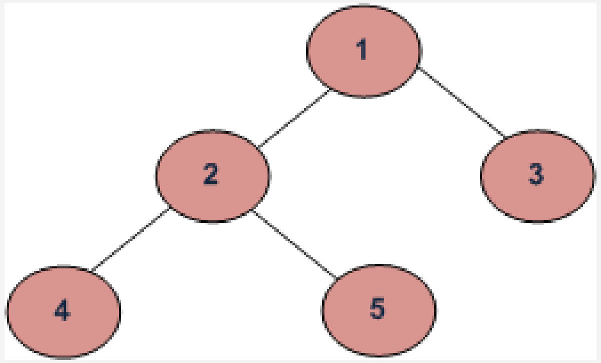
Tree Traversals

Unlike linear data structures (Array, Linked List, Queues, Stacks, etc) which have only one logical way to traverse them, trees can be traversed in different ways. Following are the generally used ways for traversing trees.



Example Tree

Depth First Traversals:
(a) Inorder
(b) Preorder
(c) Postorder

Breadth First or Level Order Traversal
Please see this post for Breadth First Traversal.

Inorder Traversal:

```
Algorithm Inorder(tree)
   1. Traverse the left subtree, i.e., call Inorder(left-subtree)
```

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
Geometric Algorithms

```
   2. Visit the root.
   3. Traverse the right subtree, i.e., call Inorder(right-subtree)
```

Uses of Inorder
In case of binary search trees (BST), Inorder traversal gives nodes in non-decreasing order. To get nodes of BST in non-increasing order, a variation of Inorder traversal where Inorder itraversal s reversed, can be used.
Example: Inorder traversal for the above given figure is 4 2 5 1 3.

Preorder Traversal:

```
Algorithm Preorder(tree)
   1. Visit the root.
   2. Traverse the left subtree, i.e., call Preorder(left-subtree)
   3. Traverse the right subtree, i.e., call Preorder(right-subtree)
```

Uses of Preorder
Preorder traversal is used to create a copy of the tree. Preorder traversal is also used to get prefix expression on of an expression tree. Please see http://en.wikipedia.org/wiki/Polish_notation to know why prefix expressions are useful.
Example: Preorder traversal for the above given figure is 1 2 4 5 3.

Postorder Traversal:

```
Algorithm Postorder(tree)
   1. Traverse the left subtree, i.e., call Postorder(left-subtree)
   2. Traverse the right subtree, i.e., call Postorder(right-subtree)
   3. Visit the root.
```

Uses of Postorder
Postorder traversal is used to delete the tree. Please see the question for deletion of tree for details. Postorder traversal is also useful to get the postfix expression of an expression tree. Please see http://en.wikipedia.org/wiki/Reverse_Polish_notation to for the usage of postfix expression.

Example: Postorder traversal for the above given figure is 4 5 2 3 1.

Popular Posts

All permutations of a given string
Memory Layout of C Programs
Understanding “extern” keyword in C
Median of two sorted arrays
Tree traversal without recursion and without stack!
Structure Member Alignment, Padding and Data Packing
Intersection point of two Linked Lists
Lowest Common Ancestor in a BST.
Check if a binary tree is BST or not
Sorted Linked List to Balanced BST

```
#include <stdio.h>
#include <stdlib.h>

/* A binary tree node has data, pointer to left child
   and a pointer to right child */
struct node
{
     int data;
     struct node* left;
     struct node* right;
};

/* Helper function that allocates a new node with the
   given data and NULL left and right pointers. */
struct node* newNode(int data)
{
     struct node* node = (struct node*)
                                  malloc(sizeof(struct node));
     node->data = data;
     node->left = NULL;
     node->right = NULL;

     return(node);
}

/* Given a binary tree, print its nodes according to the
  "bottom-up" postorder traversal. */
void printPostorder(struct node* node)
{
     if (node == NULL)
        return;

     // first recur on left subtree
     printPostorder(node->left);

     // then recur on right subtree
     printPostorder(node->right);

     // now deal with the node
     printf("%d ", node->data);
}

/* Given a binary tree, print its nodes in inorder*/
void printInorder(struct node* node)
{
     if (node == NULL)
          return;
     /* first recur on left child */
     printInorder(node->left);

     /* then print the data of node */
     printf("%d ", node->data);

     /* now recur on right child */
     printInorder(node->right);
}

/* Given a binary tree, print its nodes in inorder*/
void printPreorder(struct node* node)
{
     if (node == NULL)
          return;

     /* first print data of node */
     printf("%d ", node->data);

     /* then recur on left sutree */
     printPreorder(node->left);

     /* now recur on right subtree */
     printPreorder(node->right);
}

/* Driver program to test above functions*/
int main()
{
     struct node *root  = newNode(1);
     root->left             = newNode(2);
     root->right           = newNode(3);
     root->left->left     = newNode(4);
     root->left->right   = newNode(5);

     printf("\n Preorder traversal of binary tree is \n");
     printPreorder(root);

     printf("\n Inorder traversal of binary tree is \n");
     printInorder(root);

     printf("\n Postorder traversal of binary tree is \n");
     printPostorder(root);

     getchar();
     return 0;
```

