CSE 326
Lecture 11

Spring Quarter 1998

Splay Trees

- Splay trees are self adjusting binary search trees invented by Sleator and Tarjan around 1985.
- Basic principle of splaying: When accessing a node move it to the root by a sequence of rotations.
- Properties of splay trees - assume $m \geq n$ accesses in a tree of size n

  - No additional fields to ordinary binary search trees are needed. Weiss uses parent pointers but they are not strictly needed.

  - Good amortized time complexity: O(m log n), that is, O(log n) per access on average.
    Worst case time complexity for an access could be bigger than log n.

  - Good locality: Suppose only $k \geq 2$ distinct items are accessed in the m accesses. Time is O(m log k + n log n).

  - Insertion of n items in order into an empty tree is O(n) time.

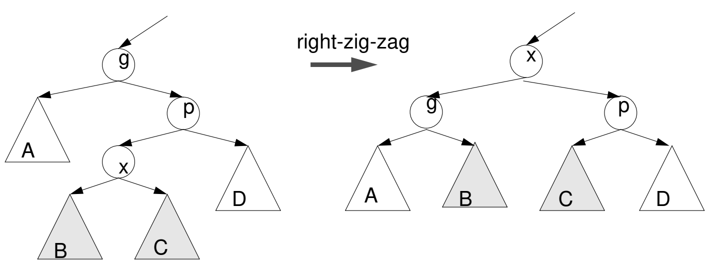
  - Deletion of n items in order from a tree of size n is also O(n) time.

Find in a Splay Tree

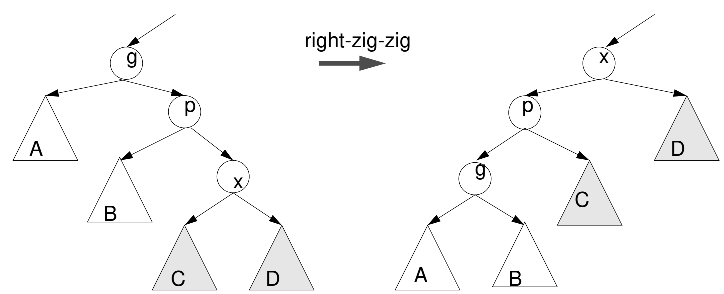
- Find(x: key):

  1. do a normal search to find the node n such that n.data = x.

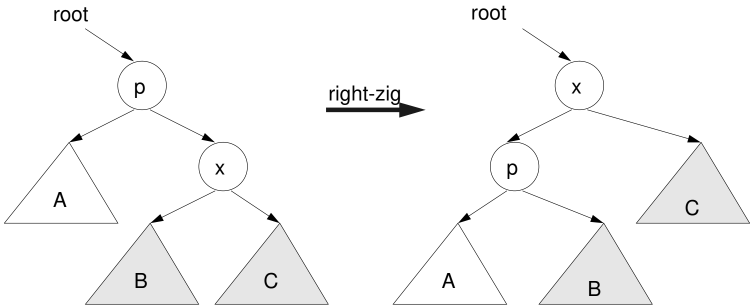
  2. Move n to the root by a series of zig-zag, zig-zig rotations, followed by a final zig if needed.
- Zig-zag is just our old friend double rotation.



Zig - Zig



Plain Old Zig

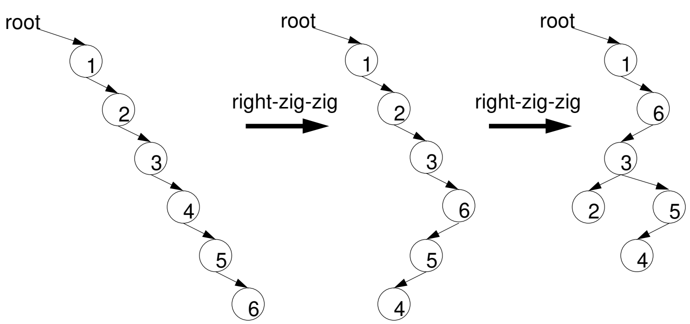


Why Does Splaying Help

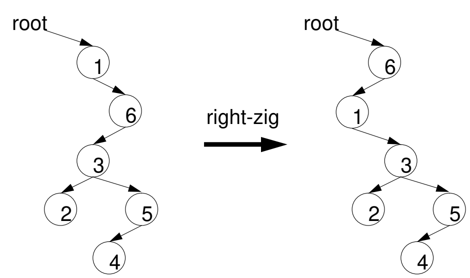
- If a node n on the access path is at depth d before splaying then it is at depth approximately d/2 after splaying.

  Exceptions are the root, the child of the root and the node splayed.
- Overall, nodes which are below nodes on the access path tend to move closer to the root.

