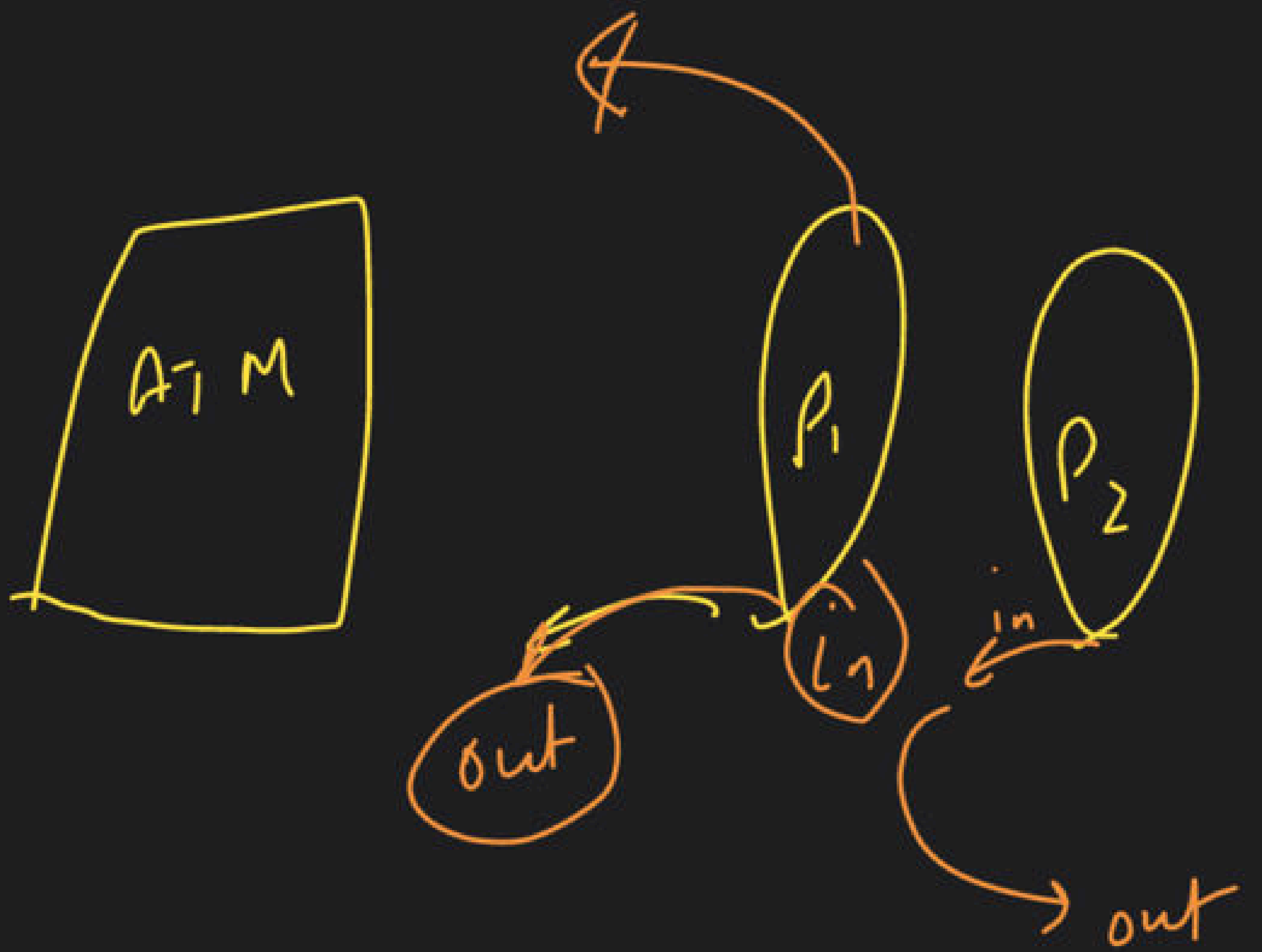
Queue - Class 1 [JOIN HERE]

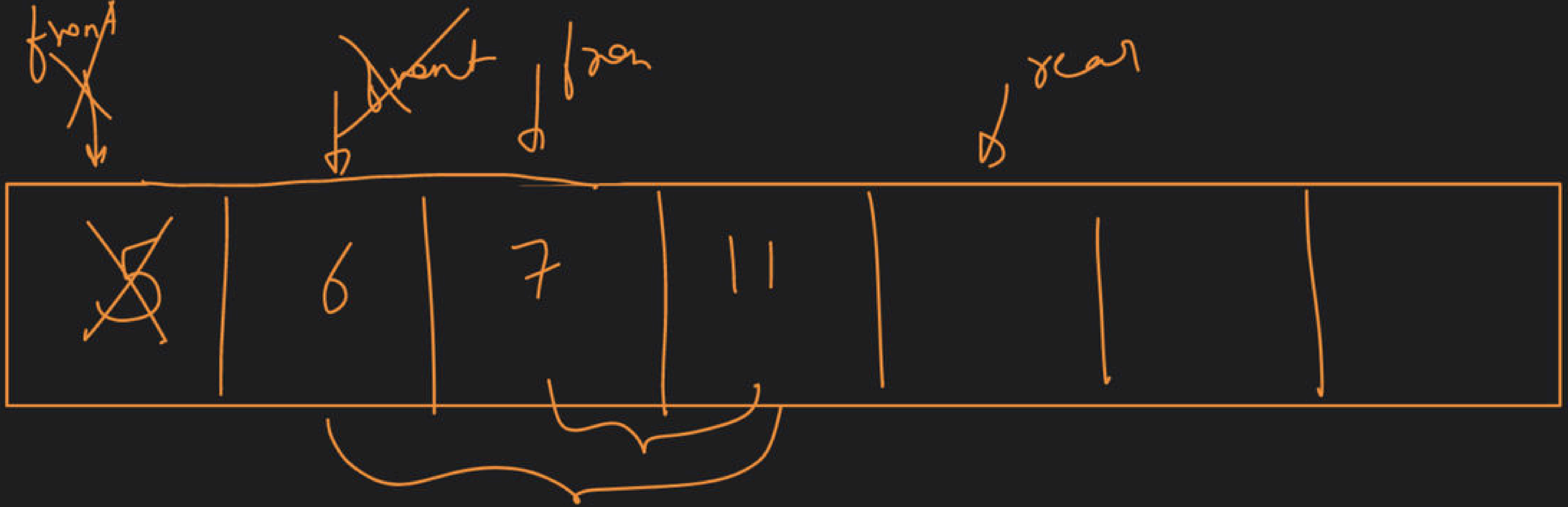
Special class

Love Babbar • Apr 29, 2023

Queue → Data Structure → FIFO

(first in first out)





push(5)

push(6)

push(7)

push(11)

pop()

