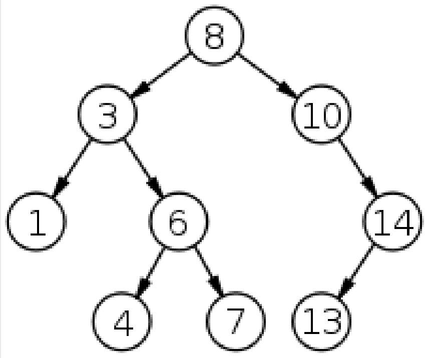
Binary Search Tree | Set 1 (Search and Insertion)

The following is definition of Binary Search Tree(BST) according to Wikipedia

Binary Search Tree, 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.
- The left and right subtree each must also be a binary search tree.
  There must be no duplicate nodes.



The above properties of Binary Search Tree provide an ordering among keys so that the operations like search, minimum and maximum can be done fast. If there is no ordering, then we may have to compare every key to search a given key.

Searching a key
To search a given key in Bianry Search Tree, we first compare it with root, if the key is present at root, we return root. If key is greater than root's key, we recur for right subtree of root node. Otherwise we recur for left subtree.

```
// C function to search a given key in a given BST
struct node* search(struct node* root, int key)
{
    // Base Cases: root is null or key is present at root
    if (root == NULL || root->key == key)
       return root;

    // Key is greater than root's key
    if (root->key < key)
       return search(root->right, key);
    // Key is smaller than root's key
    return search(root->left, key);
}
```

Insertion of a key

A new key is always inserted at leaf. We start searching a key from root till we hit a leaf node. Once a leaf node is found, the new node is added as a child of the leaf node.

```
         100                               100
        /   \        Insert 40            /    \
      20     500    --------->          20     500
     /  \                              /  \
    10   30                           10   30
                                              \
                                              40
```

```
// C program to demonstrate insert operation in binary search tree
#include<stdio.h>
#include<stdlib.h>

struct node
{
    int key;
    struct node *left, *right;
};

// A utility function to create a new BST node
struct node *newNode(int item)
{
    struct node *temp =  (struct node *)malloc(sizeof(struct node));
    temp->key = item;
    temp->left = temp->right = NULL;
    return temp;
}

// A utility function to do inorder traversal of BST
void inorder(struct node *root)
{
    if (root != NULL)
    {
        inorder(root->left);
        printf("%d ", root->key);
        inorder(root->right);
    }
}

/* A utility function to insert a new node with given key in BST */
struct node* insert(struct node* node, int key)
{
    /* If the tree is empty, return a new node */
    if (node == NULL) return newNode(key);

    /* Otherwise, recur down the tree */
    if (key < node->key)
        node->left  = insert(node->left, key);
    else if (key > node->key)
        node->right = insert(node->right, key);

    /* return the (unchanged) node pointer */
    return node;
}

// Driver Program to test above functions
int main()
{
    /* Let us create following BST
              50
           /     \
          30      70
         /  \    /  \
       20   40  60   80 */
    struct node *root = NULL;
    root = insert(root, 50);
    insert(root, 30);
    insert(root, 20);
    insert(root, 40);
    insert(root, 70);
    insert(root, 60);
    insert(root, 80);

    // print inoder traversal of the BST
    inorder(root);

    return 0;
}
```

Output:

```
20 30 40 50 60 70 80
```

Time Complexity: The worst case time complexity of search and insert operations is O(h) where h is height of Binary Search Tree. In worst case, we may have to travel from root to the deepest leaf node. The height of a skewed tree may become n and the time complexity of search and insert operation may become O(n).

Binary Search Tree Delete Operation
Quiz on Binary Search Tree

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Category: Data Structures

Tweet 0 g+1 0

38 Comments GeeksQuiz 1 Login

Recommend 3 Share Sort by Best

Join the discussion…

Joanne • 6 months ago

Could you upload this lesson in Java please?

