# BINARY SEARCH TREES (BST)

T.C : $O(\log n)$ ⇒ Balanced Trees
$O(h)$ ⇒ Worst case

Properties:- (1) All items in left subtree are less than root.
(2) All items in right subtree ≥ root.
(3) Each subtree is itself a BST.

BST provide an excellent structure for searching a list and at the same time for inserting and deleting data into the list.

```
typedef struct treenode{
    int data;
    struct treenode *right;
    struct treenode *left;
}Node1;
```

⇒ Search in a BST

```
void *search(Node1 *root, int key){
  if (root == NULL){
      printf("key not present");}
  else if (root->data== key){
      printf("Key present");}
  else if (root->data> key){
      search(root->left, key);}
  else {
      search(root->right, key);}
}
```

## ⇒ Insertion into a BST

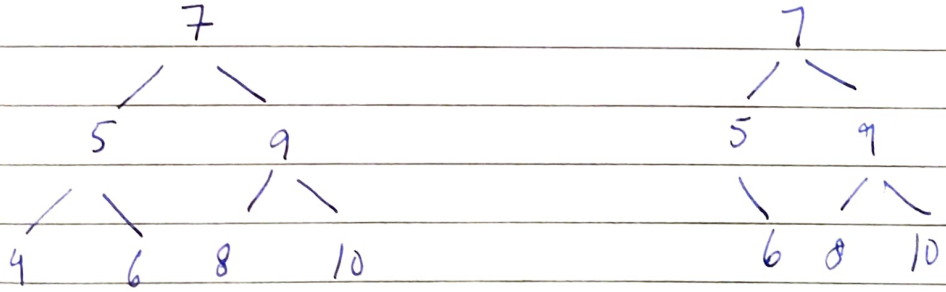
```
Node1 * insert (Node1 * root, int item) {

    if (root == NULL) {
      Node1 * temp = (Node1*) malloc( sizeof (Node1));
      temp -> data = item;
      temp -> left = temp -> right = NULL;
      return temp;
    }
    else if (item < root -> data) {
        root -> left = insert (root -> left, item);
    else if (item > root -> data) {
        root -> right = insert (root -> right, item);
    return root;
}
```

## ⇒ Removal in BST

Case I: Removing a node with 2 EMPTY SUBTREES

Eg.

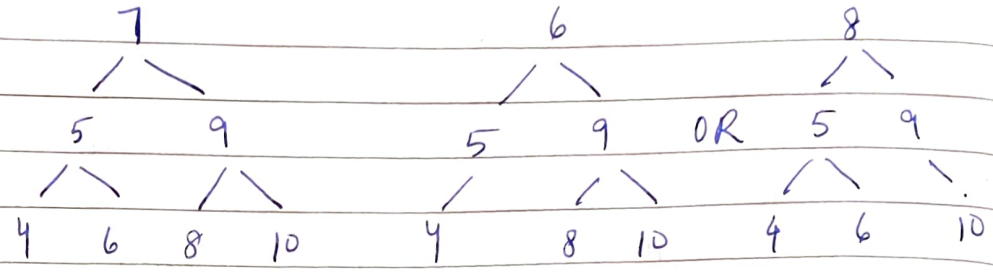



Removing 4 ⇒ Replace the link in parent with NULL

Case 2: Removing a node with 2 subtrees

- Replace the node's value with max value in left subtree or with min. value in right subtree.
- Delete that node

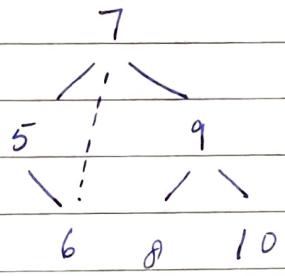
eg.




Removing 7

Case 3: Removing a node with NO left child

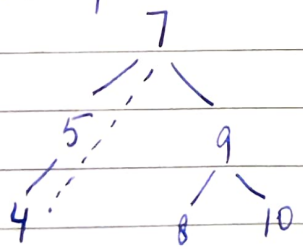
- Link parent node to right subtree




Removing 5

Case 4: Removing a node with NO right child

- Link parent node to left subtree




Removing 5

The complexity of operations get (search), insert & remove in BST is $O(h)$, where $h$ is the height.

$O(\log n)$ when tree is balanced. Operations cause tree to become unbalanced.

** A tree is said to be balanced if the difference in height of left & right subtree is -1, 0 or 1.

Balance Factor = $H_L - H_R$

If a BST containing $n$ nodes is balanced, insertion, deletion & retrieval will all take $O(\log n)$ time.