CS 466 Fall 2019 Lecture 4 - Splay Trees

Anna Lubiw
Univ. Waterloo

Splay Trees

Self-adjusting data structure — alter structure on a query.

e.g. move-to-front heuristic for a list

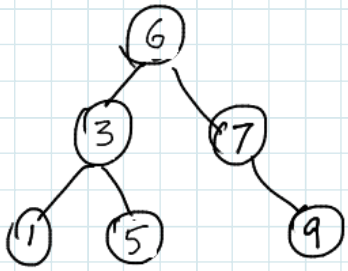
Note – we will do work ahead of time in hopes of saving later
— the opposite approach from procrastinating (as in Lazy Binomial Heaps)

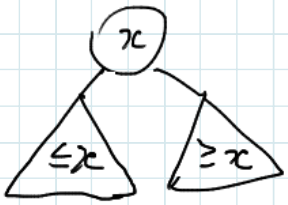
Recall Dictionary

- keys from totally ordered universe
- operations: Insert, Delete, Search

Binary Search Tree

e.g. search for 4





Insert – make new node where search fails

Delete – easy for node w/ 0 or 1 child

– else replace value by predecessor or successor and delete that node (which has 0 or 1 child)

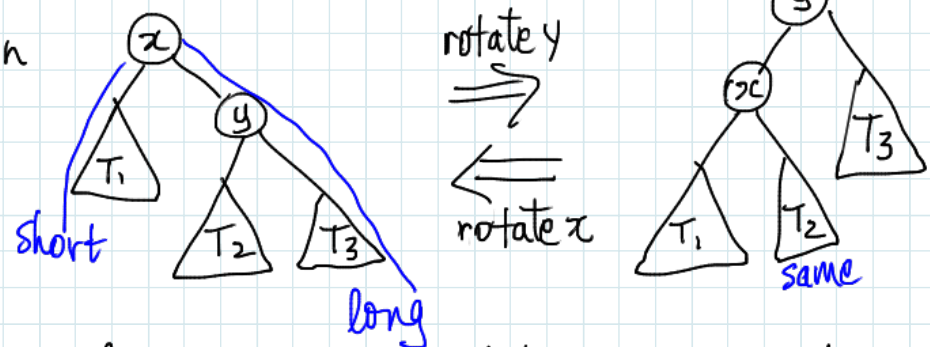
All operations take $O(h)$ h = height of tree

Balance – keep $h \in O(\log n)$

AVL, red-black etc. [also 2-3 but they are not binary]

technique for rebalancing

rotation



must keep balance info and update as you rotate

CS 466 Fall 2019 Lecture 4 - Splay Trees

Anna Lubiw
Univ. Waterloo

Splay Tree [Sleator & Tarjan 1985]

- $\Theta(\log n)$ amortized cost per operation, $\Theta(n)$ worst case
- easier to implement than AVL/red-black trees
- don't keep balance info – the tree may become unbalanced
- Careful! repeated searching for a deep node costs $\Theta(n)$ each time

  Fix – adjust the tree whenever a node is accessed.

  Hence "self-adjusting".

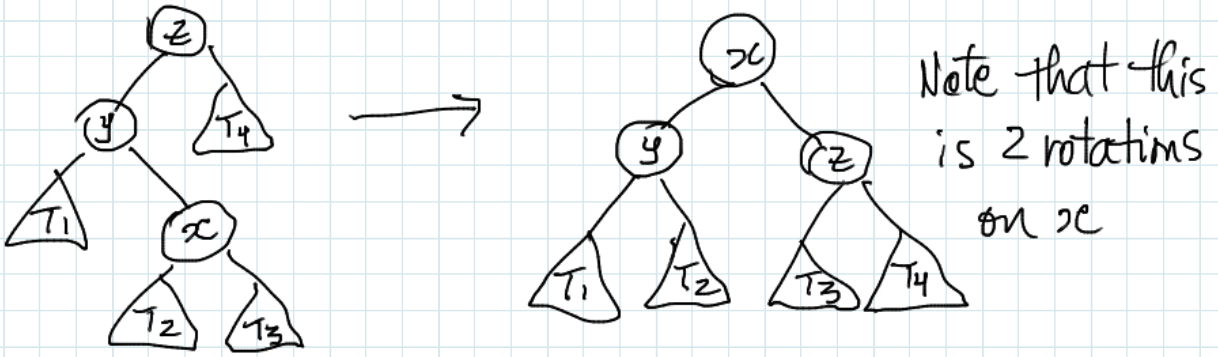
First idea: do [single] rotations to move the searched node to root. [show this gives amortized search cost $\Omega(n)$]

Splay operation

Splay(x)

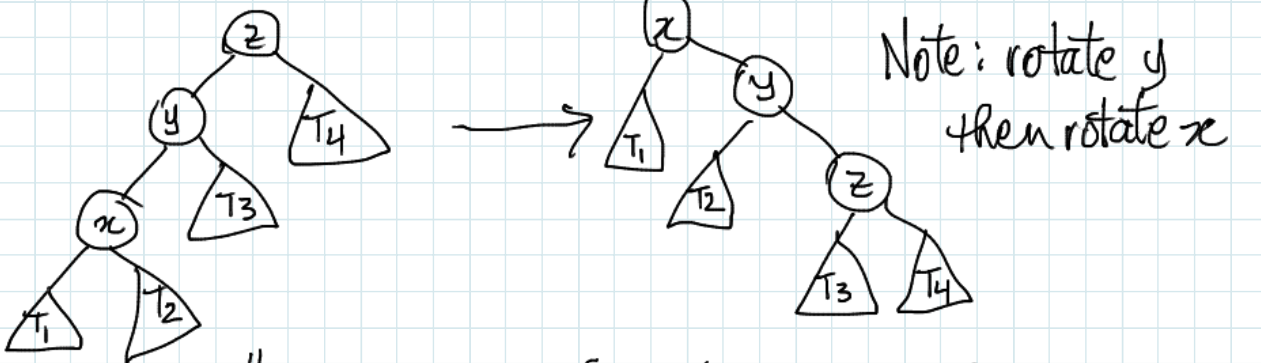
repeat til x is root

case 1 "zig-zag"



Note that this is 2 rotations on x

case 2 "zig-zig"



Note: rotate y then rotate x

case 3 "zig" At child of root, do single rotation.

CS 466 Fall 2019 Lecture 4 - Splay Trees

Anna Lubiw
Univ. Waterloo

Splay Tree Algorithm

Search – after finding x call Splay(x)

Insert – do standard insert then splay new node

Delete – do standard delete of key,
splay parent of node that gets deleted
(the node with 0 or 1 child)

Examples (next slide)

Amortized Analysis of Splay Trees

use Potential Method (reminder on slide)

Plan – use 3 steps:

1. Define potential and analyze 2-operations (zig-zag, zig-zig, zig)
2. Analyze Splay
3. Analyze Search, Insert, Delete

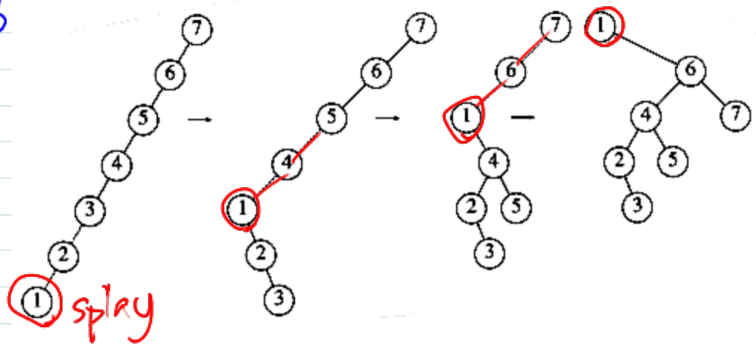
Intuition: if node x is far from root Splay(x) is expensive but potential will pay for it.
(potential will be higher for tree with large height)

