Voronoi Diagrams
in $n \cdot 2^{O(\sqrt{\lg \lg n})}$ Time

Timothy M. Chan   Mihai Pătraşcu

STOC’07
Things I hope you heard about:

2D

- Voronoi diagrams
  - 3-d convex hull
  - Delaunay triangulation
  - Euclidean MST
  - largest empty circle
  - offline nearest neighbor

- segment intersection
  - trapezoidal decomposition
  - triangulating polygons with holes

$\Theta(n \lg n)$
But just in case you didn’t…

Everything reducible to one problem
   NB: reductions not obvious, but known

“Sorting in 2 Dimensions”
Given a vertical slab with:
• \( m \) segments cutting across
• \( n \) points in slab
Output: points sorted w.r.t. segments

\[ O(n \log m) \]
   * compare all points to middle segment
   * recurse (up, down)
Things I hope you heard about:

1D

- numbers inside computers are not real
  i.e. numbers have bounded precision

Theory for something that works:
“Sorting is not $\Theta(n \log n)$”
- $O(n \cdot \sqrt{\log n})$ randomized
- $O(n \cdot \log \log n)$ deterministic
Why consider 2D?

Easy answer: [Willard, SODA’92], others

- it is not $\Theta(n\lg n)$ on my computer (or yours)
- practice uses finite precision (k-d trees, gridding etc)
  - can we have theoretical guarantees?
  - can we improve practice based on theoretical ideas?
- mathematics:
  - information, communication, algorithms – about geometry!
  - differences from 1D fascinating
(Our) Previous Work

- considered online searching (point location)
  => first sublogarithmic result
- for algorithms, it gave: \( O(n \cdot \min\{\lg n / \lg \lg n, \sqrt{w / \lg w}\}) \)
  \( w = \) precision, in bits

Here, consider offline problem directly ("2D sorting")

\[
\Rightarrow n \cdot 2^{O(\sqrt{\lg \lg n})} \ll n \lg n
\]

e.g. \(< n \lg^\varepsilon n \)

significant improvement independent of precision

significant improvement for small precision
Review of Previous Technique

- pick \( B \) segments and sketch them
- recurse \( \Rightarrow O(\log_B n) \) cost per point

Throw away (bad) segments, sketch the rest.
\( \Rightarrow \) segments closer than \( 1/2^{w/B} \) on left or right

Only far segments \( \Rightarrow \) precision \( w/B \) enough

- Can sketch \( B \) segments

- \( B \)-ary search fails when universe is reduced
  - reduction by \( 2^{w/B} \) either left or right
  - precision \( w \) \( \Rightarrow \) at most \( 2B \) universe reductions

Optimize \( O(\log_B n + B) \) \( \Rightarrow O(\log n / \log \log n) \)
Idea 1: Pack Points

“With precision \( w/B \), can pack \( B \) segments per word”

Reimplement old \( O(n\lg m) \) algorithm:
“compare all points to middle segment; recurse”

- compare \( B \) packed points to a segment in \( O(1) \) time \( \Rightarrow \) cost
  ~~ each point compared to
- segments too close:
  \( \Rightarrow \) recomputed sketches, cost
  ~~ each point resketched \( B \) times
  \[ \frac{1}{B} \times O(lg m) \]

Optimize \( \Rightarrow O(n\sqrt{\lg m}) \)

\[ \frac{1}{B} \times O(lg m) + O(1) \times \frac{B + (lg m)/B}{n} \times n \]
Idea 2: Repack Points

Before:
- reduce precision by $\sqrt{\log m}$
- solve reduced problem directly
- recurse between close segments

New idea:
- reduce precision by 10
- recurse in reduced problem
- recurse between close segments

Recursive calls work with original (unpacked) points
Recursive calls work with packed points
Repack more tightly

Information manipulation bottleneck (also in 1D):
- working with packed points \(\sim\) external memory
- external memory permutation is \(\omega(n/B)\)
- “having things in the right order” is not free

Tools from integer sorting + Careful balancing
\[ n \cdot 2^{O(\sqrt{\log \log n})} \]
Idea 3: Different Packing

Before:
“universe reduction” = either on left or right side

Low-precision grid looks like: projective map

⊄ not word parallelizable with standard operations

Now:
“universe reduction” = both on left and right side

So “universe” looks like: affine map

⊆ word parallelizable, standard operations

Minor trouble:
rounding to low-precision no longer works
Can be fixed with some geometry…
Open Problems

all reductions to 2D sorting randomized
  => still no $o(n \lg n)$ deterministic

• better bounds? can randomization help?

• red-blue intersection counting
  -- only one with no $o(n \lg n)$