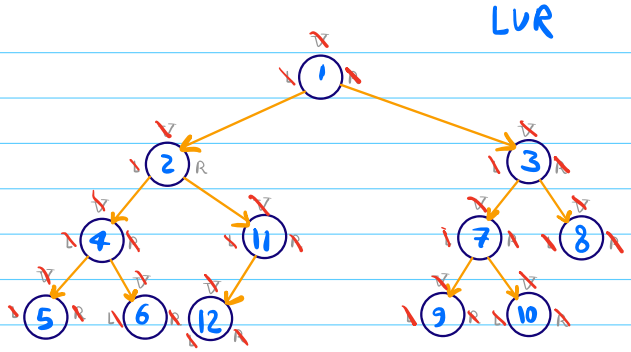
Topics: - Morris inorder - LCA in BT
- LCA
- LCA in BST

inorder traversal

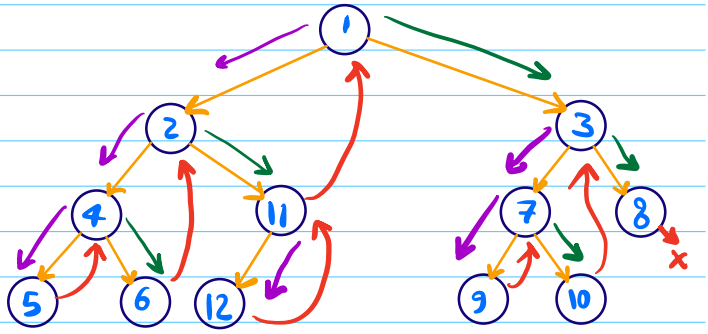
LVR



LVR ?

Morris inorder traversal. ⚠ Can be confusing & challenging to understand

no stack!



Note 1: Inorder traversal needs the path history whether recursive or iterative, we need a extra memory for this path info

Note 2: right most node in a tree has max 1 chid.

3 steps to build intution

operators = 5, 4, 6, 2, 12, 11, 1, 9, 7, 10, 3, 8, x

LVR

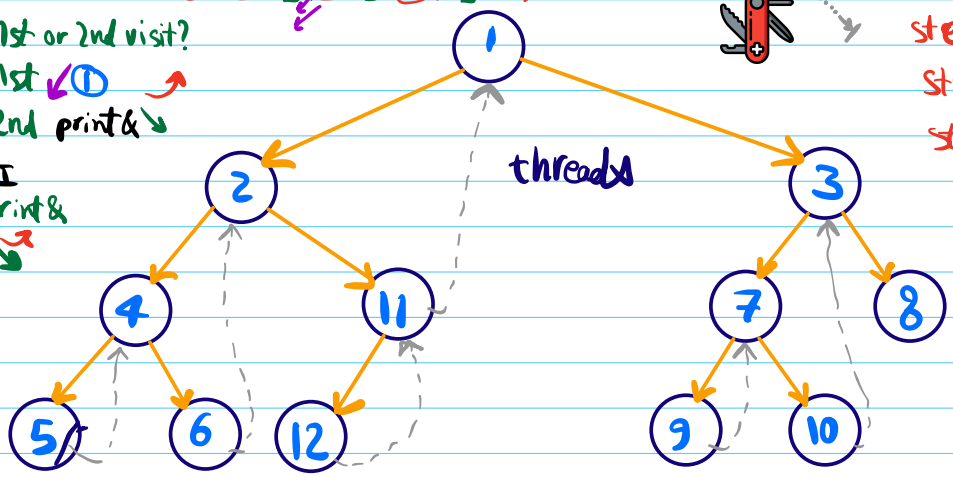
step1
step2
step3

1st or 2nd visit?
I 1st ①
II 2nd print &
III print &

① go left No print

② print & back jump

③ print & go right



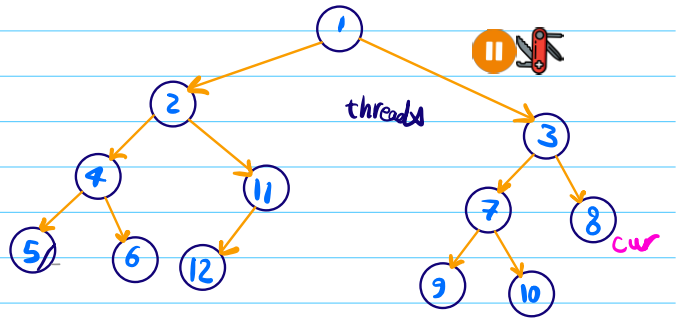
back jump ← Thread (Threaded tree)

Impatient Questions:
1 - How to form the back jump pointer? from which node to which node?
2 - How to use the back jump exactly for inorder?
3 - Should we clean it up? How?
4 - How to differentiate real right pointer from a back jump?
5 - How to code this idea?
6 - Does it change TC?
...

Code:

TC: O(n)
SC: O(1)

```
cur = root
while(cur != null){
    if(cur.left == null){ III
        print(cur.data)
        cur = cur.right
    }else{ either I or II
        R = rightMostNodeInLeftSubtree(cur);
        if(R.right == null){ // I (first time)
            R.right = cur // create
            cur = cur.left
        }
        else{ // II found cycle
            print(cur.data)
            R.right = null; // clean up
            cur = cur.right
        }
    }
}
```



1st or 2nd visit?
I 1st ✓ (1)
II 2nd print&
III print&

outputs:
5,4,6,2,12,11,1,9,7,10,3,8

```
Tnode rightMostNodeInLeftSubtree(cur){
    temp = cur.left
    while(temp.right != null && temp.right != cur){
        temp = temp.right
    }
    ret temp
}
```

right most
cycle

TC: O(n) + O(n) + O(n)

