TREES 3 : BST

Good Evening

Today's content

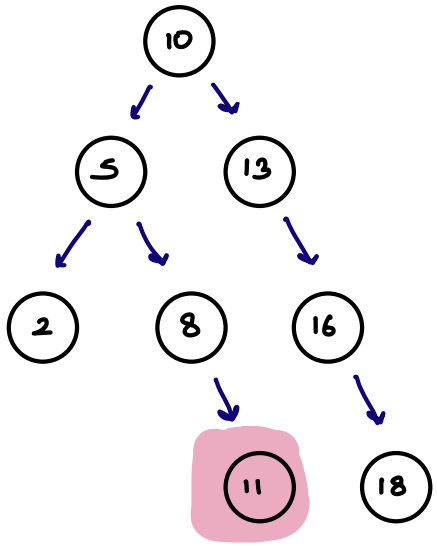
- → BST Basics + Searching
- → Insertion
- → Check if BST()
- → Deletion in BST
- → Construct BST from sorted arr

A binary tree is BST

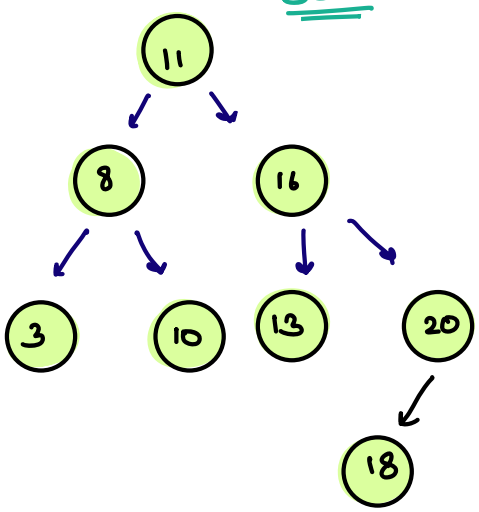
All elements < node < All elements
in LST in RST

