CS & IT ENGINEERING

Data Structures

Stack and Queues

Lecture No.- 06

By- Pankaj Sharma Sir

Recap of Previous Lecture

Topic Stack and Queues Part - 05

Simple Queue
Circular Queue

Topics to be Covered

Topic Stack and Queues Part - 06

- Priority Queue
- PYQs

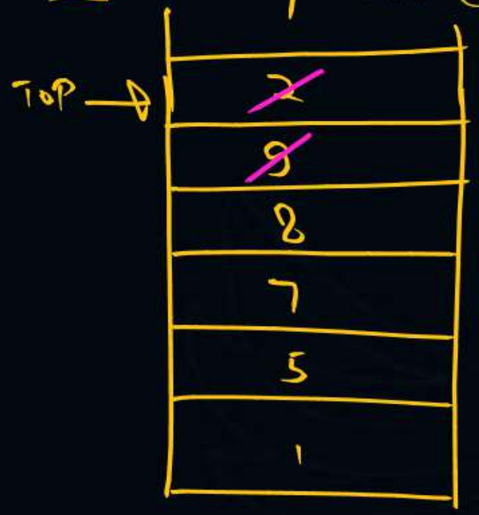
Topic : Stack and Queues

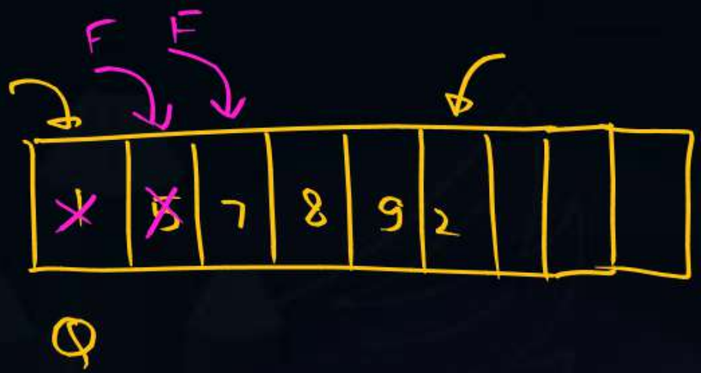
Q1: 2023-211

$a_0 = 1, a_1 = 5, a_2 = 7, a_3 = 8, a_4 = 9$
$a_5 = 2.$

S, Q } both Empty.

(i) Push ele from a_0 to a_5 in that order into S.
(ii) Enqueue the ele. from a_0 to a_5 in that order into Q.
(iii) Pop an ele from S.
(iv) Dequeue on ele from Q
(v) Pop an ele from S.
(vi) Dequeue an ele from Q
(vii) Dequeue an ele. from Q and push the same ele on S.
(viii) Repeat (vii) 3 times
(ix) Pop an ele from S.
(x) Pop an ele from S.





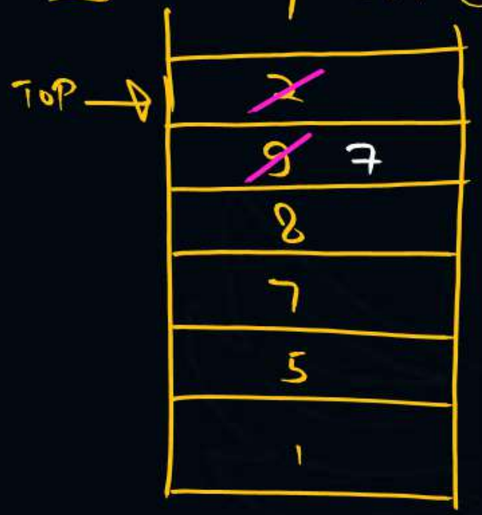
Topic : Stack and Queues

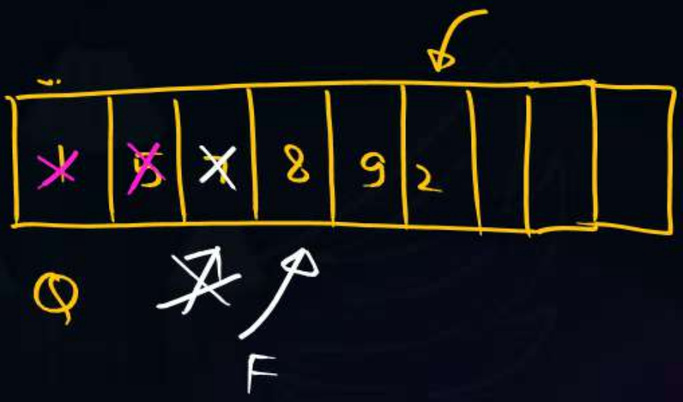
Q1. 2023-211

$a_0 = 1, a_1 = 5, a_2 = 7, a_3 = 8, a_4 = 9$
$a_5 = 2.$

S, Q } both Empty.

(i) Push ele from a_0 to a_5 in that order into S.
(ii) Enqueue the ele. from a_0 to a_5 in that order into Q.
(iii) Pop an ele from S.
(iv) Dequeue on ele from Q
(v) Pop an ele from S.
(vi) Dequeue an ele from Q
(vii) Dequeue an ele. from Q and push the same ele on S.
(viii) Repeat (vii) 3 times
(ix) Pop an ele from S.
(x) Pop an ele from S.





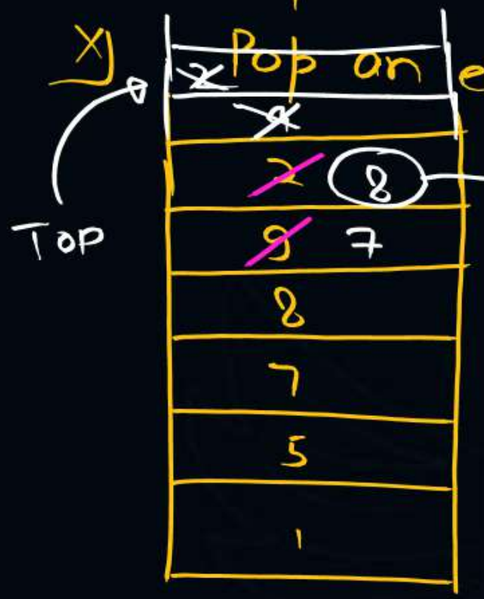
Topic : Stack and Queues

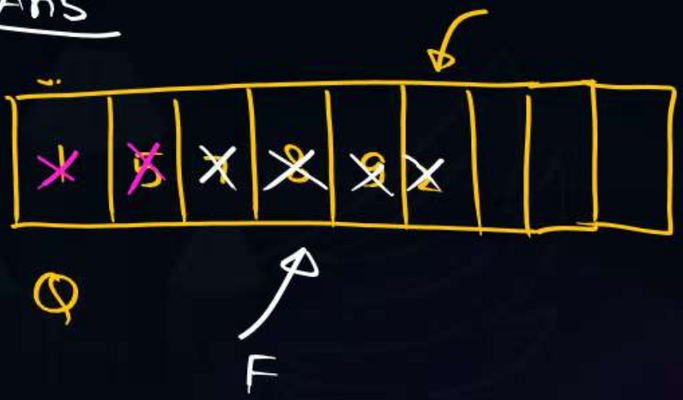
Q1. 2023-211

$a_0 = 1, a_1 = 5, a_2 = 7, a_3 = 8, a_4 = 9$
$a_5 = 2.$

S, Q } both Empty.

