CSC 172– Data Structures and Algorithms

Lecture #18

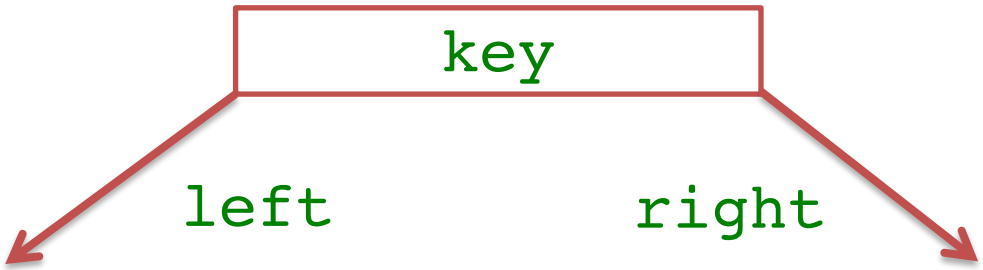
Spring 2018

Please put away all electronic devices

TREE USING JAVA

A BTNode in Java

```
public class Node
{
    public int key;
    public Node left, right;

    public Node(int item)
    {
        key = item;
        left = right = null;
    }
}
```



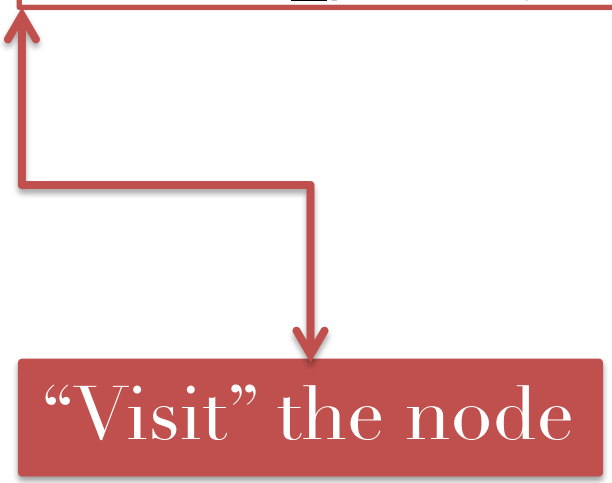
Inorder Traversal

```
Inorder-Traverse(BTNode root)

- Inorder-Traverse(root.left)
- Visit(root)
- Inorder-Traverse(root.right)
```

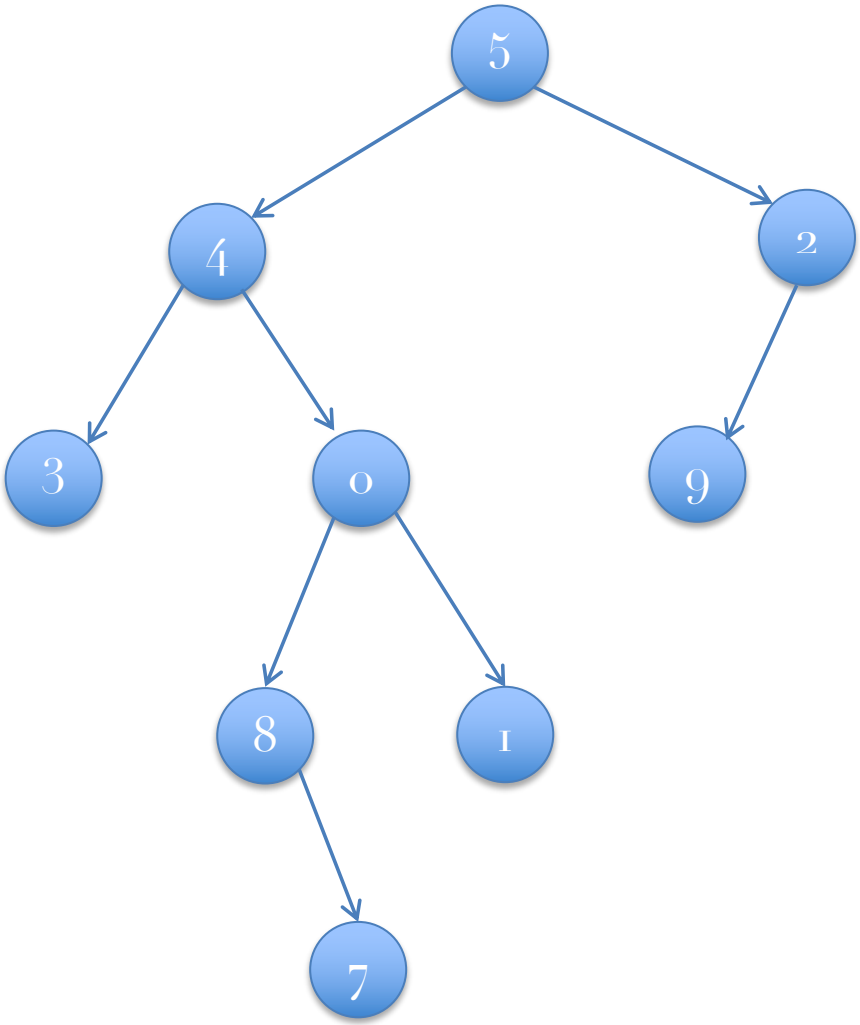
Also called the *(left, node, right)* order

Inorder Printing in C++

```
void inorder_print(BTNode root)
{
    if (root != null) {
        inorder_print(root.left);
        printNode(root);
        inorder_print(root.right);
    }
}
```



In Picture



3
4
8
7
0
1
5
9
2

Run Time

- Suppose "visit" takes O(1)-time, say *c* sec
  - n_l = # of nodes on the left sub-tree
  - n_r = # of nodes on the right sub-tree
  - Note: $n - 1 = n_l + n_r$
- $T(n) = T(n_l) + T(n_r) + c$
- Induction: $T(n) \leq cn$, i.e. $T(n) = O(n)$
- $T(n) \leq cn_l + cn_r + c$
  $= c(n-1) + c$
  $= cn$

Reverse Inorder Traversal

- `RevInorder-Traverse(root.right)`
- `Visit(root)`
- `RevInorrder-Traverse(root.left)`

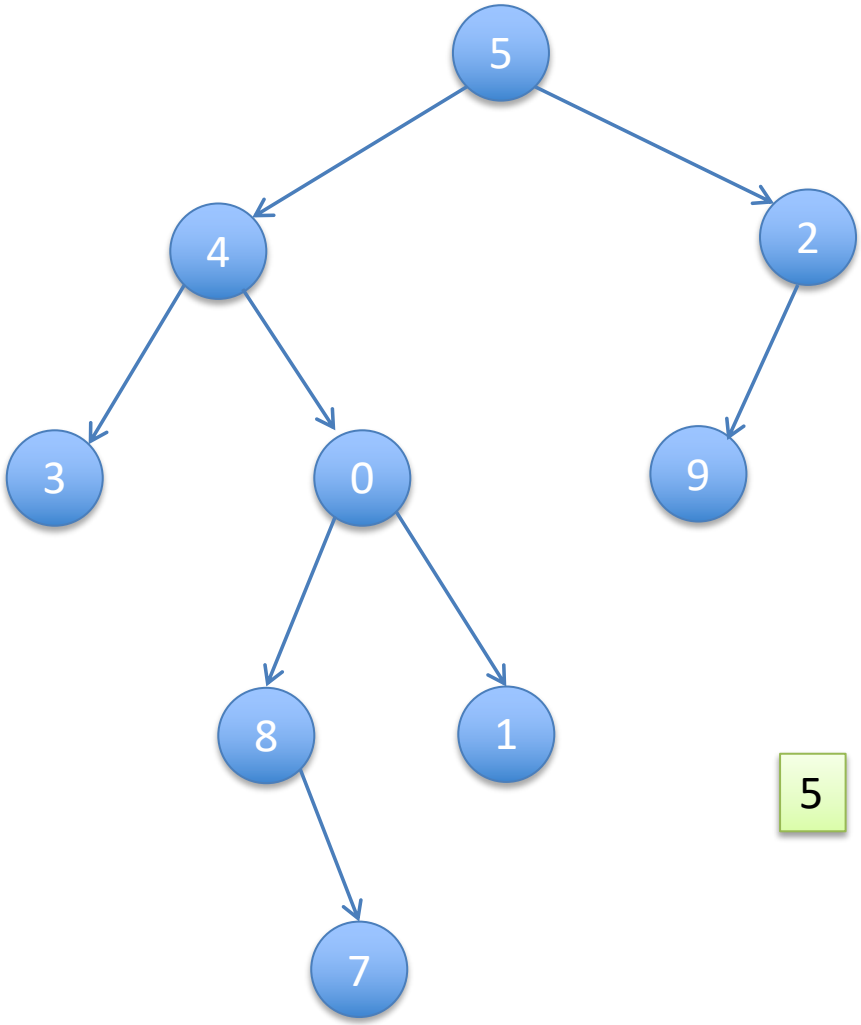
The (right, node, left) order

The other 4 traversal orders

- Preorder: (node, left, right)
- Reverse preorder: (node, right, left)
- Postorder: (left, right, node)
- Reverse postorder: (right, left, node)