for all nodes

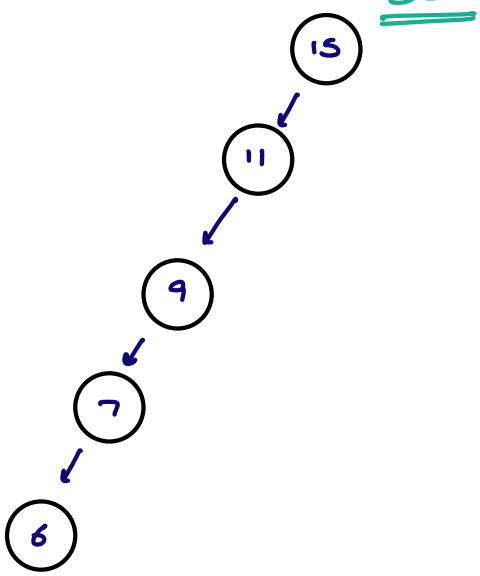
not a BST



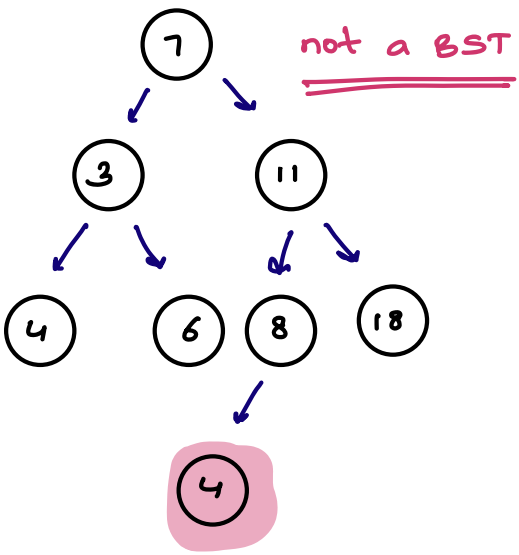
BST



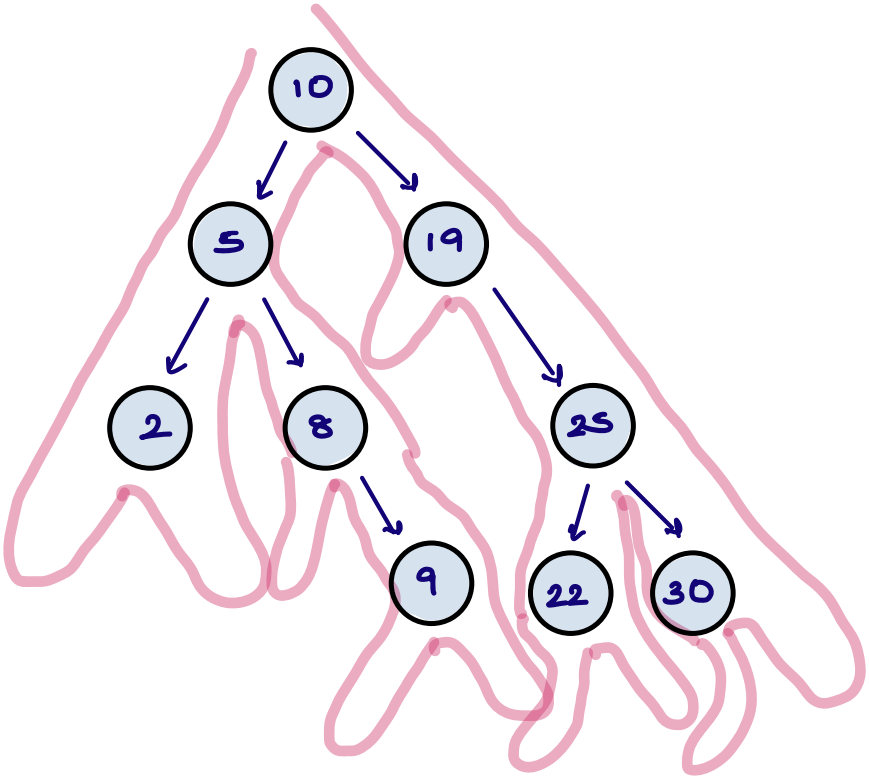
BST



not a BST



BST special property

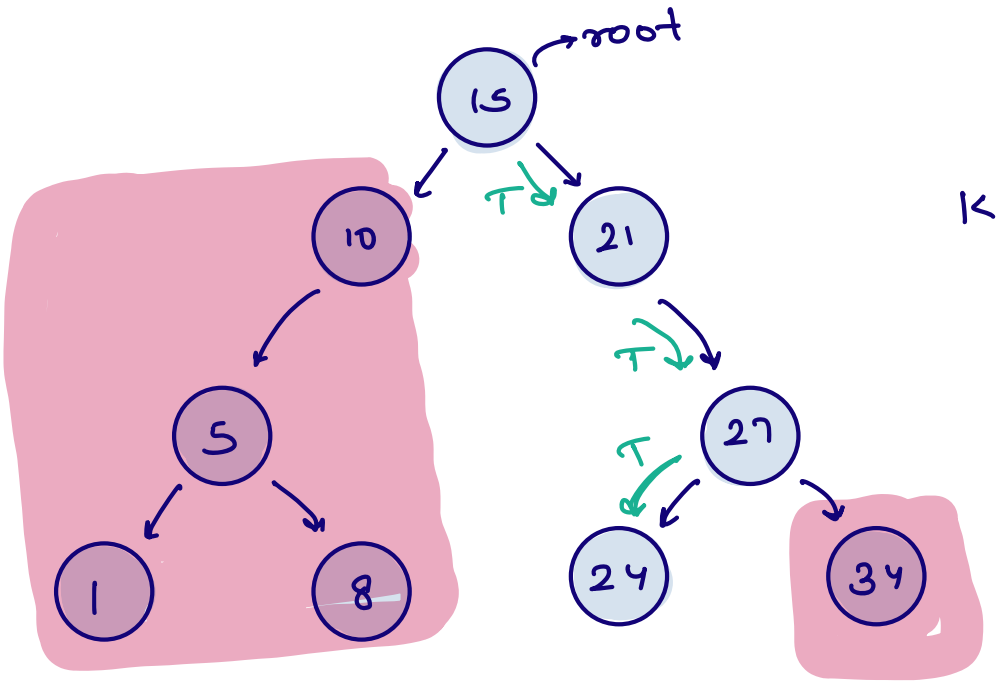


Inorder 2 5 8 9 10 19 22 25 30

sorted

