CSE 326
Lecture 16

Spring Quarter 1995

Midterm Notes

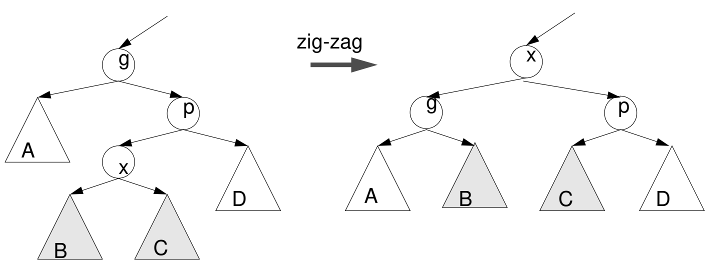
- Friday, May 5th
- 50 minutes
- Attach your help sheet - handwritten, on 8 1/2 x 11 sheet of paper, use both sides.
- Question and Answer Session - Wednesday 3:30 - 4:20
- Kinds of questions on the exam

  1. Demonstrate knowledge of algorithms and methods.

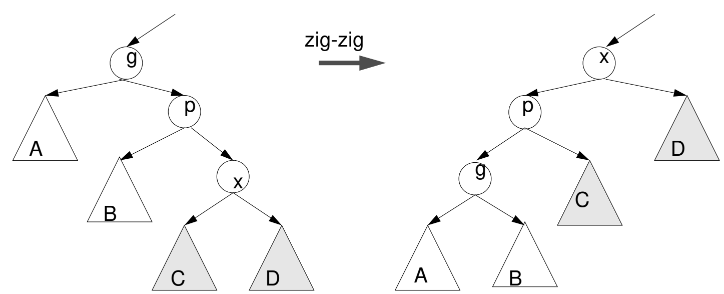
  2. Problem solving - may require short pieces of pseudocode
- What will be covered

  - Recursion vs. Iteration
  - Time complexity analysis
  - Stacks, Queues, Lists, Multilists including adjacency lists
  - General Trees - tree traversals and their applications
  - Binary Search Trees, AVL and Splay trees.

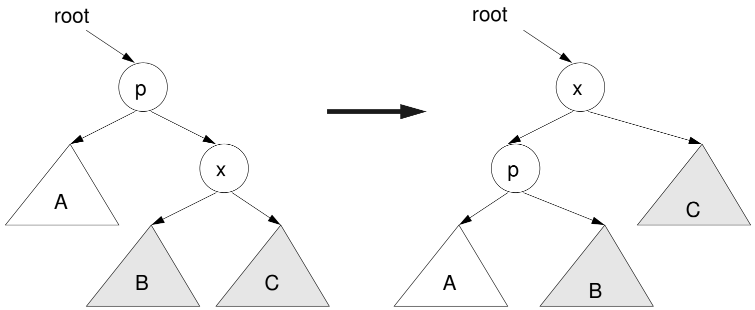
Splaying Steps



Zig - Zig



Plane Ole Zig



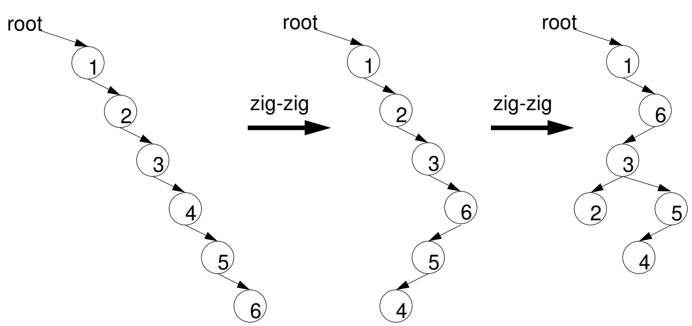
Splaying

- Splaying a node to the root is done by a series of zig-zag, and zig-zig steps moving the node two levels at a time.
- A final zig may be needed if the node starts out on an odd level
- Splaying has a good effect because nodes on the path to the root are moved approximately half the distance to the root. They, in turn, drag their subtrees with them.
- Good properties:

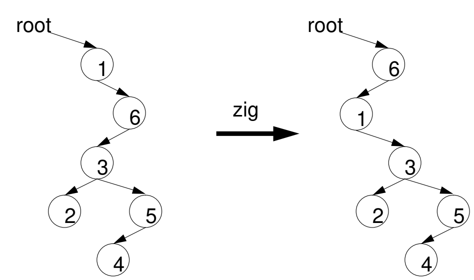
  1. O(log n) amortized time per splaying operation.

  2. Even better amortized time per splaying operation if the number of items accessed are small or are close together in the overall ordering.

