Dictionary operations:

- **Find**: straight forward
- **Insert**: find leaf node where key “belongs” + add it (may split)
- **Delete**: find/replacement/merge or adopt

**Implementation:**

```java
class TwoThreeNode {
  int nChildren;
  int[] children;
  Key key[];
}
```

**Example:**

#### Insertion example:

- `insert(6)`

```
        4
       /|
      2 8
     /  |
    1 3 12
   /    |
  5 7 9 14
```

#### Delete Example:

- `delete(5)`

```
        4:8
       /|
      2 6
     /  |
    1 3 10
```

**Deletion remedy:**

- Have a 3-node neighboring sibling → adopt
- O.w.: Merge with either sibling + steal key from parent

---

**2-3 Trees II**
**AVL Height Balance**

For each node $v$, the heights of its subtrees differ by at most 1.

**AVL Tree** A binary search tree that satisfies this condition.

**AVL Trees I**
- Basic defs
- Height props
- Rotations

**Theorem:** An AVL tree of height $h$ has at least $F_{h+3}$ nodes, where $F_n$ is the $n$th Fibonacci number.

**Proof:** (Induct. on $h$)

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

$n(h) = \sum_{i=0}^{h-1} n(i) + n(h-1) + n(h-2)$

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

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

$n \geq F_{h+3} = c \cdot \varphi^h \Rightarrow h \leq \log \varphi n + c$

$\Rightarrow h \leq \log \varphi n / \log \varphi = O(\log n)$

I. H. $\Rightarrow n(h) = \sum_{i=0}^{h-1} (F_{i+3})^2 = F_{h+3}^2$
AVL Trees II

double rotations: left-right LR

right-left RL

AVLNode rebalance (AVLNode p)

if (p == null) return p

if (balanceFactor(p) < -1)
  if (height(p.left.left) >= height(p.left.right))
    p = rotateRight(p)
  else p = rotateLeftRight(p)

else if (balanceFactor(p) > 1)
  if (height(p.right.right) >= height(p.right.left))
    p = rotateLeft(p)
  else p = rotateRightLeft(p)

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)

AVL Tree:

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

Utilities:

int height (AVLNode p)
return { p == null → -1
          ow. → p.height
  }

void updateHeight (AVLNode p)
  p.height = 1 + max (height(p.left),
  height(p.right))

int balanceFactor (AVLNode p)
  return height(p.right) -
           height(p.left)

BSTNode rotate LeftRight (BSTNode p)
  p.left = rotateLeft (p.left)
  return rotateRight (p)

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

How to rebalance? Bal = -2

Left-left heavy:

Left-right heavy:

Double rotations:
left-right LR
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)
**Node types:**
- **2-node:**
  - 1 key
  - 2 children

- **3-node:**
  - 2 keys
  - 3 children

- **AVL:**
  - Height balanced
  - Binary

- **2-3 tree:**
  - Height exact

**Recap:**
- **Identical heights:** D-tree
- **Variable width:** Red-black

**Def:** A 2-3 tree of height $h$ is either:
- Empty ($h=-1$)
- A 2-node root and two subtrees, each 2-3 tree of height $h-1$
- A 3-node root and three subtrees... height $h-1$

**Thm:** A 2-3 tree of $n$ nodes has height $O(\log n)$
Roughly: $\log_3 n \leq h \leq \log_2 n$

**Conceptual tool:**
- We'll allow 1-nodes
- 4-nodes temporary

**How to maintain balance?**
- **Split**
- **Merge**
- **Adoption (key rotation)**

**Example:**
- 2-3 tree of height 2

**Adoption (key rotation):**
- $2 + 2 = 4$

**Merge:**
- $1 + 2 / 2 + 1 \rightarrow 3$

**Split:**
- $4 \rightarrow 2 + 2$

**Example:**
- Conceptual tool: 2-node

**Example:**
- 3-node root and two subtrees, each 2-3 tree of height $h-1$
Announcements - 9/27
- HW1: Due this Thu, 11:59 pm
  - don't wait until last second
  - do a practice scan/submit
  - may deduct pts for poor contrast

- Exam dates:
  Midterm 1: Thu, Oct 20 - 1
  Midterm 2: Thu, Nov 17 - 1
  - In class
  - Closed book/closed notes
  - Allowed "cheat sheet" - front + back

- Prog Assignment 1: Leftist Heap
  - insert, extract-min, merge,...
  - new: split
  - new: split
  - 10/80 pts
  - 137

- 19
- 26
- 42
- 48
- 67
- 94

\[ h_1 = h_2 \text{split}(x) \]
\[ h_2 = \text{merge}(v, u) \]
\[ v = \text{merge}(v, u) \]
\[ h_1 = \text{merge}(v, u) \]
\[ v_1 = \text{merge}(v, u) \]
\[ h_2 = \text{merge}(v, u) \]
\[ h_1 = \text{merge}(v, u) \]
\[ h_2 = \text{merge}(v, u) \]
\[ h_1 = \text{merge}(v, u) \]
\[ h_2 = \text{merge}(v, u) \]
\[ h_1 = \text{merge}(v, u) \]
\[ h_2 = \text{merge}(v, u) \]