Increasing order

Q Search in BST

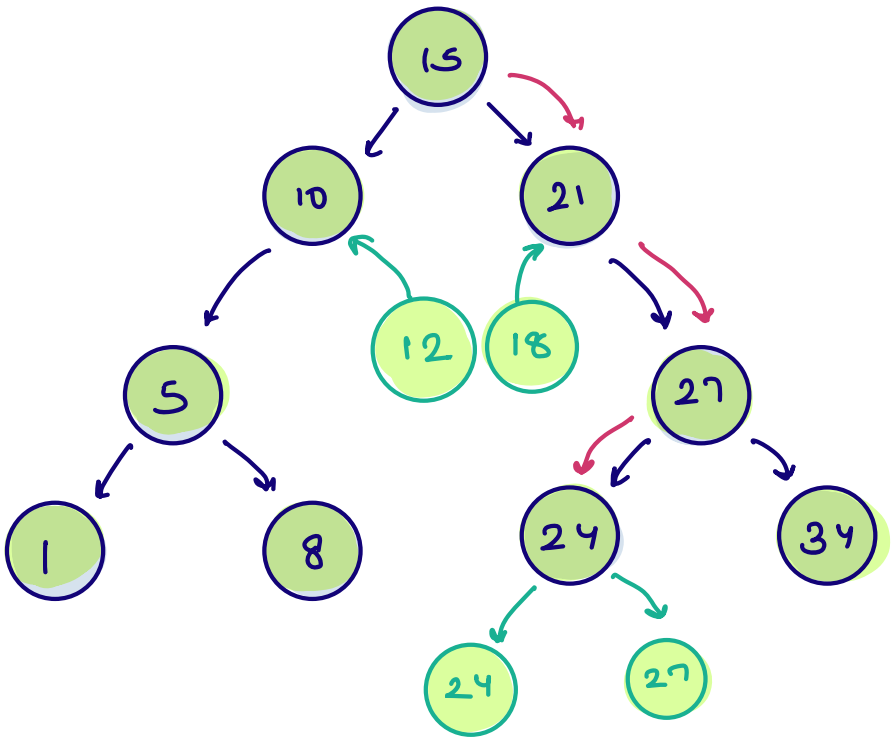


K = 24

```
boolean search (Node root, K)
    if (root == null) return false;
    if (root.val == K) return true
    else if (root.val > K)
    return search (root.left, K)
    else
    return search (root.right, K)
}
```

