Lecture 19

Wednesday, February 28, 2018 9:36 AM

Goals

- 3 Traversal options
- BST delete-node function

Traversing a tree:

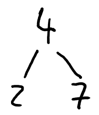
- 3 conventions

Pre-order: root, left, right
4, 2, 7

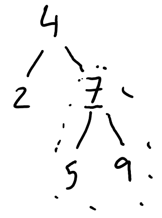
In-order: left, root, right
2, 4, 7

Post-order: left, right, root
2, 7, 4

e.g.



e.g.



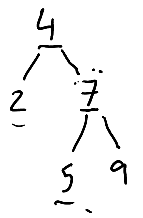
Pre-order: 4, 2, 7, 5, 9

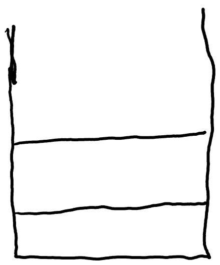
In-order: 2, 4, 5, 7, 9

Post-order: 2, 5, 9, 7, 4

Algorithms:

```
displayPreOrder ( n )
{
    cout << n ->key <<endl;
  ->if (n->LC != null) -
      -> displayPreOrder(n->LC); //a <-
    if (n->RC != null) -
        displayPreOrder(n->RC); //b <-
}  ↓
```





display 4, 2, 7, 5, 9

```
displayInOrder ( n )
{   if (n->LC != null)
      { displayInOrder(n->LC); }
    cout << n ->key <<endl;
```

```
        if (n->RC != null)
         { displayInOrder(n->RC); }
    }
```

Delete() method

- algo for removing a node w/ a specified value from treeee

1. Find node
2. Check for children

(assume Node found not ROOT)

Case 1) Node has no children

```
if (node->LC == NULL  AND  node->RC == NULL)
{
    if (node == node->parent->LC)
        node->parent->LC = NULL;

    else
        node->parent->RC = NULL;
}
```

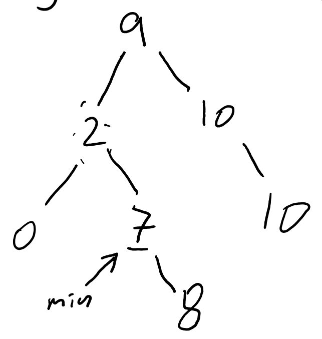
Case 2) two children

```
else if (node->LC != NULL AND node->RC != NULL)
{  // left child (relationship to parent)
   // approach: find the min node in
                right branch and use it
                to replace the deleted node
```

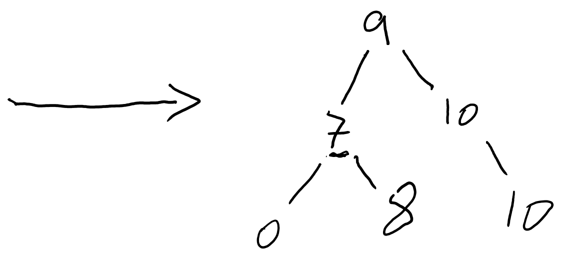
e.g. a

e.g. delete 2 a

e.g.



e.g. delete 2



```
*min = getMin(node);
if (min == node->rightChild)
```