Augmenting Binary Search Trees

Michael Goodrich

CS 165

Slides adapted from Erik Demaine, Charles Leiserson

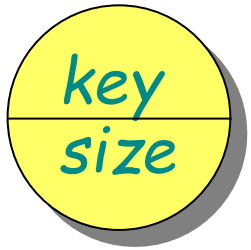
Dynamic order statistics

OS-SELECT(i, S): returns the ith smallest element in the dynamic set S.

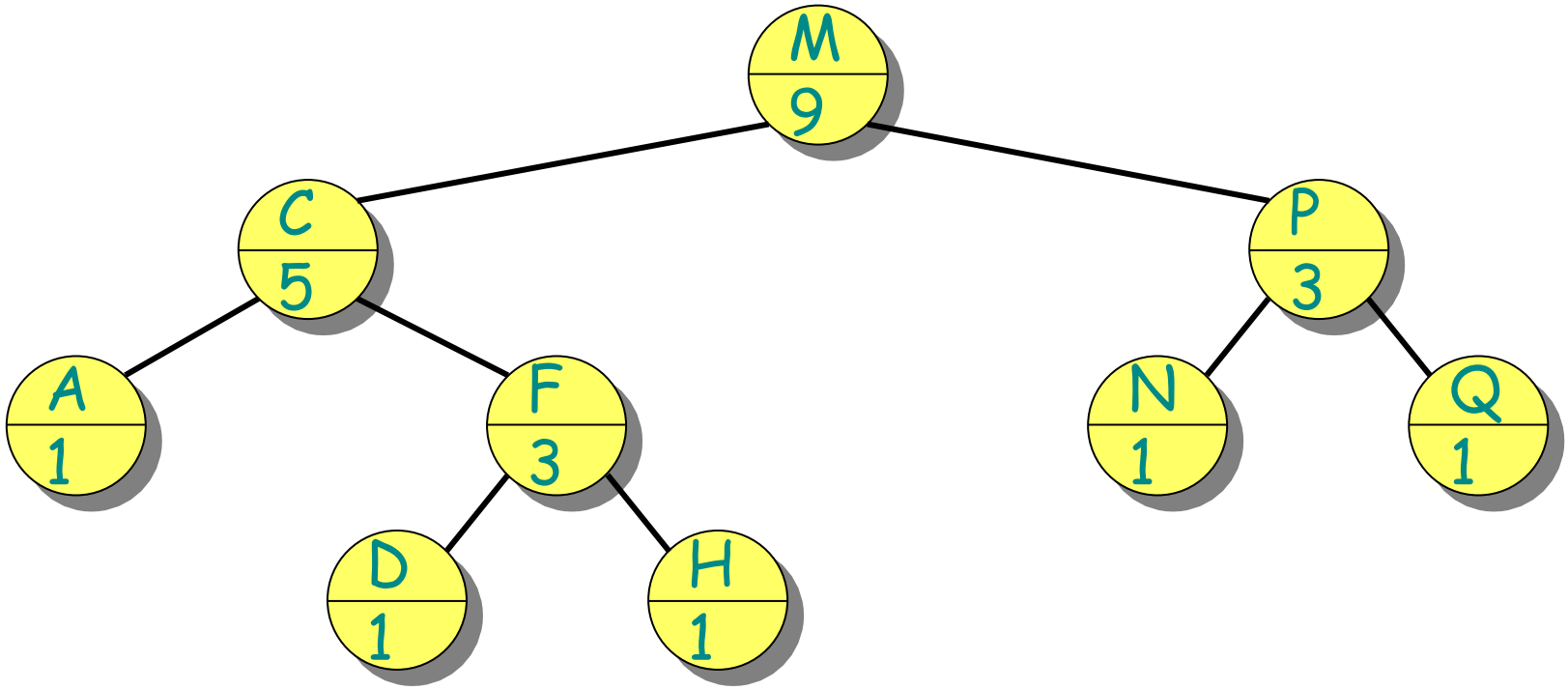
OS-RANK(x, S): returns the rank of $x \in S$ in the sorted order of S's elements.

