Computational Geometry

Voronoi Diagrams

or

The Post-Office Problem

Lecture #7

[Comp. Geom A&A : Chapter 7]

Steven Chaplick

Winter Semester 2019/20
The Post-Office Problem
The Post-Office Problem
The Post-Office Problem
The Post-Office Problem
The Post-Office Problem
The Post-Office Problem
The Post-Office Problem
The Post-Office Problem
The Post-Office Problem
The Post-Office Problem
The Post-Office Problem
The Post-Office Problem

\[ b(p, q) \]
The Post-Office Problem

\[ b(p, q) = \{ x \in \mathbb{R}^2 : |xp| = |xq| \} \]
The Post-Office Problem

\[ b(p, q) = \{ x \in \mathbb{R}^2 : |xp| = |xq| \} \]
The Post-Office Problem

$$b(p, q) = \{ x \in \mathbb{R}^2 : |xp| = |xq| \}$$

$$h(p, q) = \{ x : |xp| < |xq| \}$$
The Post-Office Problem

\[ b(p, q) = \{ x \in \mathbb{R}^2 : |xp| = |xq| \} \]

\[ h(p, q) = \{ x : |xp| < |xq| \} \]

\[ h(q, p) = \{ x : |xp| > |xq| \} \]
The Post-Office Problem

\[ b(p, q) = \{ x \in \mathbb{R}^2 : |xp| = |xq| \} \]

\[ h(p, q) = \{ x : |xp| < |xq| \} \]

\[ h(q, p) = \{ x : |xq| < |xp| \} \]
The Post-Office Problem
The Post-Office Problem

\[ P = \{ p_1, p_2, \ldots, p_n \} \]
The Post-Office Problem
The Post-Office Problem

\[ \text{Vor}(P) = \text{Voronoi diagram of } P \]
The Post-Office Problem

\[ \text{Vor}(P) = \text{Voronoi diagram of } P \]
The Post-Office Problem

\[ \text{Vor}(P) = \text{Voronoi diagram of } P \]
The Post-Office Problem

Tasks: 1) Define Voronoi cells, edges and vertices!
The Post-Office Problem

Tasks:  
1) Define Voronoi cells, edges and vertices!  
2) Are Voronoi cells convex?
The Voronoi diagram

Let $P$ be a set of points in the plane and let $p, p', p'' \in P$. 
The Voronoi diagram

Let \( P \) be a set of points in the plane and let \( p, p', p'' \in P \).
The Voronoi diagram

Let $P$ be a set of points in the plane and let $p, p', p'' \in P$. 

[Voronoi diagram] 

\[\text{Vor}(P)\]
The Voronoi diagram

Let $P$ be a set of points in the plane and let $p, p', p'' \in P$. 

$\mathcal{V}(\{p\}) = \text{Vor}(P)$
The Voronoi diagram

Let $P$ be a set of points in the plane and let $p, p', p'' \in P$. 

\[ \text{Voronoi diagram} \]

\[ \text{Voronoi cell} \]

\[ \mathcal{V}(\{p\}) = \mathcal{V}(p) = \]
The Voronoi diagram

Let $P$ be a set of points in the plane and let $p, p', p'' \in P$.

\[ \text{Voronoi diagram} \quad \text{Vor}(P) \]

\[ \text{Voronoi cell} \quad \mathcal{V}(\{p\}) = \mathcal{V}(p) = \{ x \in \mathbb{R}^2 : |xp| < |xq| \text{ for all } q \in P \setminus \{p\} \} \]
The Voronoi diagram

Let $P$ be a set of points in the plane and let $p, p', p'' \in P$.

\[ V(P) = \left\{ x \in \mathbb{R}^2 : |xp| < |xq| \text{ for all } q \in P \setminus \{p\} \right\} = \bigcap_{q \neq p} h(p, q) \]
The Voronoi diagram

Let $P$ be a set of points in the plane and let $p, p', p'' \in P$.

$V_{	ext{Voronoi diagram}}$

$V_{\{p\}} = \{x \in \mathbb{R}^2 : |xp| < |xq| \text{ for all } q \in P \setminus \{p\}\}$

$V_{\text{Voronoi cell}}$

$V(p) = \cap_{q \neq p} h(p, q)$
The Voronoi diagram

Let $P$ be a set of points in the plane and let $p, p', p'' \in P$.

$\mathcal{V}(\{p\}) = \mathcal{V}(p) = \{ x \in \mathbb{R}^2 : |xp| < |xq| \text{ for all } q \in P \setminus \{p\} \}$

$= \bigcap_{q \neq p} h(p, q)$

$\mathcal{V}(\{p, p'\}) = \quad$
The Voronoi diagram

Let $P$ be a set of points in the plane and let $p, p', p'' \in P$.

[Voronoi diagram]

$V(P) = \{ x \in \mathbb{R}^2 : |xp| < |xq| \text{ for all } q \in P \setminus \{p\} \} = \bigcap_{q \neq p} h(p, q)$

[Voronoi cell]

$V(\{p\}) = V(p) = \{ x \in \mathbb{R}^2 : |xp| < |xq| \text{ for all } q \in P \setminus \{p\} \}$

[Voronoi edge]

$V(\{p, p'\}) = \{ x : |xp| = |xp'| \text{ and } |xp| < |xq| \forall q \neq p, p' \}$
The Voronoi diagram

Let $P$ be a set of points in the plane and let $p, p', p'' \in P$.

**[Voronoi diagram]**

$$\text{Vor}(P)$$

**[Voronoi cell]**

$$\text{V}(\{p\}) = \text{V}(p) = \left\{ x \in \mathbb{R}^2 : |xp| < |xq| \text{ for all } q \in P \setminus \{p\} \right\}$$

$$= \bigcap_{q \neq p} h(p, q)$$

**[Voronoi edge]**

$$\text{V}(\{p, p'\}) = \left\{ x : |xp| = |xp'| \text{ and } |xp| < |xq| \text{ for all } q \neq p, p' \right\}$$

$$= \partial \text{V}(p) \cap \partial \text{V}(p')$$
The Voronoi diagram

Let $P$ be a set of points in the plane and let $p, p', p'' \in P$.

**Voronoi diagram**

$V(p) = \bigcap_{q \neq p} h(p, q)$

**Voronoi cell**

$V({p, p'}) = \{ x \in \mathbb{R}^2 : |xp| < |xq| \text{ for all } q \in P \setminus \{p\} \}$

**Voronoi edge**

$V({p}) = \partial V(p) \cap \partial V(p')$
The Voronoi diagram

Let $P$ be a set of points in the plane and let $p, p', p'\prime\prime \in P$.

**Voronoi diagram**

\[ \mathcal{V}(\{p\}) = \{ x \in \mathbb{R}^2 : |xp| < |xq| \text{ for all } q \in P \setminus \{p\} \} = \bigcap_{q \neq p} h(p, q) \]

**Voronoi cell**

\[ \mathcal{V}({\{p, p'\}}) = \{ x : |xp| = |xp'| \text{ and } |xp| < |xq| \text{ for all } q \neq p, p' \} = \text{rel-int}(\partial \mathcal{V}(p) \cap \partial \mathcal{V}(p')) \text{ (w/o the endpts)} \]

**Voronoi edge**

\[ \mathcal{V}(p, p') \]
The Voronoi diagram

Let $P$ be a set of points in the plane and let $p, p', p'' \in P$.

