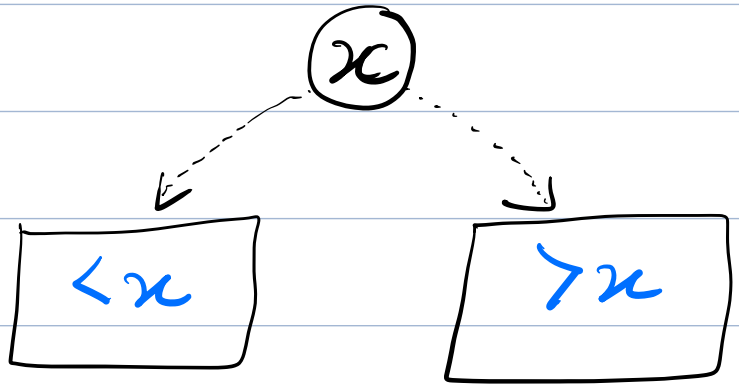
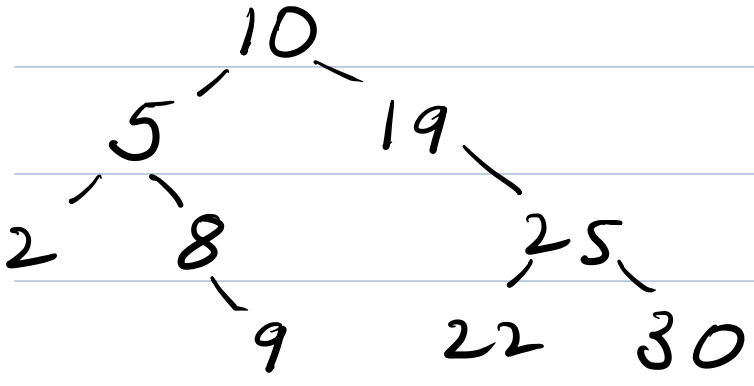
Binary Search Trees

Binary trees

all nodes in LST < node.data < all nodes in RST





This is BST

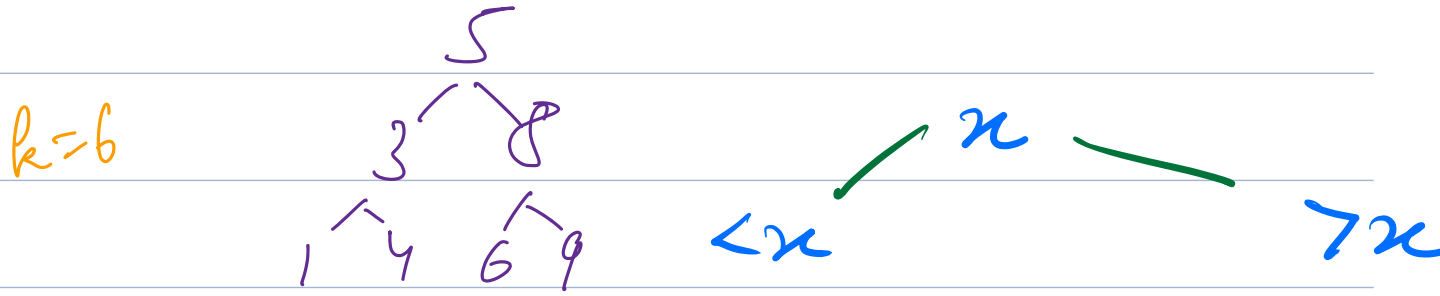
inorder

2 5 8 9 10 19 22 25 30

inorder of BST → sorted

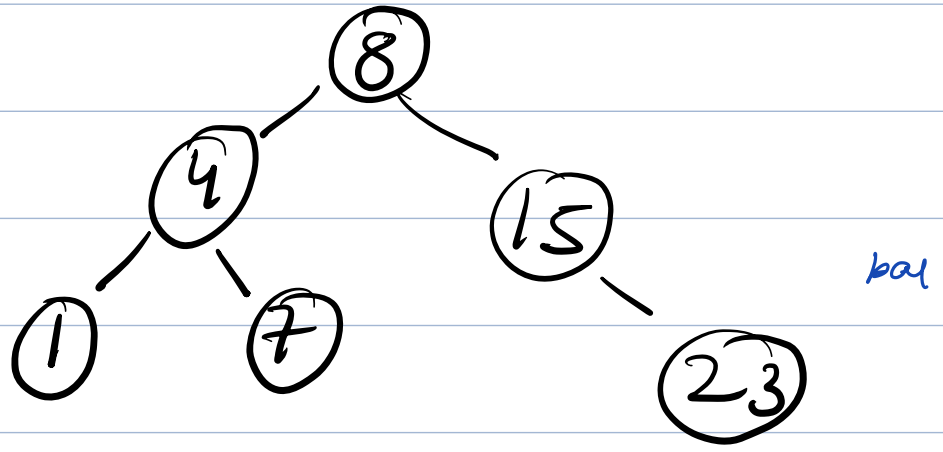
LST < root < RST

- Search elem K in BST




```
bool search (root, k) {
    1) root == null        false
    2) root.val == k       true
    3) if k > root.val     search(root.right, k)
    4) if k < root.val     search(root.left, k)
```

