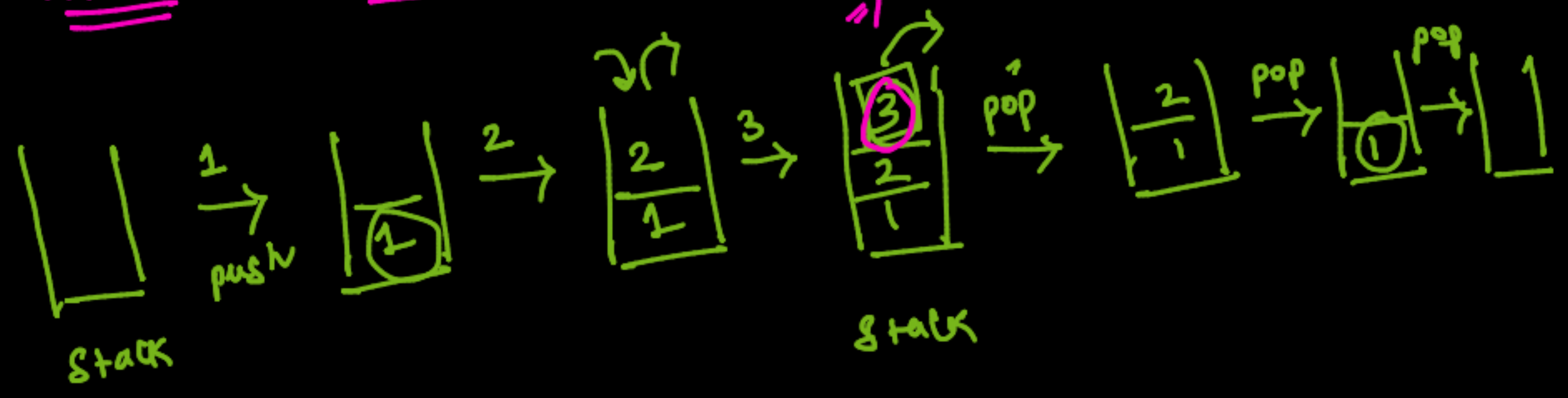
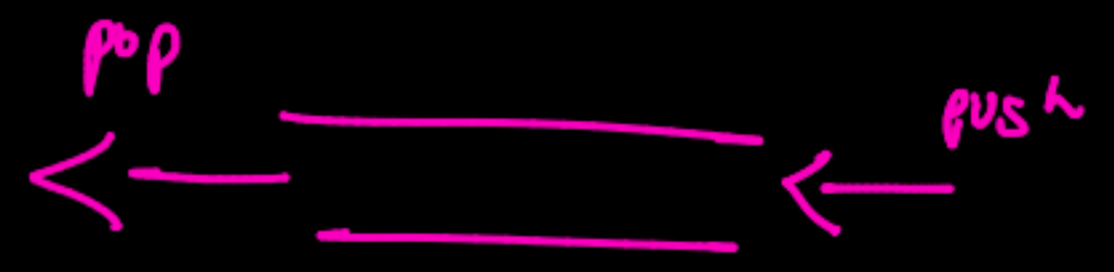
Stacks

↳ Ds where the elements are inserted and extracted from the same end.

↳ LIFO or FILO

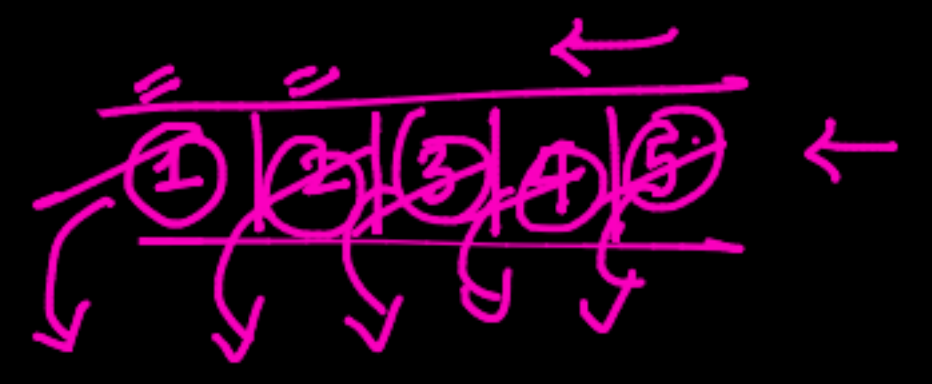


Queue



FIFO or LILO

Q



operations a stack supports

push ( )

pop ( )

top ( )

size ( ) → int

empty ( ) → true/false

C++ ⇒ Container Adapter

vector ✓

array ✓

(forward-list) ✓

list ✓

sequence containers

↳ (using vectors)

