Computational Geometry
Chapter 4: Voronoi Diagrams

Prof. Dr. Sándor Fekete

Algorithms Division
Department of Computer Science
TU Braunschweig

Technische Universität Braunschweig

1. Introduction and Motivation
2. Definitions
3. Representing planar partitions
4. Properties
5. Fortune's algorithm
6. Variations
7. The Voronoi game
8. Summary and conclusions

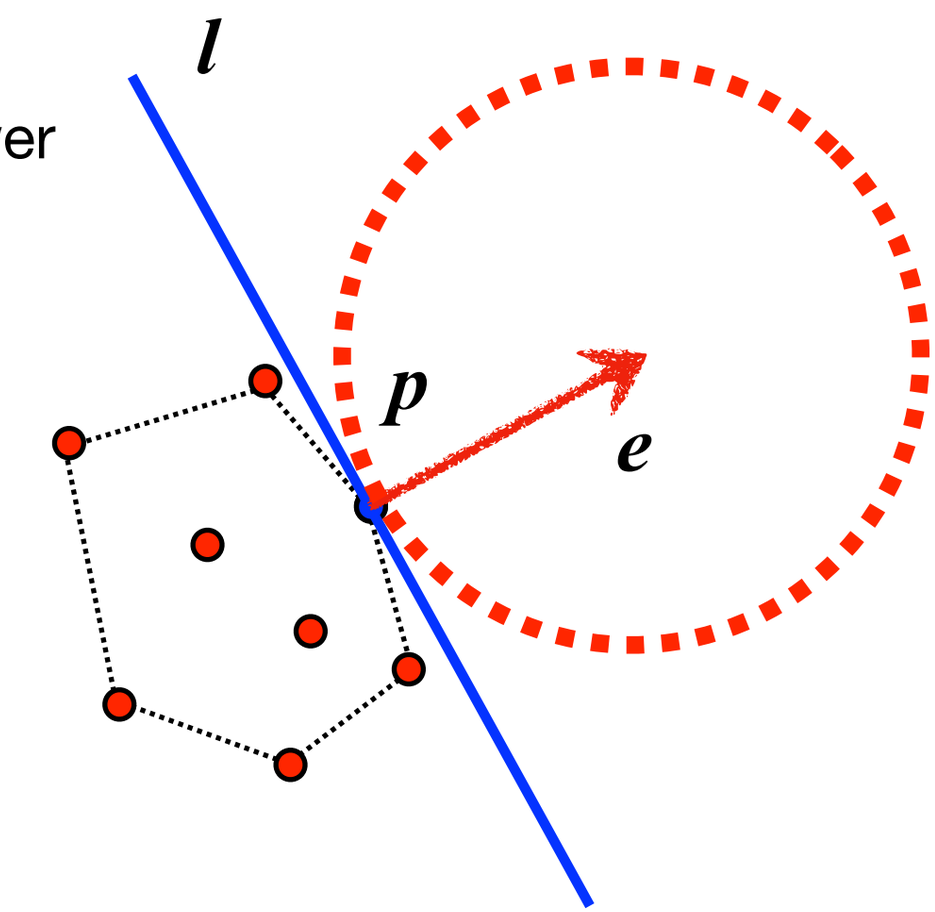
1. Introduction and Motivation
2. Definitions
3. Representing planar partitions
4. Properties
5. Fortune's algorithm
6. Variations
7. The Voronoi game
8. Summary and conclusions

Lemma 4.13

$p \in \mathcal{P}$ lies on boundary of $\mathrm{conv}(\mathcal{P}) \Leftrightarrow V(p)$ unbounded.

Corollary 4.14:

Computing the Voronoi diagram for n points has a lower bound of $\Omega(n \log n)$.



Vironoi Man

Approach:

- Consider a moving „frontier“ between resolved and unresolved part.

Crucial issue:

- $p \in \mathcal{P}$ below ℓ can influence $Vor(p)$ above ℓ.

Observation:

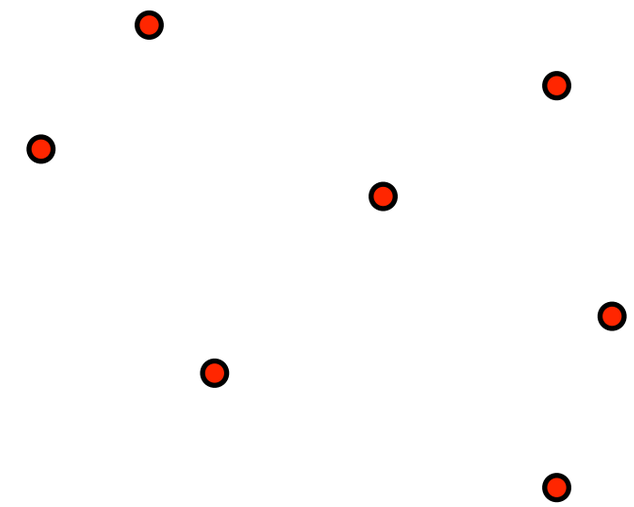
- The separation between resolved and unresolved part for a point p and line ℓ is a curve consisting of points that have equal distance from p and ℓ.

Approach:

- Consider a moving „frontier“ between resolved and unresolved part.

Crucial issue:

- $p \in \mathcal{P}$ below ℓ can influence $Vor(p)$ above ℓ.

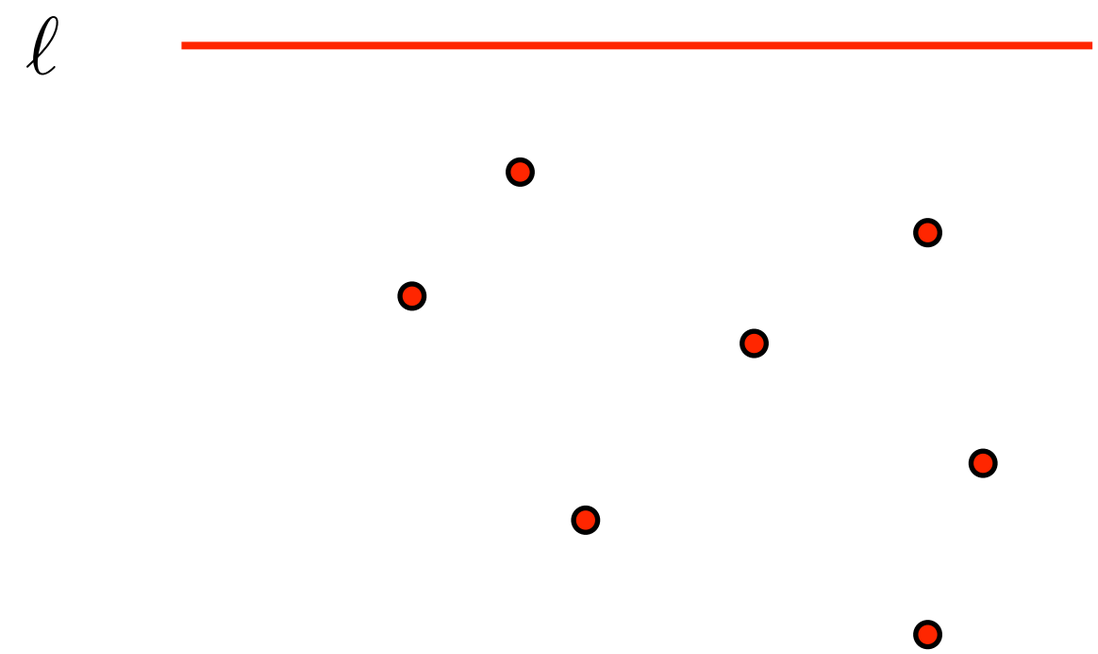


Observation:

- The separation between resolved and unresolved part for a point p and line ℓ is a curve consisting of points that have equal distance from p and ℓ.

Approach:

- Consider a moving „frontier“ between resolved and unresolved part.



Crucial issue:

- $p \in \mathcal{P}$ below ℓ can influence $Vor(p)$ above ℓ.

Observation:

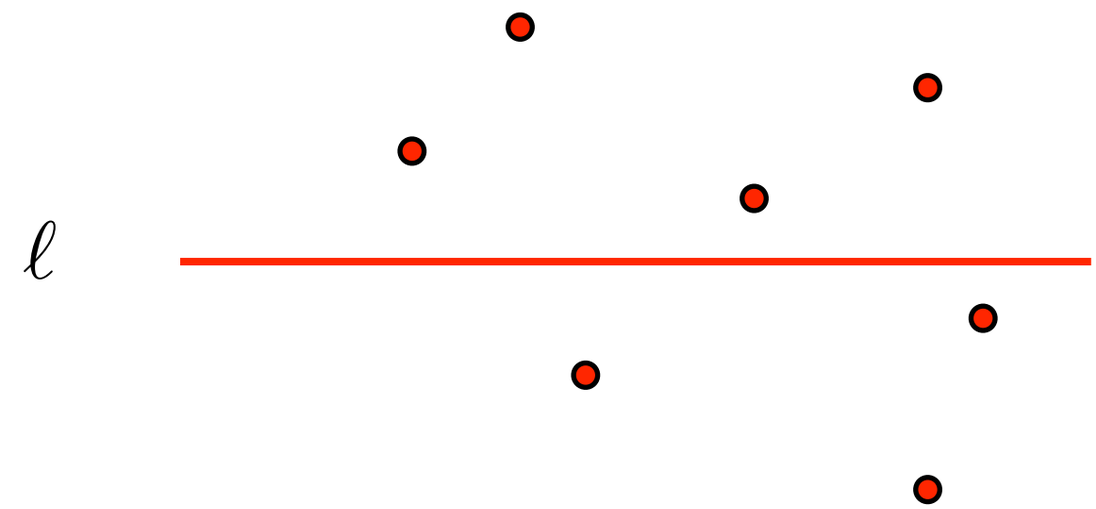
- The separation between resolved and unresolved part for a point p and line ℓ is a curve consisting of points that have equal distance from p and ℓ.

