

## Discussion 6: Trees

Feb.11 2005

Caijie Zhang

<http://www.cs.ucsb.edu/~caijie>

4.31 Write efficient functions that take only a pointer to the root of a binary trees, T, and compute:

- The number of nodes in T.
- The number of leaves in T.
- The number of full nodes in T.

### **Solution:**

```
a. int num_node ( Node* root )
{
    if ( root == NULL )
        return 0;
    else
        return 1 + NumNode(root->leftChild) + NumNode(root->rightChild);
}

b. int num_leaf ( Node* root )
{
    if ( root == NULL )
        return 0;
    else if ( root->leftChild == NULL && root->rightChild == NULL )
        return 1;
    else
        return num_leaf (root->leftChild) + num_leaf (root->rightChild);
}

c. int num_full (Node* root)
{
    if ( root == NULL )
        return 0;
    else if ( root->leftChild != NULL && root->rightChild != NULL )
        return 1 + num_full (root->leftChild) + num_full (root->rightChild);
    else
        return num_full (root->leftChild) + num_full (root->rightChild);
}
```

4.32 Design a recursive linear-time algorithm that tests whether a binary tree satisfies the search tree order property at every node.

**Solution:**

```
bool isBST ( Node* root, int min, int max )
{
    if ( root == NULL )
        return true;
    else if ( min <= root->key && root->key <= max )
        return isBST(root->leftChild, min, root->key) && isBST( root->rightChild, root->key, max)
    else
        return false;
}
```

4.40 Write a routing to list out the nodes of a binary tree in *level-order*. List the root, then nodes at depth 1, followed by nodes at depth 2, and so on. You must do this in linear time. Prove your time bound.

**Solution:**

```
void WST (Node* root)
{
    queue.push(root);
    while( queue is not empty ){
        node = queue.pop();
        print( node->key );
        if( node->leftChild != NULL )
            queue.push(node->leftChild);
        if( node->rightChild != NULL )
            queue.push(node->rightChild);
    }
}
```