unacademy

Trees - Part I

Course on Data Structure

Pankaj Sharma • Lesson 21 • July 31, 2024

unacademy

CS & IT Engineering

Data Structure

Stack & Queue

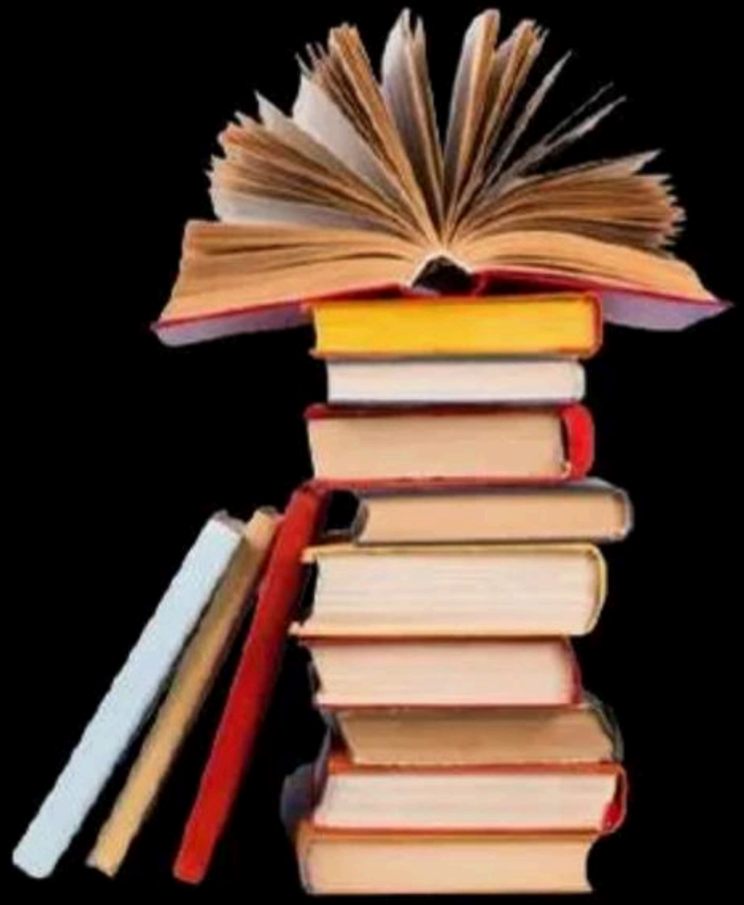
Lecture Number- 21

By- Pankaj Sir

unacademy

Topics

to be covered

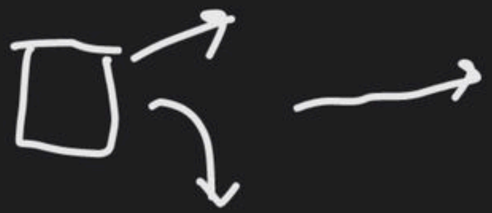


1. Queue

1. Test Series → 2 subjects → Oct, Nov → 7W, 5V, M → 1 – 2 – ⋮ – 5–10

30-30 ₹

Jan



Test Series

① TWI → = 

Test Series → Marks X

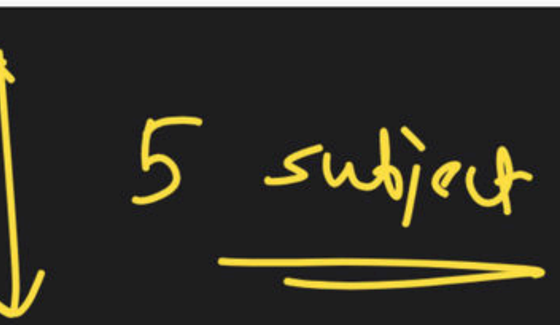
① ✓
② ↓

demarcation
↳ Subject X

Revise → Test → Analysis → Ranks

Notes

2 subject → ① 1 live ② recorded



5 subject

1.5x

2x

revising 2 subjects.