- insert node in a tree



Code

```
Node cur = root , par = null
while ( cur != NULL) {
    par = cur
    if ( cur.data < k )
        cur = cur. right
    else
        cur = cur. left
}
if ( k < par.data)
    par. left = new Node (k)
else
    par. right = new Node (k)
```

Smallest in BST

Go left till null

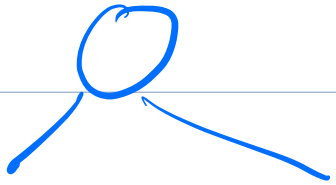
```
cur = root
while (cur.left != null)
    cur = cur. left
return cur . data
```

Largest in BST

Go right till null

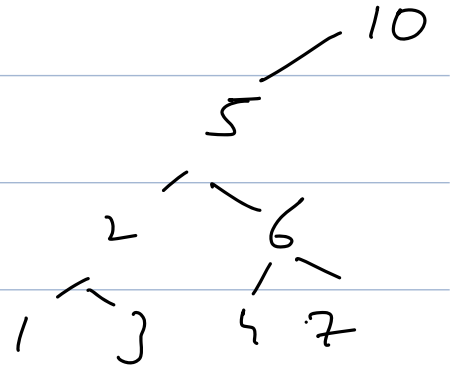
```
cur = root
while (cur. right != null)
    cur = cur. right
return cur . data
```

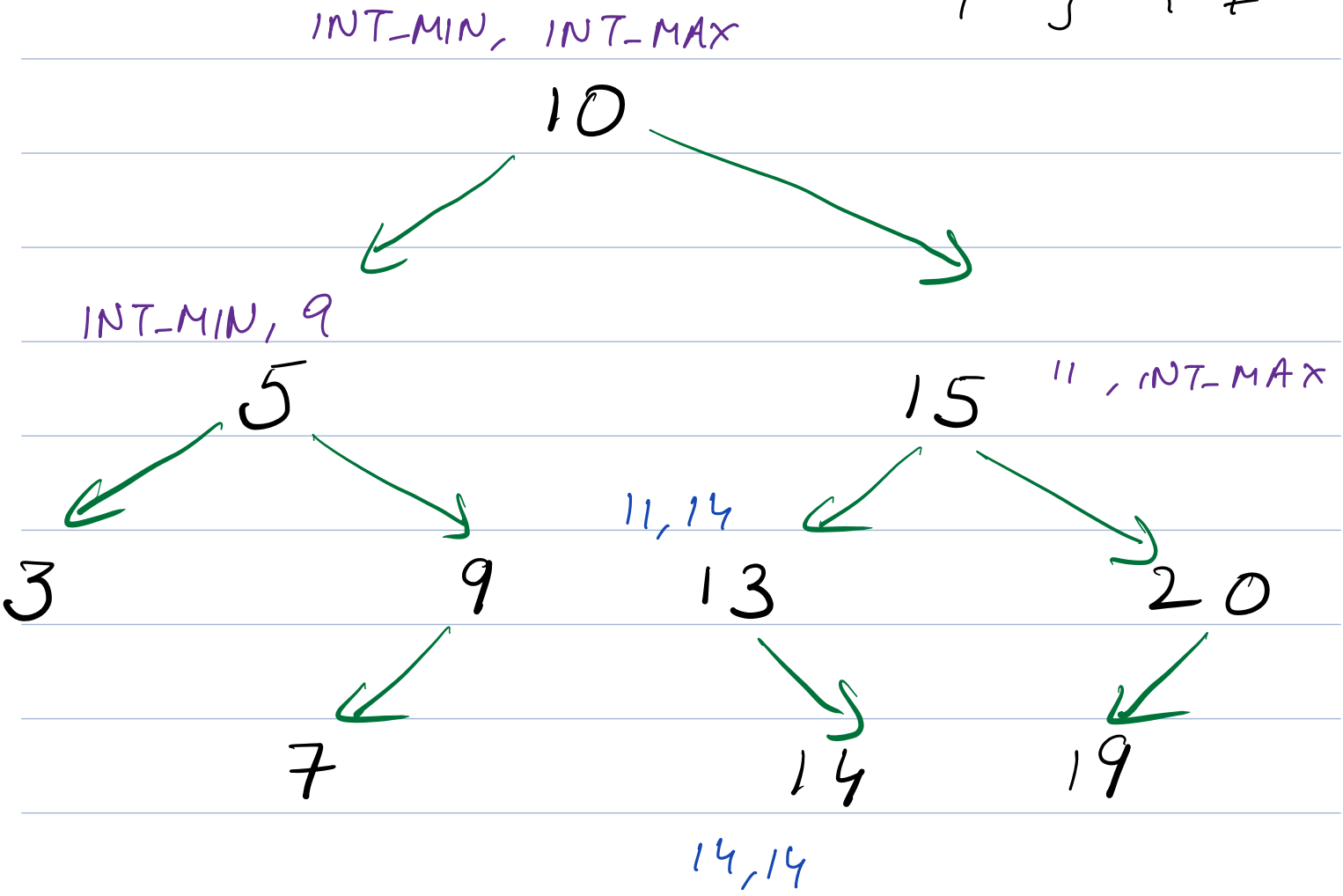
Check if given tree is BST ?