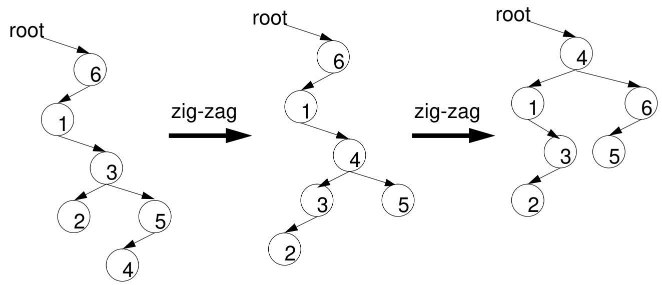
Example - Find 6



Find 6 Continued

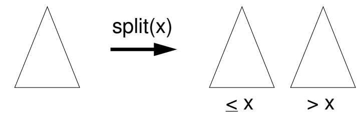


Find 4

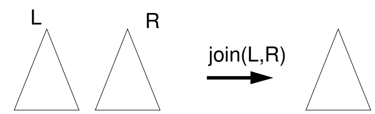


Insertion and Deletion

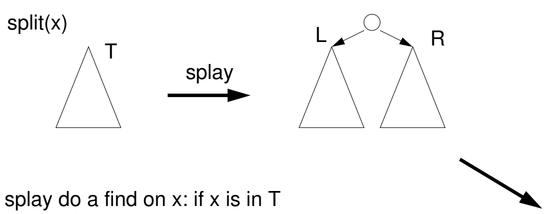
- In order to define insert and delete in splay trees we introduce the operations of split and join.
- Split(x) returns two binary search trees L and R where L contains all nodes with keys ≤ x and R contains all nodes > x



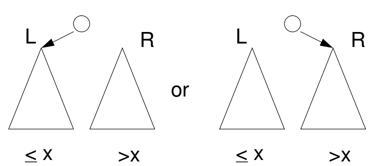
- Assuming the keys in L are all smaller thant the keys in R, Join(L,R) returns a single tree with the same data L and R combined.



Split

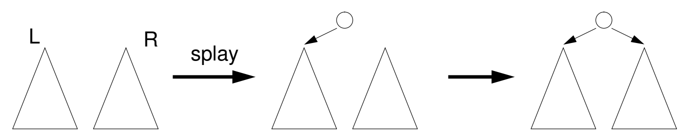


To splay do a find on x: if x is in T then splay it to the root, otherwise splay the last node found to the root. After splaying split the tree at the root.



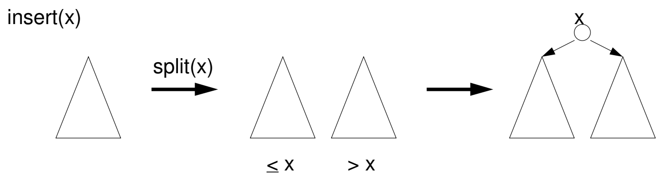
Joining Two Trees

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

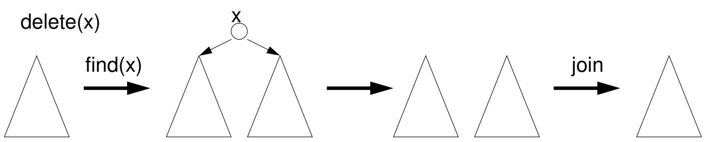


Splay on the maximum element in L then attach R as a right child.

