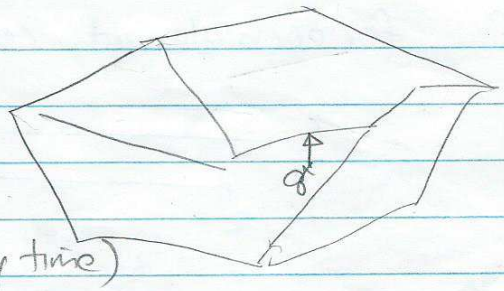
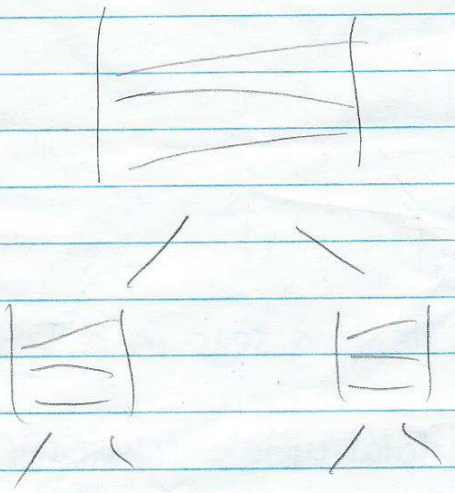
Dynamic 2D Point Location

DS to support query,
insert line segment
delete " "
(assume no intersection at any time)



Suffice to do vertical ray shooting

Method 1 : Dynamic Segment Tree



space $O(n \log n)$

query : given pt q,
$O(\log n)$ nodes → for each node u whose slab contains q,
do binary search for q
$\Rightarrow$ $O(\log^2 n)$

insert: given new seg s,
$O(\log n)$ nodes → for each node u that s belongs to,
do binary search for s
insert s in u's list
$\Rightarrow$ $O(\log^2 n)$

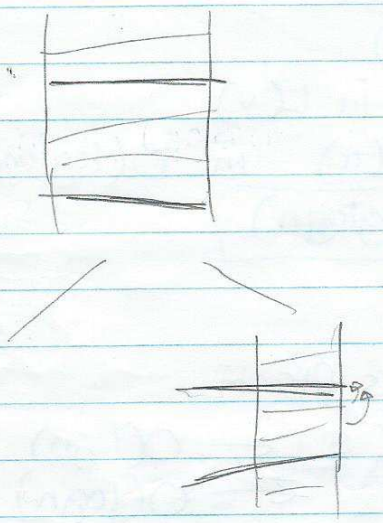
each node's list is itself stored in balanced search tree

delete: similarly $O(\log^2 n)$

issue - how to maintain balance in global tree?
weight balancing...

Method 2 ... ? Dynamic Fractional Cascading

idea - pass fraction $\frac{1}{b}$ of u's list to each child v's list

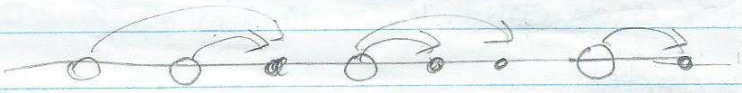


$L(u)$ = original list at u
$L^+(u)$ = "augmented" list

sample$(L^+(u))$ = random sample w. prob $\frac{1}{b}$

$L^+(v) := \overbrace{L(v)}^{\text{red}} \cup \underbrace{\text{sample}(L^+(u))}_{\text{blue}}$

Lemma can maintain list of n red & blue elements s.t. can find the blue succ of each red element in $O(\log\log n)$ time with $O(\log\log n)$ updates ["union-split-find"]



Pf idea: 1. maintain an integer label $\ell(x) \in [U]$ of each element x
s.t. $x < y \Rightarrow \ell(x) < \ell(y)$
(called monotone list labeling)
Known soln has $U = n^{O(1)}$ reducible to $O(1)$ by working w. blocks of $O(\log n)$ size
$O(\log n)$ amort. # label changes
(or $U = O(n)$ → $O(\log^2 n)$ label changes)
by balanced search tree techniques

2. apply vEB trees to the labels
($O(\log\log U) = O(\log\log n)$) □

DIL

Knowing succ of q in $L^+(v)$
$\Rightarrow$ know succ of q in $L(v)$
& in $L^+(u)$ in $O(\log\log n)$ time

$\Rightarrow$ expected query time $\boxed{O(\log n \log\log n)}$

insert s:
for each node u that s fits in,
do binary search for s $\leftarrow O(\log n)$
insert s to $L(u)$ $\leftarrow O(\log n)$
augment(u, s).