Recent Comments

karthik it should have been max_wrap= max_wrap -...

Maximum circular subarray sum · 2 minutes ago

affiszerv Your example has two 4s on row 3, that's why it...

Backtracking | Set 7 (Sudoku) · 46 minutes ago

RVM Can someone please elaborate this Qs from above...

Flipkart Interview | Set 6 · 1 hour ago

Vishal Gupta I talked about as an Interviewer in general,...

Software Engineering Lab, Samsung Interview | Set 2 · 1 hour ago

@meya Working solution for question 2 of 4f2f round....

Amazon Interview | Set 53 (For SDE-1) · 1 hour ago

sandeep void rearrange(struct node *head) {...

Given a linked list, reverse alternate nodes and append at the end · 3 hours ago

```
}
```

Time Complexity: O(n)
Let us prove it:

Complexity function T(n) — for all problem where tree traversal is involved — can be defined as:

$T(n) = T(k) + T(n – k – 1) + c$

Where k is the number of nodes on one side of root and n-k-1 on the other side.

Let’s do analysis of boundary conditions

Case 1: Skewed tree (One of the subtrees is empty and other subtree is non-empty)

k is 0 in this case.
$T(n) = T(0) + T(n-1) + c$
$T(n) = 2T(0) + T(n-2) + 2c$
$T(n) = 3T(0) + T(n-3) + 3c$
$T(n) = 4T(0) + T(n-4) + 4c$

…………………………………………
……………………………………….…
$T(n) = (n-1)T(0) + T(1) + (n-1)c$
$T(n) = nT(0) + (n)c$

Value of T(0) will be some constant say d. (traversing a empty tree will take some constants time)

$T(n) = n(c+d)$
T(n) = (-)(n) (Theta of n)

Case 2: Both left and right subtrees have equal number of nodes.

T(n) = 2T(|_n/2_|) + c

This recursive function is in the standard form (T(n) = aT(n/b) + (-)(n)) for master method http://en.wikipedia.org/wiki/Master_theorem. If we solve it by master method we get (-)(n)

Auxiliary Space : If we don't consider size of stack for function calls then O(1) otherwise O(n).

Related Tpoics:

- Print a Binary Tree in Vertical Order | Set 2 (Hashmap based Method)
- Print Right View of a Binary Tree
- Red-Black Tree | Set 3 (Delete)
- Construct a tree from Inorder and Level order traversals
- Print all nodes at distance k from a given node
- Print a Binary Tree in Vertical Order | Set 1
- Interval Tree
- Check if a given Binary Tree is height balanced like a Red-Black Tree

3 Tweet 0 3

Writing code in comment? Please use **ideone.com** and share the link here.