(i) Push ele from a_0 to a_5 in that order into S.
(ii) Enqueue the ele. from a_0 to a_5 in that order into Q.
(iii) Pop an ele from S.
(iv) Dequeue on ele from Q
(v) Pop an ele from S.
(vi) Dequeue an ele from Q
(vii) Dequeue an ele. from Q and push the same ele on S.
(viii) Repeat (vii) 3 times
(ix) Pop an ele from S.
(x) Pop an ele from S.





Gate. 2014
2M

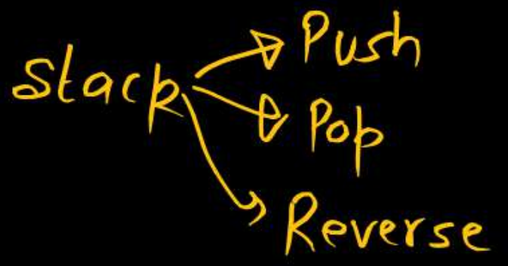
Suppose, a stack implementation supports an instruction REVERSE, which reverse order of elem. on stack, in addition to PUSH & POP instruction.

A) A queue cannot be implem. using this stack.

B) A queue can be implem. where ENQUEUE takes 1 inst. & DEQUEUE takes a seq. of 2 inst.

C) ✓ A queue can be implem. where ENQUEUE takes a seq. of 3 inst. & DEQUEUE takes 1 inst.

D) A queue can be implem. where both ENQUEUE & DEQUEUE takes 1 inst. each.

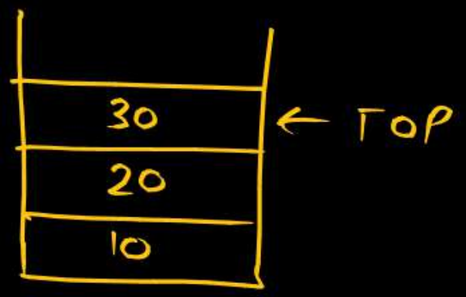


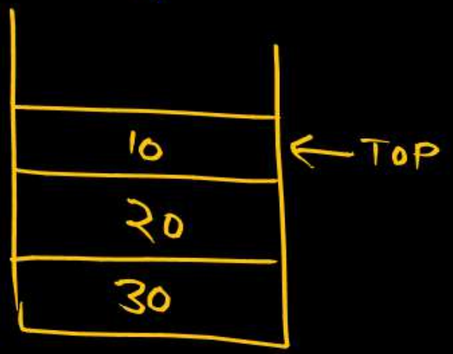
Push(10)

Push(20)

Push(30)

Reverse

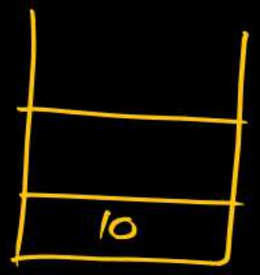




Q → 10,20,30,40

Stack: Enqueue is ⇒ Push

(i) Enqueue(10) → Push(10)

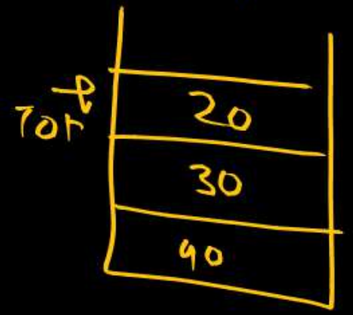


(ii) Enqueue(20) → Push(20)

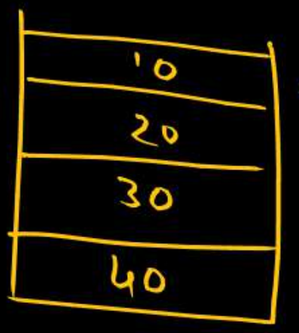
(iii) Enqueue(30) → Push(30)

(iv) Enqueue(40) → Push(40)

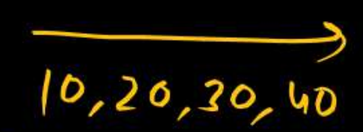
Reverse



(iii) Pop()



(i) REVERSE



dequeue? → FIFO ⇒ 10 delete

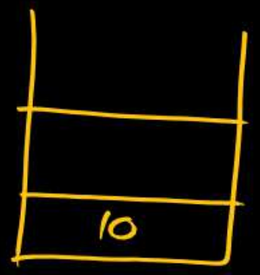
↓

we want 10 at top

Q → 10, 20, 30, 40

Stack: Enqueue is ⇒ Push

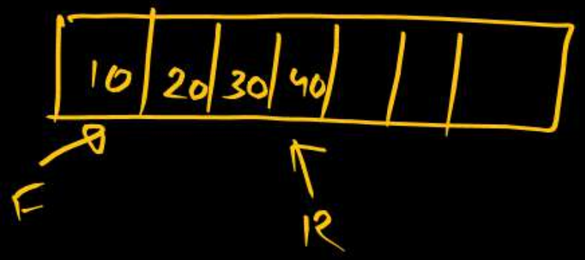
(i) Enqueue(10) → Push(10)



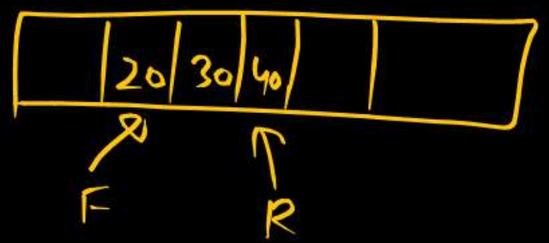
(ii) Enqueue(20) → Push(20)

(iii) Enqueue(30) → Push(30)

(iv) Enqueue(40) → Push(40)