PYQs → subject

unacademy

Revision

21 days

7 days

1. Notes → short-notes → Theory + Questions

2. 15 Nov → End of Oct

unacademy

College → — 2 hrs

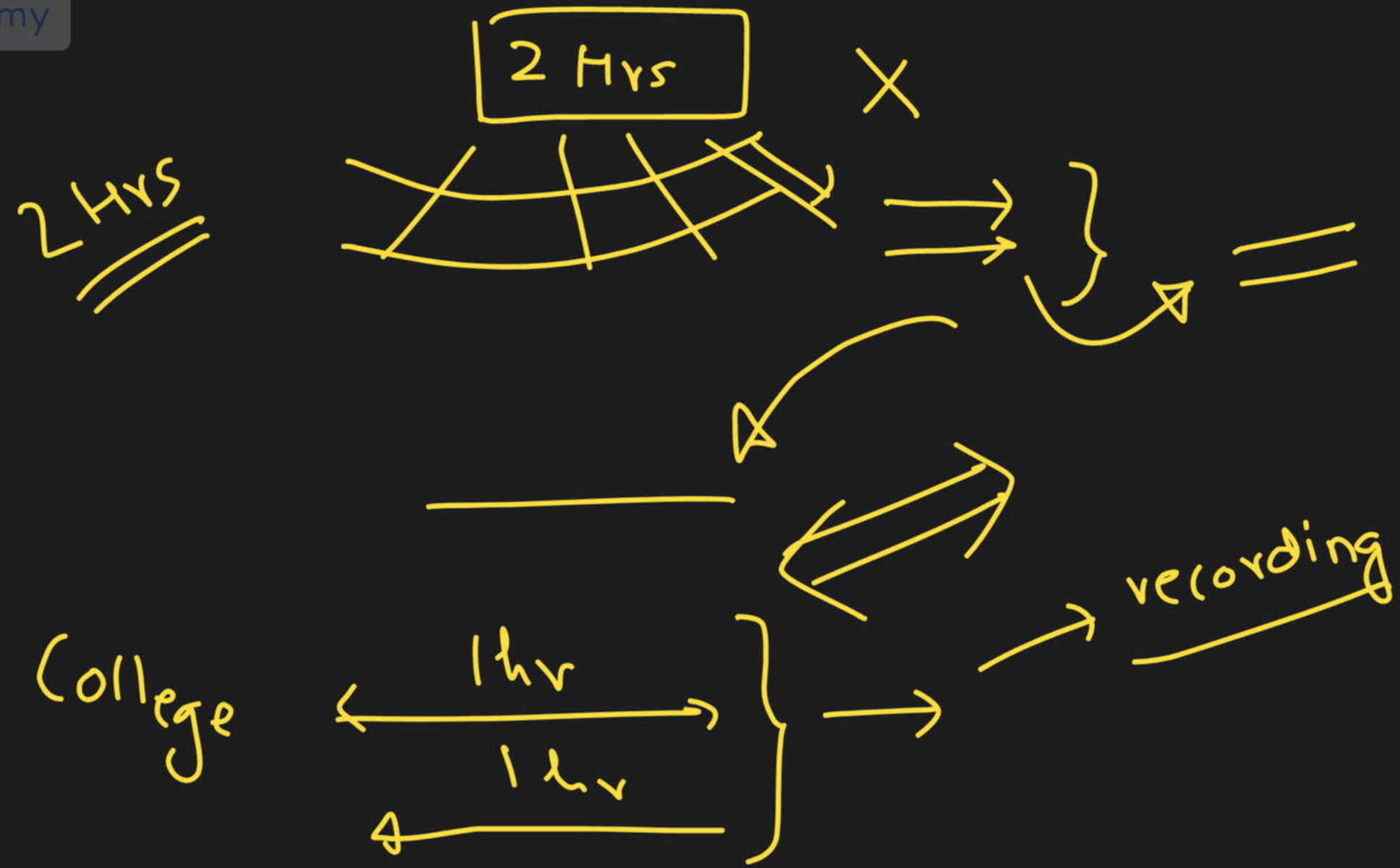
→ 7:60 AM → Mobile

→ 

DBMS

06:30 AM

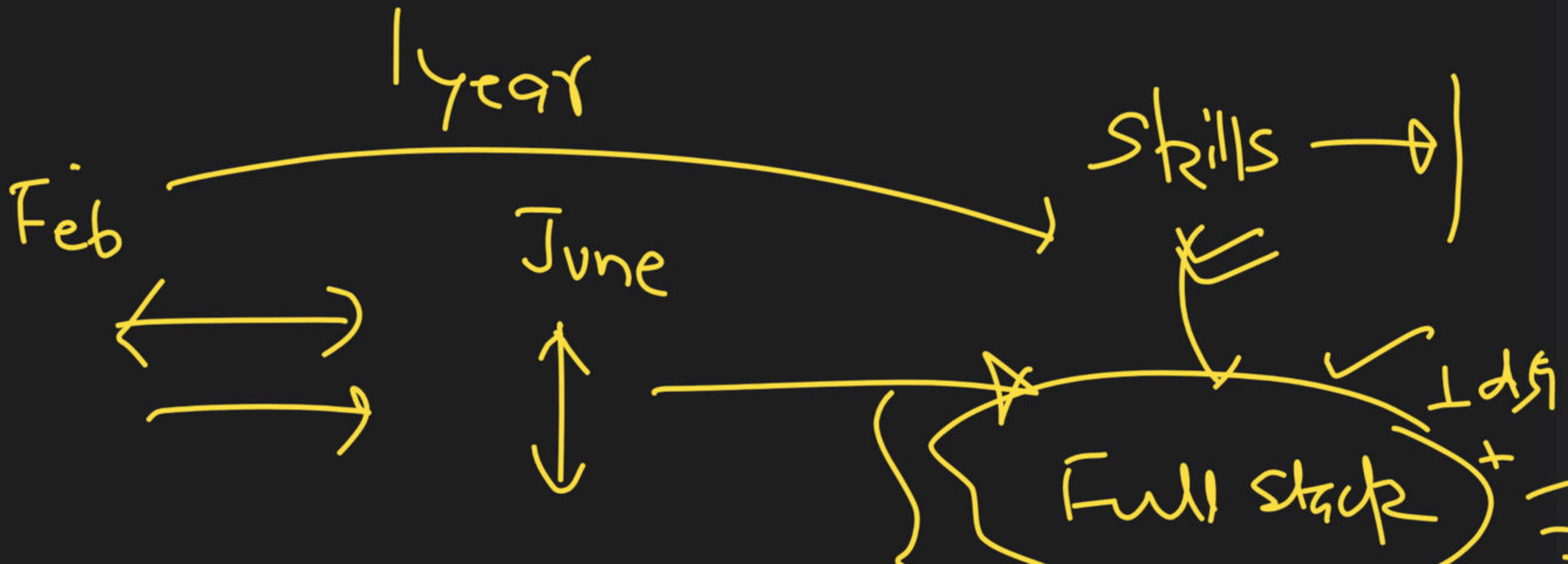
07:00 AM

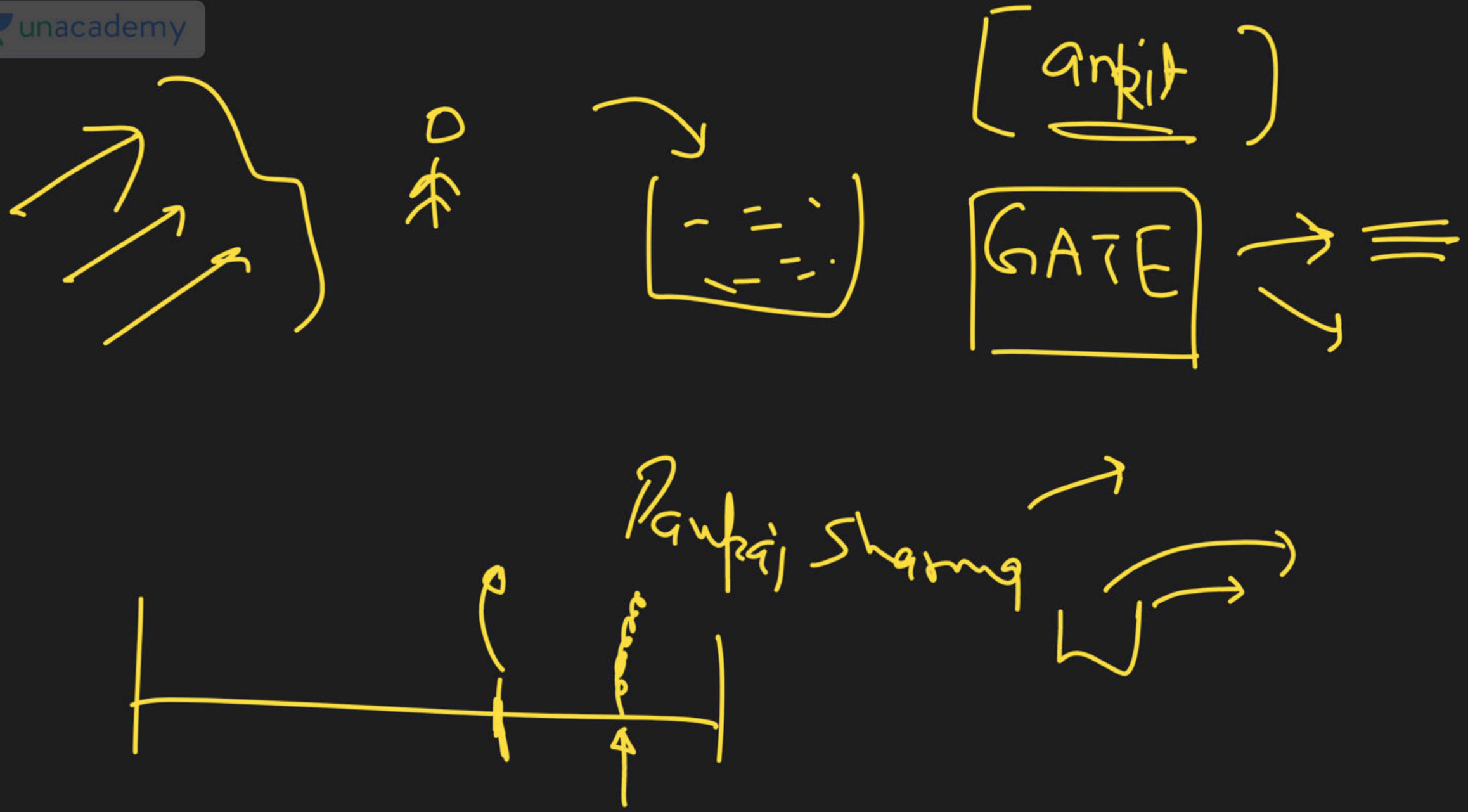


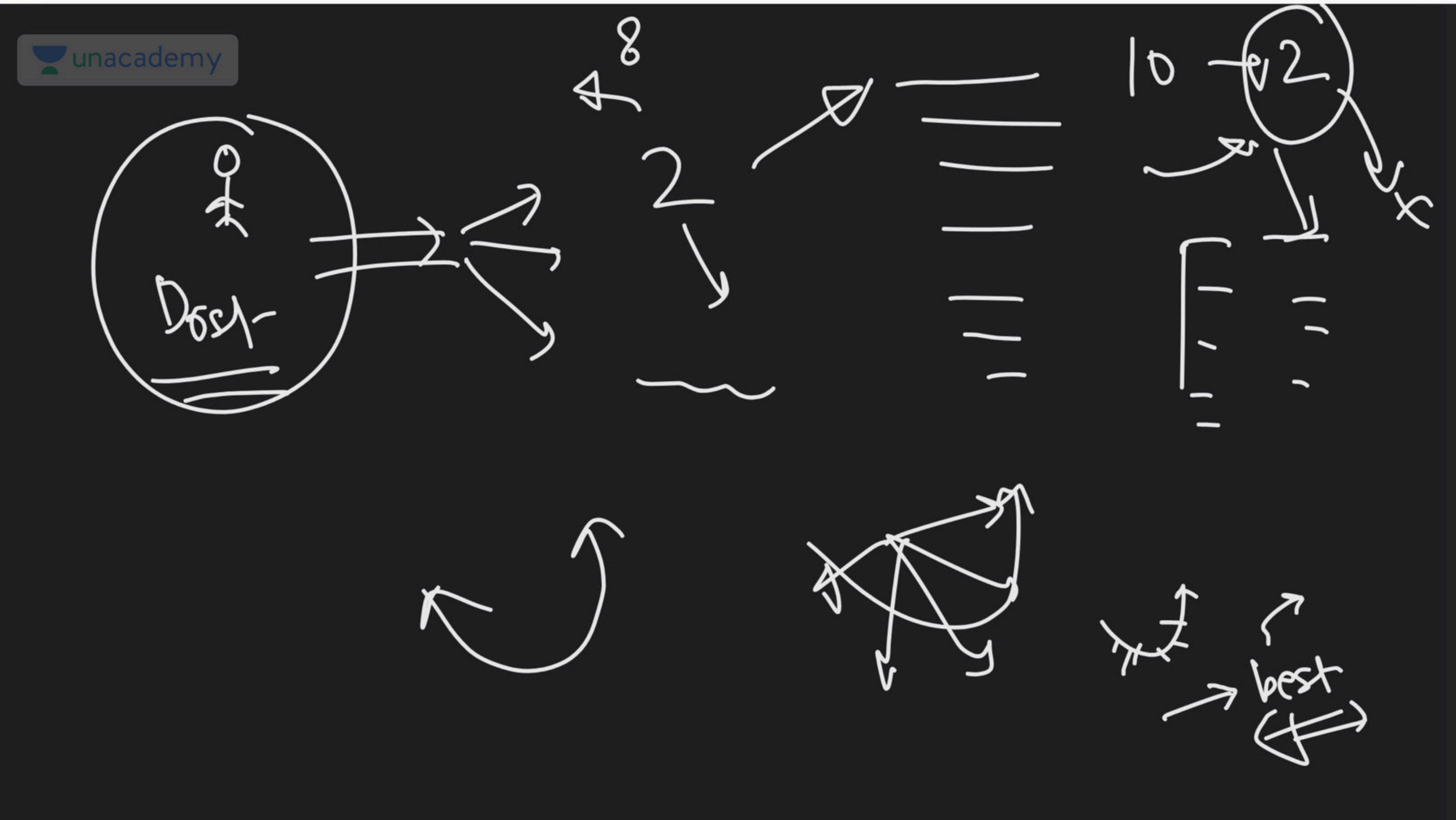