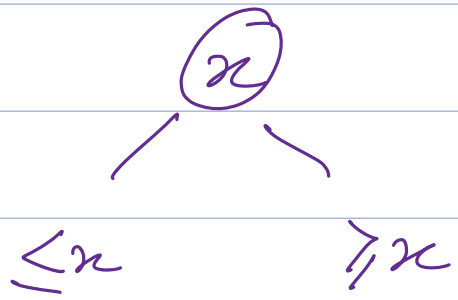


everything smaller

everything bigger



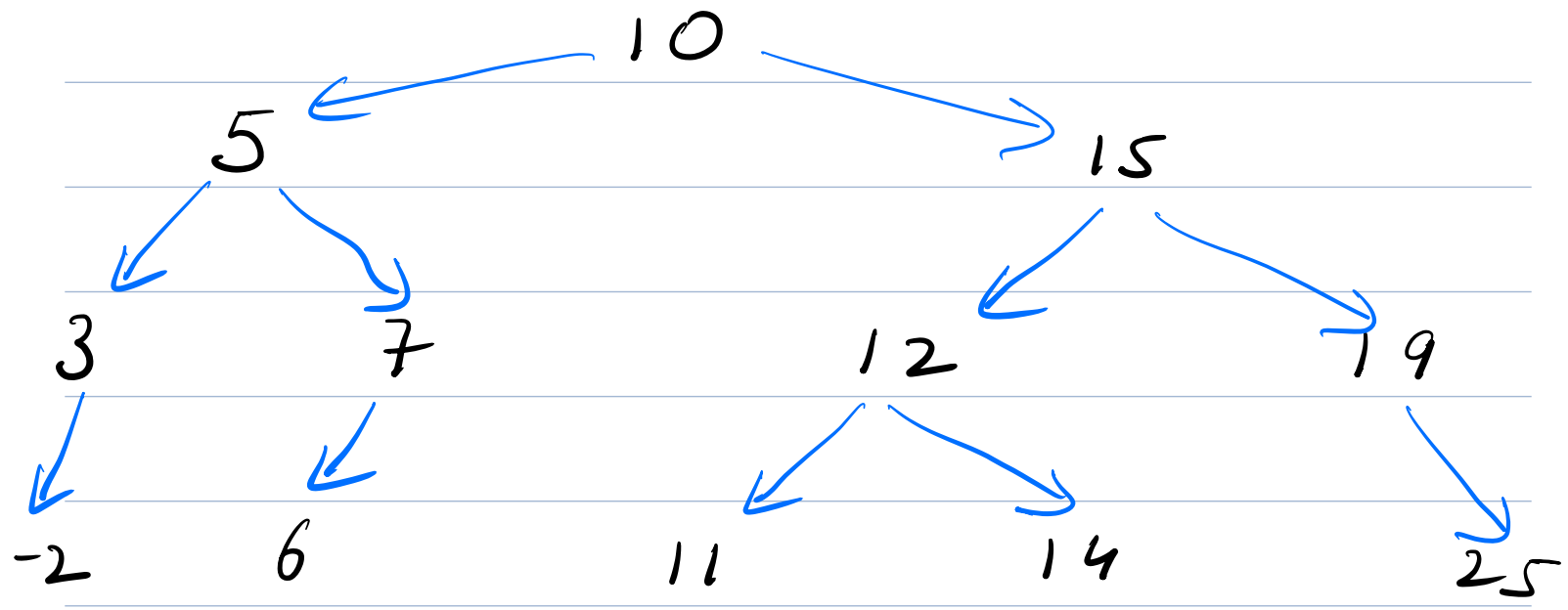



x
≤x
≥x

Code

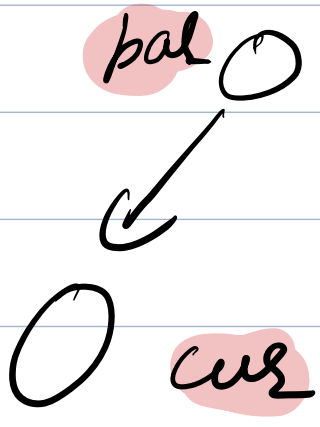
```
bool isBST(Node root, int l, int r){
    if (root == null)
        return true
    if ( root.data >= l  &&
        root.data <= r ) {
        bool x = isBST( root.left , l ,
