3110: Data Structures & Functional Programming

Red-Black Trees

Logistics

- Hmwk solutions

- Project

Today

- Binary Search Trees
- Red Black Trees

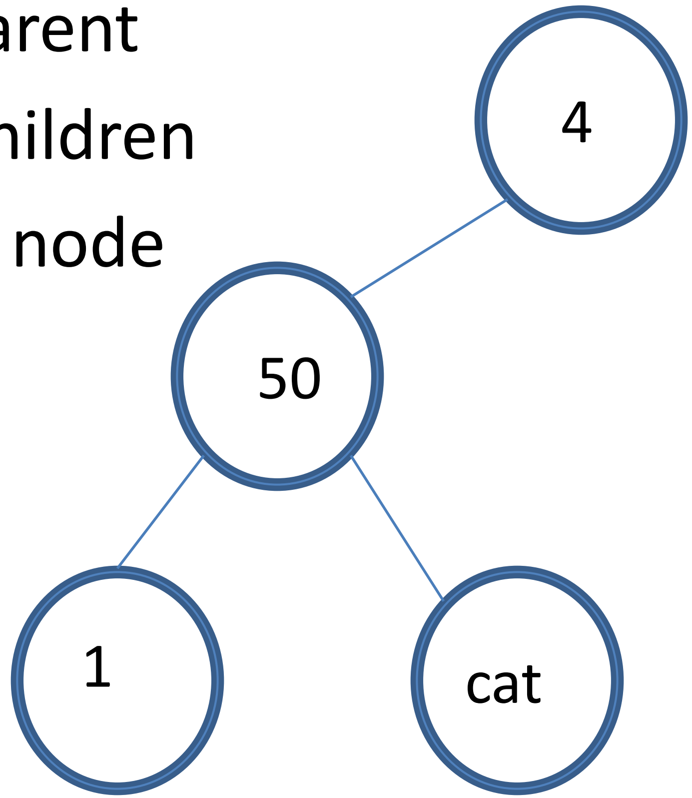
- Algorithms:
  - DFS
  - BFS
  - Heap Sort

- Forests

- Disjoint Sets

Tree?

- Each node is an object with children and 1 parent.
- Root Node – no parent
- Leaf Nodes – no children
- Key / Data in each node

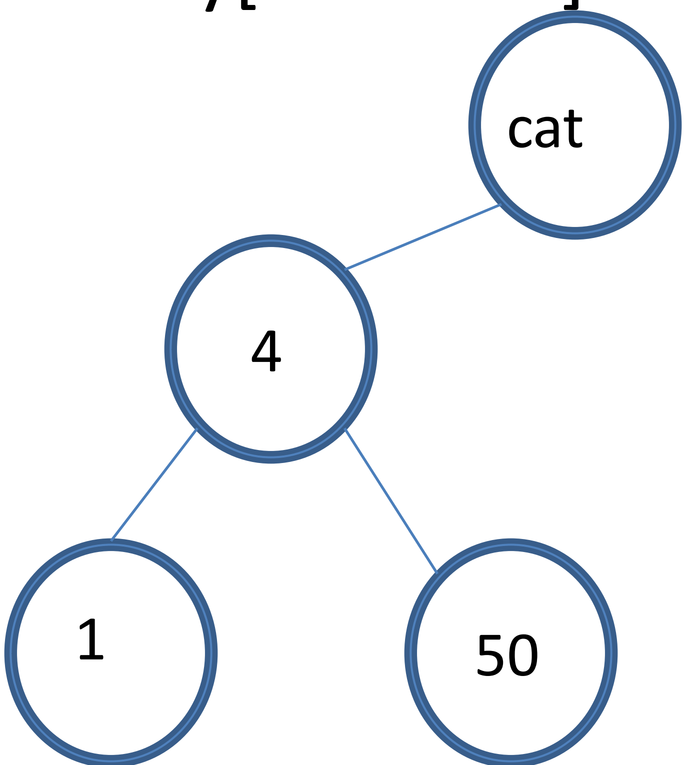


[value, left node, right node, parent]