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

Q **Insert in BST**



K=12

K=24

K=18 ✓

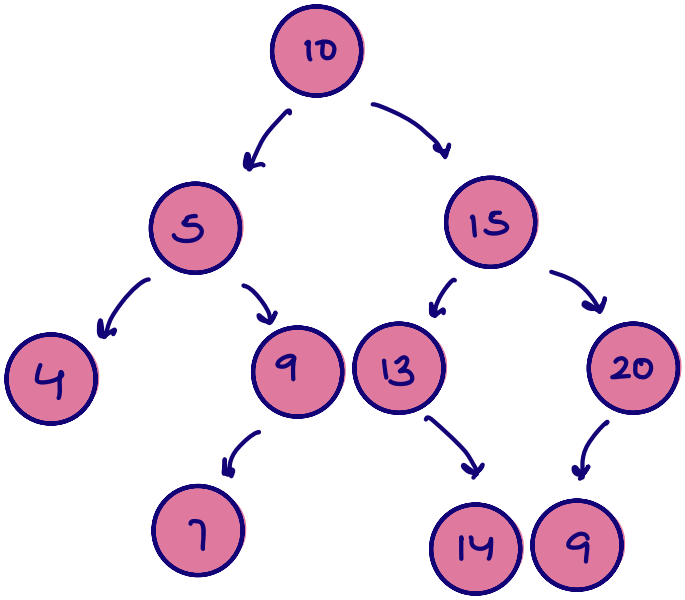
K=27

```
Node insert ( root, k )
{
    if (root == null) {
        Node nn = new Node(k)
        return nn;
    }
    if (root.data > k)
        root.left = insert ( root.left , k)
    else {
        root.right = insert (root.right, k)
    }
    return root;
}
```

TC: O(h)

SC: O(h)

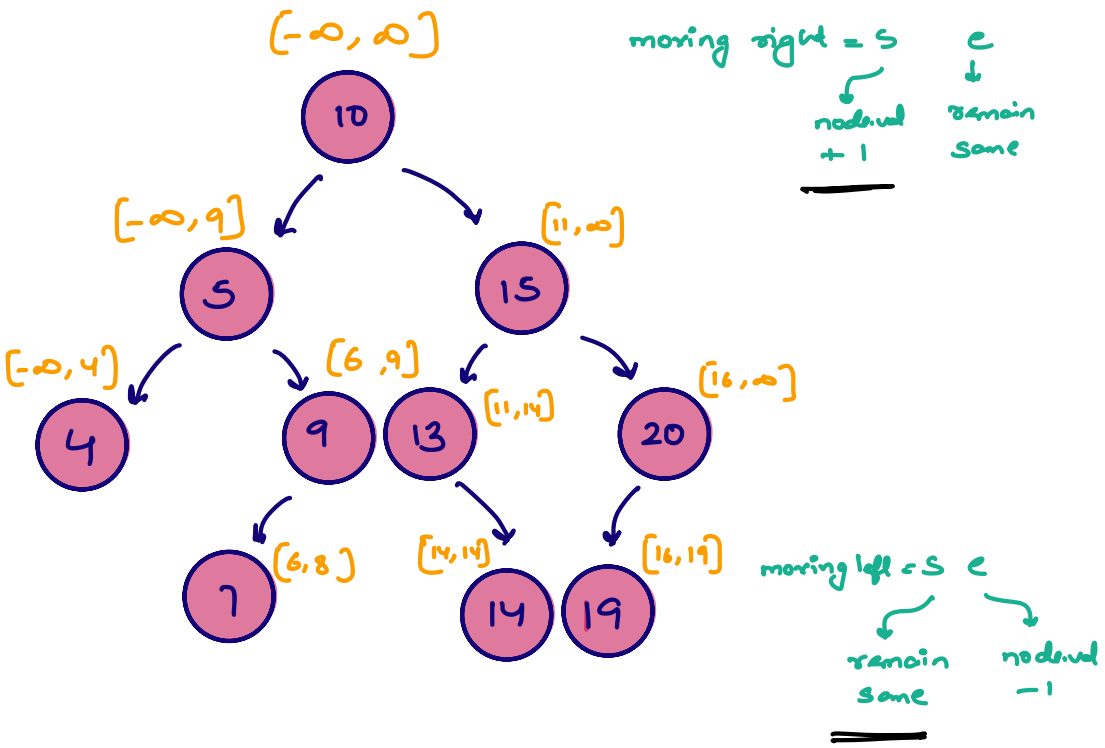
Q Given BT, check if BST or not.



Idea 1 → Check if inorder is sorted or not.

