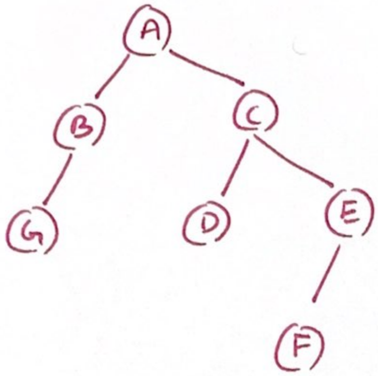
LEVEL ORDER TRAVERSAL

(ITERATIVE – WITH EVERY LEVEL REPORTED AS A NEW ARRAY)



Result:-

```
[
  [A],
  [B, C],
  [G, D, E],
  [F]
]
```

There ARE multiple ways of doing This, one is, maintaining a snapshot size of The Q, of The next level & always iterating only That 'x' which represents a snapshot size of The number of elements in That Level.

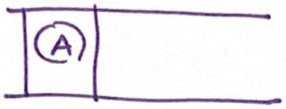
We will have 2 loops:-

Ⓐ Loop ① => iterating over N levels.

Ⓑ Inner Loop ② => iterating over 2^k elements in each of Those levels.

The iteration begins wiThe The ROOT Ⓐ as ①st element of The Queue.

Queue =



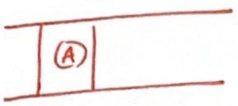
size of Queue = 1

The Result [] is a 2D array

level = 0

(As Root => Level = 0)

PASS ①:-

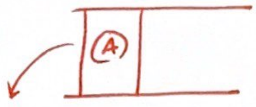


Start of Pass ①

Q size = 1.

Note That These main passes or top level loops That represent number of levels Can be used to create a new result[] per level & also keep track of level Count.

PASS ①A



deqNode = Ⓐ

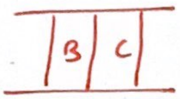
A.Left ? TRUE. => enQ(A.Left) => enQ(B)

A.right ? TRUE => enQ(A.right) => enQ(C)

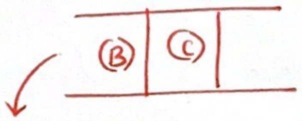
result[0].push(A.val);

--size

level



PASS ① B



But at This point our size variable is zero. So The inner loop terminates.

Whatever's in The Queue, belongs to The Next Level & we have Completed iterating Current Level.

Re-Compute size of The Queue.

Since The inner Loop has terminated, Before The next pass of outer loop, re-compute size, as whatevers in The Queue is The full list, comprising of Next level elements.

$$\boxed{size = Q.length = 2}$$

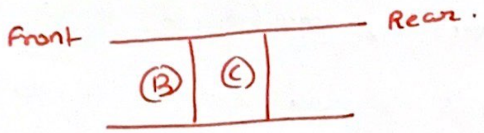
Increment Level => $\boxed{level = 1}$

Pass - ②

size = 2.

level = 1

result[level] = [] (A new entry in The 2D result[] is created to push elements of level = 2)

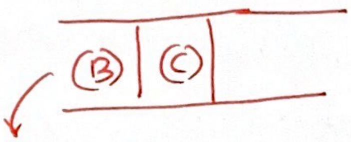


Pass 2A

size = 2

deqNode = (B)

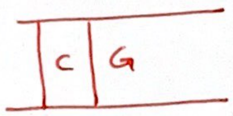
B.Left ? TRUE => enQ (B.Left)
=> enQ (G)



B.right ? False .

result[level]. push (B.val)

--size .

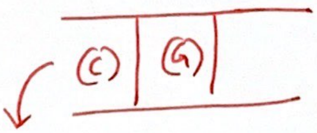


Pass 2B

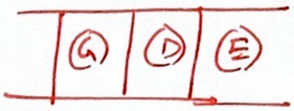
size = 1

deqNode = (C)

C.Left ? TRUE => enQ (C.Left)
=> enQ (D)



C.Right ? TRUE => enQ (C.Right)
=> enQ (E)

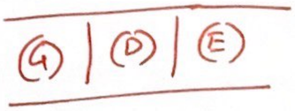


result[level]. push (C.val);

--size .

Pass 2c

size = 0



At This point, size = 0
So The inner Loop terminates
whatever exists in The Q,
belongs to The Next Level &
The iteration for Level (1) is over.

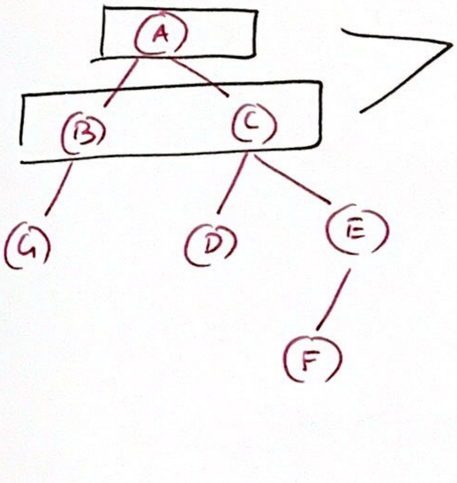
Re-Compute size :-

Since The inner loop terminated, Before The Next pass of outer loop, re-compute size, as, whatever exists in The queue now, is The full list of next level elements.

sizE = Q.lengTh = 3.

Increment level => level = 2

So, far,



These are over & The Result array is

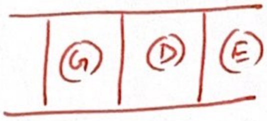
```
[
  [A],
  [B, C]
]
```

Pass-(3)

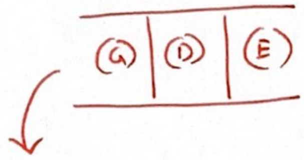
size = 3

level = 2

result[level] = [] (A new entry in The 2D result[]
created to push elements
of level = 2)



Pass 3A **size = 3**



deqNode = G

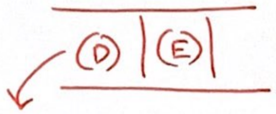
G.left ? false

G.Right ? false.

result[level].push(G.val)

--size.

Pass 3B **size = 2**



deqNode = D

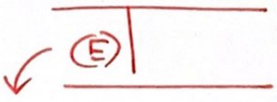
D.left ? false.

D.Right ? false.

result[level].push(D.val)

--size.

Pass 3C **size = 1**



deqNode = E

E.left ? TRUE => enQ(E.left) => enQ(F)

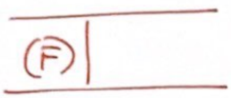
E.Right ? false.

result[level].push(E.val)

--size.

Pass 3D

size = 0



size = 0

=> end of Current level = 2

Whatever elements are left in The Queue, belong to the next level.

The iteration for level = 2 is over & inner loop is terminated.

Re-compute size

Since The inner loop has terminated, Before The next pass of outer loop, Compute size again That represents snapshot of next level elements Count.

size = Q. length = 1

Increment level => Level = 3.

Pass - 4

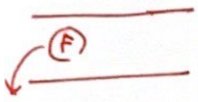
size = 1

Level = 3

result [level] = [] (A new entry in The 2D result [] created to push elements into Level = 3)

Pass 4A

size = 1



deqNode = F

F. Left ? False

F. right ? False.

result[level].push (F.val)

--size.

Pass 4B

size = 0

size = 0 => end of Current level.

∴ compute size for next Level.

=> size = 0 & level = 4

inner loop Breaks. outer loop checks if Q has elements => outer loop ~~empty~~ Breaks as Q is empty. => end of iteration.