IDEA: Use a binary search tree for the set S, but keep subtree sizes in the nodes.

Notation for nodes:



Example of an OS-tree



$size[x] = size[left[x]] + size[right[x]] + 1$

Selection

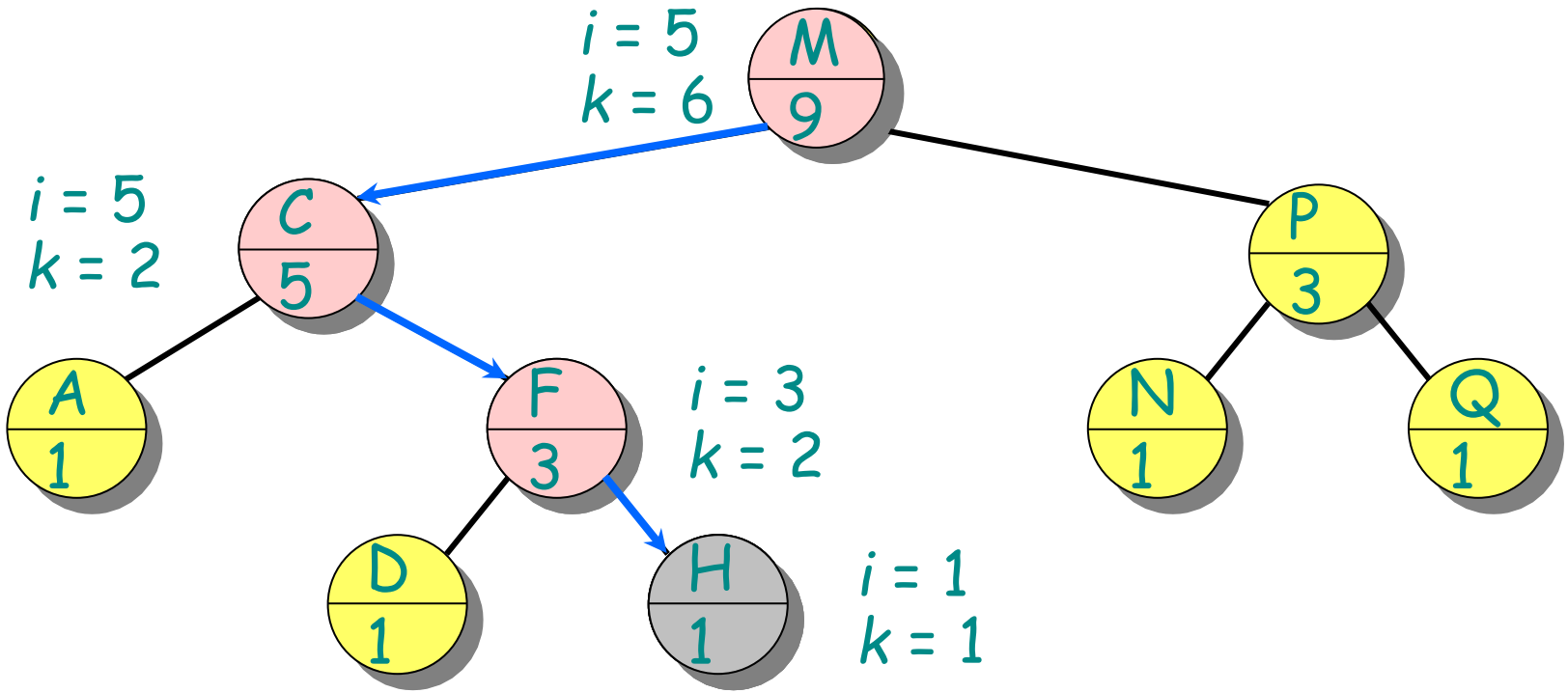
Implementation trick: Use a ***sentinel*** (dummy record) for NIL such that *size*[NIL] = 0.