Binary Tree

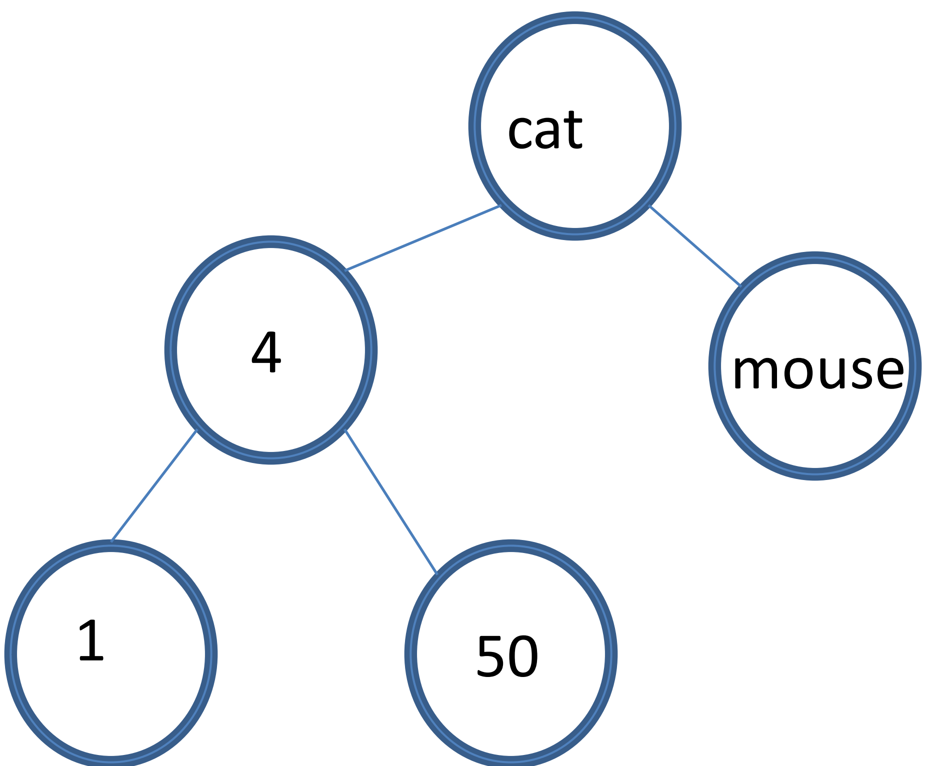
- Max 2 Children
- If search tree:

  key[left child] ≤ key[parent] ≤ key[right child]



Basic Add Operation

- For a Binary Tree. Follow left/right children until find value hit None. Create new node and add as left or right child of last node



To be programmed in a functional style, have to return a new tree.

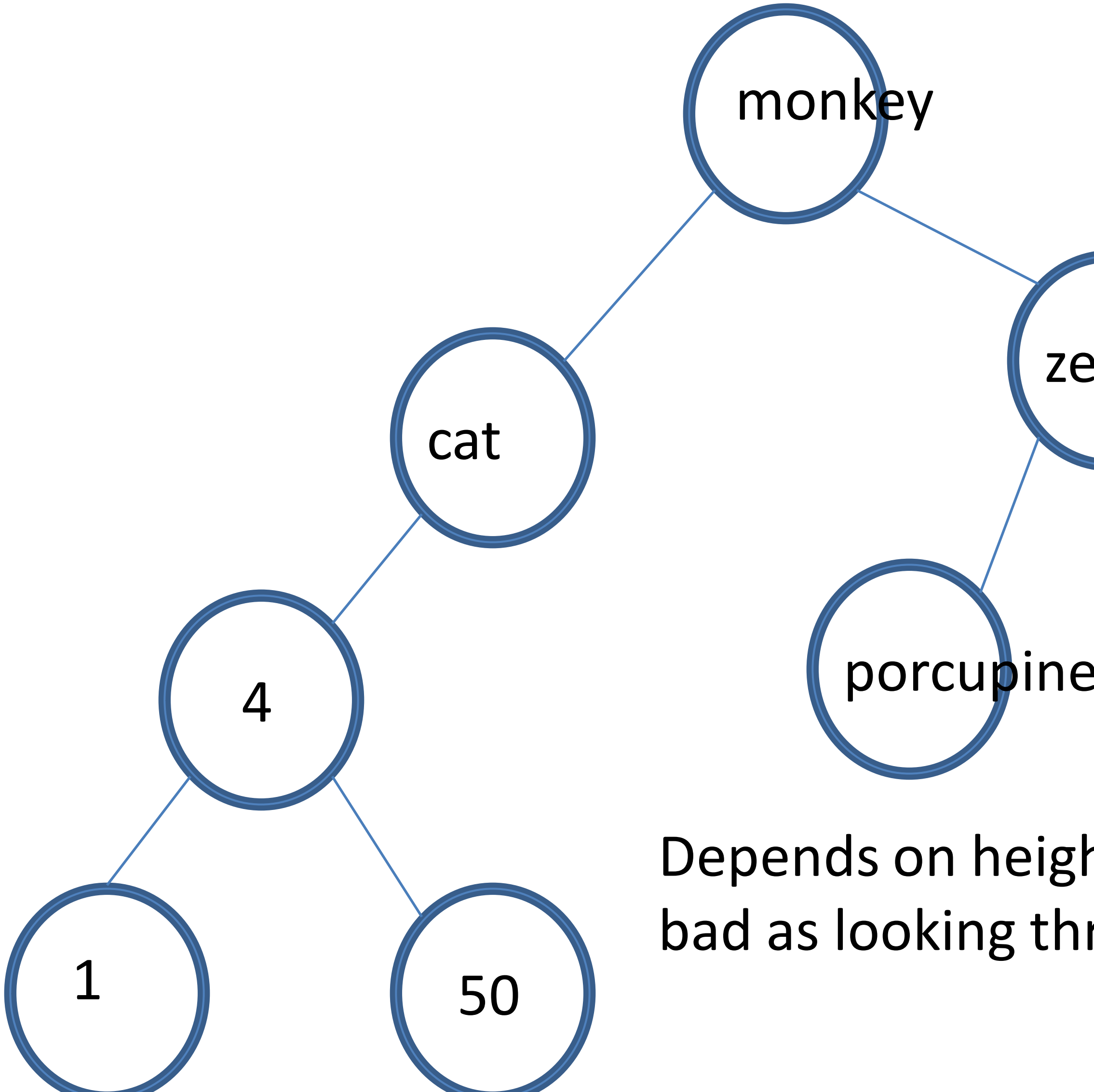
Min/Max in a Binary Search Tree

- For min, follow left children until reach leaf node

- For max, follow right children until reach leaf node.