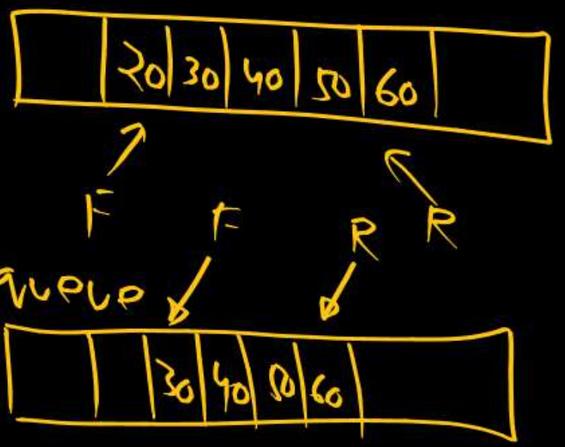


dequeue



Enqueue 50

Enqueue 60



10, 20, 30, 40

deque

~~10~~, 20, 30, 40

Enquee 50, 60

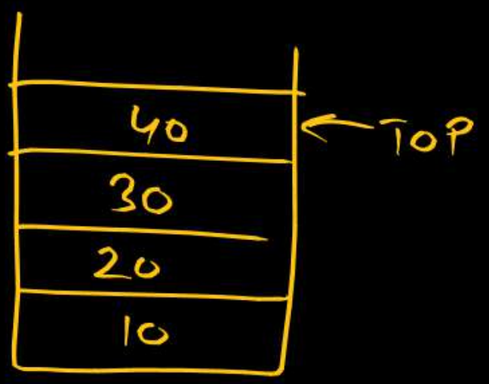
20, 30, 40, 50, 60

deq:

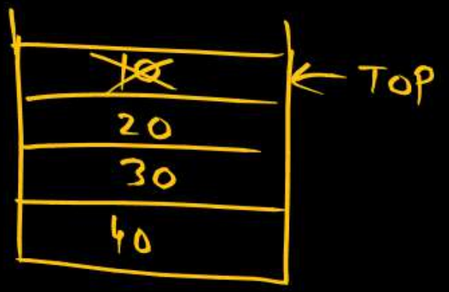
~~20~~, 30, 40, 50, 60

Enq → Push

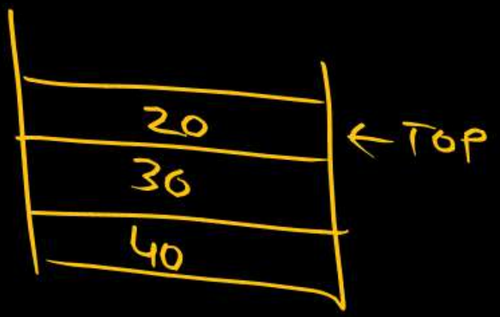
(i) Push 4



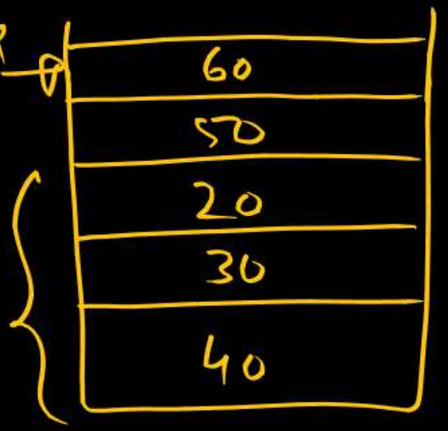
(ii) Reverse Pop()



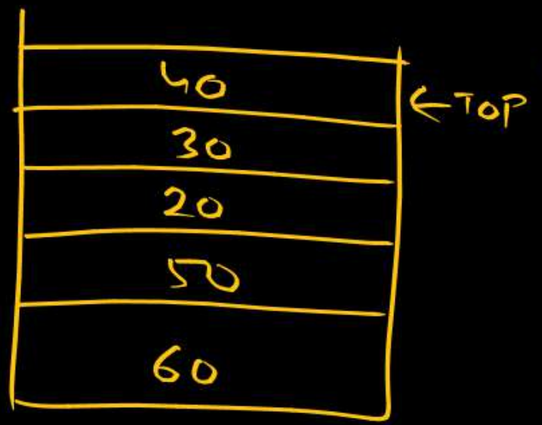
(iii)



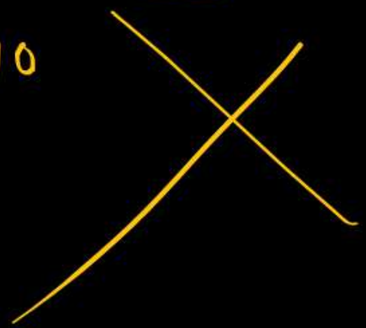
(iv)



Enq



Pop → 40



(i) 10,20,30,40

(ii) deque

~~10~~,20,30,40

Enqueue 50,60

20,30,40,50,60

deq:

~~20~~,30,40,50,60

(i) Enqueue → Push

(ii) Dequeue → a) Reverse
b) Pop()
c) Reverse

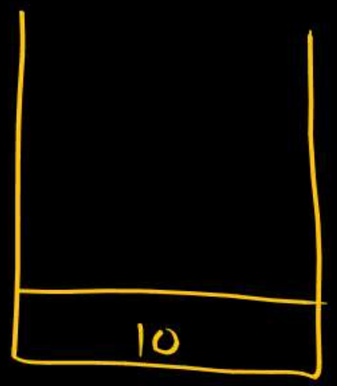
≡

(i) Dequeue → Pop

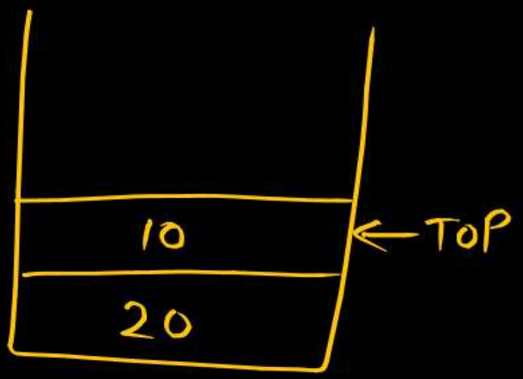
(ii) Enqueue → Reverse
Push
Reverse

Enqueue → (i) Reverse
(ii) Push
(iii) Reverse

① Enqueue 10

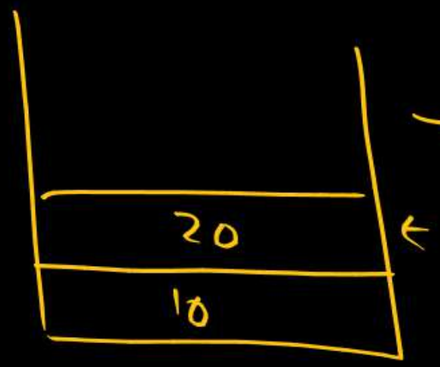


Reverse



Bholu

② Enqueue 20



Gate - 2021

Empty stack

Push(54)

