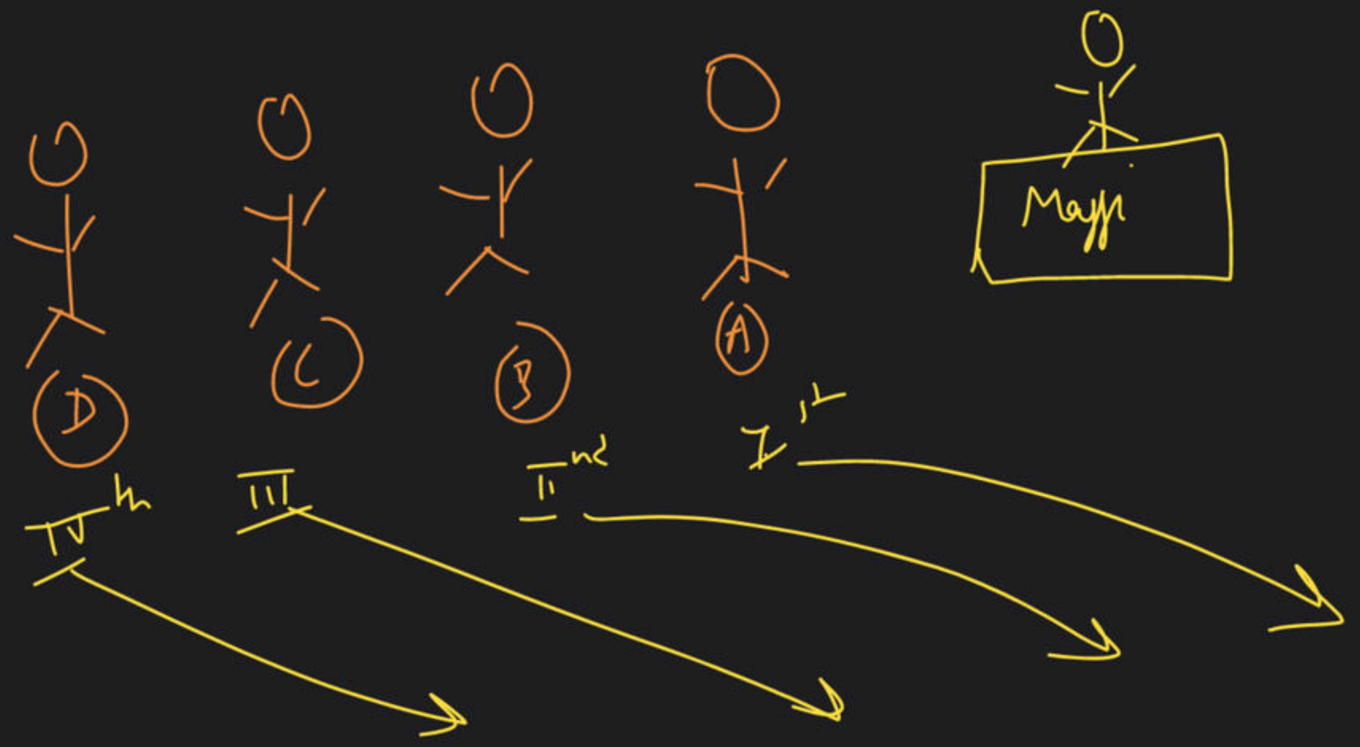
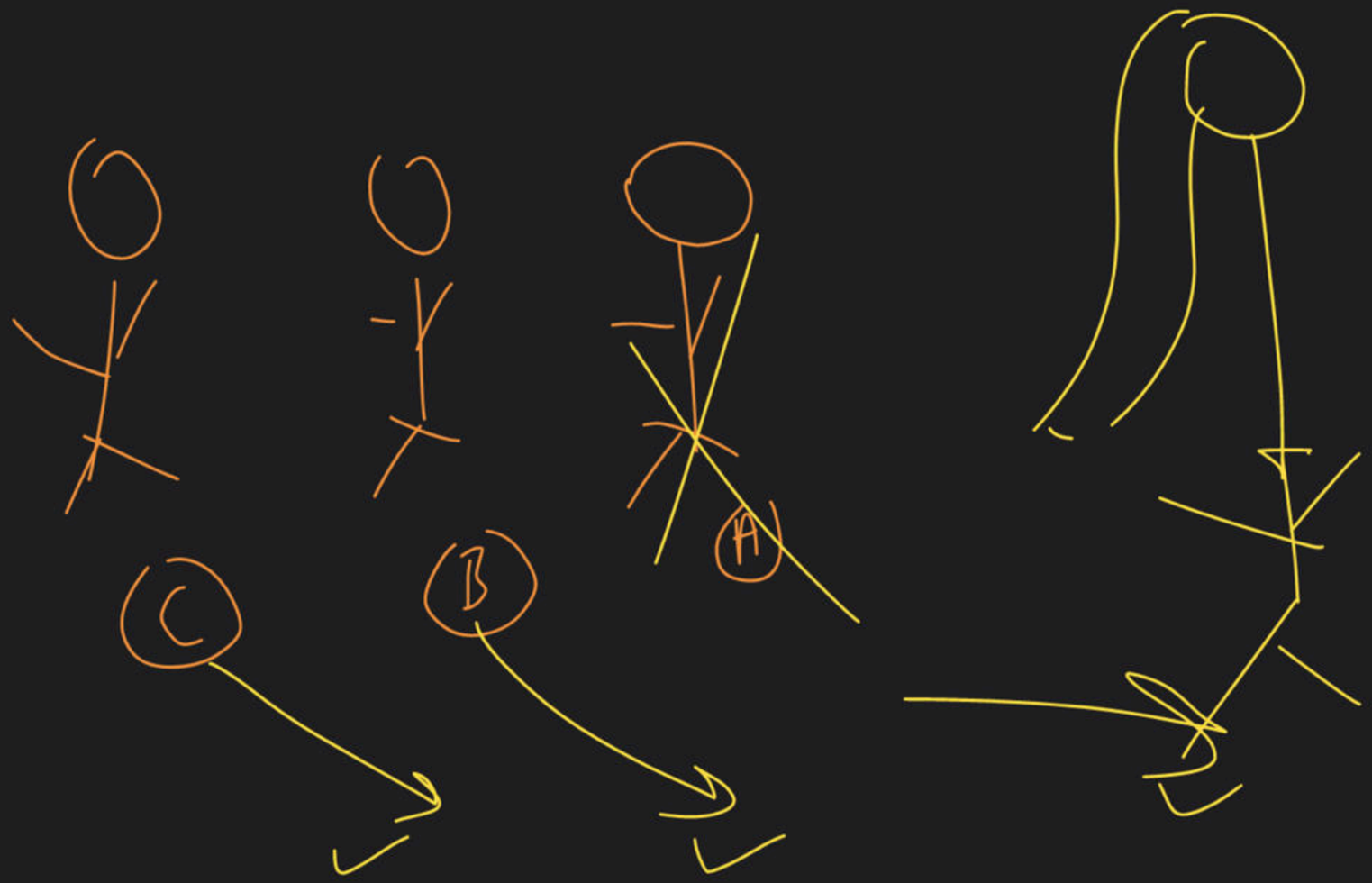
Queue Class - 1 [Join Here]

Special class

Love Babbar • Nov 21, 2023

→ Queue: D.S ⟶ FIFO (first in first out)