6 Hrs

Weekends

1. Test Series
2. Revision
3. Analysis






unacademy
8
2
10
2
Dost-
best

unacademy

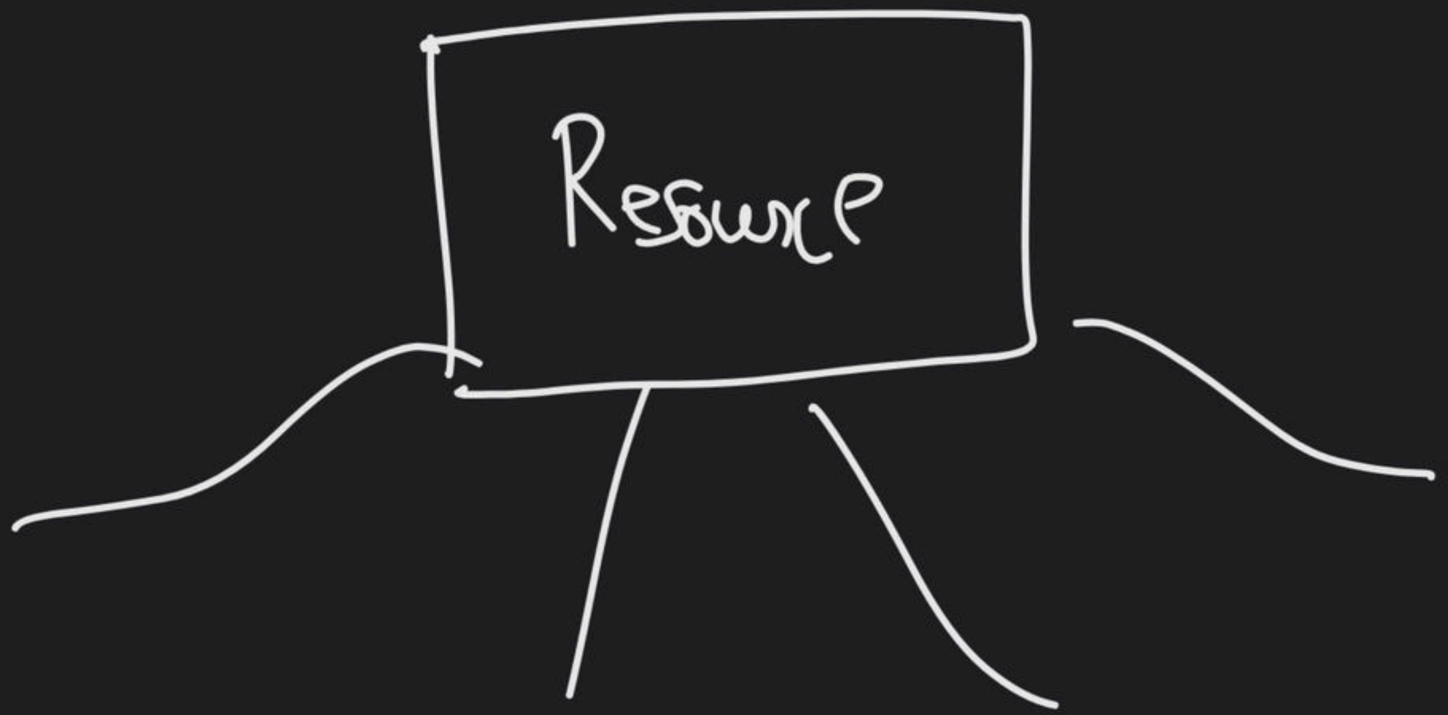
Queue

* Linear data structure

  – First In First Out

(i) Insertion: Rear

(ii) Deletion: Front



Array implementation

```
#define SIZE 5
int Queue[SIZE];
int Rear;
int Front;
```

