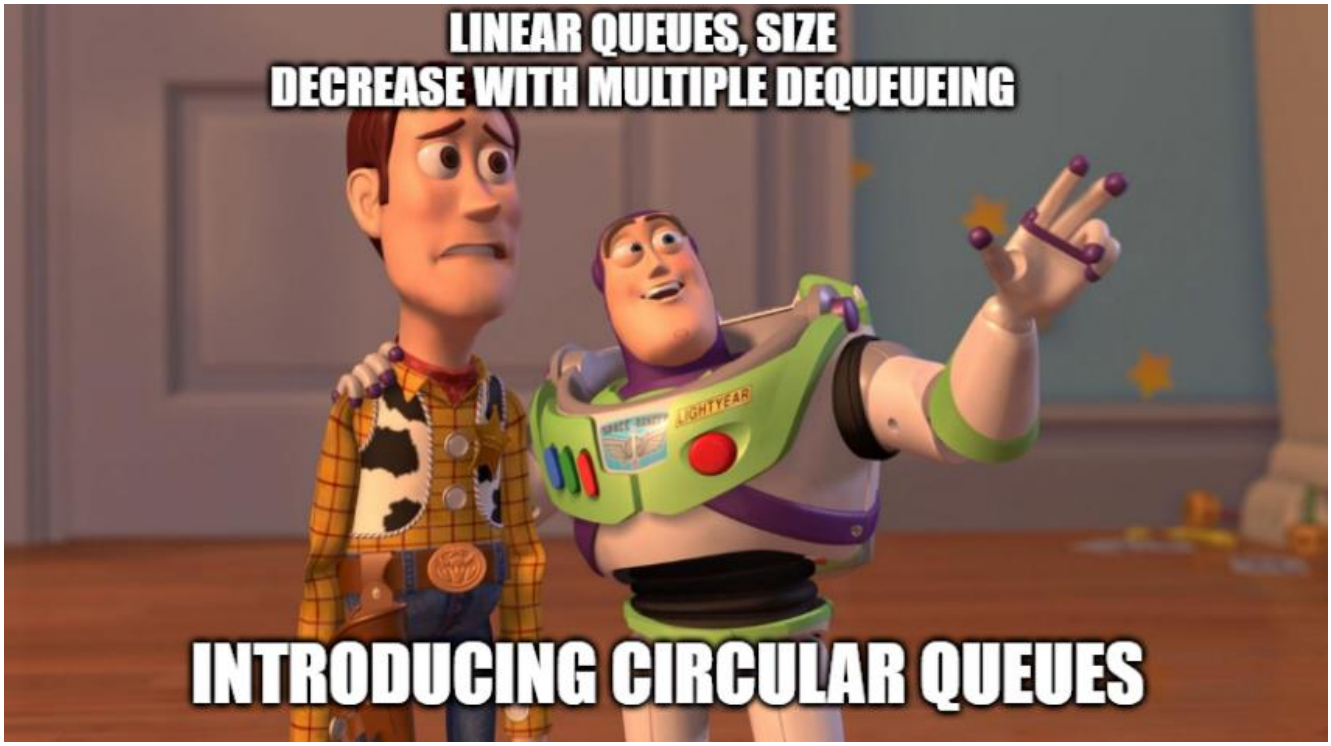
Why Circular Queues are needed ?

When we talk about linear queues, once the queue becomes full, we can not insert more elements. Even though we dequeue a few of the elements, only once all the elements are dequeued then only queue is reset and then new elements can be inserted. The example of the same is shown in image below –