OS-SELECT(x, i) // ith smallest element in the subtree rooted at x

$k \leftarrow size[left[x]] + 1$
if $i = k$ **then return** x
if $i < k$
  then return OS-SELECT$(left[x], i)$
  else return OS-SELECT$(right[x], i - k)$

Example

OS-SELECT(*root*, 5)



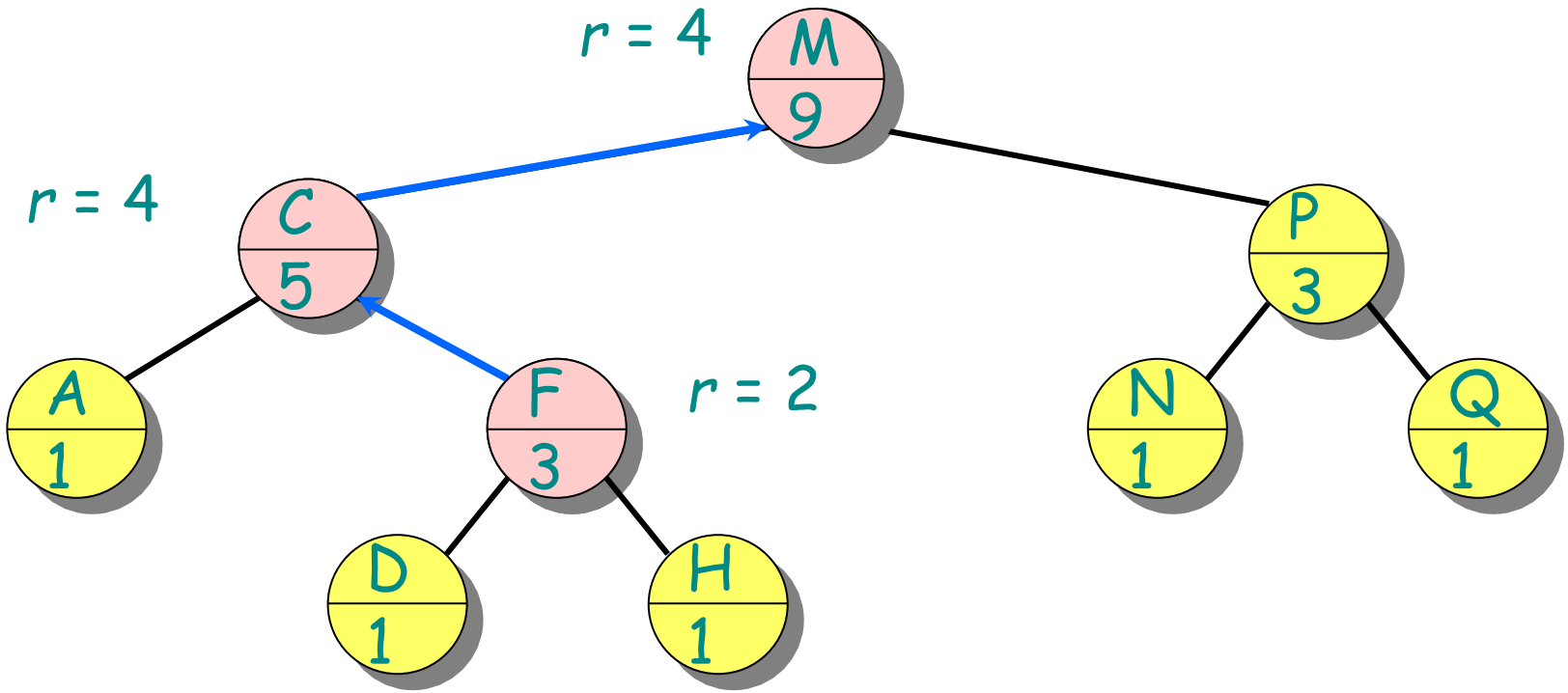
Running time = $O(h) = O(\lg n)$ for balanced binary trees.

