Splay Trees

15-750
1/28/19

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

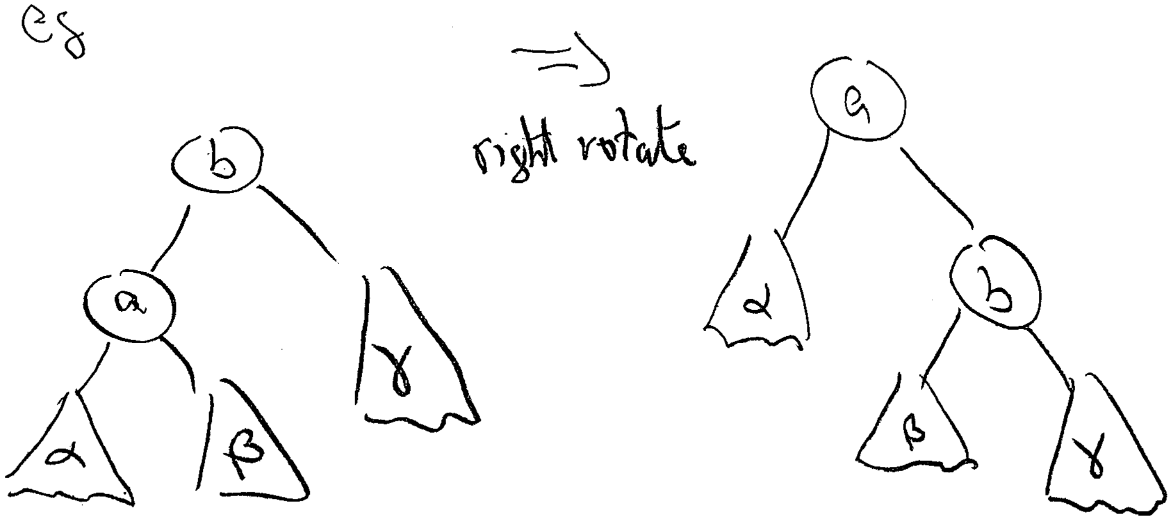
Splay Trees

Two ways to describe them.

1) rotation rules

2) As Treap with rules to change priorities (insertion order)

eg



insertion order

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

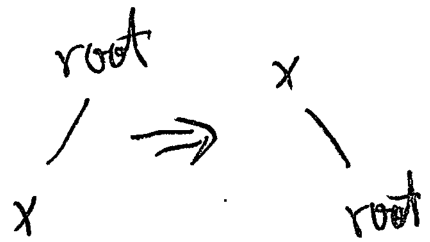
move a in front of b.

3 Rotation Rules for Splay

<u>Goal</u> Def Splay(x) ≡ rotate x to root.

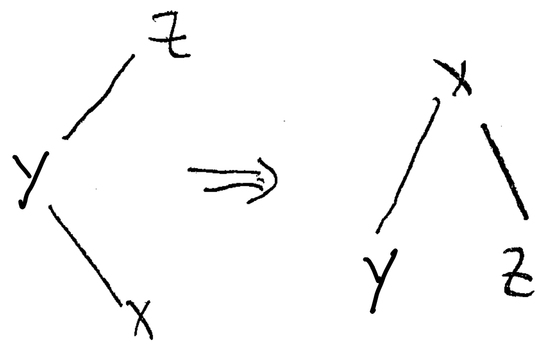
<u>Def</u> Splay(x,T)

1) Zig ≡



(move x to front)

2) Zig-Zag



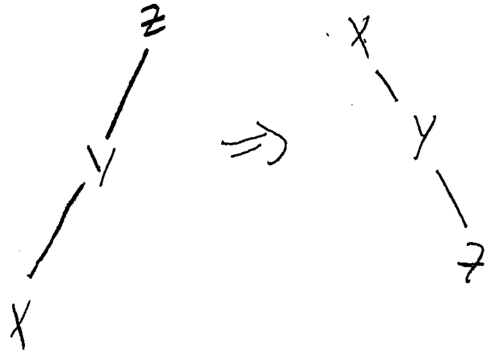
(move x in front of z)

or

1) (move y in front of z)
2) (move x " " y)

The interesting case

3) Zig-Zig



(move y in front of z
move x " " y)

same as 2)

Priority version of splay:

Given x, y, z s.t. $\text{Parent}(x) = y$ $\text{Parent}(y) = z$

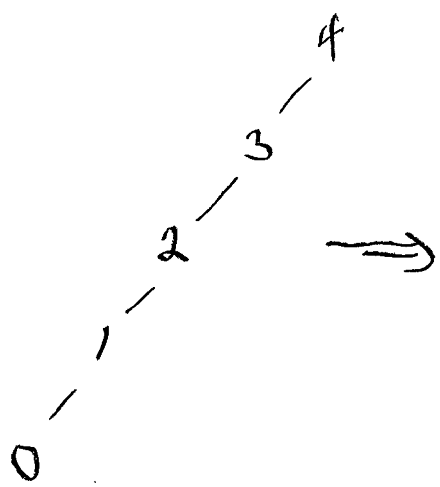
move y in front of z
move x " " y

Why not simple rule: move x in front of y.
i.e. move x to front!
no zig-zig rule!

Keys 0, 1, 2, 3, 4

Priorities: (4, 3, 2, 1, 0)

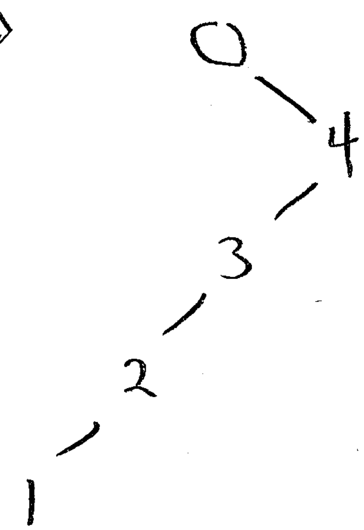
Tree



#rotations = 4

move 0 to front $\Longrightarrow$

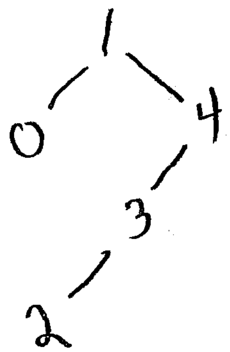
(0, 4, 3, 2, 1)



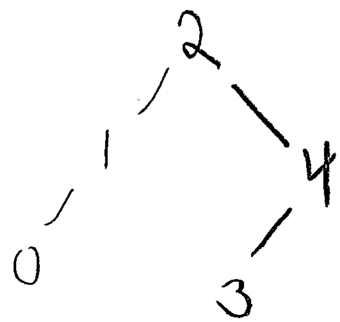
rotations = 4

move 1 to front:

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

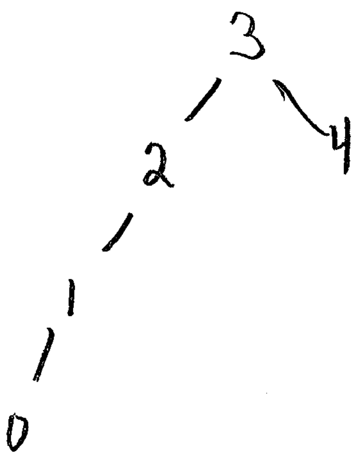


MTF 2

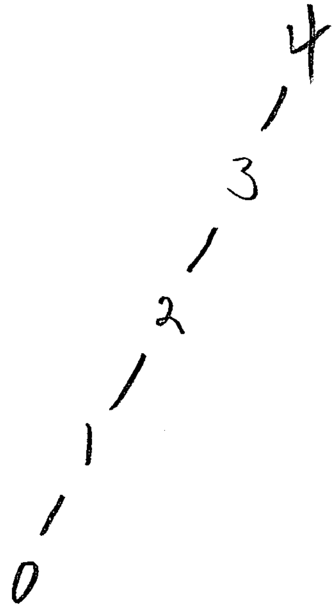


#rotations = 3

#rotations = 2



⇒



Priorities: $(n, n-1, \ldots, 0)$

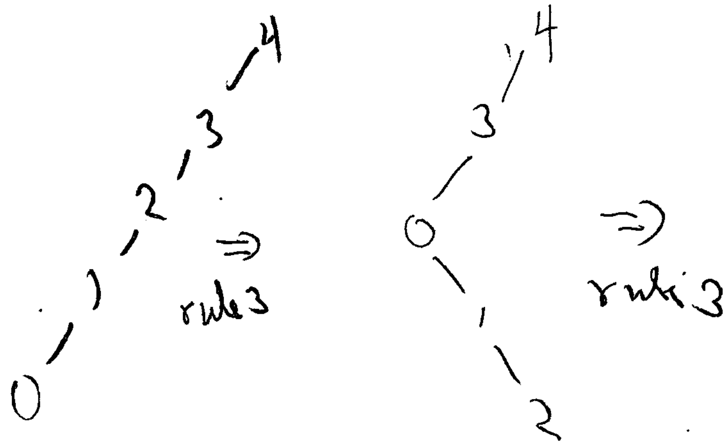
	MTF 0,	...		, MTF n	$n+1$ ops
rotations	n	n	$n-1$ ...	1	$\Omega(n^2)$ rotations

Average # rotations $\Theta(n)$.