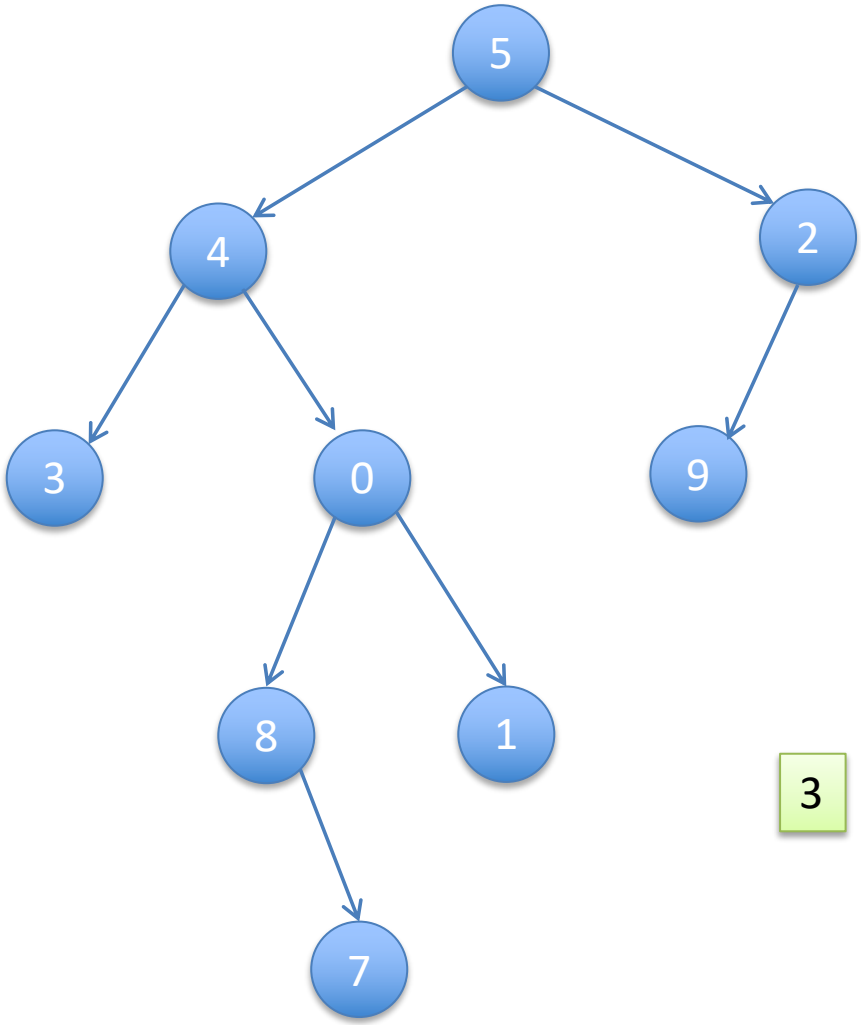
We'll talk about level-order later

What is the preorder output for this tree?



5	4	3	0	8	7	1	2	9

What is the postorder output for this tree?



3	7	8	1	0	4	9	2	5

Questions to Ponder

```
void inorder_print(BTNode root) {
    if (root != NULL) {
        inorder_print(root.left);
        printNode(root);
        inorder_print(root.right);
    }
}
```

Write the above routine without the recursive calls?

Using a stack

Without using a stack

Read about this just for fun!
Morris Inorder Tree Traversal
Not reqd. for quiz or exam.

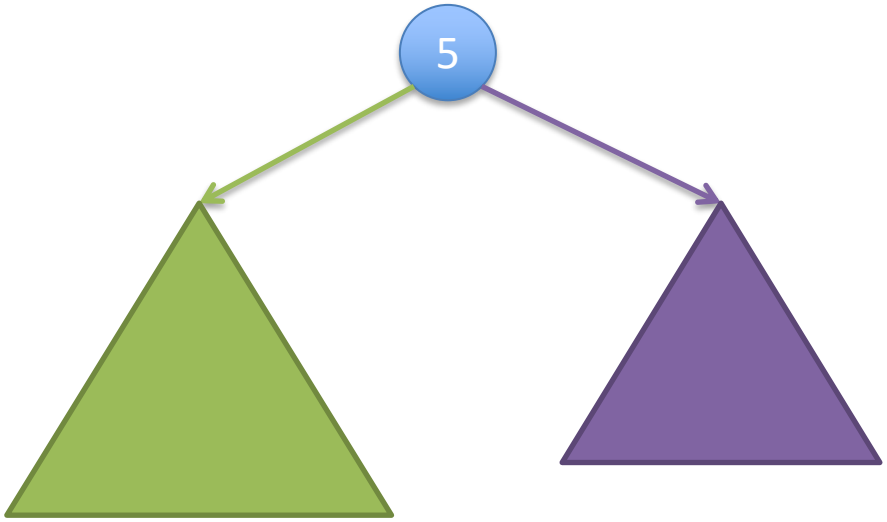
Reconstruct the tree from inorder+postorder

Inorder

3	4	8	7	0	1	5	9	2

Preorder

5	4	3	0	8	7	1	2	9



Questions to Ponder

- Can you reconstruct the tree given its postorder and preorder sequences?
- How about inorder and reverse postorder?
- How about other pairs of orders?
- How many trees are there which have the same in/post/pre-order sequence? (suppose keys are distinct)

Number of trees with a given inorder sequence

Catalan numbers

$$C_n = \sum_{i=1}^{n} C_{i-1} C_{n-i}$$

$$C_0 = 1$$

$$C_n = \frac{1}{n+1}\binom{2n}{n} \approx \frac{4^n}{n^{3/2}\sqrt{\pi}}$$

https://en.wikipedia.org/wiki/Catalan_number

What is a traversal order good for?

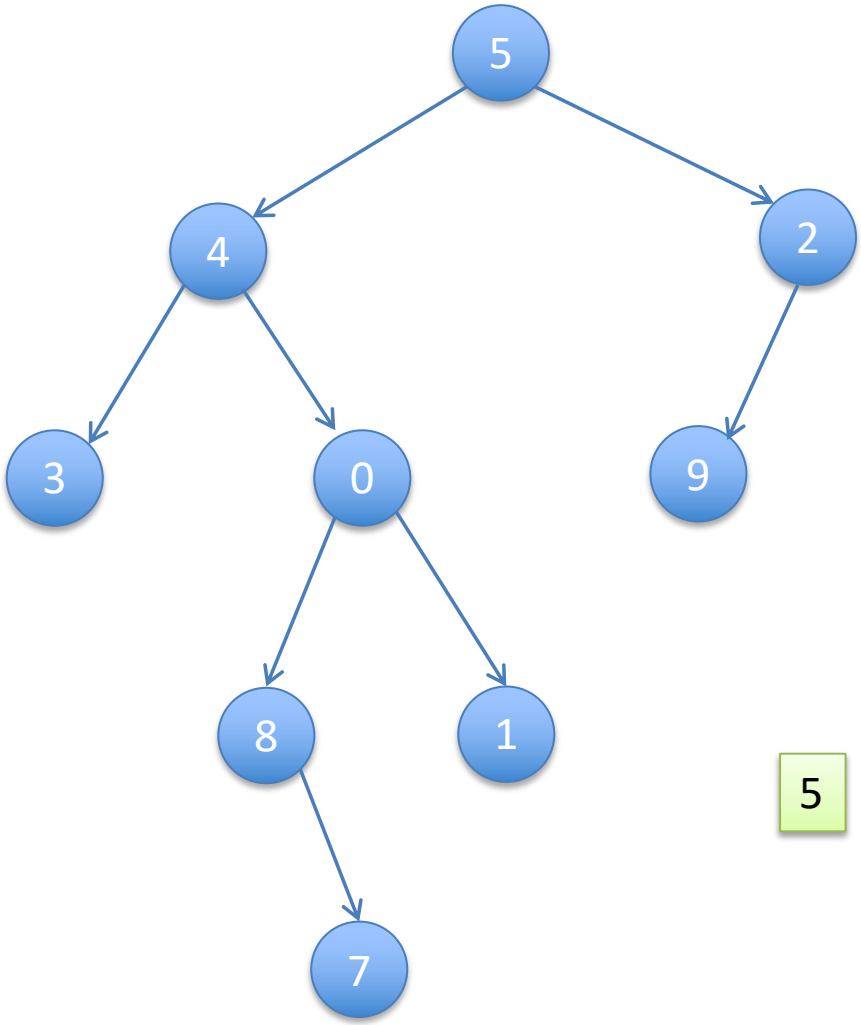
- Many things

- E.g., Evaluate(root) of an expression tree
  - If root is an operand, return the operand
  - Else
    - A = Evaluate(root.left)
    - B = Evaluate(root.right)
    - Return A root.key B
      - root.key is one of the operator

- What traversal order is the above?

