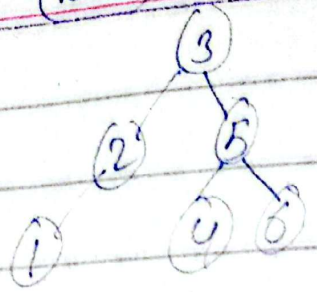
Binary Search Tree (BST)

→ Special Binary Tree.
left > par < right



Properties :-

→ Search in BinaryTree (O(n))

→ As the sorting already done, search space is halved ⇒ O(height of tree)
↓
O(log n)

→ Inorder seavence will also be sorted
(1,2,3,4,5,6) → Sated seavence

→ Build a BST
arr : [3,2,1,5,6,4]

Same Node class as in Binary Tree.

```
Node* Insert(root, val){
    If(root==NULL){
        return newNode(val)
    If(val <root->data){
    root->left = insert(root->left, val)
    else
    root->right = Insert(root->right, val)
        return root;
```

Printing Inorder

```
if(root=NULL){
return }
Inorder(root->left)
```

```
cout<<root->data
Inorder(root->right)
```

Search in BST

```
Bool search(root, key){
if(root==NULL){
    return false
if(root->data==key){
    retur false
if(root->d >key)
return search(root->left,key)
else
return search
    (root->right, key)
```

Delete Node in BST

① No child
② One child
③ 2 children

② One child
↓
replace the parent with child

③ 2 children
↓
① Find Inorder successor → left most node in right subtree
② root = IS->data
③ delete the IS

```
Node* NewNode(root,key){
if(root=NULL){
    return NULL
If(key<root->data)
root->left = del Node(
                root->left,key)
else if (key>root->data)
root->right = delNode(
        r->right,key)
else delete : root = key
```

① 0 children
delete root
return root

② 1 child
delete root
return non null child

③ 2 children

```
Node* IS = get IS
        (root->right)
root->data = IS->data
root->right =
del Node(root->right,
        IS->data)
```