- Correctness?

Time to find Min/Max



Depends on height of tree, can be just as bad as looking through a list.

Balanced Trees

- Have a guarantee on the height. No path from root to leaf will be significantly longer than any other path.
- If perfectly balanced tree, then

asdg

$$n = 2^{h+1} - 1$$

$$h = \log(n+1) - 1 = O(\log(n))$$

- New find min/max time?

Ex. of Balanced Search Trees

- B-trees
- Red-black trees
- Splay trees
- 2-3-4 trees
- 2-3 trees

Red-Black Trees

- New node representation:

[red/black, value, left node, right node]

Red-black tree satisfies:

1. Every node is either red or black
2. Every leaf node is black (use NIL nodes)!
3. A red node's children are black
4. Every simple path between a node and it's descendent leaves contains the same number of black nodes.

Black Height

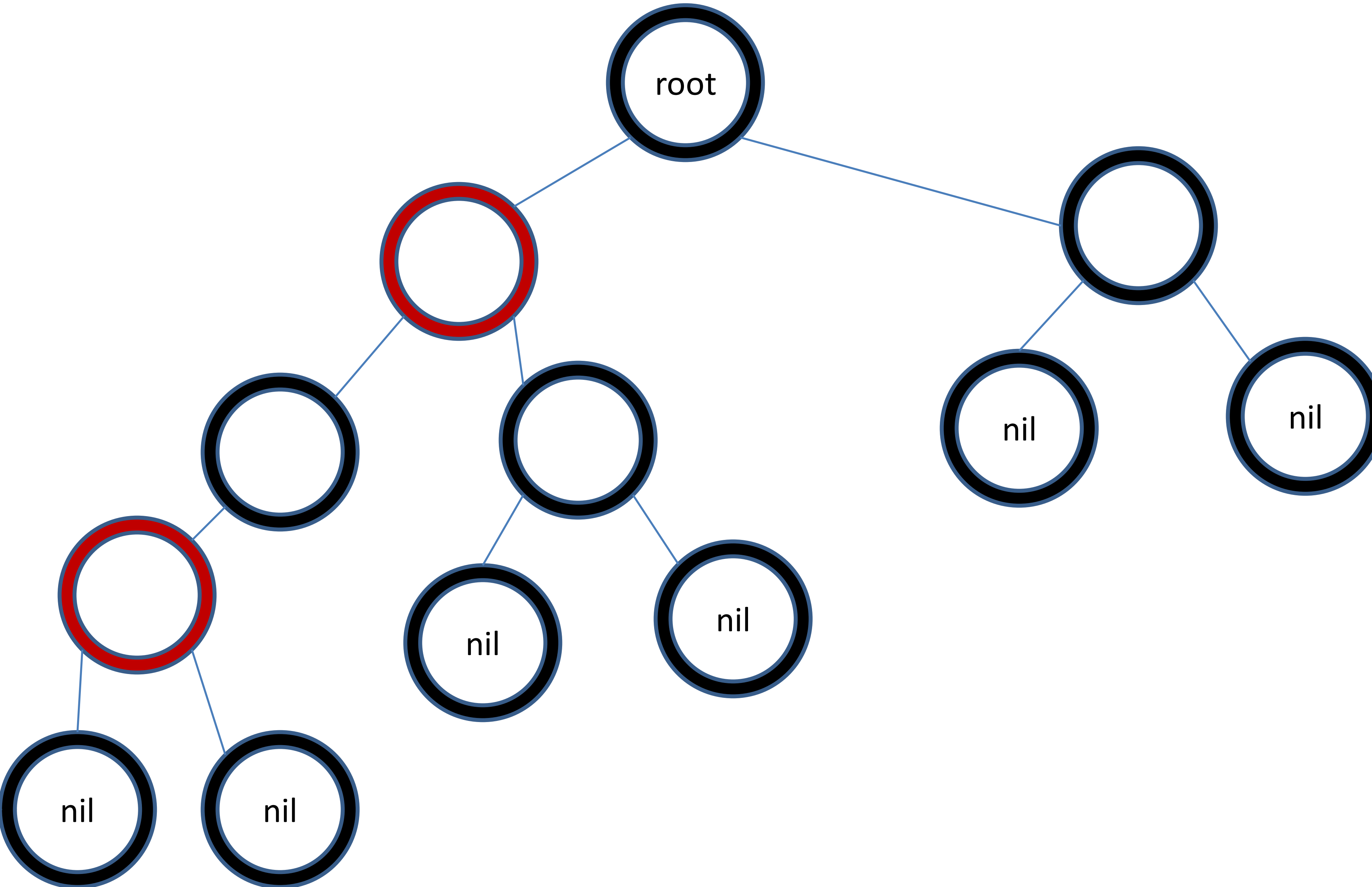
- Given that every path from a node to its descendent children contains the same number of black nodes the function:

bh(x), the ‘**black height**’, is well defined

- The black height of a tree is the bh(root)

- Without loss of generality, set the root node to black(will result in another red-black tree)

Ex. Red Black Tree



Lemma:

- The maximum ratio between the shortest path from the root to a leaf node and the longest path from the root to a leaf node is 2.