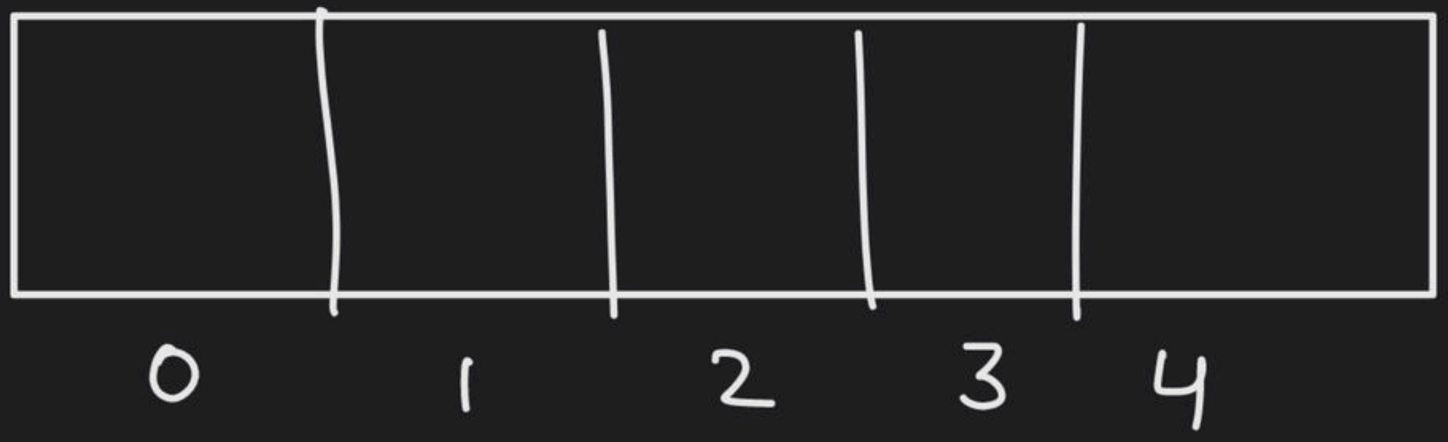
Front : index of element that can be deleted from Queue

Rear : index of most recently added element

unacademy

Initially, When the Queue is Empty.