Push(52)

Pop();

Push(55);

Push(62);

s = Pop();

Empty Queue

Enqueue(21);

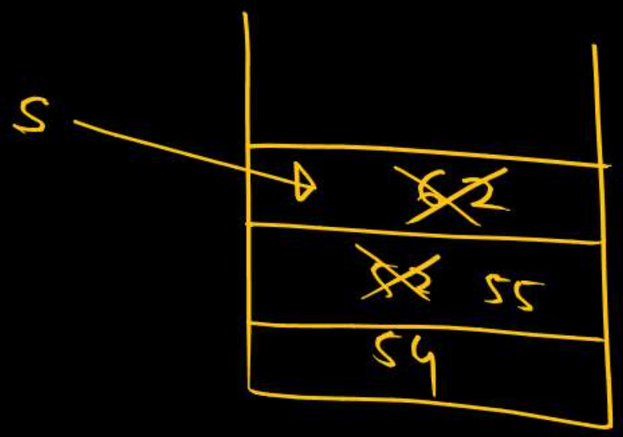
Enqueue(24);

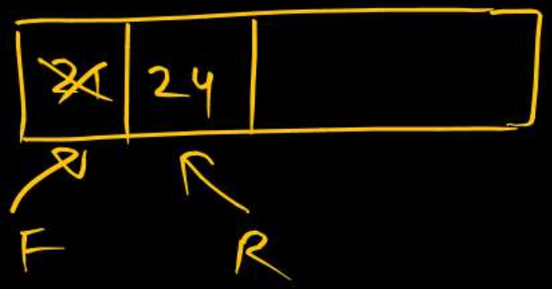
dequeue();

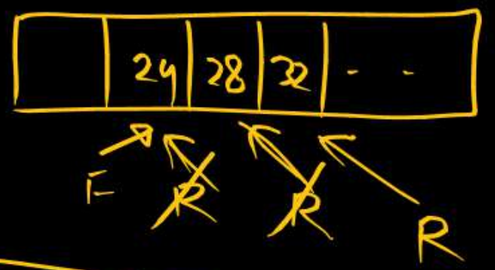
Enqueue(28);

Enqueue(32);

q = dequeue();