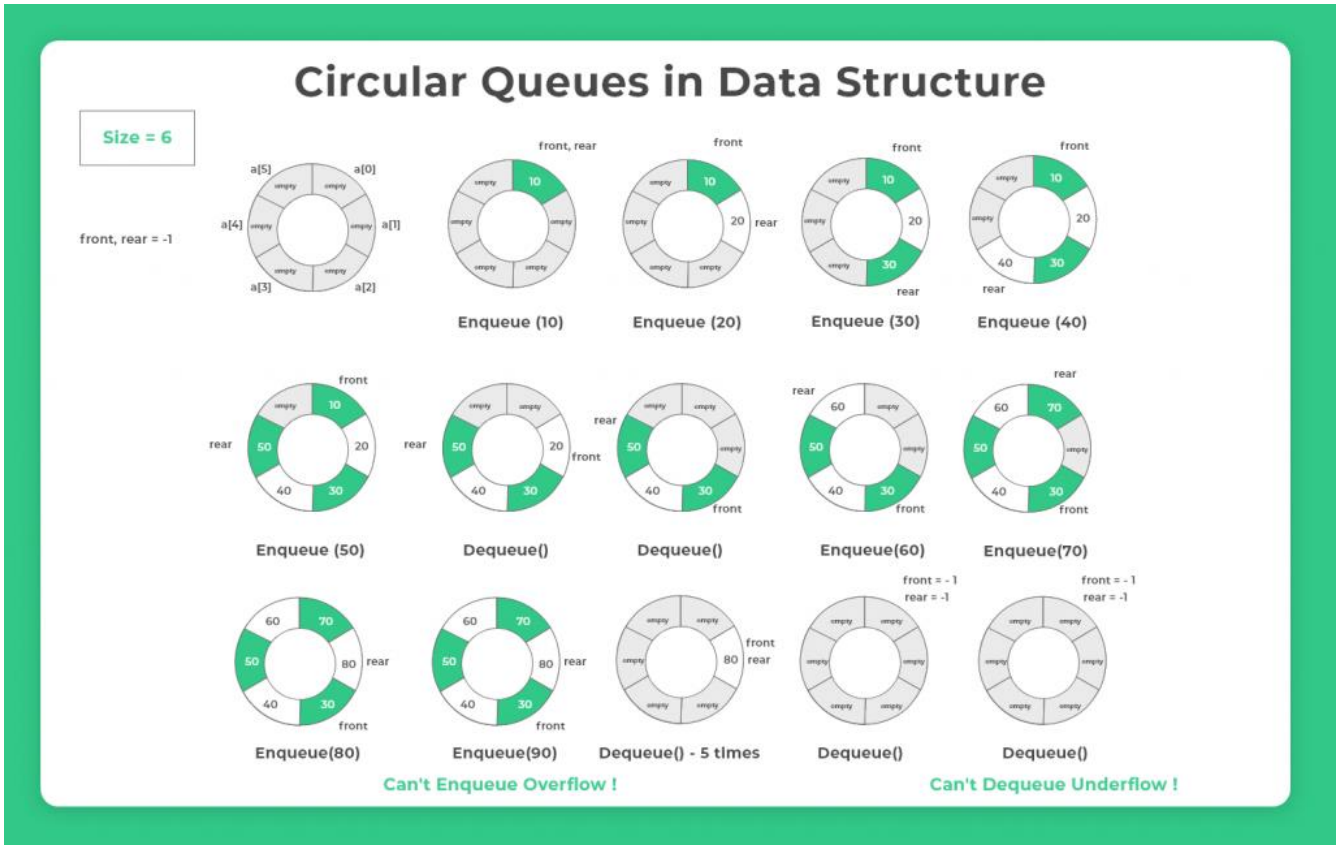


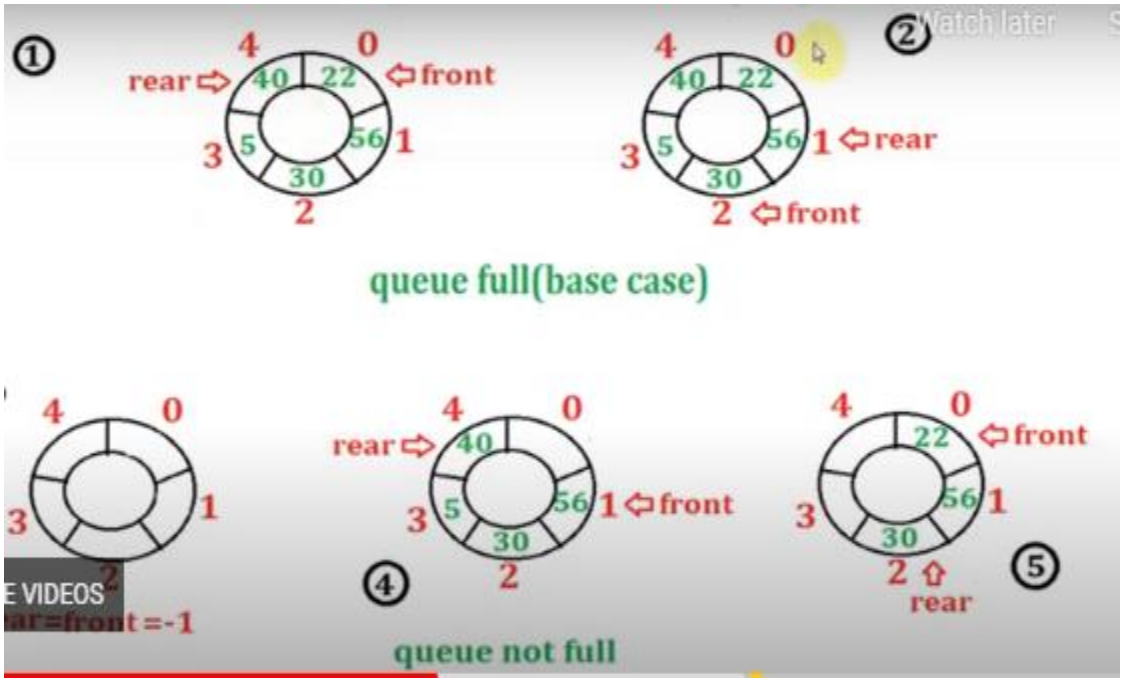


How Circular Queues work in Data Structure

Circular queues work very similarly as linear queues with minor addition and enhancements. Any circular queue as the following –

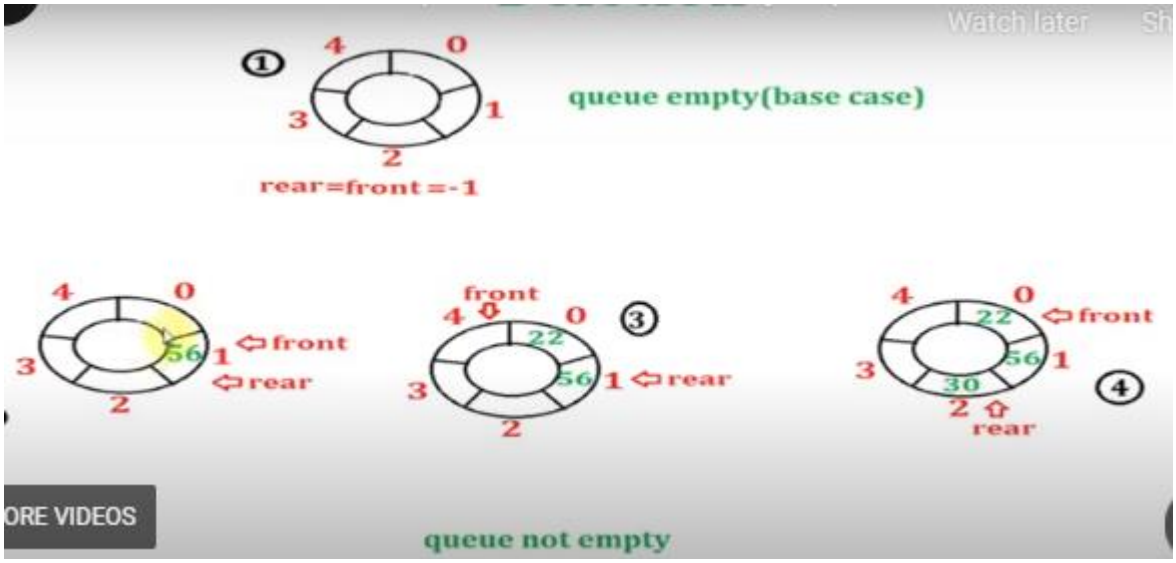
- **Front –** The starting head of the circular queue
- **Rear –** The ending tail of the circular queue
- **Enqueue Operation –** Process of adding a new item in the queue
- **Dequeue Operation –** Process of removing an existing item from the queue
- **Overflow Condition –** When the queue is full
- **Underflow Condition –** When queue is empty
- **Size –** The max number of items the circular queue can hold