pop

insertion
↳ rear

removed
↳ front

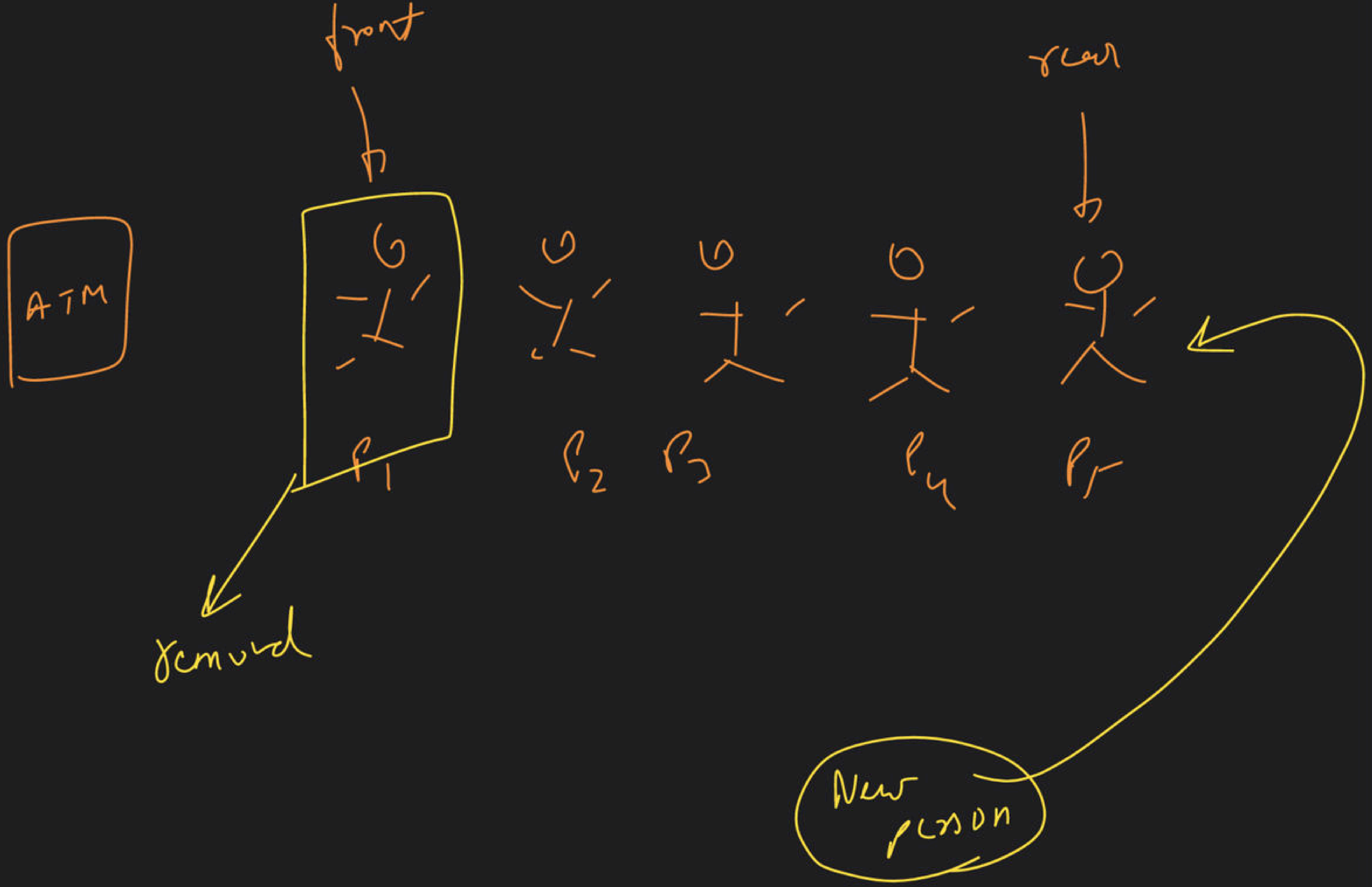
push →

pop →

front →

empty →

size →

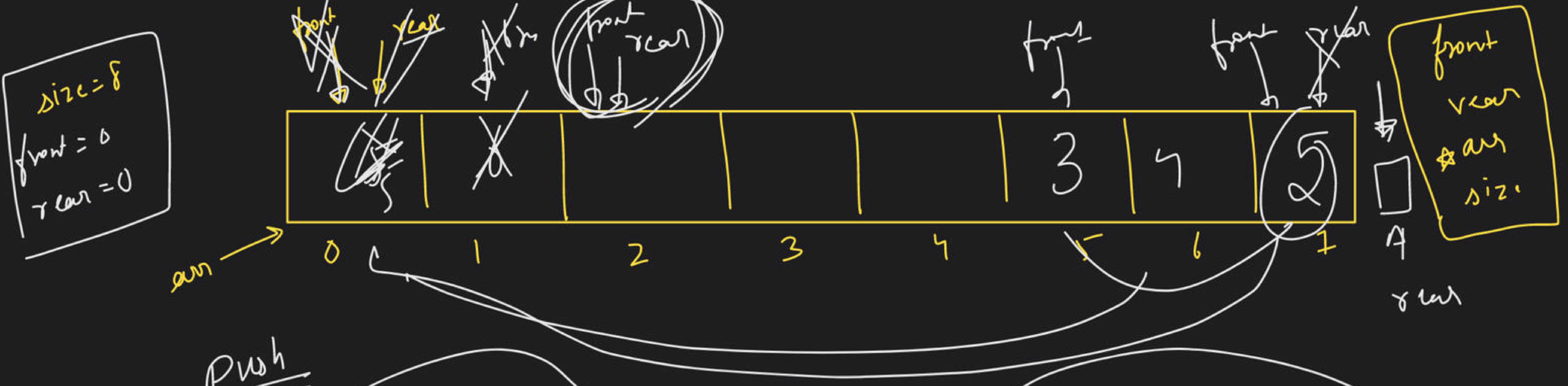


class Queue

- push ✓
- pop ✓
- getfront() ✓
- isEmpty() ✓

Queue

- vector | arr
- L.L



Push

```
if Queue full
    bolo → Q is full
else
    insert data
```

```
if (rear == size)
    cout << Q is full;
else
    arr[rear] = data
    rear++;
```

rear+1

arr(rear) = data

pop

if (Empty)
↳ Q is Empty

else
pop karo

```
if ( front == rear )
    cout << Q is Empty
else
{
    arr[front] = -1;
    front ++;
    if ( front == rear )
    { front = 0
      rear = 0;
    }
}
```

