Balanced Binary Search Trees

- Pathological BST
  - Insert nodes from ordered list: $O(\_\_)$?
  - Subsequent 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 (and deletion) 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 a “rotation”
  - Rotations restore balance to the tree

AVL Trees

- Single Rotation
  - An insertion into the left subtree of the left child of tree
  - Adapted from Weiss, pp 567-568
  /** Used if insertion has caused loss of balance
   * (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**

![Single Rotation Diagram]

```c
/** @return root of adjusted tree */
TNode rotateWithRightChild(TNode k2)
{
    TNode k1 = k2.right;
    k2.right = k1.left;
    k1.left = k2;
    return k1;
}
```

- **Double Rotation**

  - Mirror image case

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

- Double Rotation

![Diagram of AVL Tree with Double Rotation]

AVL Trees

- Double Rotation

![Diagram of AVL Trees with Double Rotation]

//** Mirror Image
* @return root of adjusted tree
 */
TNode doubleRotateWithRightChild(TNode k3) {
    k3.right = rotateWithLeftChild(k3.right);
    return rotateWithRightChild(k3);
}

AVL Tree

- Double Rotation

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

AVL Trees

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