

# Data Structure 2018

## Lab 08

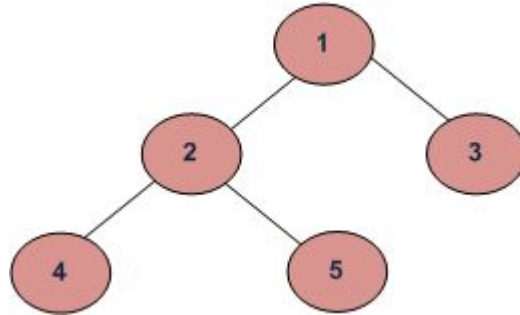
MohammadSadegh Najafi

Nov 1, 2018

# Binary Tree Traversal

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.



# Binary Tree Traversal

Depth First Traversals:

(a) Inorder (Left, Root, Right) : 4 2 5 1 3

(b) Preorder (Root, Left, Right) : 1 2 4 5 3

(c) Postorder (Left, Right, Root) : 4 5 2 3 1

Breadth First or Level Order Traversal : 1 2 3 4 5

# Inorder

# Traversal

Algorithm Inorder(tree)

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

# Preorder Traversal

Algorithm

Preorder

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)

# 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.

# Today's Task

First: Iterative Postorder Traversal (Using one Stack).

The idea is to move down to leftmost node using left pointer. While moving down, push root and root's right child to stack. Once we reach leftmost node, print it if it doesn't have a right child. If it has a right child, then change root so that the right child is processed before.

# Today's Task

Second: implement Inorder, Preorder and Postorder