		update time	extreme-pt query
History	C'99	$O(\log^{1+\varepsilon} n)$	$O(\log n)$
	Brodal-Jacob'00	$O(\log n \log\log\log n)$	$O(\log n)$
	Brodal-Jacob'02	$O(\log n)$	$O(\log n)$

(CH area $O(\sqrt{n})$ still best)

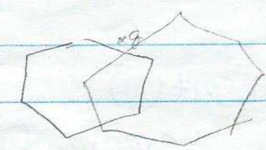
100 pages!

General Dynamization Techniques

Def A query problem is decomposable if answer for input set $S_1 \cup S_2$ can be computed from answer for S_1 & answer for S_2 in $O(1)$ time

Ex
- range counting ✓
- range emptiness ✓
- nearest neighbor search ✓
- extreme pt in CH ✓

- membership in CH ✗
- intersect CH with line ✗



Technique 1: The "Logarithmic Method" (Bentley-Saxe '80)

If there is a static DS $\mathcal{S}$ with $P(n)$ preproc time, $S(n)$ space, $Q(n)$ query time & problem is decomposable,

then there is a semidynamic DS $\mathcal{S}'$ with $O(S(n))$ space, $O\left(\frac{P(n)}{n} \log n\right)$ insert time amortized

i.e. total time for n inserts is $O(P(n)\log n)$

$O(Q(n) \cdot \log n)$ query time

Ex triangle range counting in 2D

$P(n) = O(n \log n)$, $S(n) = O(n)$, $Q(n) = O(n^{1/2+\varepsilon})$

$\Rightarrow$ $O(n)$ space, $O(\log^2 n)$ insert (amort), ($D(n) = O(\log n)$ by

$O(n^{1/2+\varepsilon} \log n)$ query time

($O(\log n)$ delete)

CH extreme pt query in 2D $\quad$ ($D(n)=O(\log n)$ by Chazelle)

$P(n) = O(n \log n)$, $S(n) = O(n)$, $Q(n) = O(\log n)$

$\Rightarrow$ $O(n)$ space, insert $O(\cancel{\log^2 n})$ $O(\log n)$, query $O(\log^2 n)$ (delete $O(\log n)$)

worse than Preparata '80!

nearest neighbor search in 2D $\quad$ ($M(n)=O(n)$ by Kirkpatrick)

$P(n) = O(n \log n)$, $S(n) = O(n)$, $Q(n) = O(\log n)$

Voronoi diagram

$\Rightarrow$ $O(n)$ space, insert $O(\cancel{\log^2 n})$ $O(\log n)$, query $O(\log^2 n)$

Proof: divide input set S into $O(\log n)$ subsets

invariant: for each $i = 0, \ldots, \log n$,

at most 1 subset at <u>level</u> i of size 2^i

insert(p):