↳ using linked list

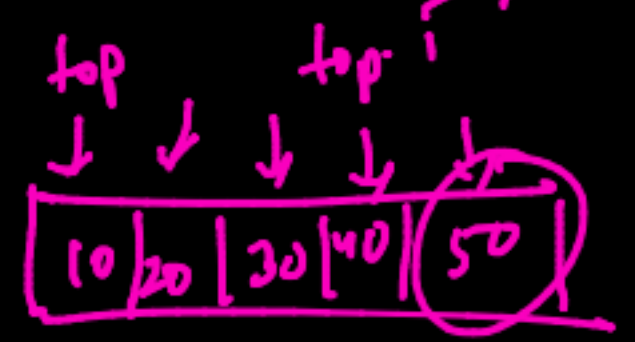
Stacks → vectors

→ linked list

→ array → fixed

Stack Overflow

arr



→ fixed size

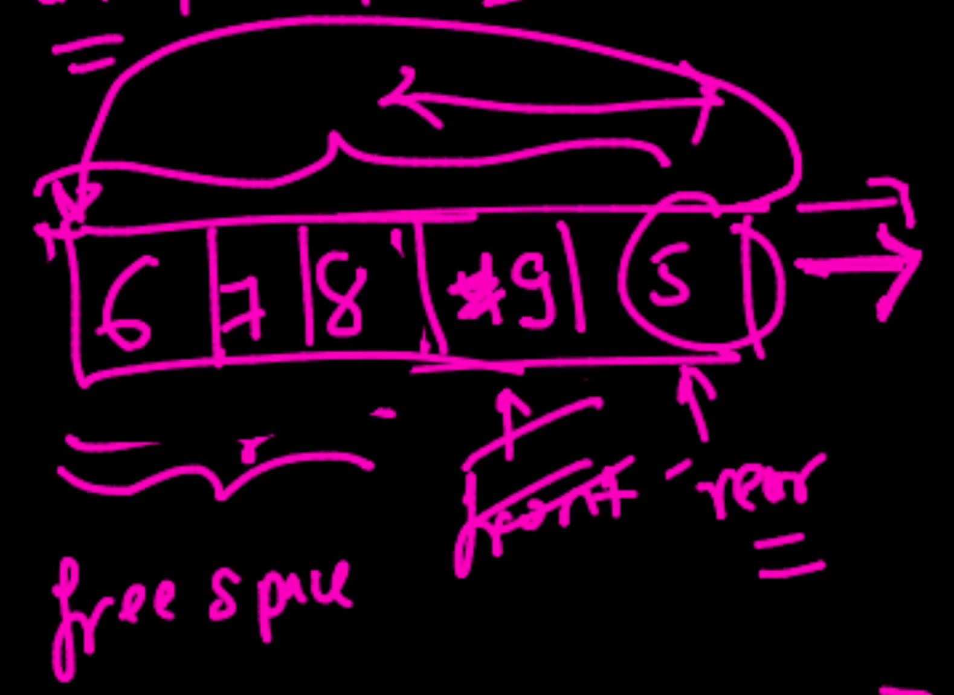
top = -1

arr [top++] = data

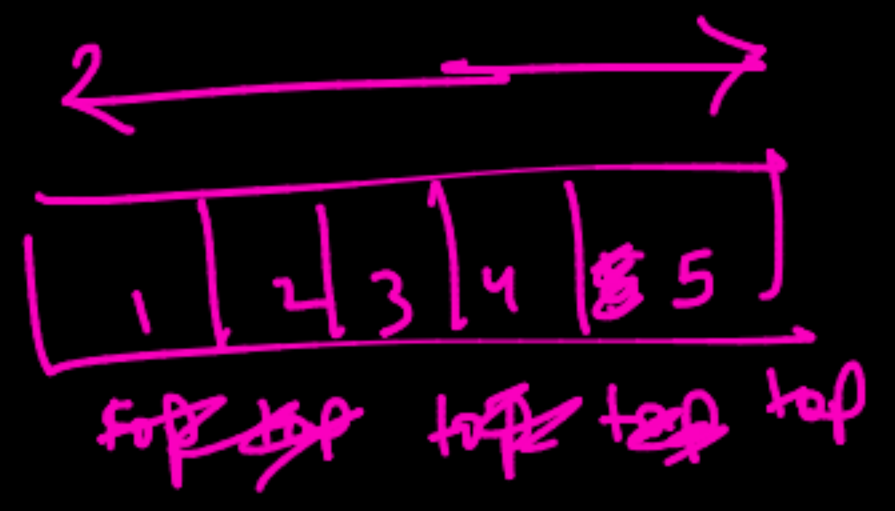
top()