                        root.data -1 )

        bool y = isBST( root.right ,
                        root.data + 1 , r )
        return  x & y
    }
    return false
}

isBST( root, INT_MIN, INT_MAX )
```

Delete node in BST

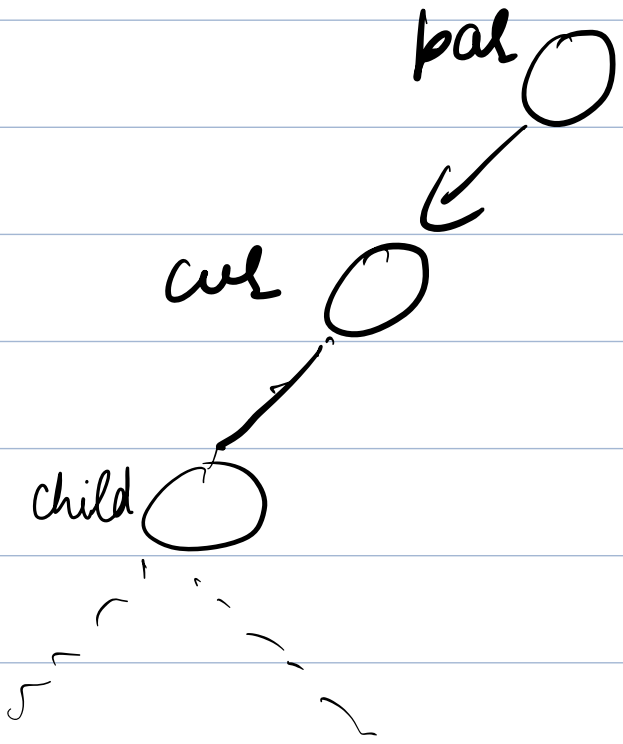


Case I when node is leaf

```
if (par.left == curr)
    par.left = null
else
    par.right = null
```