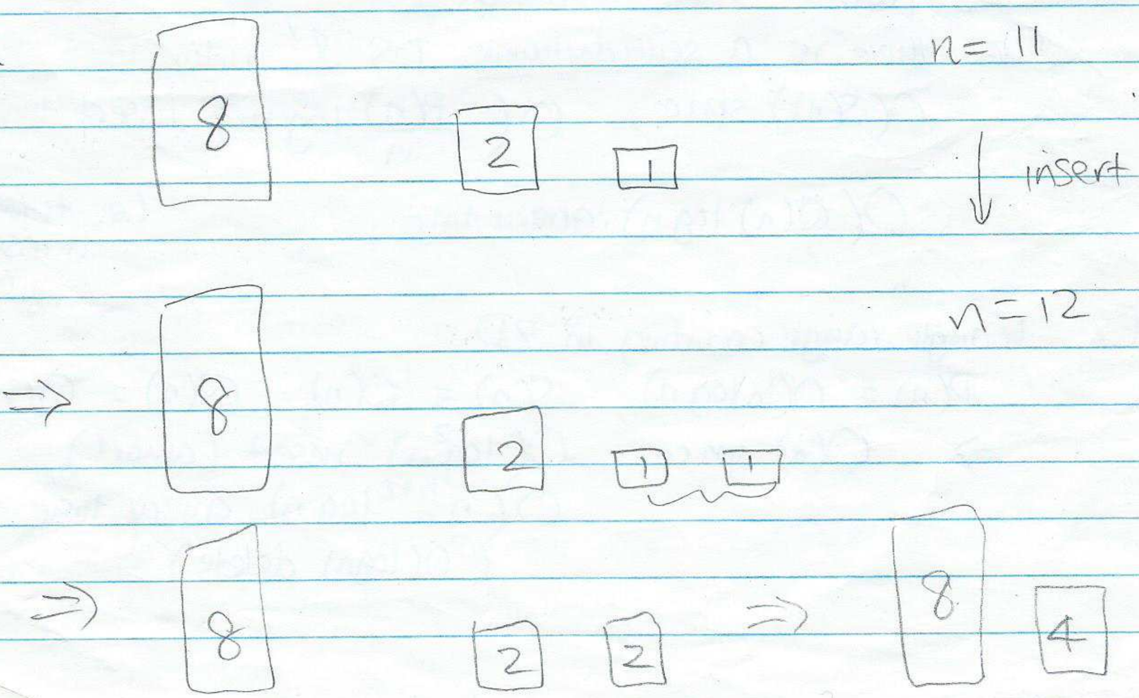
create $\{p\}$ at level 0

whenever $\exists$ 2 subsets S_1, S_2 at same level i

destroy S_1, S_2

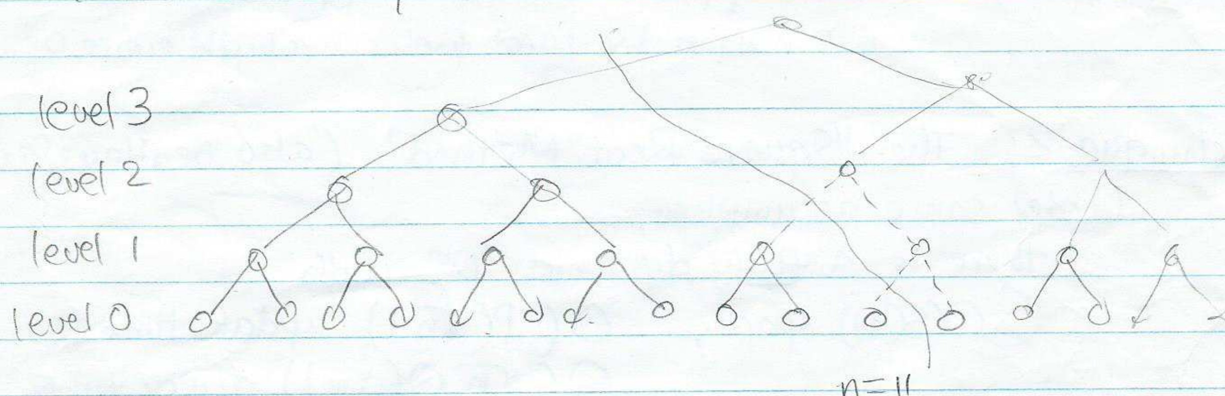
build $S_1 \cup S_2$ at level $i+1$

e.g.



(similar to binary counter) $(1011 \to 1100)$

Viewed as a tree, over time:



(similar to mergesort)

Space $\sum_{i=0}^{\log n} S(2^i) = O(S(n))$ [assume $\frac{S(n)}{n}$ is nondecreas.]

query $\leq \sum_{i=0}^{\log n} Q(2^i)$ by decomposability

$\leq O(Q(n) \log n)$

total time for n inserts

$\leq \sum_{i=0}^{\log n} P(2^i) \frac{n}{2^i}$ (# builds at level i is $\frac{n}{2^i}$)

$\leq \sum_{i=0}^{\log n} P(n) \frac{n}{n}$ [assume $\frac{P(n)}{n}$ is nondecreas.]