front rear
front
rear

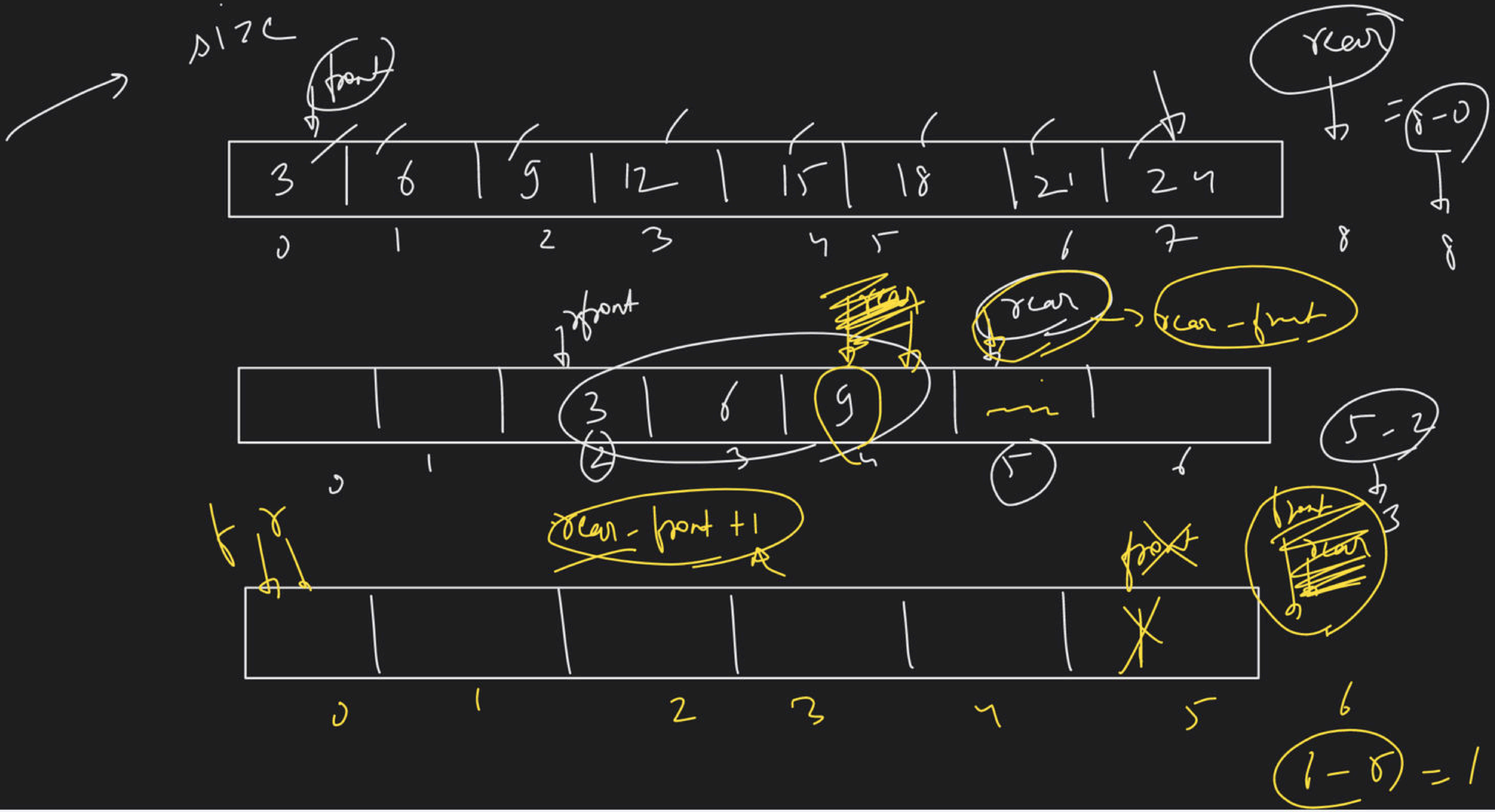
getFront ( )

```
if (q Empty)
    q is empty
else
return arr[front];
```

```
if (front == rear)
    cout ________
else
    return arr[front];
```

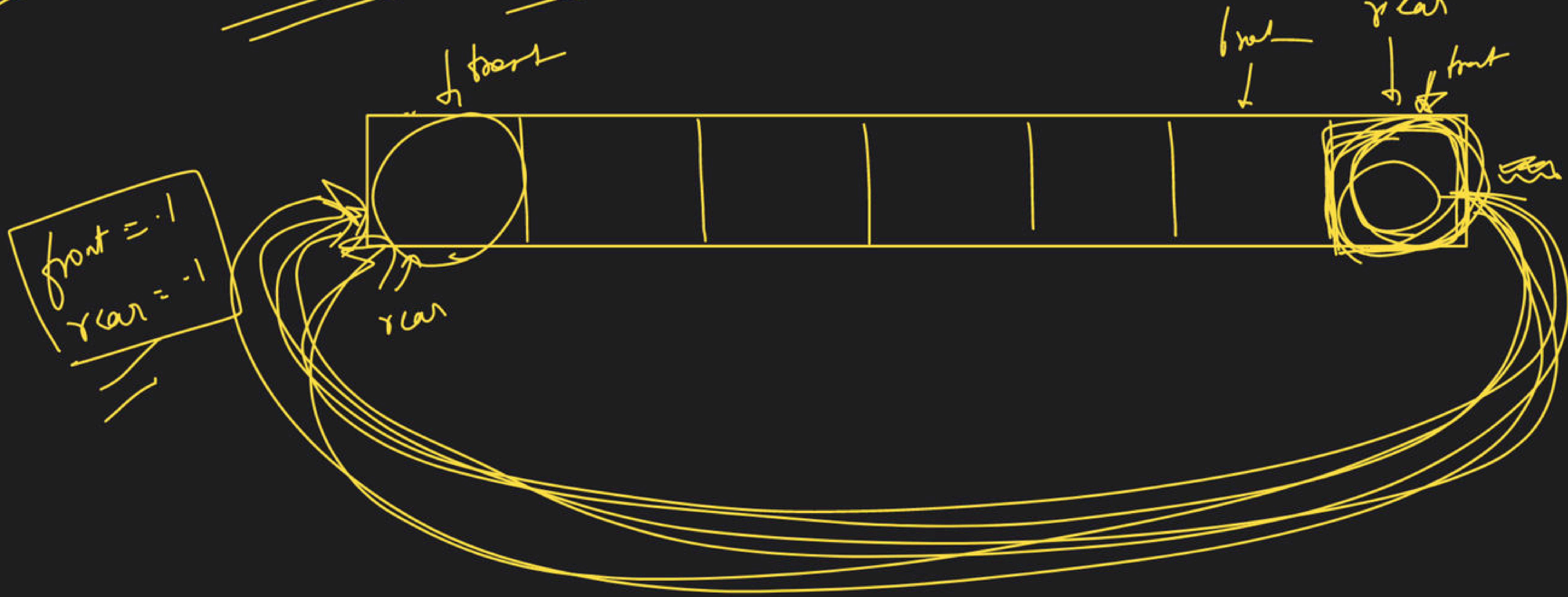
isEmpty()

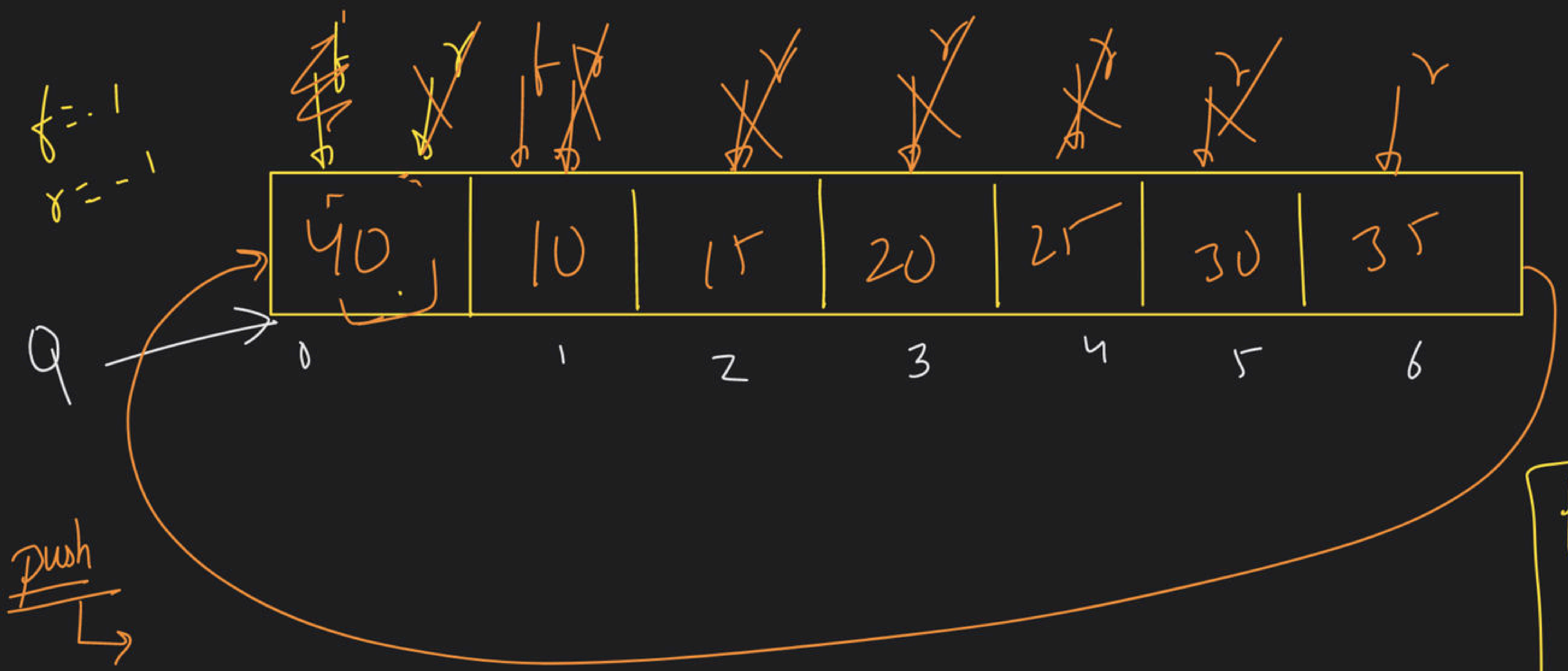
```
if (front == rear)
    return true
else
    return false.
```



