# Cajun SODA

Benoît Hudson

January 21, 2004

# Going Retro, 1984-style

Demaine, Iacono, Langerman: *Retroactive Data Structures*.

- Data structures let you query and update at the current time.
- What if you want to rewrite history?
- **Definition 1.** Partially Retroactive*: query at the current time, update now or past.*
- **Definition 2.** Fully Retroactive*: query any time, update any time.*
- Automating the process: rollback & redo. Slow. Databases do this.
- Lower bound: $\exists$ problems where you can't do better.
  Cute proof based on **Motzkin's theorem**: evaluating a polynomial of degree $n$ takes $\Omega(n)$ time even with infinite preprocessing.

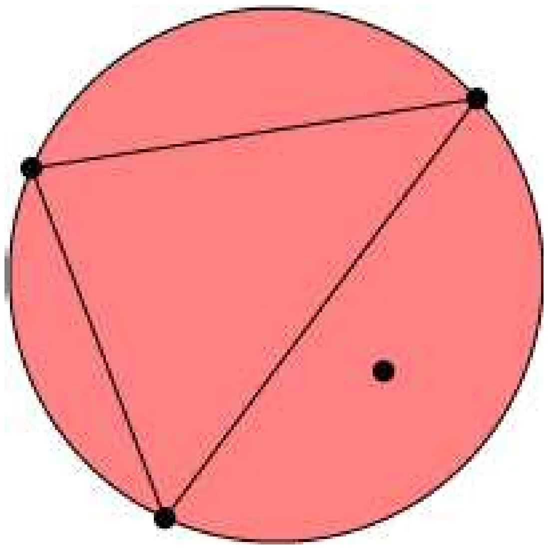
# More retroactivity

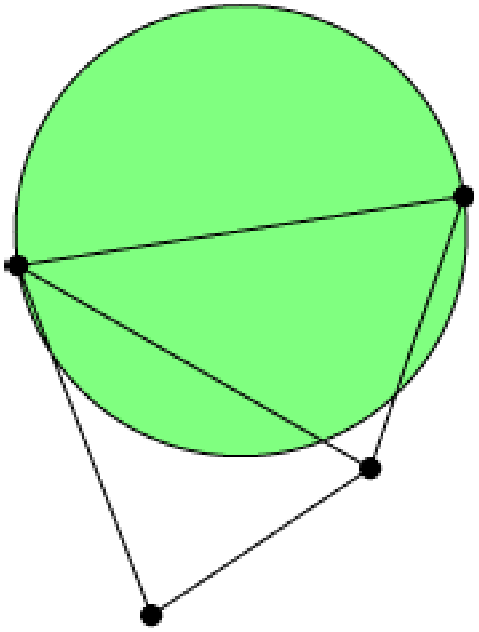
- Demaine et al: lots of upper bounds. Log-time on a bunch of problems.
- Umut et al: more detail on an automatic approach, and nice results on a particular problem.

# Empty spheres abound!

**Lots** of talks about Delaunay triangulations and nearest-neighbour.

**Definition 3.** *A simplex (i.e. segment, triangle, tetrahedron) is* Delaunay *if there is a sphere going through the vertices of the simplex that has no vertices in its interior.*