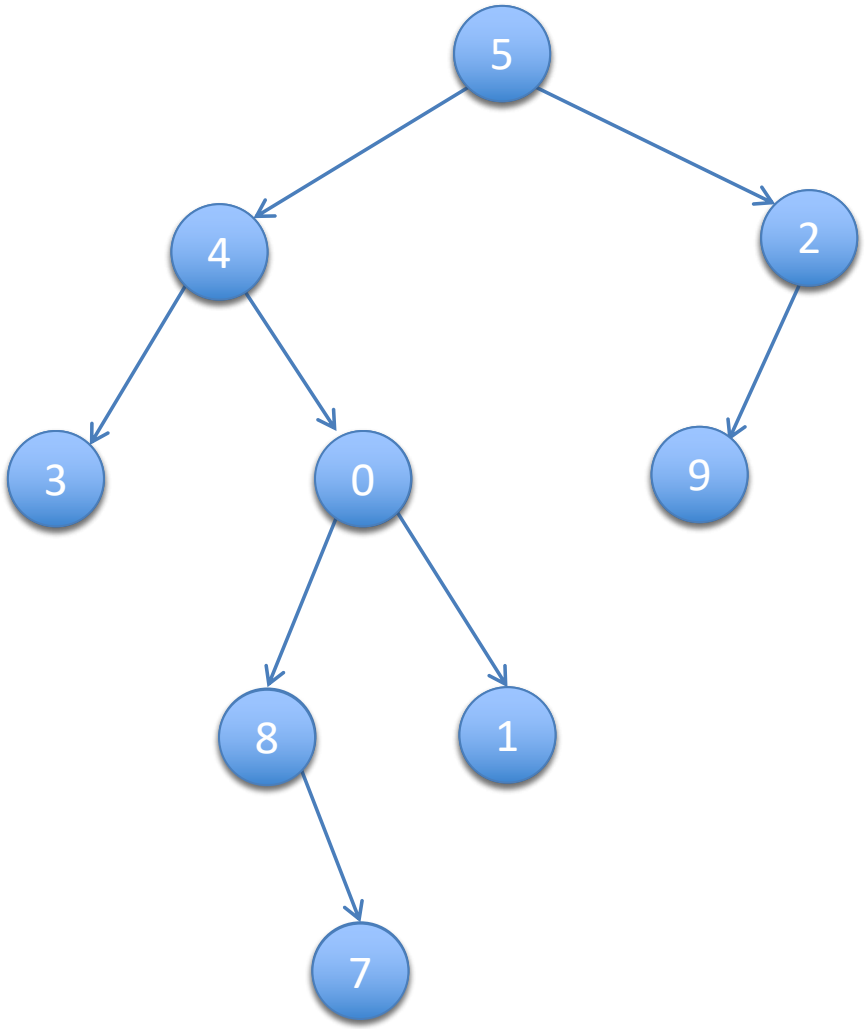
Answer: Post-order

Level-Order Traversal



5	4	2	3	0	9	8	1	7

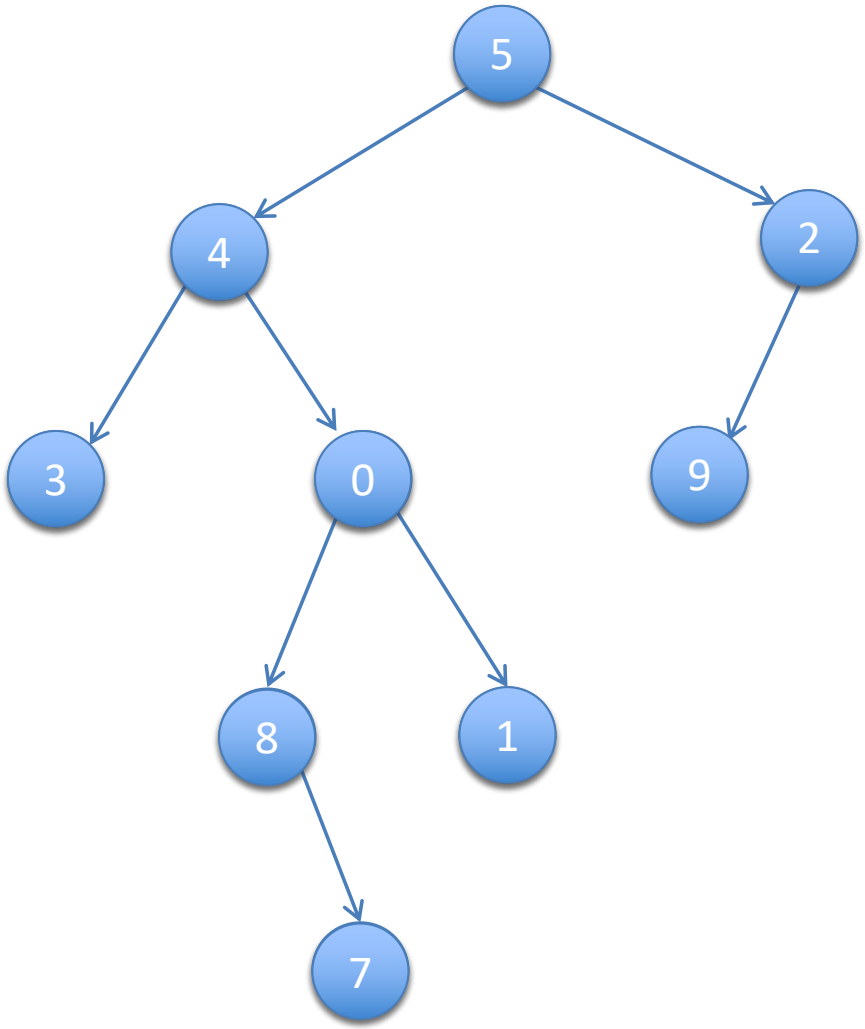
How to do level-order traversal?



Not a recursive solution!

A (FIFO) Queue

How to do in-order traversal? (without recursion)

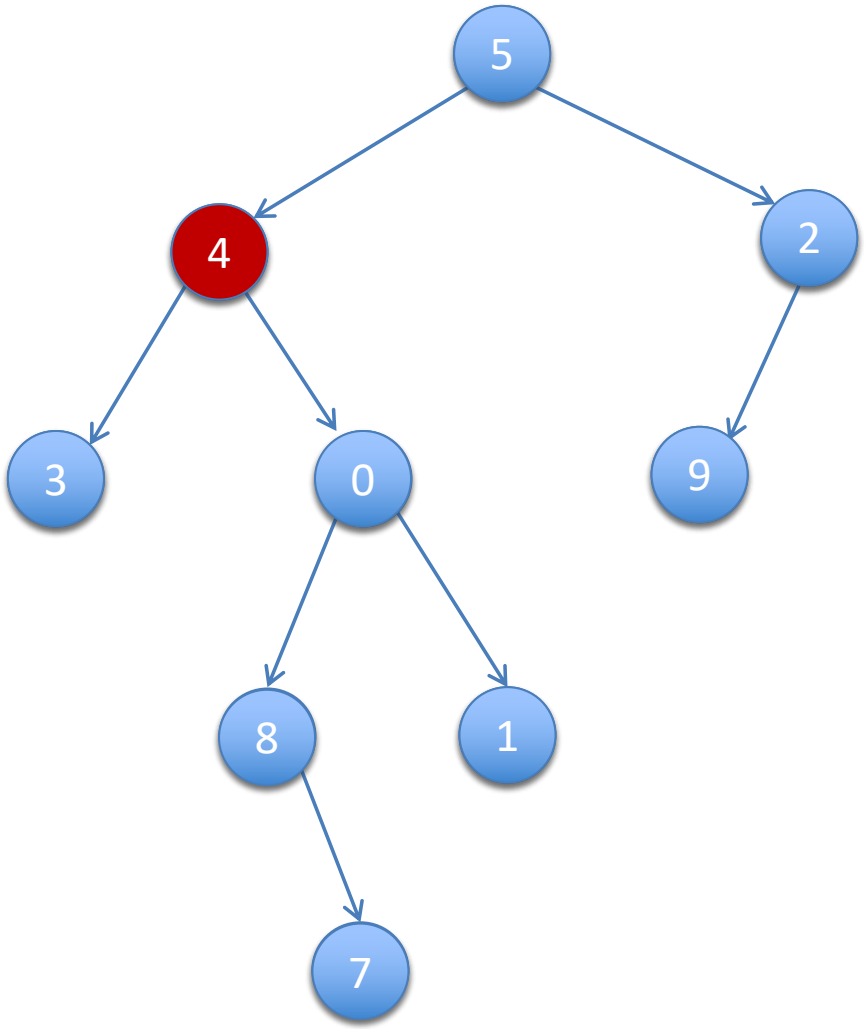


Using a Stack!
Just use a Stack instead of a Queue

A Stack

Output

How to do in-order traversal? (without recursion)