Approach:

- Consider a moving „frontier“ between resolved and unresolved part.

Crucial issue:

- $p \in \mathcal{P}$ below ℓ can influence $Vor(p)$ above ℓ.



Observation:

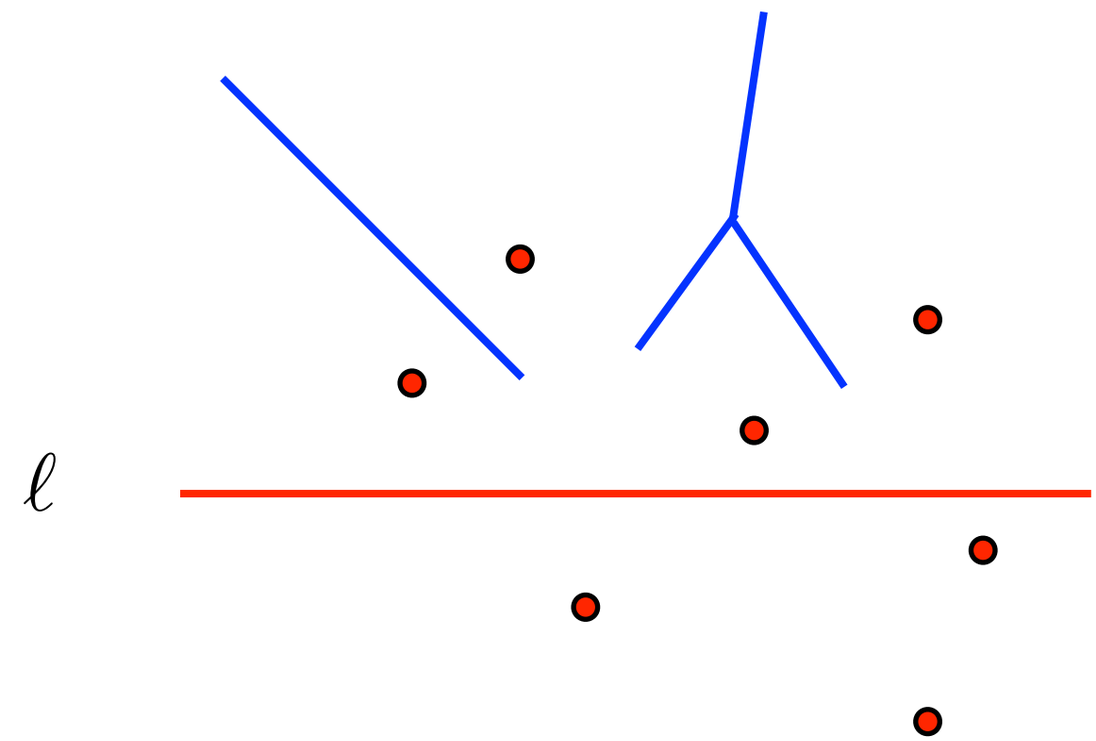
- The separation between resolved and unresolved part for a point p and line ℓ is a curve consisting of points that have equal distance from p and ℓ.

Approach:

- Consider a moving „frontier“ between resolved and unresolved part.

Crucial issue:

- $p \in \mathcal{P}$ below ℓ can influence $Vor(p)$ above ℓ.



Observation:

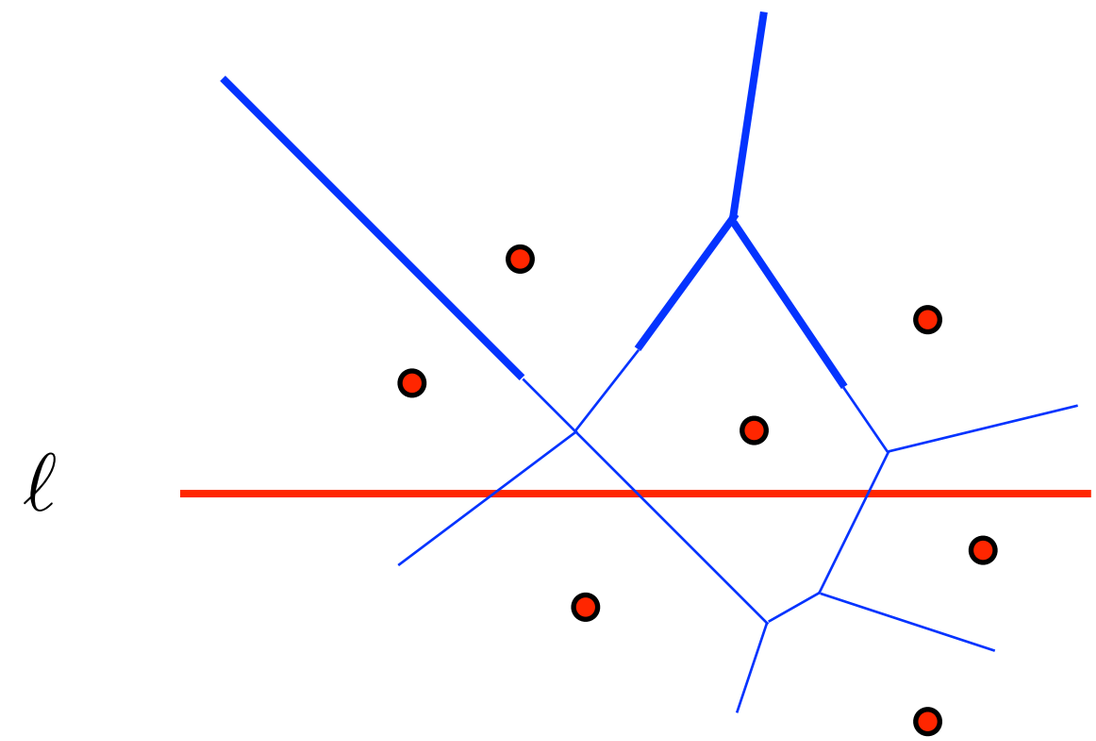
- The separation between resolved and unresolved part for a point p and line ℓ is a curve consisting of points that have equal distance from p and ℓ.

Approach:

- Consider a moving „frontier“ between resolved and unresolved part.

Crucial issue:

- $p \in \mathcal{P}$ below ℓ can influence $Vor(p)$ above ℓ.



Observation:

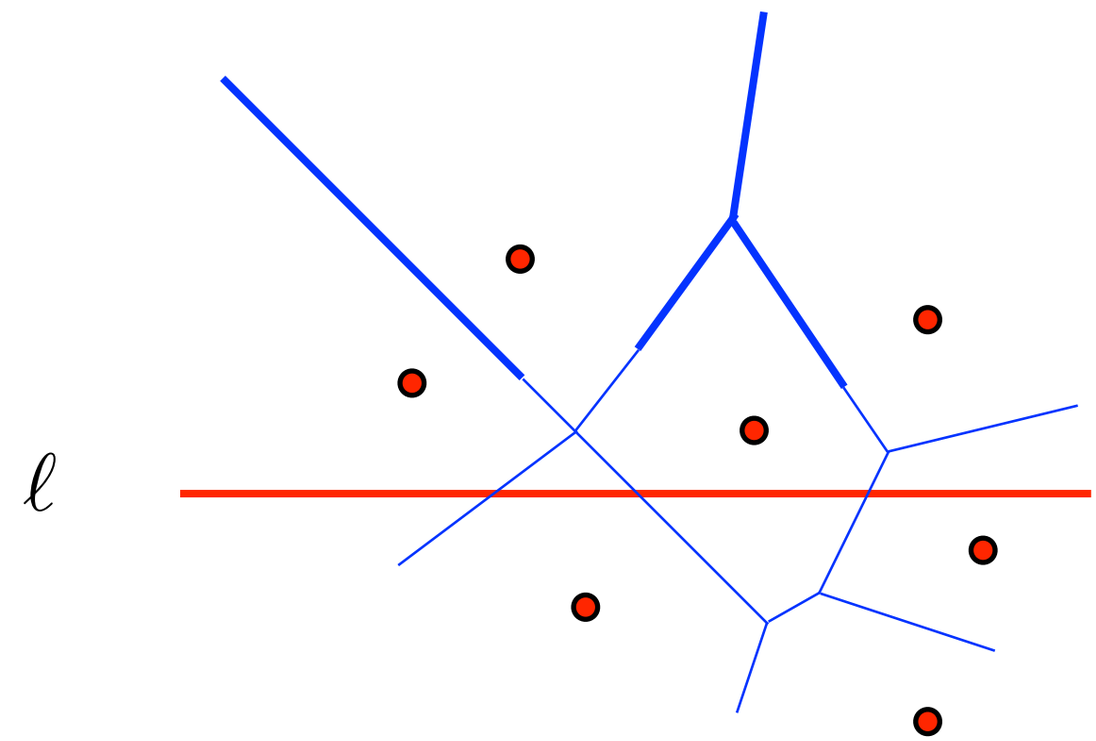
- The separation between resolved and unresolved part for a point p and line ℓ is a curve consisting of points that have equal distance from p and ℓ.

