Recap : DS → Binary Tree → Recursive codes (8)

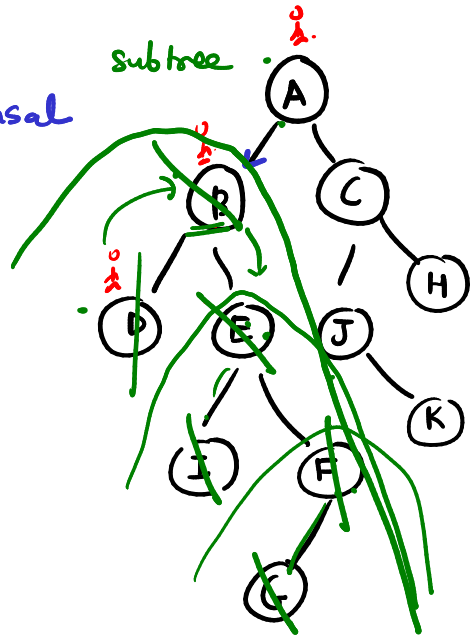
↳ CRUD

↓

assume first → have a tree | Read. / Traversal

BT is the 1st non-linear DS. ∵ non-linear → read → multiple ways.

Manually. Read → Touch all the nodes.



Random Tree.

1. Breadth First Traversal → Level Order Traversal

A B C D E J H I F K G

2. Depth first Traversal (3 types)

(a) Preorder Traversal : root left right

A B D E I F G C J K H

(b) Inorder Traversal : left root right

D B I E G F A J K C H

(c) Postorder Traversal : left right root : D I G F E B {K J H C} A

LST RST

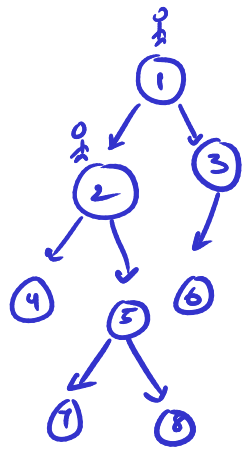
Code : depth first traversals :

```
void preorder (root):
    if root == null : return
    print (root.data)
    preorder (root.left)
    preorder (root.right)
```

f^n (root)
// Base
print data ①
f^n. (→ left) ②
f^n. (→ right) ③

① ② ③ → preorder
② ① ③ → inorder
② ③ ① → postorder

expect : 1 2 3 4 5 6 7 8



Template #1 : level Order Traversal.

connections are one way.

1 2 3 4 5 6 7 8 → LOT

crude :

```
q.push (root)
while (! q.empty):
    get 1st node of q   || Print it
    if left  → q.push(left)
       right → q.push (right)
```

level order traversal.

T.C.
$O(n)$

```
void level_order_traversal( root) :
    if root == null : return .
    queue <node> q ;
    q.push (root) ;
    while (!q.empty()) :
        int u = q.Front() ; u.pop() ;
        if u.left : q.push (u.left) ;
        if u.right : q.push (u.right) ;
```

int size = q.size() while
while size -- :
inside while
ends again and again !!

Why ?

```
while (!q.empty) :
    size = q.size()
    while (size--) :
        q.push (l)
        q.push (r)
```

→ level switch.

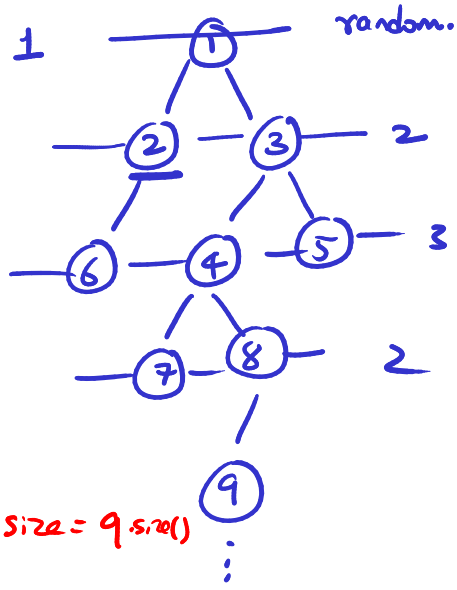
making changes to that particular level nodes →