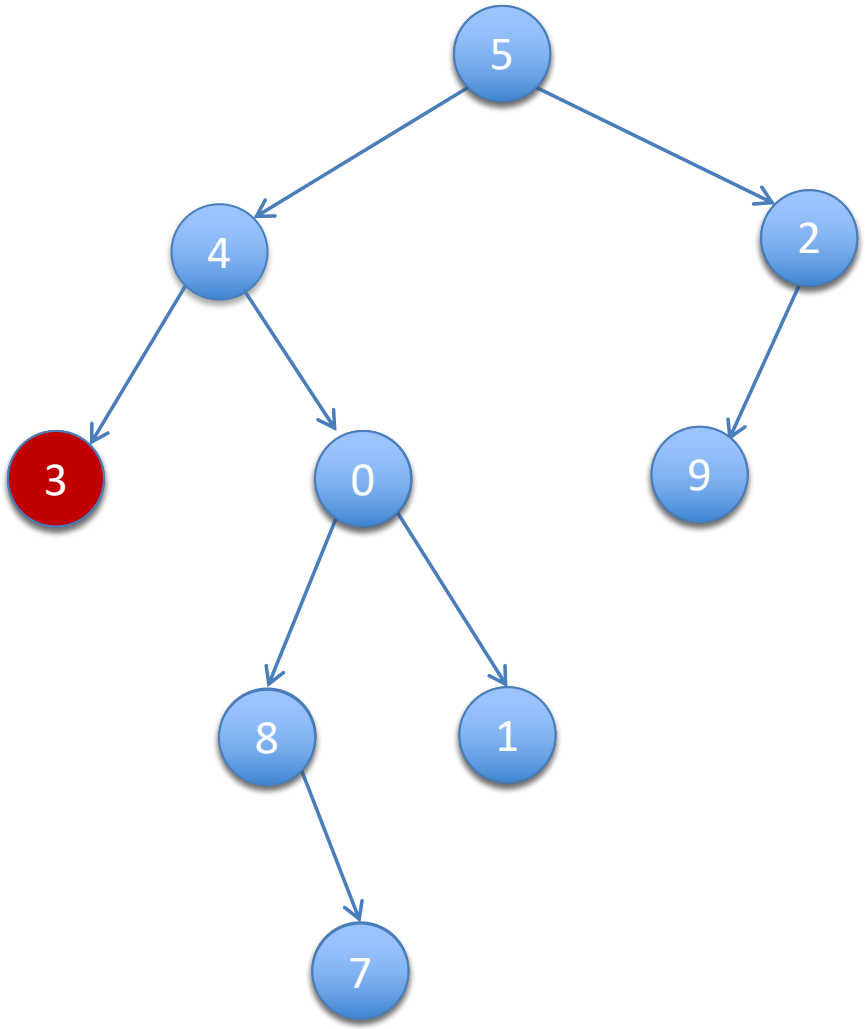


Using a Stack!
Just use a Stack instead of a Queue

A Stack

Output

How to do in-order traversal? (without recursion)

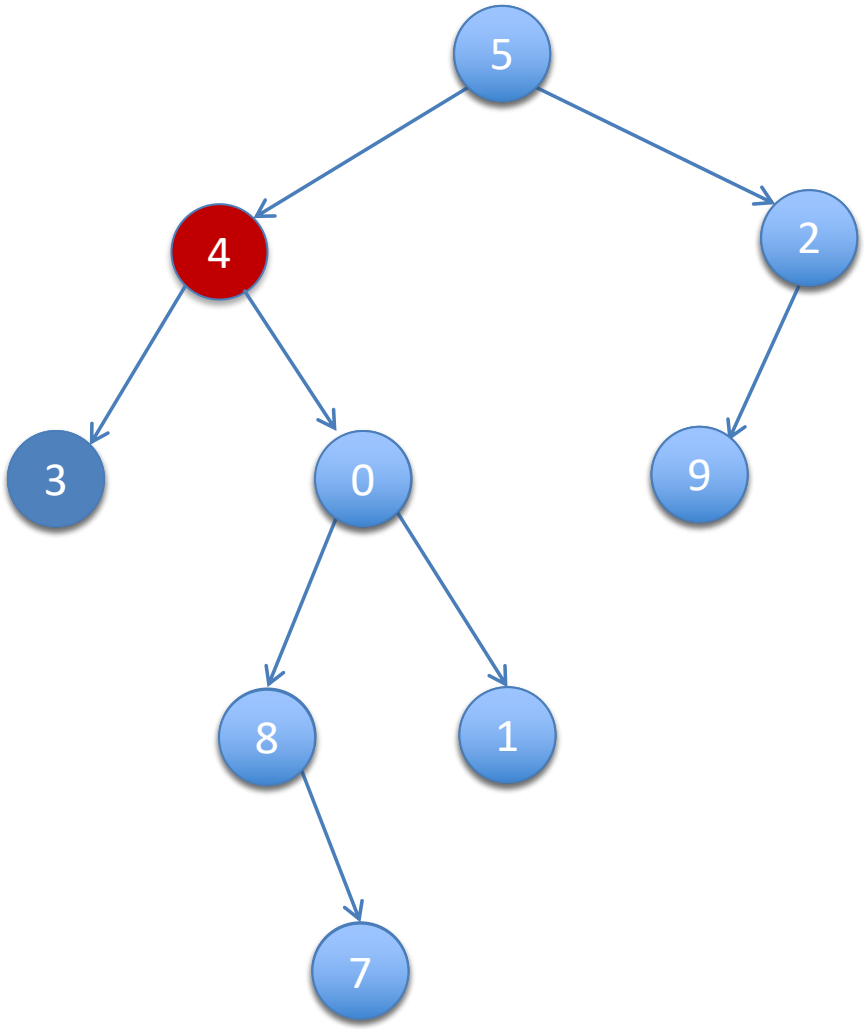


Using a Stack!
Just use a Stack instead of a Queue

A Stack

Output

How to do in-order traversal? (without recursion)

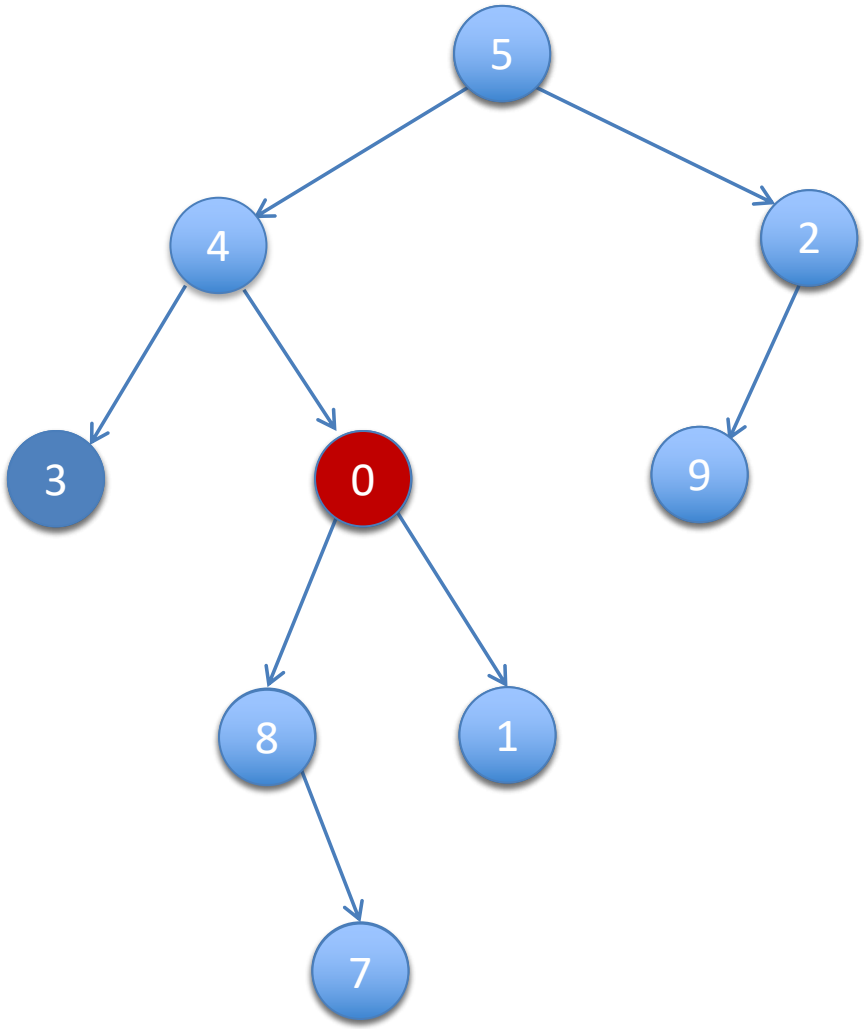


Using a Stack!
Just use a Stack instead of a Queue

A Stack

Output

How to do in-order traversal? (without recursion)

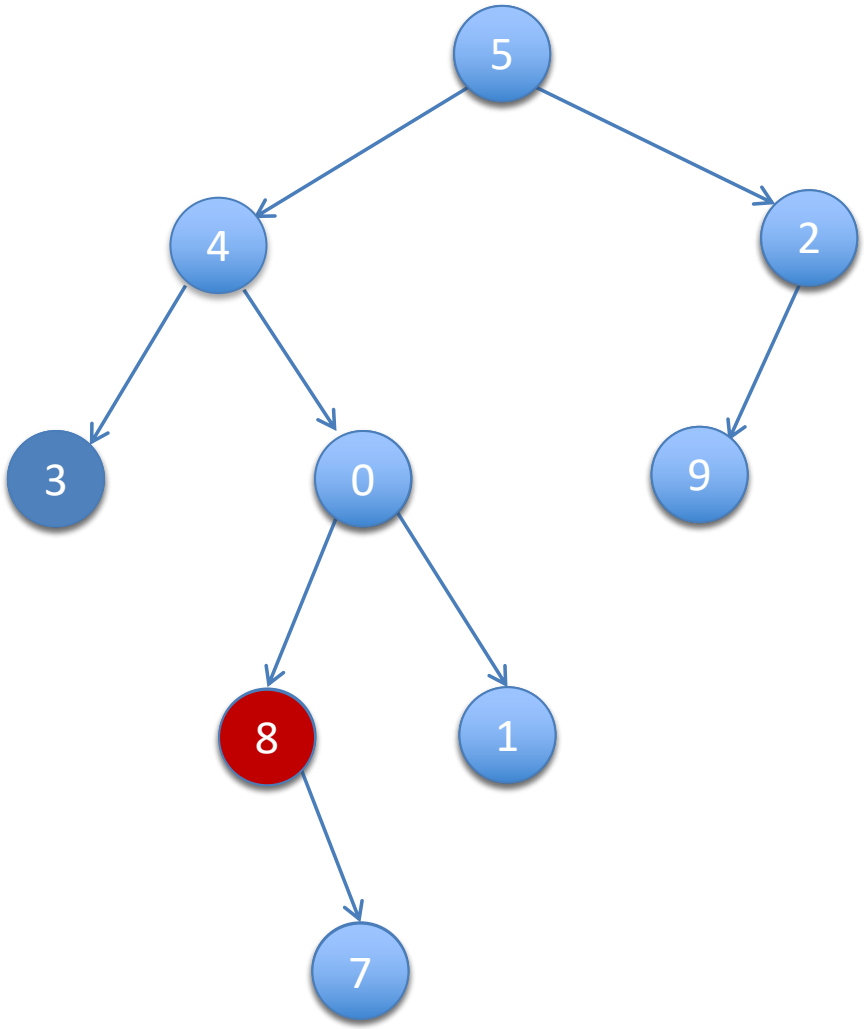


Using a Stack!
Just use a Stack instead of a Queue

A Stack

Output

How to do in-order traversal? (without recursion)