Approach:

- Consider a moving „frontier“ between resolved and unresolved part.

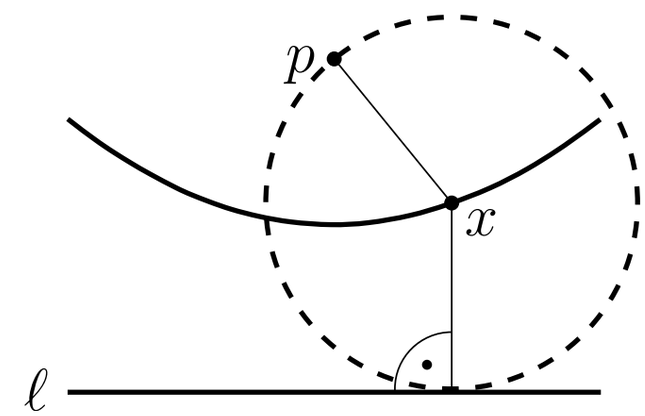
Crucial issue:

- $p \in \mathcal{P}$ below ℓ can influence $Vor(p)$ above ℓ.



Observation:

- The separation between resolved and unresolved part for a point p and line ℓ is a curve consisting of points that have equal distance from p and ℓ.

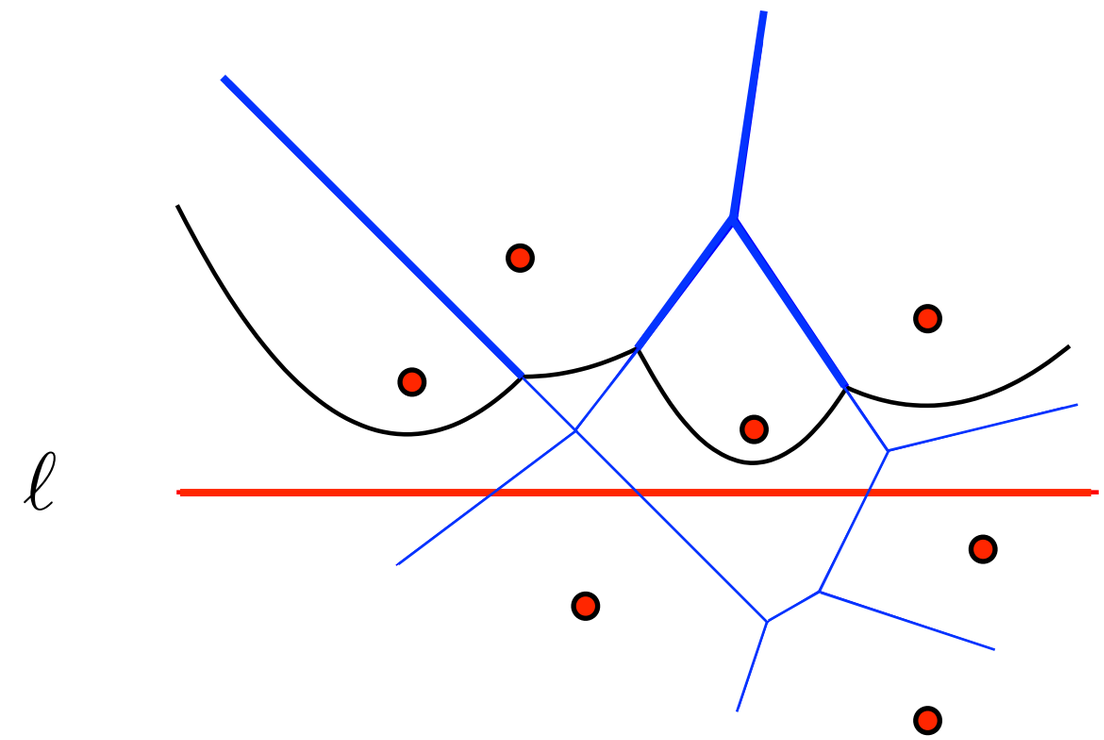


Approach:

- Consider a moving „frontier“ between resolved and unresolved part.

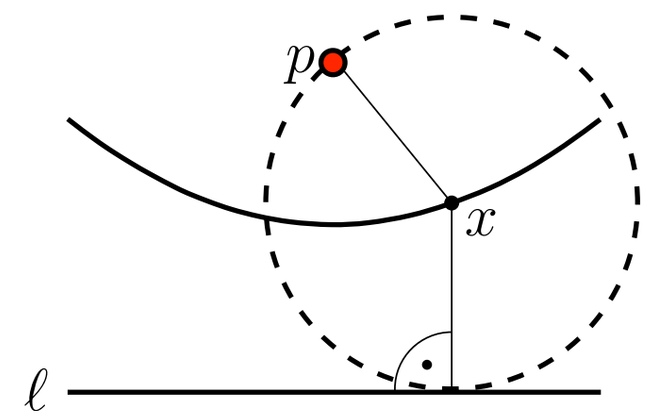
Crucial issue:

- $p \in \mathcal{P}$ below ℓ can influence $Vor(p)$ above ℓ.



Observation:

- The separation between resolved and unresolved part for a point p and line ℓ is a curve consisting of points that have equal distance from p and ℓ.

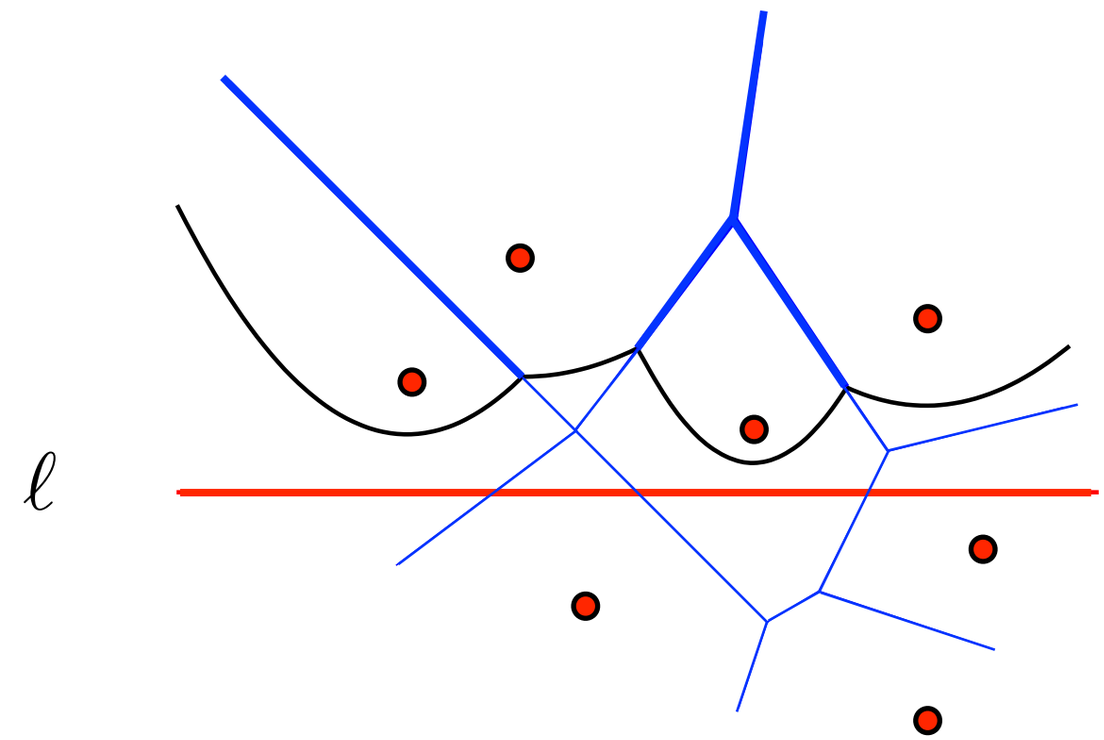


Approach:

- Consider a moving „frontier“ between resolved and unresolved part.

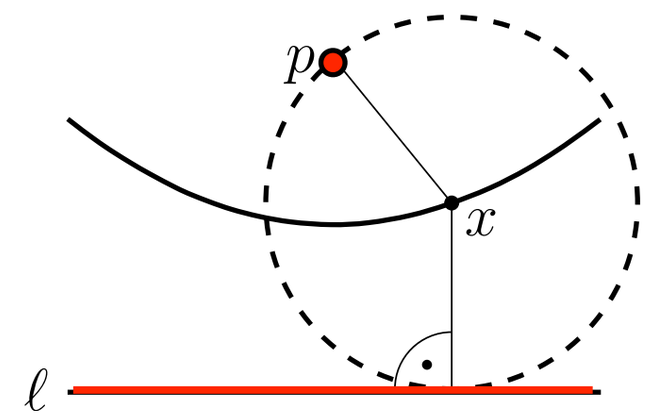
Crucial issue:

- $p \in \mathcal{P}$ below ℓ can influence $Vor(p)$ above ℓ.



Observation:

- The separation between resolved and unresolved part for a point p and line ℓ is a curve consisting of points that have equal distance from p and ℓ.

