s-day-6-circular-queue-using-class

July 25, 2024

```
[ ]:
```

1 circular queue means

first we make fixed size queue then enqueue till its size(10) if enque when queue is full,then element will not added

then dequeue,then first element(of index 0) will removed

if now we enqueue then element should be added at empty place(here at index 0)

1.1 for go at begining(0) after last element(9)

1.2 main logic is (rear+1)%size_of_queue

1.3 in simple queue it will not go at begining after queue is full

```
[13]: ## fixing size of circular queue

queue=[]
class queue:
    def __init__(self,size):
        self.size=size                              # intialise queue with␣
 ↪given size
        self.queue=[None for i in range(self.size)]      # Create an empty␣
 ↪queue with None values
        self.front=self.rear=-1                     ## Set the front and rear␣
 ↪pointers to -1
    def enqueue(self,value):
            if((self.rear+1)%self.size==self.front):   ##1. QUEUE IS FULL
                print("Queue is full.")
            elif(self.rear==-1):                       #2. EMPTY
                self.front=0
                self.rear=0
                self.queue[self.rear]=value
            else:                                      #3. NORMAL ENQUEUE (GO␣
 ↪TO NEXT POSITION)
                self.rear=(self.rear+1)%self.size
                self.queue[self.rear]=value
    def dequeue(self):
        if(self.front==-1):                         #1. EMPTY
                print("empty")
        elif(self.front==self.rear):               #2. F=R MEANS ONLY ONE␣
 ↪ELEMENT(when f and r are at 0)
            temp=self.queue[self.front]## for dequeue
            self.rear=-1
            self.front=-1
            return temp
        else:                                       #3. NORMAL DEQUEUE
            temp=self.queue[self.front]
            self.front=(self.front+1)%self.size
            return temp



    def printqueue(self):
        if(self.front==-1):                              #1. empty
            print("empty")
        elif(self.rear>=self.front):                     #2. normal
            for i in range(self.front,self.rear+1):
                print(self.queue[i])
        else:                                            #3. circular condition(when␣
 ↪R<F)
            for i in range(self.front,self.size):
                 print(self.queue[i])
            for i in range(0,self.rear+1):
                 print(self.queue[i])

        print("front is",self.front)
        print("rear is",self.rear)

q=queue(5)
q.enqueue(10)
q.enqueue(20)
q.enqueue(30)
q.enqueue(40)
q.enqueue(50)
print("1..")
q.printqueue()

q.dequeue()
q.enqueue(700)
print("2..")
q.printqueue()
```

```
q.enqueue(800)
print("3..")
q.printqueue()
```

```
1..
10
20
30
40
50
front is 0
rear is 4
2..
20
30
40
50
700
front is 1
rear is 0
Queue is full.
3..
20
30
40
50
700
front is 1
rear is 0
```

[14]:
```
def length(self):
        if self.front == -1:  # Queue is empty
            return 0
        elif self.rear >= self.front:
            return self.rear - self.front + 1
        else:
            return self.size - self.front + self.rear + 1
```

[ ]:
```
```