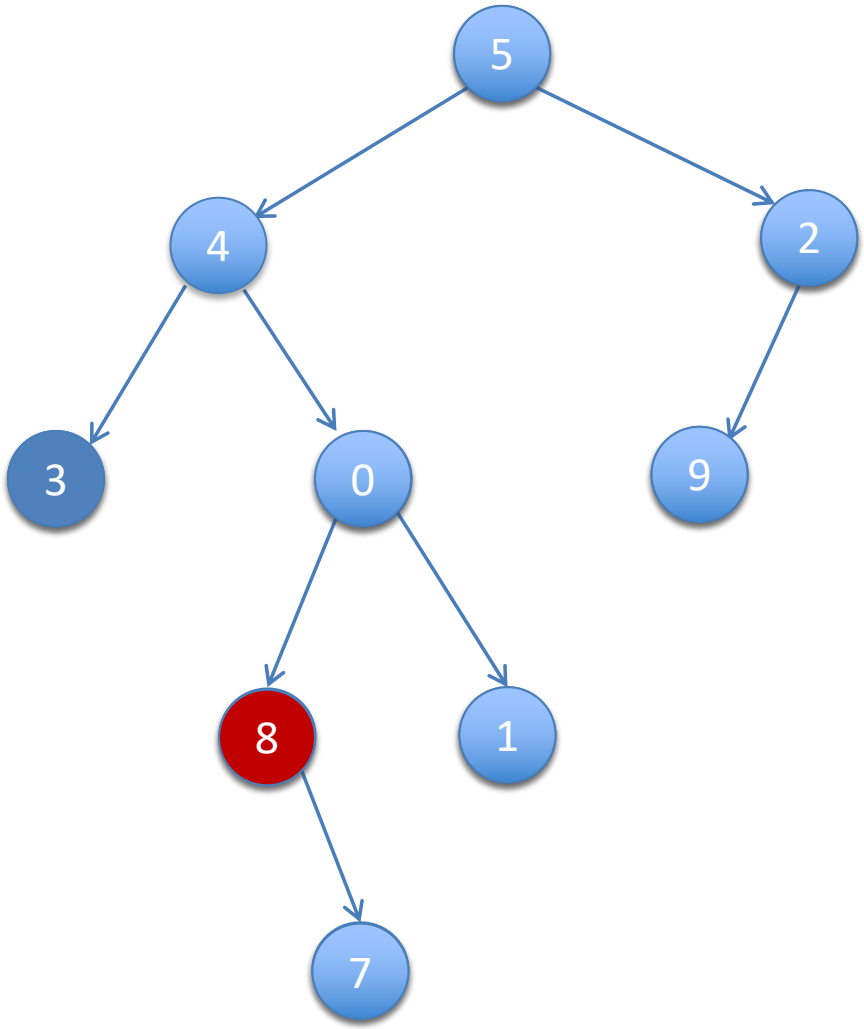


Using a Stack!
Just use a Stack instead of a Queue

A Stack

Output

How to do in-order traversal? (without recursion)

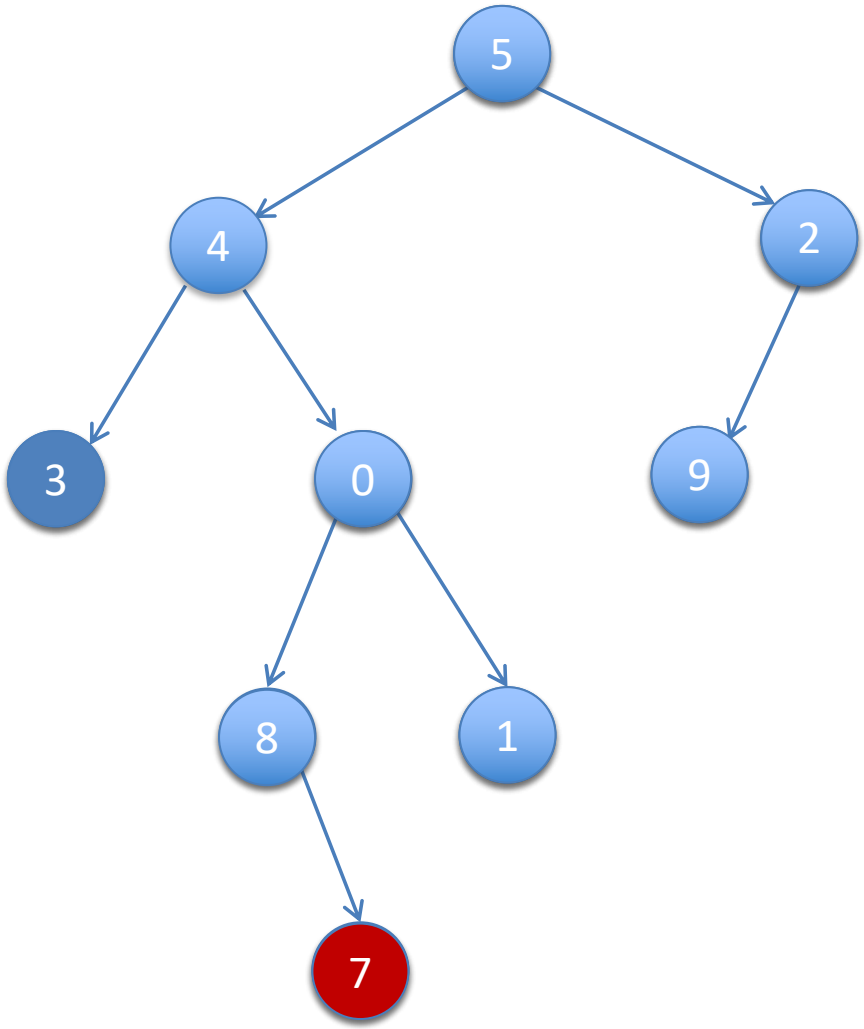


Using a Stack!
Just use a Stack instead of a Queue

A Stack

Output

How to do in-order traversal? (without recursion)

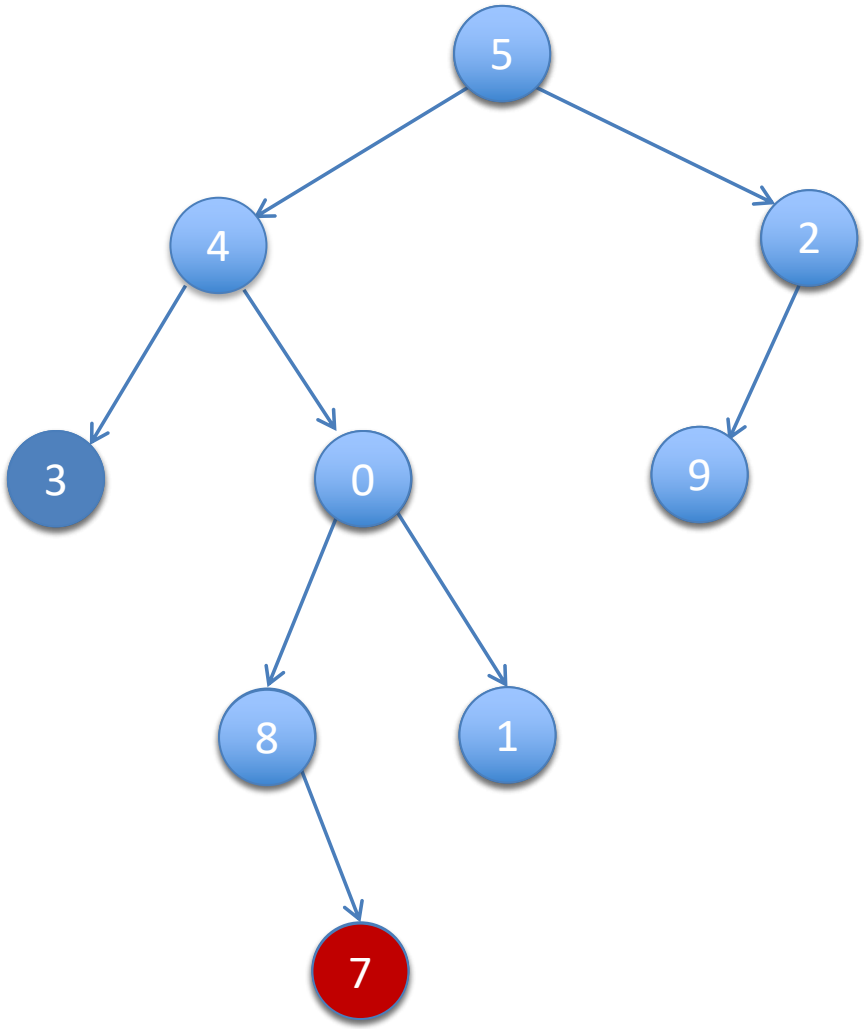


Using a Stack!
Just use a Stack instead of a Queue

A Stack

Output

How to do in-order traversal? (without recursion)

