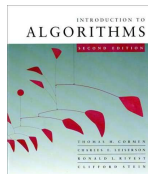


Augmented Data structures, and Dynamic Order Statistic



Original slides courtesy of Erik Demaine and Carola Wenk


Dynamic order statistics

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

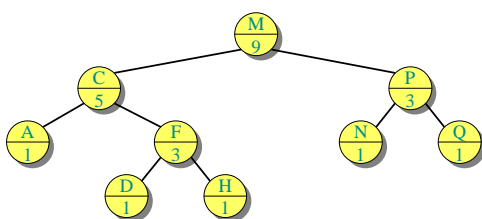
OS-RANK(x, S): returns the rank of $x \in S$ in the sorted order of S 's elements.

(the rank of the smallest element is 1)

IDEA: Use your favorite balance search tree (AVL, Red-Black, B-Trees, 2-3 trees, SkipList etc etc) 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) \triangleright returns the i 'th smallest element in the subtree rooted at x

$k \leftarrow size[left[x]] + 1$ $\triangleright k = \#keys \leq x$ in the subtree rooted at x

if $i = k$ then return x

if $i < k$

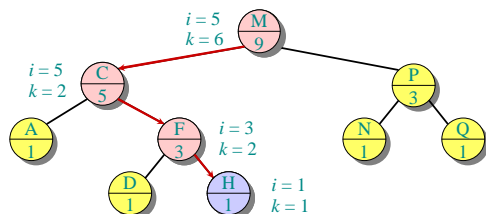
then return OS-SELECT($left[x], i$)

else return OS-SELECT($right[x], i - k$)

\triangleright when turning right, we skip k keys, all smaller than x .

Example

OS-SELECT($root, 5$)



Running time = $O(h) = O(\lg n)$ for a balanced tree, since this is the height of the tree.

Finding the rank of x

OS-Rank(T, x)

\triangleright Assume x already found, and the path from the root is known

$r \leftarrow size[left[x]] + 1$

\triangleright Recall that if $left[x]$ is NIL then its $size=0$

$y \leftarrow x$

While ($y \neq root(T)$) {

do if y is the **right** child of $parent[y]$

then $r += size[left[parent[y]]]$

$y \leftarrow parent[y]$

}

Return r

Data structure maintenance

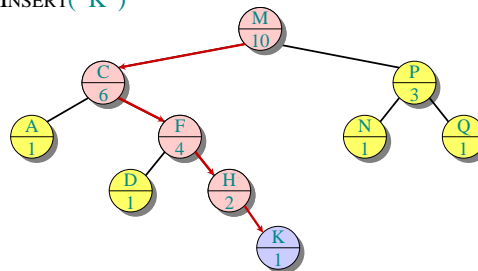
- Q.** Why not keep the ranks themselves in the nodes instead of subtree sizes?
- A.** They are hard to maintain when the tree is modified.

Modifying operations: INSERT and DELETE.

Strategy: Update subtree sizes when inserting or deleting.

Example of insertion

INSERT("K")

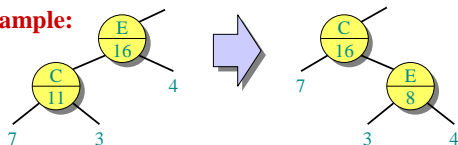


Handling rebalancing

If balancing is done via rotation, INSERT and DELETE may also need to modify the tree in order to maintain balance.

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

Example:



∴ 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 (*red-black 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 (*RB-INSERT, RB-DELETE — don't forget rotations*).
4. Develop new dynamic-set operations that use the information (*OS-SELECT and OS-RANK*).

These steps are guidelines, not rigid rules.