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

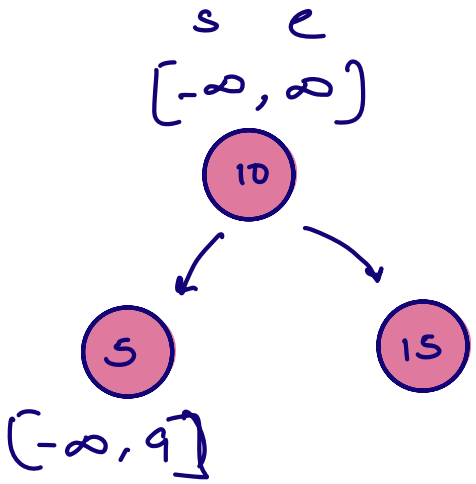
Idea 2



$s = -\infty$

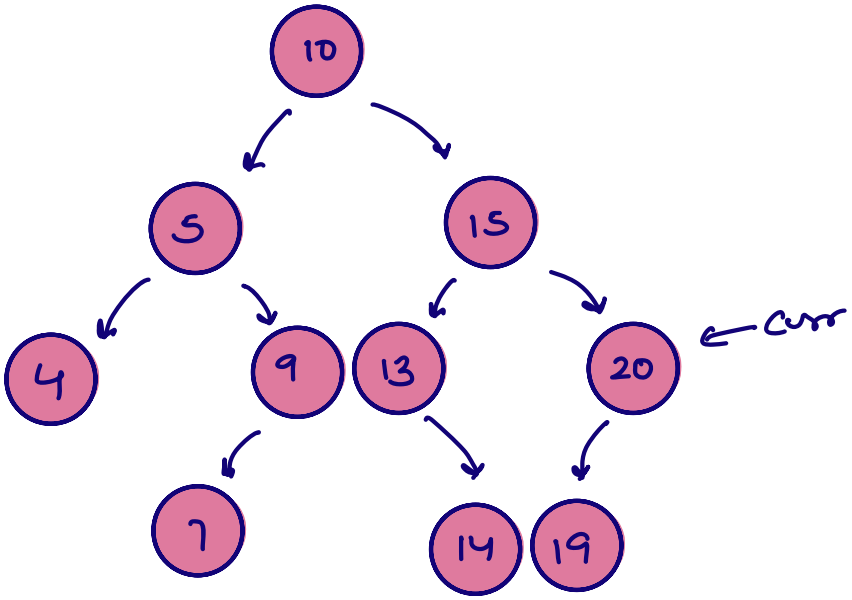
$e = \infty$

```
boolean isBST ( root, int s, int e )
{
    if (root == null) return true;

    if (s ≤ root.data && root.data ≤ e)
    {
        bool left = isBST ( root.left, s, root.data - 1)

        if (left == false) return false

        boolean right = isBST (root.right, root.data + 1, e)

        if (right == false) return false;

        return true;
    }
    else {
        return false;
    }
}
```