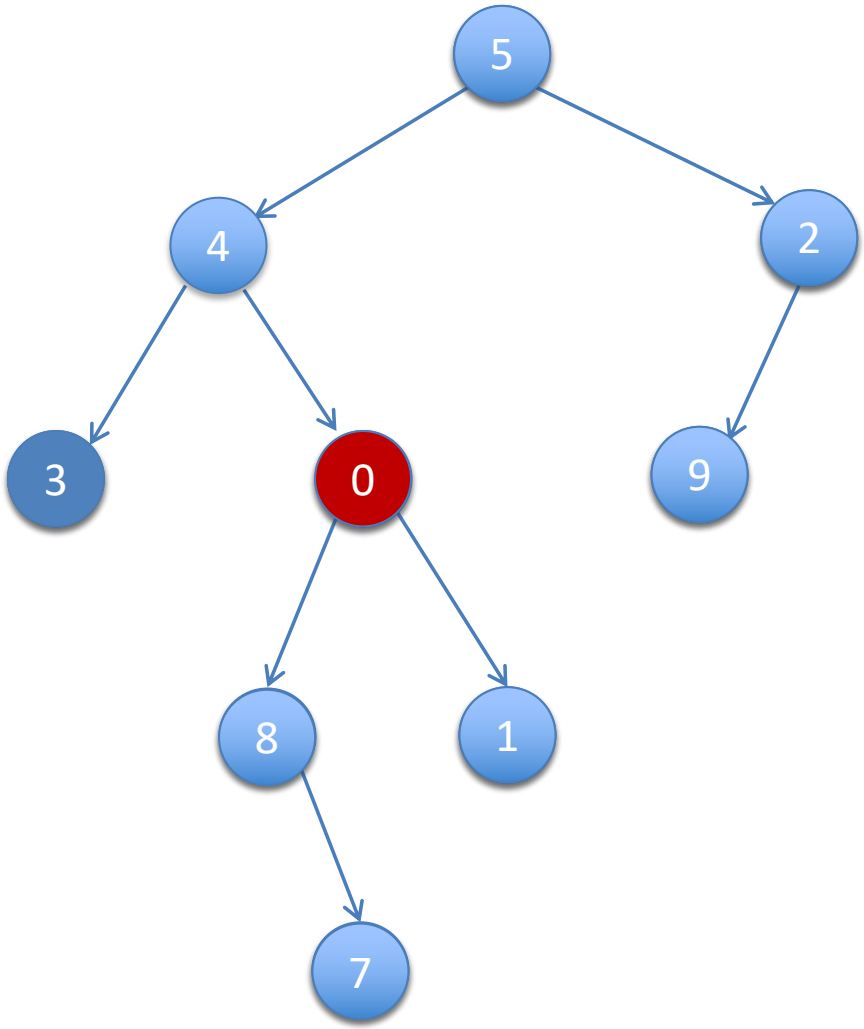


Using a Stack!
Just use a Stack instead of a Queue

A Stack

Output

How to do in-order traversal? (without recursion)

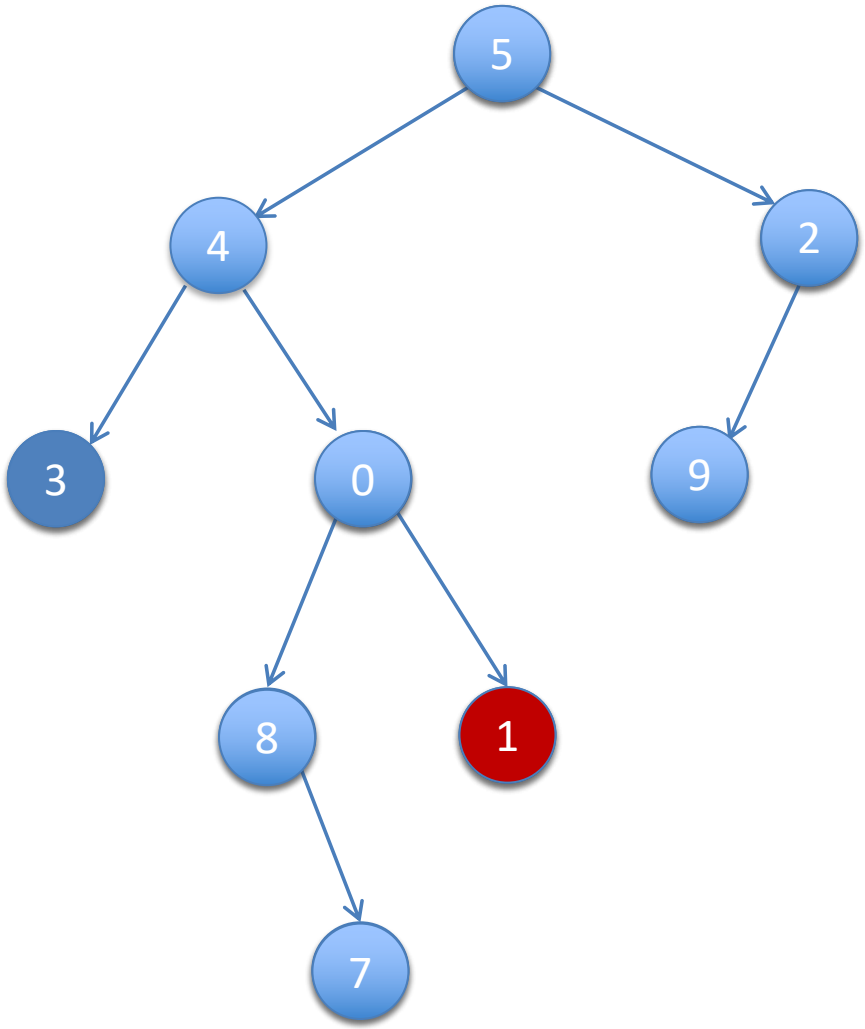


Using a Stack!
Just use a Stack instead of a Queue

A Stack

Output

How to do in-order traversal? (without recursion)

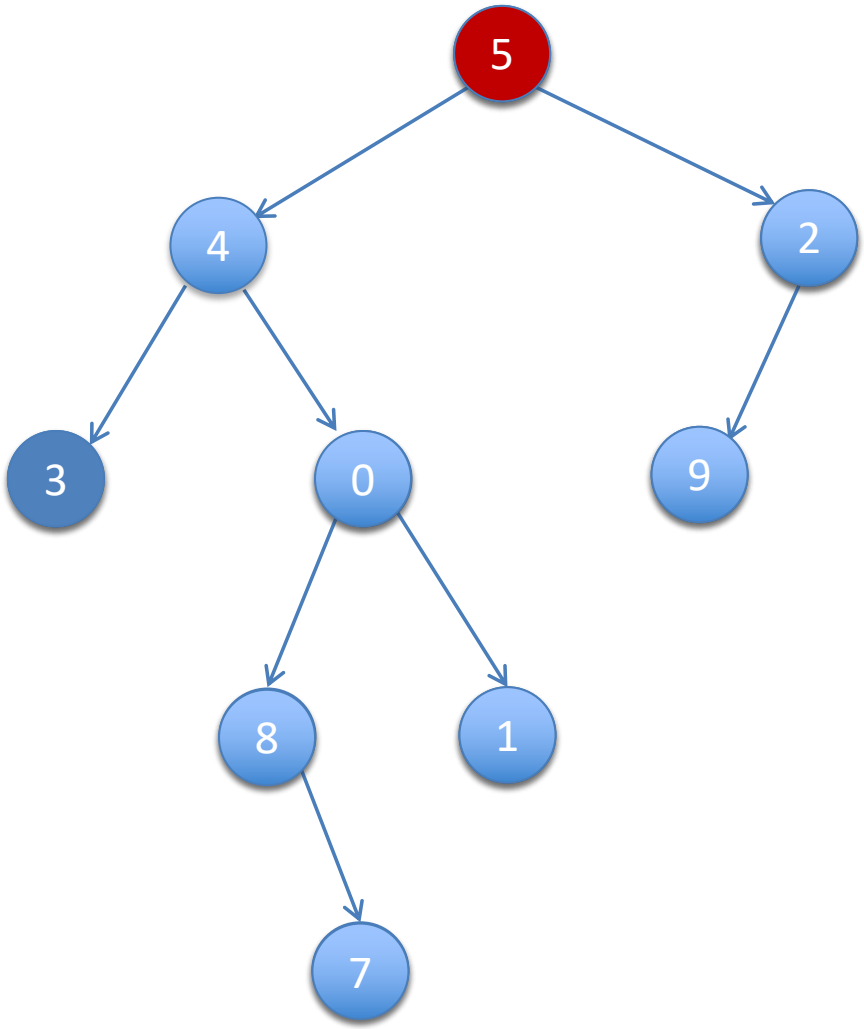


Using a Stack!
Just use a Stack instead of a Queue

A Stack

Output

How to do in-order traversal? (without recursion)

