Lecture 20

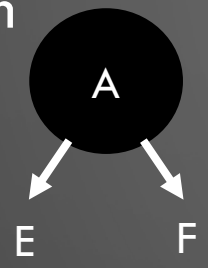
RED BLACK TREES

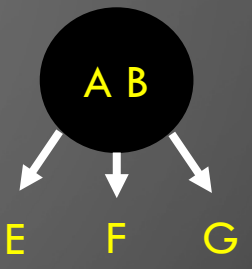
Outline

- Review 2-3 Trees
- 2-3-4 Trees
- Red-Black Trees

Recall 2-3 Trees

- 2-3 tree is a tree data structure in which every internal node (non-leaf node) has either one data element and two children or two data elements and three children





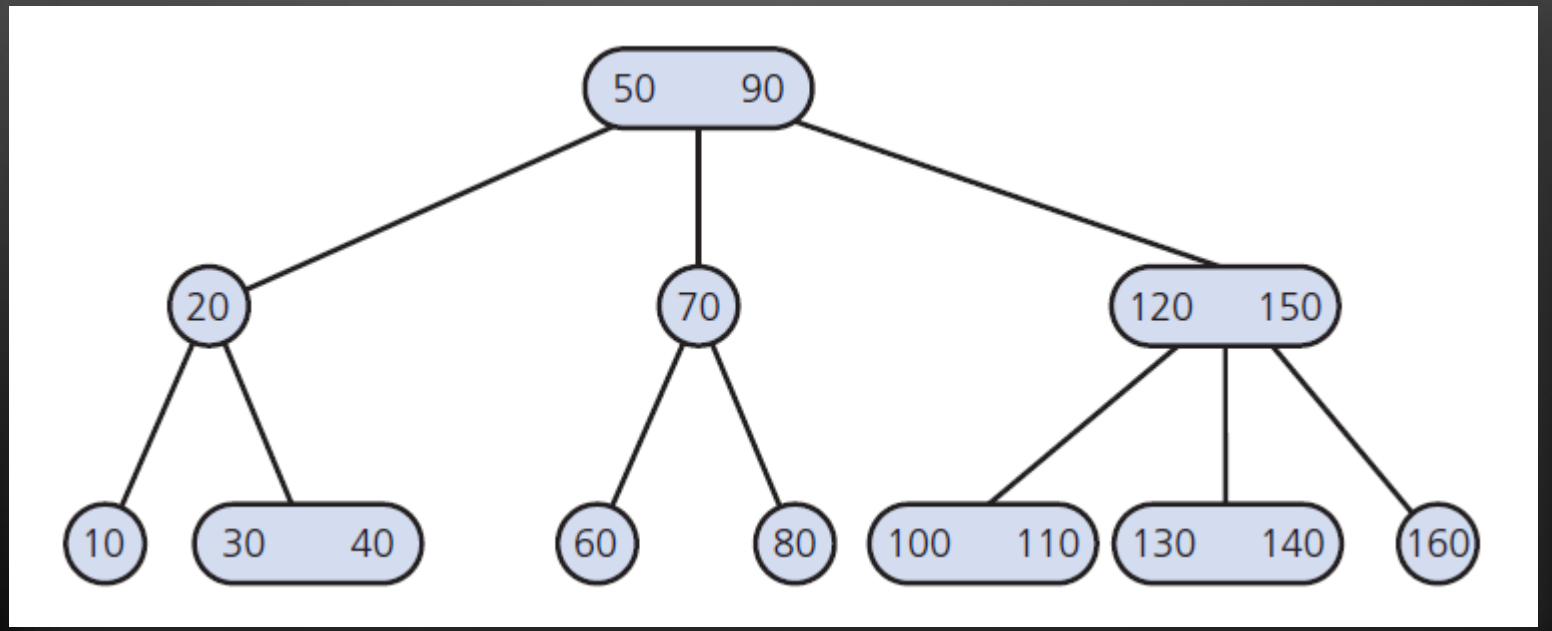
E is items < A

F is items > A

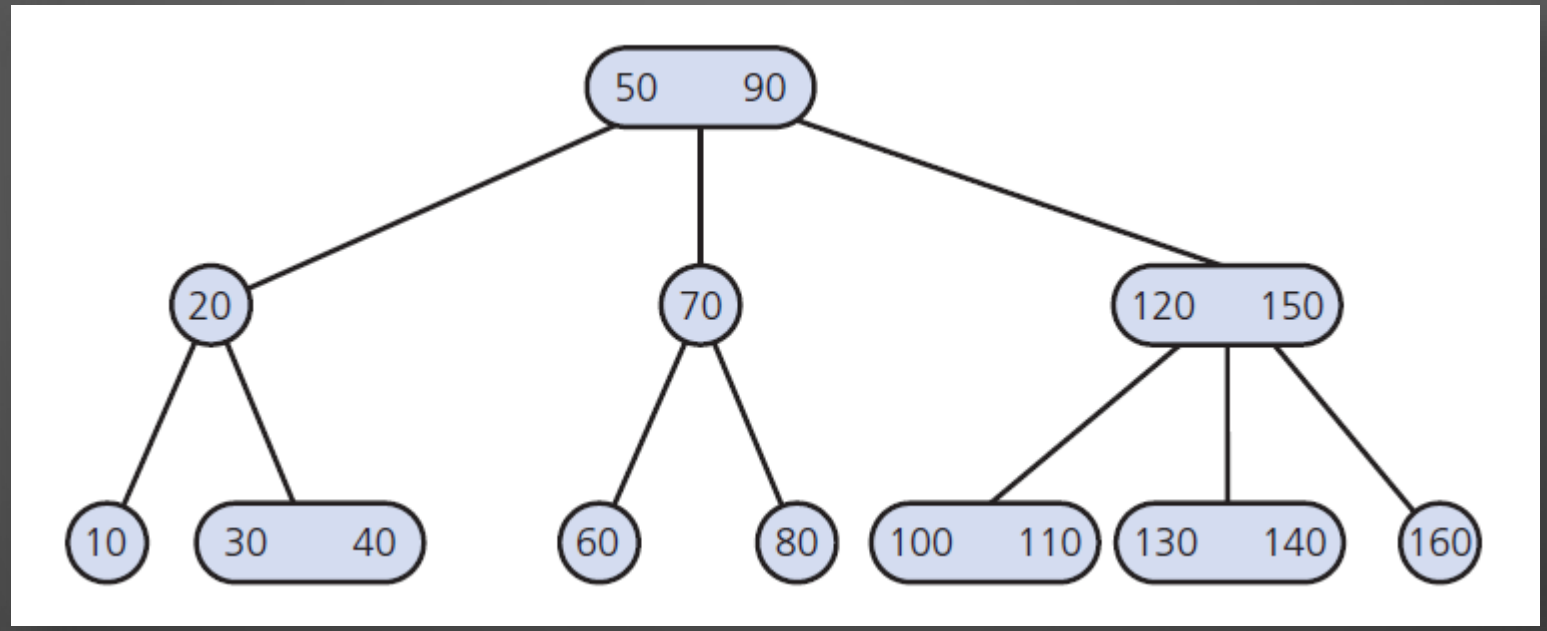
E is items < A

F is items > A and < B

G is items > B