CS 466 Fall 2019 Lecture 4 - Splay Trees

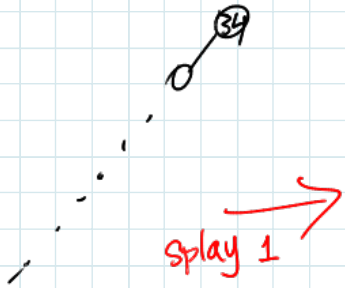
Anna Lubiw
Univ. Waterloo

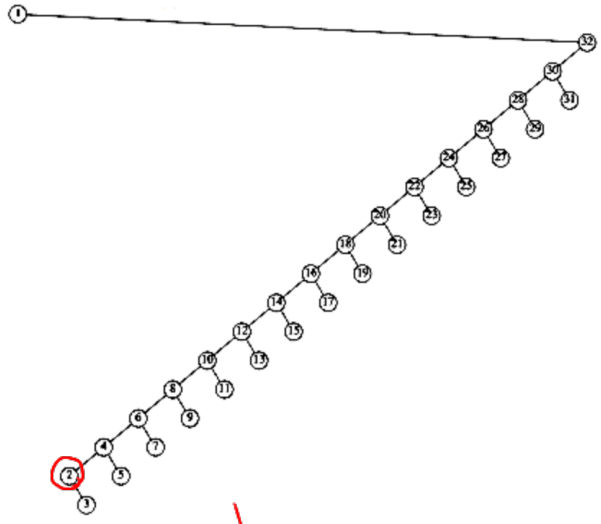
Examples

1

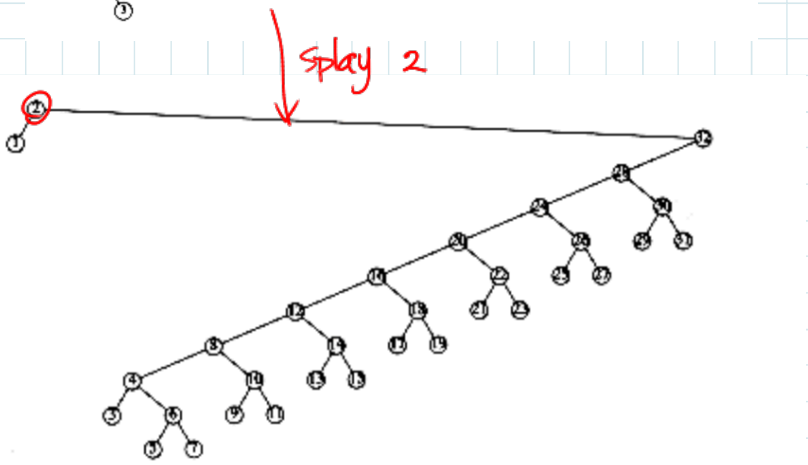


2

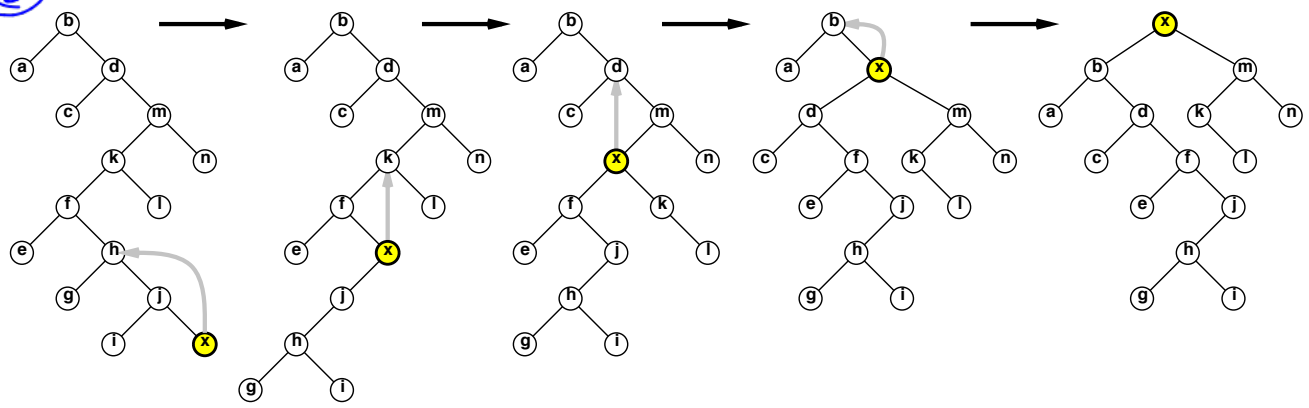




1



3



CS 466 Fall 2019 Lecture 4 - Splay Trees

Anna Lubiw
Univ. Waterloo

Reminders (repeat from Lecture 3)

Definition sequence of m operations
total cost is $T(m)$
Then the **amortized cost** per operation is $T(m)/m$

cost(i) = [true] cost of i^{th} operation
charge(i) = [artificial] charge for i^{th} operation

Φ_i – potential after i^{th} operation.

$$\Phi_i = \Phi_{i-1} + \text{charge}(i) - \text{cost}(i)$$
$$\text{charge}(i) = \text{cost}(i) + \underbrace{\Phi_i - \Phi_{i-1}}_{\Delta\Phi_i}$$

Thm If final potential $\geq$ initial potential
then amortized cost $\leq$ max charge.

