Data Structures
Traversal of a Binary Tree

Cain Susko

Queen's University
School of Computing

February 5, 2022

Traversal

Traversal means to visit every node in a tree, any time we want to traverse a tree we start from the root. There are 4ways to traverse a tree.

Preorder Traversal

the algorithm for this operation is:

1. process current node
2. process the nodes in the left subtree
3. process the nodes in the right subtree

Note: process means something like the following:

- print the current node
- search node to see if its value matches a target
- add node value to sum
- etc…

This algorithm is implemented recursively as one processes a node and then follows the path down to the leftmost node and then works its way back up processing the right subtrees of each node that was processed in the left step. The path can be summarized as following the perimiter of the tree counterclockwise. A node is read when the path passes the left side of a node.

Inorder Traversal

1. process the nodes in the left subtree
2. process the current node
3. process the nodes in the right subtree

if you draw a counterclockwise path around the circle, the node is read when the path passes underneath a node

Postorder Traversal

This algorithm is very similar to the Preorder trvaersal.

1. Process the nodes in the left subtree
2. process the nodes in the right subtree
3. process the current node

using a similarly counterclockwise path, a node is only read when the path passes to the right of a node.

Level order Traversal

in a level order traversal algorithm we visit each level's nodes from left to right before then visiting the nodes on the next level. This algorithm is non-recursive. The order in which the nodes are read can be determined by defining each row and reading top-down left-right.

Implementation

One could use a Queue in order to traverse a tree like so

1. use a temo variable and queue
2. insert the root node into the queue
3. while the queue is not empty
   (a) dequeue the top node and put it in temp
   (b) process the node
   (c) enqueue the nodes children to the queue if the are not None