Searching a 2-3 Tree



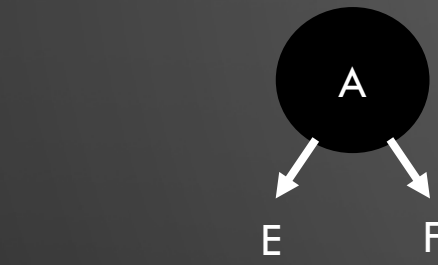
- Compare search complexity of a 2-3 and shortest binary search tree
  - Complexity is?
- A binary search tree with *n* nodes cannot be shorter than $\log_2(n + 1)$
- A 2-3 tree with n nodes cannot be taller than $\log_2(n + 1)$
- Node in a 2-3 tree has at most two data items
- Searching 2-3 tree is O(log *n*)

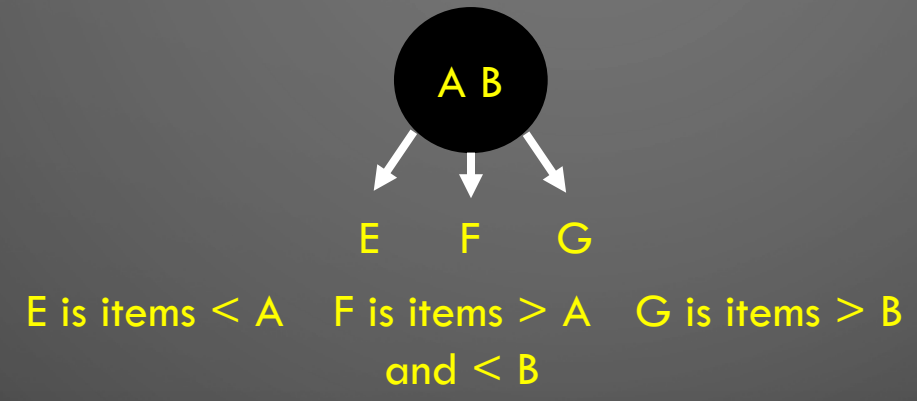
If 2-3 trees are so great

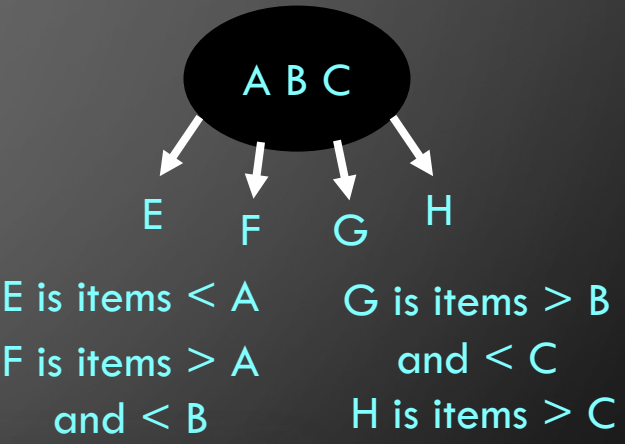
wouldn't a 2-3-4 tree be even better?