CS 466 Fall 2019 Lecture 4 - Splay Trees

Anna Lubiw
Univ. Waterloo

Define $D(x)$ = # descendants of x (including x)

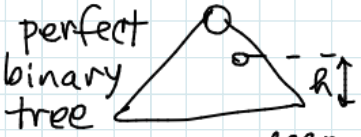
$r(x) = \log D(x)$

$\Phi(T) = \sum_x r(x)$

EX. what is max and min potential of tree of n nodes?

Path: $r = \log n$, $\log(n-1)$, ..., $\log 1$

$\Phi = \sum_{i=1}^{n} \log i = O(n \log n)$



node x at height h has $r(x) = h$

$\Phi = \sum_{h=1}^{\log n} h \frac{n}{2^h}$ number of nodes at height h

$= n \sum \frac{i}{2^i}$ — constant

$= O(n)$

Analyze a Z operation at node x.

$r(x)$ - current rank $r'(x)$ - new rank

Claim: amortized cost (charge) of one Z operation on x is

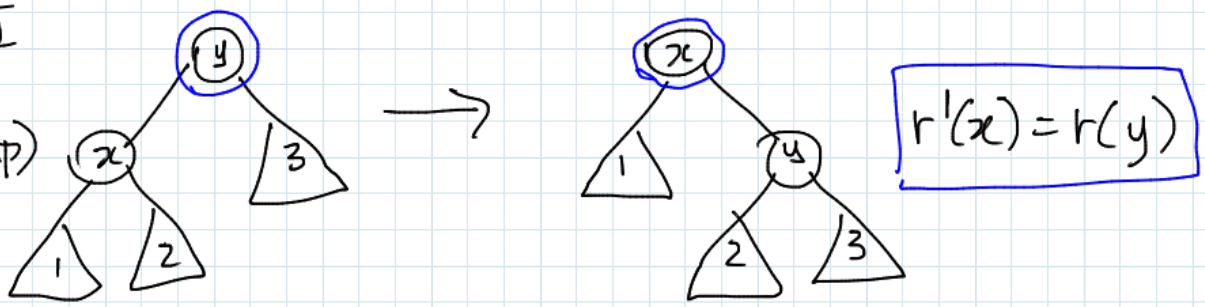
$\leq 3(r'(x) - r(x))$ for zig-zig, zig-zag

$+1$ for zig

Note for bounding Splay: this sum will telescope!

Proof

zig. (warm up)



$r'(x) = r(y)$

charge = cost + $\Delta\Phi$ (change in potential)

CS 466 Fall 2019 Lecture 4 - Splay Trees

Anna Lubiw
Univ. Waterloo

amortized cost (charge) $= 1 + (r'(x) + r'(y) - r(x) - r(y))$

1: cost of rotation

$r'(x) + r'(y) - r(x) - r(y)$: change in potential

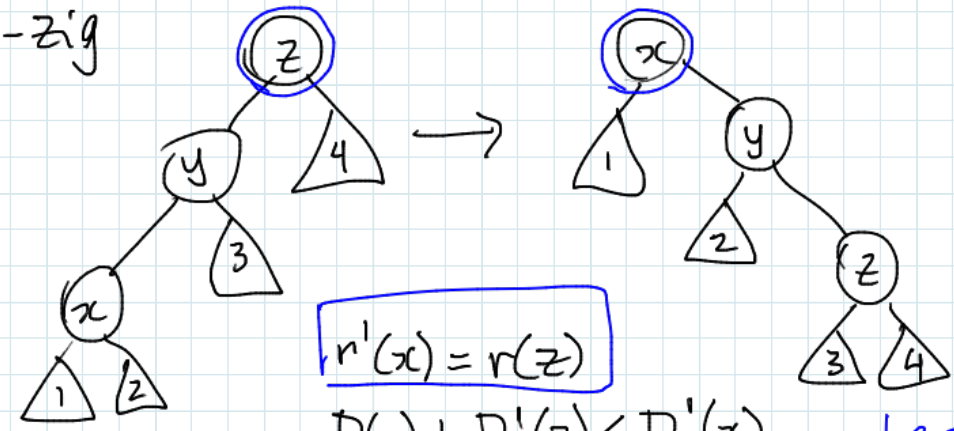
$-r(y)$ cancel

$= 1 + r'(y) - r(x)$

$\leq 1 + r'(x) - r(x) \leq 1 + 3(r'(x) - r(x))$

because y is child of x

zig-zig



$r'(x) = r(z)$

$D(x) + D'(z) \leq D'(x)$ — because no node is double-counted

amortized cost $= \underbrace{2}_{\text{2 rotations}} + (r'(x) + r'(y) + r'(z) - r(x) - r(y) - r(z))$

change in potential.

