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

\[ O(\log n) \text{ 3D orthogonal range searching: } (\text{static}) \]

[Chazelle & Guibas - Alg. 1986]

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

\[\begin{array}{c}
\uparrow \\
x_{(b_2, b_3)}
\end{array}\]

- draw horizontal segments through points
- subdivide faces to have bounded degree by extending some horizontal segments
  - like fractional cascading: insert \(\leq \frac{1}{2}\)
    into left neighbor, recurse: ditto right

\[ O(n) \text{ space } [\text{Chazelle - SiCOMP 1986}] \]
- query searches for \(b_3\) among left rays
  then walks right \(k\) steps in \(O(k)\)
  (each crossed ray = 1 point in output)
\[ (a_1, b_1) \times (-\infty, b_2) \times (-\infty, b_3) : O(\lg n \cdot \text{search} + k) \]
- range tree on \( x \)
- each node stores \( \ref{query1} \) on points in subtree
  \( \Rightarrow \) query reduces to \( O(\lg n) \) \( \ref{query1} \) queries

\[ (a_1, b_1) \times [a_2, b_2] \times (-\infty, b_3) : O(\lg n \cdot \text{search} + k) \]
- "range tree" on \( y \)
- node \( v \) stores \( \text{key} = \max(\text{left}(v)) \) (as before)
  & \( \ref{query2} \) on points in \( \text{right}(v) \)
  & \( y \)-inverted \( \ref{query2} \) on points in \( \text{left}(v) \)

  \( \Rightarrow \) query \( (a_1, b_1) \times (a_2, \infty) \times (-\infty, b_3) \)

- query: walk down tree
  - if \( \text{key} < a_2 < b_2 \): walk right
  - if \( \text{key} > b_2 > a_2 \): walk left
  - if \( a_2 \leq \text{key} \leq b_2 \): stop
    - query \( \ref{query2} \) for \( (a_1, b_1) \times (-\infty, b_2) \times (-\infty, b_3) \)
    - query \( \ref{query2} \) for \( (a_1, b_1) \times (a_2, \infty) \times (-\infty, b_3) \)

  \( \Rightarrow O(\lg n) + O(1) \) \( \ref{query2} \) queries

\[ (a_1, b_1) \times [a_2, b_2] \times [a_3, b_3] : O(\lg n \cdot \text{search} + k) \]
- same as \( \ref{case3} \) but on \( z \) & recursing with \( \ref{case3} \)

  - 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 \( \ref{query2}, \ref{case2}, \ref{case3} \))
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 $a + bt$
  initial velocity
- bounded-degree algebraic $a + bt + ct^2 + ...$
- 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 $O(\log n)$
- let's try a BST
- certificates: $\exists_1 x_i \leq x_{i+1}$
  where $x_1, x_2, \ldots, x_n$ is an in-order traversal
- failure$_i = \inf \{ t \geq \text{now} \mid x_i(t) \geq x_{i+1}(t) \}$
  (next time certificate $i$ will fail)
- advance($t$):
  - while $t \geq Q.$min:
    - now = $Q.$min
    - event($Q.$delete-min)
  - now = $t$
- event ($x_i \leq x_{i+1}$): (in fact, $x_i = x_{i+1}$ now)
  - swap $x_i$ & $x_{i+1}$ in BST
  - add certificate $x'_i \leq x'_{i+1}$
  - replace certificate $x_{i-1} \leq x_i$ with $x_{i-1} \leq x'_i$
    & certificate $x_{i+1} \leq x_{i+2}$ with $x_{i+1} \leq x_{i+2}$
  - update failure times in priority queue
Metrics:

1. **Responsive**: when certificate expires (event), can fix DS quickly \( O(lg \, n) \)
2. **Local**: no object participates in many certs => modify is fast \( O(1) \)
3. **Compact**: # certs is small \( O(n) \)
4. **Efficient**: worst-case # DS events is small \( O(1) \)

Separately:

- 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: \( O(n^2) \)
  - \( \Omega: \ldots \ldots \)
  - \( O: \) each pair passes \( \leq \) once for affine \( O(1) \) for pseudo-alg.
Kinetic heap: [de Fonseca & de Figueiredo-ILP 2003]
- want find-min (& delete-min) in $O(lg n)$
- could use kinetic predecessor ~ can do better
- store a min-heap
- certificates:

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

- responsive: $O(lg n)$ (priority queue)
- local: $O(1)$
- compact: $O(n)$
- efficient: $O(lg n)$
- $\Theta(n)$ changes to min in worst case
- $\Omega$: $\bullet \leftrightarrow 1 \leftrightarrow 2 \leftrightarrow 3 \bullet$ etc.
- $O$: once min changes $x \Rightarrow y$, $x$ 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)
- $\Phi(t) = \# \text{ events in future } \geq t$
  = $\sum_x (\# \text{ descendants of } x \text{ at time } t \text{ that will overtake } x \text{ in future } \geq t)$

- $\Phi(t,x) = \sum_{y \text{ of } x}(\# \text{ descendants of } y \text{ at time } t \text{ that will overtake } x \text{ in } \geq t)$

- consider event at time $t$:

- $\Phi(t^+, y) = \Phi(t, y)$ \quad $\forall y \neq x, y$
  (y gains/loses no descendants & isn't overtaken)

- $\Phi(t^+, x) = \Phi(t, x, y) - 1$
  (remaining descendants of y)

- $\Phi(t^+, y) = \Phi(t, y) + \Phi(t, y, z)$ \leq $\Phi(t, y) + \Phi(t, x, z)$
  (overtake y => overtake x)∗

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

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

- $= O(n \lg n)$
Kinetic survey:

- **2D convex hull** [Basch, Guibas, Hershberger 1999]
  - also diameter, width, min. area/perim. rectangle
  - efficiency $= \mathcal{O}(n^{2+\varepsilon})/\Omega(n^2)$
  - **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 $\mathcal{O}(n^{3+\varepsilon})/\Omega(n^2)$

- 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 pseudo triangulations

- collision detection [Kirkpatrick, Snoeyink, Speckmann 2000]
  [Agarwal, Basch, Guibas, Hershberger, Zhang 2000]
  [Guibas, Xie, Zhang 2001] $\rightarrow$ 3D

- MST
  - sorted order of edge weights
  - $O(m^2)$ easy; **OPEN**: $o(m^2)$?
  - $O(n^{2-1/6})$ for $H$-minor-free graphs (e.g. planar) [Agarwal, Eppstein, Guibas, Henzinger - FOCS 1998]