size
front
rear
3
6
9
12
15
18
21
24
0 1 2 3 4 5 6 7 8 8
= 8 - 0
front
rear
rear - front
3
6
9
0 1 2 3 4 5 6
5 - 2
3
rear - front + 1
f
r
front
rear
0 1 2 3 4 5 6
1 - 0 = 1

Circular Queue





full

circular nature

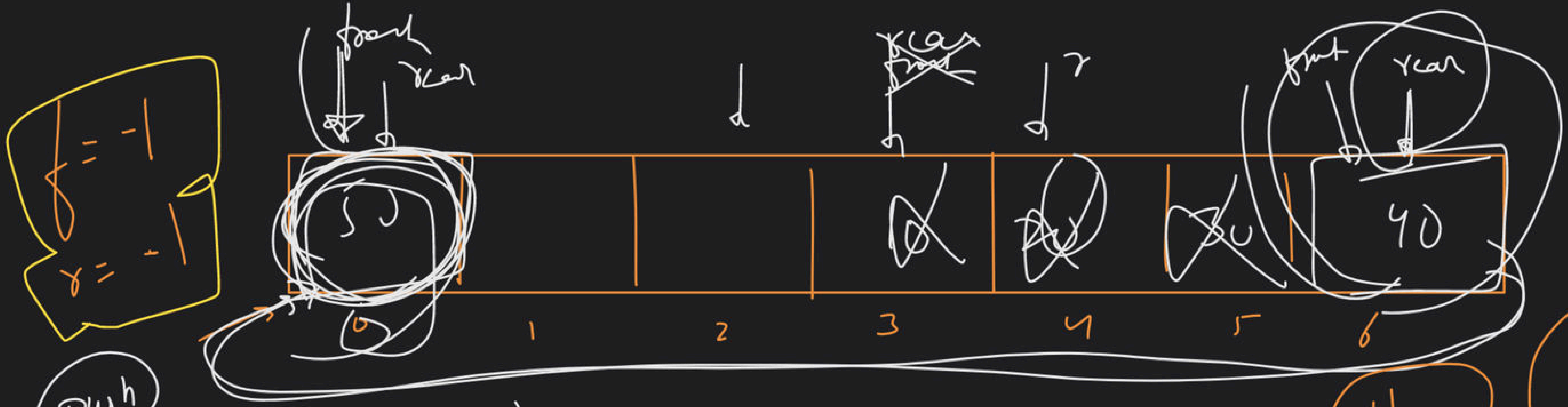
first element insertion

//

single element removed

