How Tall Is An “Average” BST?

BST after $\Theta(N^2)$ insert/remove pairs

But deleted node is always replaced with a node from right subtree

$\Rightarrow$ deletions favor making left subtree deeper than the right!

“Balance” in a BST

- Issue: tree may be imbalanced
  - Deletions—favor making left subtree deeper
  - Insertions—if data is presorted, tree has only nodes with no left children

- Add an extra structural condition—balance
  - No node is allowed to get too deep
  - Updates take longer on average

- How to balance?
  - Left and right subtrees should have same height
  - *Every* node must have left and right subtree of same height
AVL Trees

- For every node, height of left and right subtrees can differ by at most 1
  - Smallest AVL tree of height 9

AVL Trees: Operations

- All lookup-based operations exactly the same as BST
  - Upper bound on height: \(1.44 \log(N+2) - 1.328\)
  - Typical height \(\sim \log N\)

- Updates must preserve the balance property, though
  - insert
  - remove
**AVL Trees: insert**

- **Strategy**: do a regular insert first, then correct imbalance.
- **Only nodes on path from insertion point to root might have balance destroyed**
  - Rebalance deepest such node (A)
  - Rebalancing only A can correct balance at all other node.
- **A has at most two children**
  - Imbalance $\Rightarrow$ heights of two subtrees differ by two
  - **Cases**:
    - Insertion into left subtree of left child of A
    - Insertion into right subtree of right child of A
    - Insertion into right subtree of left child of A
    - Insertion into left subtree of right child of A

**AVL: insert into left subtree of left child**

- **Heights have to be ordered as shown**:
  - Left subtree X is two levels deeper than right subtree Z
    - Else $k_2$ would not be imbalanced
  - Y is 1 level less than X
    - Else $k_2$ would be imbalanced even before the insertion
  - Y can not be same level as Z
    - Else $k_1$ would be the deepest node with imbalance
- **Single rotation** – move X up a level and Z down a level
AVL: Single Rotation Example

Again, height have to be ordered as shown

- Single rotation works again
AVL: insert into right subtree of left child

- Heights have to be ordered as shown:
  - Subtree Y is two levels deeper than subtree Z
    - Else $k_2$ would not be imbalanced
  - X is one level less than Y
    - Else $k_2$ would be imbalanced even before the insertion
  - X can not be same level as Z
    - Else $k_1$ would be the deepest node with imbalance
- Single rotation – doesn’t work!

AVL insert: Left-Right Double Rotation

- Y is guaranteed to be non-empty
  - So tree can be viewed as four subtrees connected by
    three nodes
- Exactly one of tree B or C is two levels deeper than D
- New root can not be $k_1$ (single rotation) – so use $k_2$
- New tree satisfies AVL property
AVL insert: Right-Left Double Rotation