2 • Reply • Share ›

disqus_8vHOSNH3Ot Joanne • 6 months ago

```
class TNode{
int data;
TNode left;
TNode right;
public TNode(int item){
data = item;
left = null;
right = null;
}
}

public class searchNinsert {

public static TNode insert(TNode root, int item){

if(root == null)
return new TNode(item);

if(item < root.data)
```

see more

3 • Reply • Share ›

Holden disqus_8vHOSNH3Ot • 4 months ago

very useful, thank you so much :)

• Reply • Share ›

shashsriv93 • 9 months ago

in the search function what if we do not find the data in the tree ?

will the function not give an error ? (since the base conditions only check for cases when the tree is empty or if the data is found)

2 ^ | ⌄ • Reply • Share ›

Gaurav Chauhan ➦ shashsriv93 • a month ago

Yes, the function will return NULL indicating that either the tree was not found or the data was not found in the tree. See if tree is not found i.e. root = null in the first time only, function return null. Now, if tree is present, we recursively keeps on deciding whether to go to left subtree or to right subtree. If the data is not found, at the end when we have reached to a node at the last level of the tree then going to either subtree will result to NULL as its both subtrees will be null. Hence this result to returned back and null is returned by the function. See recursion more carefully.

^ | ⌄ • Reply • Share ›

akash kumar • 10 months ago

insert function return struct node type pointer....
but in main function when we call insert function ..there is no pointer present to catch the address returned by the insert function...
we call insert function in main as if it is void returned type function...
correct me whether i m right or wrong???

2 ^ | ⌄ • Reply • Share ›

thevagabond85 ➦ akash kumar • 5 months ago

here's what you might be looking for :

```
typedef struct node* Node;
void insert(Node &root, int key)

{

if(root == NULL)
root = newNode(key);
else{
if(root->data > key)
insert(root->left, key);
else
insert(root->right, key);
}

}
```

^ | ⌄ • Reply • Share ›

Aditya Goel ➦ akash kumar • 9 months ago

We are only catching the address of root node in main().ie. when we call insert() first time.

struct node *root = NULL;
root = insert(root, 50);

After setting address of root node to local pointer we are passing that pointer to insert function. Apart from root node where the control is returned to the main() in first line of insert(), when we call insert() on other keys, the address of new nodes created are returned and set in insert() function itself.
ie.
node->left = insert(node->left, key);
node->right = insert(node->right, key);

Hope I made myself clear.

• Reply • Share ›

pratik Aditya Goel • 5 months ago

plzzz help ....when i m trying to set the address of the first call inside function insert() itself instead of setting it in main function its showing error the pointer is staying null........... heres my code

struct node* insert(struct node *root,int data)

{

if(!root)

{

struct node *tmp=create(data);

root=tmp;

return root;

}

see more

• Reply • Share ›

Aditya Goel pratik • 5 months ago

Call insert() as root=insert(root,20); for the first time. You were passing the root as call by value. So it was not updating.

Also inside insert() function, call left and right child as -

root->left = insert(root->left,data);
root->right = insert(root->right,data);

```
struct node* insert(struct node *root,int data)
{
        if(!root)
        {
                struct node *tmp=create(data);
                root=tmp;
                return root;
        }
        if(data<root->data)
                root->left = insert(root->left,data);
        else
```

see more

• Reply • Share ›

pratik Aditya Goel • 5 months ago

if i m passing a pointer to a function why is it a case of call by value ????????
and how do we pass a pointer by call by refernce????
i do not understand why do we need to equate the pointer the first time i.e. root=insert(.....);

• Reply • Share ›

guesT akash kumar • 9 months ago

i think you are right..
the code is written properly in set2..
here..
http://geeksquiz.com/binary-se...

• Reply • Share ›

Aditya Goel • 9 months ago

@GeeksforGeeks
Pls correct the typo. It is Wikipedia not WikiPedia. Maybe it doesn't make much difference but to the eyes Wikipedia is much more soothing and to brain much more reliable than WikiPedia.

1 • Reply • Share ›

GeeksforGeeks Mod Aditya Goel • 3 months ago

Aditya, thanks for pointing this out. We have updated it.

• Reply • Share ›

Rahul4u • a year ago

how the returned pointer node in insert unchanged??

