Saturday, 02 April 2022 03:06 AM

Binary Search Tree

→ atmost 2 children for each node.

left subtree < root

right subtree ≥ root

TC → height

BST →

Traversal

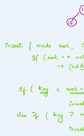
```
inorder (root)
if (root != null) return ;
Inorder (root.left);
cout (root.data);
Inorder (root.right);
```

&

operations

1) Traversal / Searching

2) Insertion



```
Insert (Node root, int key) {
  if (root == null)
    → (add a new node)

  if ( key < root.data) {
    Insert (root.left, key);
  else if ( key > root.data) {
    Insert (root.right, key);
```

→ left < root right

3) Deletion of node

1) node to be deleted is a leaf node

del (13)

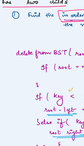
5 7 9 10 11 12 14

2) node to be deleted has only one child

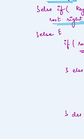
3) has two childs

1) find the in order successor of the node.

```
deleteNodeBST(root, key) {
  if (root == null) {
    return null;
  }
  if ( key < root.data) {
    root.left = deleteNodeBST(root.left, key);
  else if ( key > root.data) {
    root.right = deleteNodeBST(root.right, key);
  else {
    if ( root.left == null && root.right == null) {
      delete root;
      return null;
    } else if ( root.right == null) {
      Node temp = root.left;
      delete root;
      return temp;
    } else if ( root.left == null) {
      Node temp = root.right;
      delete root;
      return temp;
    } else {
      Node right_root = root.right;
      Node tm_in_temp = right_root;
      while ( tm_in_temp.left != null) {
        tm_in_temp = tm_in_temp.left;
      root.data = tm_in_temp.data;
      root.right = deleteNodeBST(right_root, root.data);
```



constructed tree based on a traversal technique




```
constructT (pre, in, pre, pre, in, in) {
  Node n = new Node();
  n.data = pre[pre];
  int x = -1;
  for( int i= in; i<n; i++) {
    if (pre[pre] == in[i]) {
      x = i;
      break;
  }
  int[] left_in = new int[x];
  for (i=0; i< x; i++) {
    left_in[i] = in[i];
  int[] left_pre = new int[x];
  for (i=1; i< x+1; i++) {
    left_pre[i-1] = pre[i];
  n.left = constructT( pre, in, pre+1, pre, left_in, left_pre);
  int[] right_in = new int[n-x-1];
  for (i=x+1; i< n; i++) {
    right_in[i-x-1] = in[i];
  int[] right_pre = new int[n-x-1];
  for (i=x+1; i< n; i++) {
    right_pre[i-x-1] = pre[i];
  n.right = constructT( pre, in, pre+x+1, pre, right_in, right_pre);
  return n;
}
```