Insert (Key x, Value v)
- find x in tree
- if found ⇒ error! duplicate key
- else: create new node where we "fell out"

Replacement Node?

3. x has two children
Find replacement node
▷, copy to x, and then delete ▷

3 cases:
(1) x is a leaf
(2) x has single child

Delete (Key x)
- find x
- if not found ⇒ error
- else: remove this node + restore BST structure
How?
### Binary Search Trees III

- **Deletion**
- **Analysis**
- **Java**

**Java Implementation:**
- Parameterize `Key` + Value types: `extends Comparable`
- `class BinSearchTree<K,V>`
- `BSTNode` - inner class
- Private data: `BSTNode root`
- `insert, delete, find`: local
- provide public `fns` `insert, delete, find`

But height can vary from $O(\log n)$ to $O(n)$...

**Expected case is good**

**Thm:** If $n$ keys are inserted in random order, expected height is $O(\log n)$.

**Analysis:**
All operations (find, insert, delete) run in $O(h)$ time, where $h = \text{tree's height}$
AVL Trees I

- Basic defs
- Height props
- Rotations

Theorem: An AVL tree of height \( h \) has at least \( F_{h+3} - 1 \) nodes.

Conjecture: Min no. of nodes in AVL tree of height \( h \) is \( F_{h+3} - 1 \)

Proof: (Induct. on \( h \))

- \( h = 0 \): \( n(0) = 1 = F_3 - 1 \)
- \( h = 1 \): \( n(1) = 2 = F_4 - 1 \)

\[
n(h) = \left\lfloor \frac{F_{h+2}}{2} \right\rfloor + n(h-1) + n(h-2)
\]

I. H. \( \Rightarrow \)

\[
n(h) = \left\lfloor \frac{F_{h+2}}{2} \right\rfloor + n(h-1) + F_{h+2} - 1
\]

Corollary: An AVL tree with \( n \) nodes has height \( O(\log n) \).

Proof: Fact: \( F_n \approx \phi^n / \sqrt{5} \) where \( \phi = (1 + \sqrt{5}) / 2 \) "Golden ratio".

\[
n \geq \phi^{h+3} / 2 \Rightarrow h \leq \log_\phi n + c
\]

\[
\Rightarrow h \leq \log_2 n / \log_2 \phi = O(\log n)
\]

AVL Height Balance

- for each node \( v \), the heights of its subtrees differ by \( \leq 1 \)

AVL tree: A binary search tree that satisfies this condition.

Does this imply \( O(\log n) \) height?

Worst cases:

- Height: \( h = 0, 1, 2, \ldots, h \)
- Nodes: \( n = 1, 2, 3, 5, 8, 13, \ldots \)

Recall: \( F_0 = 0, F_1 = 1, F_h = F_{h-1} + F_{h-2} \)

- AVL Height Balance
  - This is an AVL tree
  - Not an AVL tree

AVL Trees I

- Basic defs
- Height props
- Rotations

Theorem: An AVL tree of height \( h \) has at least \( F_{h+3} - 1 \) nodes.

Consider the minimum number of nodes in an AVL tree of height \( h \).

AVL tree: A binary search tree where the heights of the subtrees differ by \( \leq 1 \).

AVL Height Balance

- for each node \( v \), the heights of its subtrees differ by \( \leq 1 \).

AVL tree: A binary search tree that satisfies this condition.

How to maintain the AVL property?

- Root rotation
- Left rotation

- Rotations
AVL Trees II

- Double rotations: left-right, right-left
- Insertion

AVL Tree:
- AVL Node: Same as BSTNode (from Lect 4) but add: int height

Utilities:
- int height (AVLNode p) return \{ p == null \rightarrow -1, \text{otherwise} \rightarrow p.\text{height} \\
- void updateheight (AVLNode p) p.\text{height} = 1 + \max (\text{height}(p.\text{left}), \text{height}(p.\text{right}))
- int balanceFactor (AVLNode p) return \text{height}(p.\text{right}) - \text{height}(p.\text{left})

AVLNode rebalance (AVLNode p)
if (p==null) return p
if (balanceFactor(p) < -1)
    if (ht.s.left.left) = ht.s.left.right)
        p = rotateRight (p)
    else p = rotateLeftRight (p)
else if (balanceFactor(p) > +1)
    [...symmetrical]
    updateHeight(p); return p

AVLNode insert (key x, Value v, AVLNode p)
if (p==null) p = new AVLNode (x, v)
else if (x < p.key) p.left = insert (x, v, p.left)
else if (x > p.key) p.right = insert (x, v, p.right)
else throw Error - Duplicate!
return rebalance (p)

Find: Same as BST.
Insert: Same as BST but as we "back out" rebalance

How to rebalance? Bal = -2
- Left-right heavy
- Left-left heavy

Double rotations: left-right, right-left
Cases:
- Balance factor -2
  - Left-left heavy
  - Left-right heavy

Deletion: Basic plan
- Apply standard BST deletion
  - Find key to delete
  - Find replacement node
  - Copy contents
  - Delete replacement
  - Rebalance

Example 4:
- delete(7)

Example 3:
- delete(7)

AVL Trees III
- Deletion
- Examples

AVLNode delete (Key x, AVLNode p)
: same as BST delete
: return rebalance(p)

Examples:
- Insert(5)
- Insert(3)
- Delete(7)
Announcements - 9/22

- HW 1 - Due in a week, Thu 9/29, 11:59pm
  - Submit by Gradescope
  - Challenge problem coming
- Midterm dates? Soon
- Prog Assign 1? Soon