10:05 → 10:15 pm

9 Max in BST



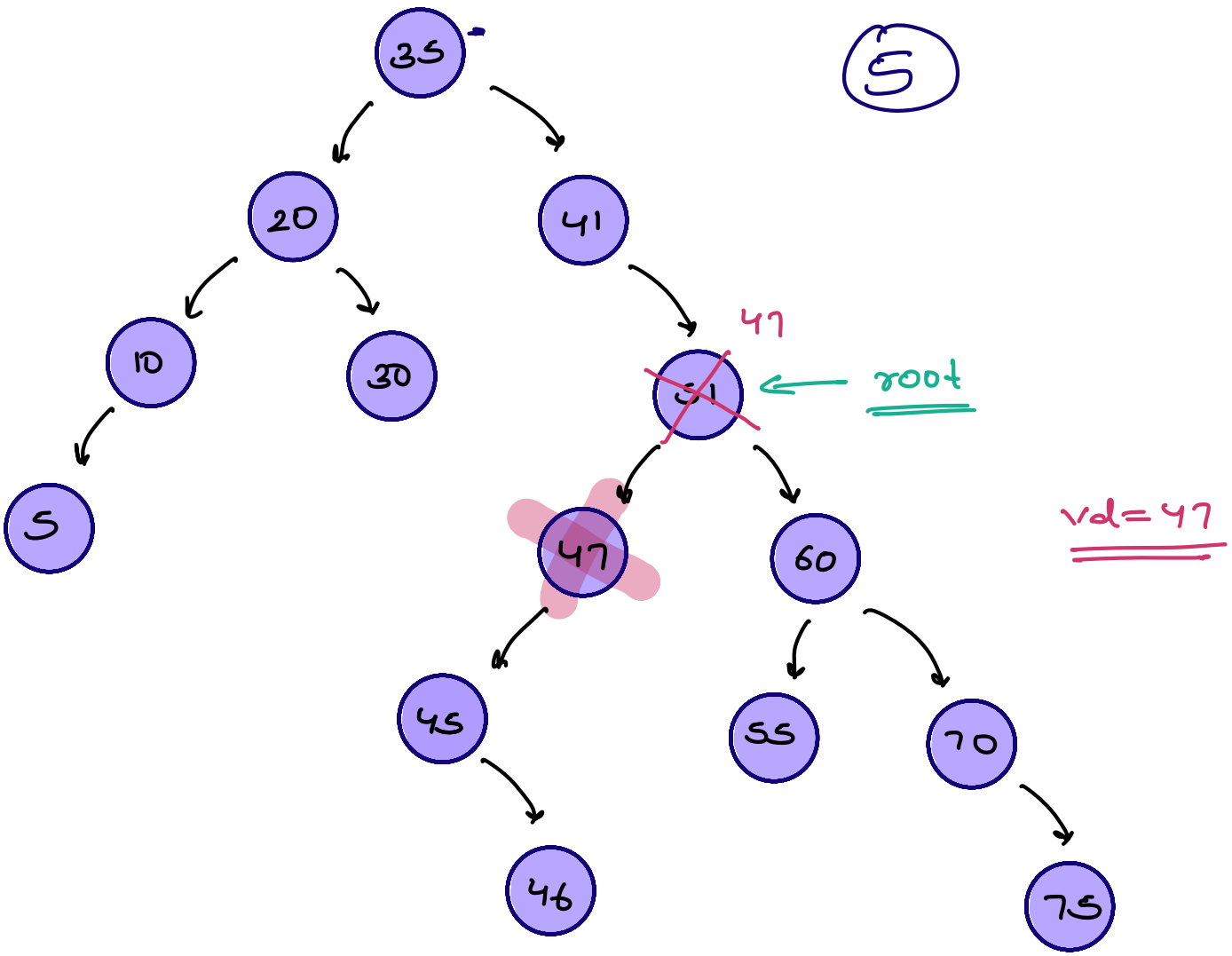
TC = O(h)

SC = O(1)

```
while (curr.right != null)
    curr = curr.right;
```

* find min in BST {TODO}

Q Delete a node in BST



K = 5
K = 46
} Delete a node having 0 child

K = 10
K = 41
} Delete a node having 1 child

K = 51
K = 20
} Delete a node having 2 child

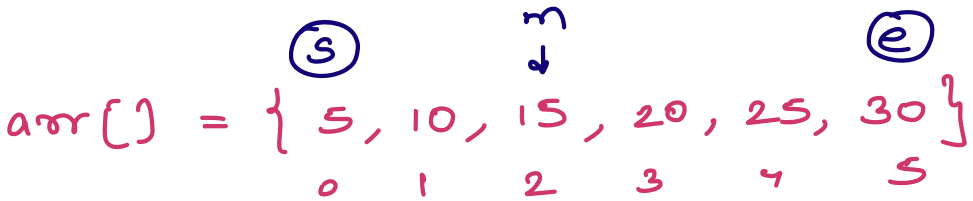
```
Node deleteBST ( root, k)
    if (root.data > k)
        root.left = deleteBST (root.left, k)
    } else if (root.data < k)
        root.right = deleteBST (root.right , k)
    } else {
        if (root.left == null && root.right == null)
            // k is leaf node
            return null;
        }
        else if (root.left != null && root.right == null )
            return root.left
                                } // single child on left
        }
        else if (root.right != null && root.left == null )
            return root.right
                                } // single child on right
        }
        else {
            int val = max (root.left)
            root.data = val;
            root.left = deleteBST (root.left, val)
        }
        return root;
    }
```