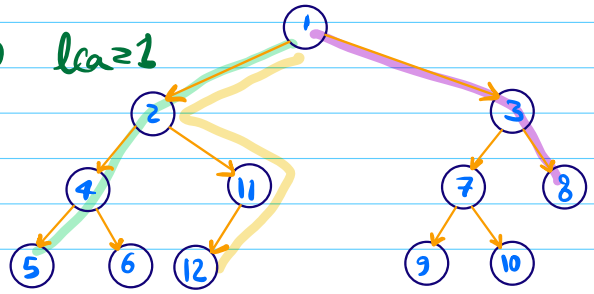
Lowest Common Ancestor 8

? All the nodes in chain from cur node to root

5 & 8

5 : 1, 2, 4, 5
8 : 1, 3, 8

lca = 1



& 5 : 1, 2, 4, 5
12 : 1, 2, 11, 12

lca = 2

& 5 : 1, 2, 4, 5
6 : 1, 2, 4, 6

lca = 4

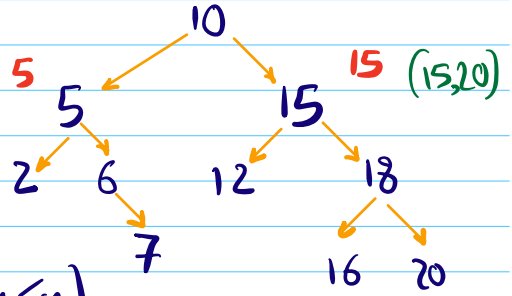
LCA(3, 9) = 3

LCA(8, 8) = 8

P1 Find LCA of two nodes (e.g. x, y) in a BST (distinct)

ex LCA(12, 20) = 15

LCA(7, 7) = 7



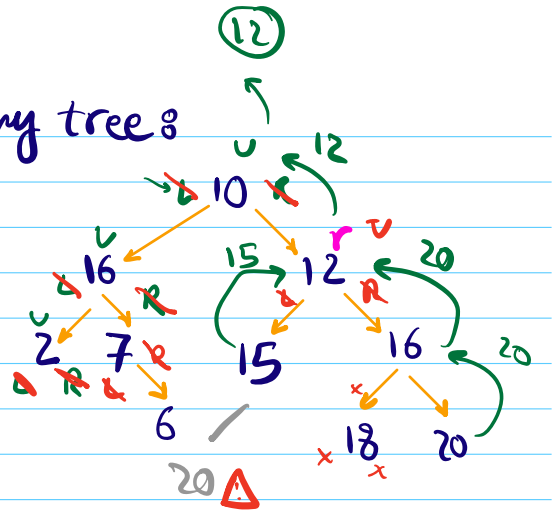
code

TC: O(H)

SC: O(1)

```
cur = root
while(cur != null){
    if( cur.data < x && cur.data < y)
        cur = cur.right
    else if( cur.data > x && cur.data > y)
        cur = cur.left
    else branching
        ret cur
}
```

P2 Find LCA in an unordered binary tree:



approach 1 & 2

LCA(15, 20) find Path O(n)

LRV

find first node both present

LRV leak for 15 } don't print

LRV leak for 20 } don't print

current root

LRV

```
Tnode LCA(Tnode r, int x, int y){
    if(r == null) ret null;
    if(r.data == x || r.data == y) ret r
    L = LCA(r.left, x, y) //L
    R = LCA(r.right, x, y) //R
    if(L != null && R != null) ret r; //V
    if(L != null) ret L
    if(R != null) ret R
    ret null
}
```

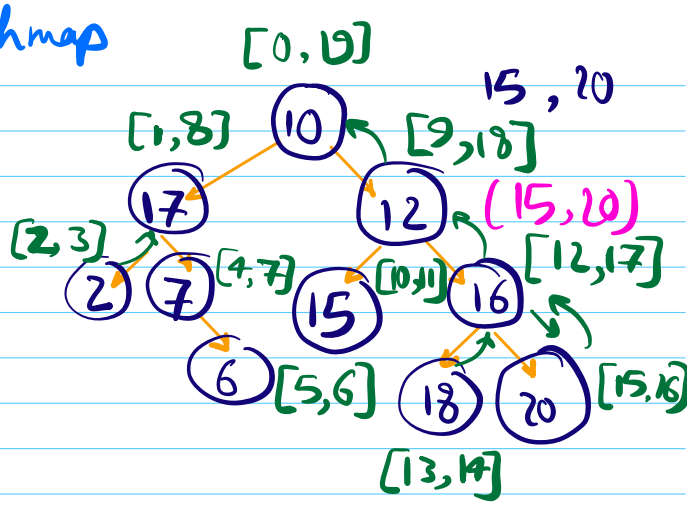
TC: O(n)
SC: O(H)

key <Tnode, int>

in[Tnode] } dictionary or hashmap
out[Tnode] }

LRV

approach 2 : Start time End time



LRV T=0

```
void traverse(root){
    if(root==null) ret;
    in[root] = T
    T++
    traver(r.left)  // L
    traver(r.right) // R
    out[root] = T   // V
    T++
}
```

TC: O(n)
SC: O(H)
HM: O(n) + O(ñ)

(12) is ancestor of 15?

[9,18] t1
[10,11] t2

t1 [s1, e1] t2 [s2, e2]

s2 > s1 && e2 < e1

* (2) was in the subtree of (1)

bool isancestor?

first null check

```
Tnode LCA(x,y){
    while(cur!=null){
        if( isancestor(x, cur.left) && isancestor(y, cur.left) )
            cur = cur.left
        else if( isancestor(x, cur.right) && isancestor(y, cur.right) )
            cur = cur.right
        else
            ret cur
    }
```

TC: O(H)
BST

Assignment → P3 Find the kth smallest element in a BST.

Hint: