Splay Trees

Class 23

AVL Summary

- maintain height-balanced BST
- every insert or remove can potentially unbalance the tree
- an unbalanced tree requires one rotation to rebalance
- there are 4 types of rotations

AVL Tree Problems

- AVL trees are cool
- they work well
- they have $O(\lg n)$ access time to any element
- but . . .

AVL Tree Problems

- AVL trees are cool
- they work well
- they have $O(\lg n)$ access time to any element
- but ...

- they have two big problems
  1. they require the cumbersome maintenance of height or balance
  2. they do not take into account real-world data access patterns

Access Patterns

- in human activity, data access exhibits
  1. temporal locality of reference
  2. spatial locality of reference
- temporal: the most likely data to be accessed next was recently accessed
- spatial: the most likely data to be accessed next is close to data that was recently accessed
- the AVL tree completely fails to take either into account

Splay Tree

- the splay tree is a BST that
  - is simpler than AVL
  - does take access patterns into account

Splay

- in the AVL tree, a rotation is performed sometimes after insert or remove
- in the splay tree, a splay operation is performed after every find, insert, and remove
- the splay process ends when the distinguished node becomes the root

the distinguished node is:

find
: the node containing the searched-for element, or the last node accessed if contains returns false

insert
: the newly inserted node

remove
: the parent of the removed node

Splay Rotations

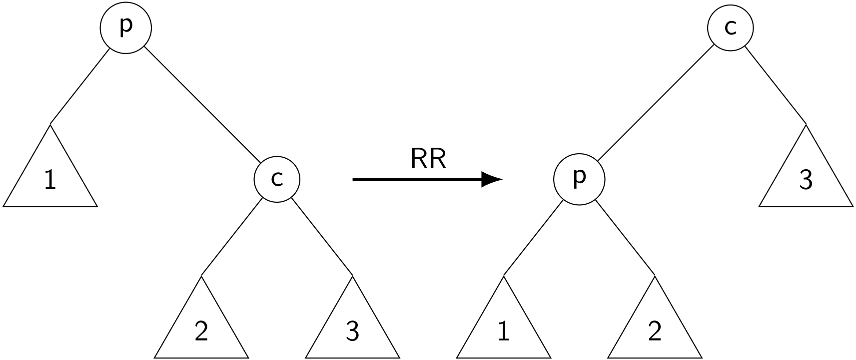
- the splay process consists of a series of rotations

- four of the splay rotations are identical to AVL rotations
- two splay rotations are specific to the splay tree and do not occur in AVL

- the splay process continues until distinguished node is root

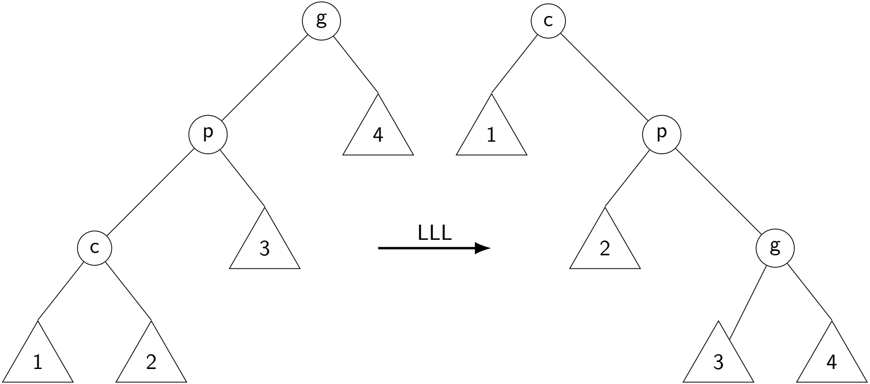
RR

this is identical to AVL



LLL

this does not occur in AVL

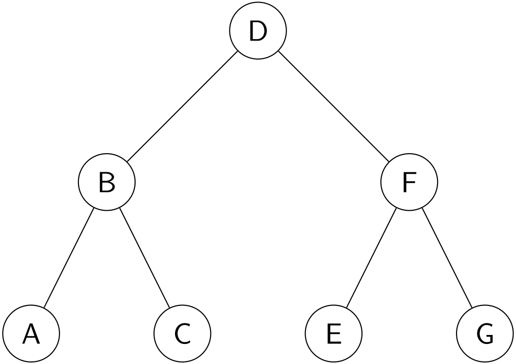


The Other Two

- RL and LR are identical to the RL and LR rotations of AVL

Splay Example

splay on A, then subsequently splay on F



Splay Tree Analysis

- the analysis of splay trees is very complex
- it is analyzed using amortized cost
- amortization means the cost is spread out over time
- when a splay tree path is long, access exceeds lg n comparisons
- but the subsequent rotations tend to make future operations fast
- ideally future operations are much faster than lg n, approaching constant

Amortized Cost Analysis

- the simplest form of amortized cost analysis is empirical
- for a given very long sequence of accesses, run a simulation
- compare AVL and splay tree performance
- for the splay tree, random accesses never exceed $\lg n$ per operation averaged over multiple operations
- typical real-life access patterns perform much better
- malicious access patterns can make splay approach an average of n (but never worse, of course)