[Voronoi diagram]

\[ \text{Vor}(P) \]

[Voronoi cell]

\[ \text{V}({p}) = \text{V}(p) = \{ x \in \mathbb{R}^2 : |xp| < |xq| \text{ for all } q \in P \setminus \{p\} \} = \bigcap_{q \neq p} h(p, q) \]

[Voronoi edge]

\[ \text{V}({p, p'}) = \{ x : |xp| = |xp'| \text{ and } |xp| < |xq| \text{ for all } q \neq p, p' \} = \text{rel-int}(\partial \text{V}(p) \cap \partial \text{V}(p')) \text{ (w/o the endpts)} \]
The Voronoi diagram

Let $P$ be a set of points in the plane and let $p, p', p'' \in P$.

[Voronoi diagram]  \[ \text{Vor}(P) \]

[Voronoi cell]
\[
\mathcal{V}(\{p\}) = \mathcal{V}(p) = \{ x \in \mathbb{R}^2 : |xp| < |xq| \text{ for all } q \in P \setminus \{p\} \} \\
= \bigcap_{q \neq p} h(p, q)
\]

[Voronoi edge]
\[
\mathcal{V}(\{p, p'\}) = \{ x : |xp| = |xp'| \text{ and } |xp| < |xq| \text{ } \forall q \neq p, p' \} \\
= \text{rel-int}(\partial \mathcal{V}(p) \cap \partial \mathcal{V}(p')) \text{ (w/o the endpts)}
\]

[Voronoi vertex]
\[
\mathcal{V}(\{p, p', p''\})
\]
The Voronoi diagram

Let $P$ be a set of points in the plane and let $p, p', p'' \in P$.

**[Voronoi diagram]**

\[ \text{Vor}(P) \]

**[Voronoi cell]**

\[ \mathcal{V} \{p\} = \{ x \in \mathbb{R}^2 : |xp| < |xq| \text{ for all } q \in P \setminus \{p\} \} = \bigcap_{q \neq p} h(p, q) \]

**[Voronoi edge]**

\[ \mathcal{V} \{p, p'\} = \{ x : |xp| = |xp'| \text{ and } |xp| < |xq| \ \forall q \neq p, p' \} = \text{rel-int}(\partial \mathcal{V}(p) \cap \partial \mathcal{V}(p')) \text{ (w/o the endpts)} \]

**[Voronoi vertex]**

\[ \mathcal{V} \{p, p', p''\} = \partial \mathcal{V}(p) \cap \partial \mathcal{V}(p') \cap \partial \mathcal{V}(p'') \]
The Voronoi diagram

Let $P$ be a set of points in the plane and let $p, p', p'' \in P$.

[Voronoi diagram]  
\[ \text{Vor}(P) \]

[Voronoi cell]  
\[ \mathcal{V}(\{p\}) = \mathcal{V}(p) = \{ x \in \mathbb{R}^2 : |xp| < |xq| \text{ for all } q \in P \setminus \{p\} \} = \bigcap_{q \neq p} h(p, q) \]

[Voronoi edge]  
\[ \mathcal{V}(\{p, p'\}) = \{ x : |xp| = |xp'| \text{ and } |xp| < |xq| \text{ for all } q \neq p, p' \} = \text{rel-int}(\partial \mathcal{V}(p) \cap \partial \mathcal{V}(p')) \text{ (w/o the endpts)} \]

[Voronoi vertex]  
\[ \mathcal{V}(\{p, p', p''\}) = \partial \mathcal{V}(p) \cap \partial \mathcal{V}(p') \cap \partial \mathcal{V}(p'') = \{ x : |xp| = |xp'| = |xp''| \text{ and } |xp| \leq |xq| \text{ for all } q \} \]
The Voronoi diagram

Let $P$ be a set of points in the plane and let $p, p', p'' \in P$.

**[Voronoi diagram]**

![Voronoi diagram]

**[Voronoi cell]**

$$\mathcal{V}(\{p\}) = \mathcal{V}(p) = \{ x \in \mathbb{R}^2 : |xp| < |xq| \text{ for all } q \in P \setminus \{p\} \} = \bigcap_{q \neq p} h(p, q)$$

**[Voronoi edge]**

$$\mathcal{V}(\{p, p'\}) = \{ x : |xp| = |xp'| \text{ and } |xp| < |xq| \forall q \neq p, p' \} = \text{rel-int}(\partial \mathcal{V}(p) \cap \partial \mathcal{V}(p')) \text{ (w/o the endpts)}$$

**[Voronoi vertex]**

$$\mathcal{V}(\{p, p', p''\}) = \partial \mathcal{V}(p) \cap \partial \mathcal{V}(p') \cap \partial \mathcal{V}(p'') = \{ x : |xp| = |xp'| = |xp''| \text{ and } |xp| \leq |xq| \forall q \}$$
The Voronoi diagram

Let $P$ be a set of points in the plane and let $p, p', p'' \in P$.

**[Voronoi diagram]**

$Vornoi(diagram)$

**[Voronoi cell]**

$V({p}) = \{x \in \mathbb{R}^2 : |xp| < |xq| \text{ for all } q \in P \setminus \{p\}\} = \bigcap_{q \neq p} h(p, q)$

**[Voronoi edge]**

$V({p, p'}) = \{x : |xp| = |xp'| \text{ and } |xp| < |xq| \forall q \neq p, p'\} = \text{rel-int}(\partial V(p) \cap \partial V(p'))$ (w/o the endpts)

**[Voronoi vertex]**

$V({p, p', p''}) = \partial V(p) \cap \partial V(p') \cap \partial V(p'')$

$= \{x : |xp| = |xp'| = |xp''| \text{ and } |xp| \leq |xq| \forall q\}$
The Voronoi diagram

Let $P$ be a set of points in the plane and let $p, p', p'' \in P$.

[Voronoi diagram]

$\text{Vor}(P) \xrightarrow{\text{subdivision of } \mathbb{R}^2}$

[Voronoi cell]

$\mathcal{V}(\{p\}) = \mathcal{V}(p) = \{x \in \mathbb{R}^2 : |xp| < |xq| \text{ for all } q \in P \setminus \{p\}\} = \bigcap_{q \neq p} h(p, q)$

[Voronoi edge]

$\mathcal{V}(\{p, p'\}) = \{x : |xp| = |xp'| \text{ and } |xp| < |xq| \ \forall q \neq p, p'\} = \text{rel-int}(\partial \mathcal{V}(p) \cap \partial \mathcal{V}(p')) \ (\text{w/o the endpts})$

[Voronoi vertex]

$\mathcal{V}(\{p, p', p''\}) = \partial \mathcal{V}(p) \cap \partial \mathcal{V}(p') \cap \partial \mathcal{V}(p'') = \{x : |xp| = |xp'| = |xp''| \text{ and } |xp| \leq |xq| \ \forall q\}$
The Voronoi diagram

Let $P$ be a set of points in the plane and let $p, p', p'' \in P$.