Front = -1
Rear = -1

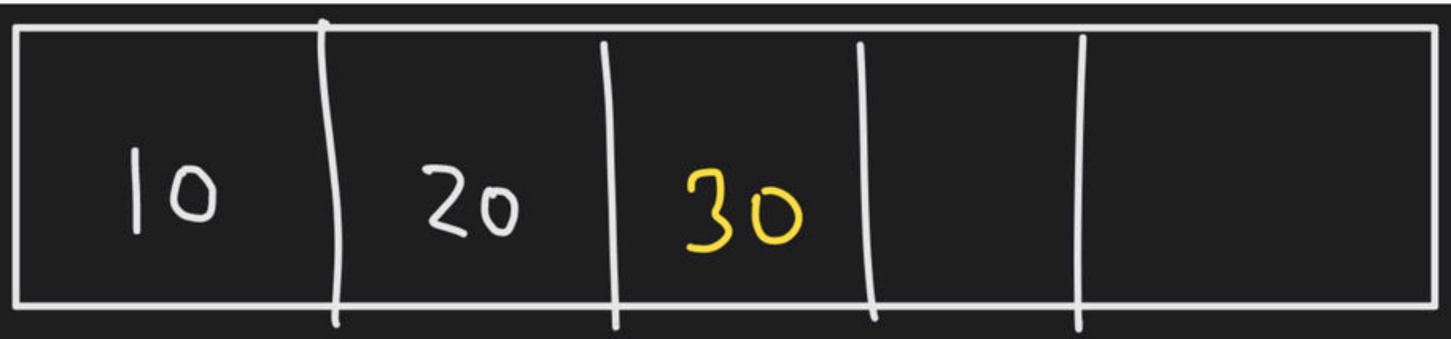


	Front	Rear
Initially	-1	-1
Insert(10)	0	0

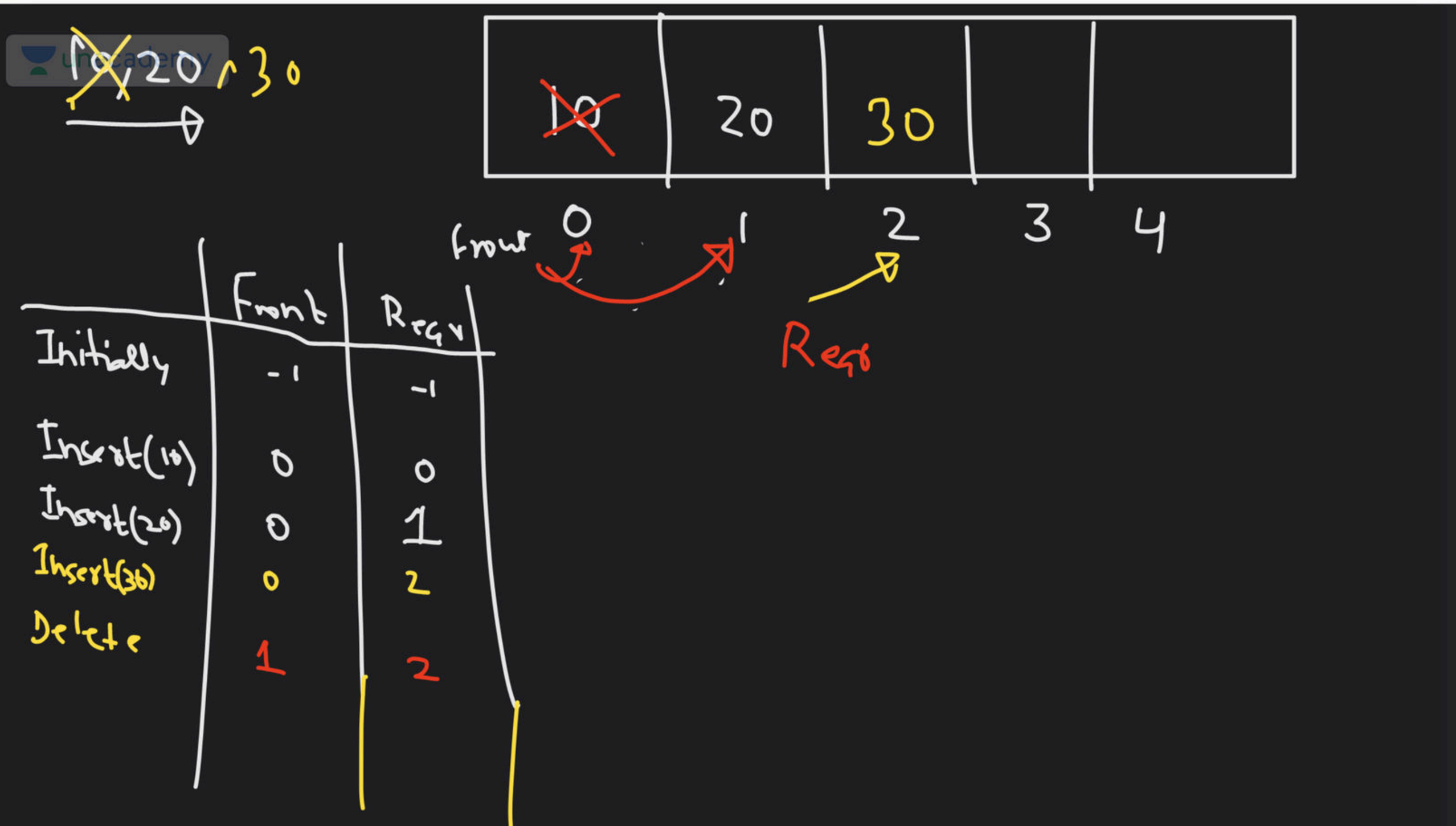
10, 20 →

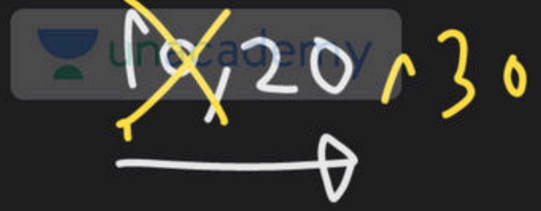
	Front	Rear
Initially	-1	-1
Insert(10)	0	0
Insert(20)	0	1

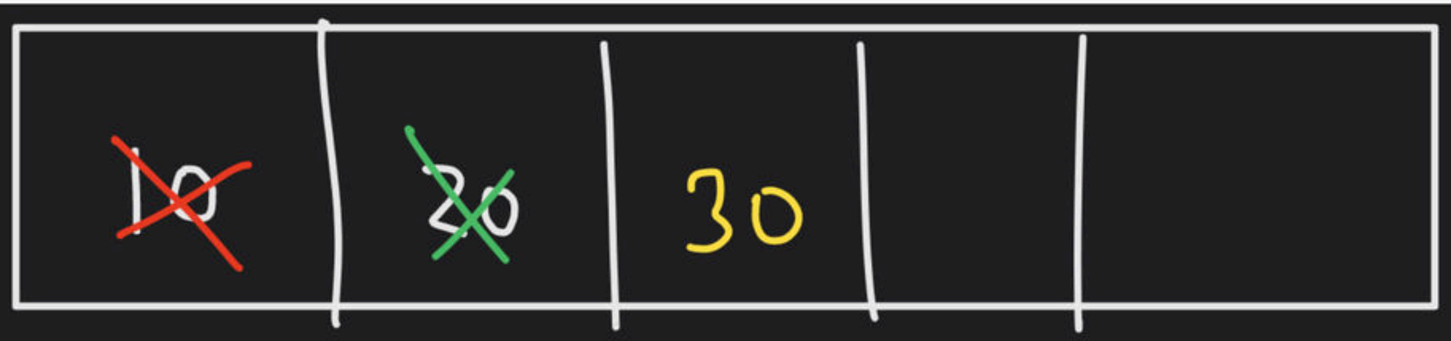
[0, 20)



	Front	Rear
Initially	-1	-1
Insert(10)	0	0
Insert(20)	0	1
Insert(30)	0	2


10
20
30
0
1
2
3
4
Front
Rear
Front
Rear
Initially
-1
-1
Insert(10)
0
0
Insert(20)
0
1
Insert(30)
0
2
Delete
1
2



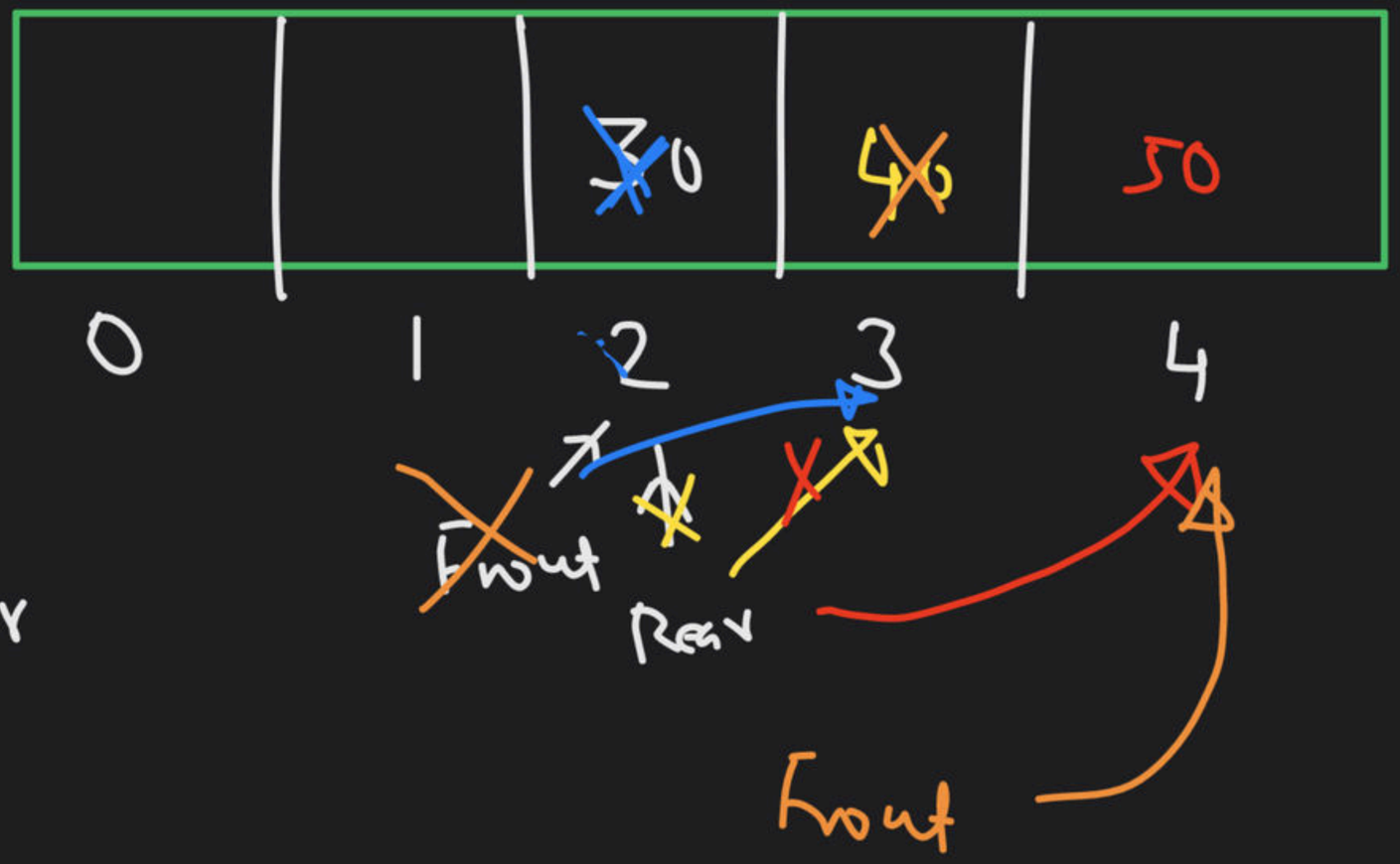


	Front	Rear
Initially	-1	-1
Insert(10)	0	0
Insert(20)	0	1
Insert(30)	0	2
Delete	1	2
Delete	2	2