↓

[illegible] ke handle



push

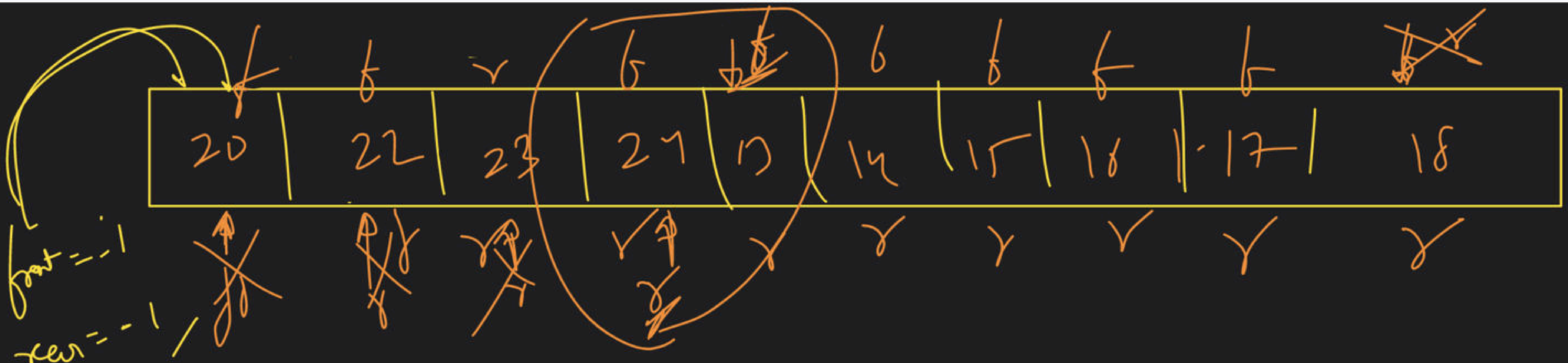
1 → if (Q full) → cout << Q is full

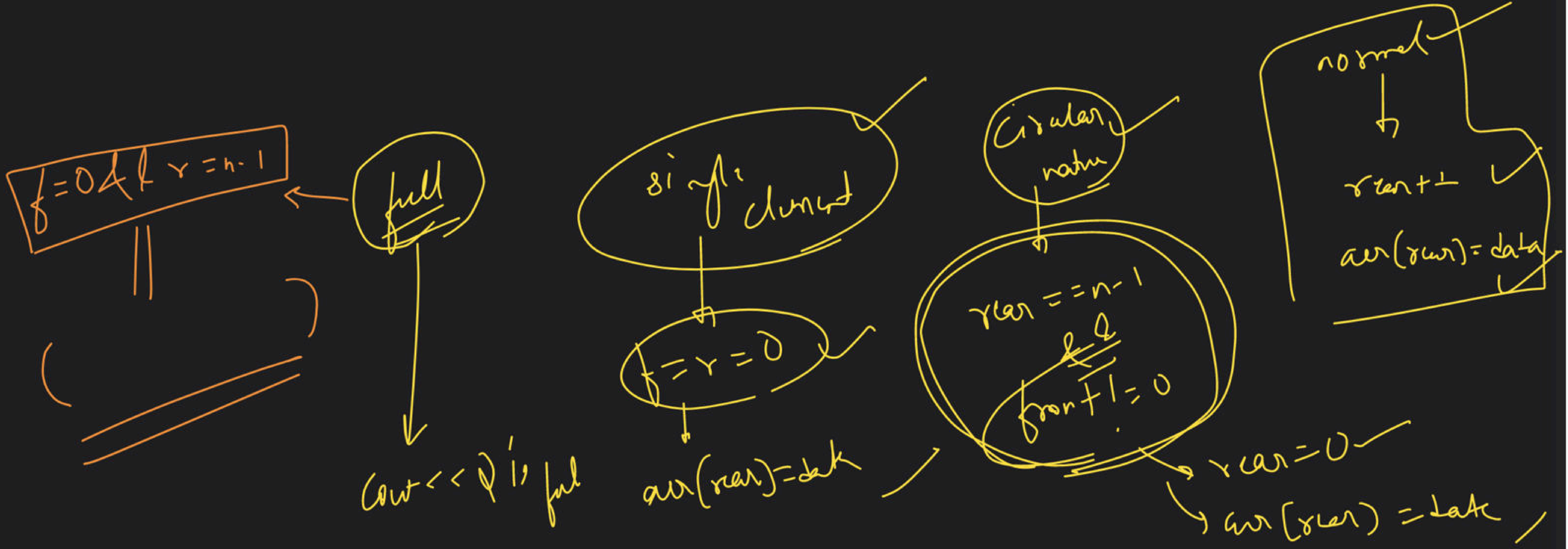
→ first element → f = r = 0 → insert

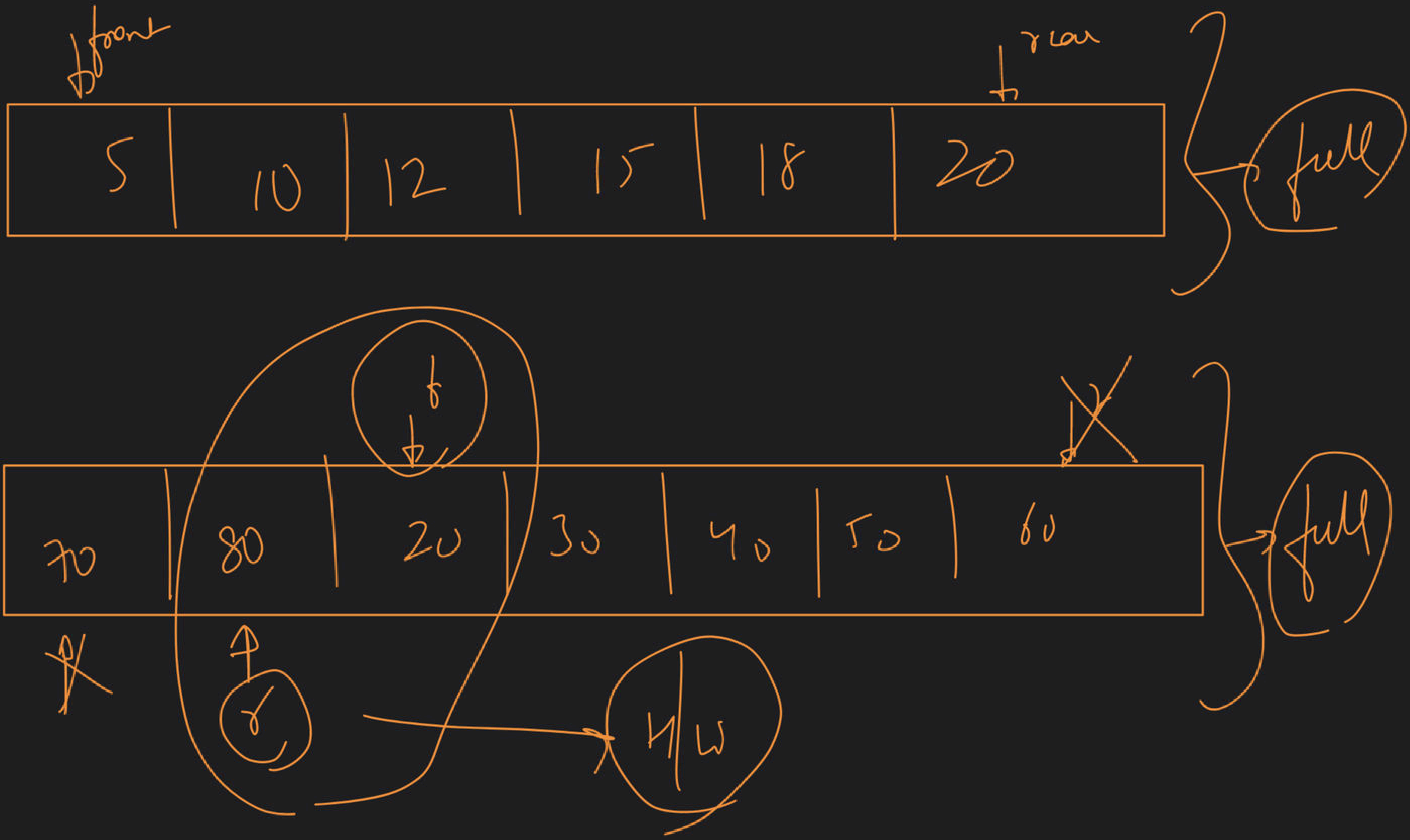
→ circular nature → (rear == n-1 && front != 0) → rear = 0 → insert