**Voronoi diagram**

\[
\text{Vor}(P) \quad \text{subdivision of } \mathbb{R}^2
\]

**Voronoi cell**

\[
\mathcal{V}(\{p\}) = \mathcal{V}(p) = \{ x \in \mathbb{R}^2 : |xp| < |xq| \text{ for all } q \in P \setminus \{p\} \} = \bigcap_{q \neq p} h(p, q)
\]

**Voronoi edge**

\[
\mathcal{V}(\{p, p'\}) = \{ x : |xp| = |xp'| \text{ and } |xp| < |xq| \text{ for all } q \neq p, p' \} = \text{rel-int}(\partial\mathcal{V}(p) \cap \partial\mathcal{V}(p')) \quad (w/o \text{ the endpts})
\]

**Voronoi vertex**

\[
\mathcal{V}(\{p, p', p''\}) = \partial\mathcal{V}(p) \cap \partial\mathcal{V}(p') \cap \partial\mathcal{V}(p'') = \{ x : |xp| = |xp'| = |xp''| \text{ and } |xp| \leq |xq| \text{ for all } q \} \]
Overall Shape of $\text{Vor}(P)$

**Theorem.** Let $P \subset \mathbb{R}^2$ be a set of $n$ pts (called *sites*). If all sites are collinear, $\text{Vor}(P)$ consists of $n - 1$ parallel lines. Otherwise, $\text{Vor}(P)$ is connected and its edges are line segments or half-lines.
Overall Shape of Vor($P$)

**Theorem.** Let $P \subset \mathbb{R}^2$ be a set of $n$ pts (called *sites*). If all sites are collinear, Vor($P$) consists of $n - 1$ parallel lines. Otherwise, Vor($P$) is connected and its edges are line segments or half-lines.

**Proof.**
Overall Shape of Vor\( (P) \)

**Theorem.** Let \( P \subset \mathbb{R}^2 \) be a set of \( n \) pts (called *sites*). If all sites are collinear, \( \text{Vor}(P) \) consists of \( n - 1 \) parallel lines. Otherwise, \( \text{Vor}(P) \) is connected and its edges are line segments or half-lines.

**Proof.**

- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
-
Overall Shape of $\text{Vor}(P)$

**Theorem.** Let $P \subset \mathbb{R}^2$ be a set of $n$ pts (called sites). If all sites are collinear, $\text{Vor}(P)$ consists of $n - 1$ parallel lines. Otherwise, $\text{Vor}(P)$ is connected and its edges are line segments or half-lines.

**Proof.** Assume that $P$ is not collinear.
Overall Shape of Vor(P)

**Theorem.** Let $P \subset \mathbb{R}^2$ be a set of $n$ pts (called *sites*). If all sites are collinear, $\text{Vor}(P)$ consists of $n - 1$ parallel lines. Otherwise, $\text{Vor}(P)$ is connected and its edges are line segments or half-lines.

**Proof.** Assume that $P$ is not collinear.

– Assume that $\text{Vor}(P)$ contains an edge $e$ that is a full line, say, $e = b(p, q)$. 

![Diagram](image.png)
Overall Shape of Vor(\(P\))

**Theorem.** Let \(P \subset \mathbb{R}^2\) be a set of \(n\) pts (called *sites*). If all sites are collinear, Vor(\(P\)) consists of \(n - 1\) parallel lines. Otherwise, Vor(\(P\)) is connected and its edges are line segments or half-lines.

**Proof.** Assume that \(P\) is not collinear.

– Assume that Vor(\(P\)) contains an edge \(e\) that is a full line, say, \(e = b(p, q)\).

Let \(r \in P\) be not collinear with \(p\) and \(q\).
Overall Shape of Vor(P)

**Theorem.** Let $P \subset \mathbb{R}^2$ be a set of $n$ pts (called *sites*). If all sites are collinear, $\text{Vor}(P)$ consists of $n - 1$ parallel lines. Otherwise, $\text{Vor}(P)$ is connected and its edges are line segments or half-lines.

**Proof.** Assume that $P$ is not collinear.

– Assume that $\text{Vor}(P)$ contains an edge $e$ that is a full line, say, $e = b(p, q)$.

Let $r \in P$ be not collinear with $p$ and $q$. Then $e' = b(q, r)$ is not parallel to $e$. 

![Diagram showing Voronoi diagram with a full line and a point not collinear with other points.](image-url)
Overall Shape of Vor($P$)

**Theorem.** Let $P \subset \mathbb{R}^2$ be a set of $n$ pts (called sites). If all sites are collinear, Vor($P$) consists of $n - 1$ parallel lines. Otherwise, Vor($P$) is connected and its edges are line segments or half-lines.

**Proof.** Assume that $P$ is not collinear.

– Assume that Vor($P$) contains an edge $e$ that is a full line, say, $e = b(p, q)$.

Let $r \in P$ be not collinear with $p$ and $q$. Then $e' = b(q, r)$ is not parallel to $e$. 

$\Rightarrow$ $e \cap h(r, q)$ is closer to $r$ than to $p$ or $q.$
**Overall Shape of Vor(P)**

**Theorem.** Let $P \subset \mathbb{R}^2$ be a set of $n$ pts (called *sites*). If all sites are collinear, Vor($P$) consists of $n - 1$ parallel lines. Otherwise, Vor($P$) is connected and its edges are line segments or half-lines.

**Proof.** Assume that $P$ is not collinear.

– Assume that Vor($P$) contains an edge $e$ that is a full line, say, $e = b(p, q)$.

Let $r \in P$ be not collinear with $p$ and $q$. Then $e' = b(q, r)$ is not parallel to $e$.

$\Rightarrow e \cap h(r, q)$ is closer to $r$ than to $p$ or $q$.

$\Rightarrow e$ is bounded on at least one side. \(\square\)
Task: Construct a set $P$ of point sites such that $\text{Vor}(P)$ has a cell of linear complexity!
Complexity

Task: Construct a set $P$ of point sites such that $\text{Vor}(P)$ has a cell of linear complexity!
Complexity

**Task:** Construct a set $P$ of point sites such that $\text{Vor}(P)$ has a cell of linear complexity!
Complexity

**Task:** Construct a set $P$ of point sites such that $\text{Vor}(P)$ has a cell of linear complexity!
Task: Construct a set $P$ of point sites such that $\text{Vor}(P)$ has a cell of linear complexity!

Theorem. Given a set $P \subseteq \mathbb{R}^2$ of $n$ sites, $\text{Vor}(P)$ consists of at most $2n - 5$ vertices and $3n - 6$ edges.
Complexity

Task: Construct a set $P$ of point sites such that $\text{Vor}(P)$ has a cell of linear complexity!

Theorem. Given a set $P \subset \mathbb{R}^2$ of $n$ sites, $\text{Vor}(P)$ consists of at most $2n - 5$ vertices and $3n - 6$ edges.
Task: Construct a set $P$ of point sites such that $\text{Vor}(P)$ has a cell of linear complexity!

Theorem. Given a set $P \subset \mathbb{R}^2$ of $n$ sites, $\text{Vor}(P)$ consists of at most $2n - 5$ vertices and $3n - 6$ edges.

Proof. Euler
Complexity

Task: Construct a set $P$ of point sites such that $\text{Vor}(P)$ has a cell of linear complexity!

Theorem. Given a set $P \subset \mathbb{R}^2$ of $n$ sites, $\text{Vor}(P)$ consists of at most $2n - 5$ vertices and $3n - 6$ edges.

Proof. Problem: unbounded edges!

Euler
Complexity

**Task:** Construct a set $P$ of point sites such that $\text{Vor}(P)$ has a cell of linear complexity!

**Theorem.** Given a set $P \subset \mathbb{R}^2$ of $n$ sites, $\text{Vor}(P)$ consists of at most $2n - 5$ vertices and $3n - 6$ edges.