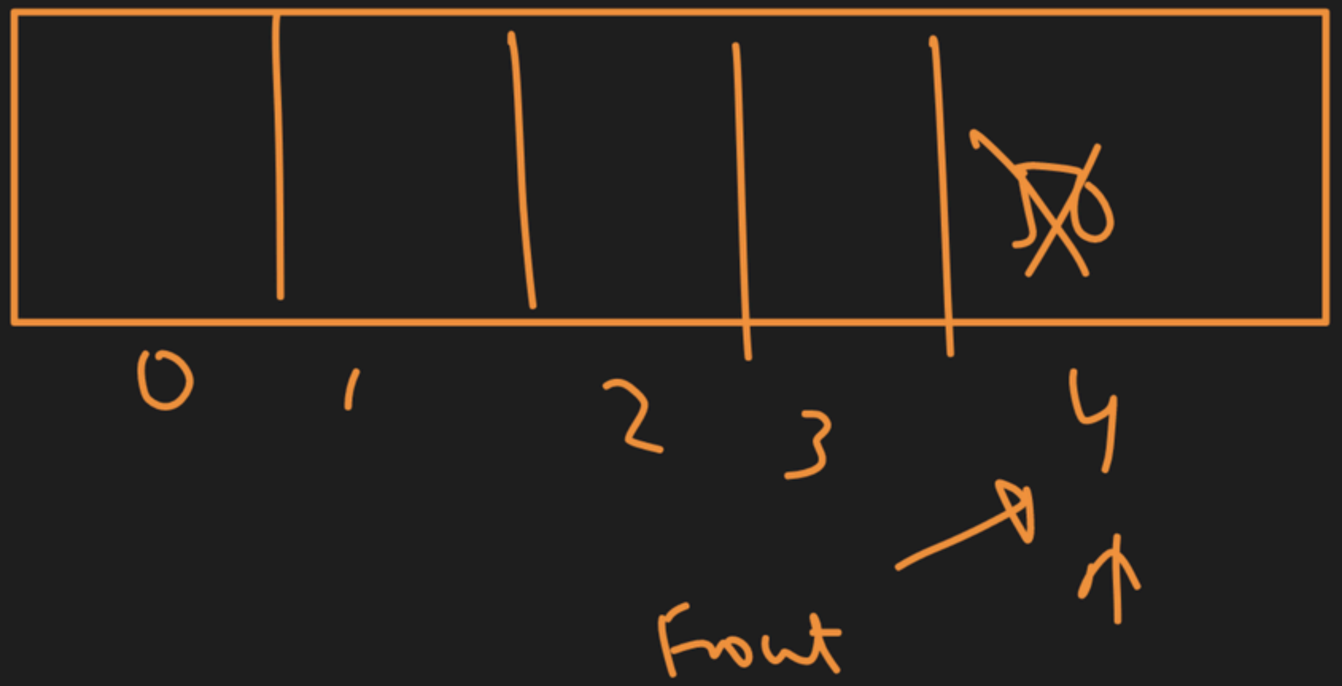
Front & Rear are same

① Empty ⇒ -1

② Only 1 element in Queue



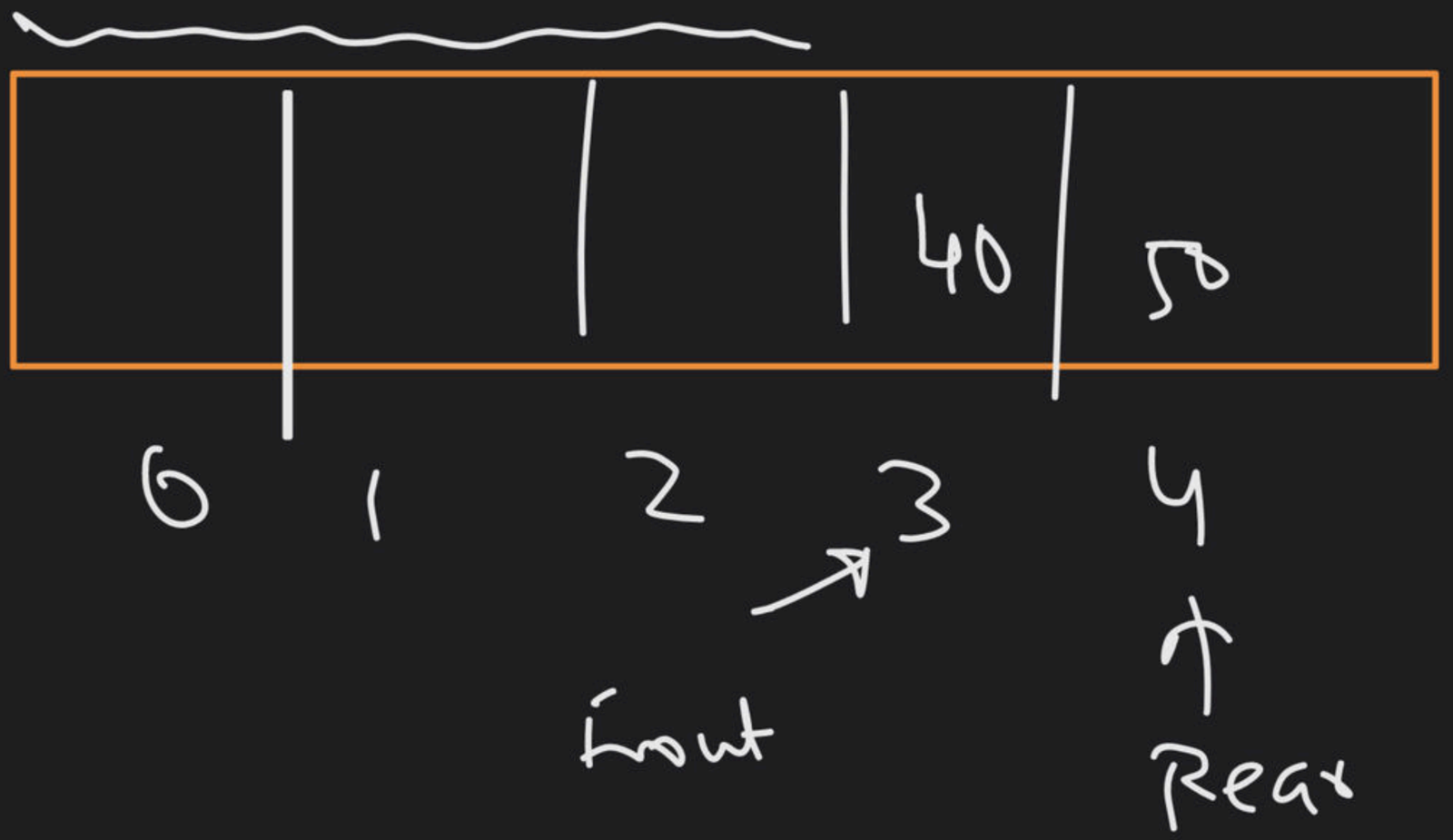
	Front	Rear
	2	2
Insert(40)	2	3
Insert(50)	2	~~3~~ 4
Delete ()	3	4
Delete	4	4



Delete()

Front = -1

Rear = -1



Insert()