18 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

DarkProtocol • 7 months ago

Two Types:

BFS-

1)Inorder - LRoot R --T(n) O(n) , Space used by stack in recursion S(n)=O(n)
Space used by stack in recursion S(n)=O(n) 3)Postorder - L R Root--T(n) O(r
S(n)=O(n)

DFS -

1) Level Order - With Recursion T(n) - O(n2)
- With Queue's - T(n) - O(n) and S(n) - O(n)

3 ⌃ | ⌄ • Reply • Share ›

dipak ➦ DarkProtocol • a month ago

preorder space used is O(n)...I am a bit confused can someone pleas

⌃ | ⌄ • Reply • Share ›

initialcoder • 11 months ago

```
 #include<stdio.h>
#include<stdlib.h>

typedef struct NodeTag{
        char SYMBOL;
        struct NodeTag * LLINK;
        struct NodeTag * RLINK;
} TreeNode;
```

```
typedef enum{PreOrder, InOrder, PostOrder} OrderOfTraverse;

void visitNode(TreeNode * node){
        if(node == NULL)
                return;
        printf("%c ", node->SYMBOL);
}

void treeTraverse(TreeNode * treeRoot, OrderOfTraverse TraverseOrder)
```

see more

Reply • Share ›

abhishek08aug • a year ago

Here is the C++ design/code for BST traversals. Recursive insert is bit tricky.

http://stackoverflow.com/quest...

```
 #include<iostream>
using namespace std;

class tree_node {
  private:
    int data;
    tree_node * left;
    tree_node * right;
  public:
    tree_node() {
      left=NULL;
      right=NULL;
    }
    void set_data(int data) {
      this->data=data;
```

see more

1 ⌃ | ⌄ • Reply • Share ›

abhishek08aug • a year ago

Here is the C++ design/code for BST traversals. Recursive insert is bit tricky.

http://stackoverflow.com/quest...

```
[sourcecode language="C++"]
#include<iostream>
using namespace std;

class tree_node {
private:
int data;
tree_node * left;
tree_node * right;
public:
tree_node() {
left=NULL;
right=NULL;
}
void set_data(int data) {
```

see more

⌃ | ⌄ • Reply • Share ›

Marsha Donna ➦ abhishek08aug • 2 months ago

i hav written a simpler code to recursively insert in bst..but it gives erro
Runtime error time: 0 memory: 2288 signal:11
can some1 help me out the link is
http://ideone.com/HVLJ1q

| • Reply • Share ›

Nikin • a year ago

```
 void inorder(node *sr)
{
if(sr == NULL)
return;
inorder(sr->left); cout<<sr->data<<" "; inorder(sr->right);
}
```

Reply • Share ›

Deepak • 2 years ago

When inorder traversing a tree resulted E A C K F H D B G; the preorder trave
… but how does it so……. can u explain it

Reply • Share ›

Avinash • 2 years ago

Post order is the toughest out of all the tree traversals when solved iteratively.

```
 //Post Order Traversal
        public void printpostorder()
        {
            Node current = Root;
            Node temp;
            if (current == null)
            {
                Console.WriteLine("Empty Tree");
                return;
            }
            else
            {
                Stack<Node> myStack = new Stack<Node>();
                while (true)
                {
                    if (current !=null)
```

see more

• Reply • Share ›

Jacopo • 3 years ago

Your analysis is correct but you wronged the math. The last step is: T(n) = nT
Right?

• Reply • Share ›

raa • 3 years ago

but what about complexity in the following case:
if number of nodes in left and right sub trees are not zero and not equal.

• Reply • Share ›

kartik ➦ raa • 3 years ago

For general cases, we can use substitution method given in CLRS boo
O(n).

• Reply • Share ›

Algoseekar • 3 years ago

WHATS THE k in time complexity

T(n) = T(k) + T(n – k – 1) + c

Please reply asap

• Reply • Share ›

GeeksforGeeks ➦ Algoseekar • 3 years ago

k is the number of nodes on one side of root and n-k-1 on the other sid

⌃ | ⌄ • Reply • Share ›

devraj • 3 years ago

it could be better if the reason is mention here that why different types of trave

⌃ | ⌄ • Reply • Share ›

Venki • 4 years ago

There is a typo in the description "Uses of Postorder
Preorder traversal ... " Change postorder to preorder. Level order code is miss

1 ⌃ | ⌄ • Reply • Share ›

GeeksforGeeks ➦ Venki • 4 years ago

@Venki: Thanks for pointing this out. we have corrected the typo. We h
(http://geeksforgeeks.org/?p=26... for Level Order Traversal

⌃ | ⌄ • Reply • Share ›

raa • 4 years ago

You are the one - giving the clear details with space complexity also - nice pos

⌃ | ⌄ • Reply • Share ›

✉ Subscribe   Add Disqus to your site