- Use fact that no red node can have a red node chlid.

Homework question

- Consider a red-green-blue tree.
- Define the blue height in the same way we defined the black height.
- What rules would you need to guarantee the ratio of the shortest path from the root to a leaf and the longest path from the root did not exceed 3?

Height of red-black tree

- Lemma:

A red-black tree with n nodes has height at most

$$2\log(n+1)$$

Proof

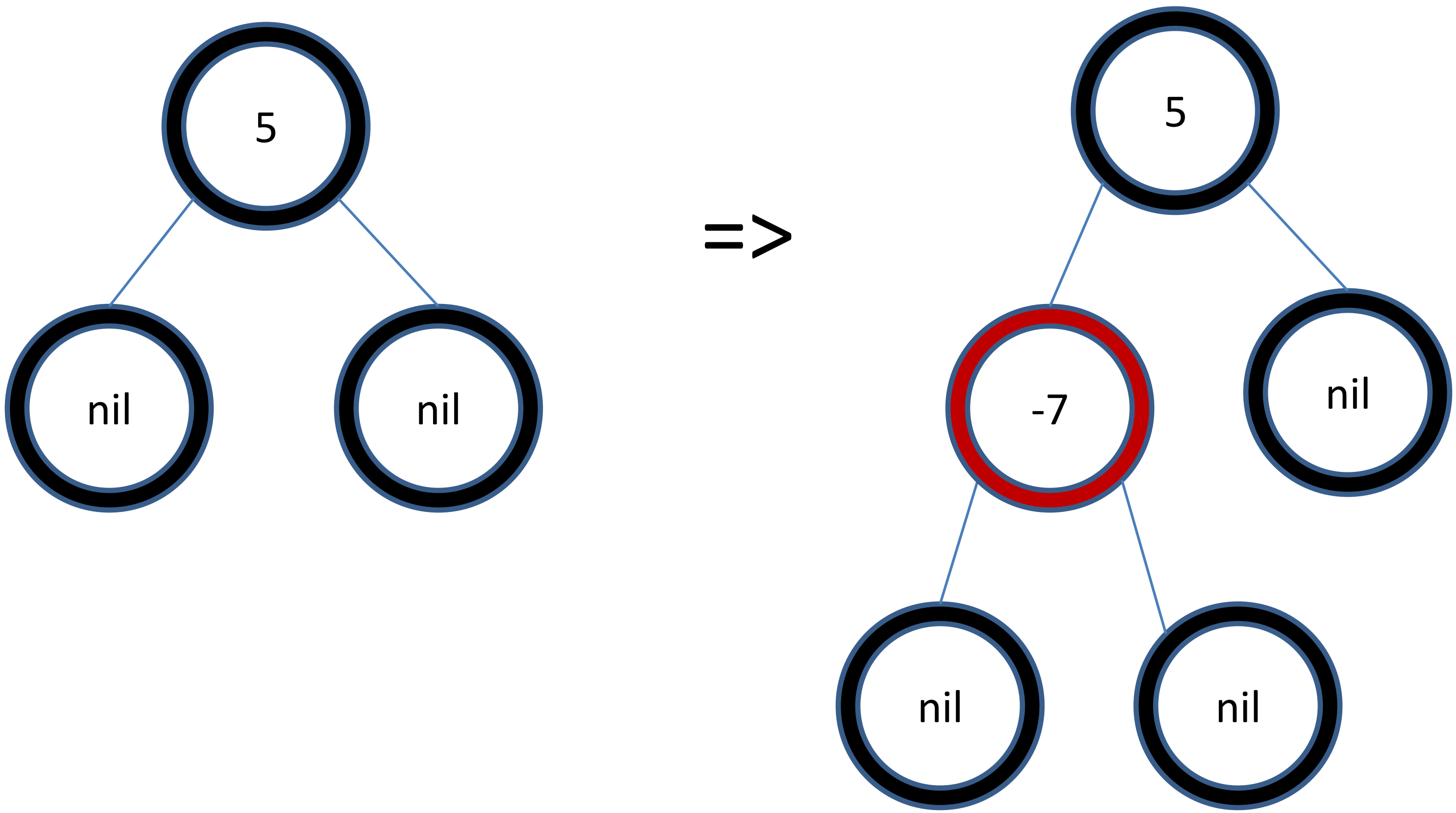
- A red-black tree with only black nodes must be perfectly balanced. (recall black height definition)
- A black only tree has height log(n+1)
- We can recolor and rearrange nodes, but we will not change the black height.
- The resulting tree will have a longest path of 2*black height