Insertion ⇒ EnQueue

Deletion ⇒ DeQueue

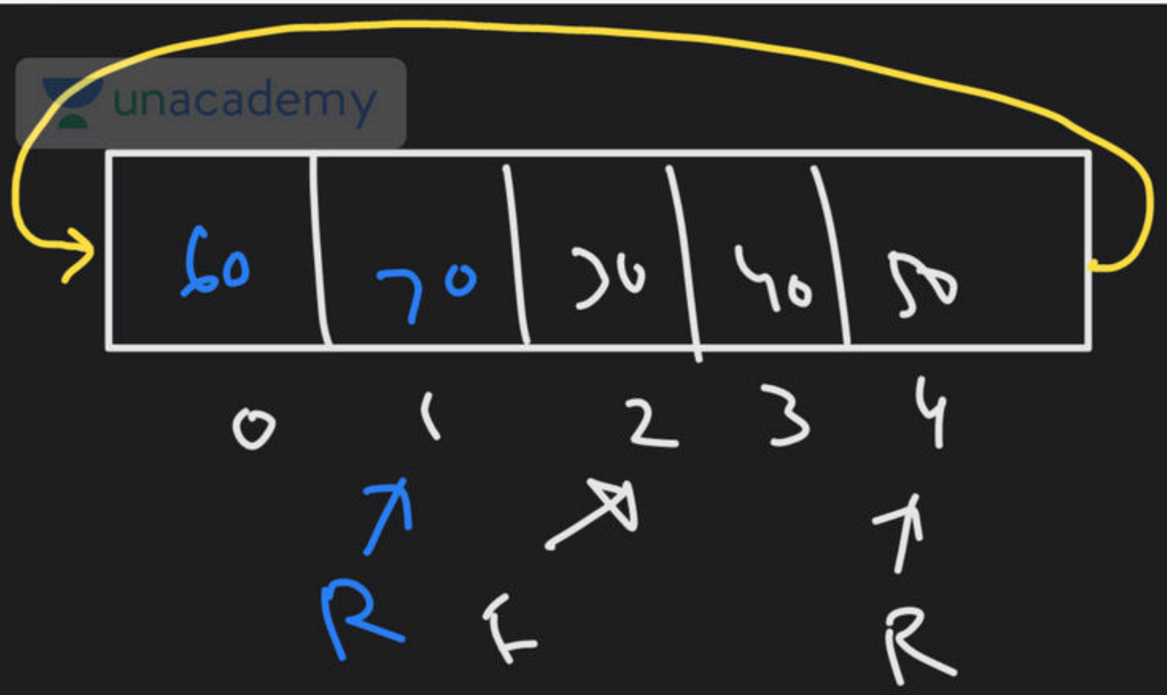
```
void Enqueue(int x){
if(Rear == SIZE-1)
 {
printf("Overflow");
return;
 }
else if (front == -1){
        Front = 0;
        Rear = 0; }
else
        Rear = Rear+1;
Queue[Rear] = x;
}
```

Constant time
O(1)

```
int Dequeue(){
    int temp;
if(front==-1){
    return INT_MIN;
}
else if(Front==Rear){
    temp= Queue[front];
    front = Rear = -1;
}
else{
    temp= Queue[Front];
    Front =Front+1;
}
return temp;
}
```

Queue is Empty

O(1)