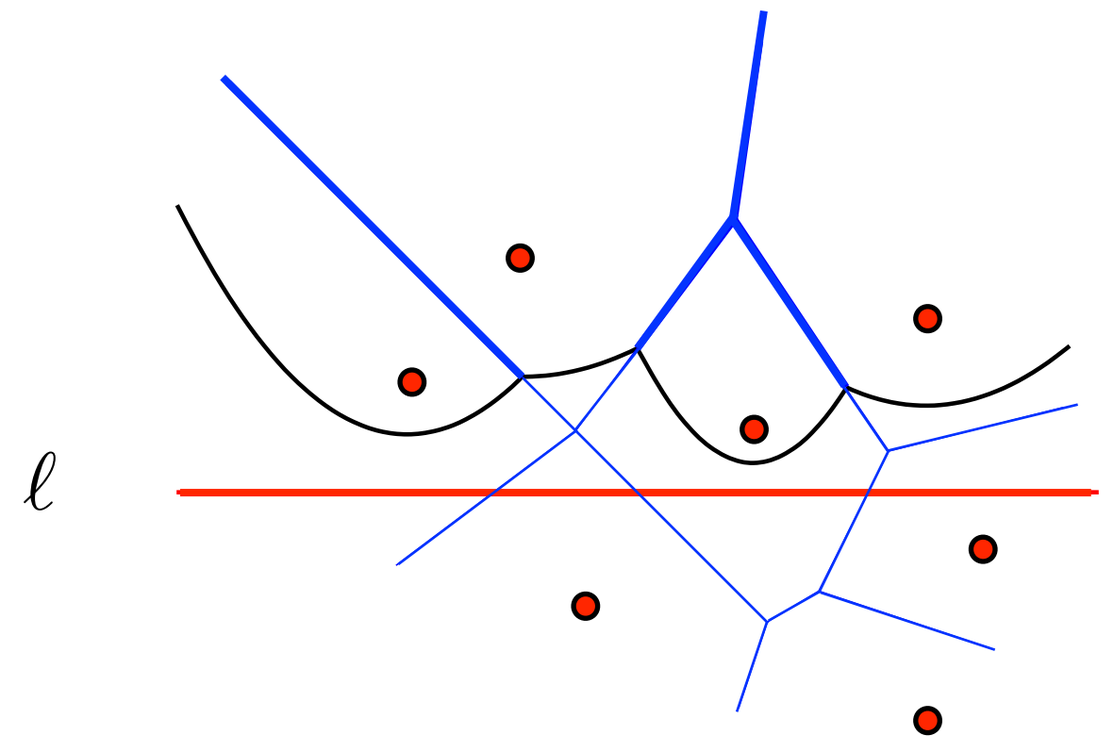


Approach:

- Consider a moving „frontier“ between resolved and unresolved part.

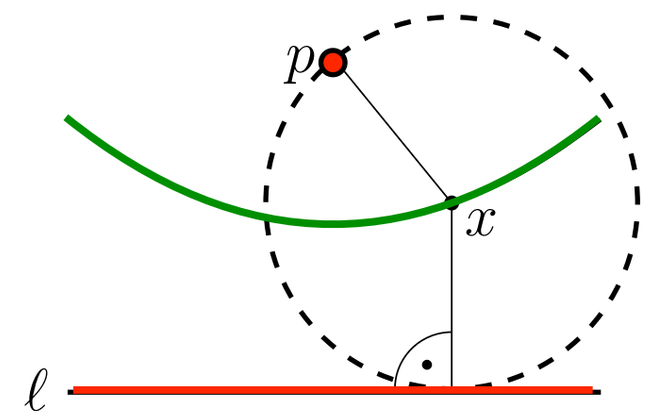
Crucial issue:

- $p \in \mathcal{P}$ below ℓ can influence $Vor(p)$ above ℓ.



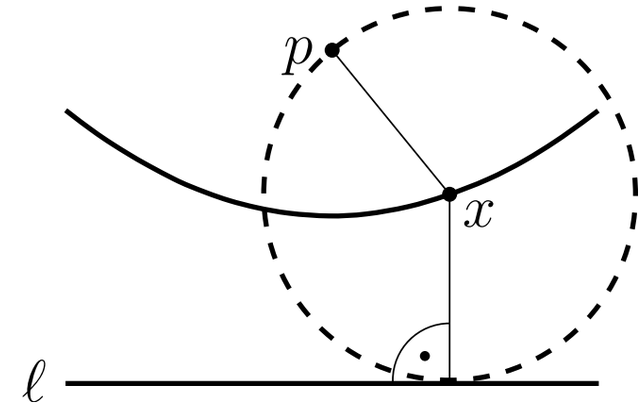
Observation:

- The separation between resolved and unresolved part for a point p and line ℓ is a curve consisting of points that have equal distance from p and ℓ.



Consider:

$$\{x \in \mathbb{R}^2 \mid d(x,p) = d(x,\ell)\}$$



Theorem 4.15:

The curve is a parabola (with *focus* p and *directrix* ℓ).

Proof:

Consider $p=(0,s)$ and $X=(x,0)$.
Then $C=(x,y)$ with

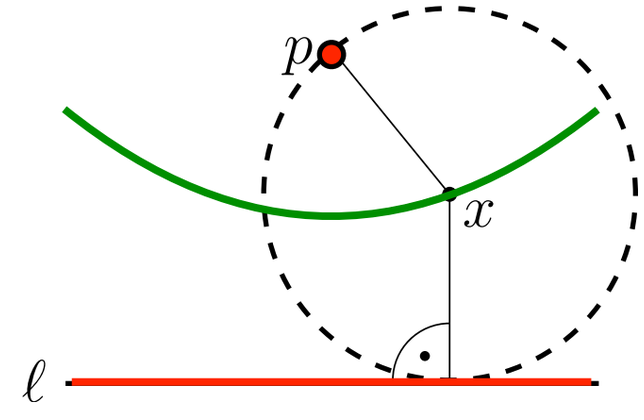
$$d_1^2 = x^2 + (y-s)^2$$
$$d_2^2 = y^2$$

So

$$y = \frac{1}{2s}x^2 + \frac{s}{2}$$

Consider:

$$\{x \in \mathbb{R}^2 \mid d(x,p) = d(x,\ell)\}$$



Theorem 4.15:

The curve is a parabola (with *focus* p and *directrix* ℓ).

Proof:

Consider $p=(0,s)$ and $X=(x,0)$.
Then $C=(x,y)$ with

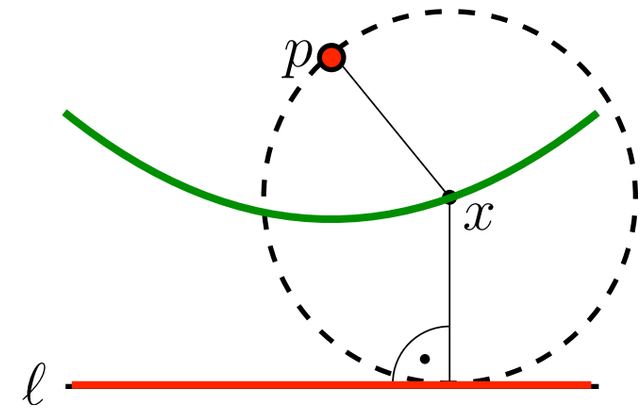
$$d_1^2 = x^2 + (y-s)^2$$
$$d_2^2 = y^2$$

So

$$y = \frac{1}{2s}x^2 + \frac{s}{2}$$

Consider:

$$\{x \in \mathbb{R}^2 \mid d(x,p) = d(x,\ell)\}$$



Theorem 4.15:

The curve is a parabola (with *focus* p and *directrix* ℓ).

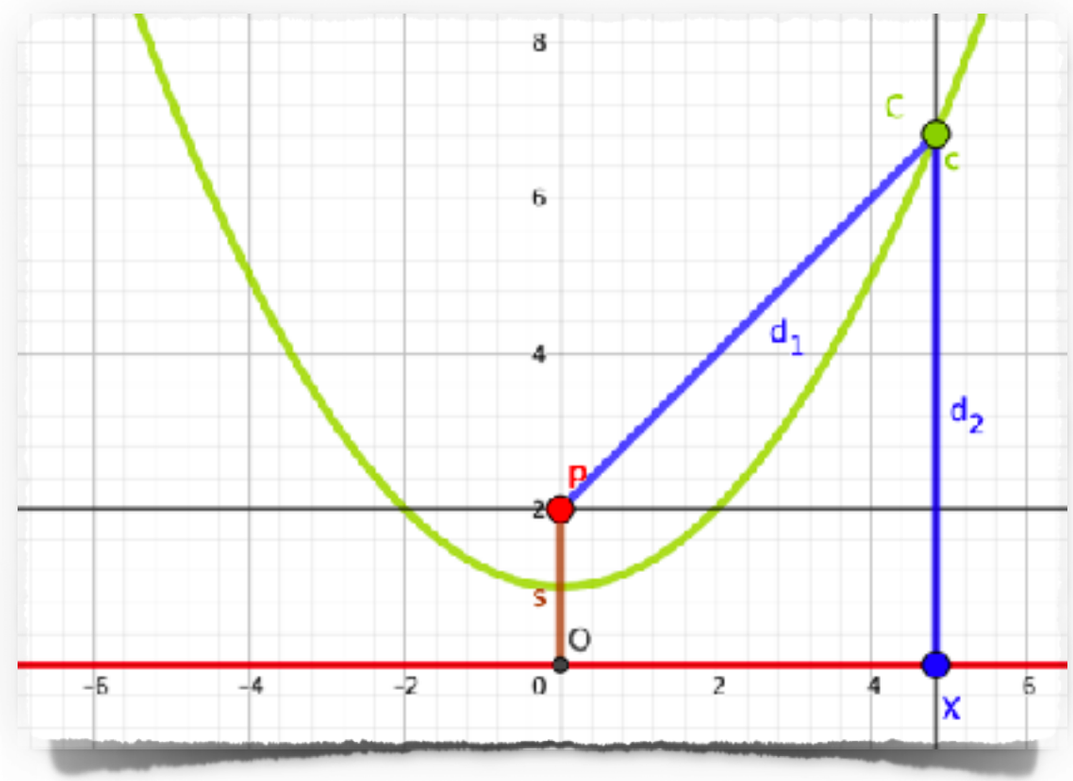
Proof:

Consider $p=(0,s)$ and $X=(x,0)$.
Then $C=(x,y)$ with

$$d_1^2 = x^2 + (y-s)^2$$
$$d_2^2 = y^2$$

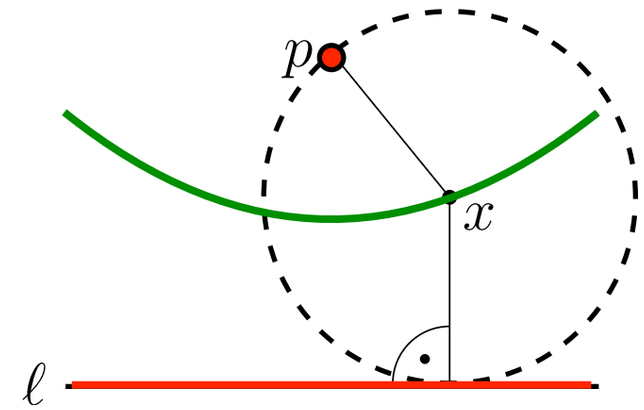
So

$$y = \frac{1}{2s}x^2 + \frac{s}{2}$$



Consider:

$$\{x \in \mathbb{R}^2 \mid d(x,p) = d(x,\ell)\}$$



Theorem 4.15:

The curve is a parabola (with *focus* p and *directrix* ℓ).

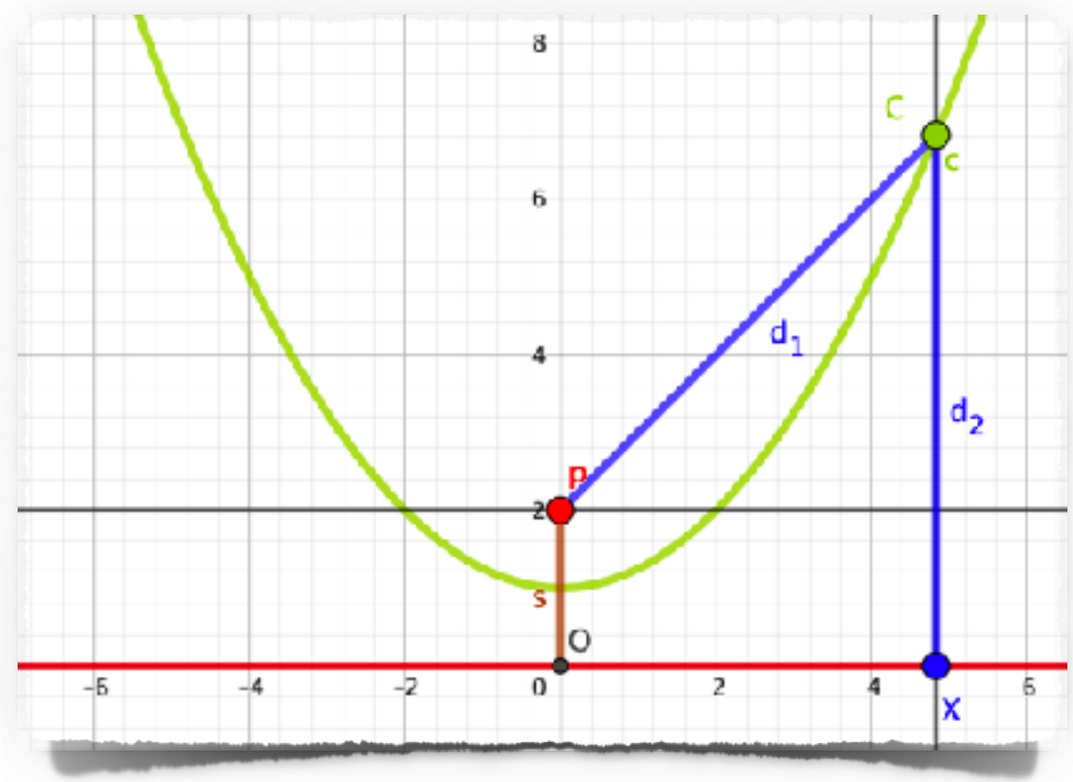
Proof:

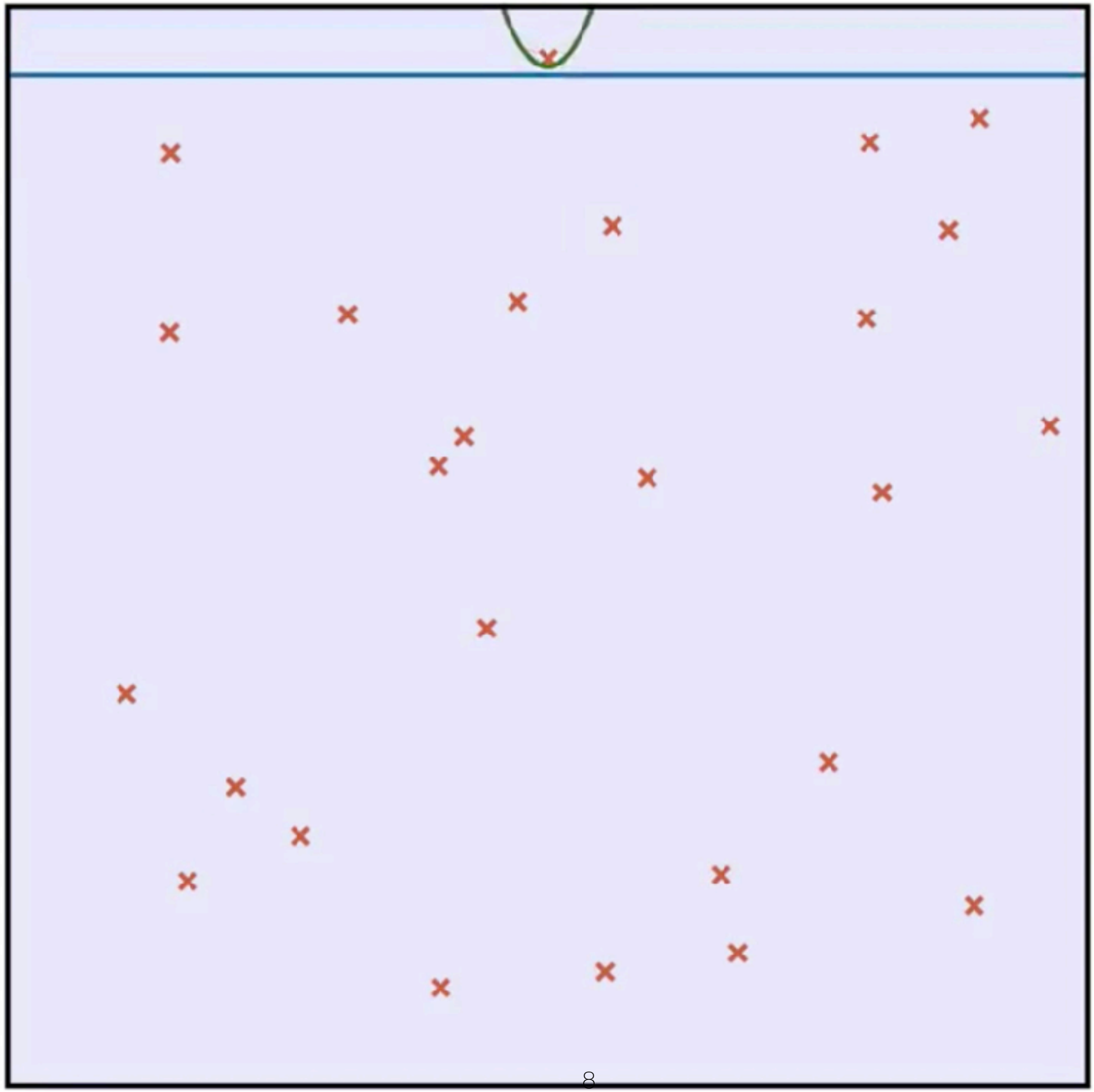
Consider $p=(0,s)$ and $X=(x,0)$.
Then $C=(x,y)$ with