pop

= Queues → linked list (dynamic)
→ arrays (fixed size)



a[rear++]=data



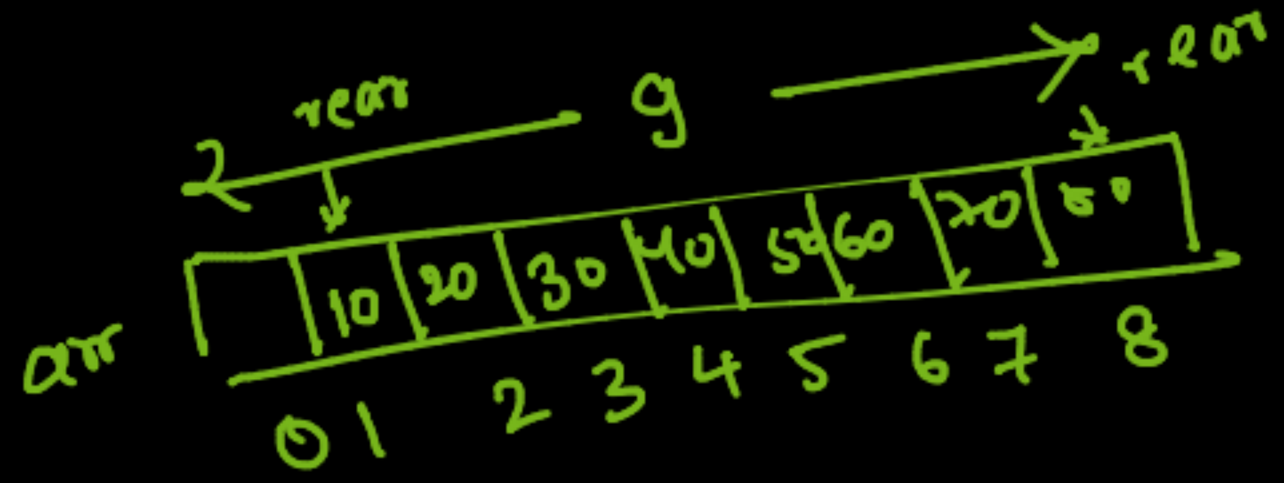
top=-1

top++

CQ ⇒ 8

len=0

capacity=9



90

(8+1)%9

= 0

⇒ insert at rear
⇒ delete from front

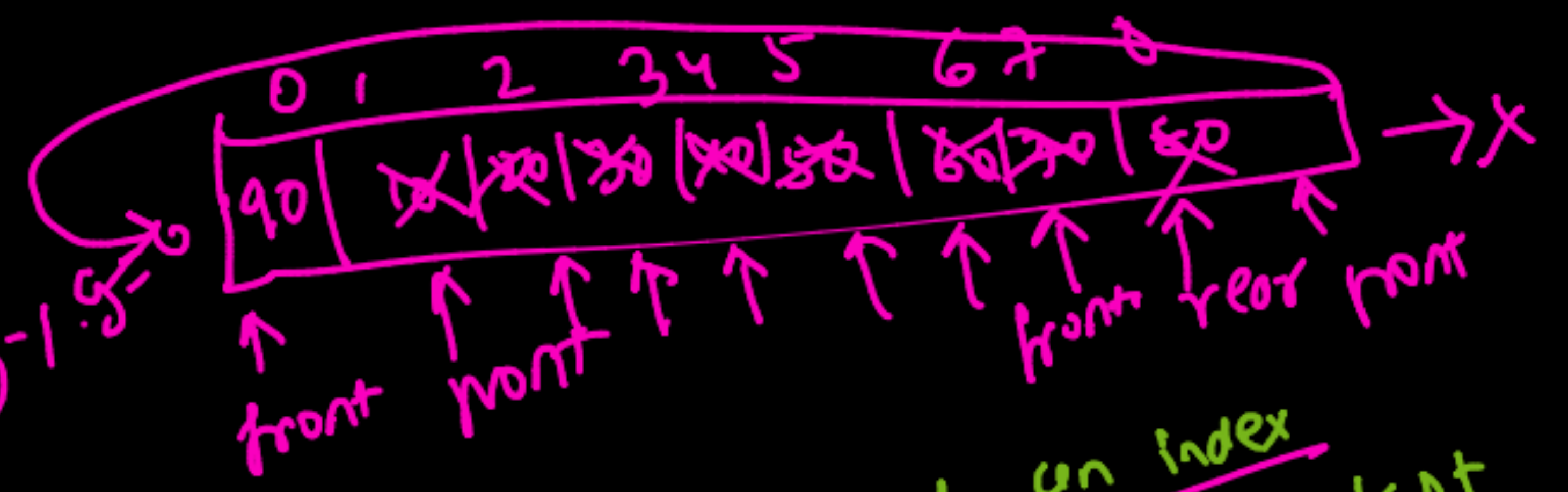
front=0
rear=0

= front = 0
= rear = 1%9=1

push → insertion into the cq.queue

```
✓ x if (front == (rear +1)% c) {
    // que is full
} else {
    rear = (rear +1)%c
    arr[rear]= data;
}
```

