Splay Trees

15-750
1/22/10

Design - Goals A BST s.t.

1) No stored balance info.
2) The tree is "self-balancing".

Properties of a splay Tree.

1) Search(x) has side effect of rotating x to root.
   (Splay(x))
2) Subtle rotation rules.
3) Uses amortized analysis

Splay Trees

Two ways to describe them.

1) rotation rules

2) as Treap and rules to change priorities (insertion order)

eg

insertion order

(- - - b - - - a - - - -) (- - - ab - - - - -)

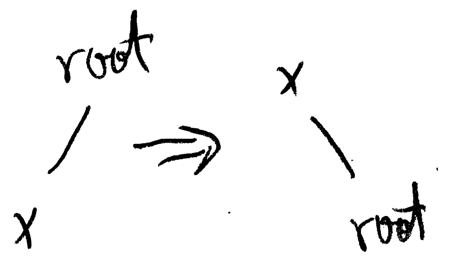
move a in front of b.

3 Rotation Rules for Splay

__Goal__ Def Splay(x) $\equiv$ rotate x to root.

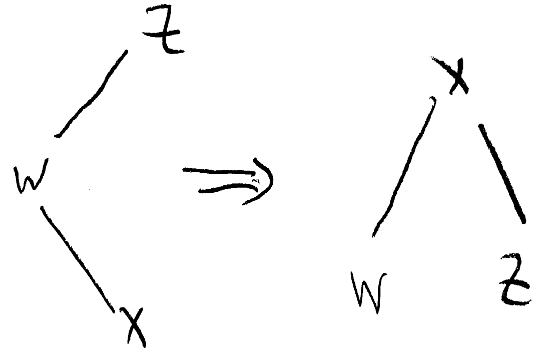
__Def__ Splay(x,T)

1) Zig $\equiv$



(move x to front)

2) Zig-Zag



(move x in front of z)

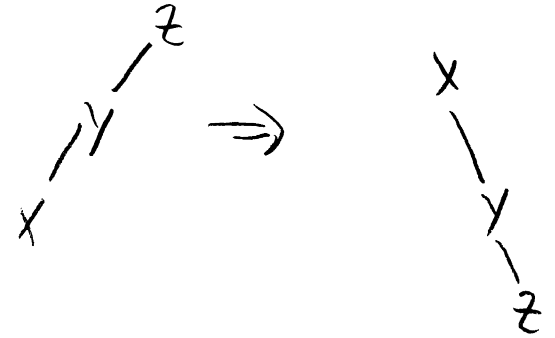
or

1) (move w in front of z)

2) (x w)