**Proof.** *Problem:* unbounded edges! 

$\Rightarrow$ can’t apply Euler directly, but…
Complexity

Task: Construct a set $P$ of point sites such that $\text{Vor}(P)$ has a cell of linear complexity!

Theorem. Given a set $P \subset \mathbb{R}^2$ of $n$ sites, $\text{Vor}(P)$ consists of at most $2n - 5$ vertices and $3n - 6$ edges.

Proof. Problem: unbounded edges! 
\[ \Rightarrow \text{can’t apply Euler directly, but…} \]
Complexity

**Task:** Construct a set $P$ of point sites such that $\text{Vor}(P)$ has a cell of linear complexity!

**Theorem.** Given a set $P \subset \mathbb{R}^2$ of $n$ sites, $\text{Vor}(P)$ consists of at most $2n - 5$ vertices and $3n - 6$ edges.

**Proof.** *Problem:* unbounded edges!

$\Rightarrow$ can’t apply Euler directly, but...
Task: Construct a set $P$ of point sites such that $\text{Vor}(P)$ has a cell of linear complexity!

Theorem. Given a set $P \subset \mathbb{R}^2$ of $n$ sites, $\text{Vor}(P)$ consists of at most $2n - 5$ vertices and $3n - 6$ edges.

Proof. Problem: unbounded edges!  
$\Rightarrow$ can’t apply Euler directly, but...  
$|F| = n$
Complexity

**Task:** Construct a set $P$ of point sites such that $\text{Vor}(P)$ has a cell of linear complexity!

**Theorem.** Given a set $P \subset \mathbb{R}^2$ of $n$ sites, $\text{Vor}(P)$ consists of at most $2n - 5$ vertices and $3n - 6$ edges.

**Proof.** Problem: unbounded edges!

$\Rightarrow$ can’t apply Euler directly, but…

$|F| = n \Rightarrow (|V| + 1) - |E| + n = 2$
Complexity

Task: Construct a set $P$ of point sites such that $\text{Vor}(P)$ has a cell of linear complexity!

Theorem. Given a set $P \subset \mathbb{R}^2$ of $n$ sites, $\text{Vor}(P)$ consists of at most $2n - 5$ vertices and $3n - 6$ edges.

Proof. Problem: unbounded edges!

$\Rightarrow$ can’t apply Euler directly, but...

$|F| = n \Rightarrow (|V| + 1) - |E| + n = 2$

min. degree 3
Complexity

**Task:** Construct a set $P$ of point sites such that Vor($P$) has a cell of linear complexity!

**Theorem.** Given a set $P \subset \mathbb{R}^2$ of $n$ sites, Vor($P$) consists of at most $2n - 5$ vertices and $3n - 6$ edges.

**Proof.** Problem: unbounded edges!

$\Rightarrow$ can’t apply Euler directly, but...

$|F| = n \Rightarrow (|V| + 1) - |E| + n = 2$

min. degree 3 $\Rightarrow 2|E| \geq 3(|V| + 1)$
**Task:** Construct a set $P$ of point sites such that $\text{Vor}(P)$ has a cell of linear complexity!

**Theorem.** Given a set $P \subset \mathbb{R}^2$ of $n$ sites, $\text{Vor}(P)$ consists of at most $2n - 5$ vertices and $3n - 6$ edges.

**Problem:** unbounded edges!

⇒ can’t apply Euler directly, but. . .

$|F| = n \Rightarrow (|V| + 1) - |E| + n = 2$

min. degree 3 ⇒ $2|E| \geq 3(|V| + 1)$

⇒ $(|V| + 1) - \frac{3}{2}(|V| + 1) + n \leq 2$
**Complexity**

**Task:** Construct a set \( P \) of point sites such that \( \text{Vor}(P) \) has a cell of linear complexity!

**Theorem.** Given a set \( P \subset \mathbb{R}^2 \) of \( n \) sites, \( \text{Vor}(P) \) consists of at most \( 2n - 5 \) vertices and \( 3n - 6 \) edges.

**Proof.** Problem: unbounded edges!

\[ |F| = n \Rightarrow (|V| + 1) - |E| + n = 2 \]

\[ \text{min. degree } 3 \Rightarrow 2|E| \geq 3(|V| + 1) \]

\[ \Rightarrow (|V| + 1) - \frac{3}{2}(|V| + 1) + n \leq 2 \]

\[ \Rightarrow \frac{1}{2}(|V| + 1) \leq n - 2 \]
Complexity

Task: Construct a set $P$ of point sites such that $\text{Vor}(P)$ has a cell of linear complexity!

Theorem. Given a set $P \subset \mathbb{R}^2$ of $n$ sites, $\text{Vor}(P)$ consists of at most $2n - 5$ vertices and $3n - 6$ edges.

Proof. Problem: unbounded edges!

$\Rightarrow$ can’t apply Euler directly, but...

$|F| = n \Rightarrow (|V| + 1) - |E| + n = 2$

min. degree 3 $\Rightarrow 2|E| \geq 3(|V| + 1)$

$\Rightarrow (|V| + 1) - \frac{3}{2}(|V| + 1) + n \leq 2$

$\Rightarrow \frac{1}{2}(|V| + 1) \leq n - 2$
Characterization of Voronoi vtc and edges

\[ C_P(x) := \text{largest circle centered at } x \text{ w/o sites in its interior} \]
Characterization of Voronoi vtc and edges

\[ C_P(x) := \text{largest circle centered at } x \text{ w/o sites in its interior} \]
Characterization of Voronoi vtc and edges

\( C_P(x) := \text{largest circle centered at } x \text{ w/o sites in its interior} \)

**Theorem:** (i) \( x \) Voronoi vtx \( \iff \)
Characterization of Voronoi vtc and edges

\[ C_P(x) := \text{largest circle centered at } x \text{ w/o sites in its interior} \]

**Theorem:** (i) \( x \) Voronoi vtx \( \iff |C_P(x) \cap P| \geq 3 \)
Characterization of Voronoi vtc and edges

\[ C_P(x) := \text{largest circle centered at } x \text{ w/o sites in its interior} \]

**Theorem:**

(i) \( x \) is a Voronoi vertex \( \iff |C_P(x) \cap P| \geq 3 \)

(ii) \( b(p, p') \) contains a Voronoi edge \( \iff \)
Characterization of Voronoi vtc and edges

\[ C_P(x) := \text{largest circle centered at } x \text{ w/o sites in its interior} \]

**Theorem:**

(i) \( x \) Voronoi vtx \( \iff \) \( |C_P(x) \cap P| \geq 3 \)

(ii) \( b(p, p') \) contains a Voronoi edge \( \iff \exists x \in b(p, p') : C_P(x) \cap P = \{p, p'\} \)
Computation

Brute force:
**Computation**

**Brute force:** For each $p \in P$, compute $\mathcal{V}(p) = \bigcap_{p' \neq p} h(p, p')$. 
Computation

**Brute force:** For each $p \in P$, compute $\nu(p) = \bigcap_{p' \neq p} h(p, p')$.

[Ch. 2, map-overlay / line-segment alg]
Computation

**Brute force:** For each $p \in P$, compute $\mathcal{V}(p) = \bigcap_{p' \neq p} h(p, p')$.

[Ch. 2, map-overlay / line-segment alg] $O(n \log^2 n)$ time
Computation

**Brute force:** For each $p \in P$, compute $\mathcal{V}(p) = \bigcap_{p' \neq p} h(p, p')$.

[Ch. 2, map-overlay / line-segment alg] \(O(n \log^2 n)\) time

[Ch. 4, half-plane intersection]
Computation

Brute force: For each $p \in P$, compute $\mathcal{V}(p) = \bigcap_{p' \neq p} h(p, p')$.

