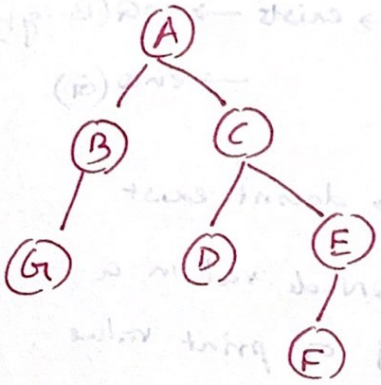
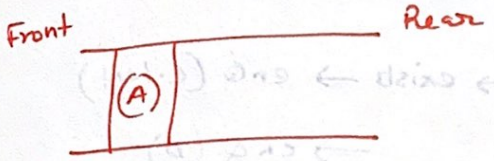
LEVEL ORDER TRAVERSAL



The Level Order TRAVERSAL for This is :-

A B C G D E F

Level Order Traversal is usually implemented with a Q. Start by enQ The Root.



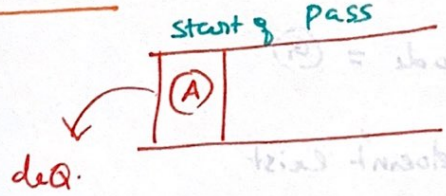
The iteration proceeds as long as The Q is not empty.

For JS, a queue can be simulated using an array via

→ push. (enQ in Rear)

→ shift (deQ in front)

PASS ①

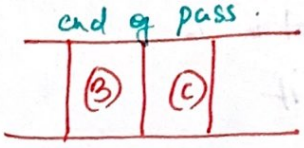


deqNode = (A)

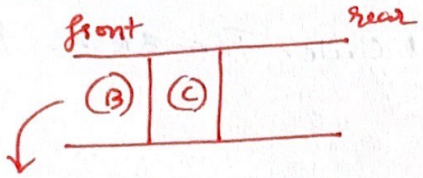
A.left → exists ⇒ enQ (A.left)

A.right → exists ⇒ enQ (A.right)

print (A) or store A.val in result.



Pass (2) :-

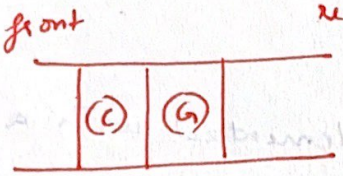


deQNode = (B)

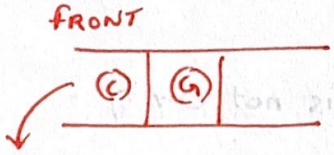
B.left → exists → enQ(B.left)
→ enQ(G)

B.right → doesn't exist

store deQNode.val in a result[] or print value.



Pass (3) :-

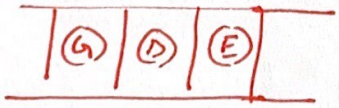


deQNode = (C)

C.Left → exists → enQ(C.left)
→ enQ(D)

C.Right → exists → enQ(C.Right)
→ enQ(E)

Store deQNode.val in a result[] or print.



Pass (4) :-

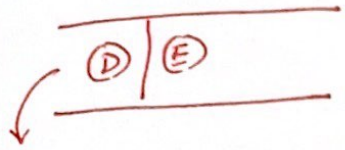
deqNode = (G)

G.Left → doesn't exist

G.Right → doesn't exist

Store deqNode.val in a result[] or print it

Pass (5) :-

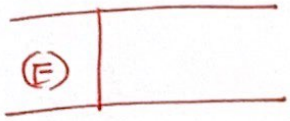


deqNode = (D)

D.Left → Doesn't exist

D.right → Doesn't exist

Either store the value in a result[] or print



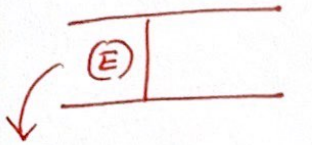
Pass (6) :-

deqNode = (E)

E. Left → exists → enQ (E.Left)

→ enQ (F)

E.right → doesn't exist.

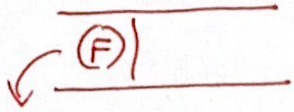


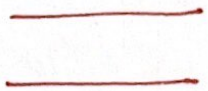
Pass (7) :-

deqNode = (F).

F.Left → NULL

F.right → NULL.





Pass (8) :-

The Queue is empty & The iteration stops.