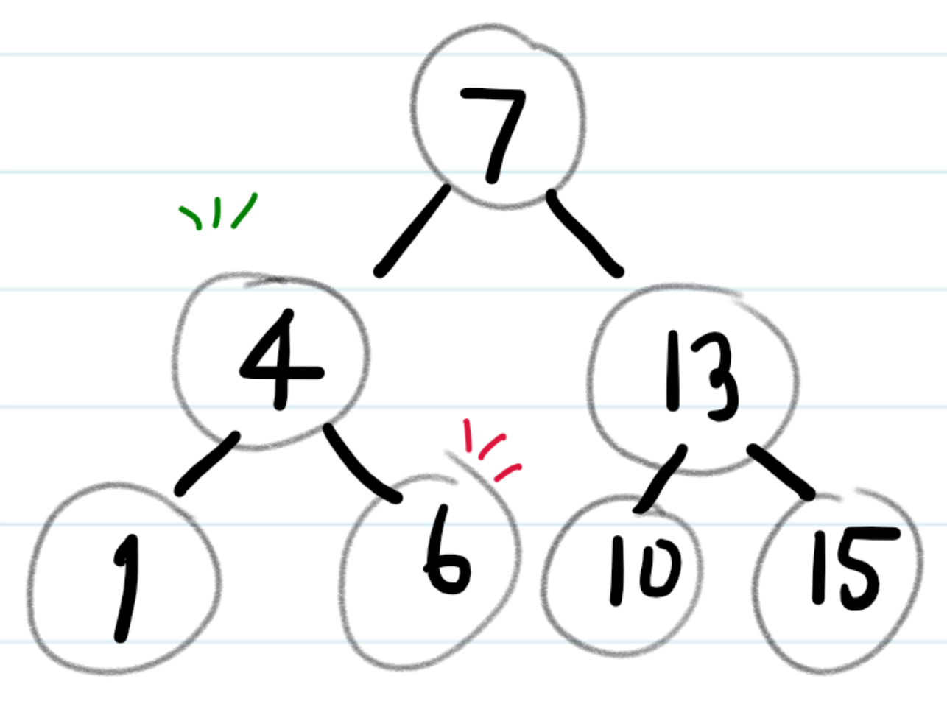
Data 27/8/2021

FindClosest Leaf

- Node ใหม่จะงอกออกมาจากใบ



FindClosestLeaf (4.2)
return Node(6) •

FindClosest (4.2)
return Node(4) •

ไม่เหมือนกันเด้อ~

FindClosest (ไม่แนะนำ recursive)

```
Node current = root
    while (current != null)
        current = current.left
```

Insertion Operation

ใช้ FindClosestLeaf (การบ้านคลับเชฟ)

```
r = FindClosest (inserting key)
r.key == inserting key
    return ( ซ้ำเหมือนกันโว้ย~~)
if (inserting key < r.key) เพิ่ม node ข้างซ้าย
if (inserting key > r.key) เพิ่ม node ข้างขวา
```

ไม่ใช้ FindClosestLeaf (O(h)) / Complete Binary Search (O(log n))
ความสูง
(อาจเขียนใน net bean ประกอบไปดูได้)

```
public Class Node{
    Node left;
    Node right;
```

ผู้จดไม่ทัน ไปดูในคลิปเอานะ T^T

```
public class Tree {
    Node root;
    public void insert(int key)
        if (root == null)
            root = new Node(key)
        else insert (root, key)
```

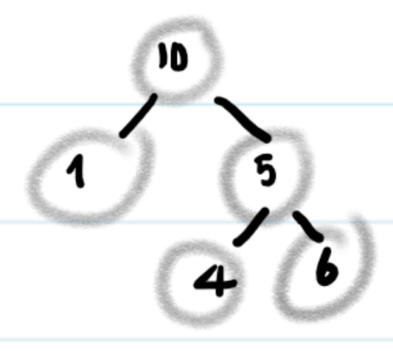
recursive

```
public static void insert (Node current , int insert_key)
    if (current.key == insert_key)
        print "Duplicate key"
    else if (insert_key < current.key)
        if (current.left == null)
            current.left = new Node(insert_key)
            current.left.parent = current
        else
            insert(current.left , insert_key)
    else
        if (current.right == null)
            current.right = new Node(insert_key)
            current.right.parent = current
        else
            insert(current.right , insert_key)
```

