Balanced Binary Search Trees

- Pathological BST
  - Insert nodes from ordered list
  - Search: \( O(\_\_\_\_\_\_) ? \)
- The Balanced Tree
  - Binary Tree is balanced if height of left and right subtree differ by no more than one, recursively for all nodes.
  - (Height of empty tree is -1)
- Examples

Balanced Binary Search Trees

- Keeping BSTrees Balanced
  - Keeps find, insert, delete \( O(\log(N)) \) worst case.
  - Pay small extra amount at each insertion to keep it balanced
- Several Well-known Systems Exist for This
  - AVL Trees
  - Red-Black Trees
  - ...
- Will “look at” AVL Trees

AVL Trees

- AVL Trees
  - Adelson-Velskii and Landis
  - Discovered ways to keep BSTrees Balanced
- Insertions
  - Insert into BST in normal way
  - If tree no longer balanced, perform “rotation(s)”
  - Rotations restore the tree balance

AVL Trees

- Single Rotation
  - An insertion into the left subtree of the left child of tree
  - Adapted from Weiss, pp 567-568
  /** Used if insert has caused loss of balance at k2
   * (Also used as part of double rotation operations)
   * @return root of adjusted tree
   */
   TNode rotateWithLeftChild(TNode k2){
     TNode k1 = k2.left;
     k2.left = k1.right;
     k1.right = k2;
     return k1;
   }
AVL Trees

**Single Rotation**

Also: mirror image

```cpp
TNode rotateWithRightChild(TNode k2)
{
    TNode k1 = k2.right;
    k2.right = k1.left;
    k1.left = k2;
    return k1;
}
```

AVL Trees

**Double Rotation**

An insertion into the right subtree of the left child of tree

Adapted from Weiss, p 57

```cpp
TNode doubleRotateWithLeftChild(TNode k3)
{
    k3.left = rotateWithRightChild(k3.left);
    return rotateWithLeftChild(k3);
}
```
**AVL Tree**

- **Double Rotation**

  - An insertion into the right subtree of the left child of tree
  - Adapted from Weiss, p 571

  ```
  /** Used after insertion into right subtree, k2, of right child, k1, of k3 (if it has caused loss of balance)
   * @return root of adjusted tree
   */
  TNode doubleRotateWithRightChild(TNode k3) {
    k3.right = rotateWithLeftChild(k3.right);
    return rotateWithRightChild(k3);
  }
  ```

**AVL Trees**

- **Double Rotation**

  - Deletions can also cause imbalance
  - Use similar rotations to restore balance
  - Big Oh?