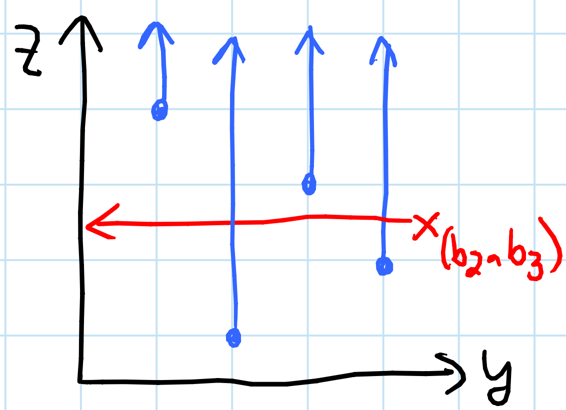
6.851	Lecture 4	Feb. 28, 2012

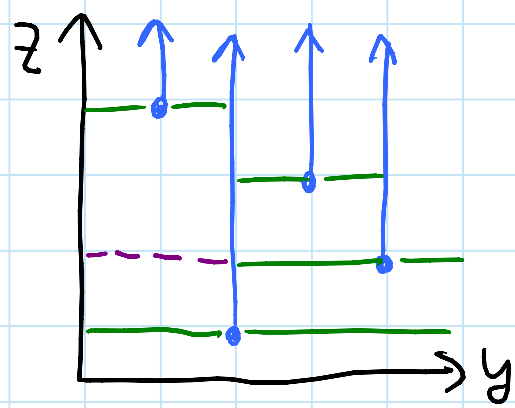
TODAY: Geometry II (of 2)
- application of fractional cascading
- kinetic data structures

$O(\lg n)$ 3D orthogonal range searching: (static)

[Chazelle & Guibas — Alg. 1986]

① $(-\infty, b_2] \times (-\infty, b_3)$: search for b_3 in z list $+ O(k)$
- equivalent to stabbing vertical rays (from points) with horizontal ray (from (b_2, b_3))





- draw horizontal segments through points
- subdivide faces to have bounded degree by extending some horizontal segments
  - like fractional cascading: insert $\leq 1/2$ into left neighbor, recurse; ditto right

$\Rightarrow O(n)$ space [Chazelle — SICOMP 1986]
- query searches for b_3 among left rays then walks right k steps in $O(k)$ (each crossed ray = 1 point in output)

② $[a_1, b_1] \times (-\infty, b_2) \times (-\infty, b_3)$: $O(\lg n \cdot \text{search} + k)$

- range tree on x
- each node stores ① on points in subtree

$\Rightarrow$ query reduces to $O(\lg n)$ ① queries

③ $[a_1, b_1] \times [a_2, b_2] \times (-\infty, b_3)$: $O(\lg n \cdot \text{search} + k)$

- "range tree" on y
- node v stores key = max(left(v)) (as before)
  & ② on points in right(v)
  & y-inverted ②′ on points in left(v)
  ↳ query $[a_1, b_1] \times (a_2, \infty) \times (-\infty, b_3)$
- query: walk down tree
  - if key $< a_2 < b_2$: walk right
  - if key $> b_2 > a_2$: walk left
  - if $a_2 \le$ key $\le b_2$: stop
    - query ② for $[a_1, b_1] \times (-\infty, b_2) \times (-\infty, b_3)$
    - query ②′ for $[a_1, b_1] \times (a_2, \infty) \times (-\infty, b_3)$

$\Rightarrow O(\lg n) + O(1)$ ② queries

④ $[a_1, b_1] \times [a_2, b_2] \times [a_3, b_3]$: $O(\lg n \cdot \text{search} + k)$

- same as ③ but on z & recursing with ③
  instead of y instead of ②
- naively $O(\lg^2 n + k)$
- fractional cascading $\Rightarrow O(\lg n + k)$
  - bounded degree 5: parent, children, 2 aux.
- space: $O(n \lg^3 n)$ ($\lg$ per ②, ③, ④)

Kinetic data structures: moving data

- think: tracking physical objects (phones, cars, ...)