# Why do I care?

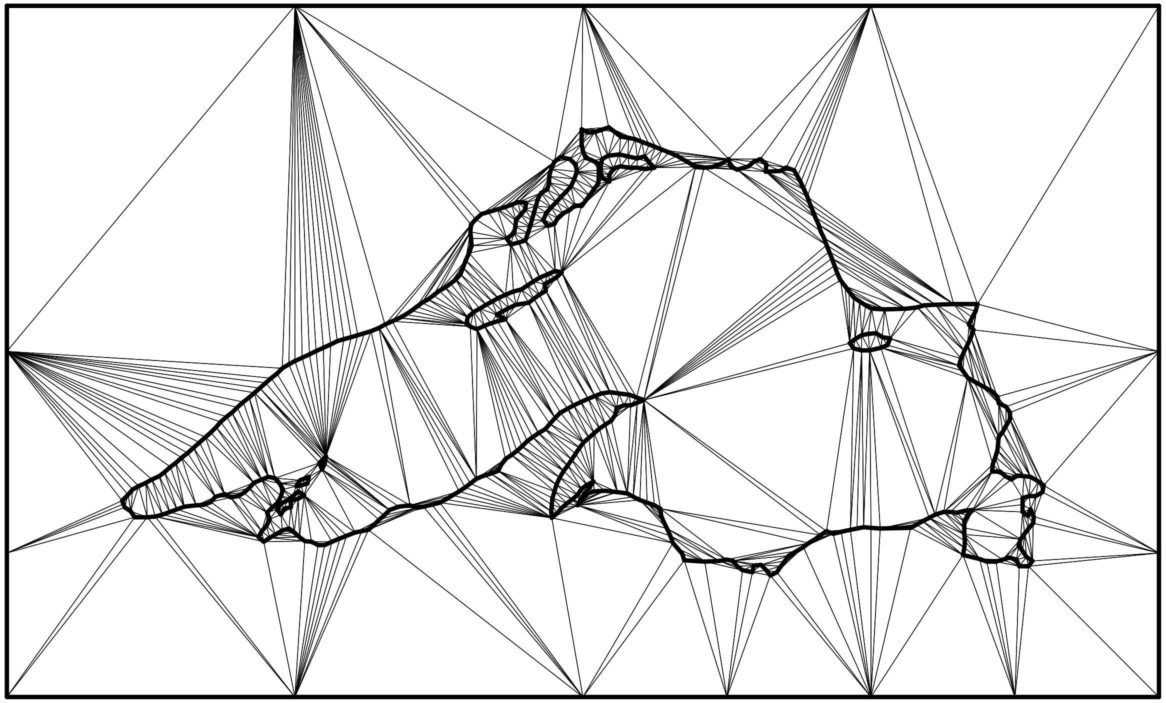
**Good properties:**

- Smallest angle maximized over all triangulations. Good for numerical issues.
- Adjacent vertices are nearest neighbours. Good for protein folding.

**Bad properties:**

- Smallest angle can still be tiny.
- Fragile under deformation: similar point sets can have dissimilar triangulations.

