CMSC 341

Splay Trees
Splay Trees

Concept

– adjust tree in response to accesses to make common operations (insert, find, remove) efficient
– after access node is moved to root by \textit{splaying}

Performance

– amortized such that \( m \) operations take \( O(m \lg n) \) where \( n \) is the number of insertions (nodes in the tree)
Splay Operation

Traverse tree from node x to root, rotating along the way until x is the root

Each rotation

– If x is root, do nothing.
– If x has no grandparent, rotate x about its parent.
– If x has a grandparent,
  • if x and its parent are both left children or both right children, rotate the parent about the grandparent, then rotate x about its parent
  • if x and its parent are opposite type children (one left and the other right), rotate x about its parent, then rotate x about its new parent (former grandparent)
Node has no grandparent
Node and Parent are Same Side

Zig-Zig
Node and Parent are Different Sides
Zig-Zag
Operations in Splay Trees

insert
- first insert as in normal binary search tree
- then splay inserted node
- if there is a duplicate, the node holding the duplicate element is splayed

find
- search for node
- if found, splay; otherwise splay last node accessed on the search path
Insertion in order into a Splay Tree
Operations on Splay Trees (cont)

remove

– splay element to be removed
  • if the element to be deleted is not in the tree, the node last visited on the search path is splayed
– disconnect left and right subtrees from root
– do one of:
  • splay max item in $T_L$ (then $T_L$ has no right child)
  • splay min item in $T_R$ (then $T_R$ has no left child)
– connect other subtree to empty child of root
Performance of Splay Trees

insert
  – regular BST insertion -- $O(\text{height})$
  – splay: $O(1)$ for each rotation, $O(\text{height})$ rotations