q = 24

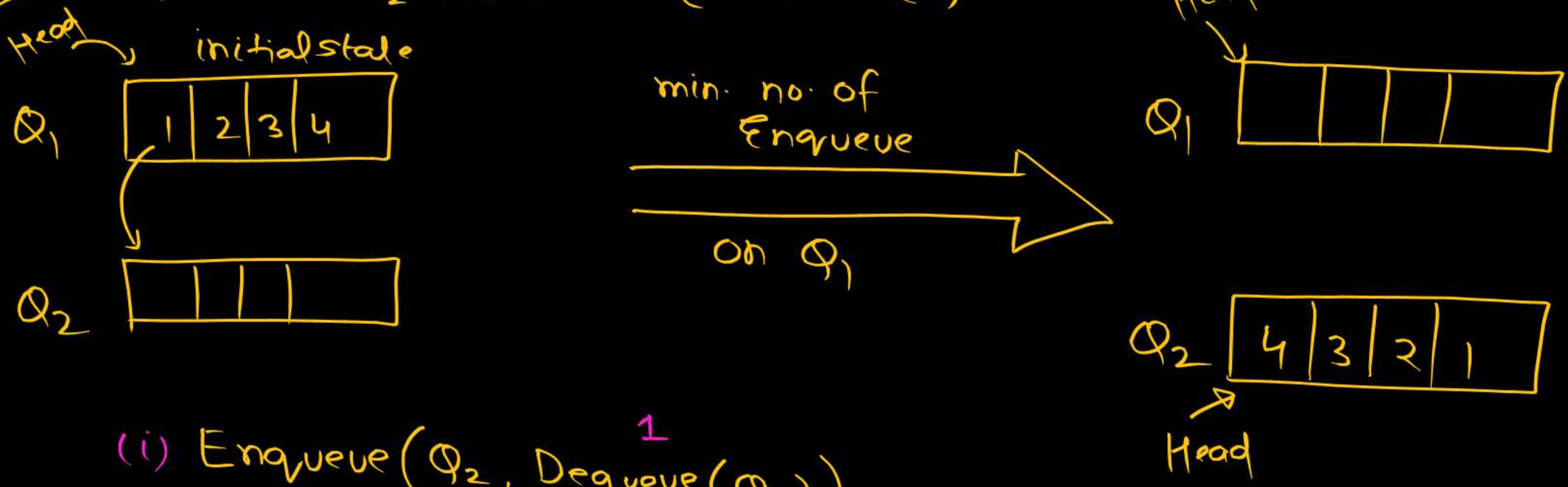
S + q is 86

62 + 24

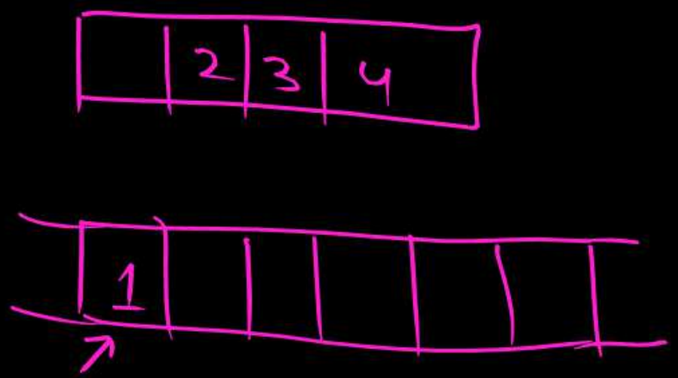
Gate 2022

Q) $Q_1 \rightarrow$ 4 ele, $Q_2 \rightarrow$ None

{ Enqueue(Q, element)
Dequeue(Q) }



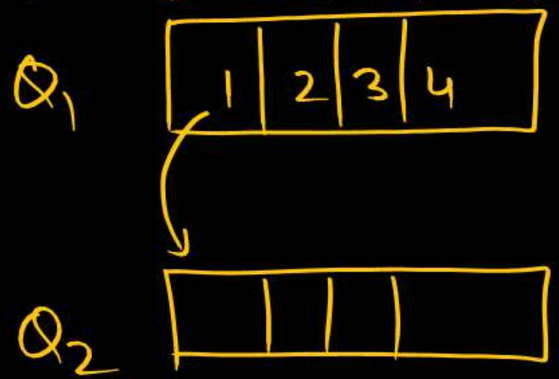
(i) Enqueue(Q_2, Dequeue(Q_1)) — 1



Gate 2022

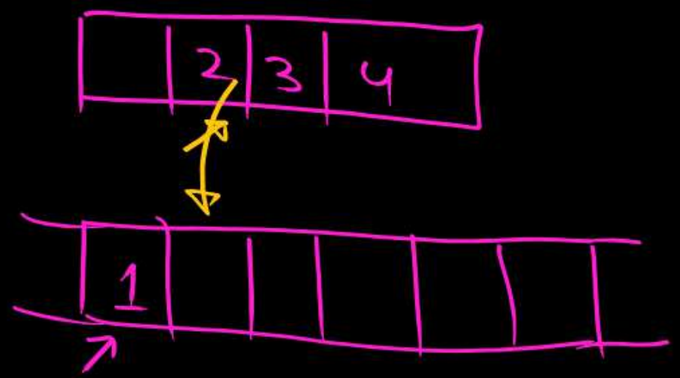
Q) $Q_1 \rightarrow$ 4 ele, $Q_2 \rightarrow$ None

Head → initial state



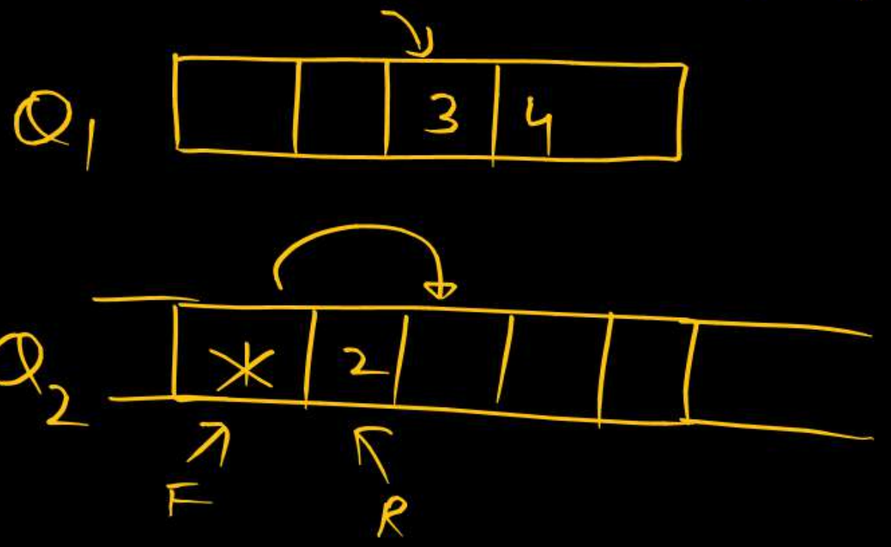
(i) Enqueue(Q_2, Dequeue(Q_1))

1



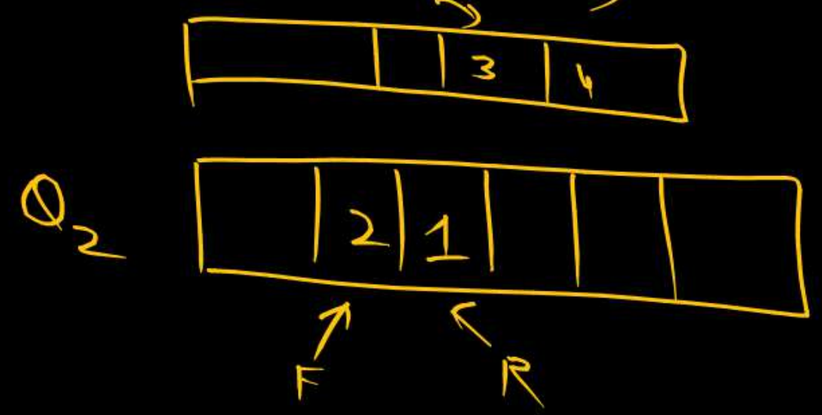
(ii) Enqueue(Q_2, Dequeue(Q_1))

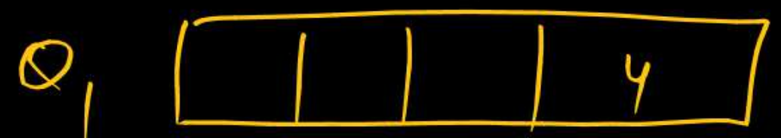
2

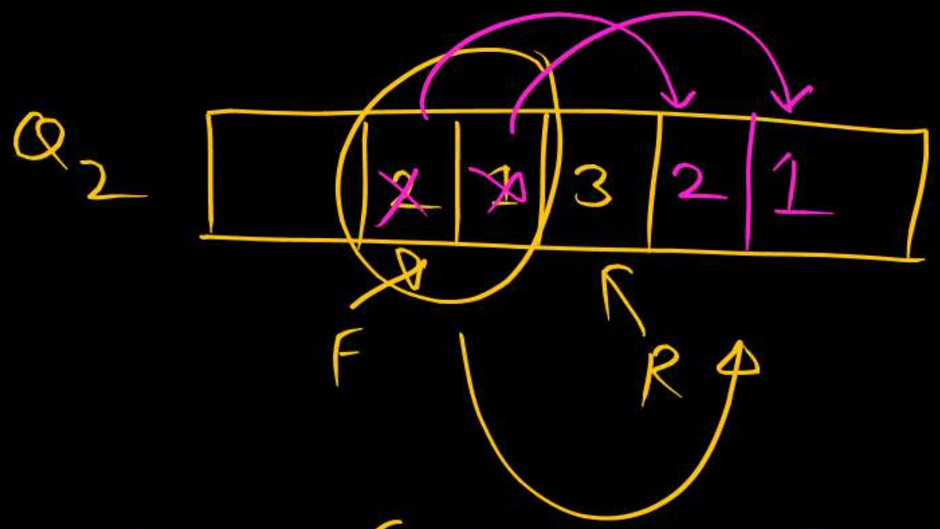


Enqueue(Q_2, Dequeue(Q_2))