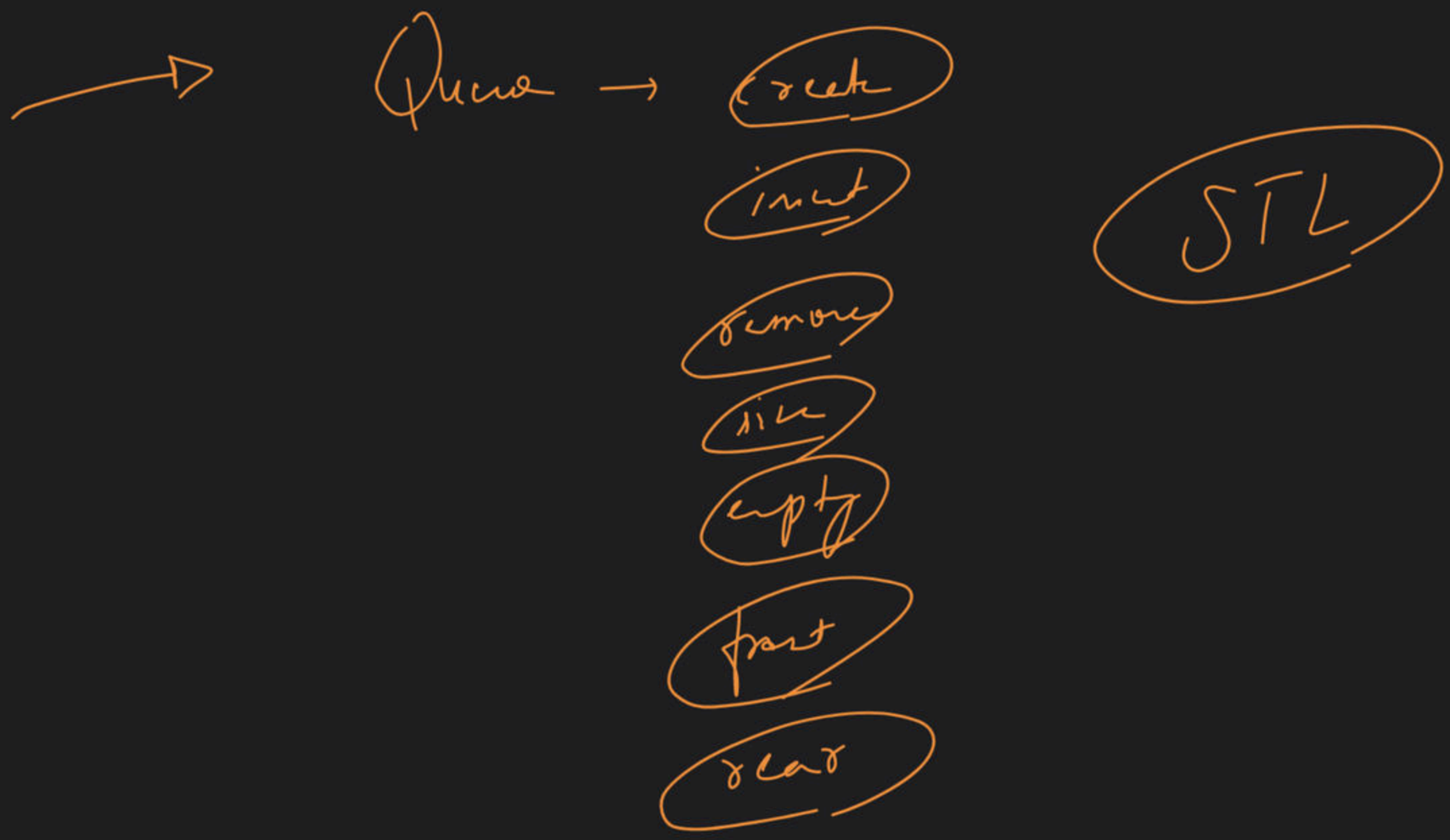



C
B
A

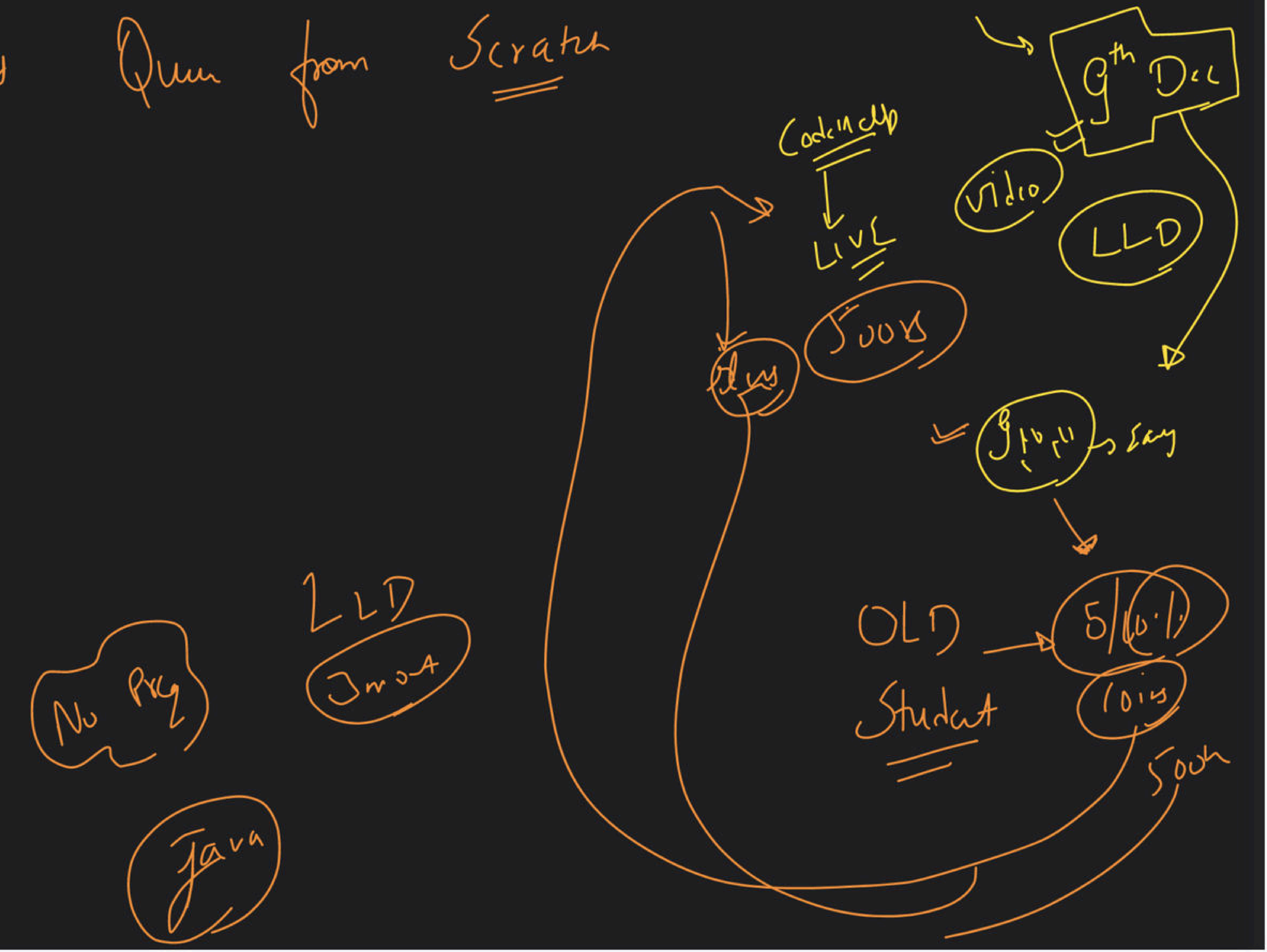
Queue

→ Stack → LIFO

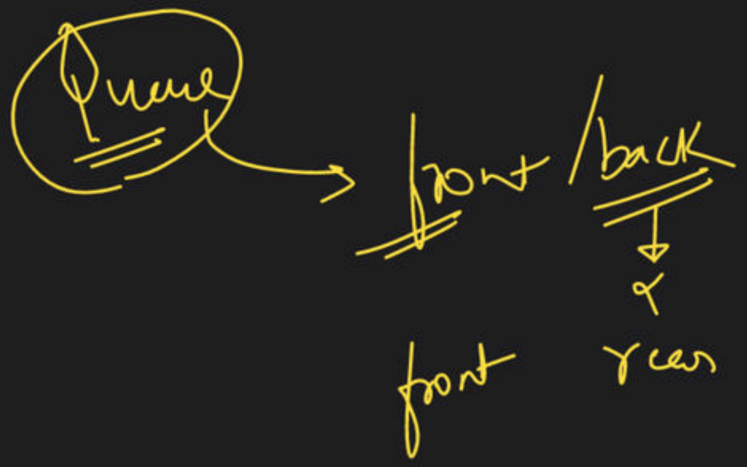
→ Queue → FIFO

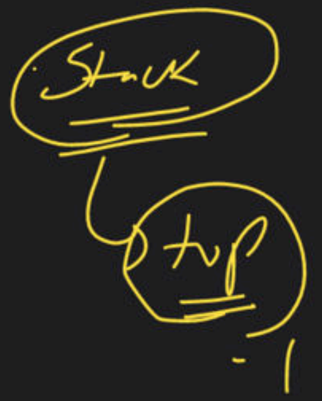

Queue
create
insert
remove
size
empty
front
clear
STL

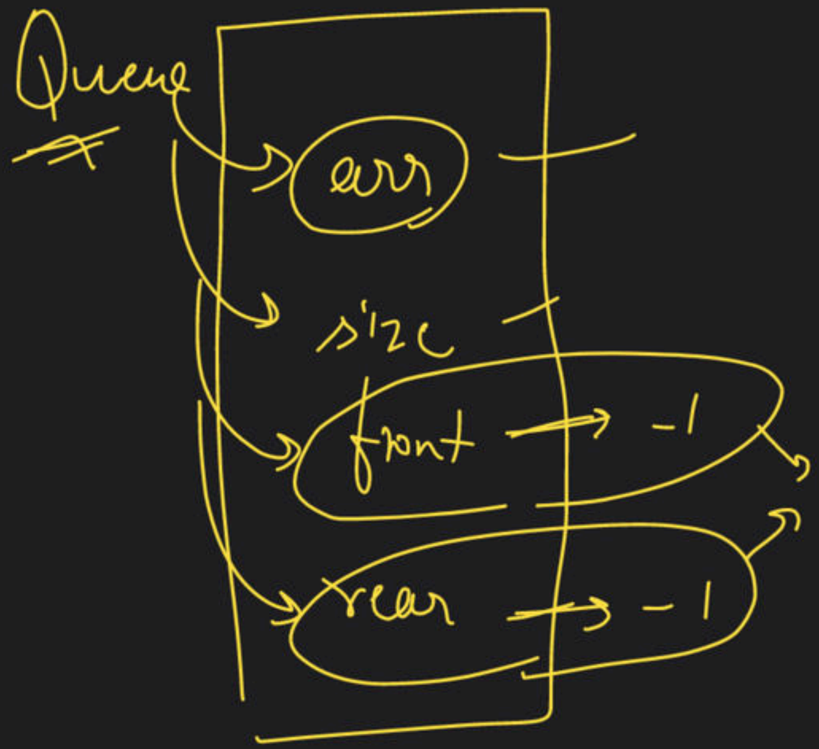
→ Implement Queue from Scratch

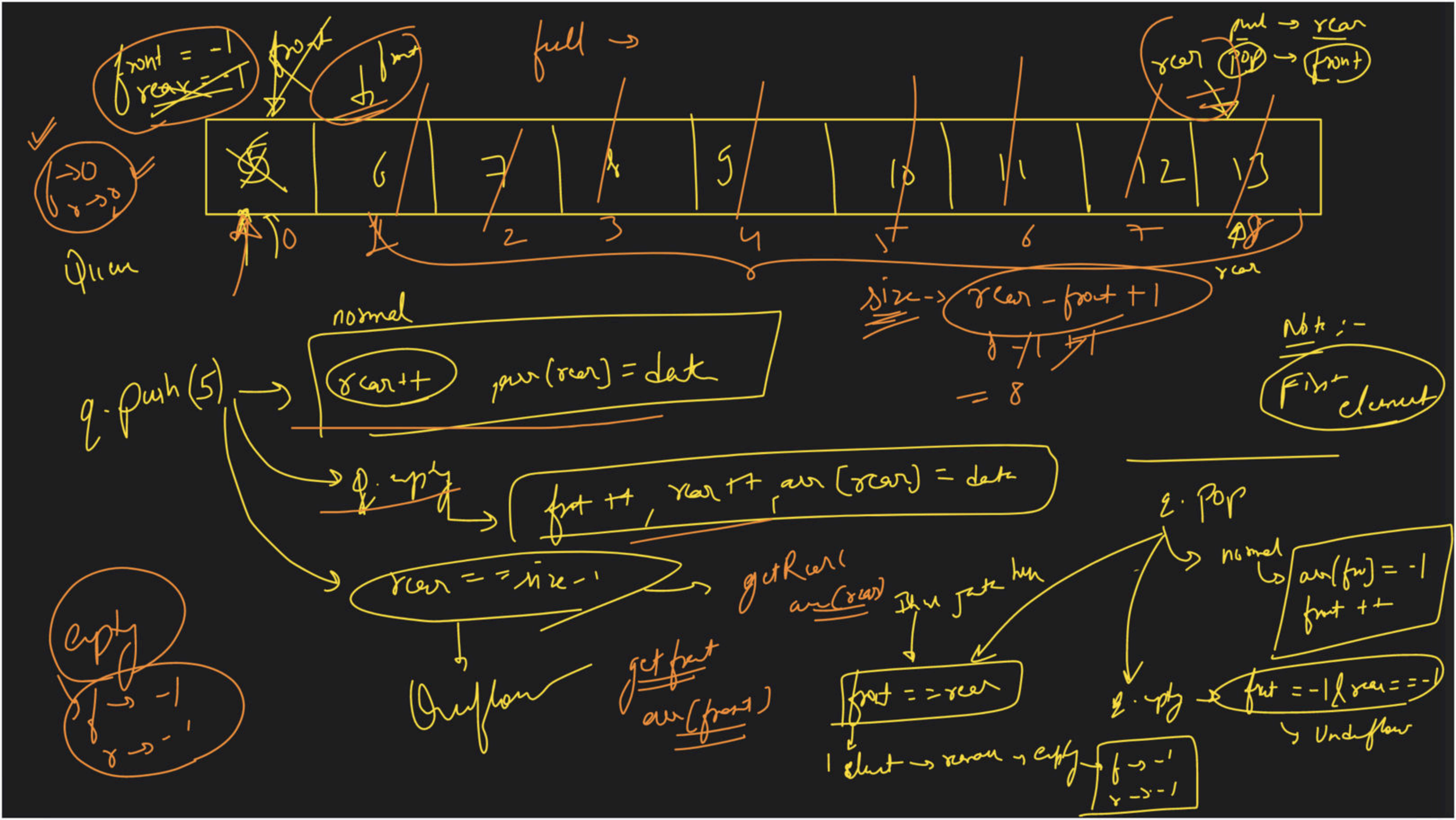


→ Implement Queue from Scratch








front = -1
rear = -1
front
front
full →
rear
pop → front
full → rear
Queue