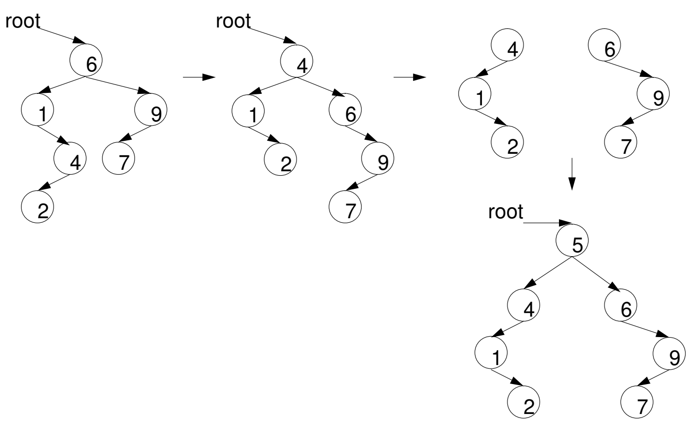
Insert



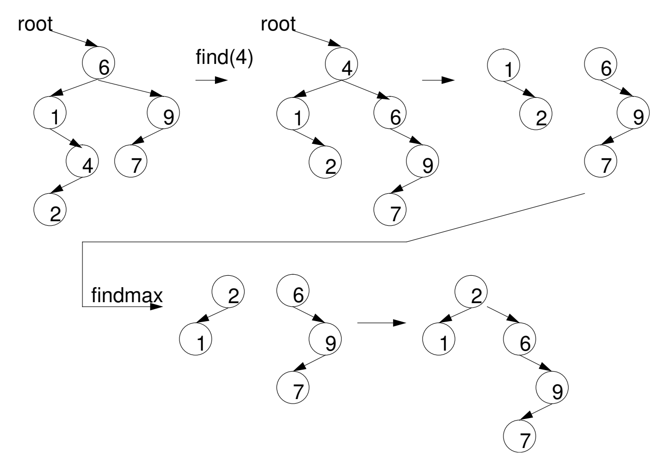
Delete



Example of Insert 5



Delete 4



Splay Tree Notes

- The book describes a method using parent pointers, but strictly speaking they are not needed.
- 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 currently the most popular binary search tree technique.

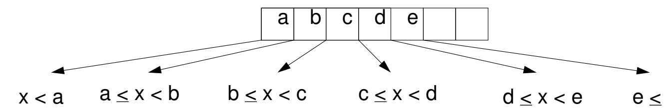
  1. It is relatively simple.

  2. No extra fields are needed.

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

B - Trees

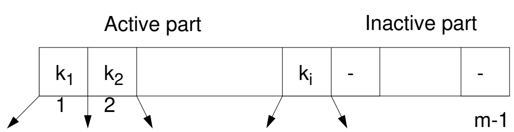
- B-trees are multiway trees which are not binary



- B-tree nodes contain search keys only, actual data is stored at the leaves.
- B-tree developed for database applications where minimizing the number of disk accesses is important. B-tree nodes are large enough to balance the disk access time with the internal search time within a B-tree node.
- B-trees are generally useful in a virtual memory environment to minimize page faults and perhaps even in a RISC environment where minimizing cache misses is important.

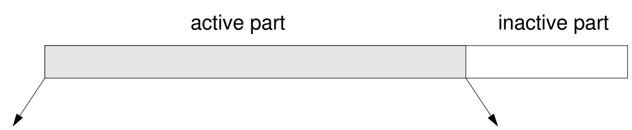
Properties of B-Trees of Order m

- The root is either a leaf or has between 2 and m children
- All non-leaf node (except the root) have between | m/2 | and m children.
- All leaves are at the same depth and contain between | m/2 | and m - 1 keys.
- A node will have keys $k_1, k_2, k_3, ..., k_i$ where | m/2 | < i < m

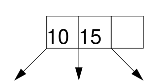


Find in a B-Tree

- If m is large then binary search within a B-tree node.



- If m is small then linear search is fine.

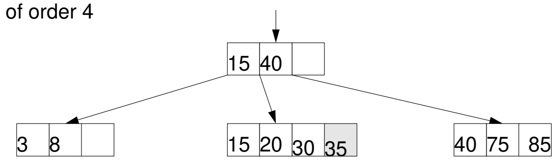


Insertion in a B-tree

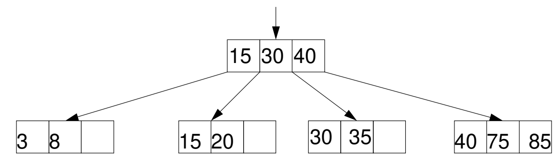
- Begin by inserting in the proper leaf.
- If the leaf ends up with m items, it has overflowed.
- An overflowed leaf node is split into two nodes with | m/ 2 | in one and | m/2 | in the other. A new child is added to the parent.
- An overflowed internal node with m+1 children is split into two nodes, one with | (m+1)/2 | children and the other with | (m+1)/2 | children. A new child is added to the parent.
- If overlow propogates to the root, then the node is split into two and a new root with two children is formed.

Insertion with Overflow - Leaf Case

B-tree of order 4

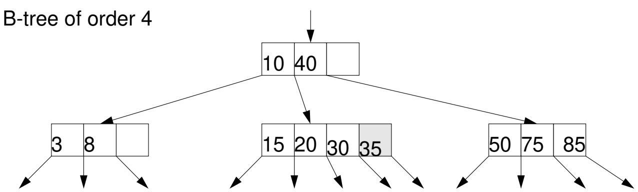


Splitting a node

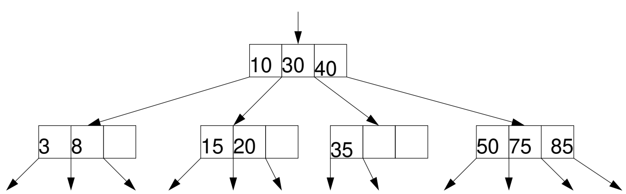


Insertion with Overflow - Internal Node Case

B-tree of order 4



Splitting a node



Example of Insertion

- Insert 1,2,3,4 5,6 7,8, ... into a B-tree of order 4