[Basch, Guibas, Hershberger – J.Alg. 1999]

Data: value/coordinate = (known) function of time (instead of a single number)

- e.g. affine $\underbrace{a}_{\text{initial}} + \underbrace{b}_{\text{velocity}} t$
- bounded-degree algebraic $a + bt + ct^2 + \cdots$
- pseudo-algebraic: any certificate of interest flips true/false $O(1)$ times

Operations:

- modify(x, $f(t)$): replace x's function
- idea: motion estimation accurate "for a while"
- advance(t): go forward in virtual time
- other updates/queries usually about present (virtual) time

Approach:

- store data structure accurate now
- augment with certificates: conditions under which DS is accurate, which are true now
- compute failure time for each certificate
- store them in a priority queue
- as certs. invalidate, fix DS & replace certs.

Kinetic predecessor:

- want pred./succ. search in present in $O(\lg n)$
- let's try a BST
- certificates: $\{x_i \le x_{i+1}\}$
  where $x_1, x_2, \dots, x_n$ is an in-order traversal
- $\text{failure}_i = \inf\{t \ge \text{now} \mid x_i(t) \ge x_{i+1}(t)\}$
  (next time certificate i will fail)
- advance(t):
  - while $t \ge$ Q.min:
    - now = Q.min
    - event(Q.delete-min)
  - now = t
- event $(x_i \le x_{i+1})$: (in fact, $x_i = x_{i+1}$ now)
  - swap x_i & x_{i+1} in BST
  - add certificate $x_i' \le x_{i+1}'$
  - replace certificate $x_{i-1} \le x_i$ with $x_{i-1} \le x_i'$
    & certificate $x_{i+1} \le x_{i+2}$ with $x_{i+1}' \le x_{i+2}$
  - update failure times in priority queue

Metrics: $\boxed{\text{above:}}$

1. <u>responsive</u>: when certificate expires (event), can fix DS quickly $\boxed{O(\lg n)}$
2. <u>local</u>: no object participates in many certs. $\Rightarrow$ modify is fast $\boxed{O(1)}$
3. <u>compact</u>: # certs. is small $\Rightarrow$ low space $\boxed{O(n)}$
4. <u>efficient</u>:

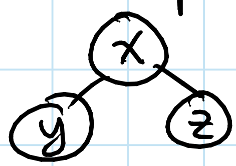
$$\frac{\text{worst-case \# DS events}}{\text{worst-case \# ``necessary changes''}} \text{ is small} \quad \boxed{O(1)}$$

Efficiency: (the vaguest part of kinetic DSs)

- if we need to "know" sorted order "at all times", need to update for each order change & that's what we do
- if we need to support fast pred./succ. "at all times", need to "approximately know" sorted order (?)
- usually study worst-case behavior for affine/pseudo-alg. data with no updates
- here: $\Theta(n^2)$
  - Ω: 
  - O: each pair passes $\leq$ once for affine — $O(1)$ for pseudo-alg.

Kinetic heap: [de Fonseca & de Figueiredo — IPL 2003]

- want find-min (& delete-min) in $O(\lg n)$
- could use kinetic predecessor ~ can do better
- store a min-heap
- certificates:

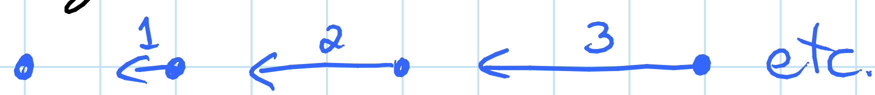


$x \le y$
$x \le z$

- event$(x \le y)$:
  - swap x & y in tree
  - update adjacent certificates

① responsive: $O(\lg n)$ (priority queue)
② local: $O(1)$
③ compact: $O(n)$
④ efficient: $O(\lg n)$

- $\Theta(n)$ changes to min in worst case
  - Ω:



  - O: once min changes $x \to y$, x cannot be min again

- claim $O(n \lg n)$ events in DS for affine motion

- OPEN: (pseudo-)algebraic motions?
- OPEN: faster advance because don't need to query interim times?