0
1
2
3
4
5
6
7
rear
size → rear - front + 1
= 8
normal
rear++
arr[rear] = data
Note :-
First element
q.push(5)
q.empty
front ++, rear ++, arr[rear] = data
rear == size - 1
Overflow
getRear()
arr[rear]
get front
arr[front]
Empty
front → -1
rear → -1
front == rear
1 element → rear → empty
front → -1
rear → -1
q.pop
normal
arr[front] = -1
front ++
q.empty
front == -1 || rear == -1
Underflow

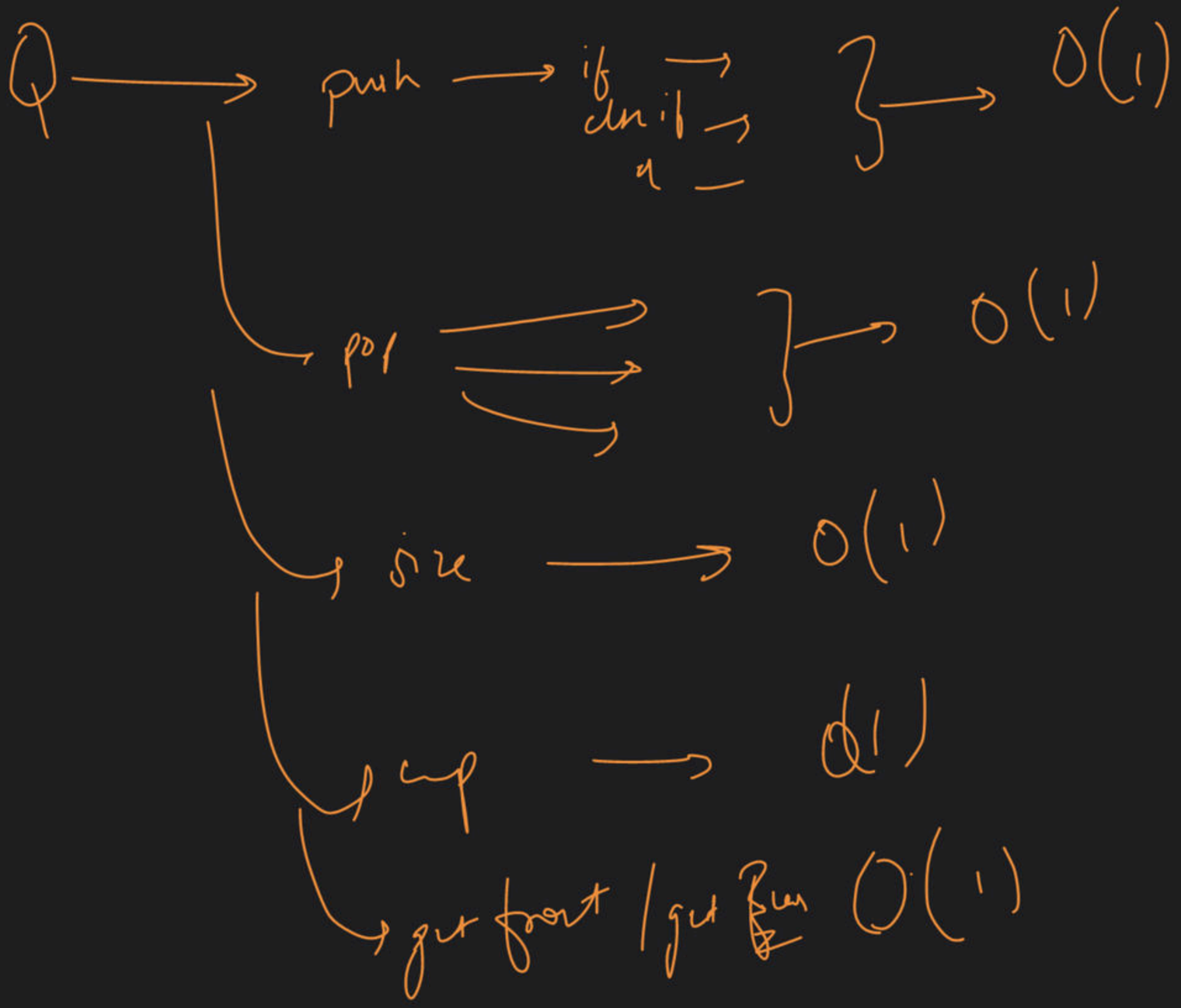
q.push(10)

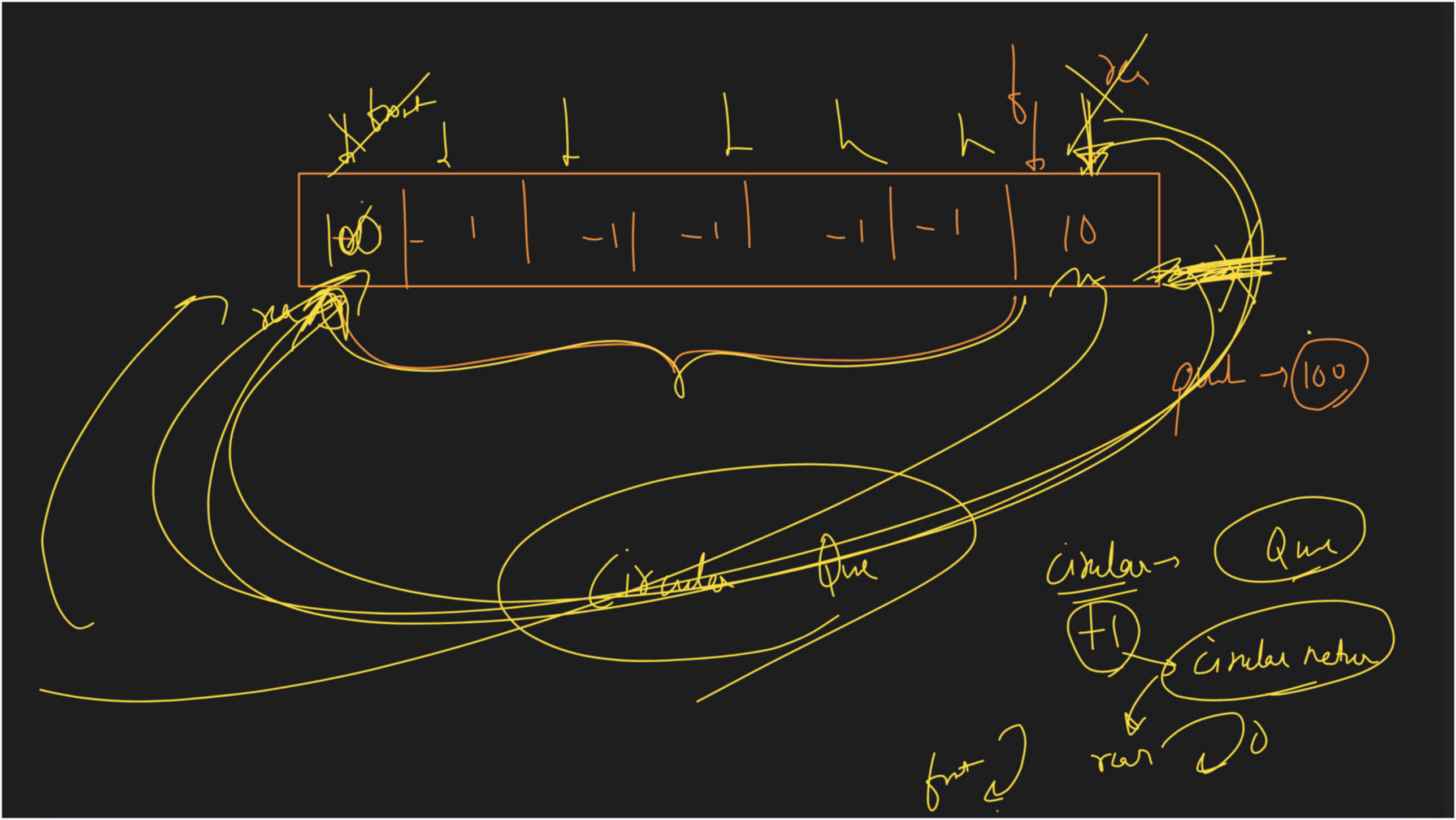
q.pop()