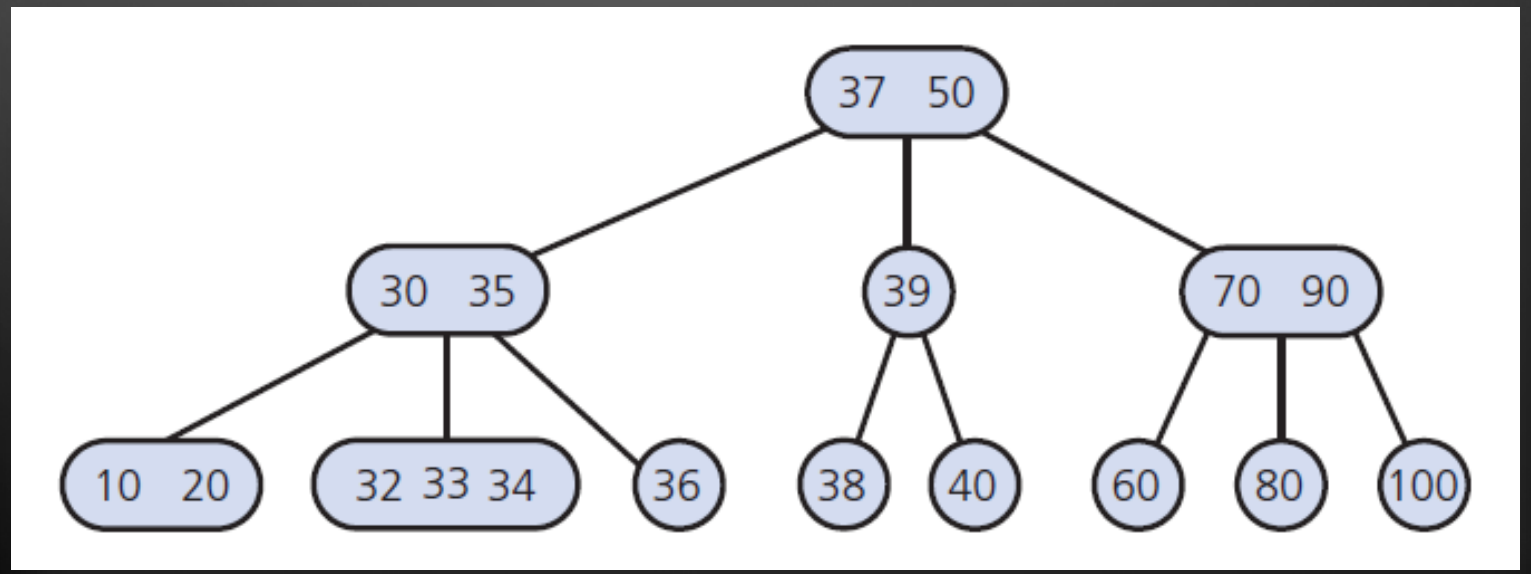
2-3-4 Trees

- 2-3-4 trees are the next logical extension of a 2-3 tree.
- Every internal node (non-leaf node) has one of; 1 data element and 2 children; 2 data elements and 3 children; or 3 data elements and 4 children







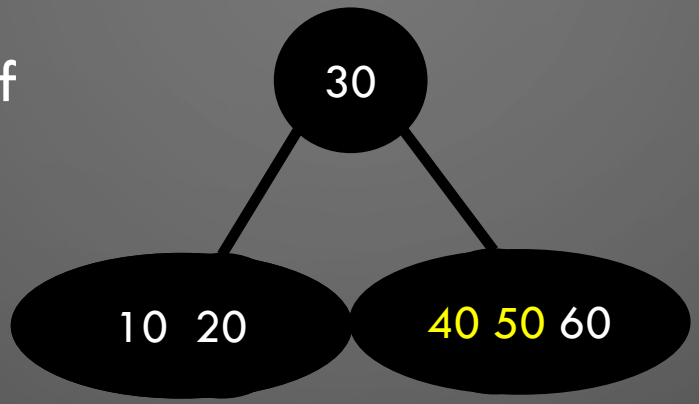


Adding Data to 2-3-4 Trees

- Add 20 to this 2-3-4 tree

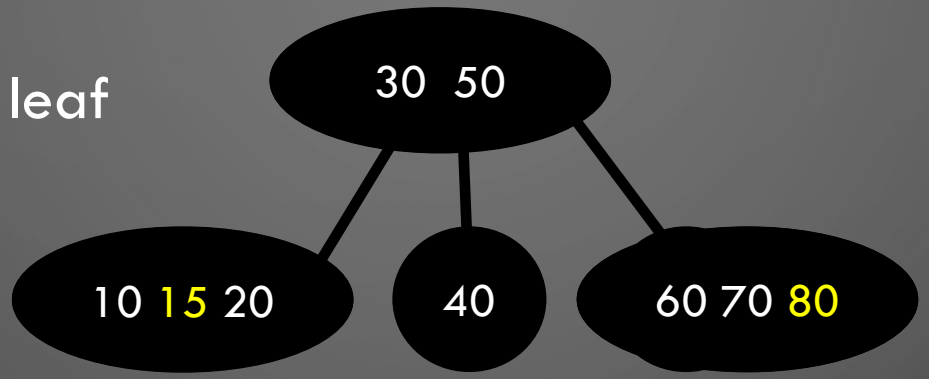
Adding Data to 2-3-4 Trees

- Add 20 to this 2-3-4 tree
- Split the tree
- Add the value to the leaf
- Add 40 and 50
- Add 70



