AVL Trees

Section 4.4
AVL Tree

- A balanced tree
- Ensures $O(\log n)$ running time for search, insert, and delete
- A simple and relaxed definition for balance
  $[\log n] \leq h \leq [\log n]$: Too restrictive
Balanced Tree

Height(null) = -1

Height(root->left) = Height(root->right)

Too weak
Balanced Tree

Height(null) = -1

Height(node->left) = Height(node->right)

Could be impossible to satisfy
AVL Balance Condition

\[ |\text{Height(node } \rightarrow \text{ left)} - \text{Height(node } \rightarrow \text{ right)}| \leq 1 \]

Height(null) = -1
AVL Example

Is this an AVL Tree? Yes
AVL Example

Is this an AVL Tree? No
AVL Example

Is this an AVL Tree? Yes
Balancing an AVL Tree

- For simplicity, we assume that we keep the height of each subtree at its root
- An imbalance can occur as a result of an insertion or deletion
- To balance an AVL tree, we carry out a rotation operation
Insertion

- Call BST.insert
- Update the height as you climb up to the root
- After each height update, check for an AVL tree violation and fix using rotation
Violation after Insertion

Cases 1 and 4 are symmetric
Cases 2 and 3 are symmetric

Cases 1, 2, 3, and 4 are symmetric.
Case 1 – Single Rotation

Status upon insertion in X
k₂ is in violation

Is this a BST? Yes
Is this an AVL Tree? Yes
Case 2 – Single Rotation?

Status upon insertion in Y

Is this a BST? Yes
Is this an AVL Tree? No
Case 2 – Double Rotation

Violating Node