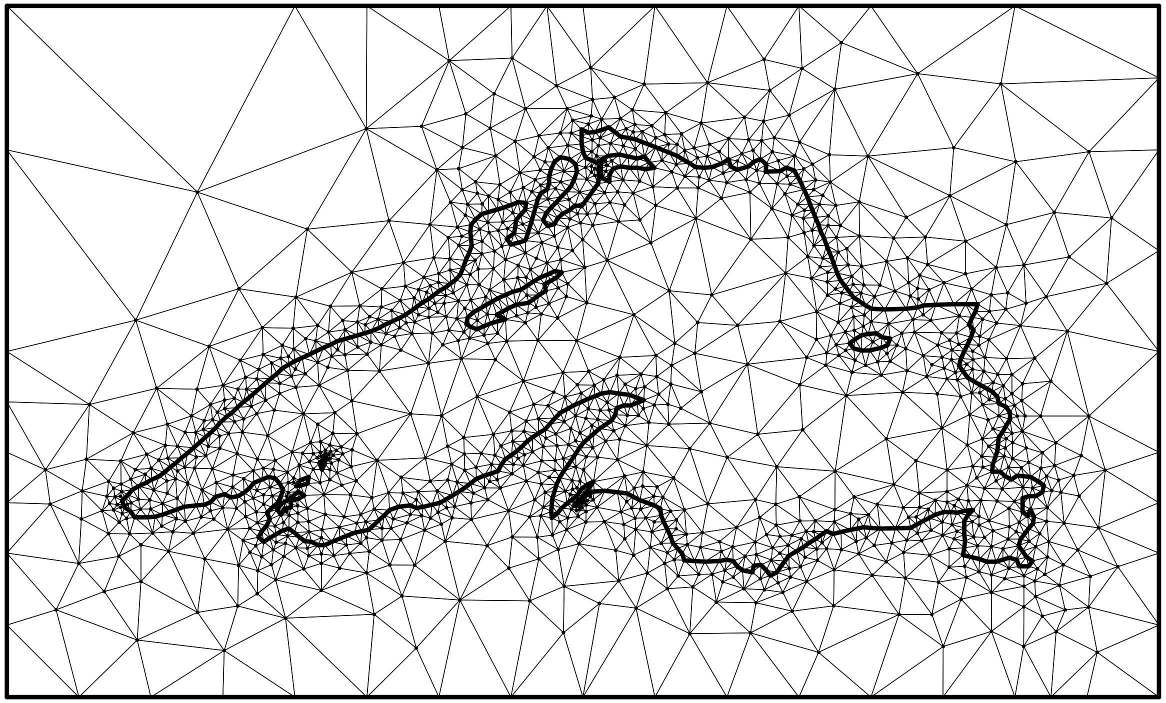
# Meshing



This mesh is Delaunay but has small angles.

# Meshing

Ruppert's algorithm adds points to make the smallest angle larger.



Problem: we don't know what order to add points in. Wrong order $\Rightarrow$ $\Omega(n^2)$ time.

# Meshing

Gary Miller: A time efficient Delaunay refinement algorithm.

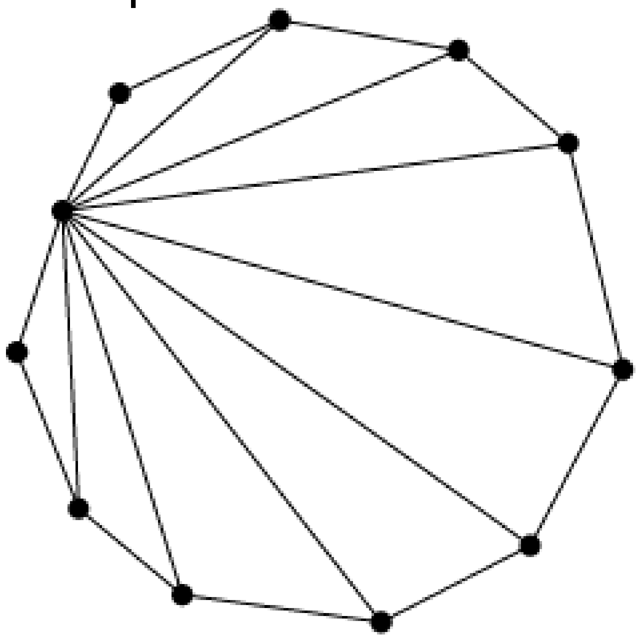
Proves $O((n \log \Gamma(G) + m) \log m)$.

- $n$ : number of input points
- $m$ : number of output points
- $\Gamma(G)$ : something related to ratio of largest feature to smallest feature.

If $\Gamma(G)$ is $\mathrm{poly}(n)$ then the bound is $O((n \log n + m) \log m)$.

# Nearest Neighbours

Bandyopadhyay & Snoeyink. Almost-Delaunay Simplices: Nearest Neighbour Relations for Imprecise Points.



The Delaunay triangulation is very sensitive to slight perturbations in the points.

Authors' motivation: what if we only know position within $\epsilon$?

# Nearest Neighbours

**Definition 4.** Almost-Delaunay*: a simplex is almost-Delaunay if it would be Delaunay if we perturb the points.*

**Definition 5.** Probably-Delaunay*: Assume point positions are gaussian clouds. What is the probability the simplex is Delaunay?*

Almost-Delaunay takes $O(n^5 \log n)$ to compute in 3d. Hacks and assumptions reduce it to $O(n^2 \log n)$ for protein data.