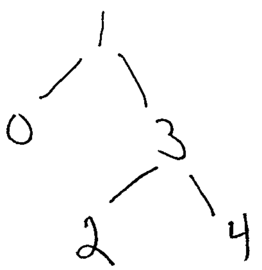
Using all 3 rules

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

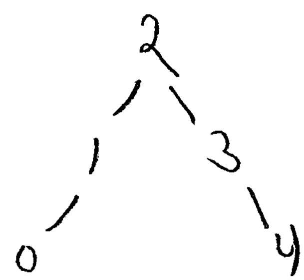
splay(0)



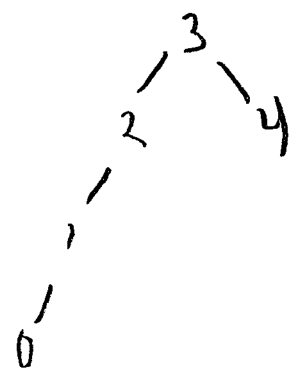
splay(1)



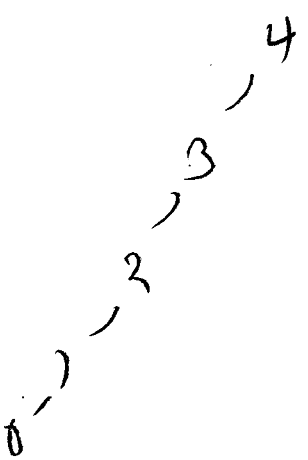
splay(2)



splay(3)



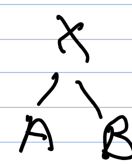
splay(4)

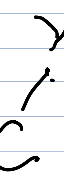
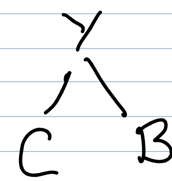


Note: Trees may be very unbalanced!

Splay Dictionary Ops

Insert$(x,T) \equiv$ 1) VanillaInsert(x,T)
2) Splay(x,T)

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

2) If B empty return A.
else Splay(largest(A), A)
giving  return 

Lookup$(x,T) \equiv$
1) Search(x,T)
2) If found then return Splay(x,T)
Else return Splay(parent(x), T)
(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 $\Phi(T) = \sum_{i=1}^{n} \lfloor \log i \rfloor = \Theta(n \log n)$

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

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

Amortized Cost $= $ Unit-Cost $+ \Delta\Phi$

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

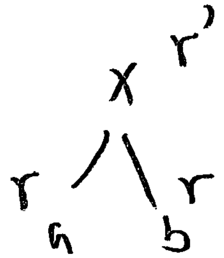
<u>Access Lemma</u> $AC(\text{splay}(x)) \leq 3(r(\text{root}) - r(x)) + 1$

<u>Cor</u> $AC \leq 3 \log n + 1$

Simple facts

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

<u>Rank Rule</u>

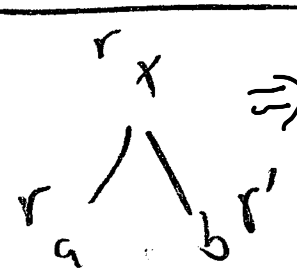


$\Rightarrow r' > r$

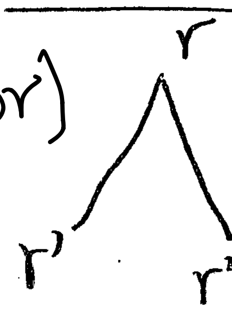
$\Rightarrow s(x) \geq 2^{r+1}$

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

<u>ContraPositive</u>



$\Rightarrow r' < r$ (or)



$2r > r' + r''$

Proof of Access Lemma

Claim: Suffice to prove following two Lemmas:

Lemma 1 $AC(zig) \le 3(r(root) - r(child)) + 1$

Lemma 2 $AC(zig\text{-}zig)$ & $AC(zig\text{-}zag) \le 3(r(GP) - r(c))$

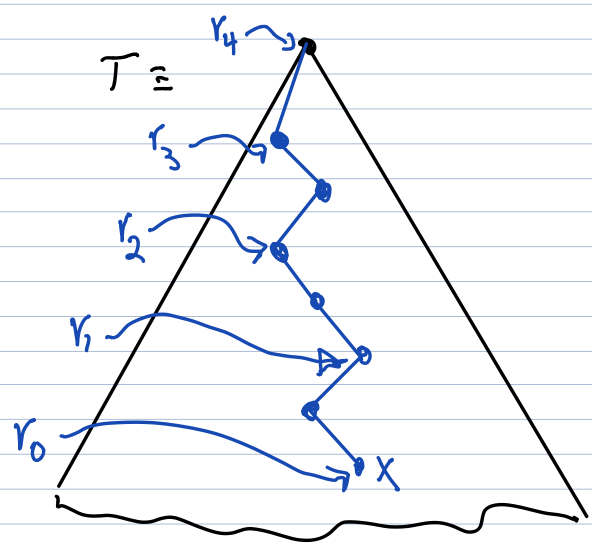
where $GP \equiv GrandParent(c)$

$P \equiv Parent(c)$

$C \equiv child$

Proof of Claim by example: Consider Splay(x)

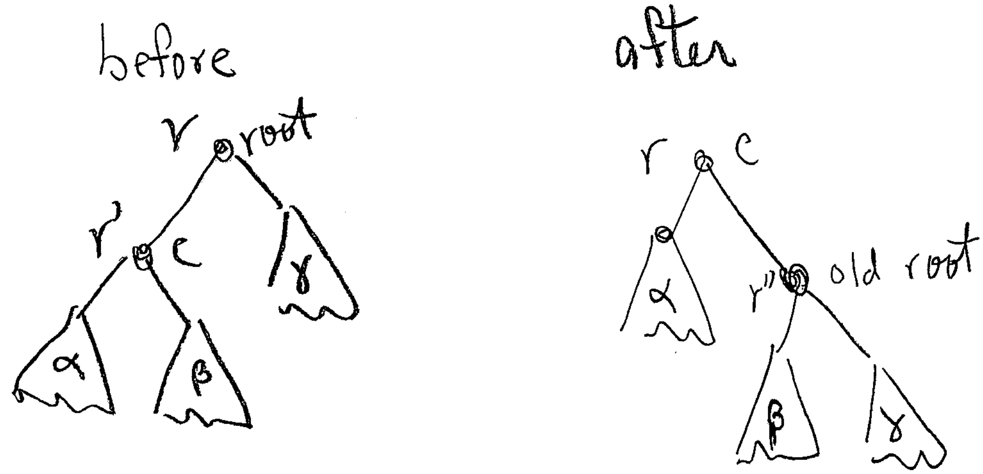
$T =$



$$AC(\mathrm{Splay}(x)) = AC(\text{zig-zag}) + AC(\text{zig-zig}) + AC(\text{zig-zag}) + AC(\text{zig})$$
$$\leq 3(r_1 - r_0) + 3(r_2 - r_1) + 3(r_3 - r_2) + 3(r_4 - r_3) + 1$$
$$= 3(r_4 - r_0) + 1$$

proof of zig-lemma

$$AC(zig) \le 3(r(root) - r(c)) + 1 \quad \text{(to show)}$$
$$= 3(r - r') + 1$$

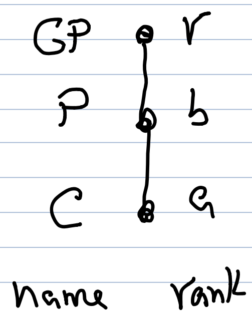


$$\Phi(after) = r + r'' + \Phi(\gamma_1) + \Phi(\beta) + \Phi(\gamma) = r + r'' + \Phi(subtrees)$$