Return the rank of an element

```
OS-RANK(T, x)
1  r = x.left.size + 1          // rank of x within the subtree rooted at x
2  y = x                        // root of subtree being examined
3  while y ≠ T.root
4       if y == y.p.right                    // if root of a right subtree ...
5            r = r + y.p.left.size + 1       // ... add in parent and its left subtree
6       y = y.p                              // move y toward the root
7  return r
```

Example

OS-RANK(T, F)



Running time = $O(h)$ = $O(\lg n)$ for balanced binary trees.

Data structure maintenance

Q. Why not keep the ranks themselves in the nodes instead of subtree sizes?

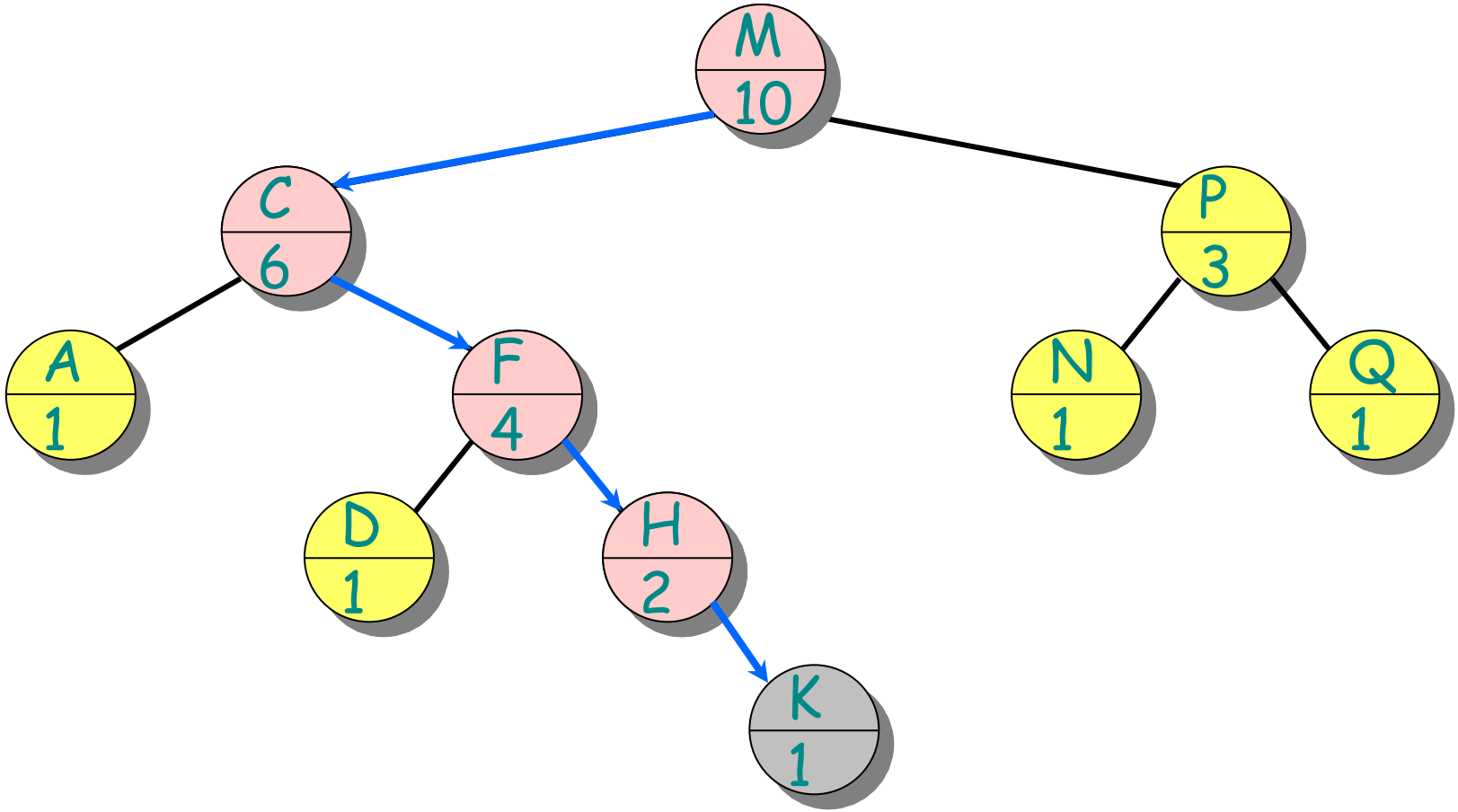
A. They are hard to maintain when the red-black tree is modified.