f == r == 0 → single element

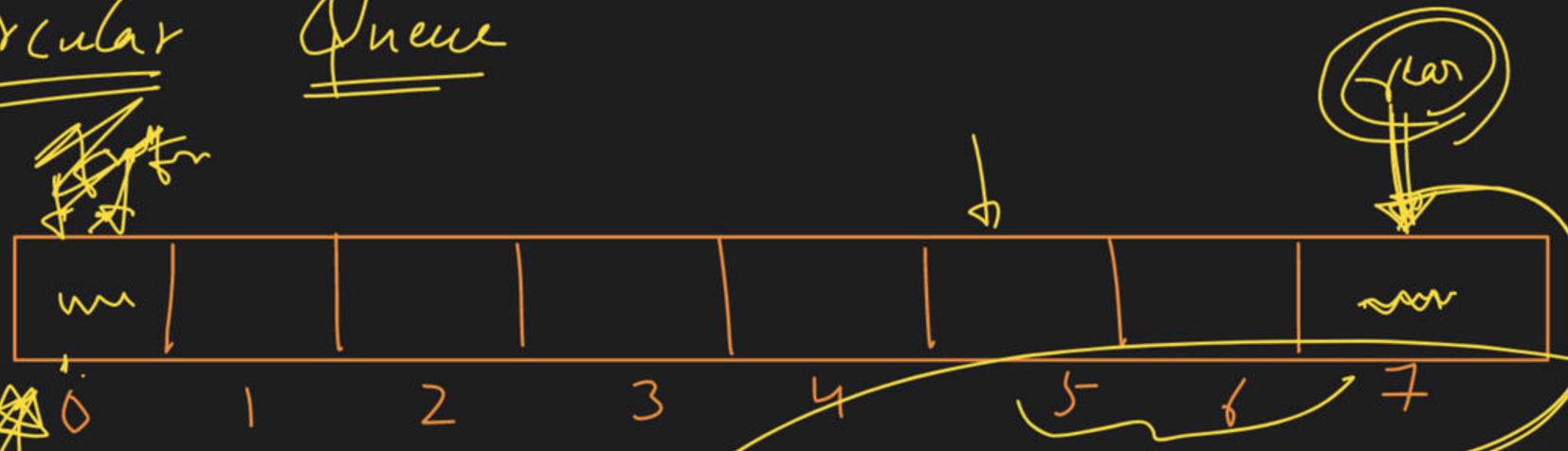
f == r → removed → q.empty()

f → -1
r → -1


Q
push
if limit
O(1)
pop
O(1)
size
O(1)
empty
O(1)
get front / get Rear O(1)