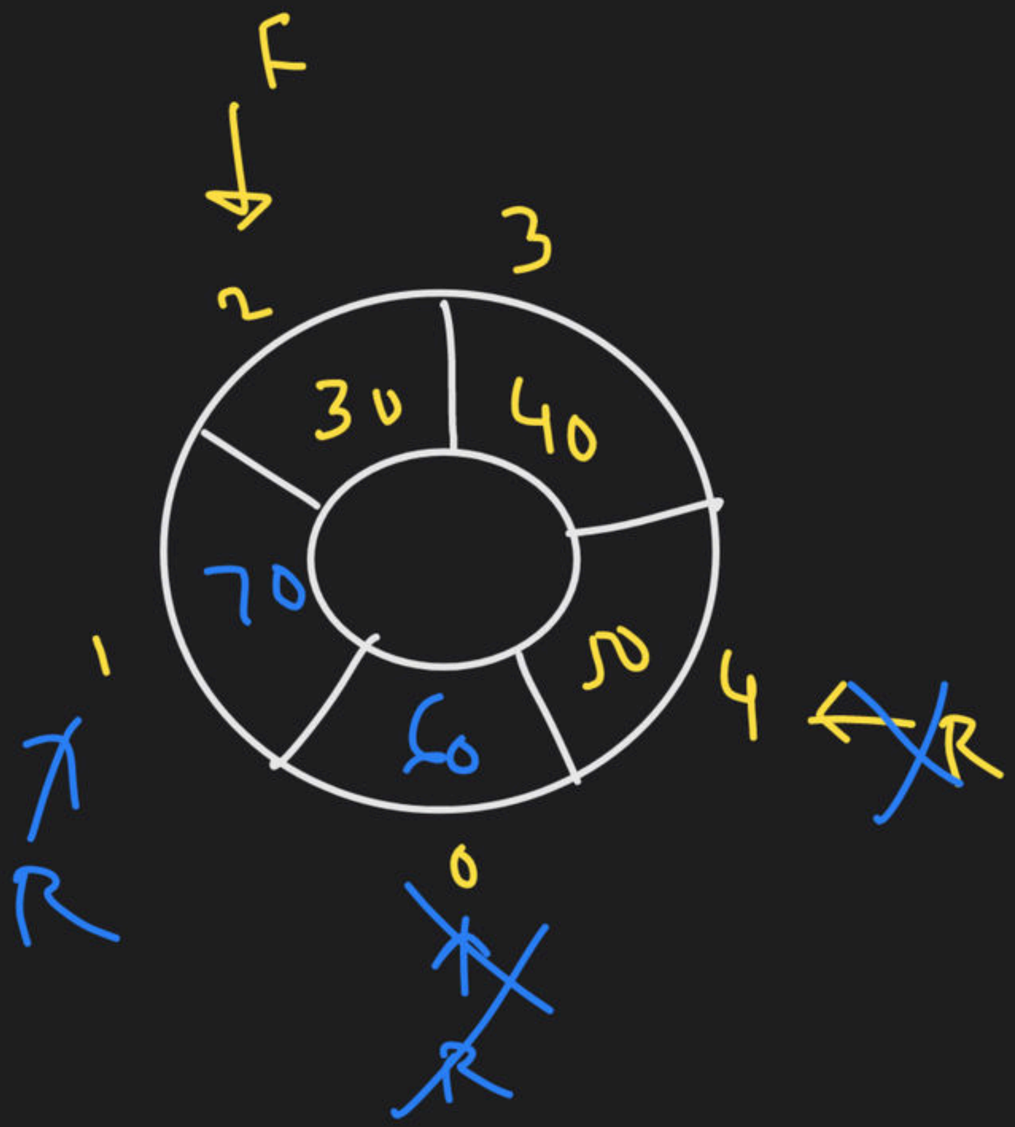


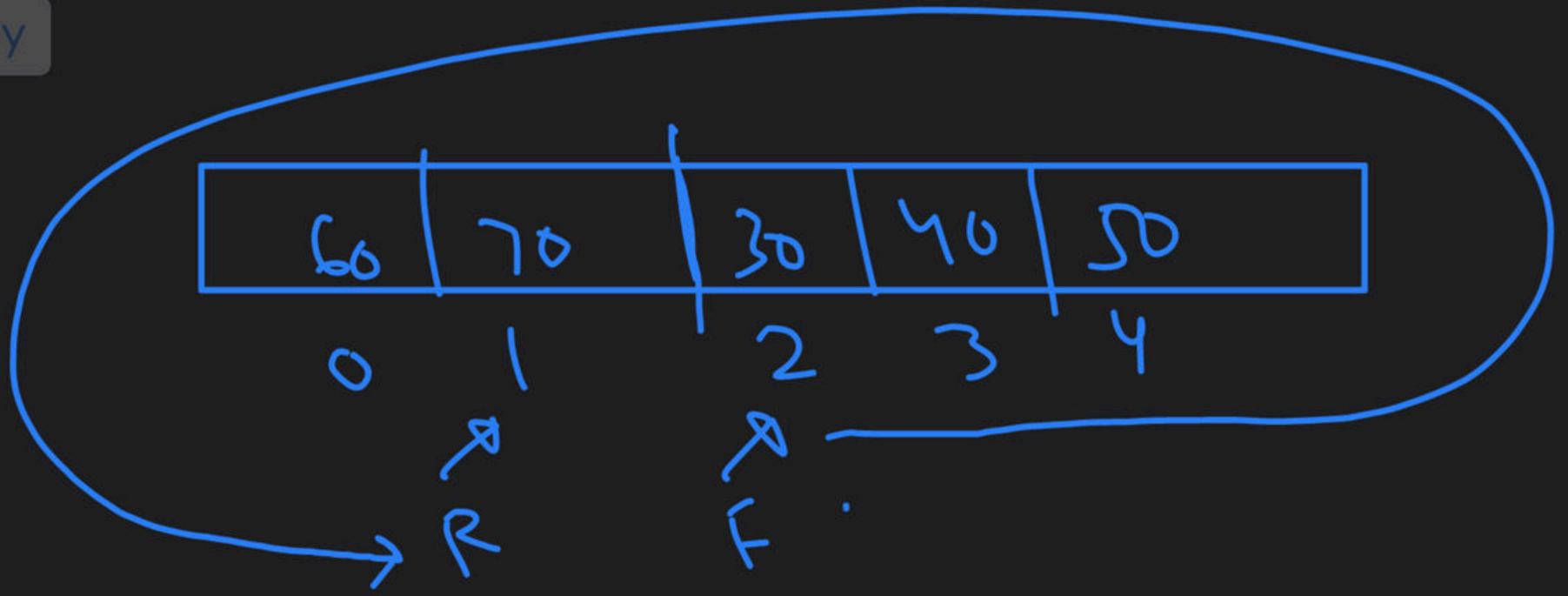
Insert(60)

$\Rightarrow$ Rear = 0

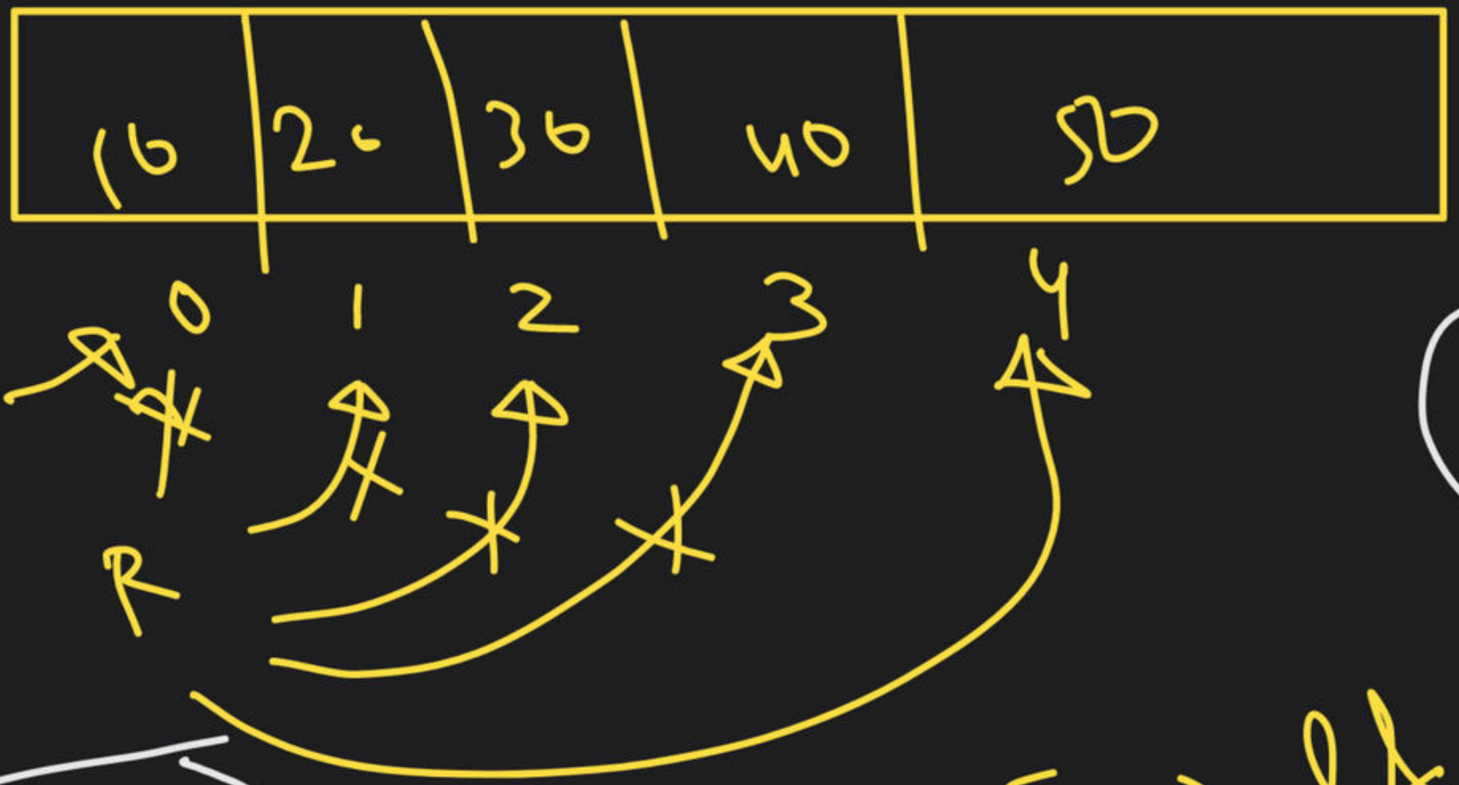
Queue[Rear] = 60

Insert(70)





$F_{new} = (R + 1) \% n$



(41) = 45

F = 0 && R = SIZE-1

F = R+1

F == (R+1) % SIZE

```
void CQ_Enqueue(int x){
    if ( Front == (Rear+1) % SIZE)
        return;
    else if ( Rear == SIZE-1)
        Rear = 0;
    else if ( Front == -1)
        Rear = Front = 0;
    else
        Rear++;
    Queue[Rear] = x;
}
```

Full

Empty

Not Full

O(1)

Q-deletion

① if front == -1
underflow
return

② special :
front == Rear
Set front = Rear = -1;

③
front == SIZE - 1;
⇒ front = 0;

④
front ++

1 element
sure hain ki
kam se kam
2 ele. hain

Application

1. CPU scheduling
2. slow & fast device ⇒ Sync.
3. spooling

Double Ended Queue

Priority Queue → Trees

unacademy

THANK YOU!

Here's to a cracking journey ahead!