**Tree FAQ 01**

# Program to check if a binary tree is BST or not

A Binary search tree (BST) is a node based binary tree data structure which has the following properties.

* The left subtree of a node contains only nodes with keys less than the node’s key.
* The right subtree of a node contains only nodes with keys greater than the node’s key.
* Both the left and right subtrees must also be binary search trees.

From the above properties it naturally follows that:

* Each node (item in the tree) has a distinct key.

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

struct tNode {

int data;

struct tNode \* left;

struct tNode \* right;

};

int isBST(struct tNode \* arg\_root);

int isBSTUtility(struct tNode \* arg\_node, int arg\_min, int arg\_max);

struct tNode \* newNode(int arg\_val);

int main(void) {

struct tNode \*root = newNode(3);

root -> left = newNode(2);

root -> right = newNode(5);

root -> left->left = newNode(1);

root -> left->right = newNode(4);

if(isBST(root)) {

printf("Tree is BST");

}

else {

printf("Tree is not BST");

}

return 0;

}

struct tNode \* newNode(int arg\_val) {

struct tNode \* node = (struct tNode \*)malloc(sizeof(struct tNode));

node -> data = arg\_val;

node -> left = NULL;

node -> right = NULL;

return node;

}

int isBST(struct tNode \* arg\_root) {

return isBSTUtility(arg\_root, INT\_MIN, INT\_MAX);

}

int isBSTUtility(struct tNode \* arg\_node, int arg\_min, int arg\_max) {

if(NULL == arg\_node)

return 1;

if( ((arg\_node -> data) < arg\_min) || ((arg\_node -> data) > arg\_max) ) {

return 0;

}

return (isBSTUtility(arg\_node->left, arg\_min, (arg\_node->data) - 1) && isBSTUtility(arg\_node->right, (arg\_node->data) + 1, arg\_max));

}

Output:

Tree is not BST

# END