3134 Data Structures in Java

Lecture 9

Feb 14 2007

Shlomo Hershkop

Announcements

- Next Monday: Review for midterm
- Wednesday Feb 21 – Midterm
  - Open book
  - There will be an overfill room
    I will email about it

- Reading:
  - Chapter 4.1-4.3

From last time

- We introduced the idea of a tree data structure
  - Root is top
  - Each node can have zero or more children up to some limit
  - Leaf are nodes with no children
    non-leafs are internal nodes
  - Operations:
    - Insert
    - Find

Trees continued:

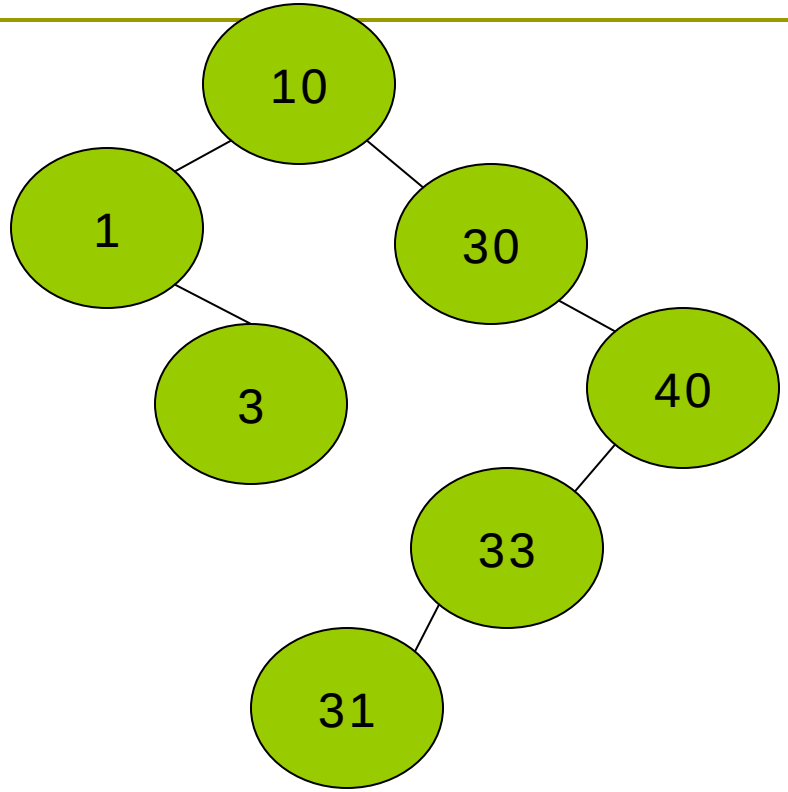
- Path describes how to drive down the tree
- Binary tree: each node can have 0,1,2 children

- ANYONE remember what is a binary search tree??

BST

- Binary search tree is simply a tree, where we force (during insert) each item to be greater than everything in the left child, but less than everything on the right child

- Example:
  - Try this yourself:
  - What does the tree look like after:
  - Insert(10), (30), (40), (1), (3), (33), (31)


10
1
30
3
40
33
31

Code time

- Ok so these pictures of trees are pretty straightforward
- We mentioned tree algorithms usually are recursive
  - Base case usually empty
  - Else what do with each child
- So in BST: HOW WOULD YOU COUNT ALL NODES ON A TREE ????
  - Take a second to scribble out some psuedo code

Recursive solution idea:

- Base case:
  - Either empty tree
  - No children
  - Plus we need to know how many nodes in each side