# Nearest Neighbours and Kazaa

Hildrum et al: A Note on the Nearest Neighbour in Growth-Restricted Metrics.

Say I want to find the nearest host that has my favourite song, in $O(\log n)$ time.

We can organize the network hierarchically:

- Pick a random sample of the network for each level $i$.
- Each node at level $i$ is connected to the nearest node at level $i - 1$.

If the number of neighbours within distance $r$ grows linearly with $r$, then we can search in time $O(\log n)$ w.h.p.

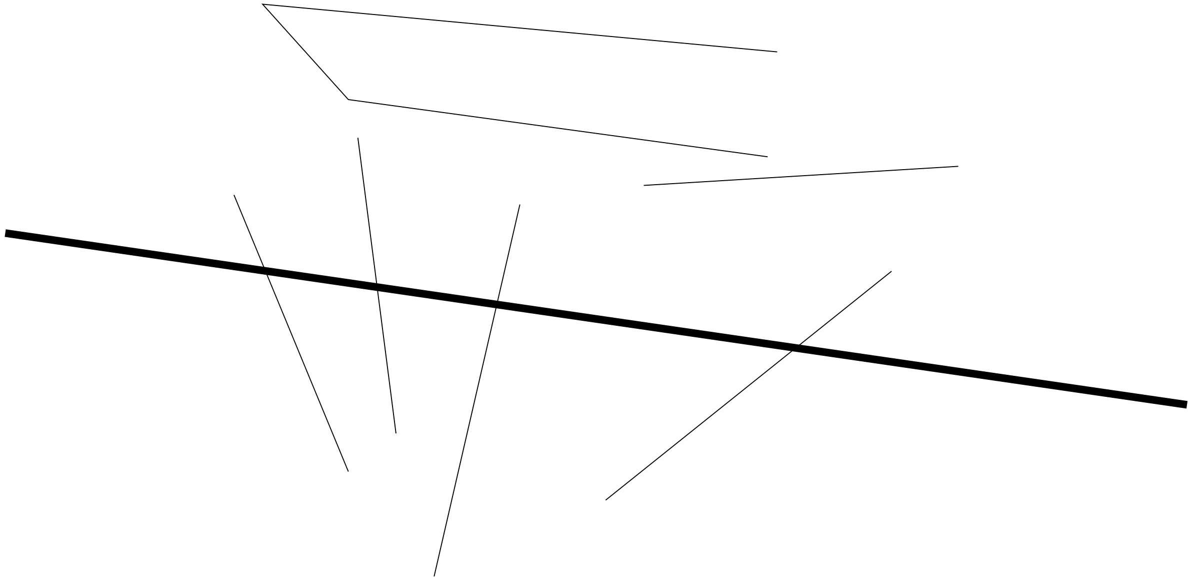
See Hildrum's thesis for details.

# From the Gadgets Dept.

Stabbing number: if you draw a line across your drawing, how many segments, triangles, . . . , do you intersect?

