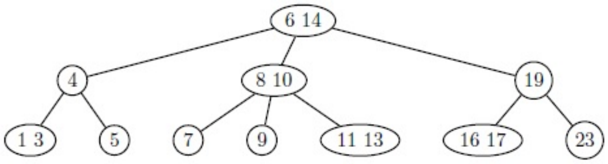
CSCI 104L Lecture 27: 2-3 Trees

2-3 Trees are another form of balanced BST. They come at a cost: the tree is no longer binary.

- In a 2-3 tree, every node has either 1 or 2 values.
- If a node has 1 value and is not a leaf, then it has 2 children, and we call it a 2-node.
- If a node has 2 values and is not a leaf, then it has 3 children, and we call it a 3-node.
- All leaves must be at the same level.
- For all 2-nodes, the binary search tree property holds.
- For all 3-nodes with keys $k_1 < k_2$, and subtrees T_1, T_2, T_3, the keys in T_1 are $\leq k_1$, the keys in T_2 are $\geq k_1$ and $\leq k_2$, and the keys in T_3 are $\geq k_2$.



A 2-3 tree is just a **3-ary search tree**.

Insert

To insert a value, search for it in the tree. If the leaf has only one value, insert the new value. You're done.

If the leaf has two values, insert it anyway (it now invalidly has 3 values, and operates as a "4-node" with 3 values and 4 children).

Move the middle value up. If the parent had one value (and now has two), you now have a valid 2-3 tree. You're done.

If the parent had two values (and now has three), keep moving the middle node up. It may eventually become your new root node. You can think of inserting a value as unzipping the tree. As the middle value propogates up the tree, you break it apart until the whole process stops.

Delete

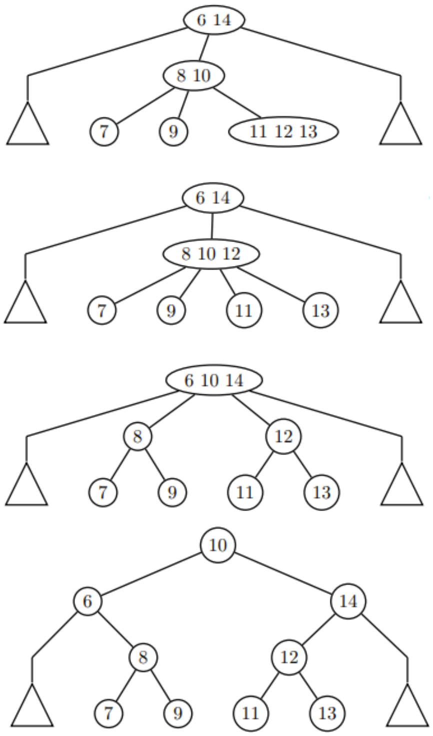
To delete a value, search for it in the tree. If it is not in a leaf node, then swap it with its predecessor or successor so that it is guaranteed to be at a leaf node.

If it is now in a leaf node with two values, delete it.

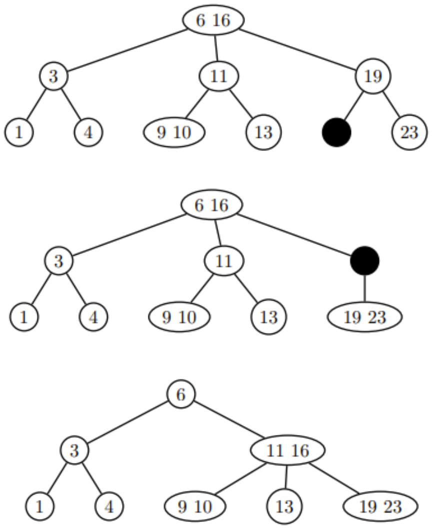
If the leaf has one value, delete it anyway (it now invalidly has 0 values, and operates as a "1-node" with 0 values and 1 child). Fill the hole:

1. If (one of) its siblings has two values, then bounce the values around to fill the hole. You're done.
2. Otherwise, if it has two siblings, the sibling with one value will steal a value from its parent. You're done.
3. Otherwise, it has one sibling with one value, and its parent has one value. Have the sibling steal the value from the parent, and move the hole up.
4. If the hole is now the root, it disappears and its one child is the new root. You're done.
5. Otherwise, repeat the above steps for the new hole node.

Insert



Delete


6 16
3
11
19
1
4
9 10
13
23
6 16
3
11
1
4
9 10
13
19 23
6
3
11 16
1
4
9 10
13
19 23