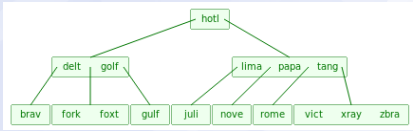
The (2, 4) tree

- A form of balanced search tree
  - comparable to Splay tree, AVL tree, Treap
- Internal nodes have 2 – 4 children
  - "Binary/Ternary/Quaternary Search Tree"
- All leaves have the same depth.
- BST property needs a minor little tweak.
- Insertion / Deletion radically different.

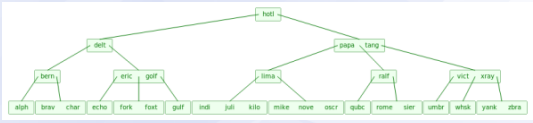
(2, 4) Example



- Internal nodes have 2, 3, or 4 children.
  - Therefore: 1, 2, or 3 keys!
- Leaves at same depth.
- Keys in node bound keys in each subtree.

2

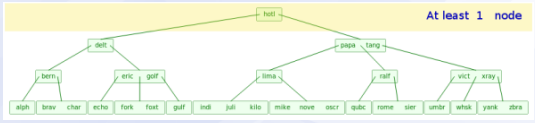
(2, 4) tree has height of $O(\log n)$



- Each internal node has at least 2 children.
- Tree is "at least binary."

3

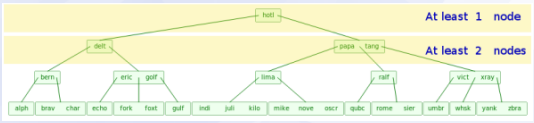
(2, 4) tree has height of $O(\log n)$



- Each internal node has at least 2 children.
- Tree is "at least binary."

4

(2, 4) tree has height of $O(\log n)$



- Each internal node has at least 2 children.
- Tree is "at least binary."

5

(2, 4) tree has height of $O(\log n)$

- Each internal node has at least 2 children.
- Tree is "at least binary."

6

(2, 4) tree has height of $O(\log n)$

- Each internal node has at least 2 children.
- Tree is "at least binary."
- # nodes at least doubles at each depth.
- # nodes $n = \Omega(2^{\text{height}})$, so height = $O(\lg n)$.

7

(2, 4) Insertion, Bottom-Up

- Find your target leaf, add new key.
  - You may temporarily overload with 4 keys.
- If the leaf has 1, 2, or 3 keys, you are done.
- If you overload a node, *fission* occurs:
  - An *interior* key pops out, sticks into parent.
  - Overloaded node splits in half: becomes two nodes, storing the leftover keys.
  - Now the parent might be overloaded! Recurse. *Possibly create a new root!*

8

(2, 4) Deletion, Bottom-Up

- Find the key you want to delete.
- Internal node? Then overwrite key with predecessor or successor, which must be in a leaf. (Same idea we used for regular BST.)
- Delete from a leaf. Does it have too few keys?
  - Try to *transfer* a key from a rich sibling, via the parent. (rich == has ≥2 keys.)
  - If siblings are poor, *fusion* occurs: two nodes merge, absorbing key from parent.
  - Parent could become empty! Recurse.

9

(2, 4) Tree Summary

- Balanced search tree, guaranteed logarithmic height.
  - An efficient dictionary operations.
  - Nodes are variable sized: 1, 2, 3 keys.
- Tree grows taller or shorter from its top!
  - Insertion and deletion can cause node fusion and fission.
- Credit shared by Hopcroft (1970) and Bayer (1972).

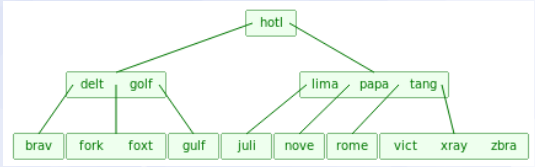
10

Red-Black Trees

- Balanced Binary Search Tree
- Nodes are ascribed a color, *red* or *black*.
- It's like a *simulation* of a (2, 4) tree:
  - (2, 4) nodes with 2 keys simulated by a black node with one red child.
  - (2, 4) nodes with 3 keys simulated by a black node with two red children.
  - Obeys all (2, 4) tree rules.

11

(2, 4) tree disguised as red-black



12

B and B+ Trees

- Abstraction of (2, 4) trees: let the nodes hold more keys. *MANY more keys.*
- Example: a "(500, 1000) tree"
  - All leaves still at the same depth.
  - Internal nodes have 500 – 1000 children.
- (except for the root)
- Nobody calls them that, however.
  - A (k/2, k) tree is a B-Tree of order k.
  - So, a (2, 4) tree is a B-Tree of order four.

13

B and B+ Trees

- Insertion and Deletion much like (2, 4) tree
  - Still do fission, transfer, fusion.
  - Tree grows taller or shorter from the root.
- Height of tree is still logarithmic, but the logarithm base changes:
  - Suppose you have a B-Tree of order 1000.
  - # nodes $n = \Omega(500^h)$, height $h = O(\log_{500} n)$.
  - Tree is VERY, VERY short and wide.

14

B and B+ Trees: why why why??

- Motivation: these work great as search trees in external memory (i.e., disks).
  - External memory access is 100,000 to 1,000,000 times slower than internal.
  - Basic unit of input/output is an entire block.
  - Strategy: make 1 node = 1 block.
  - VERY short tree = VERY few # block reads to reach the leaves. That means *faster*.
  - This is one of those few times when we can't afford to ignore a constant factor.

15

B-Tree versus B+ Tree

- The B+ Tree is a variation, so much more popular than the original B-Tree that some authors confuse the names.
- B+ stores all records (all keys with all satellite data) in *leaves*.
- B+ restricts internal nodes to store *just keys*, and just *some* keys – as guideposts.
  - Thus the insert & delete operations differ.
  - More keys per node = bigger order = faster

16

B-Tree and B+ Tree Summary

- A search tree designed for hard disks.
- Bayer and McCreight (1972). Still popular.
- Further reading: Shaffer 11.6, CLRS 18.

- Learning objectives from slide deck:
  - Be able to insert and delete from a (2, 4) tree. (Practice using Galles's website.)
  - Know relationship btwn. (2,4), red-black.
  - Know motivation for high-order B+ trees.

17