The interesting case

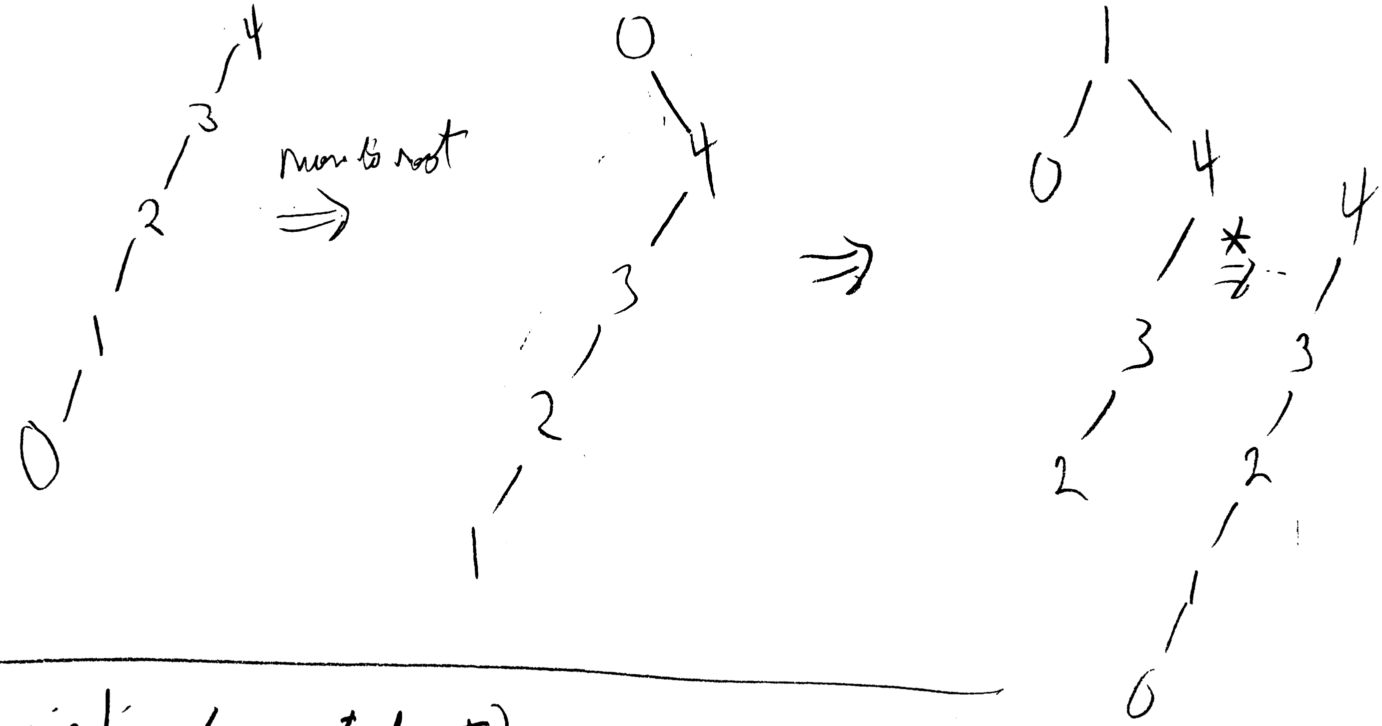
3) Zig-zig



$P(x) > P(y) > P(z)$

(y in front of z
x " " y) Same as 2)

Why rule 3)? Not using zig-zig rule! (i.e. use only move to front)



Priorities (move to front)

rotations

4, 3, 2, 1, 0) $n-1$

0, 4, 3, 2, 1) $n-1$

1, 0, 4, 3, 2) $n-2$

2, 1, 0, 4, 3

⋮

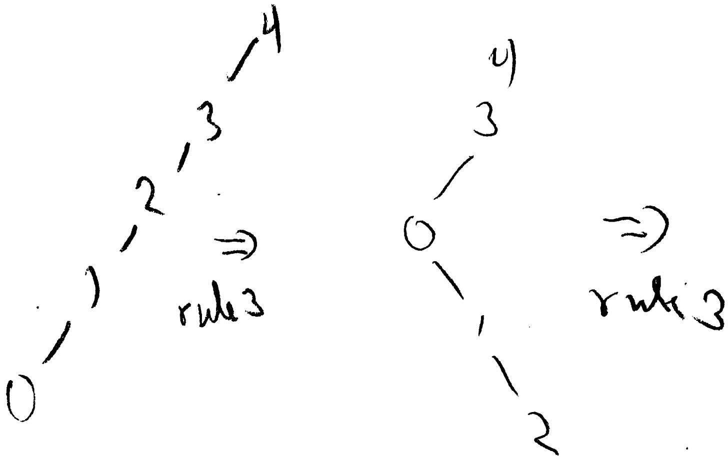
4, 3, 2, 1, 0) 1

total $\simeq \frac{n^2}{2}$ ⇒ average $\frac{n}{2}$

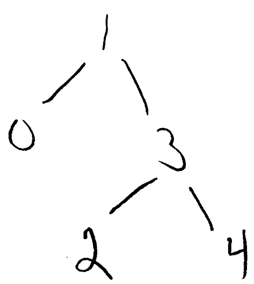
Using all 3 rules

input (4,3,2,1,0)

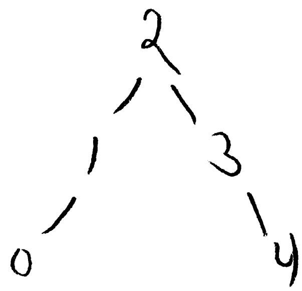
splay(0)



splay(1)

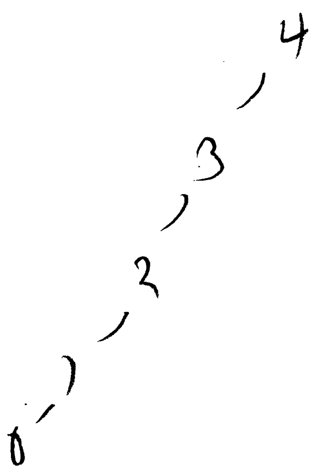


splay(2)



splay(3)

splay(4)

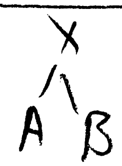


Dictionary Ops

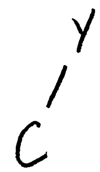
Claim

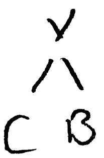
Insert $(T, x) \equiv$ 1) VanillaInsert (T, x) $\quad AC = O(\log n)$

