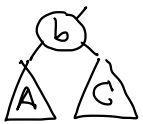
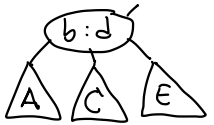
2-3 Trees I

Node types:

2-Node
1 key
2 children



3-Node
2 keys
3 children



Identical heights

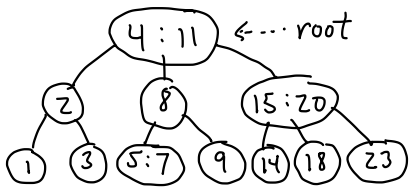
Recap:

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

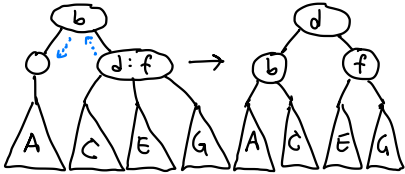


Thm: A 2-3 tree of n nodes has height $O(\log n)$

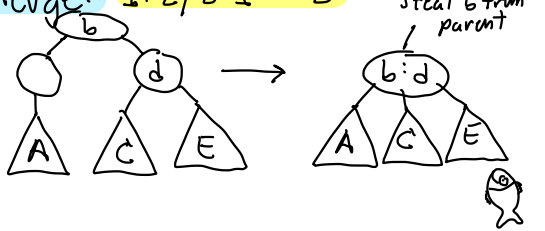
Roughly: $\log_3 n \leq h \leq \log_2 n$

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

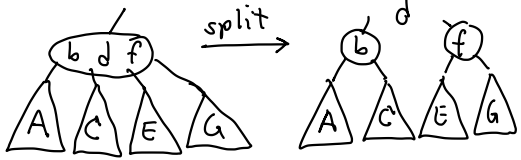
Adoption (Key-Rotation)
1+3 = 2+2



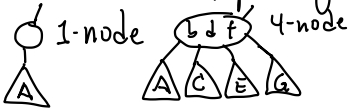
Merge: 1+2/2+1 → 3



Split: 4 → 2+2

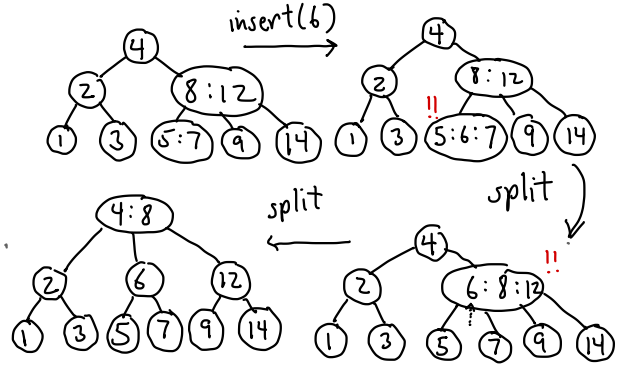


Conceptual tool:
We'll allow 1-nodes + 4-nodes temporary



2-3 Trees II

Insertion example:



Dictionary operations:

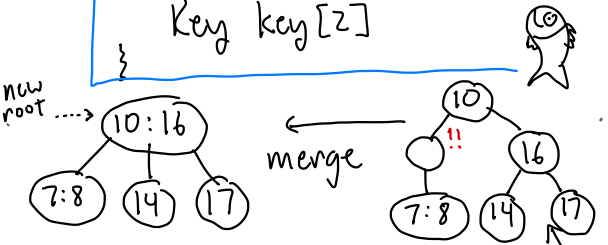
Find - straightforward

Insert - find leaf node where key "belongs" + add it (may split)

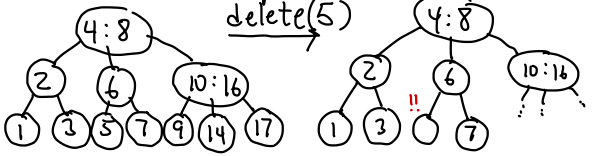
Delete - find / replacement / merge or adopt

Implementation?

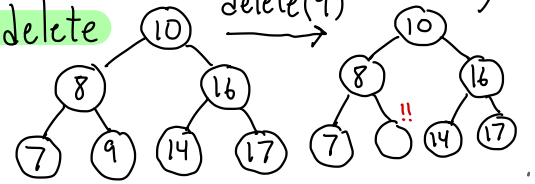
```
class TwoThree Node {
    int nChildren
    TwoThree Node children[3]
    Key key[2]
}
```



Delete Example:



Another delete example:



Deletion remedy:

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

Example (continued)