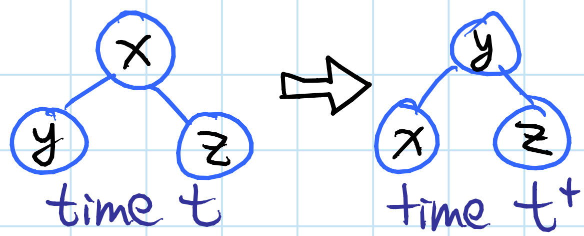
Proof: (ASSUMING AFFINE MOTION)

- $\Phi(t) = \#$ events in future $> t$

$$= \sum_x \underbrace{\left(\begin{array}{l}\text{\# descendants of } x \text{ @ time } t \text{ that} \\ \text{will overtake } x \text{ in future} > t\end{array}\right)}_{\Phi(t,x)}$$

- $$\Phi(t,x) = \sum_{\substack{y \text{ child} \\ \text{of } x}} \underbrace{\left(\begin{array}{l}\text{\# descendants of } y \text{ @ time } t \\ \text{that will overtake } x \text{ in} > t\end{array}\right)}_{\Phi(t,x,y)}$$

- consider event at time t:



  - $\Phi(t^+,v) = \Phi(t,v) \quad \forall v \neq x,y$

    (v gains/loses no descendants & isn't overtaken)

  - $\Phi(t^+,x) = \underbrace{\Phi(t,x,y)}_{\text{remaining descends.}} - \underbrace{1}_{y}$

  - $\Phi(t^+,y) = \Phi(t,y) + \Phi(t,y,z)$ (not x)

    $\leq \Phi(t,y) + \Phi(t,x,z)$ (overtake $y \Rightarrow$ overtake x)*

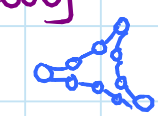
    $= \Phi(t,y) + \Phi(t,x) - \Phi(t,x,y)$

  $\Rightarrow \Phi(t^+) \leq \Phi(t) - 1$

- $\Phi(0) \leq \sum_x \underbrace{\#\text{descendants of } x}_{O(\lg n)}$

  $= O(n \lg n)$ □

Kinetic survey: [Guibas – DS Handbook 2005]

- 2D convex hull [Basch, Guibas, Hershberger 1999]
  - also diameter, width, min. area/perim. rectangle
  - efficiency = $O(n^{2+\varepsilon}) / \Omega(n^2)$
  - OPEN: 3D?
- $(1+\varepsilon)$-approximate diameter, smallest disk/rectangle in $(1/\varepsilon)^{O(1)}$ events [Agarwal & Har-Peled – SODA 2001]
- smallest enclosing disk: [Demaine, Eisenstat, Guibas, Schulz – FWCG 2010] efficiency $O(n^{3+\varepsilon}) / \Omega(n^2)$
- Delaunay triangulation [Albers, Guibas, Mitchell, Roos – IJCGA 1998]
  - $O(1)$ efficiency
  - OPEN: how many changes? $\tilde{O}(n^3)$ & $\Omega(n^2)$
    - $O(n^{2+\varepsilon})$ if unit speed [Rubin – FOCS 2013]
- any triangulation:
  - $\Omega(n^2)$ changes even with Steiner points [Agarwal, Basch, de Berg, Guibas, Hershberger – SoCG 1999]
  - $O(n^{2+1/3})$ events [Agarwal, Basch, Guibas, Hershberger, Zhang – WAFR 2000]
  - OPEN: $O(n^2)$?
  - $O(n^2)$ events for pseudotriangulations 
- collision detection [Kirkpatrick, Snoeyink, Speckmann 2000] [Agarwal, Basch, Guibas, Hershberger, Zhang 2000] [Guibas, Xie, Zhang 2001] ← 3D
- MST
  - $O(m^2)$ easy; OPEN: $o(m^2)$? ↗ sorted order of edge weights
  - $O(n^{2-1/6})$ for H-minor-free graphs (e.g. planar) [Agarwal, Eppstein, Guibas, Henzinger – FOCS 1998]