UNIVERSITY OF CALGARY
DEPARTMENT OF COMPUTER SCIENCE
WINTER 2013
CPSC 319: Data Structures, Algorithms and Their Applications
TA: Tamer N. Jarada

Tutorial #13

(March 14th, 2013)

Objectives:

- Do a recursive depth-first traversal of a tree.
- Do a breadth-first traversal of a tree.

Introduction:

An important class of algorithms is to traverse an entire data structure – visit every element in some fixed order

For trees there are two types of traversals, each with their variations:

- **Depth first traversal (Go as deep as possible before going along a level):** pre-order, in-order, post-order – each going clockwise or anticlockwise around the tree.
- **Breadth first traversal (Level by level):** Left to right across a level, or, right to left across a level.

Recursive depth-first traversal of a tree:

Depth First search of binary trees has three variations based on the order of visiting the root node relative to visiting the sub-tree nodes. These orders are:

- **Pre-order:** in pre-order, the root node is visited first prior to visiting the nodes of the sub-trees.
- **In-order:** in in-order, the root node is visited between visits to nodes of the left and right sub-trees.
- **Post-order:** in post-order, the root node is visited last after visiting the nodes of the sub-trees.

Example:

Traverse the tree shown in Figure (1) by using pre-order, in-order and post-order depth-first algorithms.

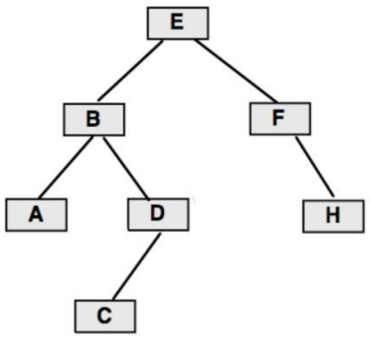

Figure (1)

- **Pre-order:** EBADCFH
- **In-order:** ABCDEFH
- **Post-order:** ACDBHFE

We also have another 3 orderings by going clockwise around the tree.

Code
```
protected void preorder(BSTNode n) {
  if (n != null) {
        visit(n);
        preorder(n.left);
        preorder(n.right);
      }
 }

protected void inorder(BSTNode n) {
  if (n != null) {
        inorder(n.left);
        visit(n);
        inorder(n.right);
      }
 }

protected void postorder(BSTNode n) {
  if (n != null) {
        postorder(n.left);
        postorder(n.right);
        visit(n);
      }
 }
```

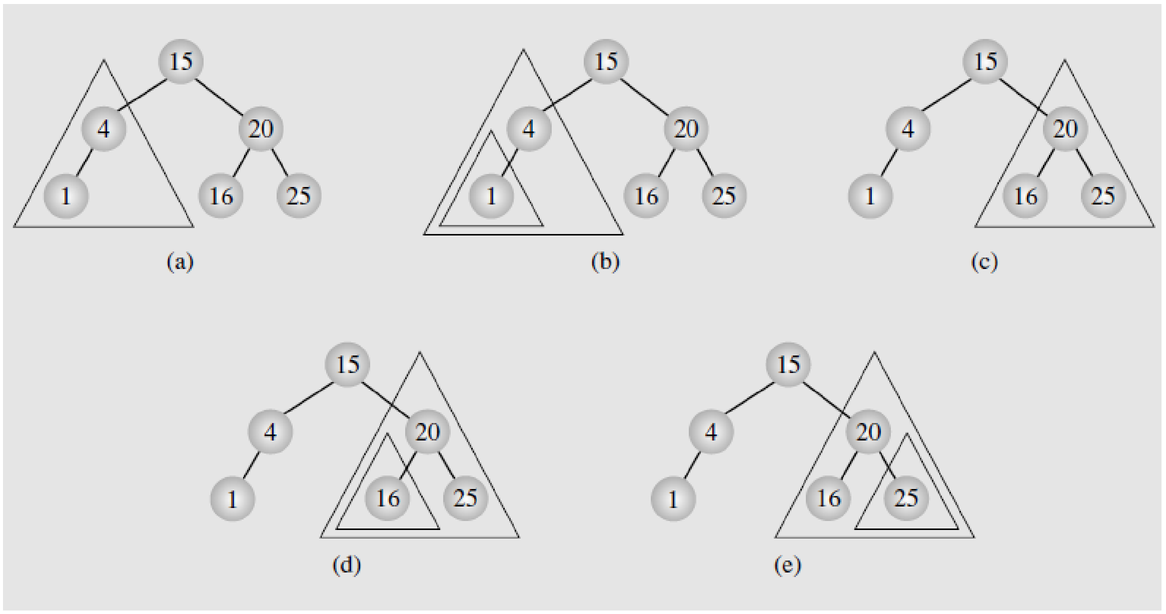

Figure (2): Example of applying depth-first traversal in-order

Breadth-first traversal of a tree:

A breadth-first search explores nodes nearest the root before exploring nodes further away.

Example:

Traverse the tree shown in Figure (1) by using breadth-first algorithm.

E B F A D H C or **E F B H D A C**

Below the Top-down, left-to-right, breadth-first traversal implementation:

Code

```
public void breadthFirst () {
  BSTNode n = root;
  Queue queue = new Queue();

  if (n != null) {
        queue.enqueue(n);

     while (!queue.isempty()) {
        n = (BSTNode) queue.dequeue();
        visit(n);
        if (n.left != null) {
             queue.enqueue(n.left);
        }
        if (n.right != null) {
             queue.enqueue(n.right);
        }
     }
  }
}
```