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

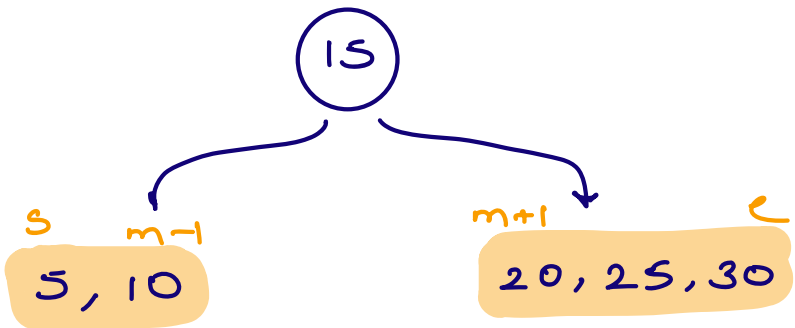
}

The height difference of a node can't be greater than 1

Q Construct a balanced Binary Search Tree using a sorted array

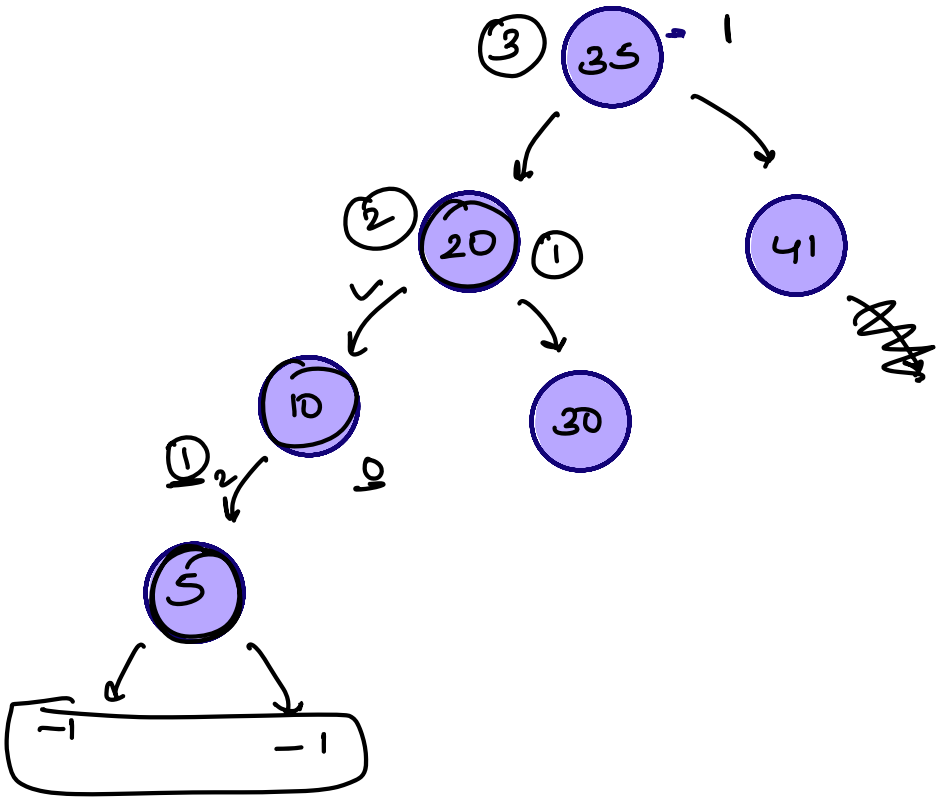


$$mid = \frac{s+e}{2}$$



Code → {TODO}

Doubts



$3-1 = 2 > 1$