1 • Reply • Share ›

Rajeev Rahul4u • a year ago
Returned pointer node in insert method is root node of the tree.
It should not be changed after the recursion completes.
• Reply • Share ›

suneel Rajeev • a year ago
actually we are changing the pointer of left and right child root is affected at all.
• Reply • Share ›

Jayesh • 17 hours ago
Java Implementation

http://javabypatel.blogspot.in...
• Reply • Share ›

harpreet • 10 days ago
my ques is: why here struct node*insert n struct node*newnode is used

for the insert n newnode function resp,

i mean what is the point of using it over void insert or int insert n

similarly int newnode n void newnode?
• Reply • Share ›

Prince Bharti • 2 months ago
java version:

https://ideone.com/gT51Xz
• Reply • Share ›

pratik • 5 months ago
struct node* insert(struct node *root,int data)

{

if(!root)

{

struct node *tmp=create(data);

root=tmp;

return root;

}

struct node* tmp=root;

while(tmp)

{

see more

• Reply • Share ›

Gaurav • 7 months ago

insert function is more easier if we use double pointer
void insert(struct node** root, int key)
{
/* If the tree is empty, return a new node */
if ((*root) == NULL){
(*root)=newNode(key);
return;
}

/* Otherwise, recur down the tree */
if (key < (*root)->key)
insert(&((*root)->left), key);
else
insert(&((*root)->right), key);
}

and main should include calls as
insert(&root, 50);
insert(&root, 30);

• Reply • Share ›

Guest • a year ago

can anyone tell me whats wrong with my code:

#include <stdio.h>

#include <stdlib.h>

#include "string.h"

struct bst

{

int data;

struct bst *left;

struct bst *right;

}newnode;

int preorder(struct bst *root)

see more

^ | ⌄ • Reply • Share ›

Avat

This comment was deleted.

Bharat → Guest • 10 months ago

The overall function returns the root in the end. This return statement is useful only for the first insertion as we must return the pointer to that. After that it will always return the pointer to the root.

^ | ⌄ • Reply • Share ›

Saurabh Anand → Guest • 10 months ago

It's basically to handle the case when you try to insert a node which is already in the tree.

^ | ⌄ • Reply • Share ›

ryan • a year ago

@GeeksforGeeks
please correct the code if some one try to insert same value the we will get not correct out put so change
else
node->right = insert(node->right, key);

to
else if(key > node->key) node->right = insert(node->right, key);

^ | ⌄ • Reply • Share ›

GeeksforGeeks Mod → ryan • a year ago

Thanks for pointing this out. We have updated the code.

^ | ⌄ • Reply • Share ›

sonu431 → GeeksforGeeks • 9 months ago

to ryan
sorry, but A/Q to definition of BST the previeous code of GeeksforGeeks is

right i.e---
else
node->right = insert(node->right, key);

because- if the number is less then root then it goes to left subtree but if the number is greater than or Equal to the root then it only goes to right subtree.

hense your code is not inserting if the no is equal to the root
else if(key > node->key) node->right = insert(node->right, key);
so this should be changed to previous code i.e.
else
node->right = insert(node->right, key);

1 • Reply • Share ›

sweety • a year ago

i have a small doubt..
if we want to insert 25 instead of 40 in the above tree it will nt be a leafnode then how can we insert it?

plzz can anyone xplain..

• Reply • Share ›

Abhishek Kannojia sweety • a year ago

It will inserted as left child of 30.

```
          100                                  100
         /     \            Insert 25      /       \
      20     500     ------------->        20      500
    /   \                               /    \
  10     30                           10      30
                                              /
                                           25
```

2 • Reply • Share ›

Kaushik Thirthappa sweety • a year ago

Irrespective of any case, the insertion has to go to the leaf node.

As abhishek puts it (and he is right), conceptually to arrange the data we go to the extreme left leaf (not root) and then the root and the right leaf. When we are on the right leaf, we will have to check if its root or not, if it is then repeat the procedure.

• Reply • Share ›

saroja • a year ago

i have a small doubt..
if we want to insert 11 in the above tree it will nt be a leafnode then how can we insert it?

plzz can anyone xplain..

• Reply • Share ›

ANA saroja • a year ago

In the Right of 10

1 • Reply • Share ›

max • a year ago

what if we skip the return node line?

• Reply • Share ›

RK max • a year ago

It wont work

• Reply • Share ›

Rohit Asati RK • a year ago

why i have same doubt???

• Reply • Share ›

Akash Maurya Rohit Asati • 10 months ago

it won't work , because the first element is inserted it needs to be returned as it will be root .
so, our function can't have void type .

for all next elements you need to return something of node * type , however it wont be used as it is always root.

• Reply • Share ›

max • a year ago

explain the functioning of the recursion in insert and role of return unchanged node

• Reply • Share ›

PureLeads Advertisement