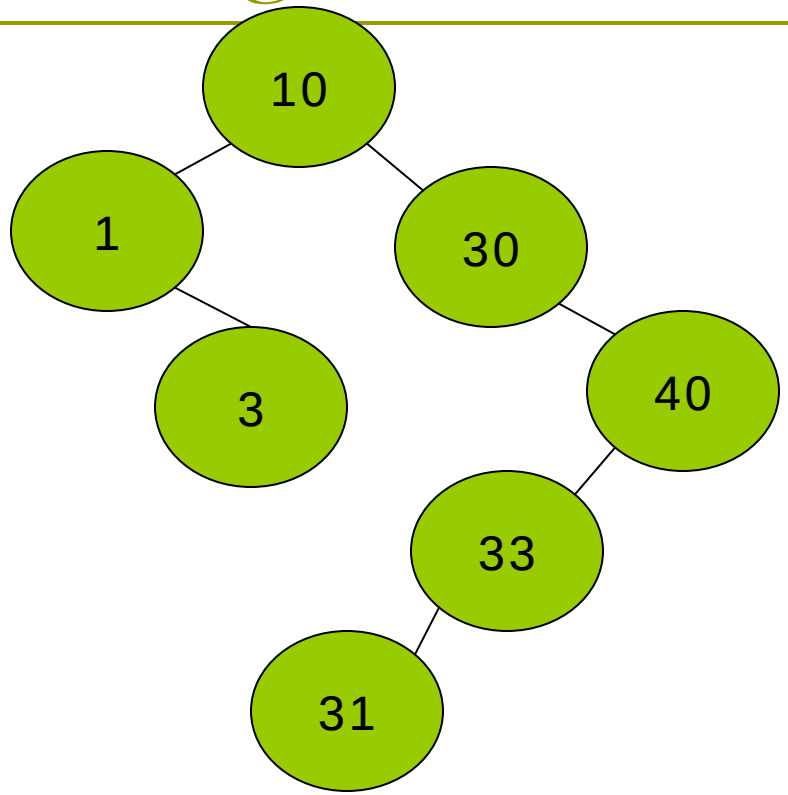
Code:

```
public int size(BinTreeNode Node) {
 if(Node == null)
     return 0;

  return size(Node.getLeft()) + 1 +
  size(Node.getRight());
  }


  //is that it ????
  //any questions ??
```

Seems to be straightforward

- The height of a node is defined as the maximum path (number of jumps) you can take to the lowest leaf node

So what is the height ??



Code

- So again,
- Can you scribble out some pseudo code to calculate the height
- Again what is the base case ?
- What else do we need to deal with?

```
private int height( BinTreeNode t )
    {
        if( t == null )
              return -1;
       else
   return
   1 + Math.max(
   height( t.getLeft() ), height( t.getRight() ) );

}


//why are we taking max ??
//can you explain this to your grandmother ??
```

Here it is again, but without max

```
public int height(BinTreeNode node) {

   if(node == null)
        return -1;

   int lefth = height(node.getLeft()  );
   int righth = height(node.getRight() );

   if(lefth > righth)
        return 1 + lefth;
   else
        return 1 + righth;
```

Printing…

- When we dealt with lists, printing out the list seemed to be straightforward

- Any ideas of how you would print out the tree ??
  - You can not use arrows ☺

Printing a tree

- There are three general ways of printing a tree structure
  - Mainly from top down….

- Inorder
- Preorder
- Postorder

- Lets discuss …

First:

- Inorder
  - inorder(left), root, inorder(right)

- When printing out a node, will first print everything to the left, then yourself and then everything to the right

- Preorder
  - root, preorder(left), preorder(right)
- Postorder
  - postorder(left), postorder(right), root

```
void printinorder(BintreeNode node){
  if(node == null)
      return;

  printinorder(node.getLeft());
  node.print();
  printinorder(node.getRight());
}
```

- find ??
- how does find work ?
- what is the running time ?

helper find function for simple int

```
boolean findh(int n, TreeNode node) {
   if (node == null)
        return false;
   else if (n < node.getItem)
        return findh(n, node.getLeft);
   else if (n > node.getIitem)
        return findh(n, node.getRight);
   else
        return true;
}
```

- how do you find the minimum on the BStree ??
- what about the maximum ??
- Where are they ALWAYS going to be ??

```
BinTreeNode findmax(BinTreeNode node){

   if(node != null)
      while(node.getRight() != null)
        node = node.getRight();

      return node;
      }
```

Adding stuff into the tree

- what would insert look like ??

- BinSearchTree BST = new …
- BST.insert(2);

helper

- Remember that trick with helper functions ??

- We can start the recursive chain by using

  insertHelper(5, root)

```
void insertHelper(int n, BinTreeNode node) {

if(node == null)
   node = new BinTreeNode(n,null,null);

??????


}
```

```
private void insertHelp( AnyType x, BinTreeNode t )
    {
        if( t == null ) {
        {    t = new BinTreeNode( x, null, null );
             return;
       }
       int compareResult = x.compareTo( t.element );

        if( compareResult < 0 )
              insertHelp( x, t.getLeft() );
        else if( compareResult > 0 )
            insertHelp( x, t.getRight() );

       return;
    }
```

- We will cover remove on a tree after midterm

- Any questions ??

Reminder

- Please don't forget to hand in theory homework now
- And hope you submitted the homework programming section electronically…
- If you want to see anything specific reviewed, please drop shlomo an email