100
10
Circular Que
Circular
+1
Circular return
front
rear

Circular Queue



full → front = 0 && rear = size - 1

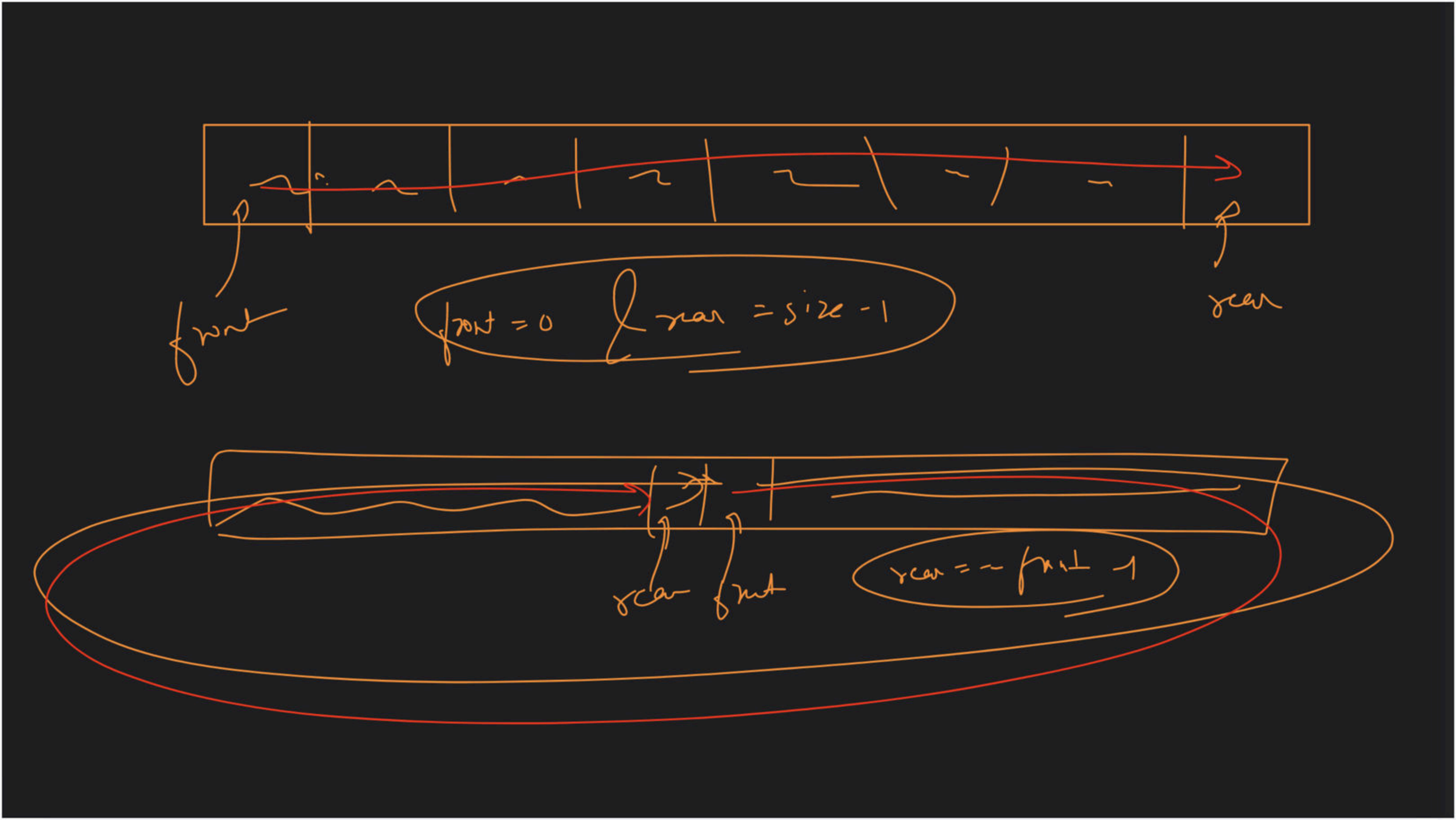
empty → front = rear = -1

Isfull → front == rear + 1

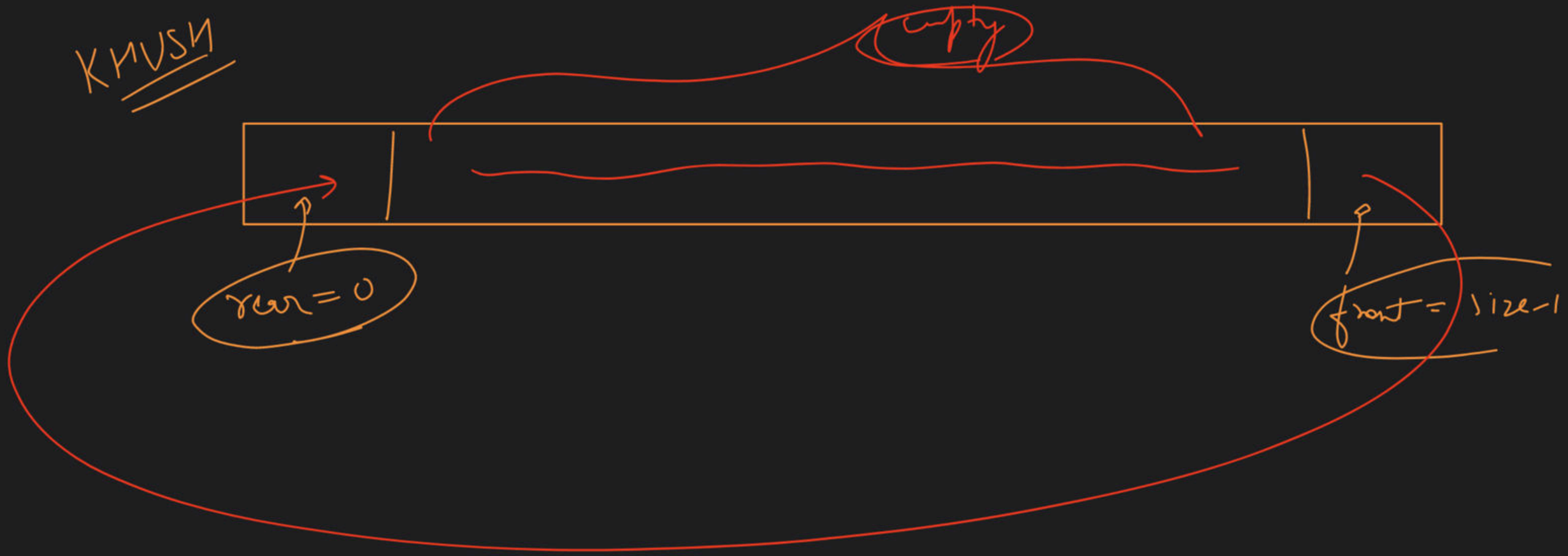
push

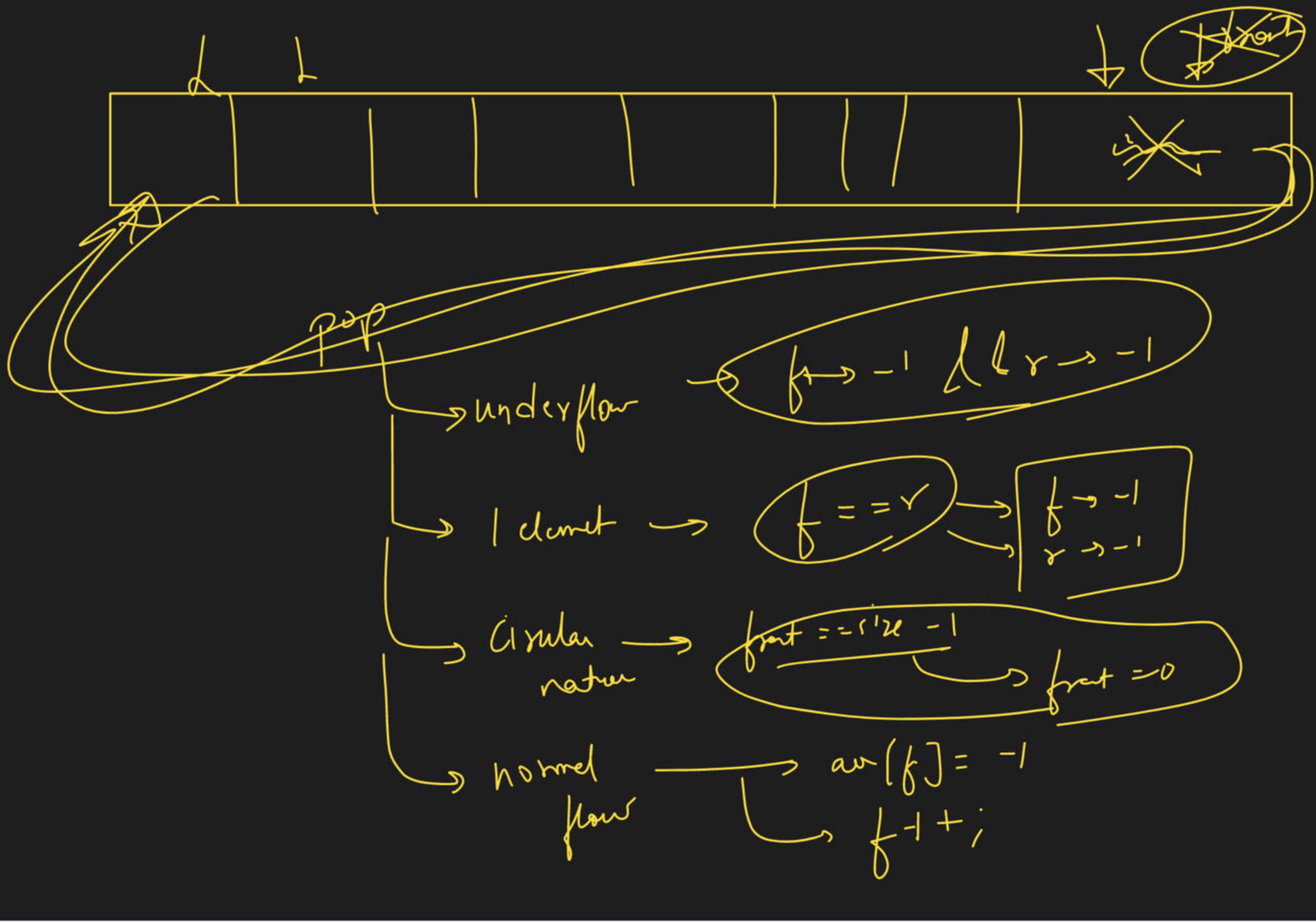
- Overflow → front == 0 && rear == size - 1
- Empty case → f == -1 && rear == -1 → front = 0 → arr[rear] = val
- circular nature → rear == size - 1 && front != 0 → rear = 0 → arr[rear] = val
- normal flow → rear++ → arr[rear] = val