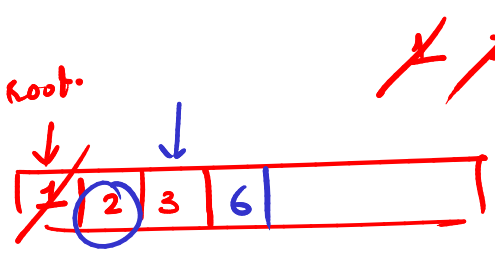
loop ends → q. → empty.

deterministic.
↳ every node goes through q only once.


```
for size no. of times :
    int u = q.front() , q.pop()
    if left : q.push (left)
    if right : q.push (right)
```

size of q ⇒ 1
size of q = 2
size of q = 3
size of q = 2
⋮





~~1~~ ~~2~~ ~~3~~ 6

q.push(root) size = q.size()
while (size--) :
✓ u = front() pop() ★ once
✓ left → q
✓ right → q
✓ size = q.size()

When is the level change happening ? →

```
while (!q.empty())
    ★ → empty (executed when level changes)
    int size = q.size()
    while size-- :
        u = front()   pop()
        left → q   right → q
```

as long as it is on the level.

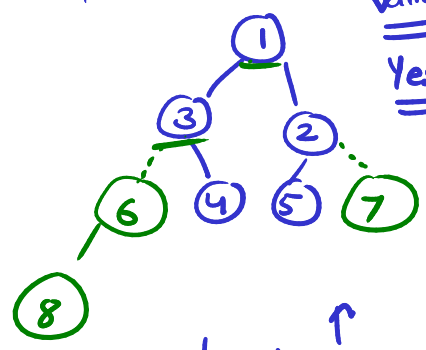
Variations. ① Create a tree CRUD / Insert a node.

Traverse the tree level order wise, and connect the node at 1st vacant position.

LoT
Insert (root, 6) → left = new node(6)
Insert (root, 7) →
Insert (root, 8)

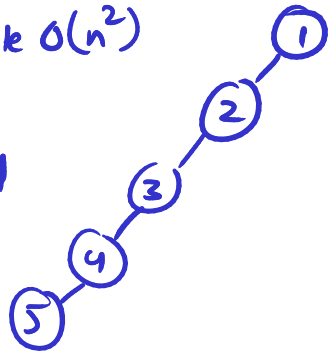
T.C Create $O(n^2)$

Valid? Yes



DFT Insert(root, 2) Create $O(n^2)$

× skewed



* Complete *. | $2^x - 1$ nodes perfect

```
Node insert (root, key) :
    Node new_node = new Node (key);
    if root == null :   root = new_node ; return root ;
    queue<Node> q :   q.push (root) ;
    while (!q.empty) :
        int u = q.front() ;   q.pop() ;
        if u.left == null :   u.left = new_node ; return root ;
        if u.right == null :   u.right = new_node ; return root ;
        if u.left :   q.push (u.left)
        if u.right :   q.push (u.right)
```

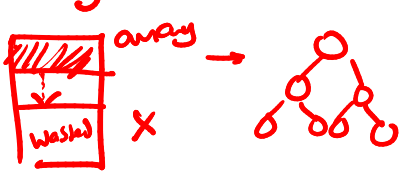
question: duplicates?

ans. Never put duplicates in a tree!

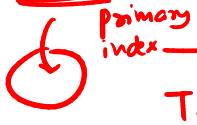
② Update update (root, value, new_value) :

```
if u.data == value :
    u.data = new_value
    return ;
```

Spoiler: hashing



B+ Trees → SQL data

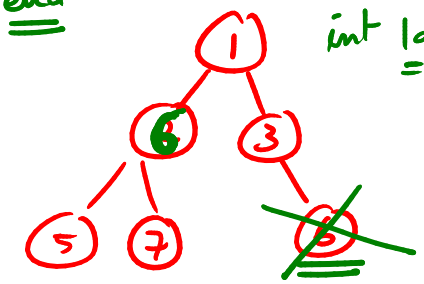


Trees are sets!