$$\Phi(before) = r + r' + \Phi(subtrees)$$

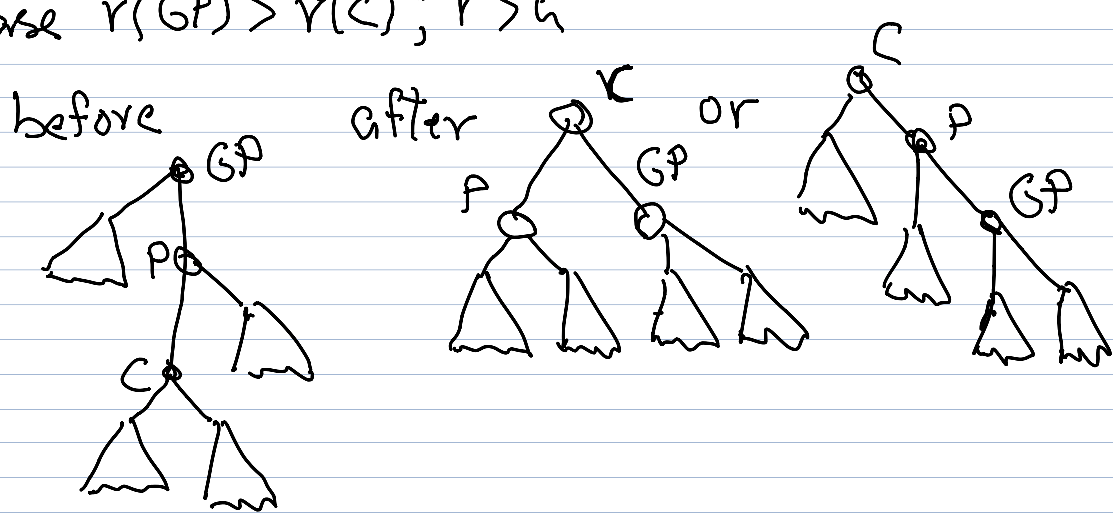
$$AC(zig) = 1 + \Delta\Phi = 1 + r'' - r' \le 1 + (r - r') \le 3(r - r') + 1$$

Table of Potential Changes



Claim	zig-zig or zig-zag
$r > a$	$\Delta\Phi \le 2(r-a)$
$r = a$	$\Delta\Phi \le -1$

Case $r(GP) > r(C)$; $r > a$



$$\Phi(\text{before}) = r + b + a + \Phi(\text{subtrees})$$

$$\geq r + 2a + \Phi(\text{subtrees})$$

$$\Phi(\text{after}) \le 3r + \Phi(\text{subtrees})$$

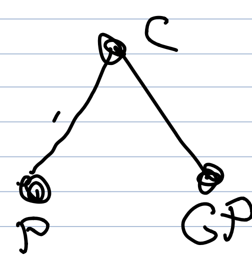
$$\Delta\Phi \le 3r - (r + 2a) = 2(r-a)$$

Case $r=b=a$ $\quad r(GP)=r(P)=r(C)$

$\Phi(\text{before}) = 3r + \Phi(\text{subtrees})$

Claim $\Phi(\text{after}) \le 3r-1+\Phi(\text{subtrees})$

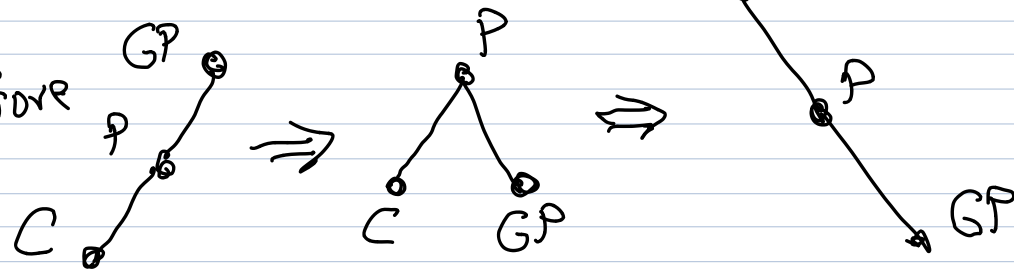
Case zig-zag



$\Rightarrow r(P)$ or $r(GP) < r$

Case zig-zig

before



$r(GP) < r$ $\qquad$ $r(GP)$ unchanged

Thus $\Delta\Phi \le \Phi(\text{after}) - \Phi(\text{before})$

≤ -1

$$AC(\text{case } r(GP) > r(C)) = 1 + \Delta\underline{\Phi} \le 1 + 2(r-a)$$
$$\le 3(r-a)$$

$$AC(\text{case } r(GP) = r(C)) = 1 + \Delta\underline{\Phi} \le 1 - 1 = 0 \le 3(r-r)$$

<u>We Finished Access Lemma!</u>

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

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

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

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

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

Important extensions

Static Optimality Thm

m searches $\wedge$ q_i = # searches of K_i eg $m = \sum q_i$

then total cost $= O\left(m + \sum_{q_i > 0} q_i \log(m/q_i)\right)$

pf

Redo access lemma with weighted nodes

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

Set $\text{rank}(x) = \log(S(x))$

Claim $A((\text{splay}(x)) \leq 3(\text{rank}(\text{root}) - \text{rank}(x)) + 1$

For Opt proof set $w(x) = q_i/m$

Dynamic Optimality Conjecture

(Sleator & Tarjan 1985)

Def: Dynamic BST Algorithm:

1) BST
2) Updated with rotations between searches.
3) May look into future requests to determine rotations.

Conjecture: Splay tree do at most a constaint factor more work than any dynamic BST.