1

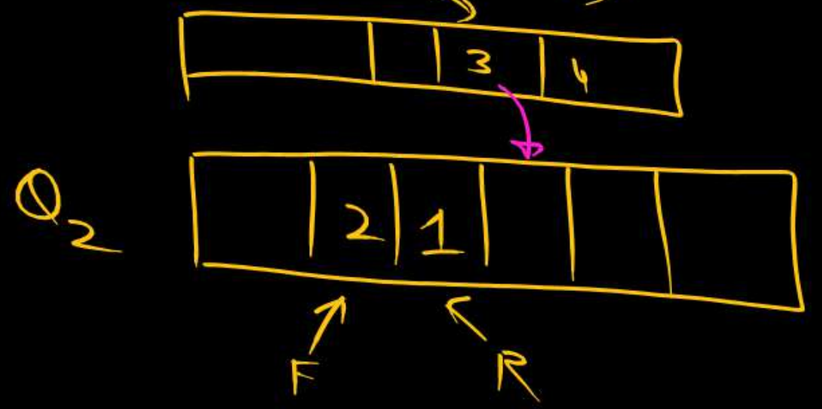






0 insert

Enqueue(Q_2 , Dequeue(Q_2))
1



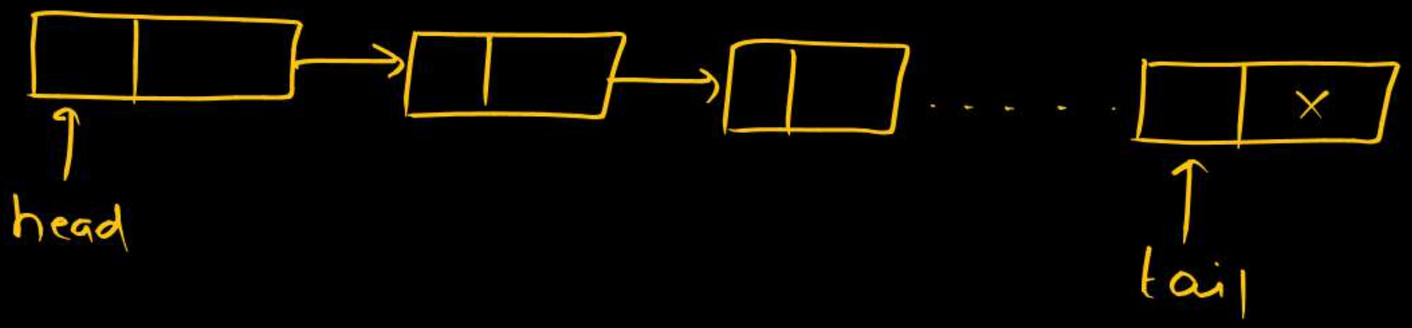
Enqueue(Q_2 , Dequeue(Q_1))

Enqueue(Q_2 , Dequeue(Q_2))
2

Enqueue(Q_2 , Dequeue(Q_2))

Q

n: no. of nodes in queue



Enqueue: Insertion at head

dequeue: deletion from tail

constant

Traverse

n

a) $\Theta(1), \Theta(1)$

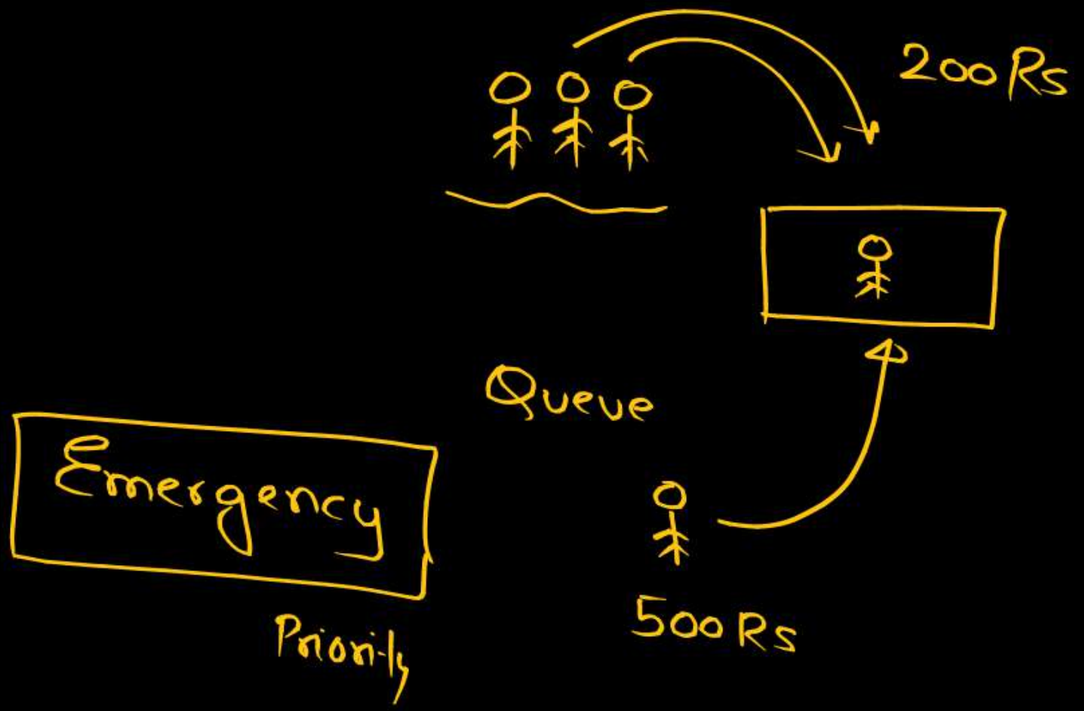
b) $\Theta(1), \Theta(n)$

c) $\Theta(n), \Theta(1)$

d) $\Theta(n), \Theta(n)$

Priority Queue

A priority is associated with each element.



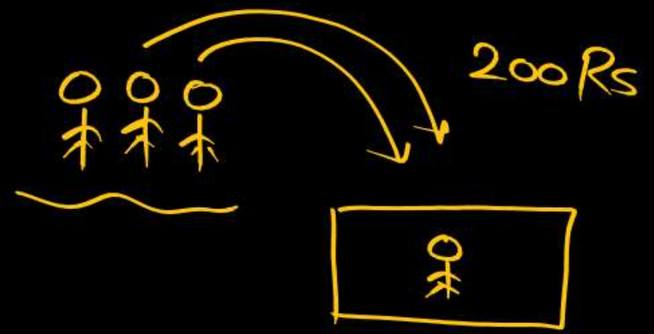
Priority Queue

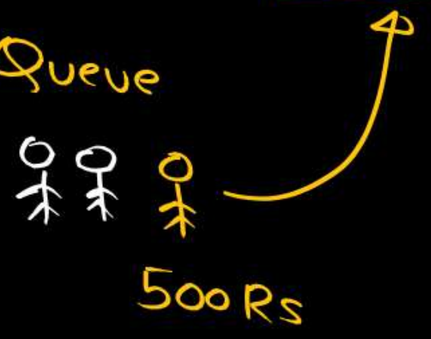
A priority is associated with each element.