[Ch. 2, map-overlay / line-segment alg] $O(n \log^2 n)$ time

[Ch. 4, half-plane intersection] $O(n \log n)$ time
Computation

**Brute force:** For each \( p \in P \), compute \( \mathcal{V}(p) = \bigcap_{p' \neq p} h(p, p') \).

[Ch. 2, map-overlay / line-segment alg] \( O(n \log^2 n) \) time
[Ch. 4, half-plane intersection] \( O(n \log n) \) time
Computation

Brute force: For each $p \in P$, compute $\mathcal{V}(p) = \bigcap_{p' \neq p} h(p, p')$.

- [Ch. 2, map-overlay / line-segment alg] $O(n \log^2 n)$ time
- [Ch. 4, half-plane intersection] $O(n \log n)$ time

In total: $O(n^2 \log n)$ time
Computation

**Brute force:** For each \( p \in P \), compute \( V(p) = \bigcap_{p' \neq p} h(p, p') \).

- [Ch. 2, map-overlay / line-segment alg] \( O(n \log^2 n) \) time
- [Ch. 4, half-plane intersection] \( O(n \log n) \) time

*in total: \( O(n^2 \log n) \) time*

– but the complexity of \( \text{Vor}(P) \) is *linear!*
Computations

**Brute force:** For each $p \in P$, compute $\mathcal{V}(p) = \bigcap_{p' \neq p} h(p, p')$.

- [Ch. 2, map-overlay / line-segment alg] $O(n \log^2 n)$ time
- [Ch. 4, half-plane intersection] $O(n \log n)$ time

In total: $O(n^2 \log n)$ time

– but the complexity of $\text{Vor}(P)$ is *linear!*

**Sweep?**
**Computation**

**Brute force:** For each $p \in P$, compute $V(p) = \bigcap_{p' \neq p} h(p, p')$.

- [Ch. 2, map-overlay / line-segment alg] $O(n \log^2 n)$ time
- [Ch. 4, half-plane intersection] $O(n \log n)$ time

in total: $O(n^2 \log n)$ time

– but the complexity of $\text{Vor}(P)$ is *linear*!

**Sweep?**
Computation

**Brute force:** For each $p \in P$, compute $\mathcal{V}(p) = \bigcap_{p' \neq p} h(p, p')$.

- [Ch. 2, map-overlay / line-segment alg] $O(n \log^2 n)$ time
- [Ch. 4, half-plane intersection] $O(n \log n)$ time

*in total: $O(n^2 \log n)$ time*

– but the complexity of $\text{Vor}(P)$ is *linear*!

Sweep?
Computation

**Brute force:** For each \( p \in P \), compute \( V(p) = \bigcap_{p' \neq p} h(p, p') \).

- [Ch. 2, map-overlay / line-segment alg] \( O(n \log^2 n) \) time
- [Ch. 4, half-plane intersection] \( O(n \log n) \) time

In total: \( O(n^2 \log n) \) time

– but the complexity of \( \text{Vor}(P) \) is linear!

Sweep?
Computation

**Brute force:** For each $p \in P$, compute $V(p) = \bigcap_{p' \neq p} h(p, p')$.

- [Ch. 2, map-overlay / line-segment alg] $O(n \log^2 n)$ time
- [Ch. 4, half-plane intersection] $O(n \log n)$ time

in total: $O(n^2 \log n)$ time

– but the complexity of $\text{Vor}(P)$ is linear!

Sweep?

Problem: We don’t know all defining sites yet :( 
Computation

**Brute force:** For each \( p \in P \), compute \( V(p) = \bigcap_{p' \neq p} h(p, p') \).

[Ch. 2, map-overlay / line-segment alg] \( O(n \log^2 n) \) time

[Ch. 4, half-plane intersection] \( O(n \log n) \) time

\[ \text{in total: } O(n^2 \log n) \text{ time} \]

– but the complexity of \( \text{Vor}(P) \) is **linear**!

Sweep?

Problem: We don’t know all defining sites yet :(
Sweep?

Which part of the plane above $\ell$ is fixed by what we’ve seen?
Sweep?

Which part of the plane above $\ell$ is fixed by what we've seen?

$\bullet p$

------------------- $\ell$
Sweep?

Which part of the plane above $\ell$ is fixed by what we’ve seen?
Sweep?

Which part of the plane above $\ell$ is fixed by what we’ve seen?
Sweep?

Which part of the plane above \( \ell \) is fixed by what we’ve seen?
Sweep?

Which part of the plane above \( \ell \) is fixed by what we’ve seen?

**Task:** Compute \( f_\ell^p \) for \( p = (0, 1) \) and \( \ell: y = -1 \)!
Which part of the plane above \( \ell \) is fixed by what we’ve seen?

Solution:
\( f_p^\ell \) is the parabola with focus \( p \) and directrix \( \ell \).

Task: Compute \( f_p^\ell \) for \( p = (0, 1) \) and \( \ell: y = -1 \)!
Sweep?

Which part of the plane above ℓ is fixed by what we’ve seen?

Solution: $f^\ell_p$ is the parabola with focus $p$ and directrix ℓ.

Task: Compute $f^\ell_p$ for $p = (0, 1)$ and ℓ: $y = -1$!

Definition. beachline $\beta \equiv$ lower envelope of $(f^\ell_p)_{p \in P \cap ℓ^+}$
Sweep?

Which part of the plane above $\ell$ is fixed by what we’ve seen?

\[
\ell^+ \quad f_p^\ell \quad (x, f_p^\ell(x))
\]

**Solution:**

$f_p^\ell$ is the parabola with focus $p$ and directrix $\ell$.

**Task:** Compute $f_p^\ell$ for $p = (0, 1)$ and $\ell : y = -1$!

**Definition.** beachline $\beta \equiv$ lower envelope of $(f_p^\ell)_{p \in P \cap \ell^+}$
Sweep?

Which part of the plane above ℓ is fixed by what we’ve seen?

\[ f^ℓ_p \]

Solution:

\[ f^ℓ_p \] is the parabola with focus \( p \) and directrix \( ℓ \).

Task:

Compute \( f^ℓ_p \) for \( p = (0, 1) \) and \( ℓ : y = -1 \! \).

Definition.

\[ \text{beachline} \ \beta \equiv \ \text{lower envelope of} \ (f^ℓ_p)_{p \in P \cap ℓ^+} \]
Sweep?

Which part of the plane above $\ell$ is fixed by what we’ve seen?

Definition. beachline $\beta \equiv$ lower envelope of $(f^\ell_p)_{p \in P \cap \ell^+}$

Solution:

$f^\ell_p$ is the parabola with focus $p$ and directrix $\ell$.

Task: Compute $f^\ell_p$ for $p = (0, 1)$ and $\ell: y = -1$!
Sweep?

Which part of the plane above $\ell$ is fixed by what we’ve seen?

