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

\(O(\log n)\) 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(1/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
  \(\Rightarrow 0(n)\) space [Chazelle - SICOMP 1986]
- query searches for \(b_3\) among left rays
  then walks right \(k\) steps in \(O(1/k)\)
  (each crossed ray = 1 point in output)
2. \([a_1, b_1] \times (-\infty, b_2) \times (-\infty, b_3): \Omega(lg n \cdot search + k)\)
   - range tree on \(x\)
   - each node stores \(1\) on points in subtree
   \(\Rightarrow\) query reduces to \(Ω(lg n) \cdot 1\) queries

3. \([a_1, b_1] \times [a_2, b_2] \times (-\infty, b_3): \Omega(lg n \cdot search + k)\)
   - "range tree" on \(y\)
   - node \(v\) stores key = \(\max(\text{left}(v))\) (as before)
     \& 2 \(\) on points in \(\text{right}(v)\)
     \& \(y\)-inverted 2 \(\) 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 key < \(a_2 < b_2\): walk right
   - if key > \(b_2 > a_2\): walk left
   - if \(a_2 \leq \text{key} \leq b_2\): stop
     - query 2 \(\) for \([a_1, b_1] \times (-\infty, b_2) \times (-\infty, b_3)\)
     - query 2 \(\)' for \([a_1, b_1] \times (a_2, \infty) \times (-\infty, b_3)\)
   \(\Rightarrow\) \(Ω(lg n) + O(1) \cdot 2\) queries

4. \([a_1, b_1] \times [a_2, b_2] \times [a_3, b_3]: \Omega(lg n \cdot search + k)\)
   - same as 3 \(\) but on \(z\) \& recursing with 3
     instead of \(y\) \(\uparrow\) 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: \( \text{value/coordinate} = (\text{known}) \text{ function of time} \) (instead of a single number)
- e.g. affine \( a + b t \)
  \( \text{initial velocity} \)
- bounded-degree algebraic \( a + b t + c t^2 + \cdots \)
- pseudo-algebraic: any certificate of interest flips true/false \( O(1) \) times

Operations:
- \text{modify}(x, f(t)): replace x's function
- idea: motion estimation accurate "for a while"
- \text{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: $\exists x_i \leq x_{i+1}$
  where $x_1 \times x_2 \times \cdots \times x_n$ is an in-order traversal
- $\text{failure}_i = \inf \{ t \geq \text{now} \mid x_i(t) \geq x_{i+1}(t) \}$
  (next time certificate $i$ will fail)
- $\text{advance}(t)$:
  - while $t > Q$.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 \( O(\log n) \)
2. **Local**: no object participates in many certs. \( \implies \) modify is fast \( O(1) \)
3. **Compact**: # certs. is small \( \implies \) low space \( O(n) \)
4. **Efficient**: worst-case # DS events / worst-case # “necessary changes” is small \( O(1) \)

**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: \( O(n^2) \)
  - \( \Omega \): \( \cdots \)
  - \( \Theta \): each pair passes \( \leq \) once for affine \( \Theta(1) \) 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:
  
  ![Diagram](attachment://heap_diagram.png)
  
  $x \leq y$, $x \leq z$
- event ($x \leq y$):
  - 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
- $\Omega$: 1 2 3 etc.
- $\Theta$: 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)
- $\Phi(t) = \# \text{ events in future } > t$
  $\quad = \sum_x \left( \# \text{ descendants of } x \text{ at time } t \text{ that will overtake } x \text{ in future } > t \right)$

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

- Consider event at time $t$:
  - $\Phi(t^+, v) = \Phi(t, v)$ for $v \neq x, y$
    ($v$ gains/loses no descendants & isn't overtaken)
  - $\Phi(t^+, x) = \Phi(t, x, y) - 1$
    (remaining descendants, $y$)
  - $\Phi(t^+, y) = \Phi(t, y) + \Phi(t, y, z) \leq \Phi(t, y) + \Phi(t, x, z)$
    (overtake $y$ ⇒ overtake $x$)*
    $= \Phi(t, y) + \Phi(t, x) - \Phi(t, x, y)$
  $\Rightarrow \Phi(t^+) \leq \Phi(t) - 1$

- $\Phi(0) \leq \sum_x \# \text{ descendants of } x$
  $\quad \leq O(n \log n)$
  $\quad = O(n \log n)$
  $\square$
Kinetic survey:
- 2D convex hull
  - also diameter, width, min. area/perim. rectangle
  - efficiency = \( O(n^{2+\epsilon})/\Omega(n^2) \)
  - OPEN: 3D?

- \((1+\epsilon)\) approximate diameter, smallest disk/rectangle
  in \((1/\epsilon)O(1)\) events
  [Agarwal & Har-El, SODA 2001]

- smallest enclosing disk:
  efficiency \( O(n^{3+\epsilon})/\Omega(n^2) \)
  [Demaine, Eisenstat, Guibas, Schulz - FCRC 2010]

- Delaunay triangulation
  - \(O(1)\) efficiency
  - OPEN: how many changes? \( O(n^{3+\epsilon}) \& \Omega(n^2) \)
  [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: \( O(n^3) \) ?
  - \( O(n^3) \) events for pseudo triangulations

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

- MST
  - \( O(n^3) \) easy; OPEN: \( o(n^3) \)?
  - \( O(n^{2-1/6}) \) for H-minor-free graphs (e.g., planar)
    [Agarwal, Eppstein, Guibas, Henzinger - FOCS 1998]