10/03/2024

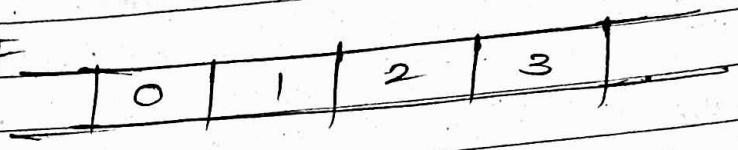
Page No.
Date

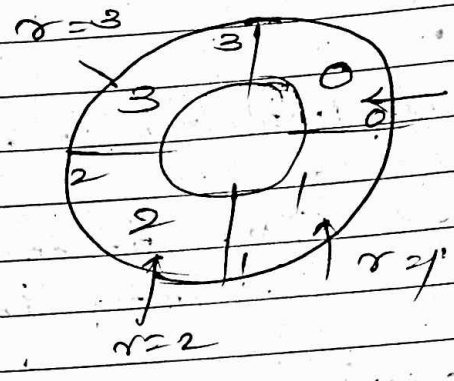
Day 13 OF DSA

Tasks :-

Check Box	Queue
☑	Circular Queue : Implementing a circular queue
☑	priority Queue : Basic priority queue implementation
☑	Implement circular queue operations
☑	Implement basic priority queue operations

* Circular Queue:-





* Implementation:-

* Initialization:-

```
int arr[N];
int rear = -1;
int front = -1;
```

① isEmpty()

```
{
    if( front == -1 & rear == -1)
    {
        return true;
    }
    else
    {
        return False;
    }
}
```

Page No.
Date

② isFull().

```
{
    if ((rear+1)%N == front))
        return true;
    else
        return false;
}
```

③ enqueue(value)

```
{
    if (isFull())
        return;
    else if (isEmpty())
    {   rear = front = 0;
    }
    else
    {
        rear = (rear+1)%N;
    }
    arr[rear] = value;
}
```

④ dequeue ()

```
{
    int x=0;
    if (isEmpty())
    {
        return x;
    }
    else if (rear == front)
    {
        x = arr[rear];
        rear = -1;
        front = -1;
        return x;
    }
    else
    {
        x = arr[front];
        arr[front] = 0;
        front = (front + 1) % N;
        return x;
    }
}
```

⑤ count()

```
{
    return (itemcount);
}
```

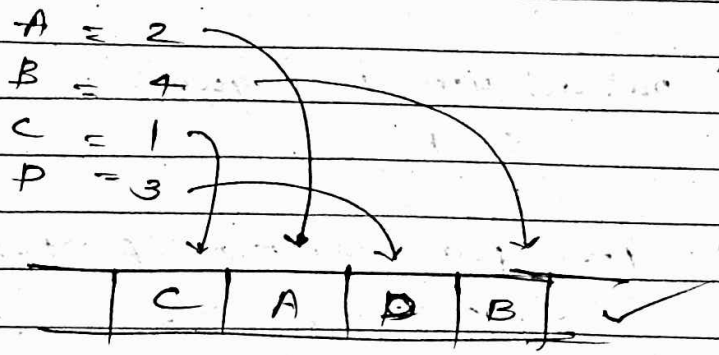
Only we track itemcount in enqueue & dequeue

Here we add that

Page No.
Date

Priority - queue :-

We insert element by priority ...



Implementat^ Using STL :-

```
priority_queue<int> pq;
```

pq

minHeap — Ascending — 1 2 3 4 5

maxheap (By Default) — Descending — 5 4 3 2 1

```
priority_queue<int, vector<int>, greater<int>> pq;
```

Operations :-

1. push()
2. pop()
3. size()
4. top()

(1) Queue Reversal

Approach:-

(1) put all elements of queue in stack.

(2) then pop all element of stack & put it in queue

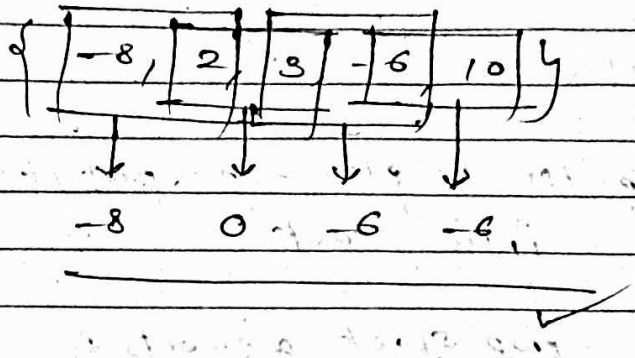
(3) return it.

Time complexity: O(N);
Space complexity: O(N)

Page No.
Date

(2) Find negative integer in every window of size k

→



Approach :-

(1) First we working on 1st windows

(2) checking negative elements & storing their index in double ended queue

(3) then checking. if (dq have some element) then we put it in new vector ans.
else, we put 0.

(4) & we do this for remaining windows.

(3) Reverse First k elements of queue.

→

Approach:-

(1) pop 1st k elements & put it into stack.

(2) Then pop stack elements & put it into queue.

(3) Then pop Take n-k elements from queue & push back within it.

That's it.

Time Complexity: O(n)
Space complexity: O(k)

✓

Page No.
Date

④ First non-repeating character in stream

eg. a a b c

Approach :-

① First we count the frequency of stream
Using map

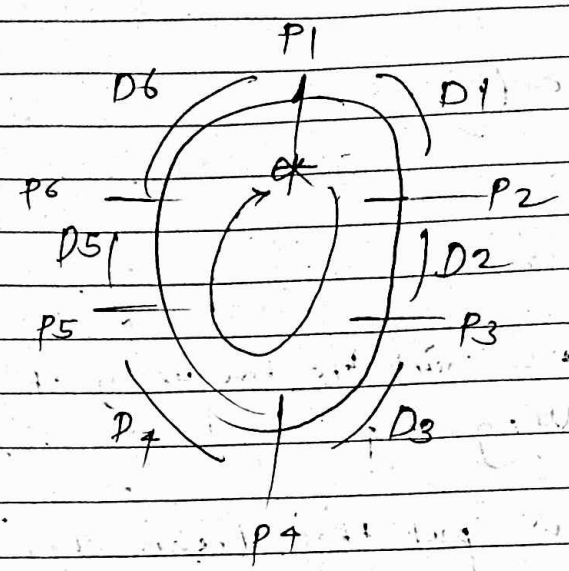
② Then we put that stream characters in
queue one by one

③ then we traverse queue
& check every characters count
if it's count is > 1
then we pop it

④ otherwise we store in ans string

yep, we clear-it :)

(5) + Circular tour :-



Approach :-

(1) Find out balance mean how much petrol we need

```
balance += p[i].petrol - p[i].distance
```

(2) if balance ~~<~~ 0

```
deficit += balance;
start = i+1
balance = 0;
```

(3) Finally if deficit + balance >= 0

return true

else

return False

✓