front=8
rear=8
(8+1)%9=0
90



pop

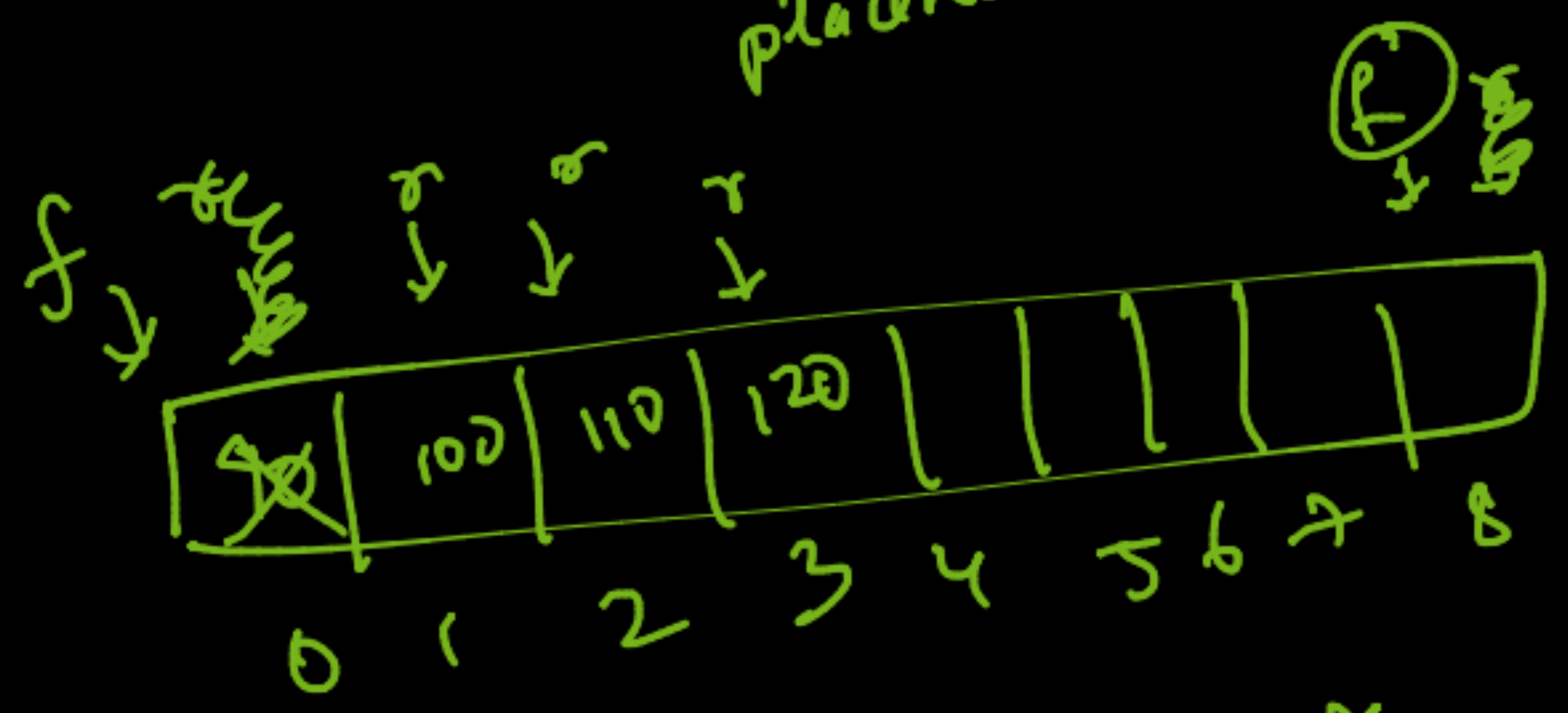
pop

⇒ front → at an index which was last popped

```
if (front == rear) {   queue = empty
} else {
    front = (front +1)%c
}
```

c+1

r=(r+1)%c
(8+1)%9



f == (r+1)%c x

r = (r+1)%c