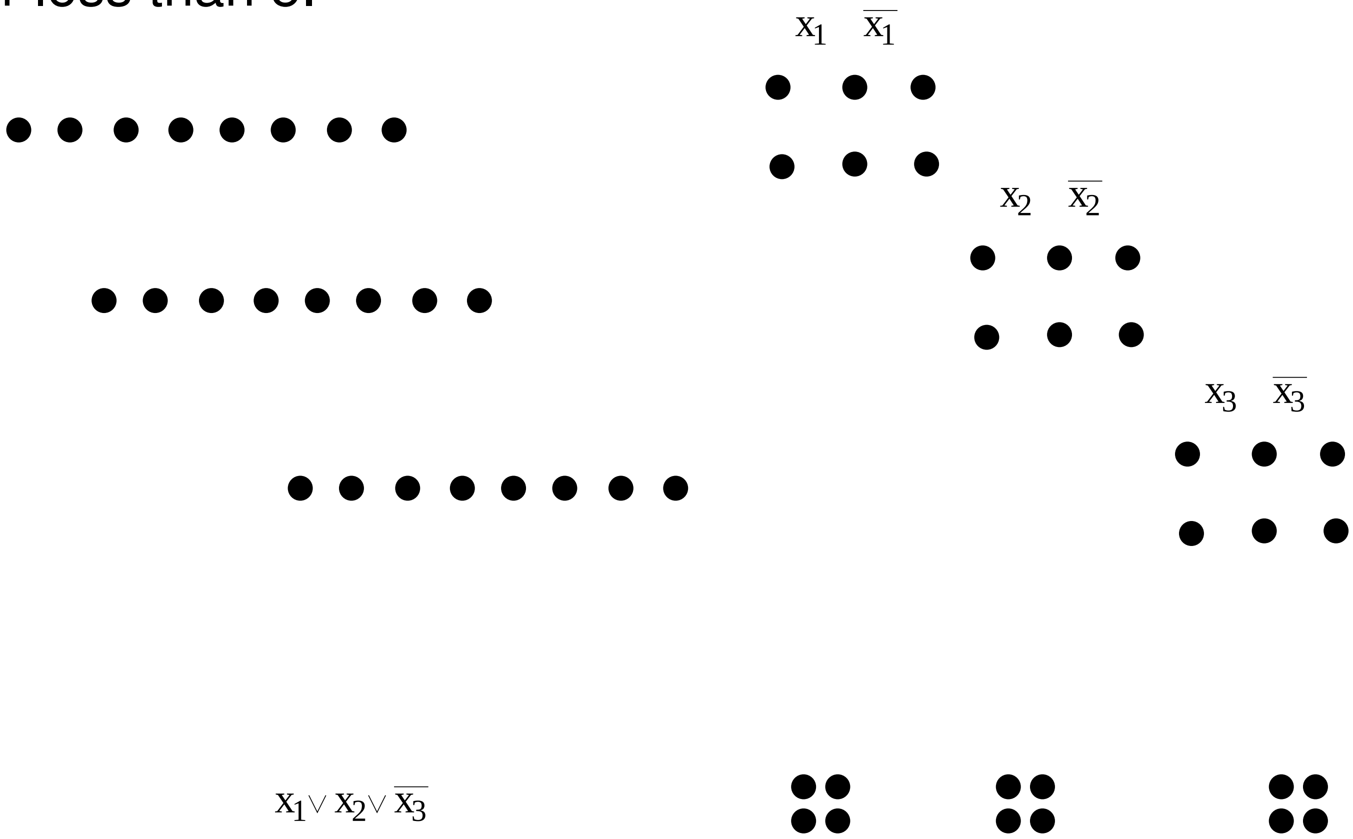
Determines running time of lots of algorithms.

For fun, let's try to find the matching (or spanning tree, or triangulation) of a set of points that has minimal stabbing number.



# Gadgets for sale, only 5 segments!

Reduction from 3SAT: $\phi$ is satisfiable iff the matching has stabbing number less than 5.