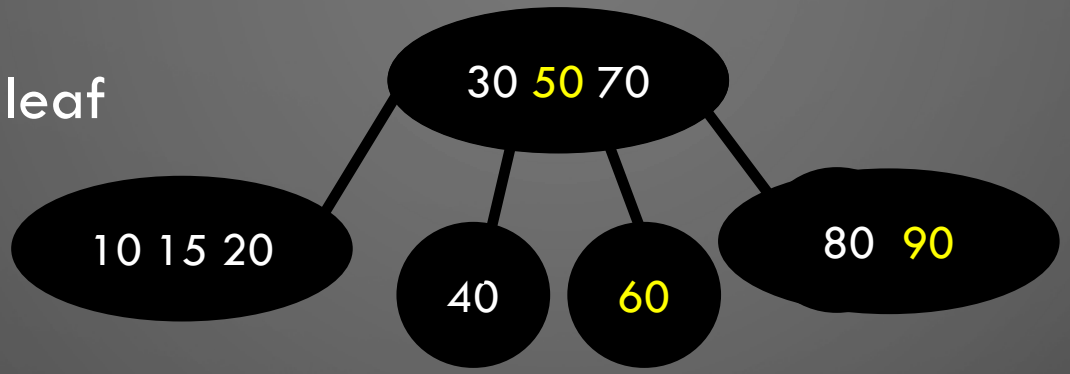
Adding Data to 2-3-4 Trees

- Add 20 to this 2-3-4 tree
- Split the tree
- Add the value to the leaf
- Add 40 and 50
- Add 70
- Split the right leaf
- Put 70 in the right leaf
- Add 15 and 80 to the left and right leaf, respectively
- Add 90



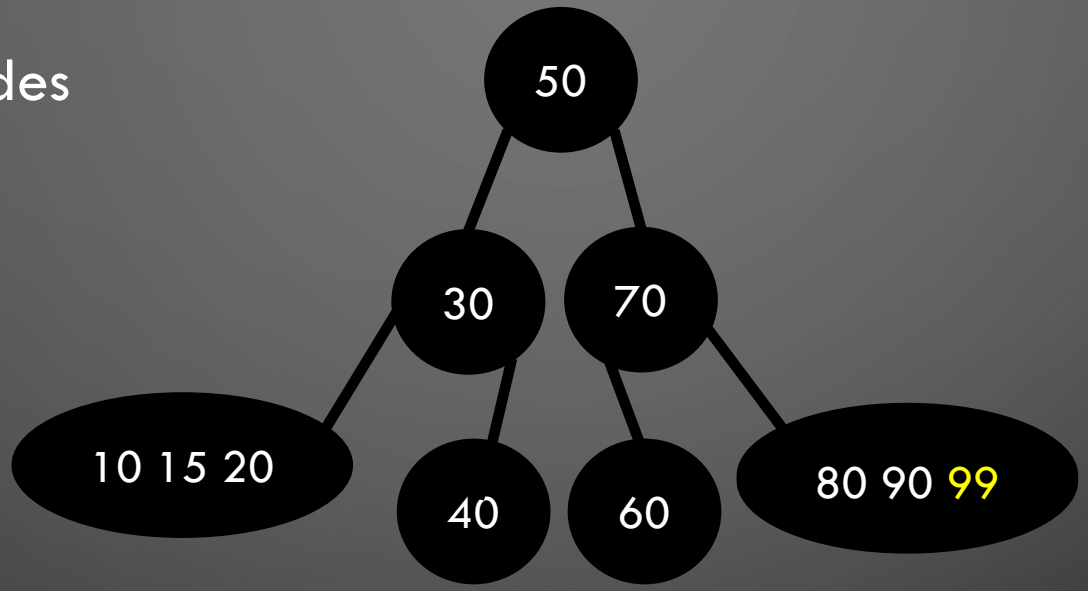
Adding Data to 2-3-4 Trees

- Add 90
- Split right leaf
- Add 90 to right leaf
- Add 99
- Normally we would just add it to the leaf node
  - Recall: to find the location to add the value we do a search
  - As part of this search we encounter a 4 node at the root.
- What if we split the 4 nodes during the search?
  - Pros? Cons?



Adding Data to 2-3-4 Trees

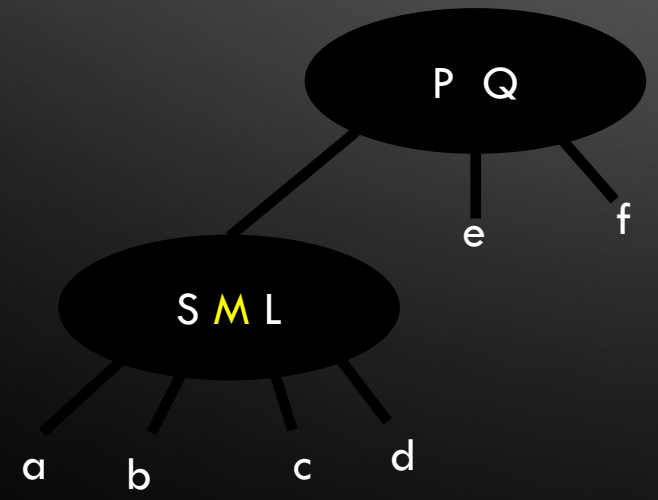
- Add 99
- Split root during search
- No further 4 nodes
- Add 99 to leaf

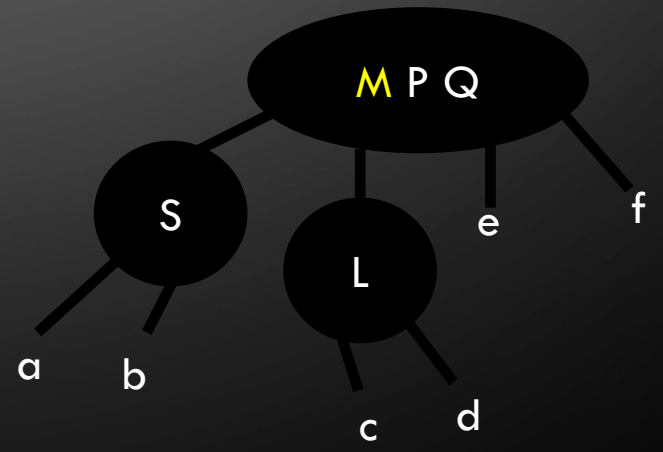


- Summarize Algorithm: Search for location of item in the tree
- If you encounter 4 non-leaf nodes, split them.
- Add item to the appropriate leaf node.
- Split leaf if necessary and push values up the tree.