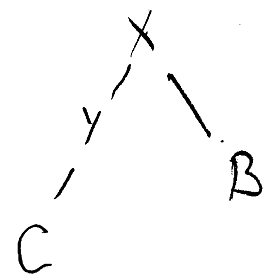
2) Splay (T, x)

Delete $(T, x) \equiv$ 1) Splay (T, x) giving 

2) if $A = \emptyset$ then return B

else splay (A, largest element) gives 

return 



Lookup $(T, x) \equiv$ 1) Search (T, x)

2) if x found then Splay (T, x)

else Splay (T, Parent(x)) (why?)

Amortized Analysis of Splay

Potential Method $\Phi(T)$

Goal: Φ large for unbalanced trees

Def $S(x)$ = # nodes in subtree rooted at x

Def $r(x) = \lfloor \log_2(S(x)) \rfloor$ **Rank**

$$\Phi(T) = \sum_{x \in T} r(x)$$

$T \equiv$ Line then $\Phi(T) = \sum_{i=1}^{n} \lfloor \log_2 i \rfloor = \Theta(n \log n)$

$T =$ balanced then $\Phi(T) = \Theta(n)$

Claim $\forall T$ $\Phi(T) = \Omega(n)$ & $\Phi(T) = O(n \log n)$

$$\text{Amortized Cost} = \text{Unit-Cost} + \Delta\Phi$$

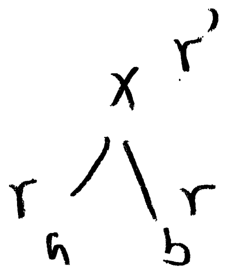
$$\text{Unit-Cost} \equiv \#\text{ rule applications} = \left\lceil \frac{\text{depth}}{2} \right\rceil \approx 2\,\#\text{rotations}$$

Access Lemma $AC(\text{splay}(x)) \leq 3(r(\text{root}) - r(x)) + 1$

Cor $AC \leq 3\log n + 1$

Simple facts

Rank Rule



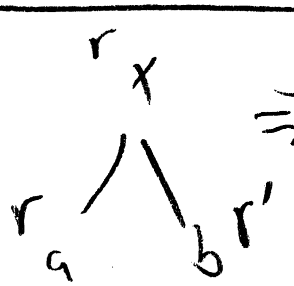
$\Rightarrow r' > r$

pf $s(a), s(b) \geq 2^r$

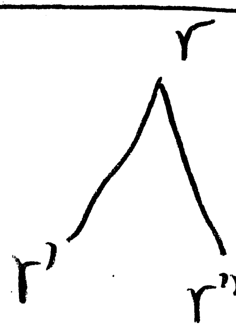
$\Rightarrow s(x) \geq 2^{r+1}$

$\Rightarrow r(x) \geq r+1$

ContraPositive



$\Rightarrow r' < r$ (or)



$2r > r' + r''$

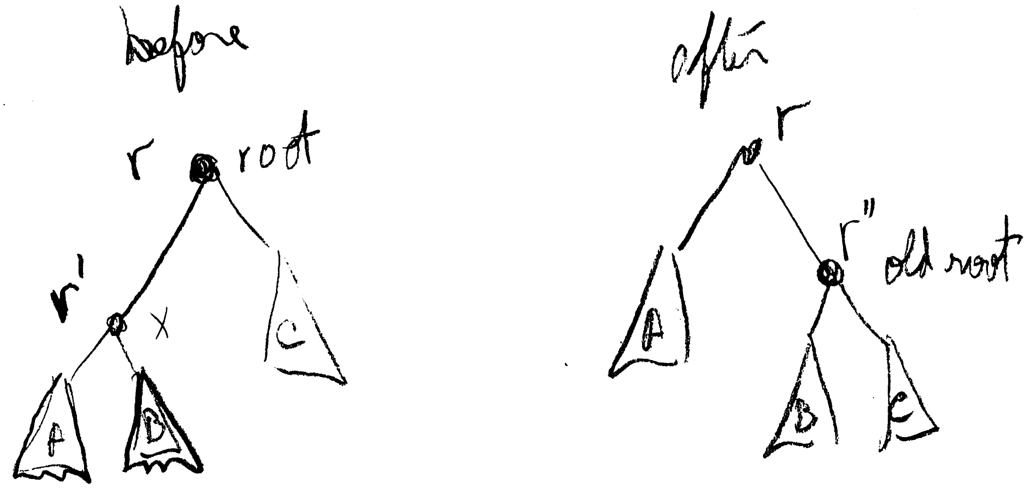
<u>Proof</u> <u>AccessLemma</u> $AC \le 3(r(root) - r(x)) + 1$

$AC(splay) = $ Sum of a collection of $AC(zig)$, $AC(zig\text{-}zag)$, $AC(zig\ zig)$

<u>Idea</u> bded each single step and use them to bd splay.

<u>Case</u> zig step $\boxed{\text{to show}}$ $AC(zig) \le 3(r(root) - r(x)) + 1$

$$\le 3(r - r') + 1$$

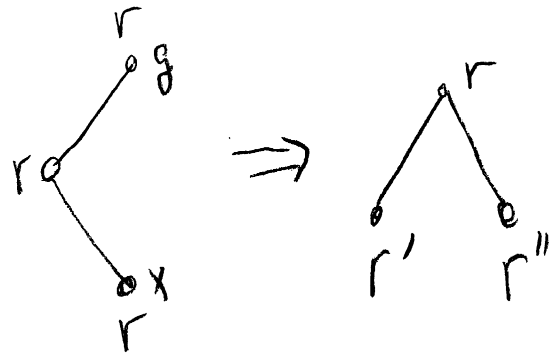


$$\Delta\Phi = (\Phi(A) + \Phi(B) + \Phi(C) + r + r'') - (\Phi(A) + \Phi(B) + \Phi(C) + r + r') = r'' - r' \le r - r'$$

$$AC(zig) = 1 + \Delta\Phi \le 1 + r - r' \le 3(r - r') + 1$$

Case zig-zag to show $AC(\text{zig-zag}) \le 3(r(\text{grandparent}(x)) - r(x))$

Subcase a)

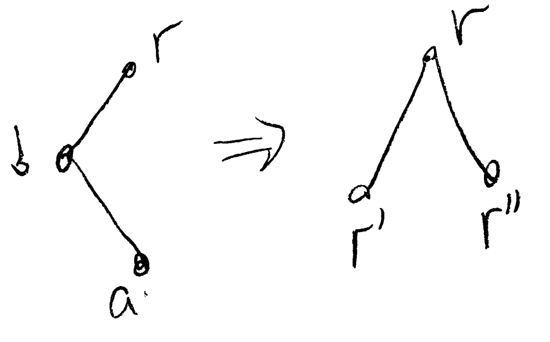


rank rule $r' + r'' < 2r$

$$\Delta\Phi = (r + r' + r'' - 3r) = r' + r'' - 2r < 0$$

$$AC(\text{zig-zag}) = 1 + \Delta\Phi \le 0 \le 3(r(\text{grandparent}(x)) - r(x))$$

Subcase b) $r(\text{grandparent}) > r(x)$



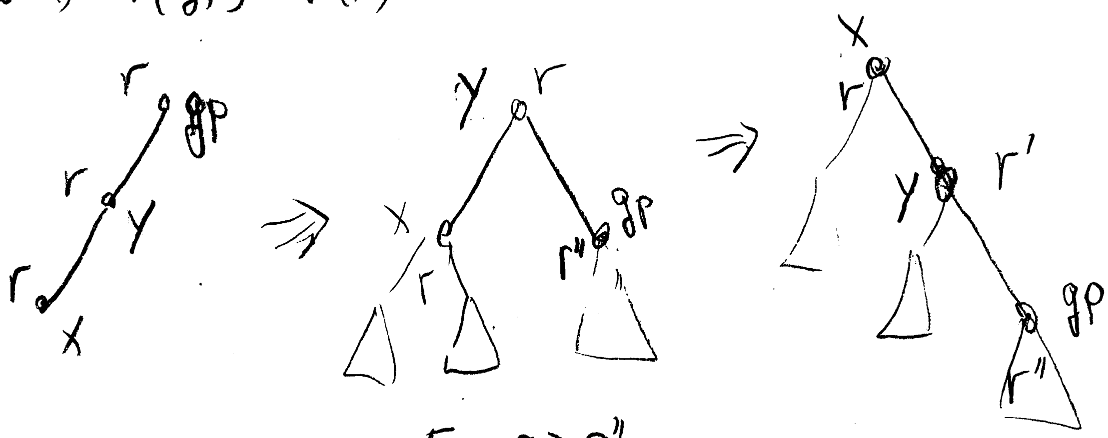
where $r \ge b \ge a$
$r > a$

$$\Delta\Phi = (r + r' + r'' - (r + b + a)) \le$$
$$\le (3r - 1 - r - 2a) = 2r - 2a - 1$$