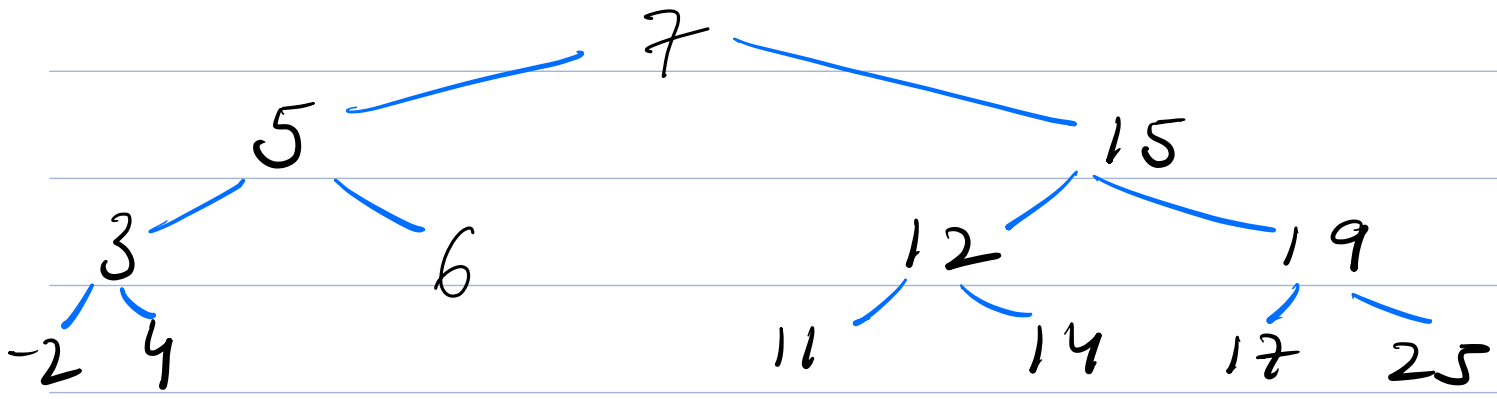


Case II when node has only one child




```
deleteUtil ( Node cur, Node par ) {
    Node child = cur.left
    if ( cur.right != null )
        child = cur.right

    if ( par.left == curr )
        par.left = child
    else
        par.right = child
}
```

Case 3 when two children



min of BST

keep going left

```
temp = root
while(temp.left != null)
    temp = temp.left
```

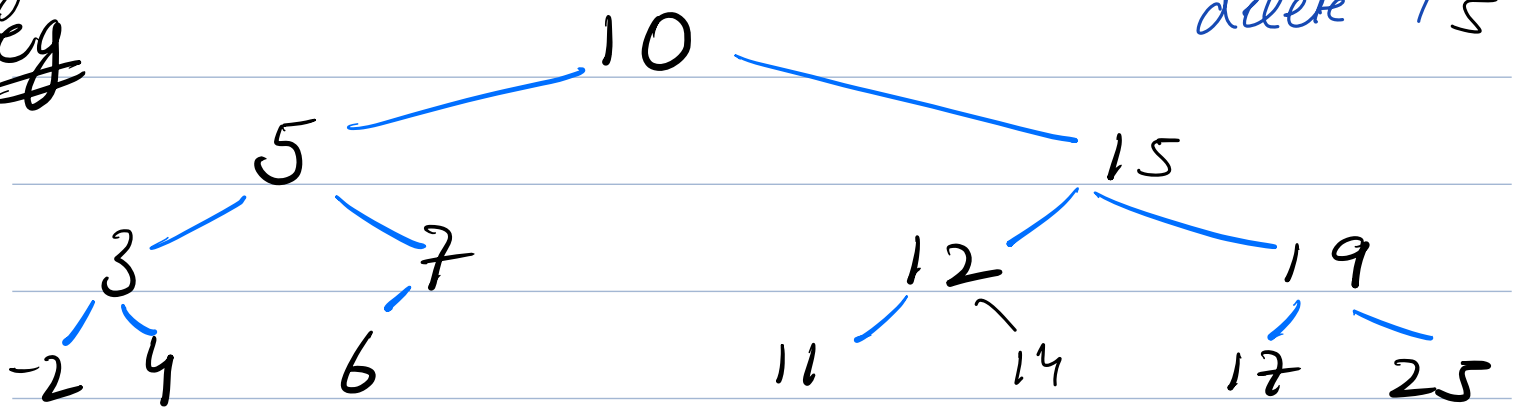
max of BST

keep going right

```
prev = cur
temp = cur.left
while (temp.right != null){
    prev = temp
    temp = temp.right
}
deleteUtil (temp, prev)
temp.left = cur.left
temp.right = cur.right
if ( par.left == curr)
    par.left = temp
else
    par.right = temp
```

Eg

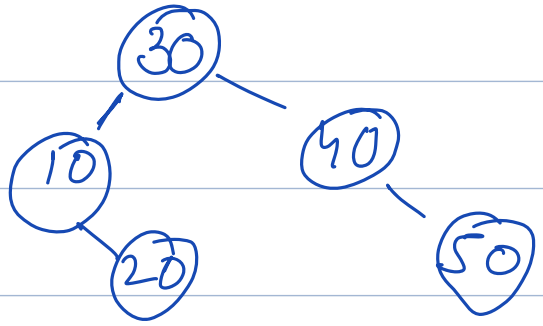
delete 15



Balanced tree → height == log n

Q Construct BBST from a sorted array

	0	1	2	3	4
Eg ⇒	10	20	30	40	50



Obs: What is root ?

middle guy of array

Idea: Root ⇒ middle elem

Left part ⇒ Left subtree

Right part ⇒ Right subtree

Recursion

l . . . . mid-1 mid mid+1 . . . . . r

Code

```
TreeNode construct (int arr[], int l, int r) {
    if (l == r) {
        return new TreeNode(arr[l])
    }
    mid = (l+r)/2
    TreeNode root = new Node(arr[mid])
    root.left = construct(arr, l, mid-1)
    root.right = construct(arr, mid+1, r)
    return root
}
```

construct(arr, 0, n-1)