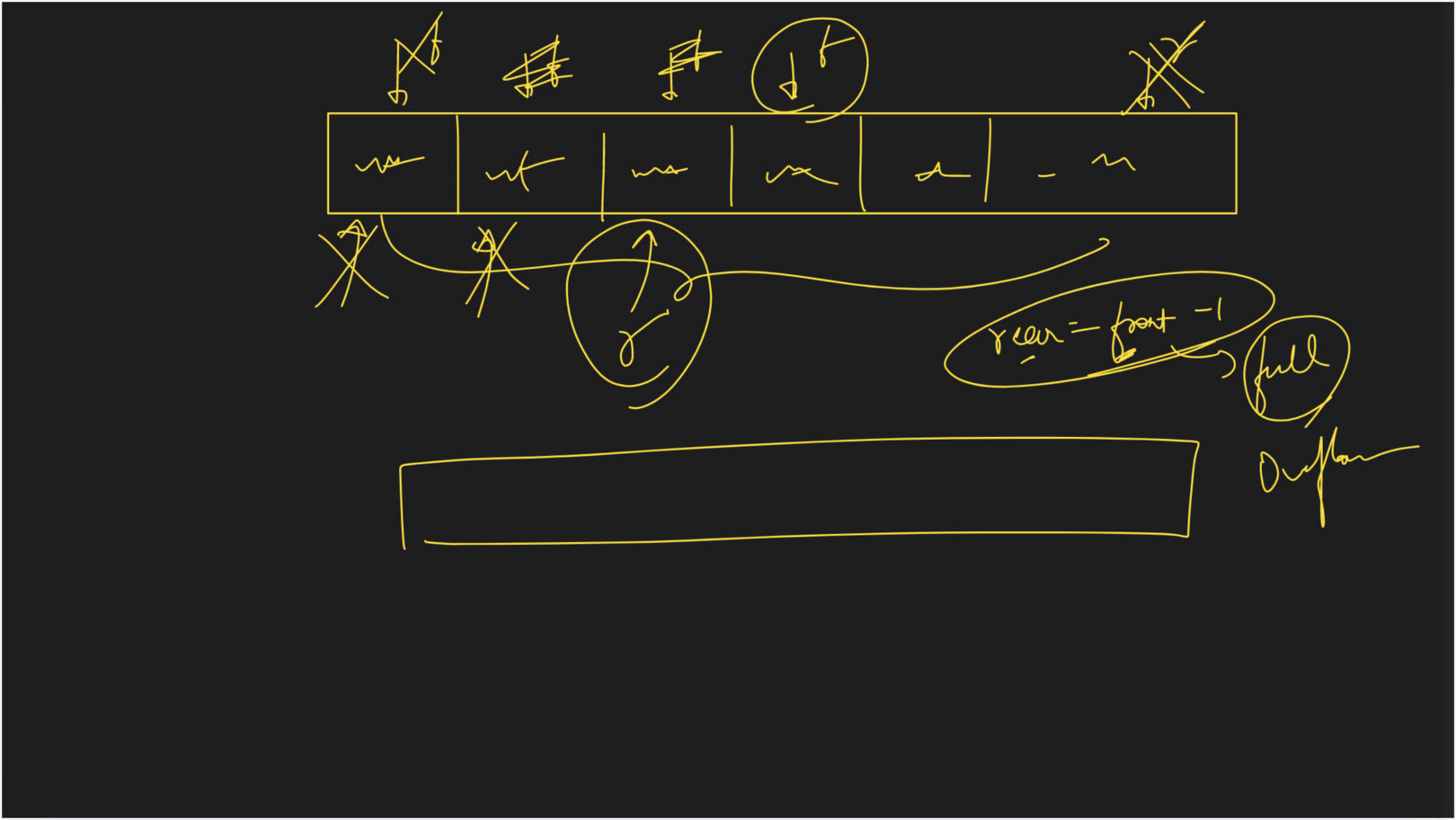
arr
size
front → -1
rear → -1

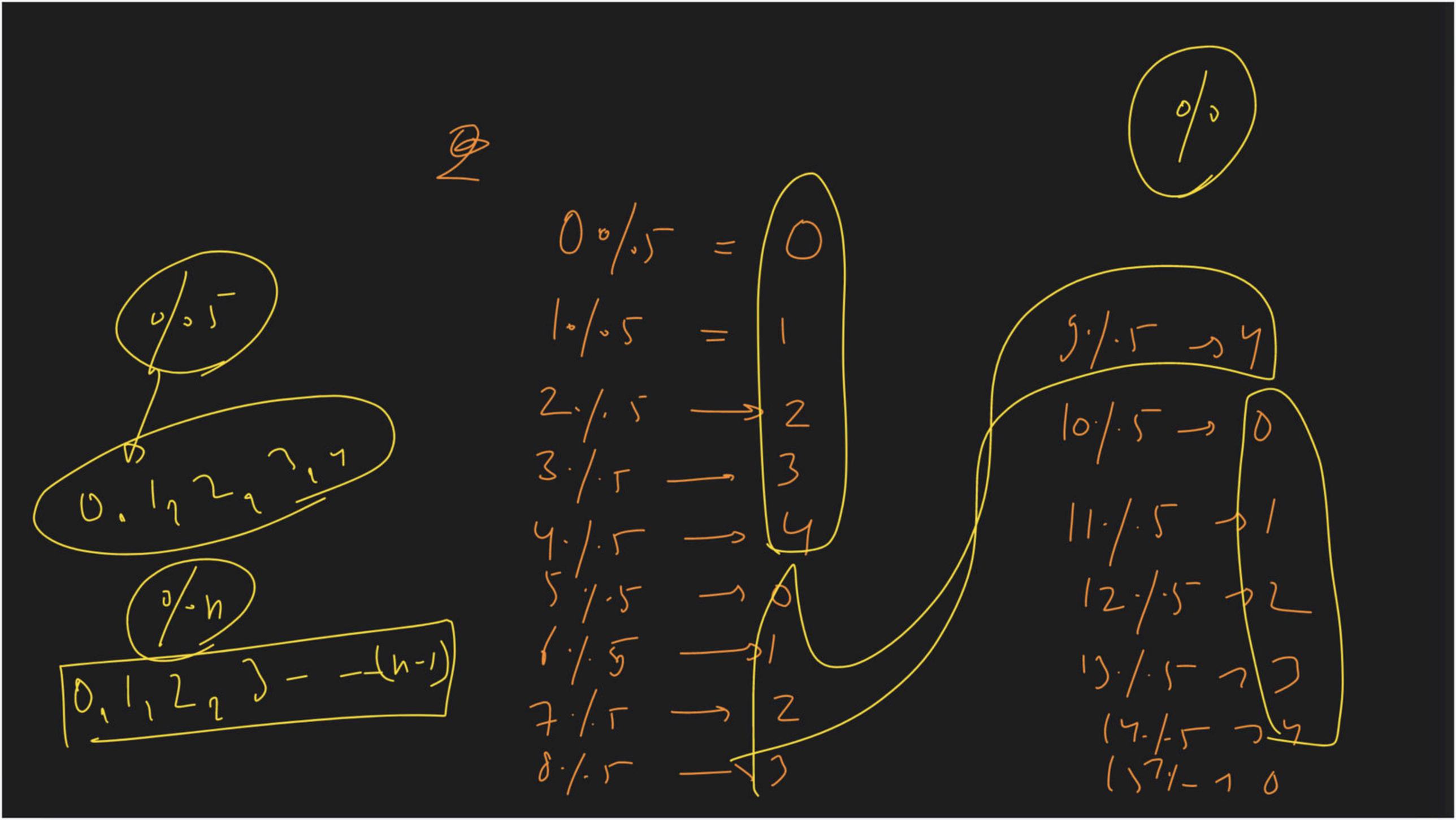

front
front = 0
rear = size - 1
rear
rear
front
rear = front - 1

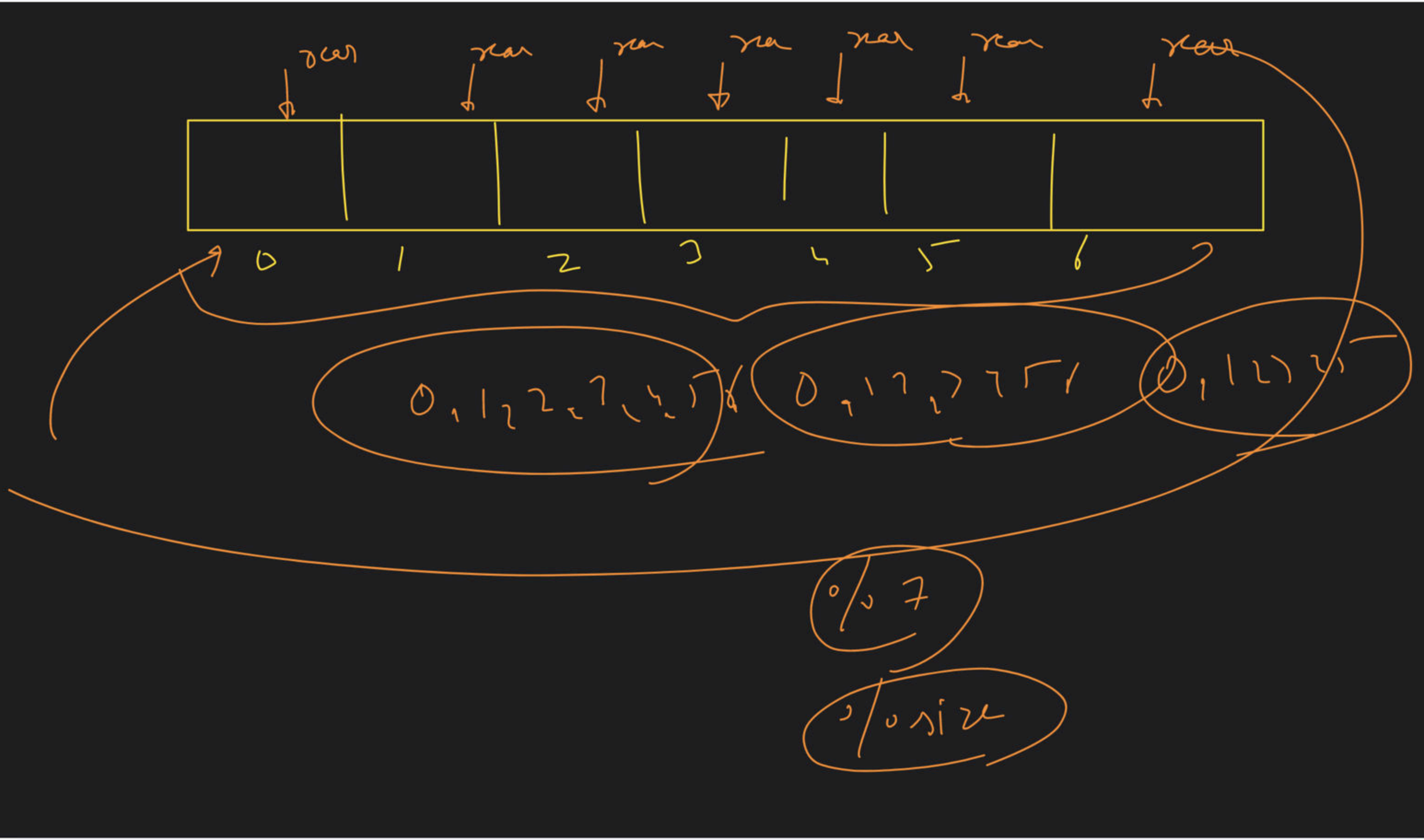
KHUSH




pop
underflow
f → -1 && r → -1
1 element
f == r
f → -1
r → -1
Circular nature
front == size - 1
front = 0
normal flow
arr[f] = -1
f++;


rear = front - 1
full


2
0 % 5
0, 1, 2, 3, 4
0 % n
0, 1, 2, 3 ---(n-1)
0 % 5 = 0
1 % 5 = 1
2 % 5 → 2
3 % 5 → 3
4 % 5 → 4
5 % 5 → 0
6 % 5 → 1
7 % 5 → 2
8 % 5 → 3
% 0
9 % 5 → 4
10 % 5 → 0
11 % 5 → 1
12 % 5 → 2
13 % 5 → 3
14 % 5 → 4
15 % 5 → 0



Doubly Ended Queue

deque

push_front
pop_front

deque

push_back()
pop_back()

Operating System

Process Scheduling

Types of Queue

600 120 110 100 10 20 200 500 1000

F — R/B

push front (101)

push (110)

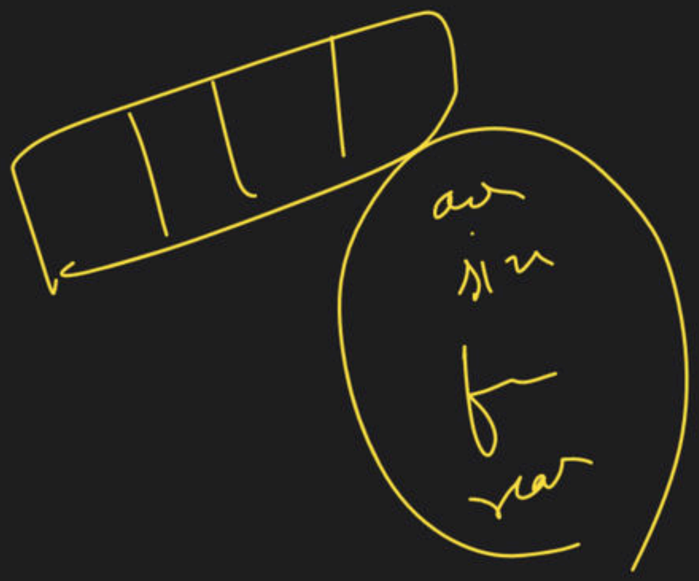
push front (10)

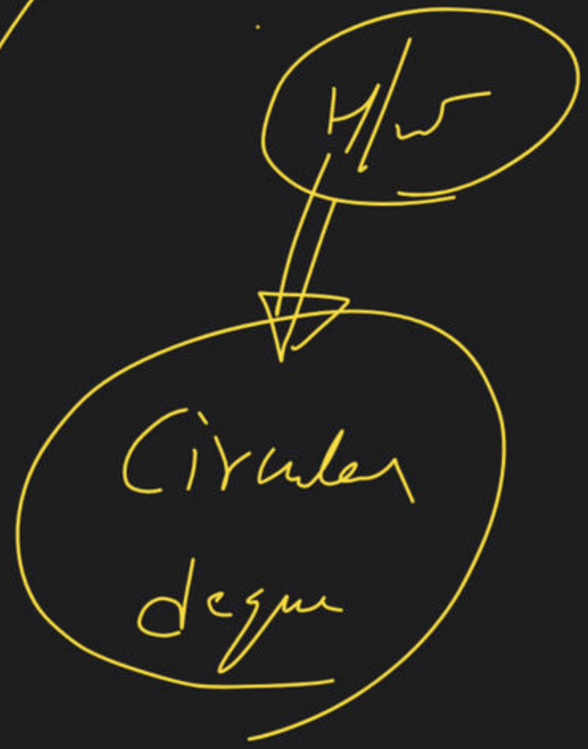
pushback (20)

