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

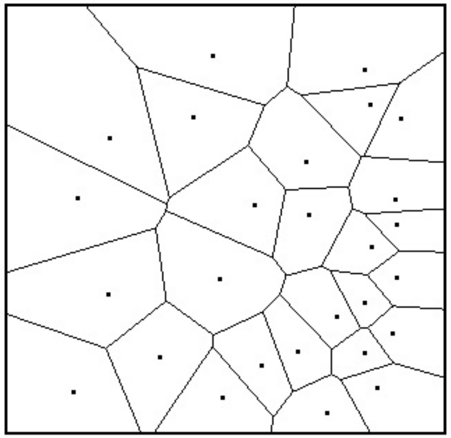
Timothy M. Chan
University of Waterloo

Mihai Pătraşcu
MIT

STOC’07

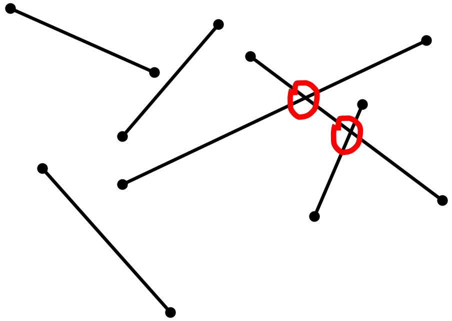
Things I hope you heard about:

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

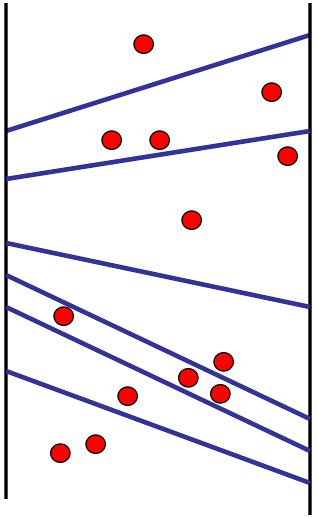
NB: reductions not obvious, but known

"Sorting in 2 Dimensions"

Given a vertical slab with:

- m segments cutting across
- n points in slab

Output: points sorted w.r.t. segments



$O(n \lg m)$: * 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

- hashing
- tries
- radix so

fference

Theory for something that works:

“Sorting is not $\Theta(n \lg n)$”

- $O(n \cdot \sqrt{\lg \lg n})$ randomized
- $O(n \cdot \lg \lg n)$ deterministic

Why consider 2D?

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

Theory Matters®

- 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

[Chan FOCS'0

significant improvement
for small precision

beat $O(n \lg n)$
for any precision

- considered online ... t location)

  => first sublog...

- 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")

=> $n \cdot 2^{O(\sqrt{\lg\lg n})} \ll n \lg n$

e.g. $< n \lg^{\varepsilon} n$

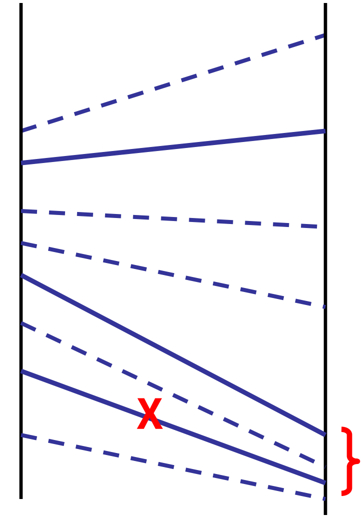
significant improvement
independent of precision

Review of Previous Technique

- pick B segments and sketch them
- recurse => $O(\lg_B n)$ cost per point

Throw away bad segments, sketch the rest.

⇨ segments closer than $1/2^{w/B}$ on left *or* right



Only far segments => 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 => at most $2B$ universe reductions

Optimize $O(\log_B n + B)$ => $O(\lg n/\lg\lg n)$

☺ ☹ 😐

Idea 1: Pack Points

"With precision w/B, can pack B ~~segments~~ points 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 => cost $1/B$ $*$ per point

  ~~ each point compared to $O(\lg m)$ segments

  $+$

- segments too close:

  => recon… cost $O(1)$ $*$ per point

  ~~ ea… B times

$B + (\lg m)/B$ $*n$

Optimize

=> $O(n\sqrt{\lg m})$

Idea 2: Repack Points

Repack more tightly

Before:

- reduce precision
- solve reduce
- recurse betwe… …egments

Recursive calls wo… original (unpacked) …ints

…calls work with packed points

Tools from integer sorting
+ Careful balancing

$n \cdot 2^{O(\sqrt{\lg \lg n})}$

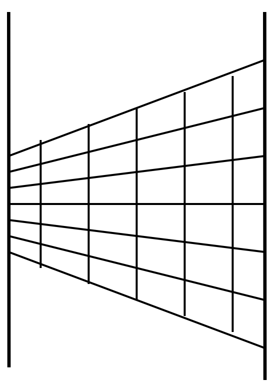
Information manipulation bottleneck (also in 1D):

- working with packed points ~~ external memory
- external memory permutation is $\omega(n/B)$
- "having things in the right order" is not free

Idea 3: Different Packing

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

Low-precision grid looks like:



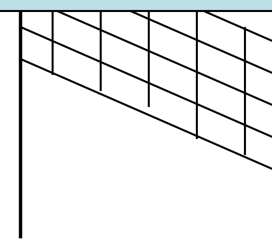
projective map

☹ not word parallelizable with standard operations

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

So “uni e” looks like:

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



affine map

☺ word parallelizable, standard operations

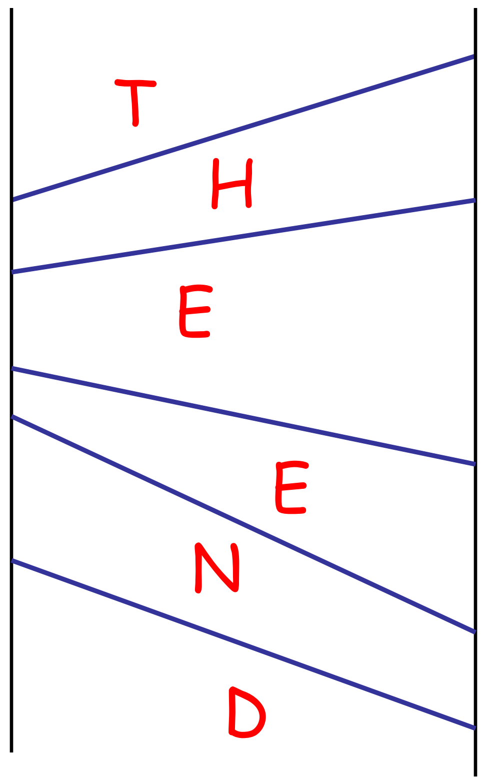
Open Problems

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 $o(n \lg n)$


T
H
E
E
N
D