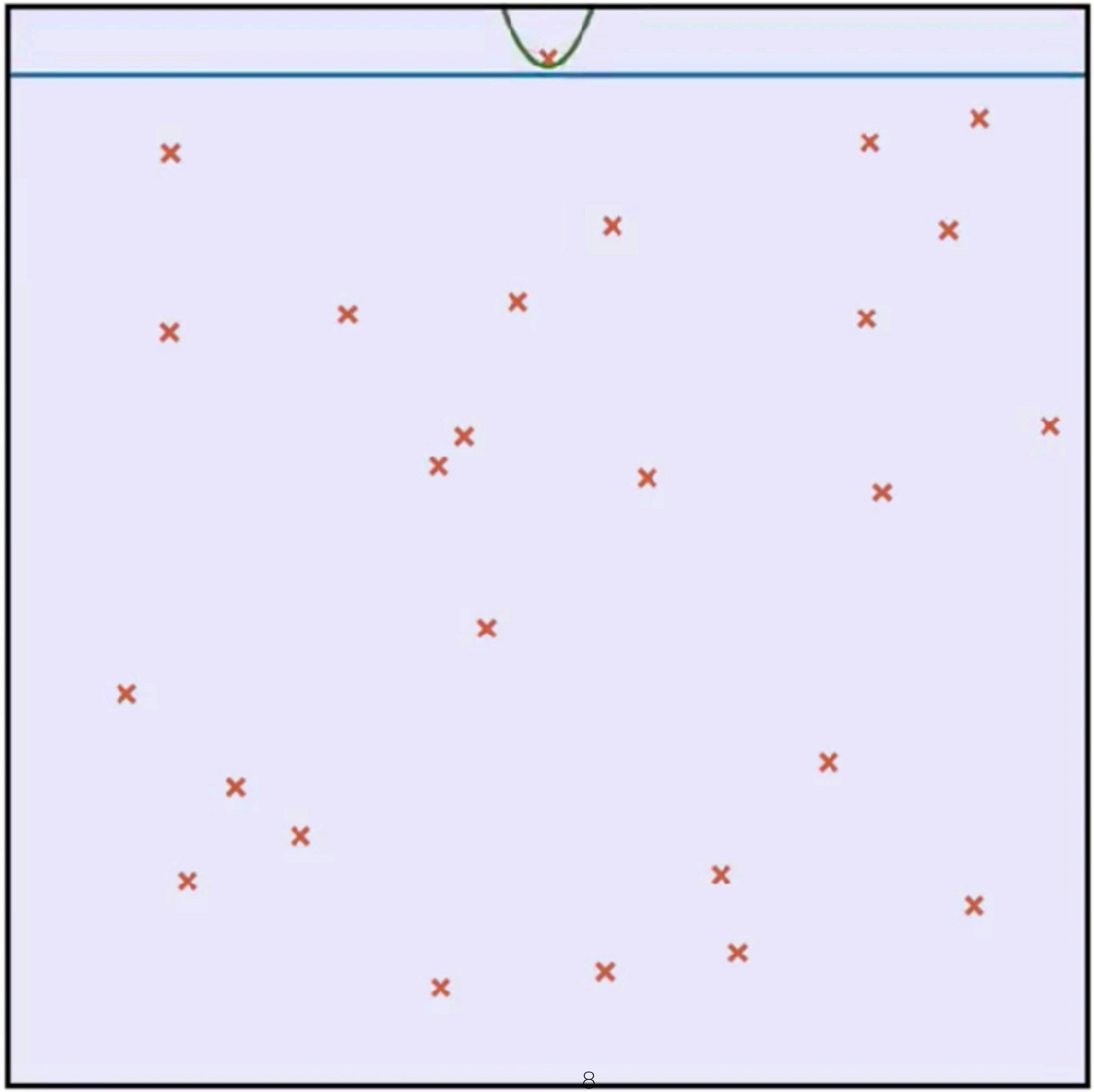
$$d_1^2 = x^2 + (y-s)^2 = x^2 + y^2 - 2ys + s^2$$
$$d_2^2 = y^2$$