$-r(z)$ cancel

$= 2 + r'(y) + r'(z) - r(x) - r(y)$

$\leq 2 + r'(x) + r'(z) - 2r(x)$

$r'(y) \leq r'(x)$

$-r(y) \leq -r(x)$

We want this $\leq 3(r'(x) - r(x))$

i.e. want $2 + r(x) + r'(z) \leq 2r'(x)$ (*)

i.e. $\log D(x) + \log D'(z) \leq 2 \log D'(x) - 2$

given $D(x) + D'(z) \leq D'(x)$

For this, we need a property of logs

CS 466 Fall 2019 Lecture 4 - Splay Trees

Anna Lubiw
Univ. Waterloo

Fact: by convexity of log
if $x, y > 0$ $x + y \leq 1$ then the log fn $\log x + \log y$ is max'd at value -2 when $x = y = \frac{1}{2}$

Hence, for $a, b > 0$ $a + b \leq c$

$$\log \frac{a}{c} + \log \frac{b}{c} \leq -2$$

$$\log a + \log b \leq 2 \log c - 2$$

Now $D(x) + D'(z) \leq D'(x)$

so $\log D(x) + \log D'(z) \leq 2 \log D'(x) - 2$

i.e. $r(x) + r'(z) \leq 2r'(x) - 2$ which gives (*)

Zig-Zag - similar (needs same log trick)

Part (2) Analyze Splay(x)

<u>Lemma</u> Tree T, root t, node x
amortized cost of splay(x) is
$\leq 3(r(t) - r(x)) + 1 \leq 3r(t) + 1 \in O(\log n)$
because $r(t) \leq \log n$

<u>Pf</u> Add up the amortized cost of each step of the splay — telescoping sum?

Let $r_i = r(x)$ after ith step of splay. So $r_0 = r(x)$ $r_k = r(t)$ where k is last step.

amortized cost is $\sum_{i=1}^{k} 3(r_i - r_{i-1}) + 1$ by claim above.
$= 3(r_k - r_0) + 1$

CS 466 Fall 2019 Lecture 4 - Splay Trees

Anna Lubiw
Univ. Waterloo

Thm Amortized cost of Insert, Search, Delete in splay tree is $O(\log n)$

Proof charge = charge(splay) + cost(operation) + $\Delta\Phi$

charge(splay): $O(\log n)$ by Lemma above

$\Delta\Phi$: change in Φ outside of splay.

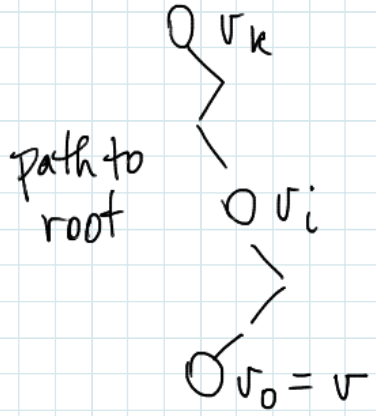
cost of each operation (walk down tree)
$\leq$ cost of ensuing splay.

Only thing we haven't taken into account is change in potential due to changes outside splay.

Search - only splay changes potential

Delete - removing a node decreases potential

Insert - adding new node v increases ranks of all ancestors of v (might be n of them)



$D'(v_i) = 1 + D(v_i) \leq D(v_{i+1})$

$r'(v_i) \leq r(v_{i+1})$

change in potential is

$$\sum_{i=0}^{k} r'(v_i) - \sum_{i=1}^{k} r(v_i)$$

$$\leq r'(v_k) \leq \log n$$

CS 466 Fall 2019 Lecture 4 - Splay Trees

Anna Lubiw
Univ. Waterloo

Dynamic Optimality Conjecture

Conjecture [Sleator & Tarjan '85] Splay trees are optimal (within a constant) in a very strong sense:

Given a sequence of items to search for

$x_1 \; x_2 \; \cdots \; x_m$

Let OPT be min cost of doing these searches + any rotations you like on BST.

(charge 1 for following tree pointer (parent → child or child → parent)
charge 1 for rotation)

Conj. cost of splay tree is $O(OPT)$.

Note that for OPT, you get to look at the seq. of searches first and plan ahead. "offline", in contrast with "online" which is usual for data structures.

We will cover online algorithms later in the course

Note 2. OPT can adjust the tree so it's even better than the static optimal binary search trees you may have seen in CS341.