Removing data from the 2-3-4 tree

- Similar algorithm to removal from 2-3 tree
- Locate item to remove
- If necessary: swap item with inorder successor to place it at the leaf.
- If that leaf is a 3-node or 4-node, you can immediately remove the data
- If guarantee 3- or 4-node leaves, deletion requires one traversal of the tree
- Remove 2-nodes from tree to guarantee this situation
  - Also center and left permutation of this case; also works fine if parent/root is 2-node

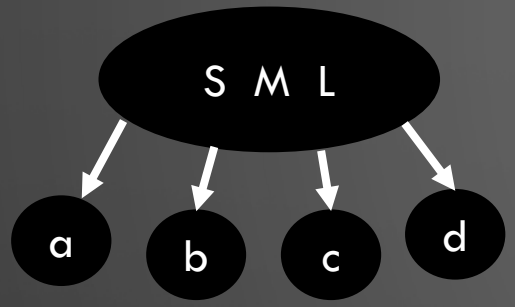




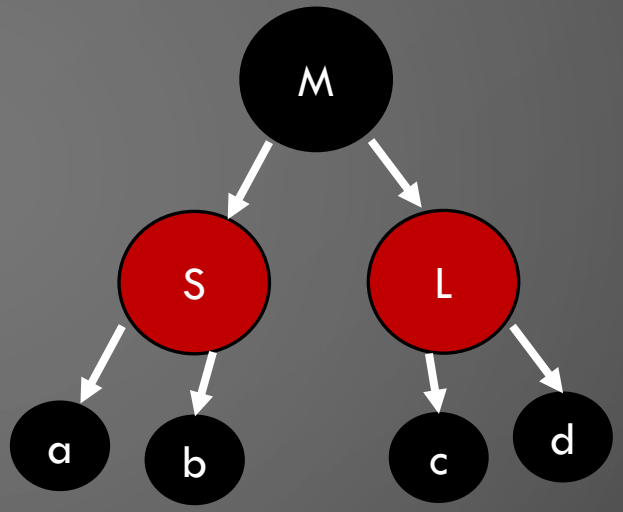
2-3-4-5 Trees?

- The value of 2-3 trees
  - Tree remains balanced
- The value of 2-3-4 trees
  - Tree remains balanced
  - Adding nodes requires a single traversal
  - Deleting nodes requires a single traversal
- We can go higher…but nodes get more complex
  - More comparisons per node/traversal, higher space complexity
  - No additional real value to computational complexity
- A binary tree that represents a 2-3-4 tree is a Red-Black Tree