Circular Queues in Data Structure
Size = 6
front, rear = -1
a[5]
a[0]
a[4]
a[1]
a[3]
a[2]
front, rear
front
rear
Enqueue (10)
Enqueue (20)
Enqueue (30)
Enqueue (40)
Enqueue (50)
Dequeue()
Dequeue()
Enqueue(60)
Enqueue(70)
Enqueue(80)
Enqueue(90)
Dequeue() - 5 times
Dequeue()
Dequeue()
front = - 1
rear = -1
Can't Enqueue Overflow !
Can't Dequeue Underflow !


①
②
rear
front
queue full(base case)
④
⑤
rear
front
queue not full
E VIDEOS
ar=front=-1

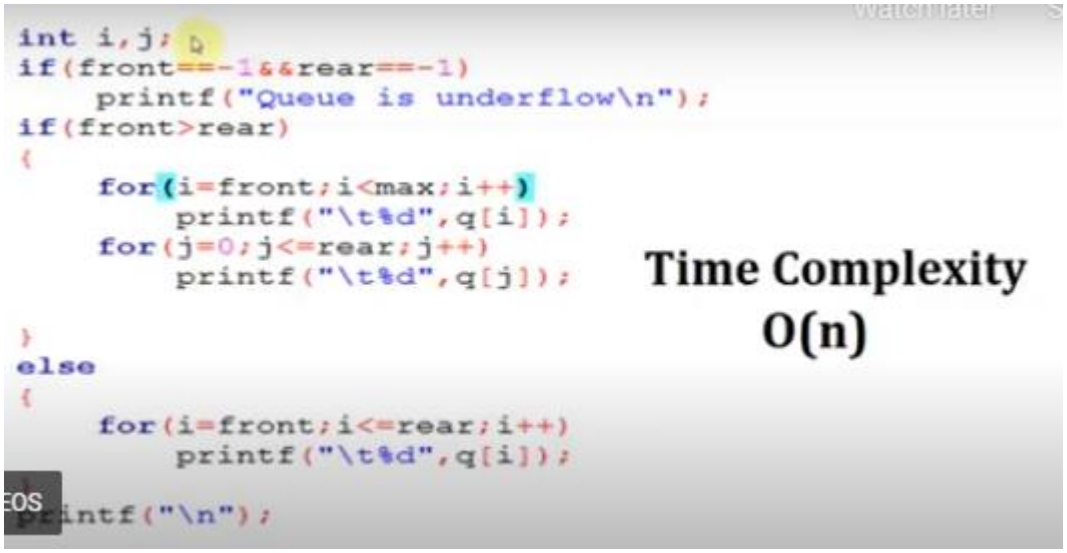
```c
int x;
if((front==0&&rear==max-1)||(rear+1==front))
    printf("Queue is overflow\n");
else
{
    printf("Enter element to be insert:");
    scanf("%d",&x);
    if(rear==-1)
        front=0,rear=0;
    else if(rear==max-1)
        rear=0;
    else
        rear++;
    q[rear]=x;
```

Time Complexity
O(1)




```c
int a;
if(front==-1)
    printf("Queue is underflow\n");
else
{
    a=q[front];
    if(front==rear)
        front=-1,rear=-1;
    else if(front==max-1)
        front=0;
    else
        front++;
    printf("Deleted element is %d\n",a);
```

Time Complexity
O(1)


int i,j;
if(front==-1&&rear==-1)
printf("Queue is underflow\n");
if(front>rear)
{
for(i=front;i<max;i++)
printf("\t%d",q[i]);
for(j=0;j<=rear;j++)
printf("\t%d",q[j]);
}
else
{
for(i=front;i<=rear;i++)
printf("\t%d",q[i]);
printf("\n");
Time Complexity
O(n)