2 ele have same priority

↓

order of arrival

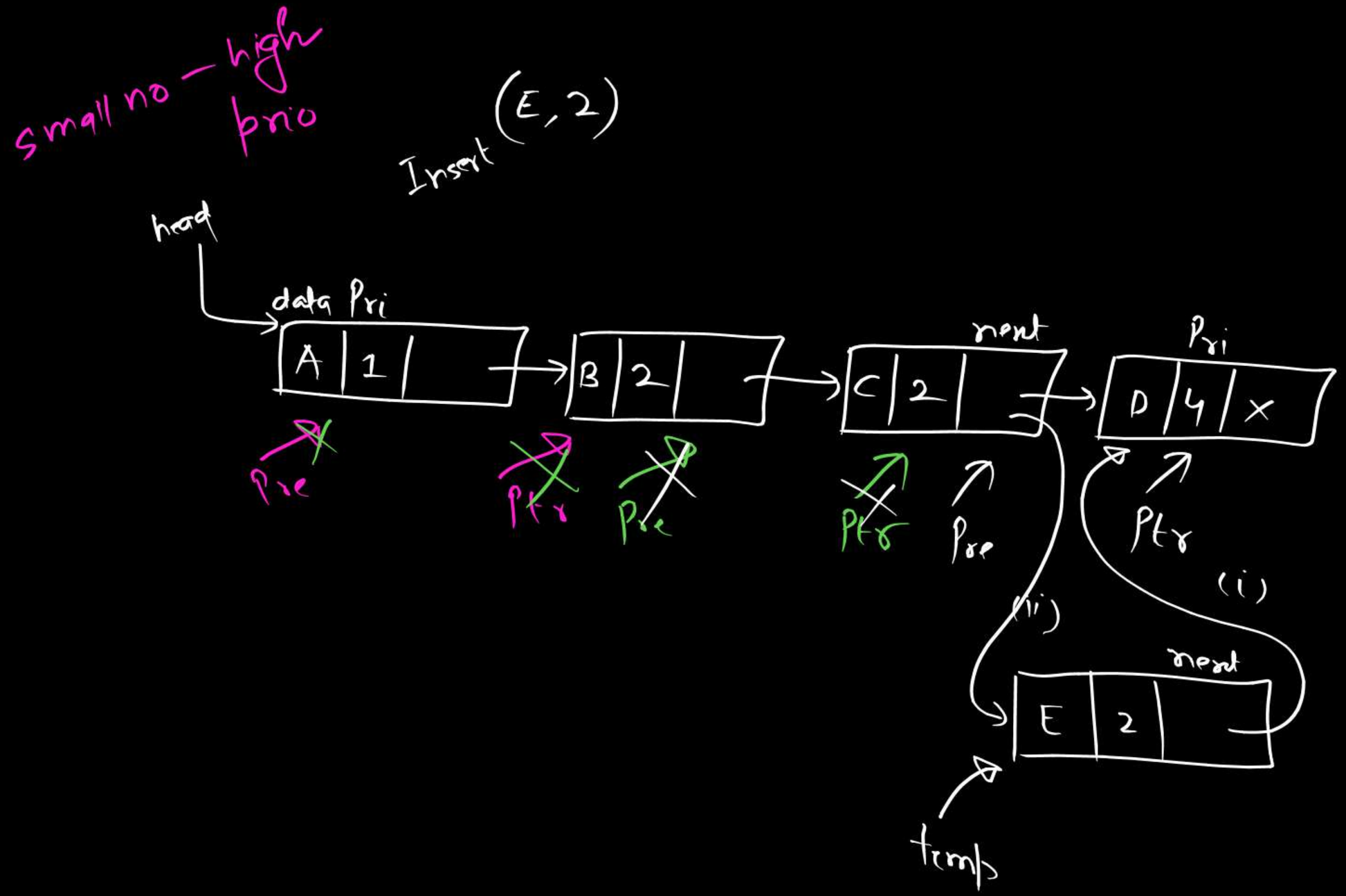




An ele. of high priority is processed before an ele. of low priority.

Ele. with same priority are processed as per their order of arrival.

High priority
- small no → high
  large no → low
- small no → low
  large no → high


Small no – high prio
Insert (E, 2)
head
data Pri
A 1
B 2
C 2
next
Pri
D 4 X
Ptr
Pre
Ptr
Pre
Ptr
Pre
Ptr
(i)
(ii)
next
E 2
temp

A Priority Queue Q is used to imp. a stack S that stores characters.

Push(c) is imp. as Insert(Q, c, k), where k is an appropriate integer key choosen by imp.

Pop() is imp. as DELETE_MIN(Q). For a seq. of operations the key choosen are in :

A) Non – inc. order

B) Non – dec. order

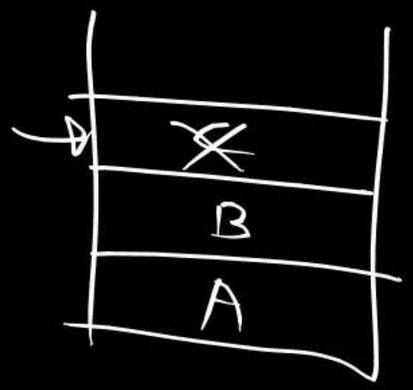
C) strictly inc. order

D) strictly - dec. order

Push(A) = Insert(Q, 'A', 1)

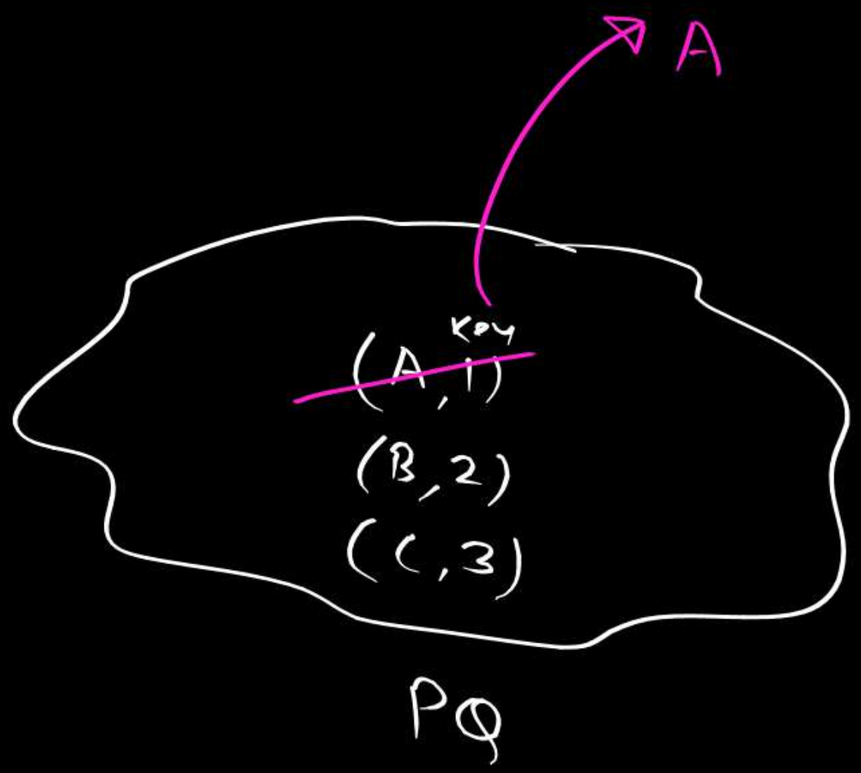
Push(B) = Insert(Q, 'B', 2)