$$AC \le 1 + 2(r-a) - 1 \le 3(r(\text{grandparent}) - r(x))$$

Case (zig-zig) to show $AC(\text{zig-zig}) \le 3(r(\text{grandparent}) - r(x))$

Subcase a) $r(gp) = r(x)$



note $r > r''$

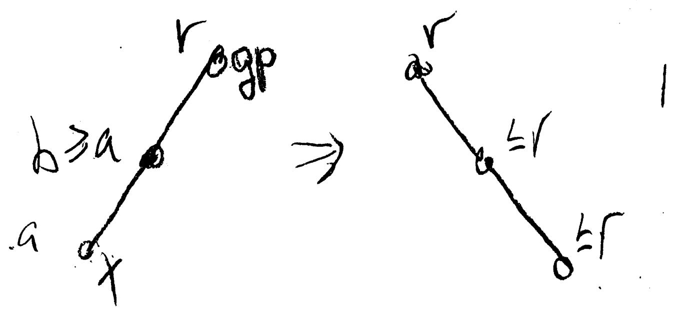
$r \ge r'$

$\Delta\Phi = (r + r' + r'' - 3r)$

$\le r + r + r - 1 - 3r) = -1$

$AC(\text{zig-zig}) \le 1 + \Delta\Phi \le 0 \le 3(r(gp) - r(x))$

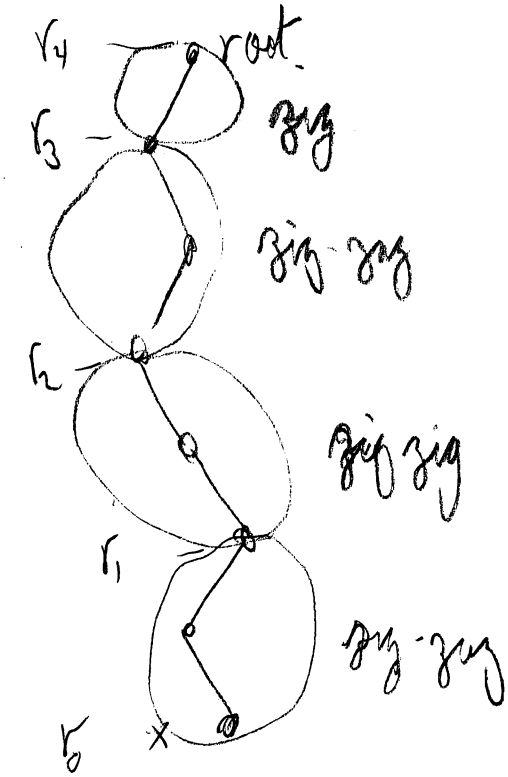
Subcase b) $r(gp) > r(x)$



$\Delta\Phi \le 3r - (r - 2a) = 2(r-a)$

$AC(\text{zig-zig}) = 1 + 2(r-a) \le 3(r-a)$ since $(r-a) \ge 1$

proof by example



$$AC(\text{splay}) = AC(\text{zig-zag}) + AC(\text{zig-zig}) + AC(\text{zag-zig}) + AC(\text{zig})$$

$$\leq 3(r_1 - r_0) + 3(r_2 - r_1) + 3(r_3 - r_2) + 3(r_4 - r_3) + 1$$

$$= 3(r_4 - r_0) + 1$$

<u>Balance Thm</u> m splays on an n-node tree

splay does $O(m \log n + n \log n)$ rotation

<u>pf</u> $AC \leq 3\log n + 1$ $\quad \Phi_{end} \geq 0$ $\quad \Phi_{begin} \leq O(n \log n)$

$$\# rotation \leq O\left(m(3\log n + 1) + \Phi_{begin} - \Phi_{end}\right)$$

$$\leq O(m \log n + n \log n)$$

Static Optimality Thm

Let q_i = # accesses of key i ie $m = \sum q_i$

$q_i > 0$ Then Total cost $O\left(m + \sum q_i \log\left(\frac{m}{q_i}\right)\right)$

Pf

redo access lemma usin weighted nodes

$$S(x) = \sum_{y \in \text{subtree}(x)} w(y)$$

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

Access Lemma Amortized # rotation to splay x

$$3(r(\text{root}) - r(x)) + 1$$

Static Opt Proof $w(x) = q_i/m$

---

eg m accesses from n keys of which we only access k of them

$w(x) = 1/k$ if $x \in K$

$= 0$ ow $x \notin K$