# Less geometry, more graphs!

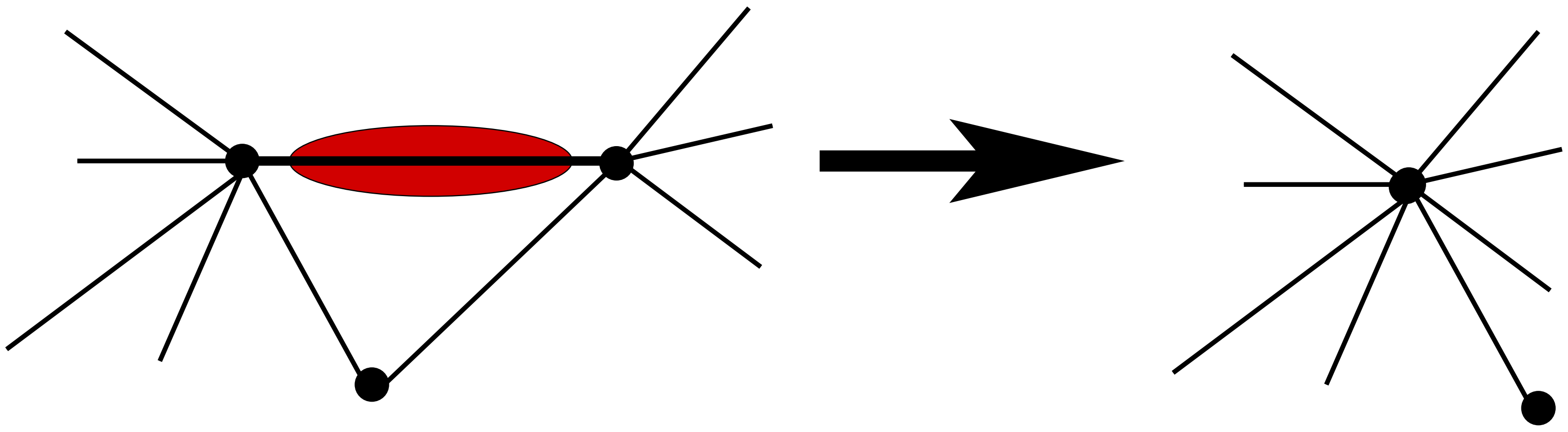
Concepts of the day:

- graph minors
- genus
- treewidth and branch width
- holes and anti-holes
- geometric graphs

# Graph minors

*Edge contraction*: take an edge, merge its two endpoints.

$H$ is a graph minor of $G$ if there is some sequence of contractions to turn $G$ into a supergraph of $H$.



# Graph Genus

*Genus* of a graph: the minimum-genus topology in which the graph can be drawn without crossings.

- embeddable in the plane: genus 0
- on a donut: genus 1
- on a pretzel: genus 3
- on a chunk of swiss cheese: genus ???

# Treewidth

Treewidth is a measure of how tree-like a graph is.

Create a tree from the graph:

- each node in the tree is a collection of graph vertices (a *bag*).
- each graph vertex is in some bag
- each graph edge has both endpoints in some bag
- each graph vertex is only included in bags of a subtree of the tree

Treewidth: size of largest bag – 1.

# Treewidth

- tree: treewidth 1
- series-parallel: treewidth 2
- planar: treewidth $O(\sqrt{n})$. Example: a $\sqrt{n} \times \sqrt{n}$ grid.
- planar, with no $r \times r$ grid minor: $O(r)$.

Generally, if $G$ doesn't have a $H$ as a minor, $tw(G)$ depends only on the treewidth of $H$ (if $G$ and $H$ are planar).

Treewidth decreases monotonically with contractions.

# Treewidth: yawn...

An algorithm for solving some hard problems:

1. Find the treewidth.
2. If it's large, say no answer.
3. If it's small, compute it in $2^{O(\mathrm{tw}(G))}n$ time.

# Treewidth: well, maybe I care

If you know your graph has bounded genus, you can solve all sorts of problems in time:

$$O(2^{\max\{a\theta,\gamma\}\tau 4\frac{\beta}{\delta}(g(G)+1)(\sqrt{k+1}+\mu\alpha\frac{\beta}{\delta}(g(G)+1))}n^{\max\{b,\lambda\}}$$

Isn't that **GREAT!**

Better result: if your graph does not have $H$ as a graph minor then you run in time

$$2^{O(\sqrt{k})}n^{O(1)}$$

Where the constants in the exponents depend on $H$.

# Treewidth papers

Demaine et al: Subexponential Parameterized Algorithms on Graphs of Bounded Genus and H-minor-free Graphs.