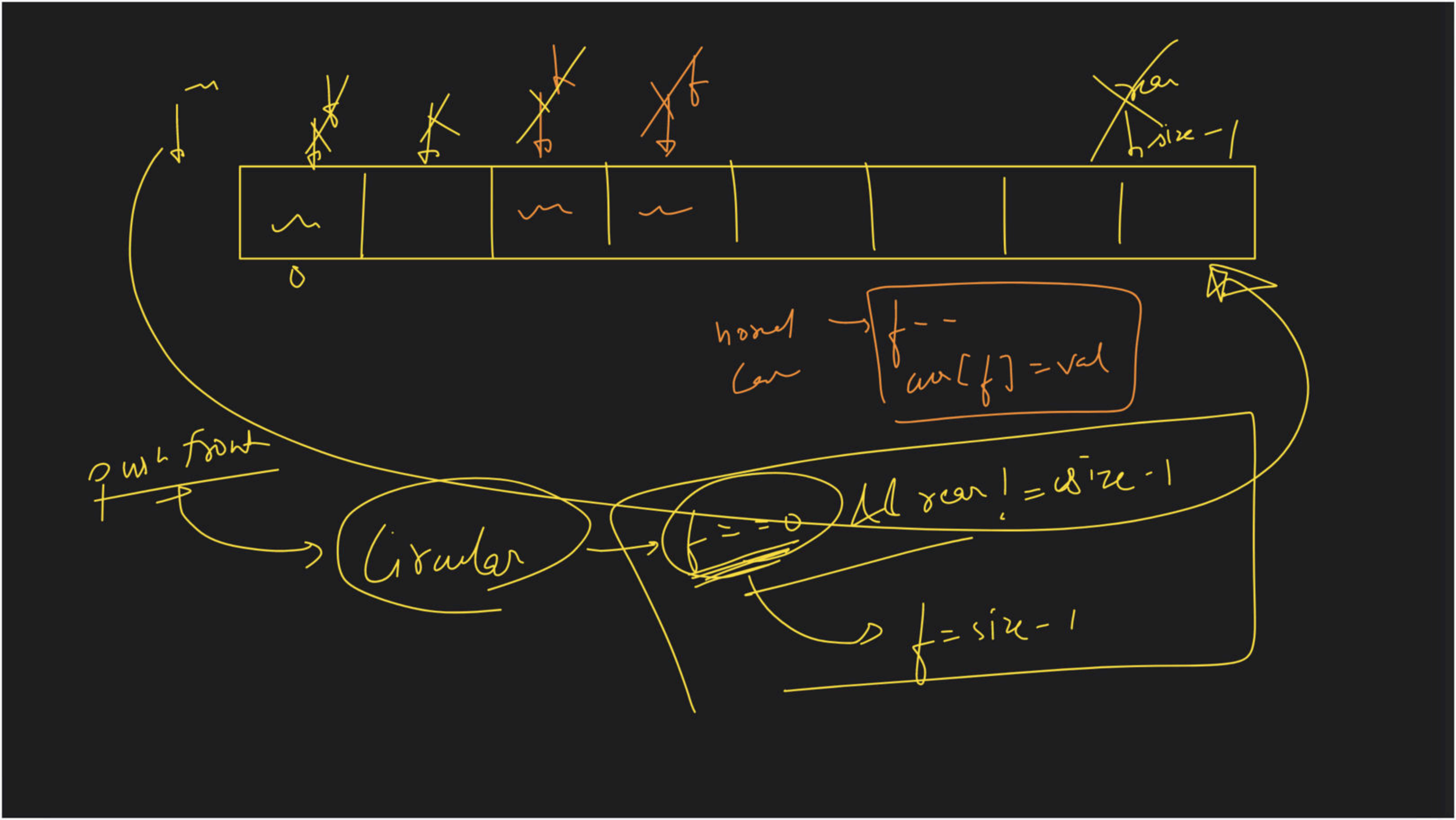
Implement Deque from Scratch

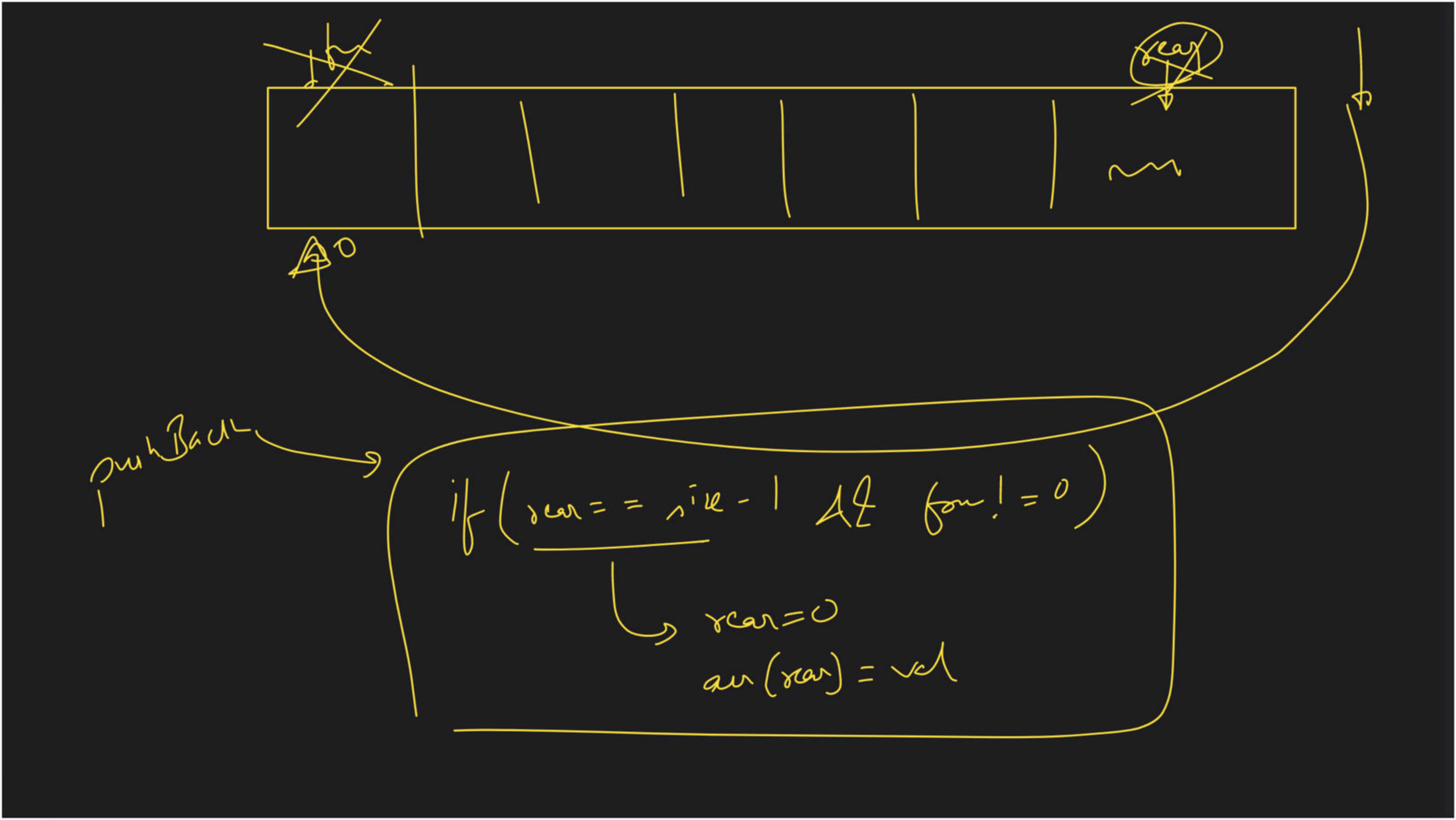
Normally

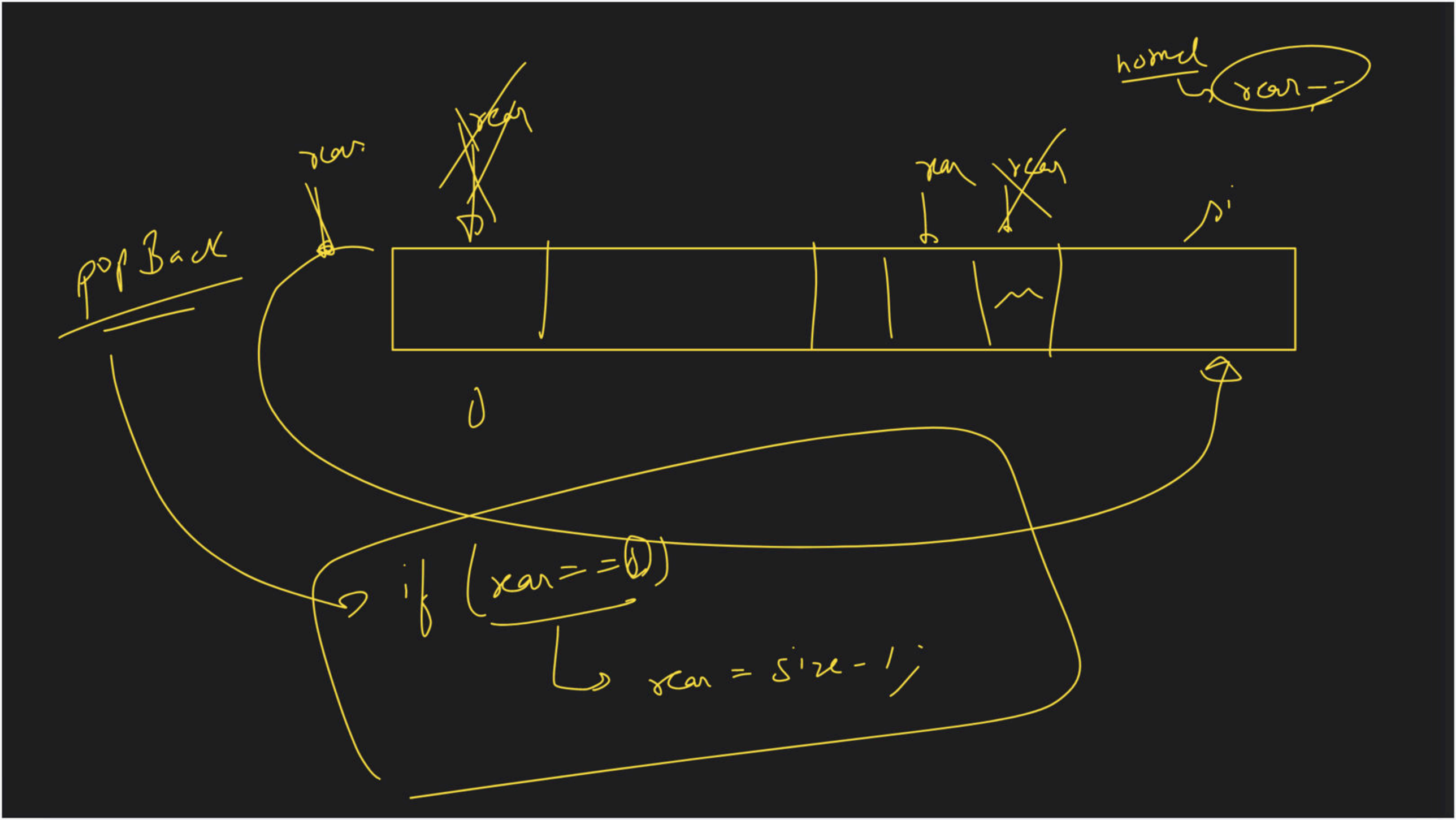
- push_front
- push_back
- pop_front
- pop_back

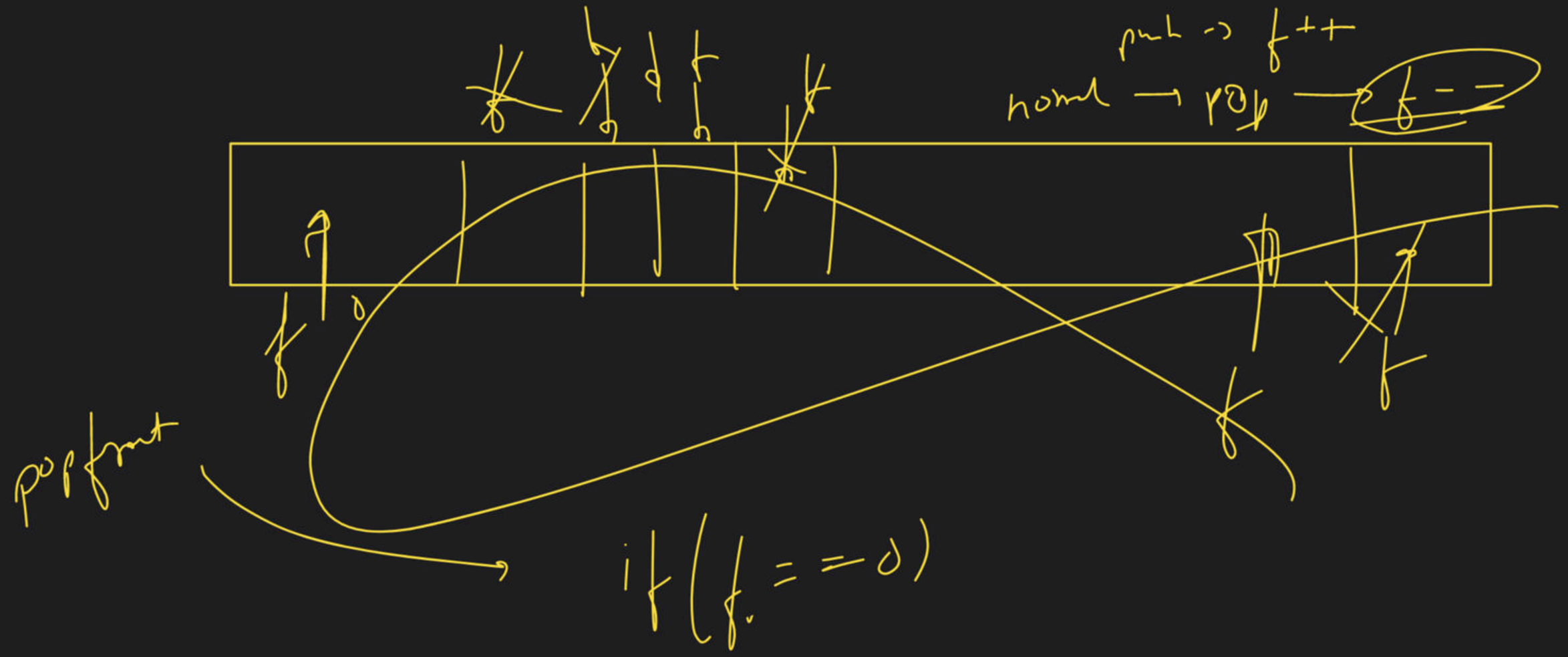


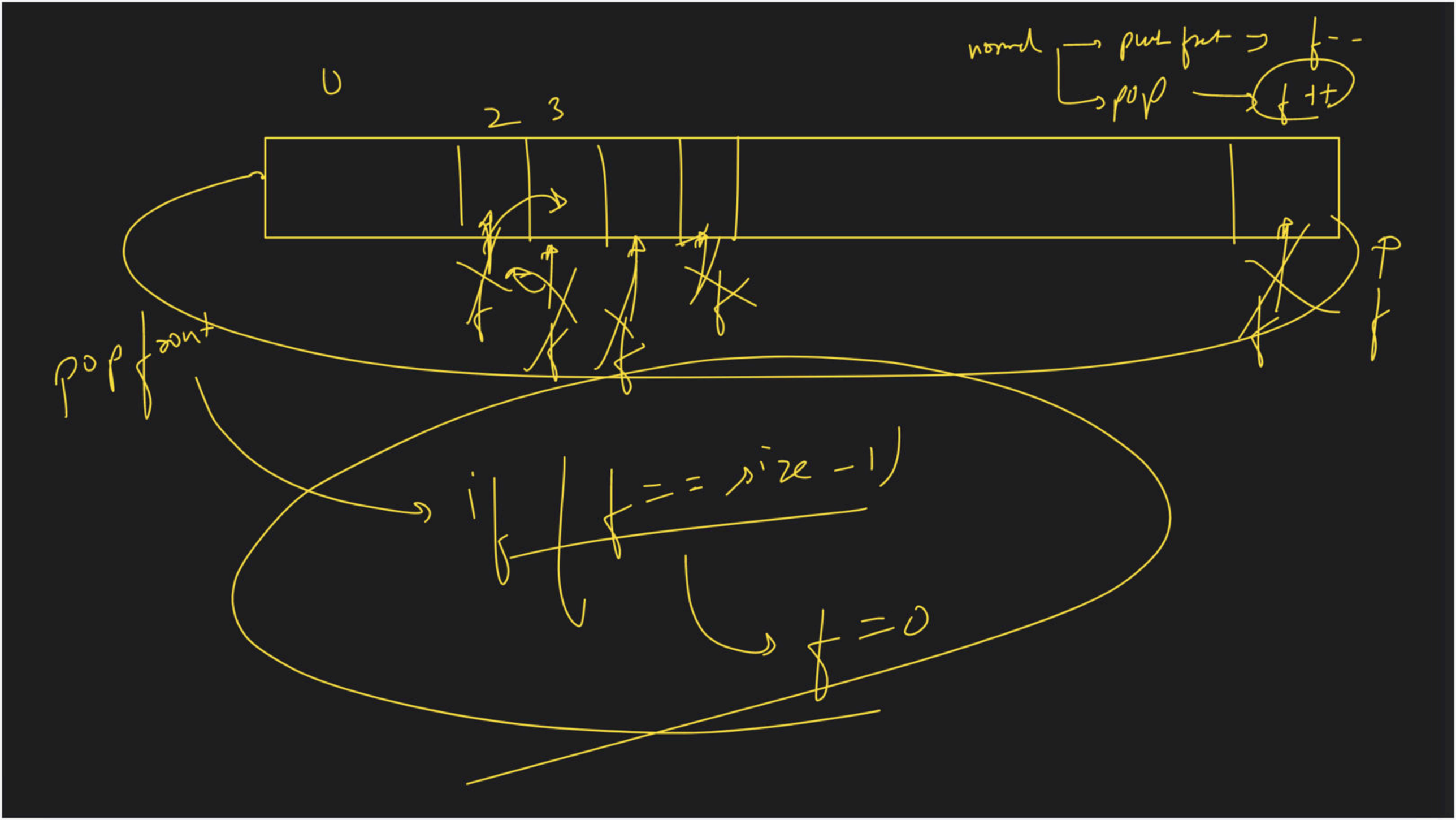



rear
size - 1
0
normal case
f--
arr[f] = val
push front
circular
f == 0
All rear != size - 1
f = size - 1


rear
0
pushBack
if(rear == size-1 && front != 0)
rear=0
arr(rear) = val


homed
rear--
rear
rear
rear
rear
size
popBack
0
if (rear==0)
rear = size-1;


push -> f++
normal -> pop
f--
popfront
if(f.==0)


normal
push front
f--
pop
f++
0
1
2
3
f
popfront
if(f==size-1)
f=0