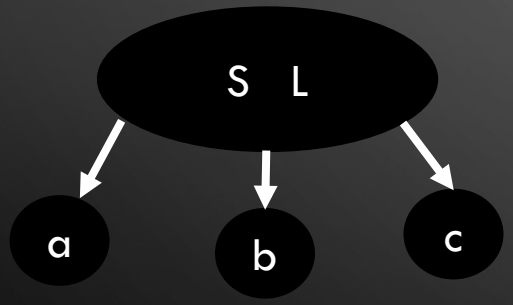
Red-Black Trees



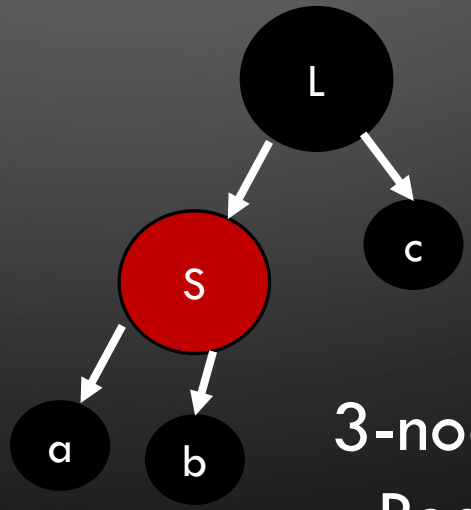
4-node

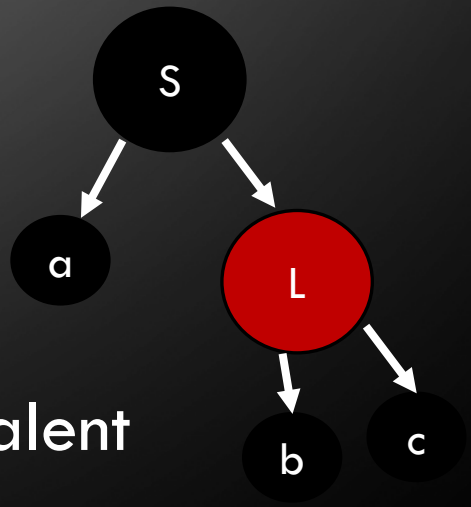


4-node equivalent
Red-black Tree



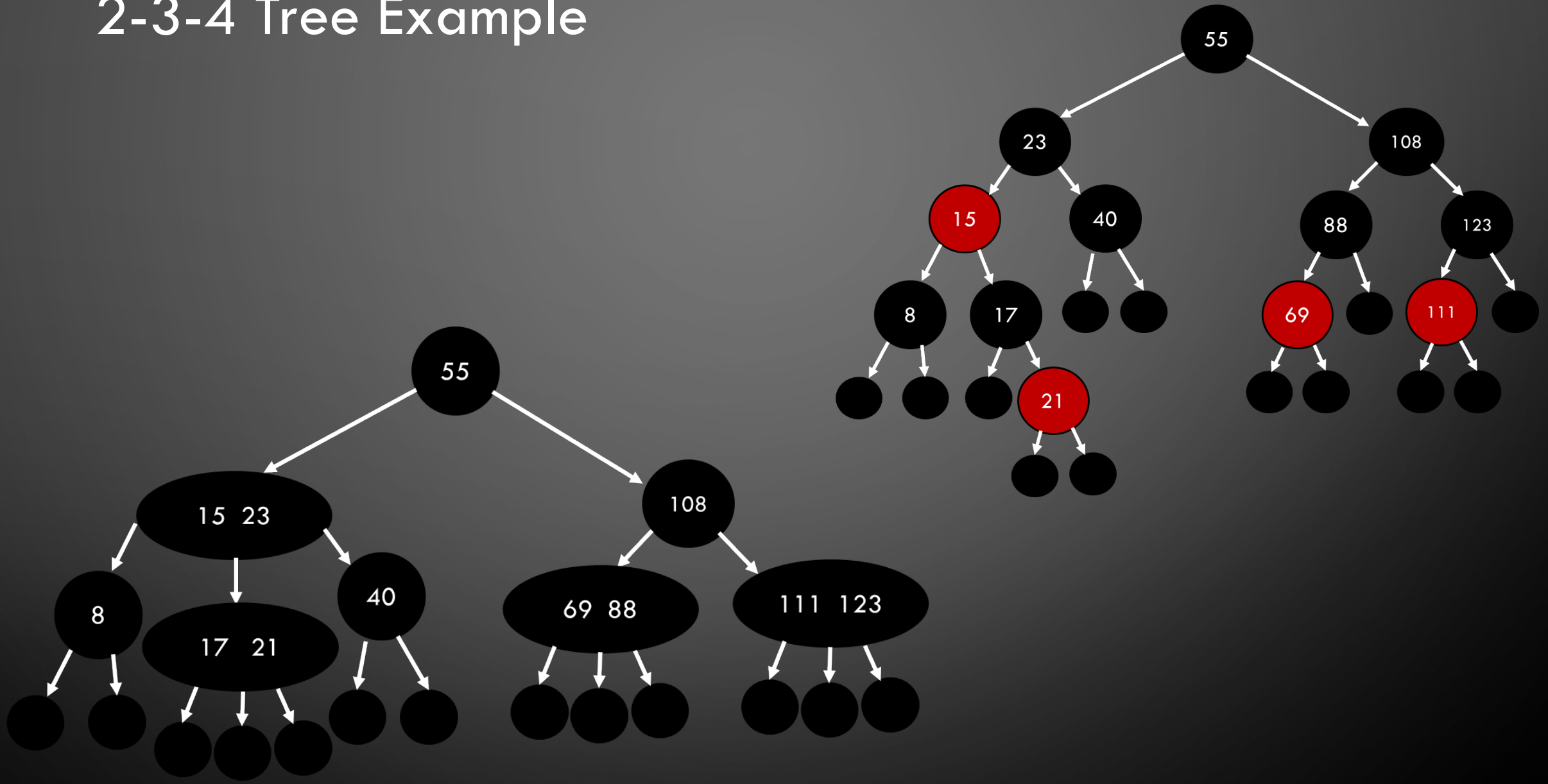
3-node



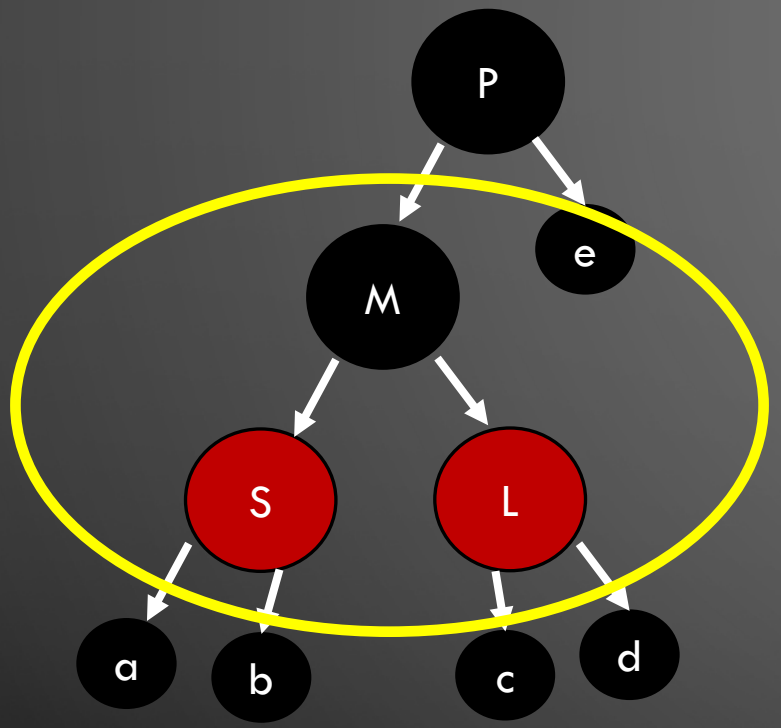


3-node equivalent
Red-black Tree

2-3-4 Tree Example

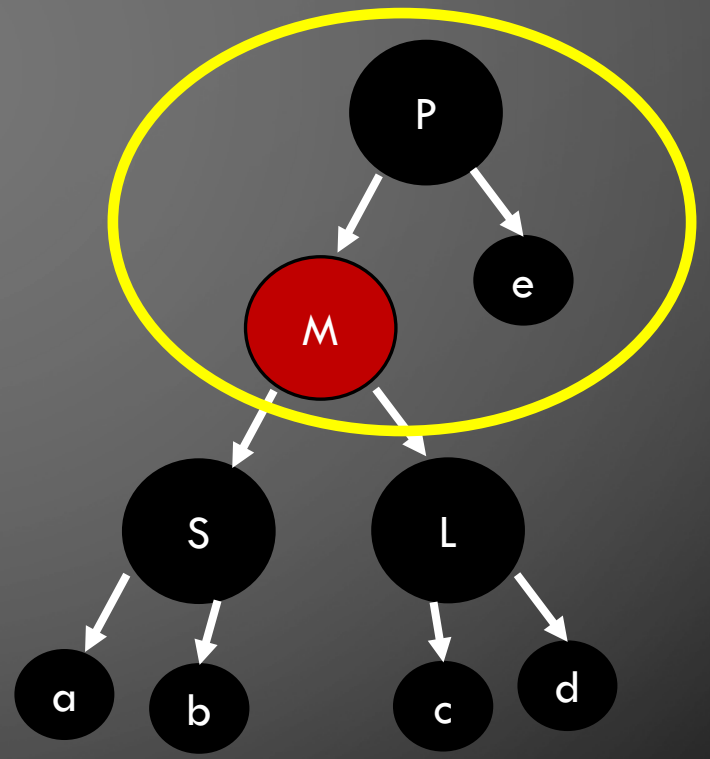


Splitting a 4-node – Parent is a 2-node

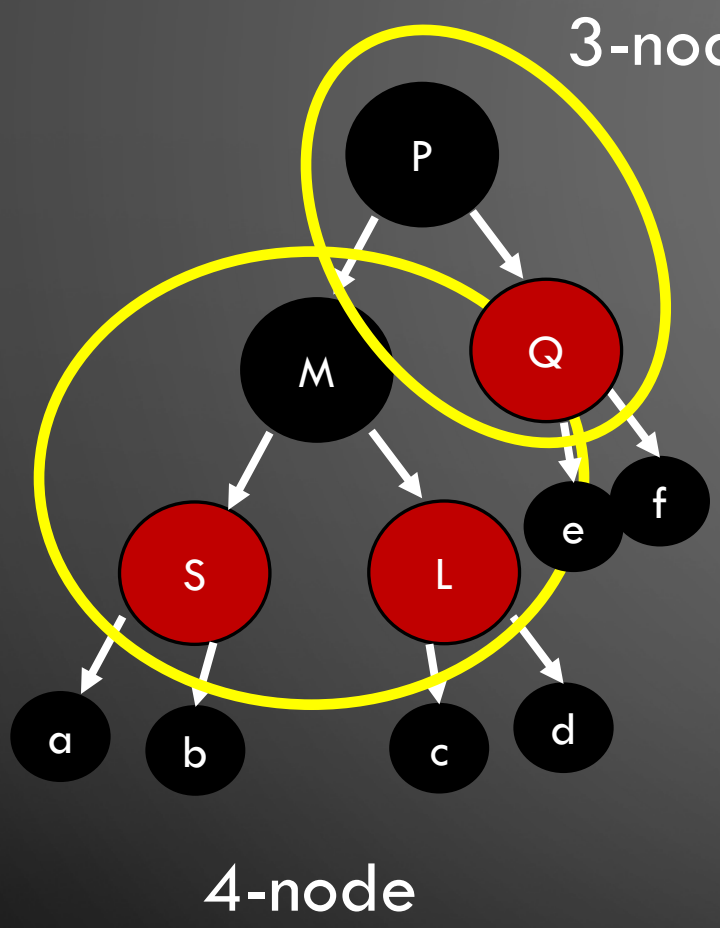


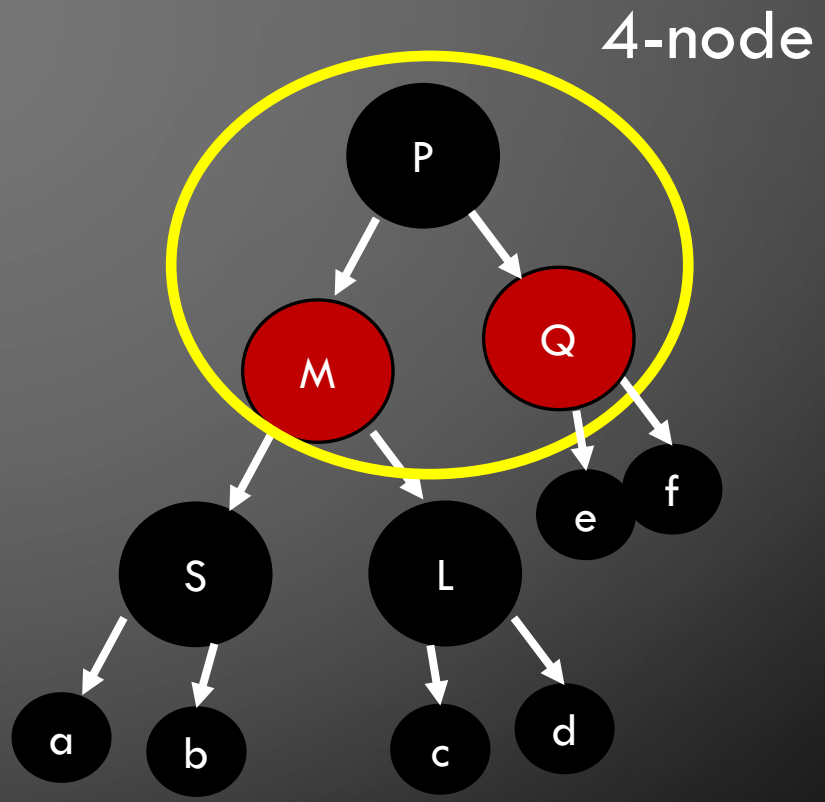
4-node

3-node

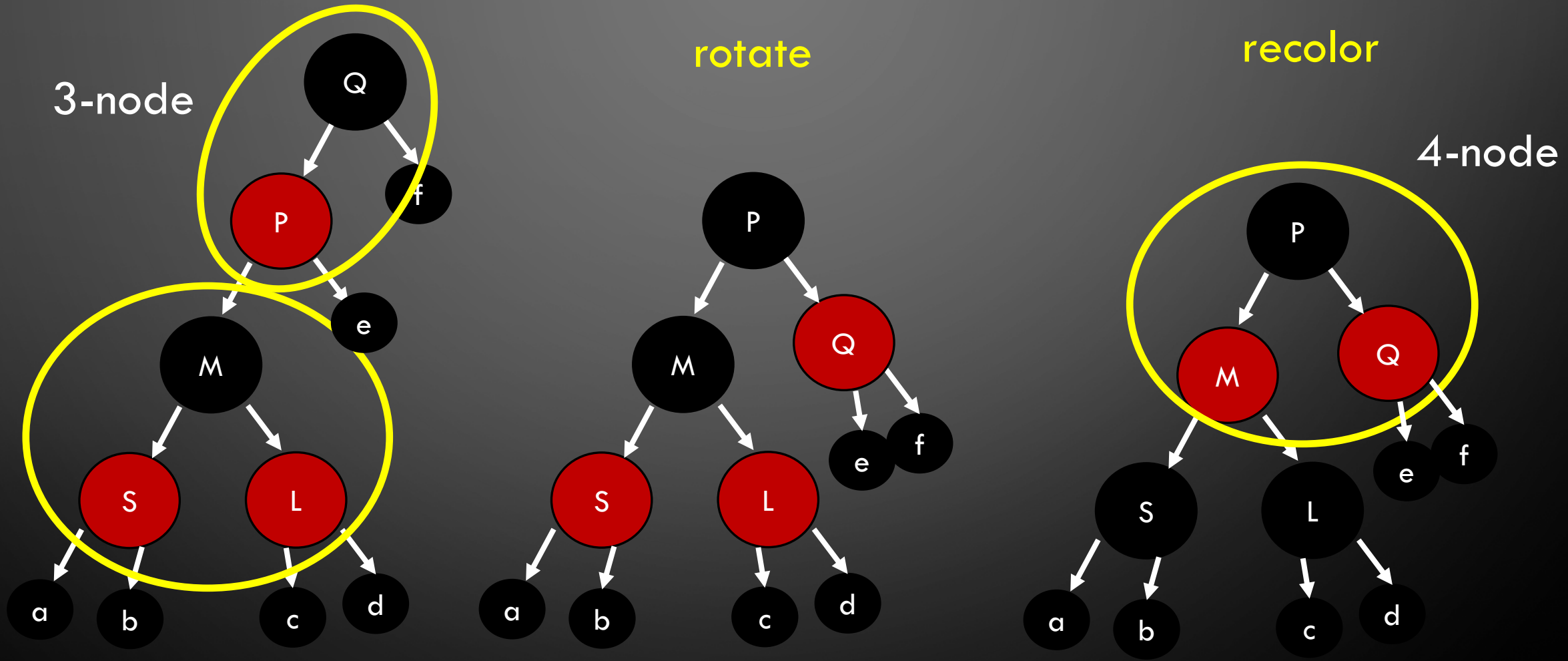


Splitting a 4-node – Parent is a 3-node





Splitting a 4-node – Parent is a 3-node



Similar Transformations (textbook Fig. 19-39)

- 2-node parent with 4-node right child