Solution:

$f_p^\ell$ is the parabola with focus $p$ and directrix $\ell$.

Task:

Compute $f_p^\ell$ for $p = (0, 1)$ and $\ell: y = -1$!

Definition. beachline $\beta \equiv$ lower envelope of $(f_p^\ell)_{p \in P \cap \ell^+}$
Which part of the plane above $\ell$ is fixed by what we’ve seen?

Solution:

$f_p^\ell$ is the parabola with focus $p$ and directrix $\ell$.

Task: Compute $f_p^\ell$ for $p = (0, 1)$ and $\ell: y = -1$!

Definition. beachline $\beta \equiv$ lower envelope of $(f_p^\ell)_{p \in P \cap \ell^+}$
Sweep?

Which part of the plane above $\ell$ is fixed by what we’ve seen?

Solution:
$f^p_\ell$ is the parabola with focus $p$ and directrix $\ell$.

Task:
Compute $f^p_\ell$ for $p = (0, 1)$ and $\ell:: y = -1$!

Definition.
beachline $\beta \equiv \text{lower envelope of } (f^p_\ell)_{p \in P \cap \ell^+}$

Observation.
$\beta$ is $x$-monotone.
The beachline $\beta$

**Question:** What does $\beta$ have to do with Vor($P$)?
The beachline $\beta$

**Question:** What does $\beta$ have to do with Vor($P$)?
The beachline $\beta$

**Question:** What does $\beta$ have to do with $\text{Vor}(P)$?

**Answer:** “Breakpoints” of $\beta$ trace out the Voronoi edges!
The beachline $\beta$

**Question:** What does $\beta$ have to do with $\text{Vor}(P)$?

**Answer:** “Breakpoints” of $\beta$ trace out the Voronoi edges!

**Lemma.** New arcs on $\beta$ only appear through site events.
The beachline $\beta$

**Question:** What does $\beta$ have to do with $\text{Vor}(P)$?

**Answer:** “Breakpoints” of $\beta$ trace out the Voronoi edges!

**Lemma.** New arcs on $\beta$ only appear through site events, that is, whenever $\ell$ hits a new site.
The beachline $\beta$

**Question:** What does $\beta$ have to do with $\text{Vor}(P)$?

**Answer:** “Breakpoints” of $\beta$ trace out the Voronoi edges!

**Lemma.** New arcs on $\beta$ only appear through *site events*, that is, whenever $\ell$ hits a new site.

**Corollary.** $\beta$ consists of at most $2n - 1$ arcs.
The beachline $\beta$

**Question:** What does $\beta$ have to do with $\text{Vor}(P)$?

**Answer:** “Breakpoints” of $\beta$ trace out the Voronoi edges!

**Lemma.** New arcs on $\beta$ only appear through site events, that is, whenever $\ell$ hits a new site.

**Corollary.** $\beta$ consists of at most $2n - 1$ arcs.

**Definition.** Circle event: $\ell$ reaches lowest pt of a circle through three sites above $\ell$ whose arcs are consecutive on $\beta$. 

---

![Diagram](image.png)
The beachline $\beta$

**Question:** What does $\beta$ have to do with $\text{Vor}(P)$?

**Answer:** “Breakpoints” of $\beta$ trace out the Voronoi edges!

**Lemma.** New arcs on $\beta$ only appear through site events, that is, whenever $\ell$ hits a new site.

**Corollary.** $\beta$ consists of at most $2n - 1$ arcs.

**Definition.** *Circle event:* $\ell$ reaches lowest pt of a circle through three sites above $\ell$ whose arcs are consecutive on $\beta$.

**Lemma.** Arcs disappear from $\beta$ only at circle events.
The beachline $\beta$

**Question:** What does $\beta$ have to do with $\text{Vor}(P)$?

**Answer:** “Breakpoints” of $\beta$ trace out the Voronoi edges!

**Lemma.** New arcs on $\beta$ only appear through site events, that is, whenever $\ell$ hits a new site.

**Corollary.** $\beta$ consists of at most $2n - 1$ arcs.

**Definition.** *Circle event:* $\ell$ reaches lowest pt of a circle through three sites above $\ell$ whose arcs are consecutive on $\beta$.

**Lemma.** Arcs disappear from $\beta$ only at circle events.

**Lemma.** The Voronoi vtc correspond 1:1 to circle events.
Fortune’s Sweep

VoronoiDiagram($P \subset \mathbb{R}^2$)

\begin{align*}
Q & \leftarrow \text{new PriorityQueue}(P) \quad // \text{site events sorted by } y\text{-coord.} \\
T & \leftarrow \text{new BalancedBinarySearchTree()} \quad // \text{sweep status } (\beta) \\
D & \leftarrow \text{new DCEL()} \quad // \text{to-be Vor}(P) \\
\text{while not } & Q.\text{empty()} \text{ do} \\
\quad & \text{treat remaining int. nodes of } T \ (\equiv \text{unbnd. edges of Vor}(P)) \\
\text{return } & D
\end{align*}
VoronoiDiagram\((P \subset \mathbb{R}^2)\)

\[Q \leftarrow \text{new PriorityQueue}(P)\] // site events sorted by y-coord.

\[T \leftarrow \text{new BalancedBinarySearchTree}()\] // sweep status (\(\beta\))

\[D \leftarrow \text{new DCEL}()\] // to-be Vor\((P)\)

\[\text{while not } Q.\text{empty}() \text{ do}
\]

\[p \leftarrow Q.\text{ExtractMax}()\]

\[\begin{align*}
\text{if } p \text{ site event then} & \\
\text{HandleSiteEvent}(p) & \\
\text{else} & \\
\alpha & \leftarrow \text{arc on } \beta \text{ that will disappear} \\
\text{HandleCircleEvent}(\alpha) & \\
\end{align*}\]

\[\text{treat remaining int. nodes of } T \ (\equiv \text{unbnd. edges of Vor}(P))\]

\[\text{return } D\]
Handling Events

HandleSiteEvent(point $p$)

HandleCircleEvent(arc $\alpha$)
Handling Events

HandleSiteEvent(point \( p \))

- Search in \( T \) for the arc \( \alpha \) vertically above \( p \).
  If \( \alpha \) has pointer to circle event in \( Q \), delete this event.

HandleCircleEvent(arc \( \alpha \))
Handling Events

HandleSiteEvent(point $p$)

- Search in $\mathcal{T}$ for the arc $\alpha$ vertically above $p$. If $\alpha$ has pointer to circle event in $Q$, delete this event.

- Split $\alpha$ into $\alpha_0$ and $\alpha_2$. Let $\alpha_1$ be the new arc of $p$.

HandleCircleEvent(arc $\alpha$)
Handling Events

HandleSiteEvent(point p)

- Search in $\mathcal{T}$ for the arc $\alpha$ vertically above $p$. If $\alpha$ has pointer to circle event in $Q$, delete this event.

- Split $\alpha$ into $\alpha_0$ and $\alpha_2$. Let $\alpha_1$ be the new arc of $p$.

HandleCircleEvent(arc $\alpha$)
Handling Events

HandleSiteEvent(point \( p \))

- Search in \( \mathcal{T} \) for the arc \( \alpha \) vertically above \( p \).
  If \( \alpha \) has pointer to circle event in \( Q \), delete this event.