augment(u, s):
insert s to $L^+(u)$
w. prob $1/b$, {
insert s to sample$(L^+(u))$
for each child v of u do augment(v, s)
}

let A_i = expected time for augment at level i

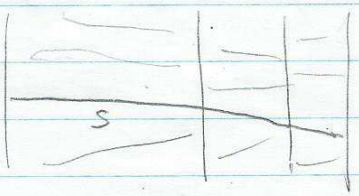
$A_i = O(\log n) + \frac{1}{b} \cdot 2 \cdot A_{i+1}$

$\Rightarrow A_i = O\left(\log n \left(1 + \frac{2}{b} + \left(\frac{2}{b}\right)^2 + \dots\right)\right)$

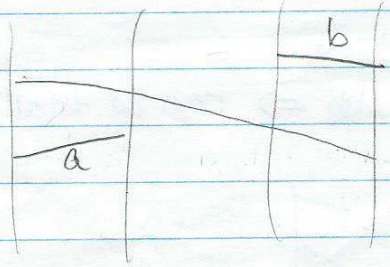
$= O(\log n)$ for $b > 2$.

$\Rightarrow$ expected insert time $\boxed{O(\log^2 n)}$
same for delete

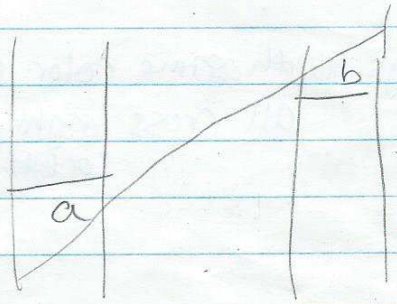
Rmk - fractional cascading! can't speed up insert



can't pass segs from one slab to diff slab since we can't totally order segs



$a < b$



$a > b$

New Method (C.-Nekrich '15)

$O(\log n (\log\log n)^3)$ query

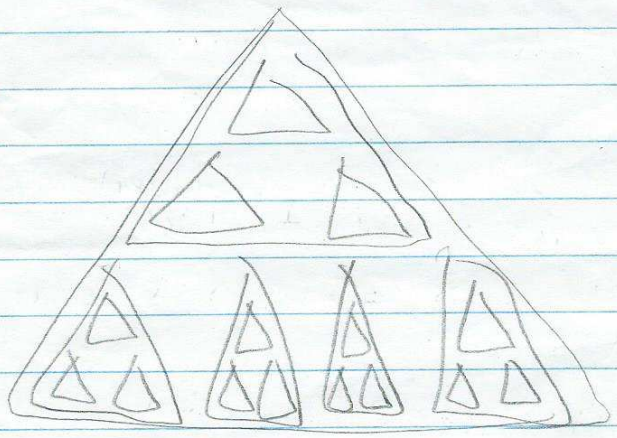
$O(\log n \log\log n)$ update

[$O(n \log n)$ space]

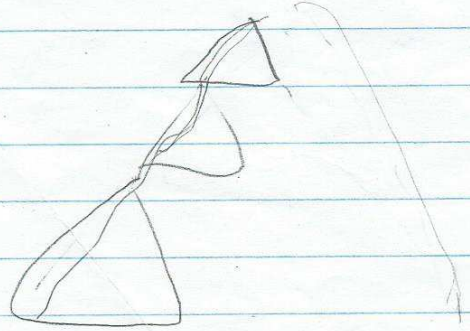
rough idea - assign colors to segs

- only totally order segs w. same color

colors = $O(\log\log n)$



hierarchy of subtrees with $O(\log\log n)$ layers

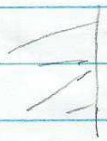


to insert, decompose path into $O(\log\log n)$ pieces that fit in $O(\log\log n)$ subtrees, of each given diff. color

D13

segs with same color in same subtree

all cross right side of root slab (or left) $\Rightarrow$ can be totally ordered



History	Query	Update	
Bentley '77	$\log^2 n$	$\log^2 n$	(seg tree)
Chiang-Tamassia '91	$\log n$	$\log^2 n$	(trapezoid tree) for connected monotone
Cheng-Janardan '90	$\log^2 n$	$\log n$	(interval tree + priority search tree)
Goodrich-Tamassia '91	$\log^2 n$	$\log n$	(mnial/dual spanning tree) (Planar graph) for connected
Baumgarten-Jung-Rehlhorn '92	$\log \log n \log n$	$\log^2 n$	
Arge-Brodal-Georgiadis '06	$\log n$	$\frac{\log^2 n}{\log \log n}$	