Modifying operations: INSERT and DELETE.

Strategy: Update subtree sizes when inserting or deleting.

Example of insertion

INSERT("K")

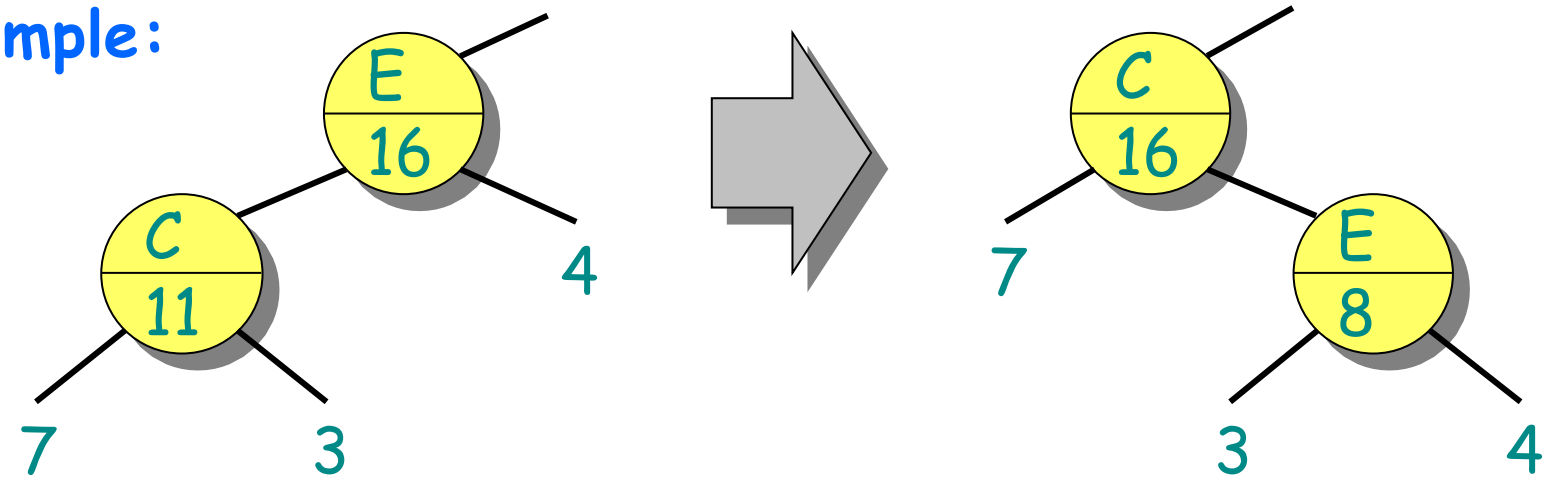


Handling rebalancing

Don't forget that INSERT and DELETE may also need to modify the binary search tree in order to maintain balance.

- *Rotations*: fix up subtree sizes in $O(1)$ time.

Example:



$\therefore$ INSERT and DELETE still run in $O(\lg n)$ time.

Data-structure augmentation

Methodology: (*e.g., order-statistics trees*)

1. Choose an underlying data structure (*balanced binary search trees*).
2. Determine additional information to be stored in the data structure (*subtree sizes*).
3. Verify that this information can be maintained for modifying operations (*INSERT, DELETE*).
4. Develop new dynamic-set operations that use the information (*OS-SELECT and OS-RANK*).

These steps are guidelines, not rigid rules.