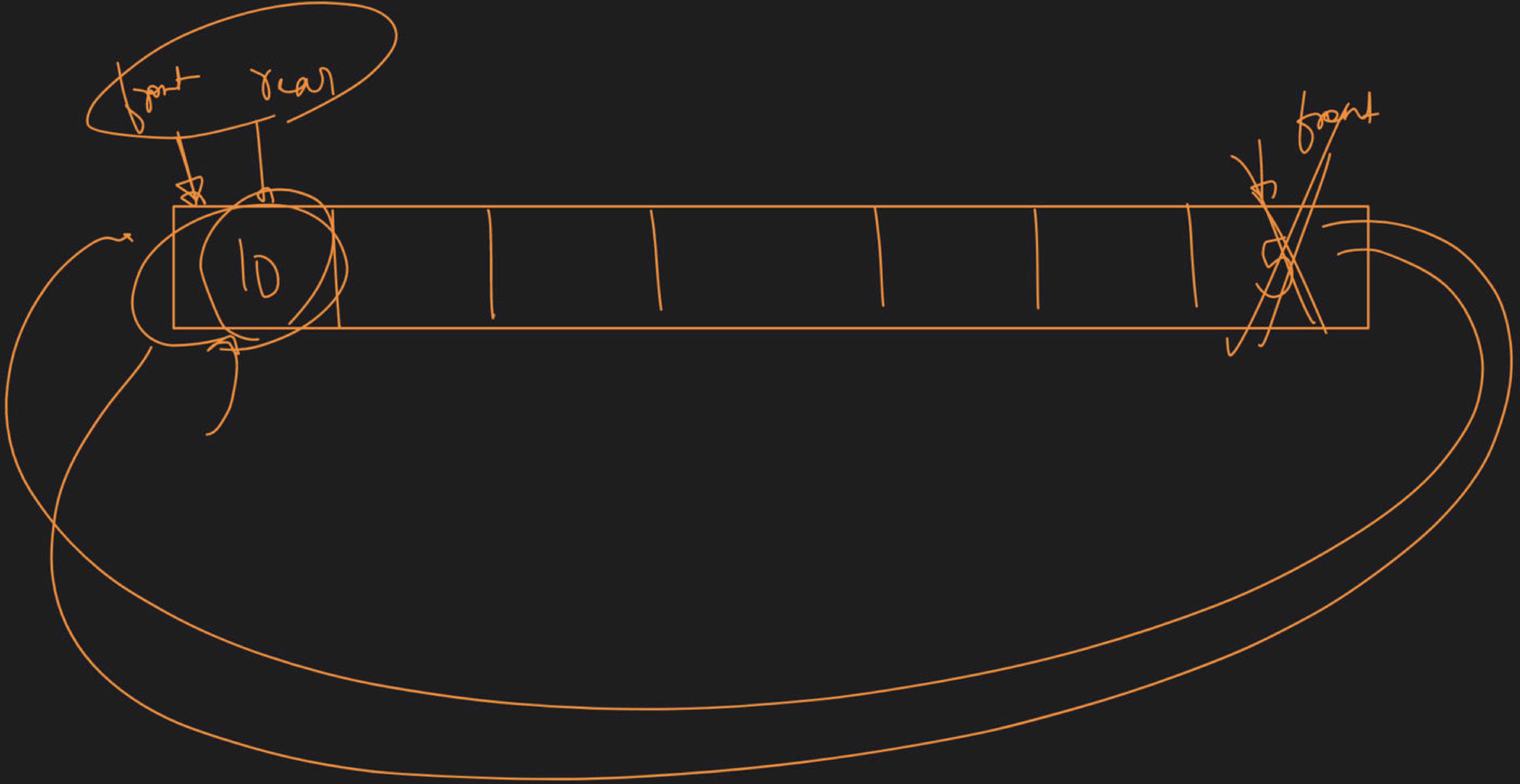
→ default → rear ++ → arr[rear] = data

t/w → use modulus in this logic

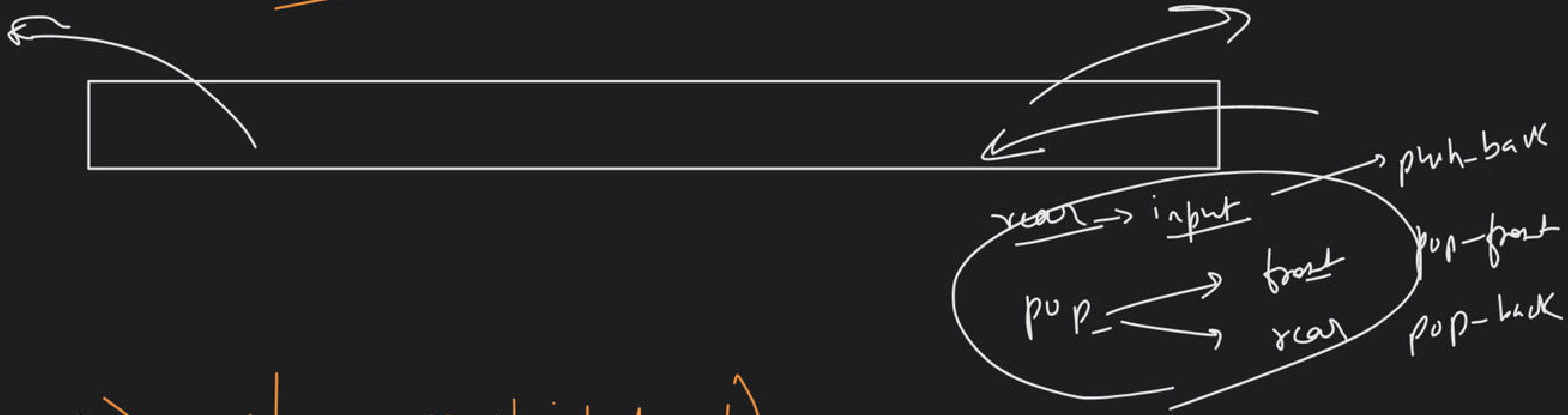

20
22
23
24
13
14
15
16
17
18
front = -1
rear = -1


f = 0 && r = n-1
full
cout << Q is full
single element
f = r = 0
arr(rear) = data
circular nature
rear == n-1 && front != 0
rear = 0
arr[rear] = data
normal
rear+1
arr(rear) = data