- Split \( \alpha \) into \( \alpha_0 \) and \( \alpha_2 \).
  Let \( \alpha_1 \) be the new arc of \( p \).

HandleCircleEvent(arc \( \alpha \))
Handling Events

HandleSiteEvent(point $p$)

- Search in $\mathcal{T}$ for the arc $\alpha$ vertically above $p$. If $\alpha$ has pointer to circle event in $Q$, delete this event.

- Split $\alpha$ into $\alpha_0$ and $\alpha_2$. Let $\alpha_1$ be the new arc of $p$.

HandleCircleEvent(arc $\alpha$)

In $\mathcal{T}$:
Handling Events

HandleSiteEvent(point p)

- Search in \( \mathcal{T} \) for the arc \( \alpha \) vertically above \( p \).
  If \( \alpha \) has pointer to circle event in \( Q \), delete this event.

- Split \( \alpha \) into \( \alpha_0 \) and \( \alpha_2 \).
  Let \( \alpha_1 \) be the new arc of \( p \).

HandleCircleEvent(arc \( \alpha \))
Handling Events

HandleSiteEvent(point \( p \))

- Search in \( \mathcal{T} \) for the arc \( \alpha \) vertically above \( p \).
  If \( \alpha \) has pointer to circle event in \( Q \), delete this event.

- Split \( \alpha \) into \( \alpha_0 \) and \( \alpha_2 \).
  Let \( \alpha_1 \) be the new arc of \( p \).

- Add Vor-edges \( \langle q, p \rangle \) and \( \langle p, q \rangle \) to DCEL.

HandleCircleEvent(arc \( \alpha \))

In \( \mathcal{T} \):
Handling Events

HandleSiteEvent(point \( p \))

- Search in \( T \) for the arc \( \alpha \) vertically above \( p \).
  If \( \alpha \) has pointer to circle event in \( Q \), delete this event.

- Split \( \alpha \) into \( \alpha_0 \) and \( \alpha_2 \).
  Let \( \alpha_1 \) be the new arc of \( p \).

- Add Vor-edges \( \langle q, p \rangle \) and \( \langle p, q \rangle \) to DCEL.

- Check \( \langle \cdot, \alpha_0, \alpha_1 \rangle \) and \( \langle \alpha_1, \alpha_2, \cdot \rangle \) for circle events.

HandleCircleEvent(arc \( \alpha \))
Handling Events

HandleSiteEvent(point p)

• Search in $T$ for the arc $\alpha$ vertically above $p$.
  If $\alpha$ has pointer to circle event in $Q$, delete this event.

• Split $\alpha$ into $\alpha_0$ and $\alpha_2$.
  Let $\alpha_1$ be the new arc of $p$.

• Add Vor-edges $\langle q, p \rangle$ and $\langle p, q \rangle$ to DCEL.

• Check $\langle \cdot, \alpha_0, \alpha_1 \rangle$ and $\langle \alpha_1, \alpha_2, \cdot \rangle$ for circle events.

HandleCircleEvent(arc $\alpha$)
Handling Events

HandleSiteEvent(point \( p \))

- Search in \( T \) for the arc \( \alpha \) vertically above \( p \).
  If \( \alpha \) has pointer to circle event in \( Q \), delete this event.
- Split \( \alpha \) into \( \alpha_0 \) and \( \alpha_2 \).
  Let \( \alpha_1 \) be the new arc of \( p \).
- Add Vor-edges \( \langle q, p \rangle \) and \( \langle p, q \rangle \) to DCEL.
- Check \( \langle \cdot, \alpha_0, \alpha_1 \rangle \) and \( \langle \alpha_1, \alpha_2, \cdot \rangle \) for circle events.

HandleCircleEvent(arc \( \alpha \))

- \( T \).delete(\( \alpha \)); update breakpts
Handling Events

HandleSiteEvent(point \( p \))

- Search in \( \mathcal{T} \) for the arc \( \alpha \) vertically above \( p \). If \( \alpha \) has pointer to circle event in \( \mathcal{Q} \), delete this event.
- Split \( \alpha \) into \( \alpha_0 \) and \( \alpha_2 \). Let \( \alpha_1 \) be the new arc of \( p \).
- Add Vor-edges \( \langle q, p \rangle \) and \( \langle p, q \rangle \) to DCEL.
- Check \( \langle \cdot, \alpha_0, \alpha_1 \rangle \) and \( \langle \alpha_1, \alpha_2, \cdot \rangle \) for circle events.

HandleCircleEvent(arc \( \alpha \))

- \( \mathcal{T}.\text{delete}(\alpha) \); update breakpts
- Delete all circle events involving \( \alpha \) from \( \mathcal{Q} \).
Handling Events

HandleSiteEvent(point p)

- Search in $T$ for the arc $\alpha$ vertically above $p$. If $\alpha$ has pointer to circle event in $Q$, delete this event.
- Split $\alpha$ into $\alpha_0$ and $\alpha_2$. Let $\alpha_1$ be the new arc of $p$.
- Add Vor-edges $\langle q, p \rangle$ and $\langle p, q \rangle$ to DCEL.
- Check $\langle \cdot, \alpha_0, \alpha_1 \rangle$ and $\langle \alpha_1, \alpha_2, \cdot \rangle$ for circle events.

HandleCircleEvent(arc $\alpha$)

- $T$.delete($\alpha$); update breakpts
- Delete all circle events involving $\alpha$ from $Q$.
- Add Vor-vtx $\alpha_{\text{left}} \cap \alpha_{\text{right}}$ and Vor-edge $\langle \alpha_{\text{left}}, \alpha_{\text{right}} \rangle$ to DCEL.
Handling Events

HandleSiteEvent(point \( p \))

- Search in \( \mathcal{T} \) for the arc \( \alpha \) vertically above \( p \). If \( \alpha \) has pointer to circle event in \( Q \), delete this event.
- Split \( \alpha \) into \( \alpha_0 \) and \( \alpha_2 \). Let \( \alpha_1 \) be the new arc of \( p \).
- Add Vor-edges \( \langle q, p \rangle \) and \( \langle p, q \rangle \) to DCEL.
- Check \( \langle \cdot, \alpha_0, \alpha_1 \rangle \) and \( \langle \alpha_1, \alpha_2, \cdot \rangle \) for circle events.

HandleCircleEvent(arc \( \alpha \))

- \( \mathcal{T}.\text{delete}(\alpha) \); update breakpts.
- Delete all circle events involving \( \alpha \) from \( Q \).
- Add Vor-vtx \( \alpha_{\text{left}} \cap \alpha_{\text{right}} \) and Vor-edge \( \langle \alpha_{\text{left}}, \alpha_{\text{right}} \rangle \) to DCEL.
## Handling Events

**HandleSiteEvent(point p)**

- Search in $\mathcal{T}$ for the arc $\alpha$ vertically above $p$. If $\alpha$ has pointer to circle event in $\mathcal{Q}$, delete this event.
- Split $\alpha$ into $\alpha_0$ and $\alpha_2$. Let $\alpha_1$ be the new arc of $p$.
- Add Vor-edges $\langle q, p \rangle$ and $\langle p, q \rangle$ to DCEL.
- Check $\langle \cdot, \alpha_0, \alpha_1 \rangle$ and $\langle \alpha_1, \alpha_2, \cdot \rangle$ for circle events.