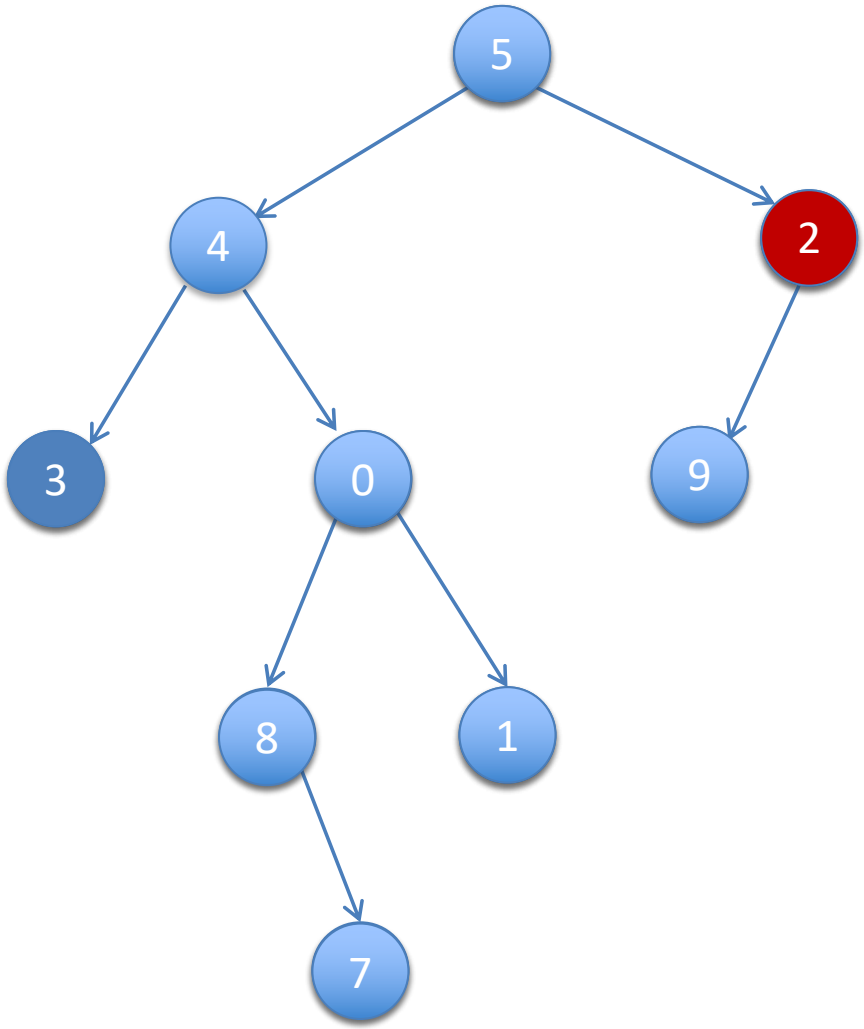


Using a Stack!
Just use a Stack instead of a Queue

A Stack

Output

How to do in-order traversal? (without recursion)

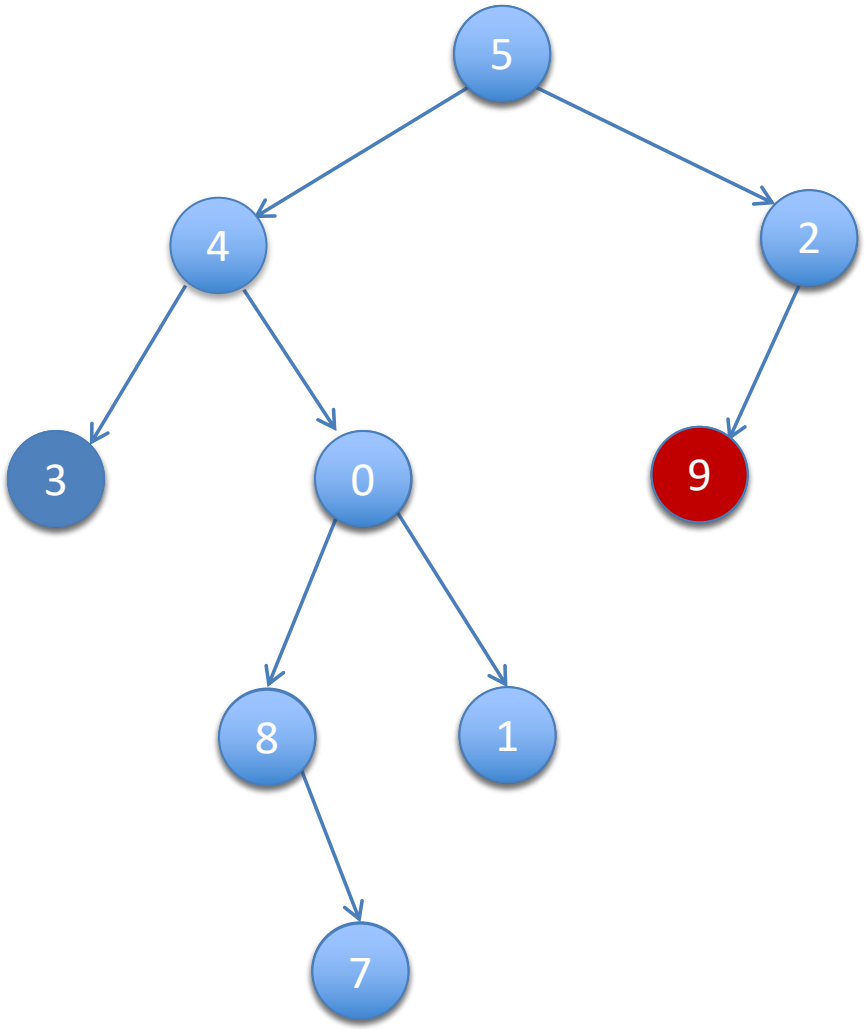


Using a Stack!
Just use a Stack instead of a Queue

A Stack

Output

How to do in-order traversal? (without recursion)

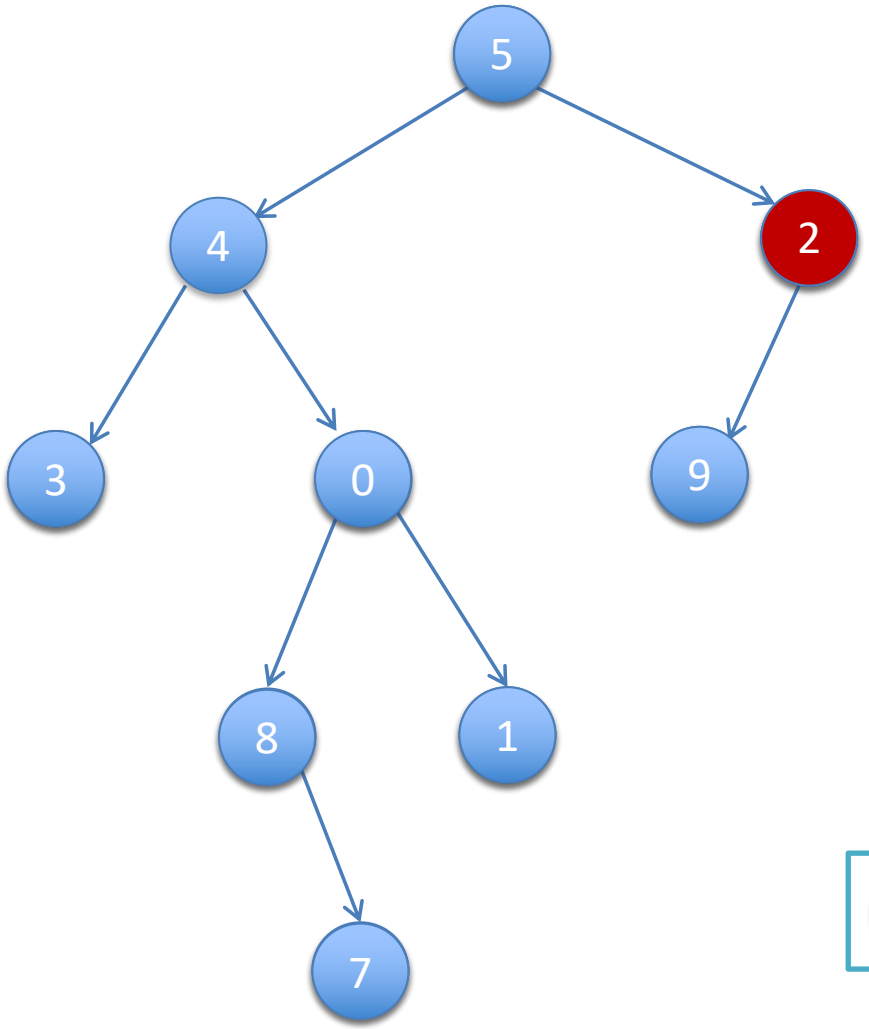


Using a Stack!
Just use a Stack instead of a Queue

A Stack

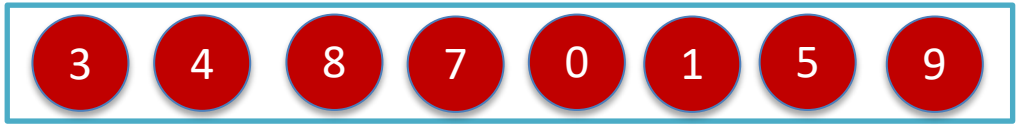
Output

How to do in-order traversal? (without recursion)



