AVL Tree

Class 22

BST Implementation

```
class Tree_Node
{
  Object data;
  Tree_node* left_child;
  Tree_node* right_child;
};
```

- the data has a key of a comparable type (i.e., $<$ is defined on data.key)
- the data in a node is greater than any value in its left subtree
- the data in a node is less than any value in its right subtree

- simplifying assumption: there are no duplicate values in the tree

Find

```
bool find(const Comparable& key, Tree_Node* root) const
{
  if (root == nullptr)
  {
    return false;
  }
  if (key < root->data.key)
  {
    return find(key, root->left);
  }
  if (key > root->data.key)
  {
    return find(key, root->right);
  }
 return true;
}
```

AVL Tree

- in 1962 Adelson-Velskii and Landis proposed a height-balanced tree
- use a BST with all its requirements
- additionally, insist that the height of any node's left and right subtrees can differ by at most 1
- more formally

1. an empty BST is height-balanced
2. a non-empty BST with root T and children T_L and T_R is height-balanced iff
   2.1 $|\text{height}(T_L) - \text{height}(T_R)| < 2$, and
   2.2 T_L and T_R are height-balanced recursively

AVL Tree

- height-balancing guarantees that for *find*, $T(n) \in O(\lg n)$
- but how to accomplish height-balancing?

- insert and delete must be modified to maintain the balance
- each tree node must have a field for balance factor

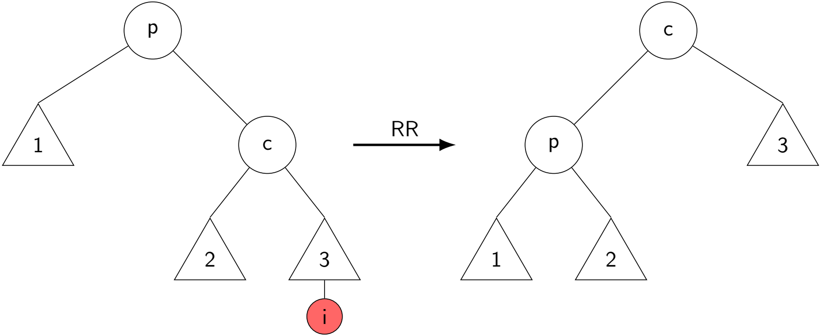
```
class AVL_node
{
  Object data;
  int balance;
  AVL_node* left_child;
  AVL_node* right_child;
};
```

AVL Implementation

- maintain height-balanced BST
- every insert or remove can potentially unbalance the tree
- an unbalanced tree requires a rotation to rebalance
- there are 4 types of rotations
    1. RR
    2. LL
    3. LR
    4. RL

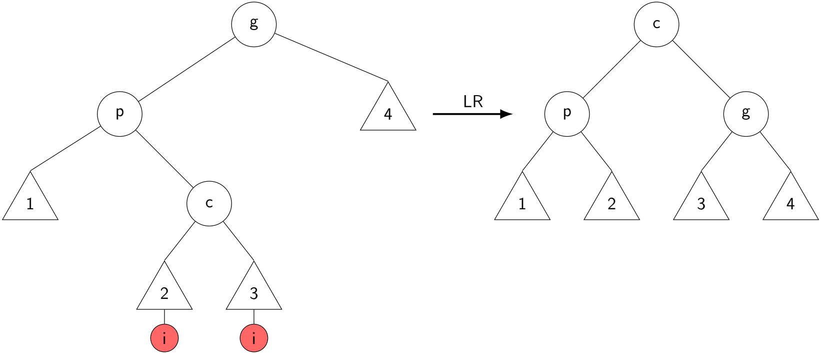
example

RR



- p is the lowest unbalanced node
- c is p's right child
- tree 3 contains the insert that caused the tree to become unbalanced, depicted by i
- LL is the mirror image of this

LR



- g is the lowest unbalanced node
- p is g's left child
- c is p's right child
- the insert that caused the tree to become unbalanced is either in 2 or 3, or is c itself