So

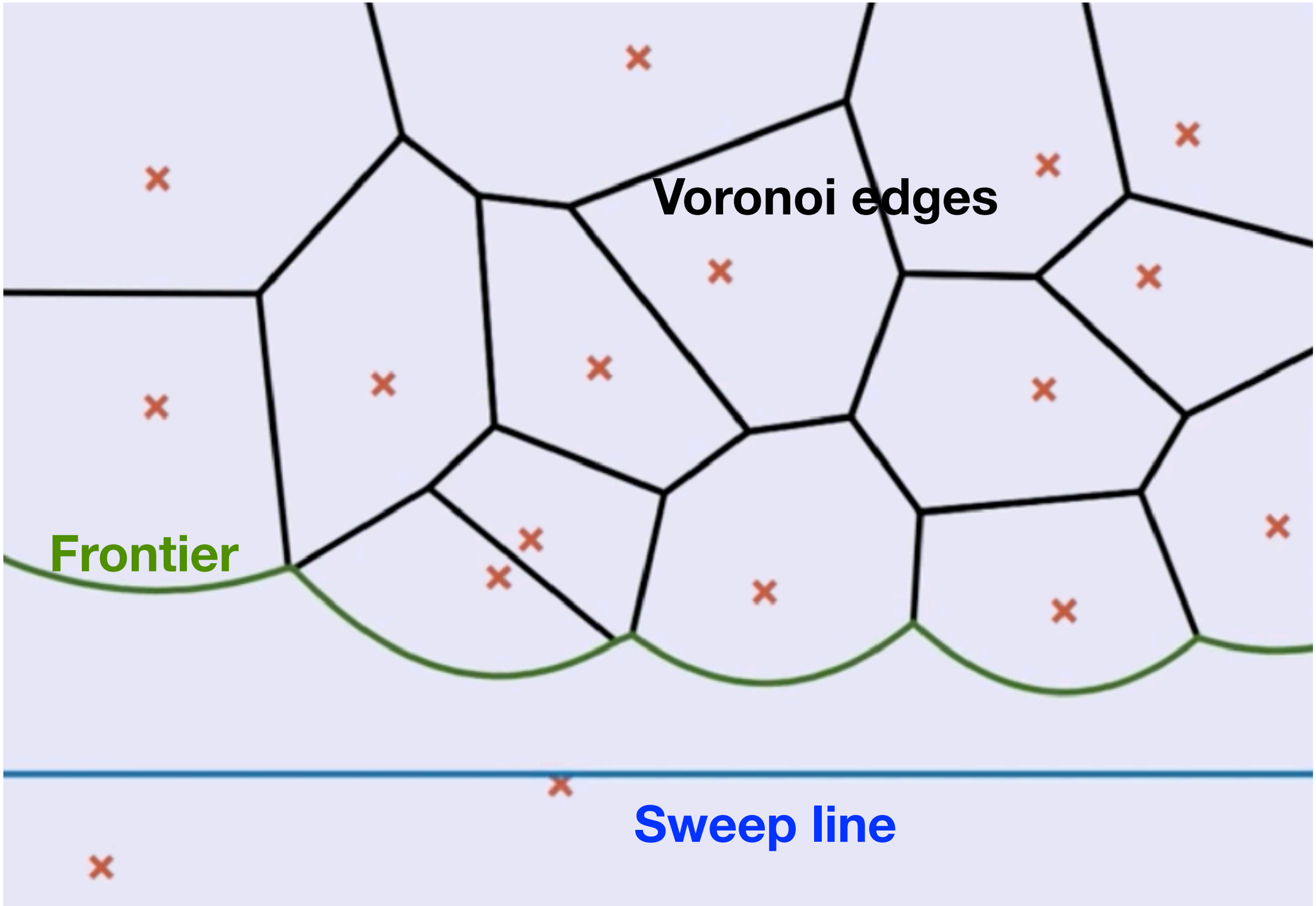
$$y = \frac{1}{2s}x^2 + \frac{s}{2}$$





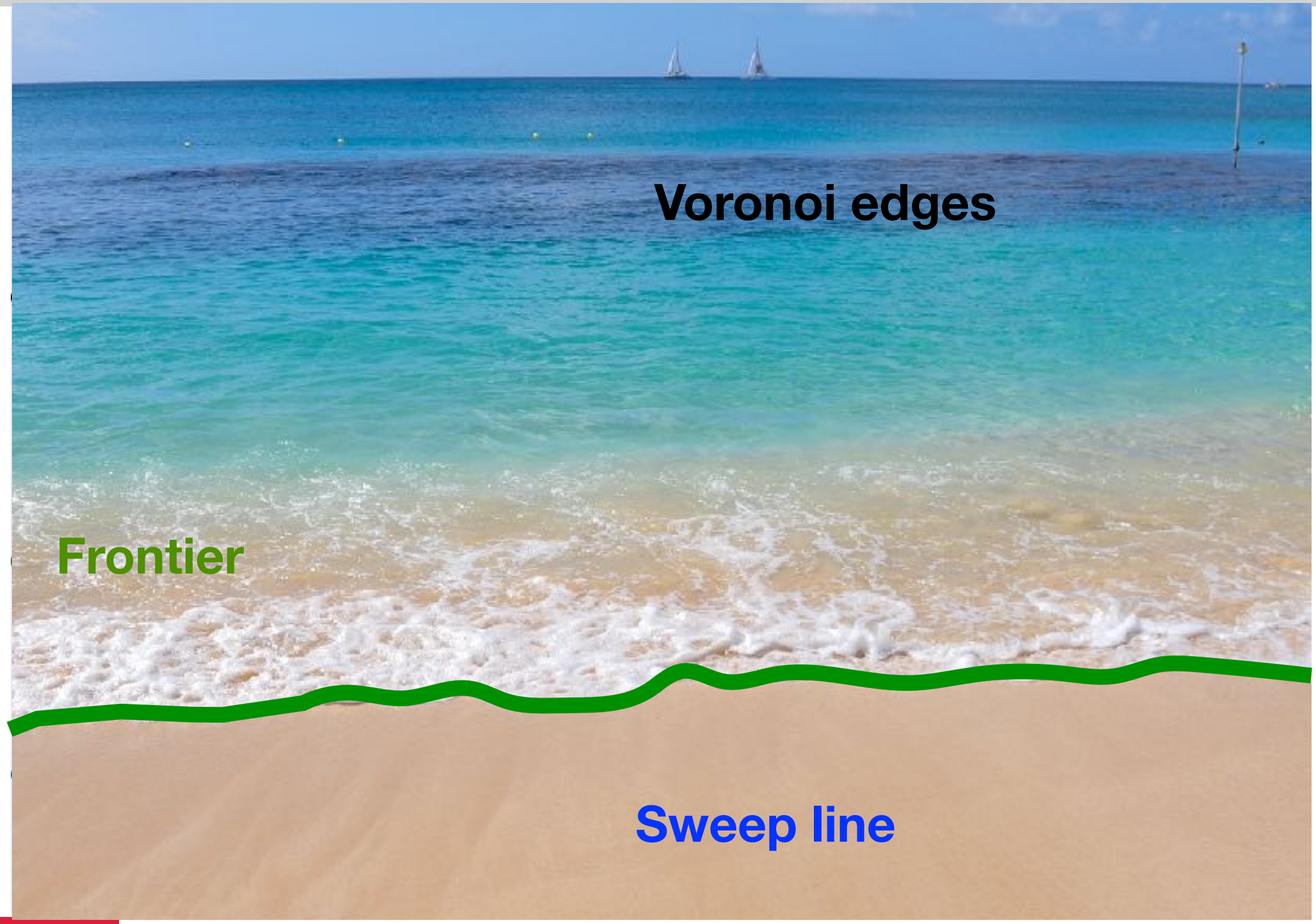


The Beach Line



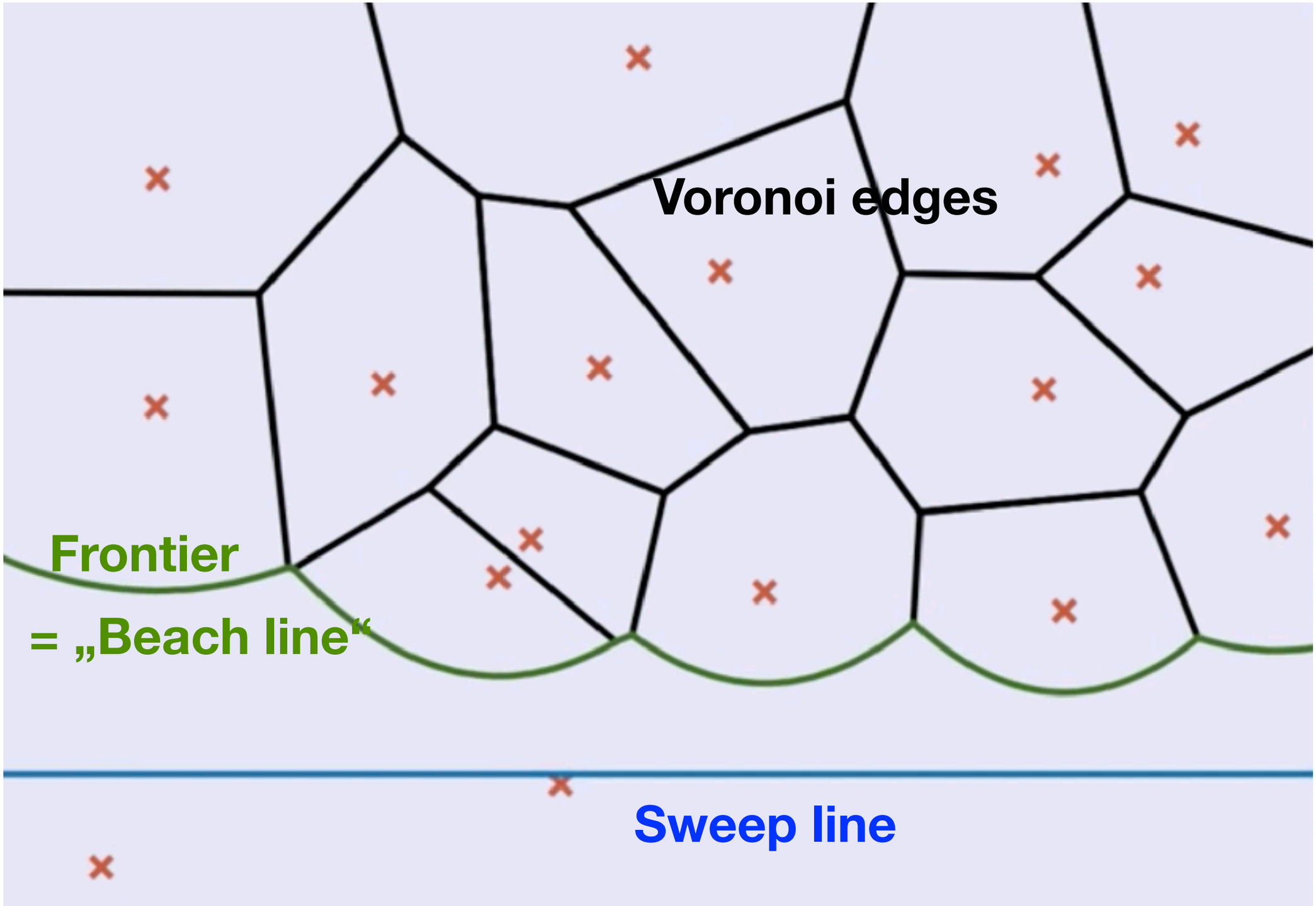
The Beach Line

The Beach Line



The Beach Line

The Beach Line



Issues:

Issues:

- How do sites and parabolas interact?

Issues:

- How do sites and parabolas interact?

- How do we describe the whole beach line?

Issues:

- How do sites and parabolas interact?
- How do we describe the whole beach line?
- How do we turn the continuous *process* into a *discrete one*?

Issues:

- How do sites and parabolas interact?
- How do we describe the whole beach line?
- How do we turn the continuous *process* into a *discrete one*?
- How can we capture discrete transitions in the continuous process?

Issues:

- How do sites and parabolas interact?
- How do we describe the whole beach line?
- How do we turn the continuous *process* into a *discrete one*?
- How can we capture discrete transitions in the continuous process?
- How do we organize the overall algorithm?

Issues:

- How do sites and parabolas interact?
- How do we describe the whole beach line?
- How do we turn the continuous *process* into a *discrete one*?
- How can we capture discrete transitions in the continuous process?
- How do we organize the overall algorithm?
- How do we guarantee correctness?

Issues:

- How do sites and parabolas interact?
- How do we describe the whole beach line?
- How do we turn the continuous *process* into a *discrete one*?
- How can we capture discrete transitions in the continuous process?
- How do we organize the overall algorithm?
- How do we guarantee correctness?
- How do we get good runtime?

Intuition:

- Let $x \in \mathbb{R}^2$ be above ℓ and $p \in \mathcal{P}$ below ℓ
  $\Rightarrow d(x,p) \geq d(x,\ell)$
- Let $q \in \mathcal{P}$ be nearest site for x.
  If $d(x,q) \leq d(x,\ell)$, then q not below ℓ.
- $\{x \in \mathbb{R}^2 \mid d(x,q) \leq d(x,\ell)\}$
  is bounded by parabola.

Consequence:

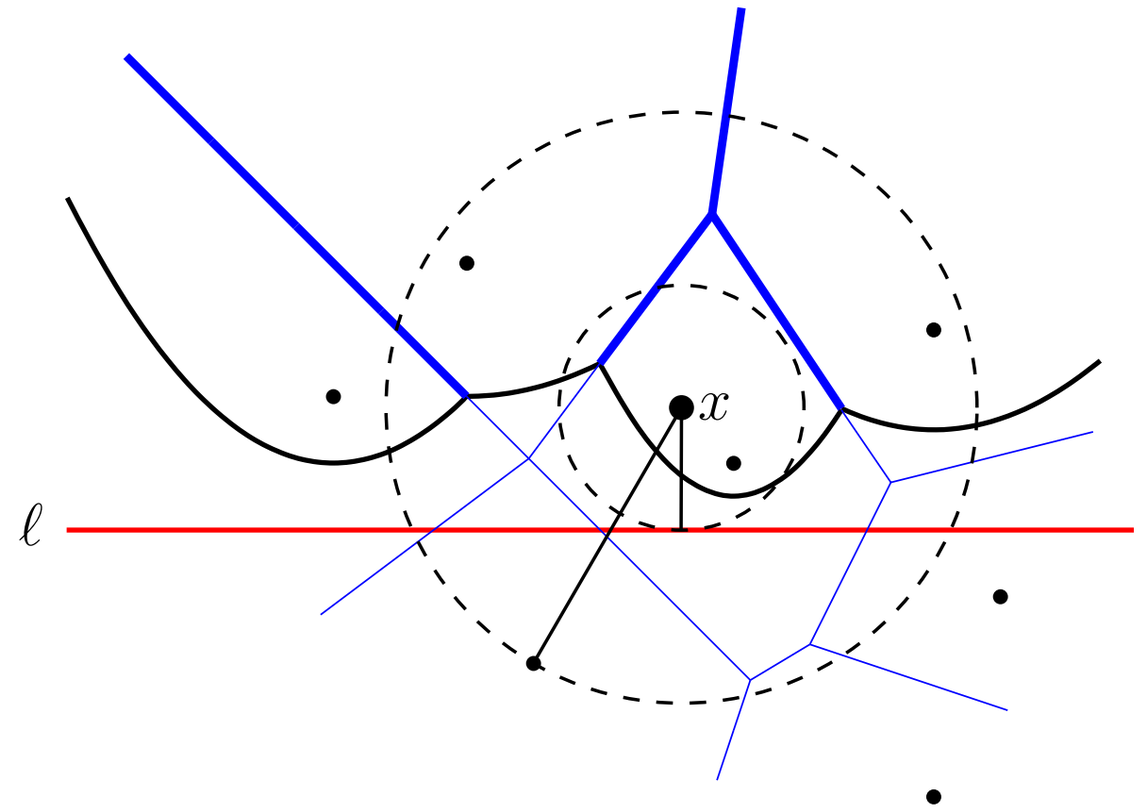
- $\exists$ parabola β: x above β
  $\Rightarrow$ nearest neighbor q not below ℓ.