- 3-node parent with 4-node as middle child
- 3-node parent with 4-node as right child

Red-Black Trees

- Red-Black Adjacency Relationships
  - Black nodes may be adjacent with Black nodes
  - Black nodes may be adjacent with Red nodes
  - Red nodes may NOT be adjacent with Red nodes
  - Every path from the root to a leaf contains the same number of BLACK nodes
- What is the maximum height of a Red-Black Tree?
- What is the complexity of search?
- What is the complexity of addition?
- What is the complexity of removal?
- What is the space complexity?

Summary of Balanced Trees

- AVL Trees
  - BST that is guaranteed to remain balanced using rotation operations
- 2-3 Trees
  - Internal 2-Node have 2 children and 3-Node have 3 children exactly.
  - Addition is done in a leaf node and splitting up if needed.
  - Removal is done by finding the inorder successor and combining down if needed.
- 2-3-4 Trees
  - Internal 2-Node have 2 children, 3-Node have 3 children, and 4-Node have 3 children exactly.
  - Addition requires a single traversal, if you split 4-nodes as you search
  - Deletion requires a single traversal, if you guarantee 3- or 4-node leaf nodes.
  - More efficient than the 2-3 Trees.
- Red-Black Trees are the binary tree representation of 2-3-4 Trees
  - Require less storage 2-3-4 trees
  - Addition and removal is more efficient than 2-3-4 trees.

Assignment/Homework

- Reading pp. 629-659
- Homework 8 due Thursday
- HW 9 is released