First Things first

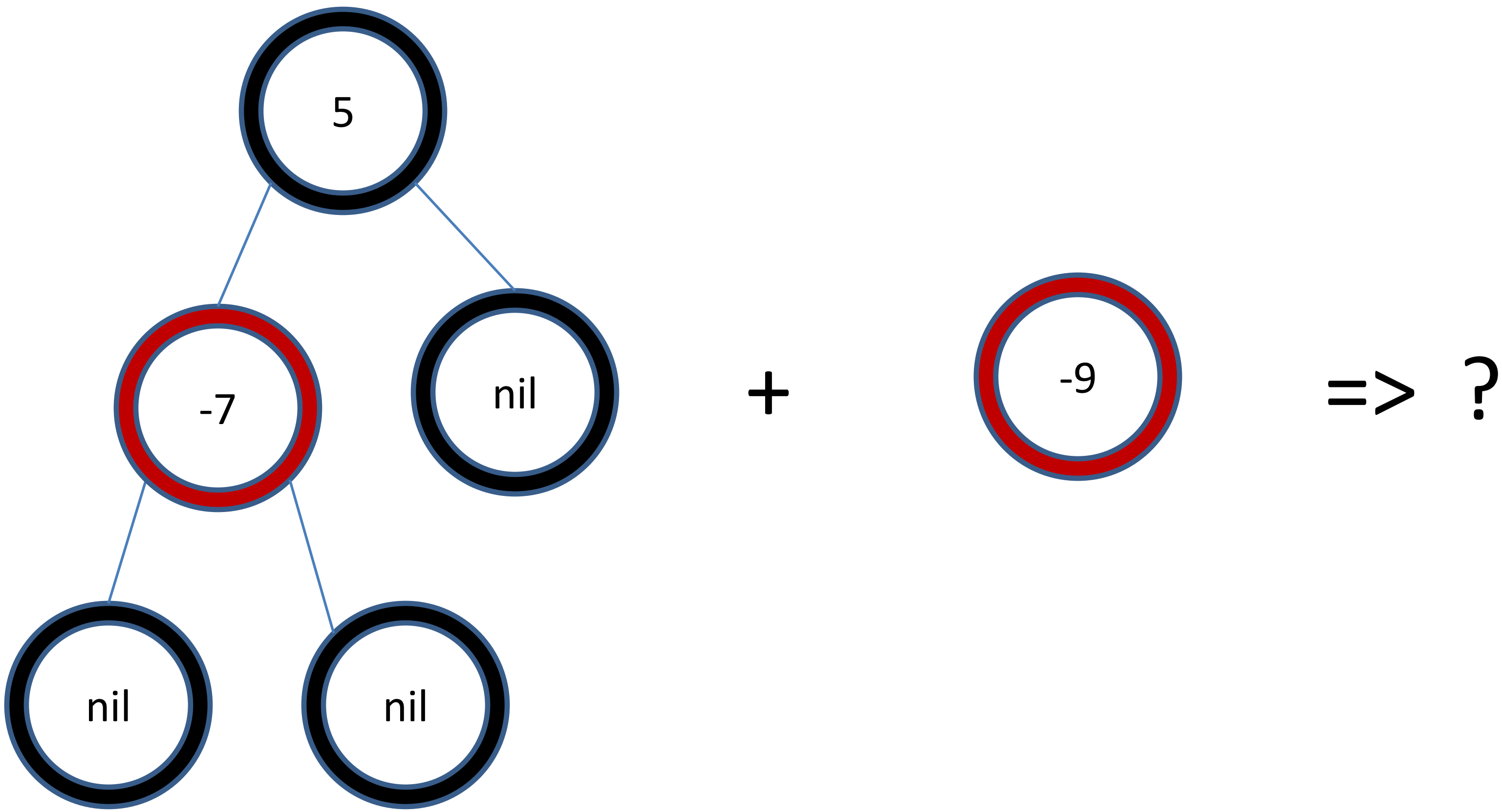
- Min/Max in red-black tree

- Check if a value is in tree?