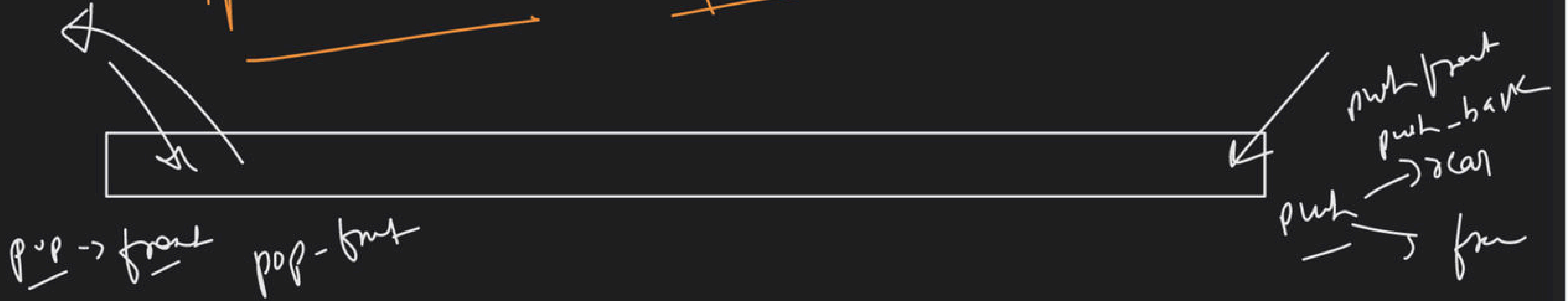

front
rear
5
10
12
15
18
20
full
f
r
70
80
20
30
40
50
60
full
H/W


front
rear
10
front

① i/p restricted Queue



(2) o/p restricted Queue



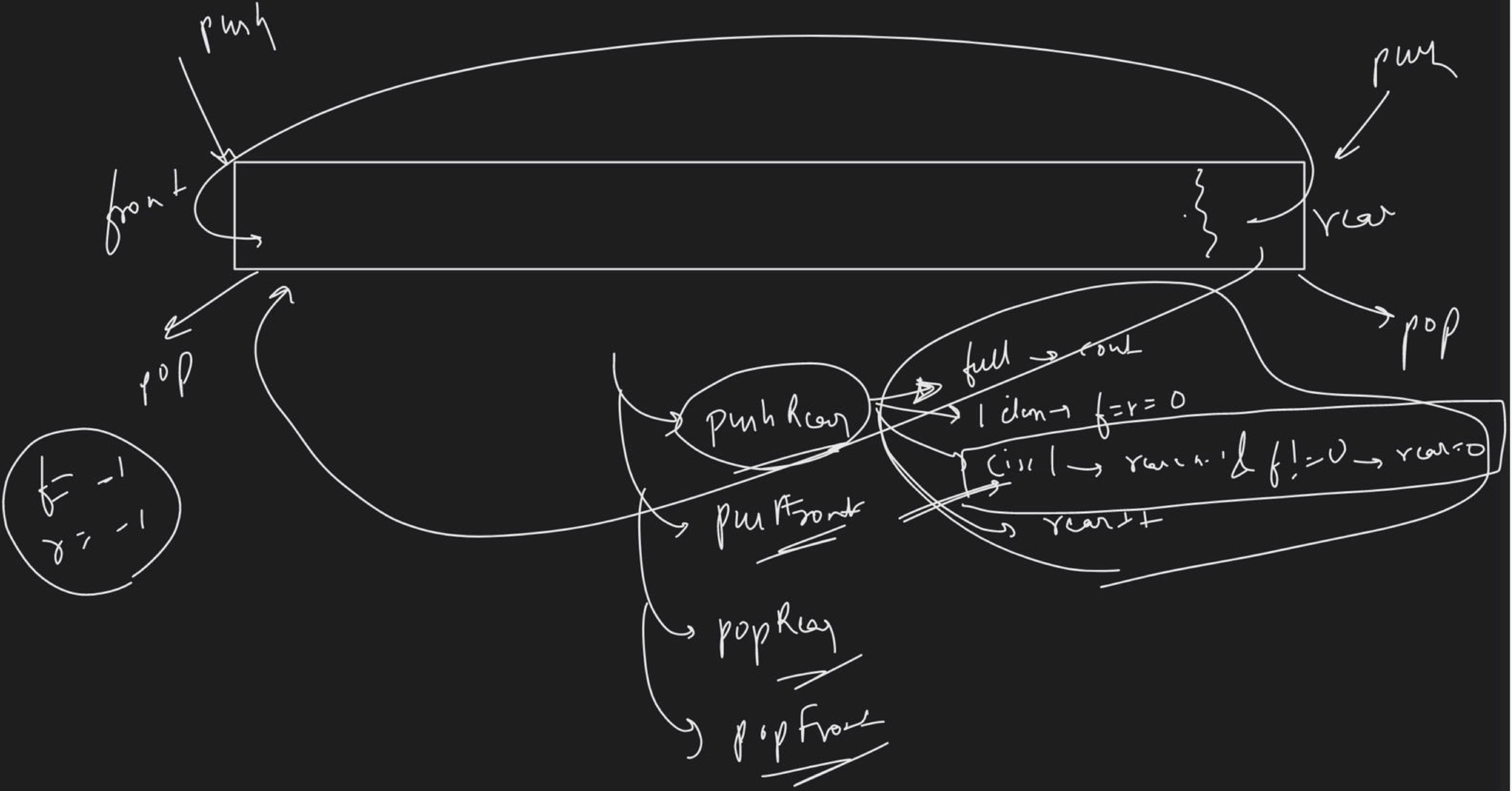
Doubly Ended Queue

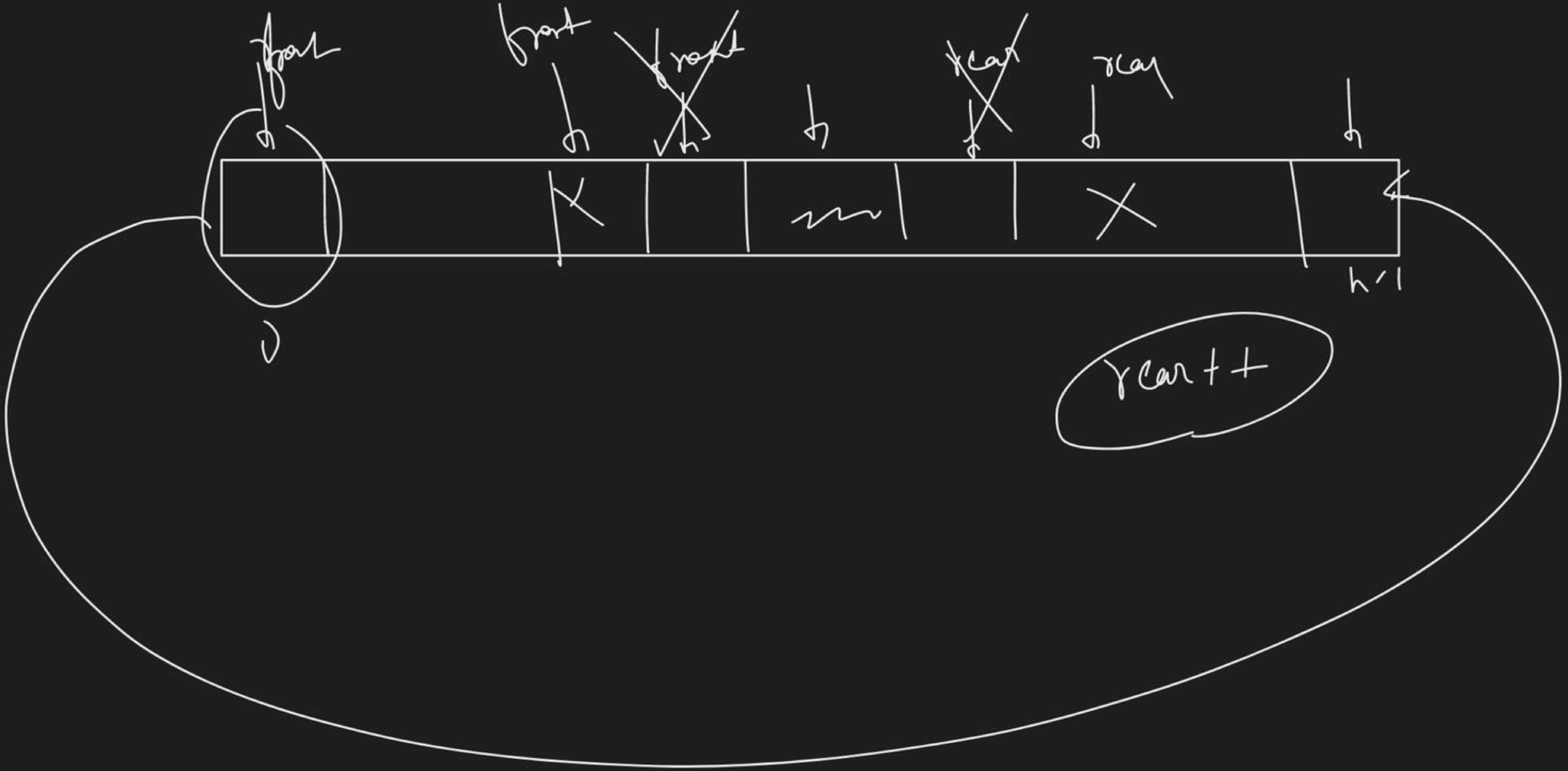
deque

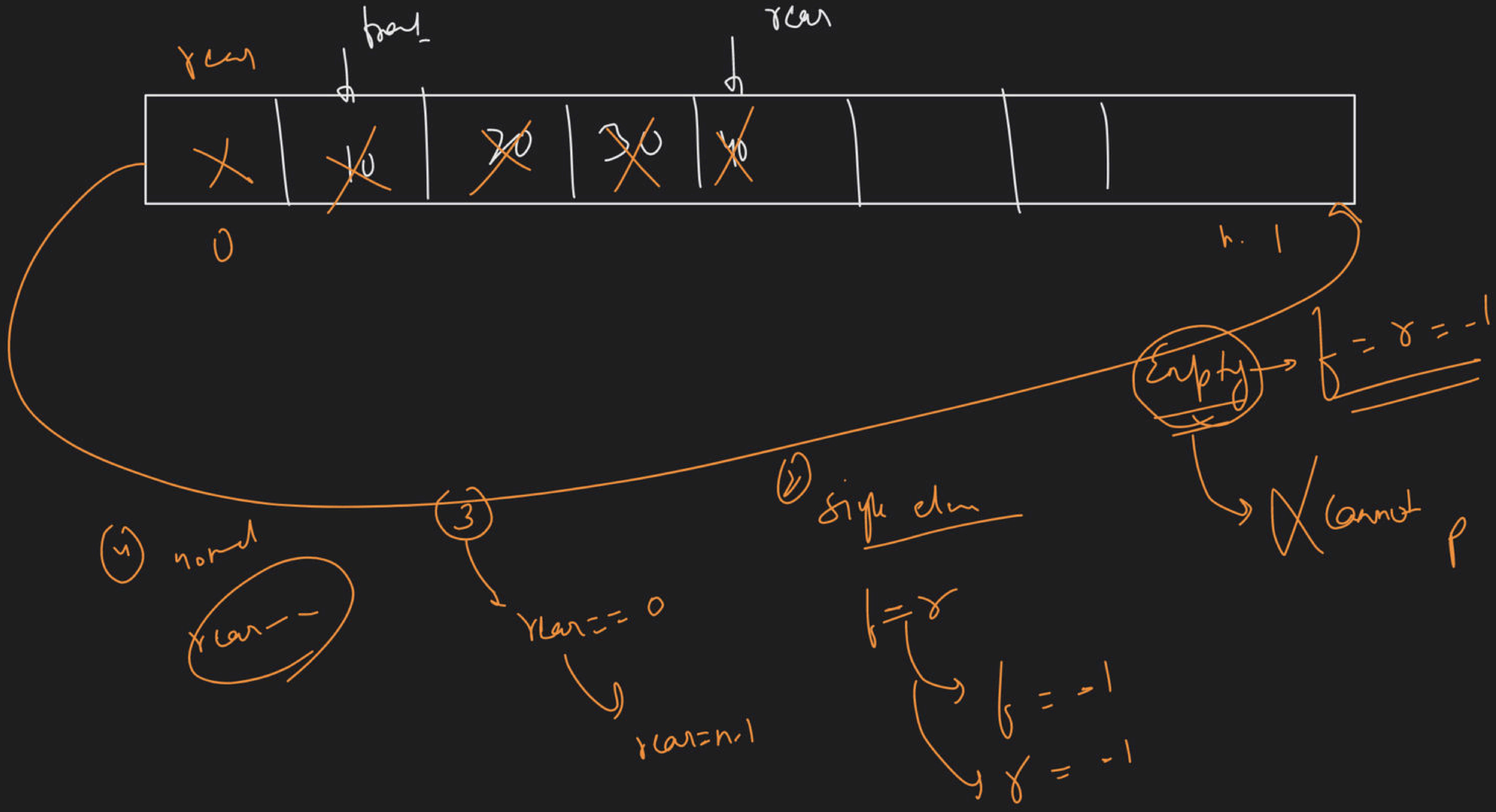
dequeue → pop

enqueue → insert
→ push