C++: Multiset multimap (×) Hacky.

③ Delete (hacky)

LOT end



int last = 6

delete (root, value) :

delete (root, 2) :

1. last node value of root. (remove node)
2. update (value, last) → update (2, 6)

node:
int data
node right
node left
int count

multiset multimap ←

— Interview questions —

Samsung.

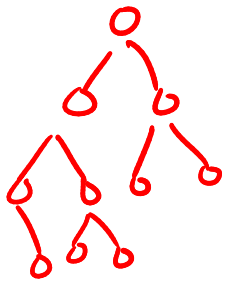
online test.

④ Width of the tree.

→ max nodes on a particular level.

o/p: 4

```
int width = 0
while (!q.empty):
    width = max(width, q.size())
    int size = q.size()
    while (size--)
        u
        → left
        → right
return width
```



→ 4

extra space

O(width)

⑤ Print corner nodes of tree.

→ v[0] v[v.length() - 1] v.reset()

while size--

u

vector v ← push(u)

q.front()
q.back()

docs → queue

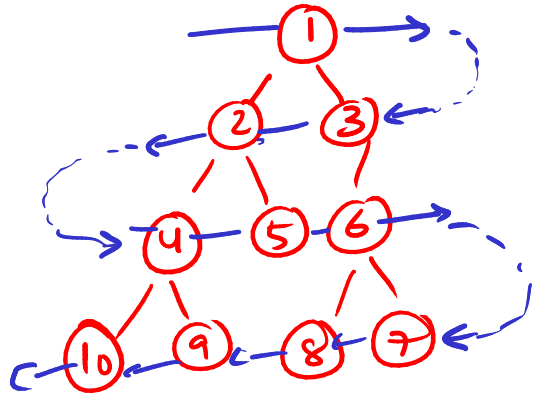
back ?

⑥ Sum of all leaf nodes.

if !u.left and !u.right : sum += u.data

7 Spiral order traversal

o/p : 1 3 2 4 5 6 7 8 9 10



ltor : true

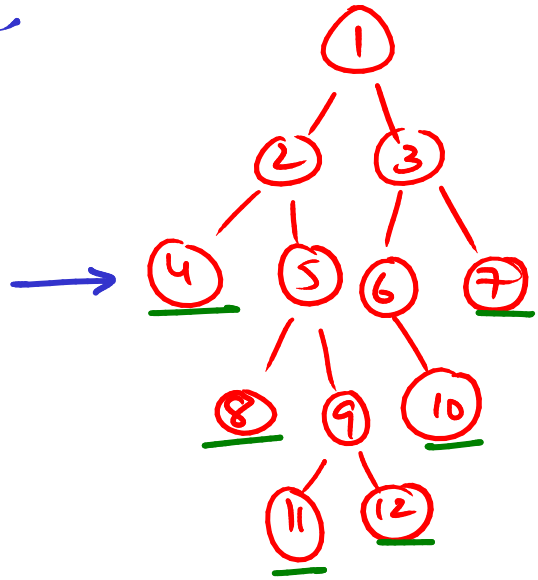
```
while (!q.empty()):
    ltor = !ltor
    if ltor:
        print array →
    else
        print array ←

    int size = q.size:
    while (size--)
        v.push(u) ...
```

GFG : 1 stack
1 queue ✗
2 stacks

8 find sum of nodes at min leaf level.

leaf nodes . min leaf level

sum = 4 + 5 + 6 + 7 = ___



```
while (!q.empty()):
    seen_a_leaf = false
    if seen_a_leaf :
        print sum(array)
        break
    while (size--):
        if !u.left and !u.right : seen_a_leaf = true
        ....
```

8 variations on LOT.