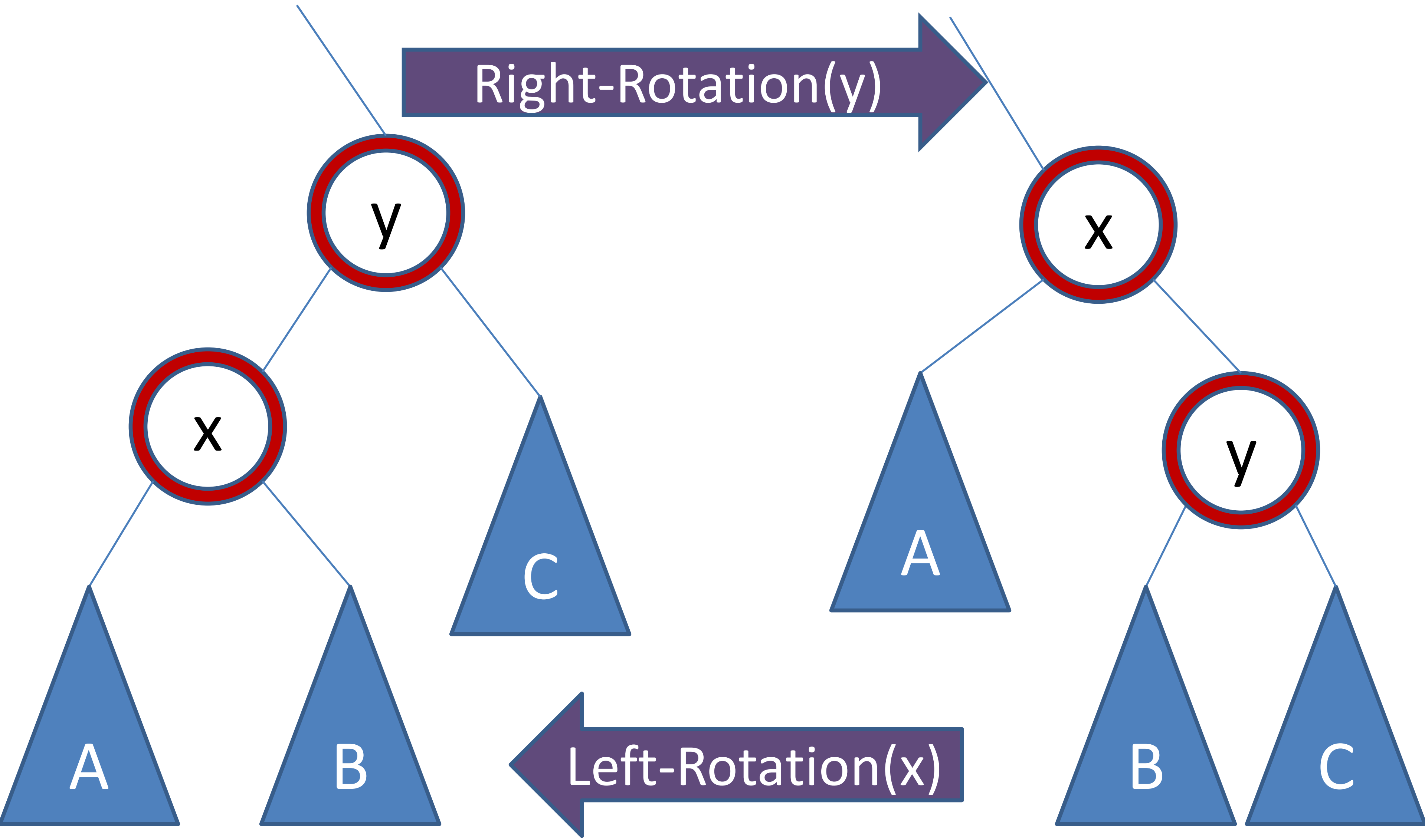
Insert Node



Easy!


5
-7
nil
nil
nil
+
-9
=> ?