Push(C) = Insert(Q, 'C', 3)



Pop ⇒ C

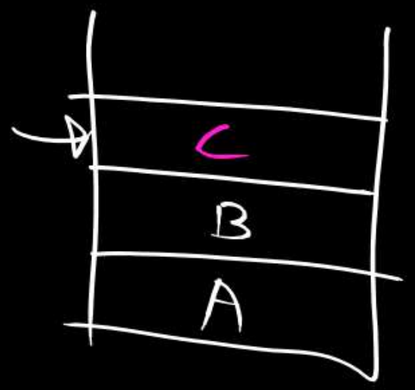
Pop() ⇒ DELETE_MIN(Q)



Push(A) = Insert(Q, 'A', 3)

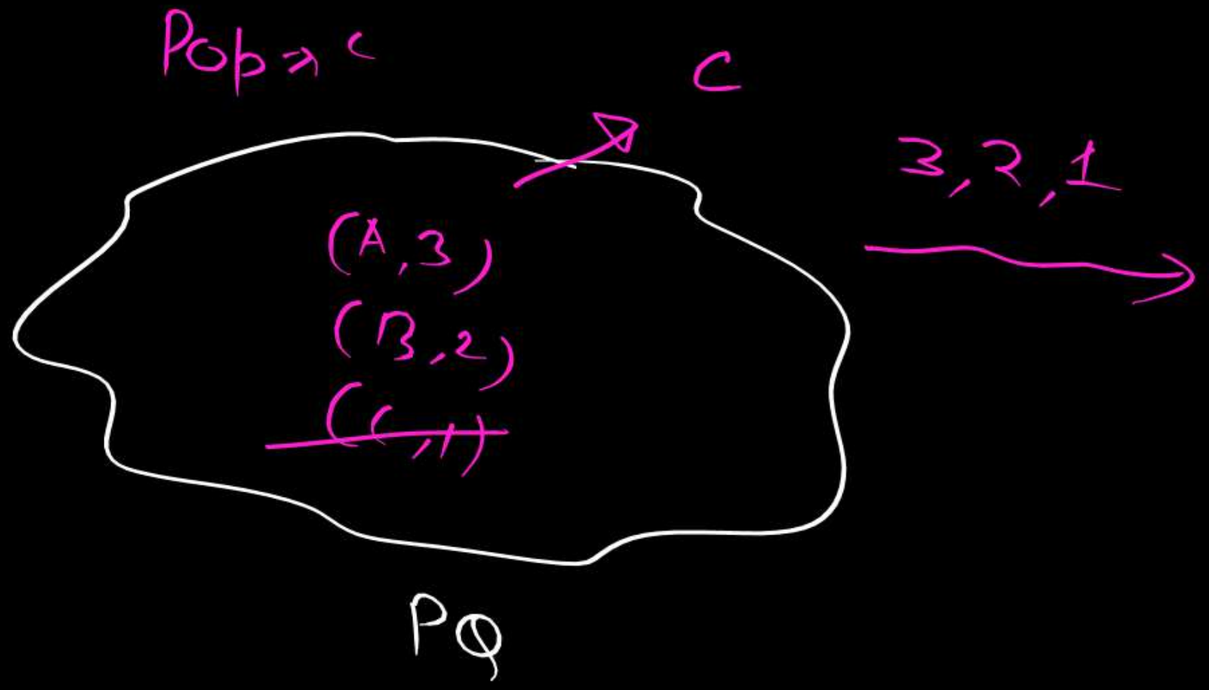
Push(B) = Insert(Q, 'B', 2)

Push(C) = Insert(Q, 'C', 1)



Pop ⇒ C

Pop() ⇒ DELETE_MIN(Q)

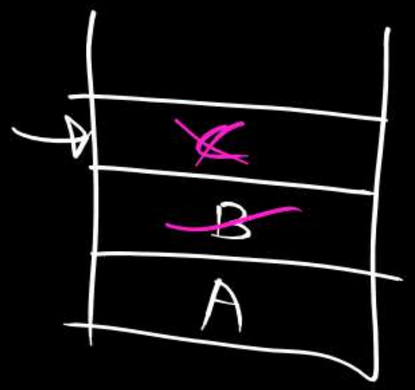


non-inc 3,3,2

Push(A) = Insert(Q, 'A', 3)

Push(B) = Insert(Q, 'B', 3)

Push(C) = Insert(Q, 'C', 2)

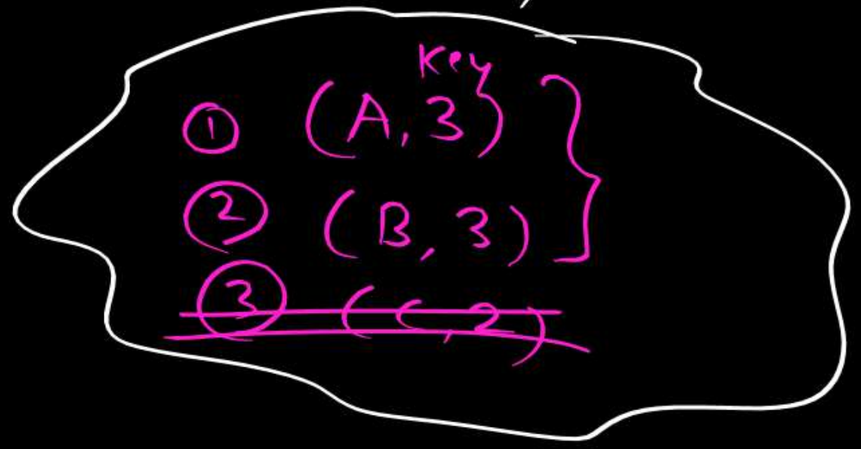


Pop ⇒ C

Pop() ⇒ DELETE_MIN(Q)

3,3,2 ✗

3,2,1 ✓



PQ

Pop → A

Pop → B

PQ

THANK - YOU