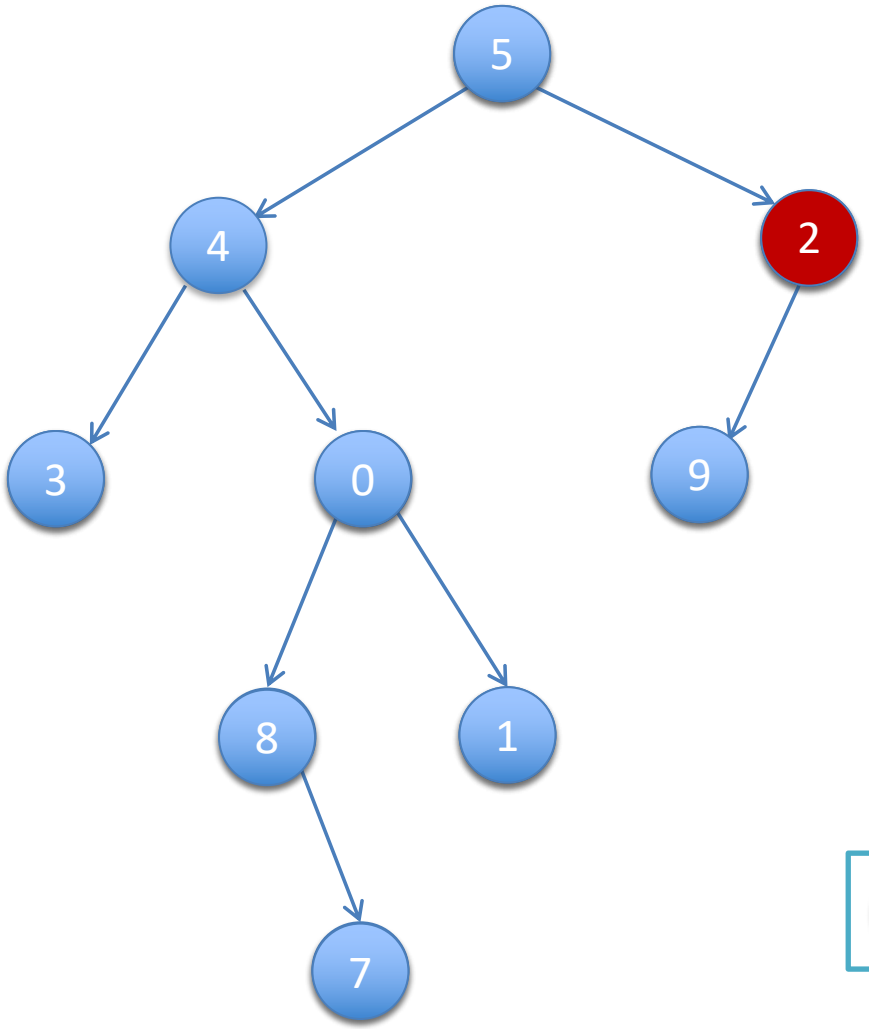
Using a Stack!
Just use a Stack instead of a Queue

A Stack



Output

How to do in-order traversal? (without recursion)



Using a Stack!
Just use a Stack instead of a Queue

A Stack

Output

Level-Order Print in Java

```
void printLevelOrder()
    {
        Queue<BTNode> queue = new LinkedList<BTNode>();
        queue.offer(root);
        while (!queue.isEmpty())
        {

            BTNode currNode = queue.poll();
            System.out.print(currNode.getPayLoad() + " ");

            if (currNode.left != null) {
                queue.offer(currNode.left);
            }

            if (currNode.right != null) {
                queue.offer(currNode.right);
            }
        }
    }
```

What if we change the Queue into a Stack

```
void print__________Order()
    {
        Stack<BTNode> stack = new Stack<BTNode>();
        stack.push(root);
        while (!stack.isEmpty())
        {


        BTNode tempNode = stack.pop();
            System.out.print(tempNode.getPayLoad() + " ");

            /*Enqueue left child */
            if (tempNode.left != null) {
                stack.push(tempNode.left);
            }

            /*Enqueue right child */
            if (tempNode.right != null) {
            stack.push(tempNode.right);
            }
        }
    }
```

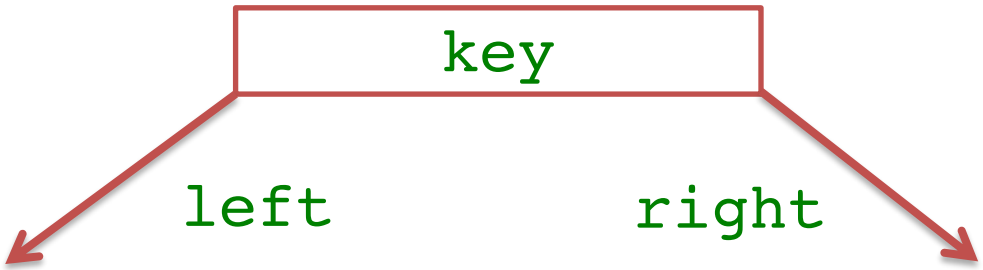
What traversal is this?

Reverse Preorder

A few exercises

A BTNode in Java

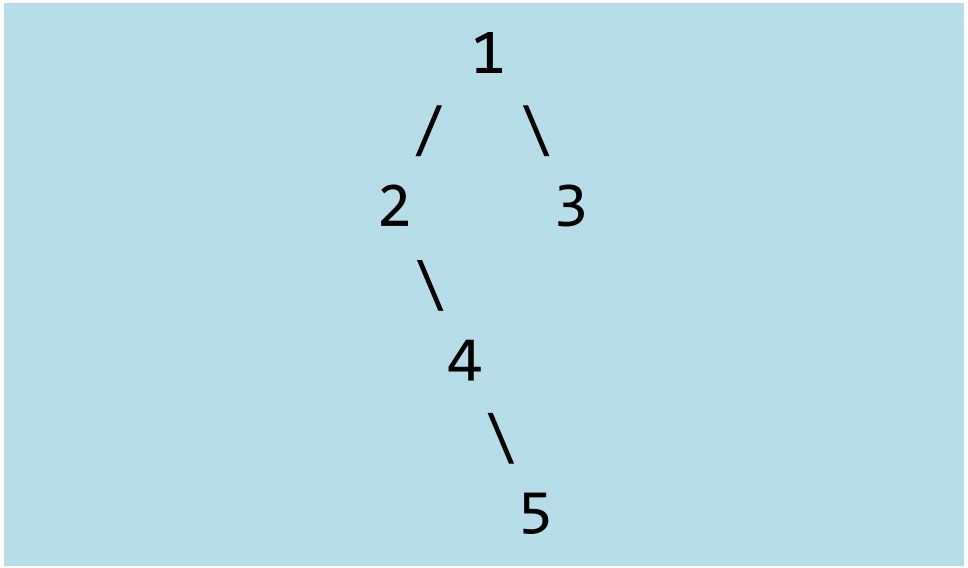
```
public class Node
{
    public int key;
    public Node left, right;

    public Node(int item)
    {
        key = item;
        left = right = null;
    }
}
```



Draw the tree!

```
Node root = new Node(1);
root.left = new Node(2);
root.right = new Node(3);
root.left.right = new Node(4);
root.left.right.right = new Node(5);
```

Tree



inorder

```
public static void inOrder(Node root)
{
        if (root == null) return;
        inOrder(root.left);
        printNode(root);
        inOrder(root.right);
}
```

```
2 4 5 1 3
```

Preorder and postorder

```
public static void preOrder(Node root) {
        if (root == null) return;
        printNode(root);
        preOrder(root.left);
        preOrder(root.right);
    }

    public static void postOrder(Node root) {
        if (root == null) return;
        postOrder(root.left);
        postOrder(root.right);
        printNode(root);
    }
```

```
1 2 4 5 3
5 4 2 3 1
```

Find the # of nodes

```
public static int numNodes(Node root) {
        if (root == null) return 0;
        return 1 + numNodes(root.left) +
                numNodes(root.right);
}
```

Find height of the tree

```
public static int height(Node root) {
        if (root == null) return -1;
        return 1 + Math.max(height(root.left) ,
        height(root.right));

    }
```

Summary

- Various Traversal Technique
  - Pre, In, Post, and Level
  - Recursive and Iterative Solution
  - Runtime
  - Reconstructing the tree from Traversal order
  - A few applications

Acknowledgement

- Douglas Wilhelm Harder.
  - Thanks for making an excellent set of slides for ECE 250 *Algorithms and Data Structures* course

- Prof. Hung Q. Ngo:
  - Thanks for those beautiful slides created for CSC 250 (Data Structures) course at UB.

- Many of these slides are taken from these two sources.