**HandleCircleEvent(arc $\alpha$)**

- $\mathcal{T}.\text{delete}(\alpha)$; update breakpts
- Delete all circle events involving $\alpha$ from $\mathcal{Q}$.
- Add Vor-vtx $\alpha_{\text{left}} \cap \alpha_{\text{right}}$ and Vor-edge $\langle \alpha_{\text{left}}, \alpha_{\text{right}} \rangle$ to DCEL.
Handling Events

HandleSiteEvent(point \( p \))

- Search in \( \mathcal{T} \) for the arc \( \alpha \) vertically above \( p \). If \( \alpha \) has pointer to circle event in \( Q \), delete this event.
- Split \( \alpha \) into \( \alpha_0 \) and \( \alpha_2 \). Let \( \alpha_1 \) be the new arc of \( p \).
- Add Vor-edges \( \langle q, p \rangle \) and \( \langle p, q \rangle \) to DCEL.
- Check \( \langle \cdot, \alpha_0, \alpha_1 \rangle \) and \( \langle \alpha_1, \alpha_2, \cdot \rangle \) for circle events.

HandleCircleEvent(arc \( \alpha \))

- \( \mathcal{T}.delete(\alpha) \); update breakpts
- Delete all circle events involving \( \alpha \) from \( Q \).
- Add Vor-vtx \( \alpha_{\text{left}} \cap \alpha_{\text{right}} \) and Vor-edge \( \langle \alpha_{\text{left}}, \alpha_{\text{right}} \rangle \) to DCEL.
- Check \( \langle \cdot, \alpha_{\text{left}}, \alpha_{\text{right}} \rangle \) and \( \langle \alpha_{\text{left}}, \alpha_{\text{right}}, \cdot \rangle \) for circle events.
Handling Events

HandleSiteEvent(point \( p \))

- Search in \( T \) for the arc \( \alpha \) vertically above \( p \). If \( \alpha \) has pointer to circle event in \( Q \), delete this event.
- Split \( \alpha \) into \( \alpha_0 \) and \( \alpha_2 \). Let \( \alpha_1 \) be the new arc of \( p \).
- Add Vor-edges \( \langle q, p \rangle \) and \( \langle p, q \rangle \) to DCEL.
- Check \( \langle \cdot, \alpha_0, \alpha_1 \rangle \) and \( \langle \alpha_1, \alpha_2, \cdot \rangle \) for circle events.

HandleCircleEvent(arc \( \alpha \))

- \( T.delete(\alpha) \); update breakpts
- Delete all circle events involving \( \alpha \) from \( Q \).
- Add Vor-vtx \( \alpha_{\text{left}} \cap \alpha_{\text{right}} \) and Vor-edge \( \langle \alpha_{\text{left}}, \alpha_{\text{right}} \rangle \) to DCEL.
- Check \( \langle \cdot, \alpha_{\text{left}}, \alpha_{\text{right}} \rangle \) and \( \langle \alpha_{\text{left}}, \alpha_{\text{right}}, \cdot \rangle \) for circle events.

Running time?
Handling Events

HandleSiteEvent(point p)

- Search in $\mathcal{T}$ for the arc $\alpha$ vertically above $p$. If $\alpha$ has pointer to circle event in $Q$, delete this event.
- Split $\alpha$ into $\alpha_0$ and $\alpha_2$. Let $\alpha_1$ be the new arc of $p$.
- Add Vor-edges $\langle q, p \rangle$ and $\langle p, q \rangle$ to DCEL.
- Check $\langle \cdot, \alpha_0, \alpha_1 \rangle$ and $\langle \alpha_1, \alpha_2, \cdot \rangle$ for circle events.

HandleCircleEvent(arc $\alpha$)

- $\mathcal{T}.\text{delete}(\alpha)$; update breakpts
- Delete all circle events involving $\alpha$ from $Q$.
- Add Vor-vtx $\alpha_{\text{left}} \cap \alpha_{\text{right}}$ and Vor-edge $\langle \alpha_{\text{left}}, \alpha_{\text{right}} \rangle$ to DCEL.
- Check $\langle \cdot, \alpha_{\text{left}}, \alpha_{\text{right}} \rangle$ and $\langle \alpha_{\text{left}}, \alpha_{\text{right}}, \cdot \rangle$ for circle events.

**Running time? $O(\log n)$ per event...**
VoronoiDiagram($P \subset \mathbb{R}^2$)

$Q \leftarrow \text{new PriorityQueue}(P)$  // site events sorted by $y$-coord.
$T \leftarrow \text{new BalancedBinarySearchTree}()$  // sweep status ($\beta$)
$D \leftarrow \text{new DCEL}()$  // to-be Vor($P$)

while not $Q$.empty() do
    $p \leftarrow Q$.ExtractMax()
    if $p$ site event then
        HandleSiteEvent($p$)
    else
        $\alpha \leftarrow \text{arc on } \beta \text{ that will disappear}$
        HandleCircleEvent($\alpha$)

    treat remaining int. nodes of $T$ (≡ unbnd. edges of Vor($P$))

return $D$
Running Time?

VoronoiDiagram($P \subset \mathbb{R}^2$)

$Q \leftarrow$ new PriorityQueue($P$)  // site events sorted by $y$-coord.
$T \leftarrow$ new BalancedBinarySearchTree()  // sweep status ($\beta$)
$D \leftarrow$ new DCEL()  // to-be Vor($P$)

while not $Q$.empty() do
    $p \leftarrow Q$.ExtractMax()
    if $p$ site event then
        HandleSiteEvent($p$)  // exactly $n$ such events
    else
        $\alpha \leftarrow$ arc on $\beta$ that will disappear
        HandleCircleEvent($\alpha$)

    treat remaining int. nodes of $T$ (≡ unbnd. edges of Vor($P$))

return $D$
Running Time?

VoronoiDiagram(\(P \subset \mathbb{R}^2\))

\[
\begin{align*}
Q & \leftarrow \text{new PriorityQueue}(P) & \quad & \text{// site events sorted by } y\text{-coord.} \\
\mathcal{T} & \leftarrow \text{new BalancedBinarySearchTree}() & \quad & \text{// sweep status } (\beta) \\
\mathcal{D} & \leftarrow \text{new DCEL}() & \quad & \text{// to-be Vor}(P) \\
\text{while not } Q.\text{empty}() \text{ do} \\
\quad p & \leftarrow Q.\text{ExtractMax}() \\
\quad \text{if } p \text{ site event then} \\
\quad \quad & \text{HandleSiteEvent}(p) & \quad & \text{exactly } n \text{ such events} \\
\quad \text{else} \\
\quad \quad \alpha & \leftarrow \text{arc on } \beta \text{ that will disappear} \\
\quad \quad & \text{HandleCircleEvent}(\alpha) & \quad & \text{at most } 2n - 5 \text{ such events} \\
\text{treat remaining int. nodes of } \mathcal{T} \left(\equiv \text{unbnd. edges of } \text{Vor}(P)) \\
\text{return } \mathcal{D}
\end{align*}
\]
Theorem. Given a set $P$ of $n$ pts in the plane, Fortune’s sweep computes $\text{Vor}(P)$ in $O(n \log n)$ time and $O(n)$ space.
Summary

**Theorem.** Given a set $P$ of $n$ pts in the plane, Fortune’s sweep computes $\text{Vor}(P)$ in $O(n \log n)$ time and $O(n)$ space.

Steven Fortune
Bell Labs