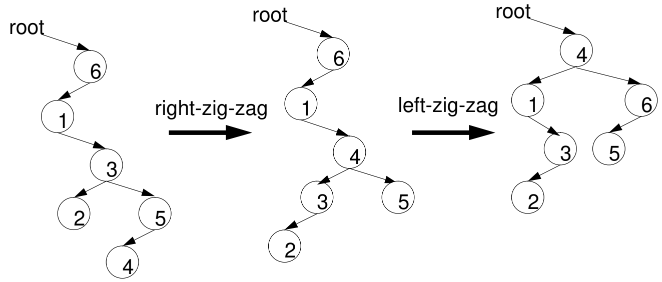
Example - Find 6



Find 6 Continued



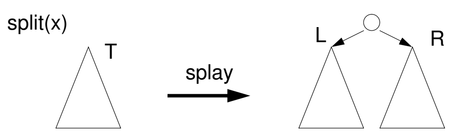
Find 4



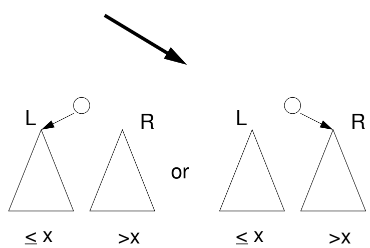
Insertion and Deletion

- In order to define insert and delete in splay trees we introduce the operations of split and join.
- Split(x) returns two binary search trees L and R where L contains all nodes with keys $\leq$ x and R contains all nodes > x
- Assuming the keys in L are all smaller than the keys in R, Join(L,R) returns a single tree with the same data L and R combined.

Split

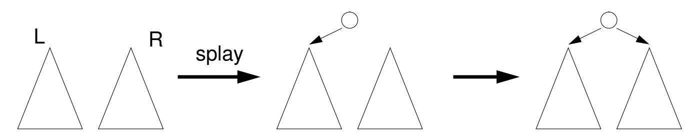


To splay do a find on x: if x is in T then splay it to the root, otherwise splay the last node found to the root. After splaying split the tree at the root.



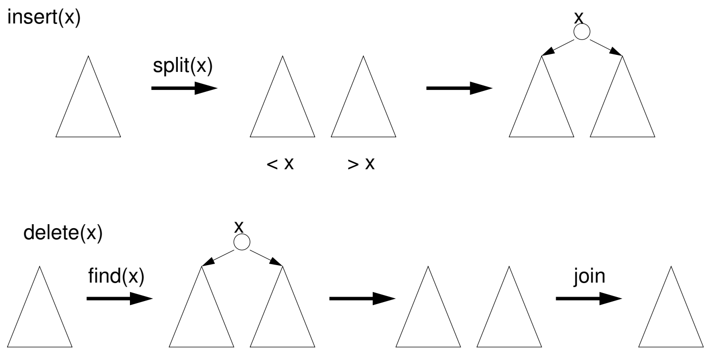
Joining Two Trees

- Join(L,R) is only done if the keys in L are smaller than the keys in R.

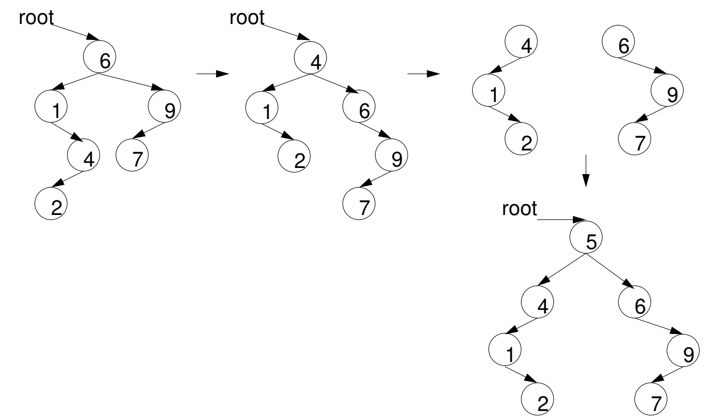


Splay on the maximum element in L then attach R as a right child.

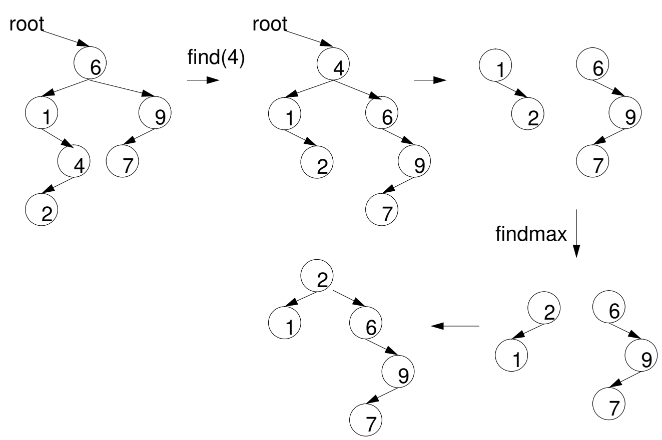
Insert and delete



Example of Insert 5



Delete 4



Splay Tree Notes

- Splaying can be done top down, which is probably the right way to do it.
- There are alternatives to the standard split-join implementation of insert and delete.

  For example, for insertion, just do a normal binary search tree insertion then splay the node to the root.
- The splay tree is one of the most popular binary search tree techniques.

  + It is relatively simple.

  + No extra fields are needed.

  + It has excellent temporal locality properties. Keys that are accessed often end up near the root.

  + $m \geq n$ operations costs $O(m \log n)$, good amortized time complexity

  - All those rotations add a significant constant factor.

  - Worst case for a single operation can be costly.