$= O(P(n) \log n)$ $\square$

Rmks: (i) amortized $\to$ worst case
(idea - spread rebuild over multiple updates
messy!)

(ii) insert time improves to $O(\frac{M(n)}{n} \log n)$
where $M(n)$ = merging time

D7

(iii) tradeoffs: base-b version ...
(iv) if $\mathcal{S}$ supports delete in $D(n)$ time,
$\mathcal{S}'$ also supports delete in $O(D(n))$ time
(if n drops by const factor, rebuild entire DS...)

Technique 2: The "Square Root Method" (also Bentley-Saxe)
Under same assumption,
there is a fully dynamic DS with
$O(S(n))$ space, $O(P(\sqrt{n}))$ update time,
$O(\sqrt{n}\, Q(\sqrt{n}))$ query time.

Proof: divide S into $\sqrt{n}$ subsets of size $\sqrt{n}$

insert(p):
let S_i = smallest subset
destroy S_i
build $S_i \cup \{p\}$

delete(p):
let S_i = subset containing p
destroy S_i
build $S_i - \{p\}$ $\square$

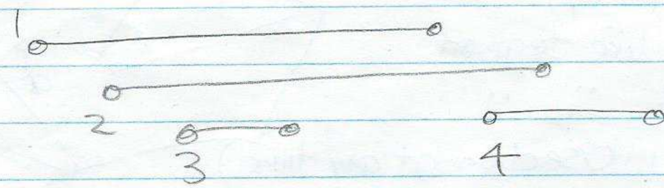
Ex nearest neighbor in 2D
update $O(\sqrt{n} \log n)$
query $O(\sqrt{n} \log n)$.

[C'19: $O(\log^4 n)$ update
$O(\log^2 n)$ query]
via the logarithmic method
+ shallow cuttings

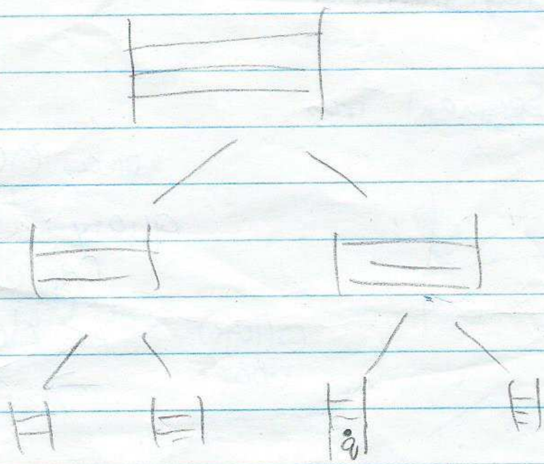
Technique 3:
Under same assumption,
there is an offline dynamic DS $\mathcal{S}'$ with
entire update sequence is given in advance

$O(S(n) \log n)$ space, $O(\frac{P(n)}{n} \log n)$ amort. update time,
$O(Q(n) \log n)$ query time.

Proof: for each element, consider its lifespan / time interval



build segment tree for these n intervals!

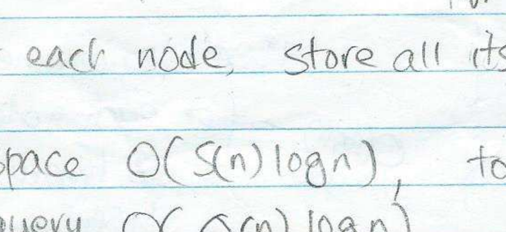


for each node, store all its long segs in a DS $\mathcal{S}$

Space $O(S(n) \log n)$, total update time $O(P(n) \log n)$,
query $O(Q(n) \log n)$ $\square$

Rmks - generalizes to <u>semi-online</u>
(insert is online; when element is inserted, told its delete time)

- don't need decomposability
if $\mathcal{S}$ supports insert in <u>worst-case</u> $I(n)$ time:
then $\mathcal{S}'$ has $O(I(n) \log n)$ amort. update time
$O(Q(n))$ query time



/09