Node types:
- 2-Node: 1 key, 2 children
- 3-Node: 2 keys, 3 children

Recap:
- AVL: Height balanced Binary
- 2-3 tree: Height exact
- Variable Width

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 $h-1$

Example:
- 2-3 tree of height 2

2-3 Trees

Thm: A 2-3 tree of $n$ nodes has height $O(\log n)$

Roughly: $\log_3 n \leq h \leq \log_2 n$

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

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

- Insert(6)

```
        4
       /|
      2 5
     / \
    3 7
   /   \
  1 8
``` 

**Dictionary operations:**

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

**Implementation:**

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

**2-3 Trees II**

**Delete Example:**

- Delete(5)

```
        4
       /|
      2 5
     / \
    3 7
   /   \
  1 8
``` 

**Deletion remedy:**

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

**Example (continued)**

```
        4
       /|
      2 5
     / \
    3 7
   /   \
  1 8
```