Sunday, 12 October 2025 8:33 PM

Binary Search Tree

→ Normal & children for each node.

→

→ L&T & R&T are also B.S.T.

left subtree < root

right subtree > root.

find (8)
↳ false

BT → find → O(n)

BST → find → O(height)

Traversal

Pre : 10 3 1 7 12 11

Inv : 1 3 7 10 11 12 (sorted)

Post : 1 7 3 11 12 10

Operations

① Traversal (same as BT)

② Insertion

→ add (19)
→ add (8)
→ add (5)

③ Deletion of a node.

→ node to be deleted is a leaf node.

del (8)

→ Node to be deleted has only one child.

→ copy the child to the node and delete the child.

del (6)

→ Node to be deleted has 2 child

① find the inorder successor of the node.

② copy the contents of inorder successor (inorder predecessor) and del them.

del (5)

Tree Representation

	unsorted BT	unsorted BST
Pre + Inv	✓	✓
Post + Inv	✓	✓
Pre + Post	X	✓

(BT) Pre : root

(BT) Post : D E B F C A → root

(BST) Pre : root, left, right

(BST) Post : 10 5 15 25 30 40 20 → root

Q. Given Inorder, unsorted BST ??

[1 2 3 4 5 6 7 8]

Q. Given a BT, check if it is a BST.

① true

② false

isBST (node, ~~int~~ long* min, ~~int~~ long* max)

if (node == null)
return true ;

if (node . data < min || node . data > max)
return false ;

return isBST (node . left, min, node . data - 1)
&& isBST (node . right , node . data + 1, max) ;

}

Q. Given the preorder traversal of a BST, find the leaf nodes of BST.

pre → [890 , 325 , 290 , 530 , 965]

output → [290 , 530 , 965]

idx

pre [890 , 325 , 290 , 530 , 965]

i = 0;

boolean isLeaf (pre, ~~int~~ min, ~~int~~ max) {

if (i >= n) return false ;

if (pre [i] > min && pre [i] < max)
{ i++ ;

bool left = isLeaf (pre , n, min, pre [i-1]) ;

bool right = isLeaf (pre , n, pre[i-1], max) ;

if (! left && ! right)
print (pre [i - 1]) ;

}

return true ;

}

Q. Given the root of a BST, convert it into a greater tree such that every key of the original BST is changed to the key plus sum of all keys greater than original key.

Eg

int pre = 0 ;

convert (root) {

if (root. right != null)
convert (root. right);

pre = root. val + pre ;
root. val = pre ;
if (root. left != null ;
convert (root. left);

}

return root ;

}

Q. Given a BST, randomly swapped 2 elements, find out the root node was swapped ??