Rotations

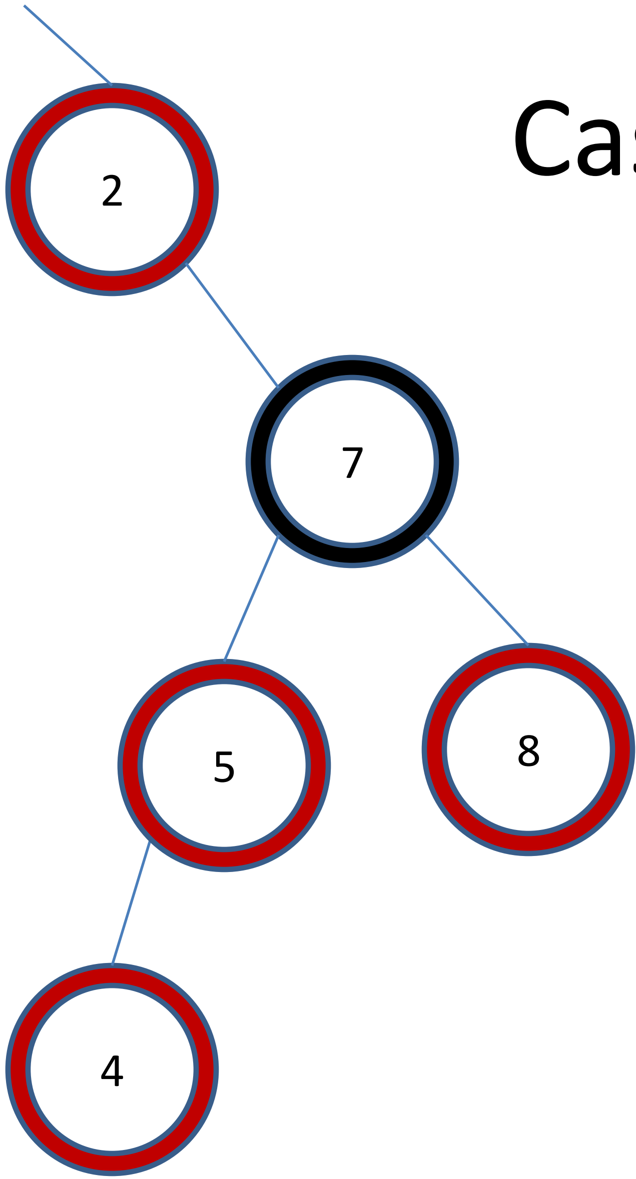


On board write out pointers

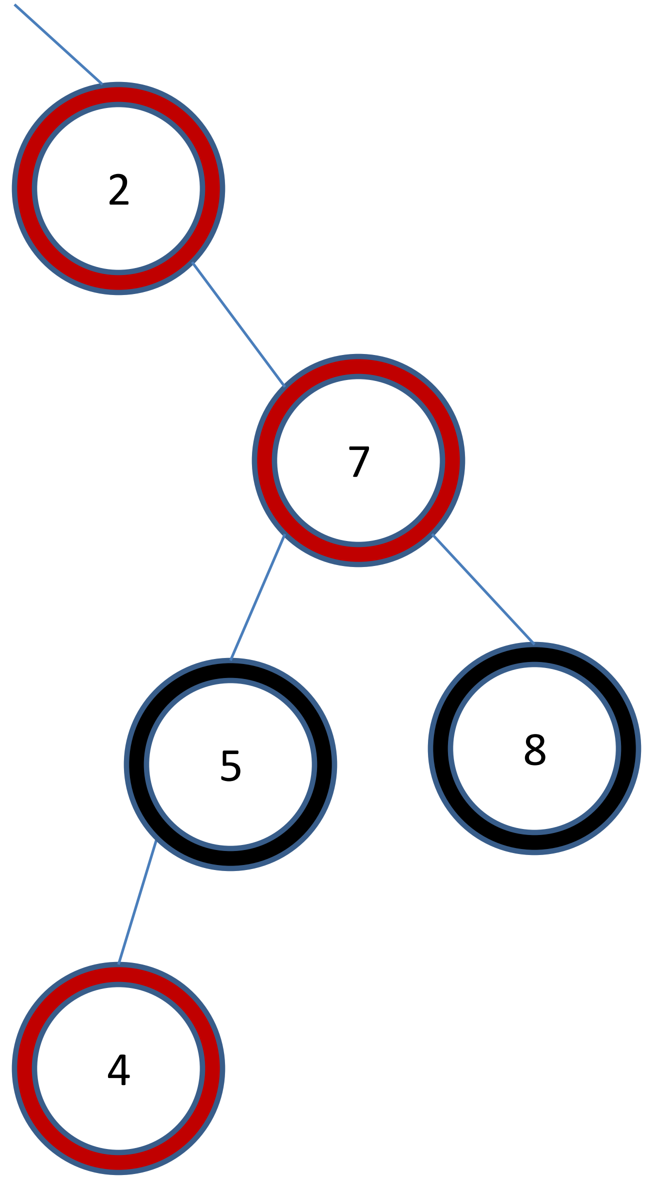
Cases

- We present how to deal with half the cases, the others are just reciprocal.
- Limit ourselves to operations that do not change the black height of a subtree