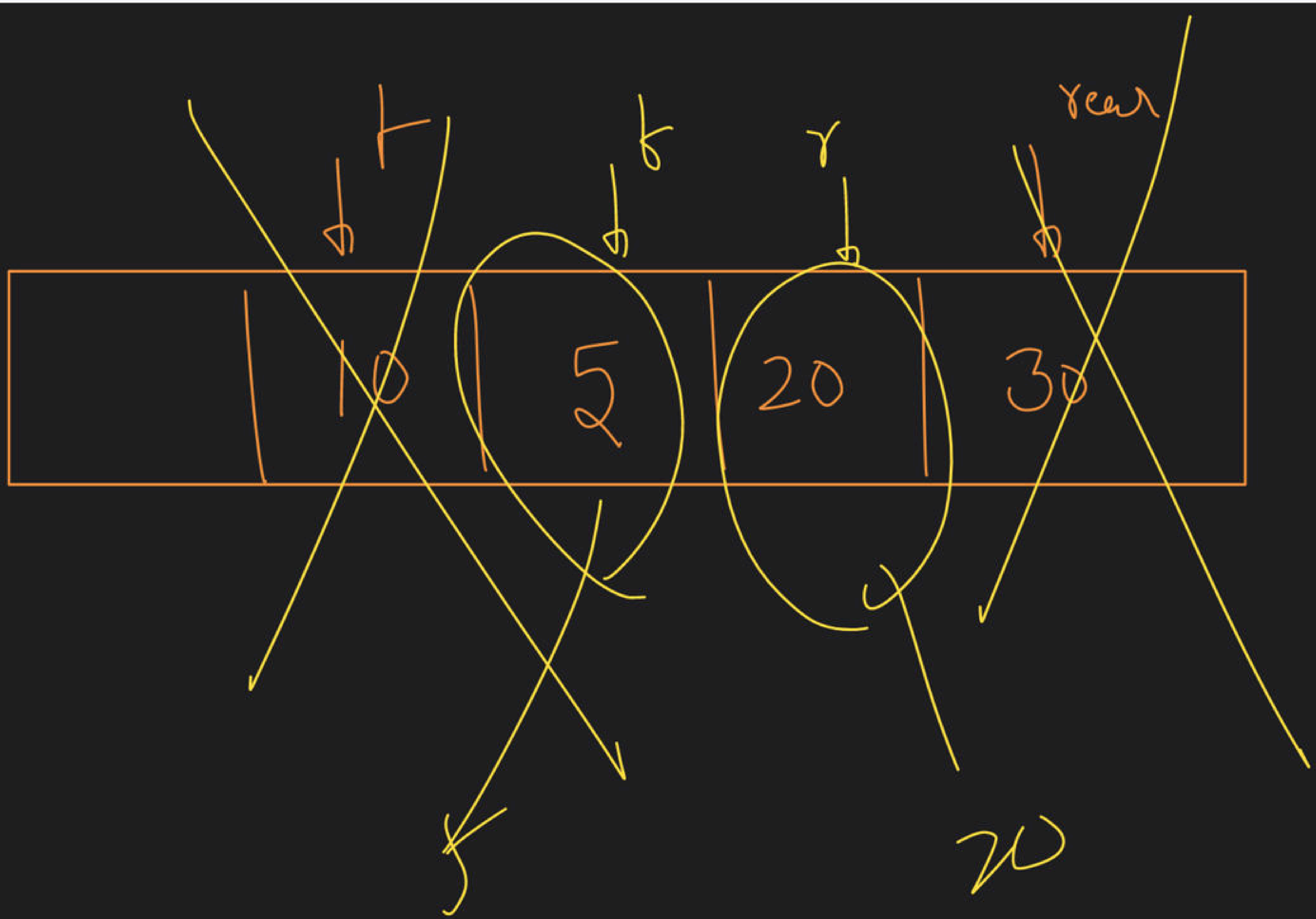
2 min


push
push
front
rear
pop
pop
f = -1
r = -1
pushRear
full → cout
1 elem → f=r=0
circl → rear=n-1 & f!=0 → rear=0
→ rear++
pushFront
popRear
popFront


front
front
~~front~~
~~rear~~
rear
0
n-1
rear++



rear
front
rear
10
20
30
40
0
n-1
Empty
f = r = -1
cannot p
2 single elem
f = r
f = -1
r = -1
3
rear == 0
rear = n-1
4 normal
rear--



f
f
r
rear
10
5
20
30
5
20