Deletion Operation

① ลบ root Node หรือ Non root Node
② มีลูกกี่คนน้า~

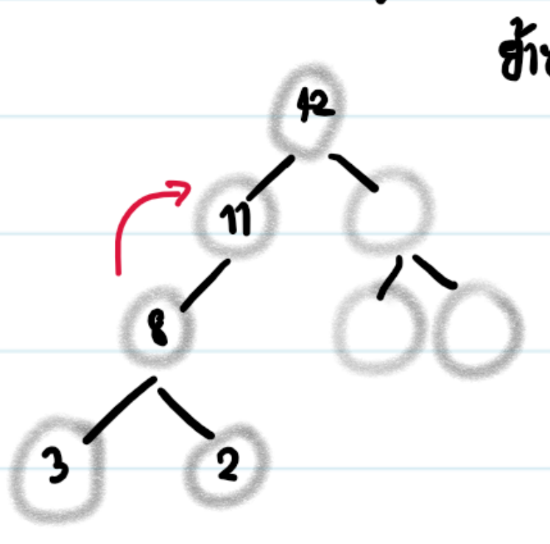
Case 1 (ไม่มีลูก)



```
Node P = node 4 . parent
P.left = null
node 4.parent = null (optional)
```

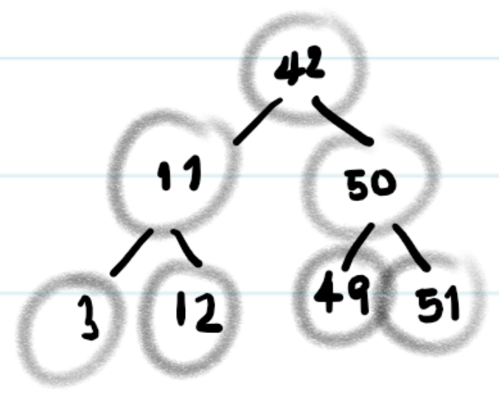
Case 2 (ลูก1)

ย้ายลูกมาแทนตำแหน่งที่ลบ



```
Node node 8 = t.find (8)
Node p      = node 8. parent
P.left      = node 8.left
node 8.left.parent = p
```

Case 3 (ลูก2)



ลบ root
เอา Min ของ right sub tree มาแทน root
รู้ได้ไงว่าถูก → left subtree < root < right sub tree

```
Node n = t. find (42)
Node min = find Min (n.right.key)
    min = n.left.parent
```

```
public void deleteRoot()
    if(root == null) print "ลบบ่ได้!!" (ไม่มีอะไรให้ลบ)
    else if (root.left == null && root.right == null) (มีแค่ root)
        root = null
    else if (root.left != null && root.right == null) (มี node ซ้าย (ขวาว่าง))
        root = root.left
        root.parent = null
    else if (root.left == null && root.right != null) (มีnode ขวา (ซ้ายว่าง))
        root = root.right
        root.parent = null
                                                      (เป็น Binary tree) งงอะดิ ดูนี้
    else  Node n = findMin(root.right)               - หาตัวที่มีค่าน้อยสุดใน right sub tree
          Node nn = new Node(n.key)                  - เอาไปใส่ที่ node ใหม่
          nn.left = root.left                        - เอา left sub tree มาเติม
          nn.left.parent = nn
          nn.right = root.right                      - เอา right sub tree มาเติม
          nn.right.parent = nn
          root = nn                                  - อันนี้จะเป็น root node ใหม่ได้เลย
          delete (n)                                 - ลบข้างเนี้ย
```