Case 1

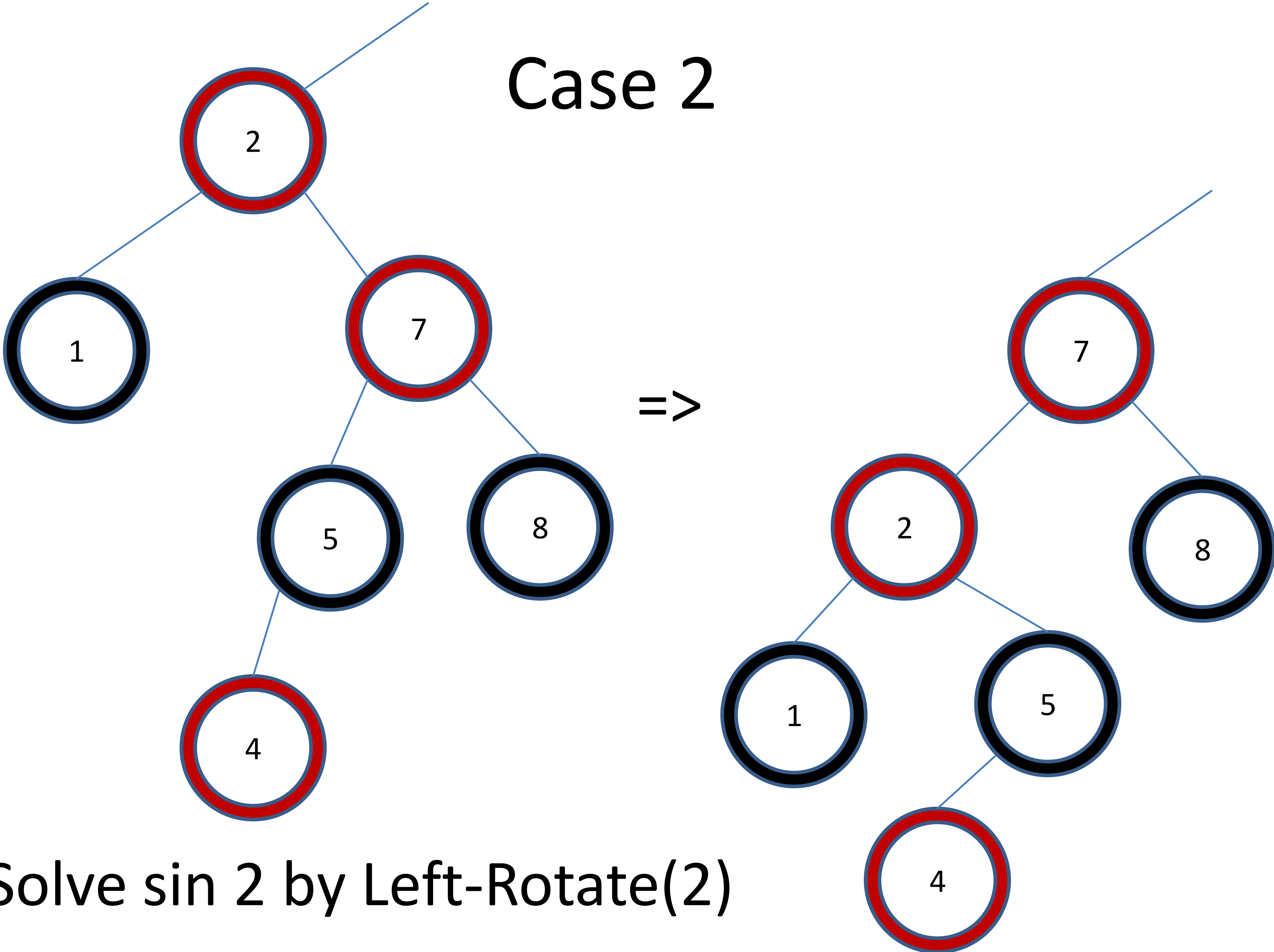


=>



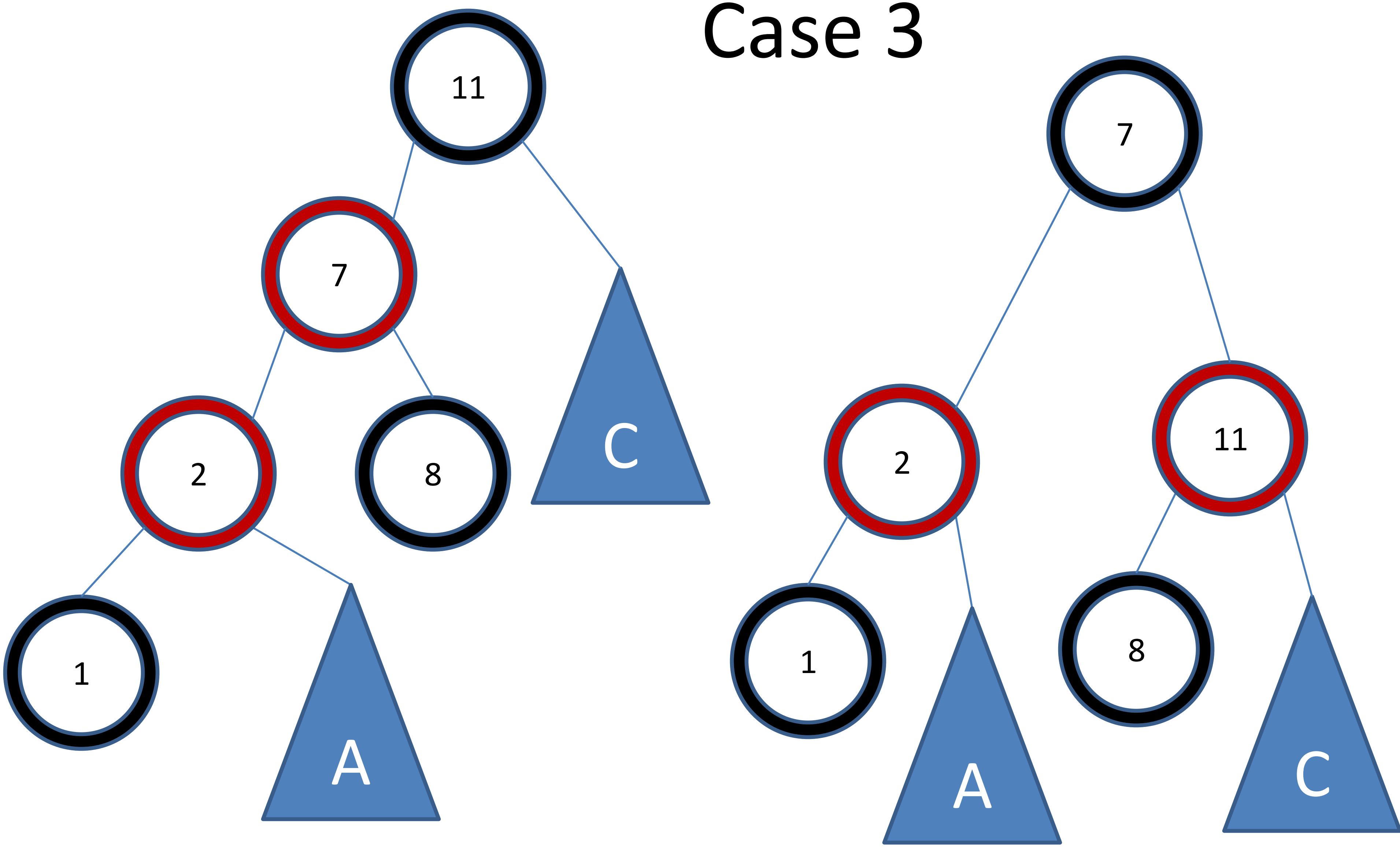
Solve sin 1 by Recolor

Case 2



Solve sin 2 by Left-Rotate(2)

Case 3



Solve sin 3 by Right-Rotate(11) and recolor

Runtime

- Never perform a two left-rotations or two right-rotations in a row.

- Hence on each rotation, move the problem up the tree by 1 level.
- Hence the number of rotations is the depth of the tree O(log(n))

Deletion

- Next time:
- Deletion and Sets

Side Notes

- Use Depth First Search to return the elements stored in a tree in order.

- To return elements in a range, use binary search to find minimum element and use DFS on right subtree until max of desired range is found. (May have to move up in the tree, still only explore right subtrees)