Voronoi Diagrams: Solving the Post Office Problem

Discrete and Computational Geometry
WS 2015/16
The Post Office Problem

What is the closest post office for a given position?

Goal: A data structure that solves this problem efficiently.
The Post Office Problem

What is the closest post office for a given position?

First: Partition into regions with the same answer.
The Post Office Problem

**Simplest Case:** Just two post offices $p_1$ and $p_2$

- Regions bounded by the bisector $b(p_1, p_2)$ of $p_1$ and $p_2$.
- Region of $p_1$: closed halfspace $H(p_1, p_2)$ containing $p_1$. 

![Diagram of the Post Office Problem](image)
The Post Office Problem

**General:** Set $P = \{p_1, \ldots, p_n\}$ of $n$ post office points in $\mathbb{R}^2$

- **Voronoi Cell** $V_P(p_i)$: all points of $\mathbb{R}^2$ that are at least as close to $p_i$ as to any other point of $P$. 

![Diagram of Voronoi diagram]
The Post Office Problem

**General:** Set $P = \{p_1, \ldots, p_n\}$ of $n$ post office points in $\mathbb{R}^2$

- **Voronoi Cell** $V_P(p_i)$: all points of $\mathbb{R}^2$ that are at least as close to $p_i$ as to any other point of $P$.

**Observation:**
- $V_P(p_i)$ is the intersection of all halfspaces $H(p_j, p_i)$ with $i \in \{1, \ldots, n\} \setminus \{j\}$
- $V_P(p_i)$ is convex and possibly unbounded

- **Voronoi Diagram** $VD(P)$: subdivision of $\mathbb{R}^2$ induced by the voronoi cells $V_P(p_i)$, for $i = 1, \ldots, n$.

- $p_i, p_j$ neighbors in $VD(P)$: $V_P(p_i)$ and $V_P(p_j)$ share an edge in $VD(P)$
The Post Office Problem

**General:** Set $P = \{p_1, \ldots, p_n\}$ of $n$ post office points in $\mathbb{R}^2$

**Note:** inside a cell of $VD(P)$: one unique closest point.
The Post Office Problem

**General:** Set \( P = \{p_1, \ldots, p_n\} \) of \( n \) post office points in \( \mathbb{R}^2 \)

**Question:** on an edge of \( \text{VD}(P) \)? \( \Rightarrow \) two closest points
The Post Office Problem

**General:** Set $P = \{p_1, \ldots, p_n\}$ of $n$ post office points in $\mathbb{R}^2$

**Question:** on a vertex of $VD(P)$?

$\Rightarrow \geq 3$ closest points
These Circles look familiar...
These Circles look familiar...
These Circles look familiar...
These Circles look familiar...
Voronoi Diagram / Delaunay Graph

- The **Voronoi diagram** $VD(P)$ of a point set $P$ is unique.
- $VD(P)$ is the dual of the **Delaunay graph** $DG(P)$.
- If $P$ has no 4 co-circular points then $VD(P)$ is the dual of the unique Delaunay triangulation $DT(P)$.
- The **unbounded cells** of $VD(P)$ contain the **extreme vertices** of $P$.
- The **duals** of the unbounded edges of $VD(P)$ form the **convex hull** of the point set.
- An edge of $DT(P)$ might not intersect its dual edge in $VD(P)$ and a triangle in $DT(P)$ might not contain its dual vertex in $VD(P)$, independent of whether or not $DT(P)$ is unique.
- $VD(P)$ has $O(n)$ vertices, edges, and regions.
Constructing Voronoi Diagrams

One approach: Construct from Delaunay Triangulation
Constructing Voronoi Diagrams

**Vertices:** Circumcenters of Delaunay triangles
Constructing Voronoi Diagrams

Vertices: Circumcenters of Delaunay triangles
Constructing Voronoi Diagrams

Edges: (parts of) bisectors of Delaunay edges.
Constructing Voronoi Diagrams

**Edges:** (parts of) bisectors of Delaunay edges.
Constructing Voronoi Diagrams

**One approach:** Construct from Delaunay Triangulation

⇒ **Time:** $O(n \log n)$ expected
Constructing Voronoi Diagrams

Different idea: Plane sweep: Maintain VD left of $\ell$. 
Constructing Voronoi Diagrams

**Problem:** How to discover new Voronoi regions in time?
Constructing Voronoi Diagrams

Solution: See sweep-line as additional line-shaped site.
Constructing Voronoi Diagrams

⇒ Left of $\ell$: Voronoi diagram of swept points and $\ell$. 
Constructing Voronoi Diagrams

⇒ Left of $\ell$: Voronoi edges and a Wavefront of parabolic arcs.
Constructing Voronoi Diagrams

⇒ Left of \( \ell \): Voronoi edges and a Wavefront of parabolic arcs.
Constructing Voronoi Diagrams

⇒ Left of $\ell$: Voronoi edges and a Wavefront of parabolic arcs.
Constructing Voronoi Diagrams

⇒ Left of \( \ell \): Voronoi edges and a Wavefront of parabolic arcs.

Point scanned: spike event
Constructing Voronoi Diagrams

⇒ Left of $\ell$: Voronoi edges and a Wavefront of parabolic arcs.

Parabula vanishes: site event
Constructing Voronoi Diagrams

⇒ Left of $\ell$: Voronoi edges and a Wavefront of parabolic arcs.
Constructing Voronoi Diagrams

⇒ Left of $\ell$: Voronoi edges and a Wavefront of parabolic arcs.
Constructing Voronoi Diagrams

⇒ Left of \( l \): Voronoi edges and a Wavefront of parabolic arcs.
Constructing Voronoi Diagrams

**Different idea:** Plane sweep: Maintain VD left of \( \ell \).

\[ \Rightarrow \textbf{Time: } O(n \log n) \text{ deterministic} \]
Constructing Voronoi Diagrams

Different idea: Plane sweep: Maintain VD left of $\ell$.

$\Rightarrow$ Time: $O(n \log n)$ deterministic

$\Rightarrow$ DT in $O(n \log n)$ deterministic!
The Post Office Problem

What is the closest post office for a given position?

Goal: A data structure that solves this problem efficiently.
The Post Office Problem

What is the closest post office for a given position?

**Theorem:** Given a triangulation $T$ for a set $P$ of $n$ points in $\mathbb{R}^2$, one can build in $O(n)$ time an data structure of size $O(n)$ that allows for any query point $q \in \text{conv}(P)$ to find in $O(\log n)$ time a triangle from $T$ containing $q$.

For example: **Kirkpatrick’s Hierarchy**

**Corollary:** The postoffice problem can be solved with $O(n \log n)$ preprocessing time and $O(\log n)$ query time with a data structure that uses $O(n)$ space.

- add Bounding box to $VD(P)$
- triangulate regions
- apply the theorem
More About Voronoi Diagrams

**Voronoi Diagrams**

- ... exist also in higher dimensions
- ... are used also for other objects than points
- ... can be defined also for other metrics than the euclidean distance

And there’s much more to know about them ...

... but: not anymore in this lecture. See for example the book *Voronoi Diagrams and Delaunay Triangulations* by Aurenhammer, Klein, Lee.

**Thank you for your attention!**