```
public int countNodes() {
	return countNodes(root);
}

private int countNodes(BSTNode<T> t) {
	if (t == null)
		return 0;
	return 1 + countNodes(t.left) + countNodes(t.right);
}

public int totalNodes() {
	return totalNodes(root);
}

private int totalNodes(BSTNode<T> t) {
	if (t == null)
		return 0;
	return t.key + countNodes(t.left) + countNodes(t.right);
}

public int avg() {
	if (root == null)
		return 0;
	else
		return totalNodes() / countNodes();
}

public int countParents() {
	return countParents(root);
}

private int countParents(BSTNode<T> t) {
	if (t == null || (t.left == null && t.right == null))
		return 0;
	return 1 + countParents(t.left) + countParents(t.right);
}

public int countLeaf() {
	return countLeaf(root);
}

private int countLeaf(BSTNode<T> t) {
	if (t == null)
		return 0;
	else if (t.left == null && t.right == null)
		return 1;
	return countLeaf(t.left) + countLeaf(t.right);
}
```

```
public int countOneChild() {
        return countOneChild(root);
}

private int countOneChild(BSTNode<T> t) {
        if (t == null)
                return 0;
        else if ((t.left == null && t.right != null)

                        || (t.left != null && t.right == null))
                return 1 + countOneChild(t.left) + countOneChild(t.right);
        return countOneChild(t.left) + countOneChild(t.right);
}

private int height(BSTNode<T> t) {
        if (t == null)
                return 0;
        return 1 + Math.max(height(t.left), height(t.right));
}

public int countLevel(int level) {
        return countLevel(root, 0, level);
}

private int countLevel(BSTNode<T> t, int l, int level) {
        if (t == null)
                return 0;
        l++;
        if (l == level)
                return 1 + countLevel(t.left, l, level) + countLevel(t.right, l, level);
        else
                return countLevel(t.left, l, level) + countLevel(t.right, l, level);
}
```