# Side Talk: Implementation of the AVL Tree

Tony Gong

ITEE
University of Queensland

In the same spirit as our discussions about the implementations of the linked list and the binary heap, today we will look at the implementation of the AVL Tree.

Given that we have already looked at the class definitions of two pointer-based data structures, hopefully you will be used to the ideas by now and the material presented here will come across as intuitive.

```
class TreeNode {
    int key;
    int leftSubtreeHeight;
    int rightSubtreeHeight;
    [Type of TreeNode Pointer] leftChild;
    [Type of TreeNode Pointer] rightChild;
    [Type of TreeNode Pointer] parent;

    [Type of TreeNode Pointer] getLeftChild();
    [Type of TreeNode Pointer] getRightChild();
    [Type of TreeNode Pointer] getParent();
    bool isLeaf();
};
```

The above models a node in our AVL tree. Notice that the only difference of the above when compared to the `HeapNode` class used for a binary heap is that we track the heights of the left and right subtrees.

```
class AVLTree {
    [Type of TreeNode Pointer] root;

    int predecessor(int k);
    int successor(int k);
    insert(int k);
    delete(int k);

    rebalance([Type of TreeNode Pointer] node);
};
```

The above is the class definition of an AVL tree. The dynamic `insert` and `delete` operations are supported, as well as the `predecessor` and `successor` queries. I have chosen to have a `rebalance` function that takes a node as an argument and will traverse up the tree and rebalance as necessary. This is to avoid code duplication between `insert` and `delete`.

And that is all! Hopefully we have provided enough guidance in showing one possible way of implementing the AVL tree data structure.

Implementation is more or less a question of organisation, and note that there is often no right or wrong and the way presented here is not the only way of structuring the code.

For example, the `rebalance` function operates on a node in our tree, hence we could possibly move it out of the `AVLTree` class and into the `TreeNode` class instead. I have elected to leave the `TreeNode` class general instead of specific to the AVL Tree, but you may not care to do so. When faced with design choices, exercise your best judgement!