Demaine & Hajiaghayi: Equivalent of Local Treewidth and Linear Local Treewidth and its Algorithmic Applications.

Second paper: more similar stuff for PTAS. Improves some PTAS from $2^{2^{2^{O(1/\epsilon)}}} n^{O(1)}$ to $2^{O(1/\epsilon)} n^{O(1)}$.

# Less geometry, more graphs!

Concepts of the day:

- graph minors
- genus
- treewidth and branch width
- **holes and anti-holes**
- geometric graphs

# Potholes and sea urchins

*Perfect graph conjecture*: a graph is perfect iff it contains no holes or antiholes on an odd number of vertices.

*Weakly chordal graph*: one that has neither holes nor antiholes.

Hole: a cycle of 5 or more vertices that has no edge across the cycle (no chord).

Anti-hole: a hole in the complement graph.

Anti-hole $\neq$ clique – but they're close.

# Finding holes

1. Find 4 vertices that form a chordless path.

2. Do DFS; if we snake back to the path, output “hole!”

The paper has the precise tricks to make this run in $O(n + m^2)$ time.

Still open: $o(m^2)$ time.

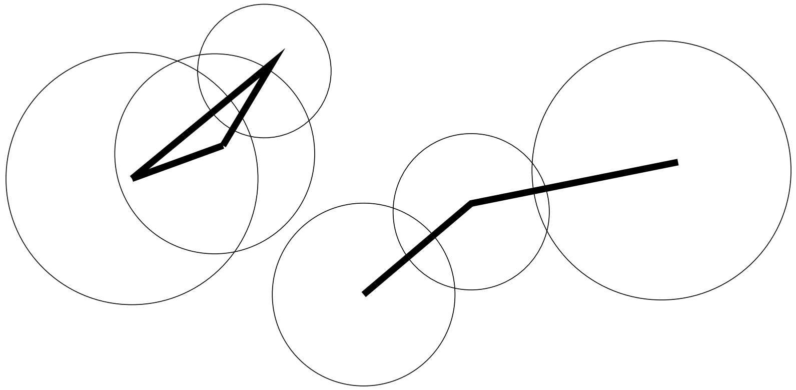
# A harmonious meld of geometry and graphs

Concepts of the day:

- graph minors
- genus
- treewidth and branch width
- holes and anti-holes
- **geometric graphs**

# Geometric graphs

Draw a bunch of segments (or triangles, circles, . . . ).



Geometric intersection graph:

- segments are the vertices
- edge between two vertices if the segments intersect

DFS takes $O(n+m)$ time. But these graphs can be dense.
Can we run various algorithms in time $o(n^2)$ ?

# Why?

The actual problem to solve:

- Given $G$, draw $G$ in $k$ different PSLGs where each PSLG has vertices in the same position. Minimize $k$.

For instance, lay down a circuit with as few layers as possible.

Testing bipartiteness: test whether the circuit has depth 2.

# Decrementality

Say we have a data structure that provides the following:

- Initialize over $n$ items in $O(n^{2-c})$
- Given an item not in the structure, return a neighbour in $O(n^{1-c})$.
- Remove an item in $O(n^{1-c})$.

Note: no insertions. ($c$ depends on the object type: ball, segment, etc).

Then we can run DFS in $O(n^{2-c})$ time:
1. remove the current item from the structure
2. query for a neighbour, recur into it
3. repeat 2 until no neighbours left

# Bipartiteness for balls in $R^d$

Faster time for bipartiteness: $O(n \lg n)$.

Basic idea: bipartite iff sparse.

Find a separator. If the separator is large, then the graph is dense.

For segments in $R^2$: same bound by a line sweep technique.

Lower "bounds": these problems are as hard as other more studied problems with equal (expected) bounds.

Lower bounds: testing for 3-colourability is hard.

# A confession

The most debauched thing I did was pet a shark.

Questions?