Beach line:

- $p_1, \ldots, p_k$ above ℓ $\rightarrow$ parabolas $\beta_1, \ldots, \beta_k$
- Beach line: $f : \mathbb{R} \rightarrow \mathbb{R}^2$ with $f(x) :=$
  point on $\beta_1 \cup \cdots \cup \beta_k$ with min. y-coord.
- By construction monotonic

Intuition:

- Let $x \in \mathbb{R}^2$ be above ℓ and $p \in \mathcal{P}$ below ℓ
  $\Rightarrow d(x,p) \geq d(x,\ell)$
- Let $q \in \mathcal{P}$ be nearest site for x.
  If $d(x,q) \leq d(x,\ell)$, then q not below ℓ.
- $\{x \in \mathbb{R}^2 \mid d(x,q) \leq d(x,\ell)\}$
  is bounded by parabola.



Consequence:

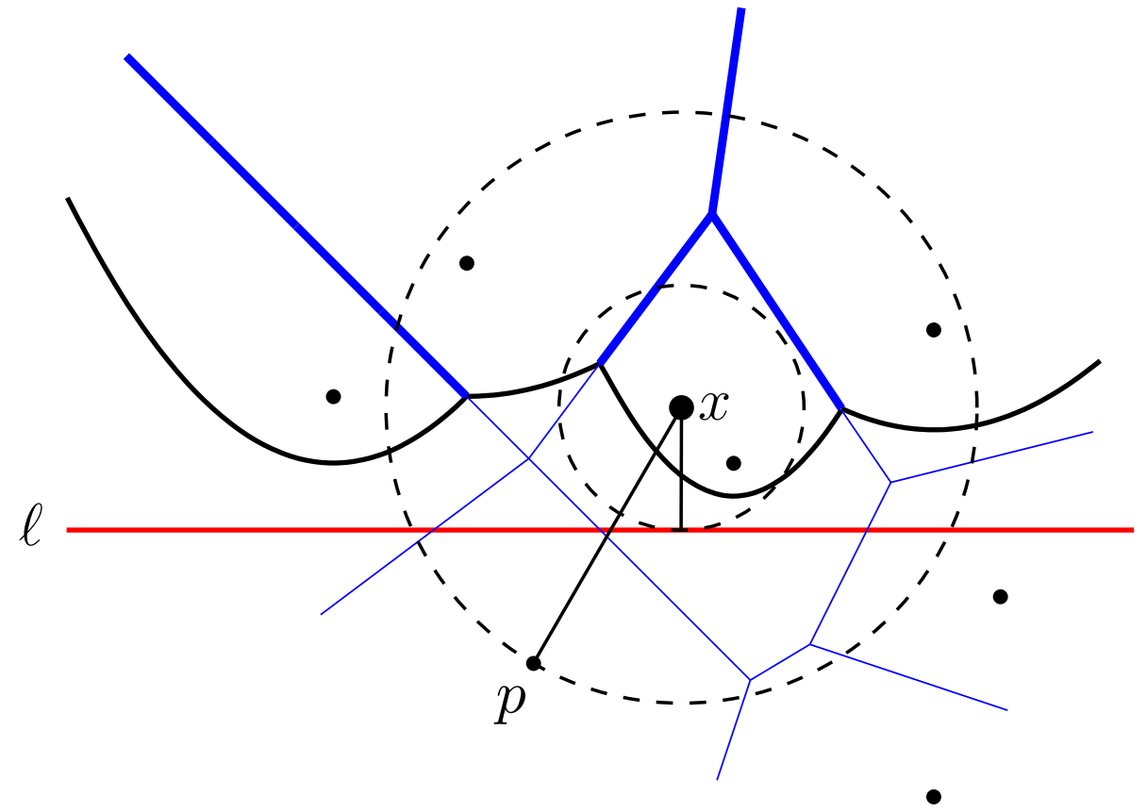
- $\exists$ parabola β: x above β
  $\Rightarrow$ nearest neighbor q not below ℓ.

Beach line:

- $p_1, \ldots, p_k$ above ℓ $\rightarrow$ parabolas $\beta_1, \ldots, \beta_k$
- Beach line: $f : \mathbb{R} \rightarrow \mathbb{R}^2$ with $f(x) :=$
  point on $\beta_1 \cup \cdots \cup \beta_k$ with min. y-coord.
- By construction monotonic

Intuition:

- Let $x \in \mathbb{R}^2$ be above ℓ and $p \in \mathcal{P}$ below ℓ
  $\Rightarrow d(x,p) \geq d(x,\ell)$
- Let $q \in \mathcal{P}$ be nearest site for x.
  If $d(x,q) \leq d(x,\ell)$, then q not below ℓ.
- $\{x \in \mathbb{R}^2 \mid d(x,q) \leq d(x,\ell)\}$
  is bounded by parabola.



Consequence:

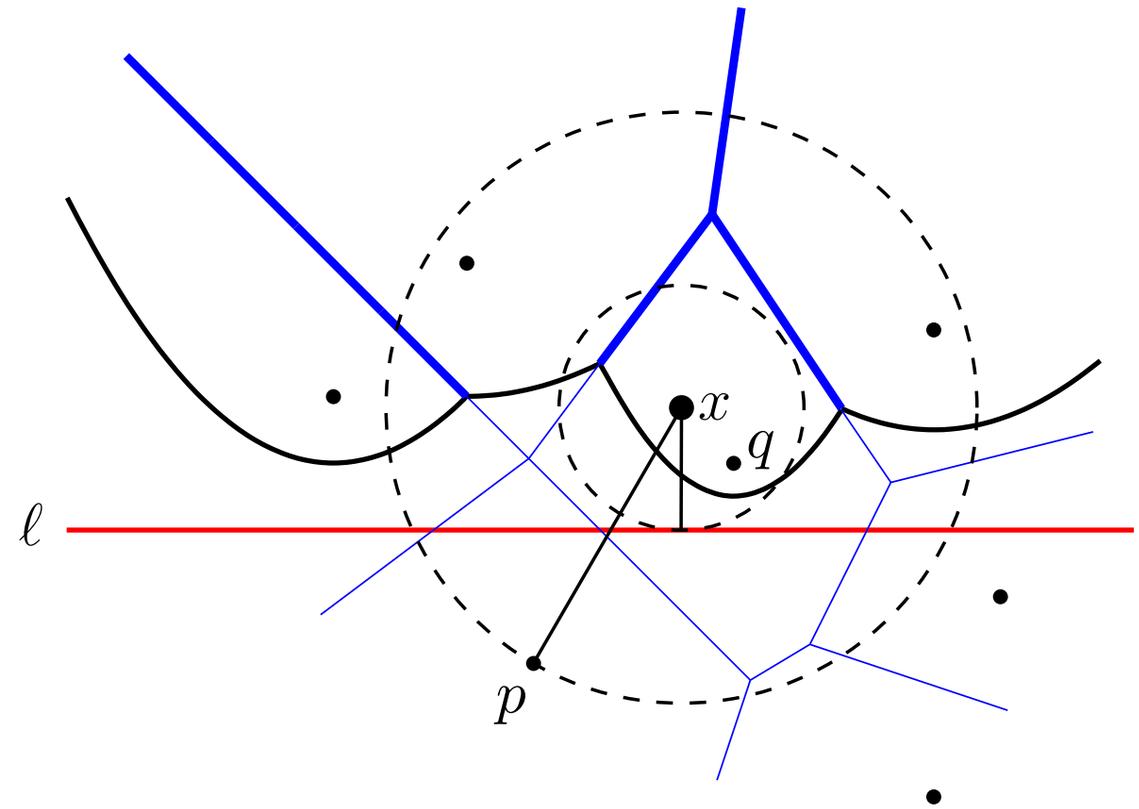
- $\exists$ parabola β: x above β
  $\Rightarrow$ nearest neighbor q not below ℓ.

Beach line:

- $p_1, \ldots, p_k$ above ℓ $\rightarrow$ parabolas $\beta_1, \ldots, \beta_k$
- Beach line: $f : \mathbb{R} \rightarrow \mathbb{R}^2$ with $f(x) :=$
  point on $\beta_1 \cup \cdots \cup \beta_k$ with min. y-coord.
- By construction monotonic

Intuition:

- Let $x \in \mathbb{R}^2$ be above ℓ and $p \in \mathcal{P}$ below ℓ
  $\Rightarrow d(x,p) \geq d(x,\ell)$
- Let $q \in \mathcal{P}$ be nearest site for x.
  If $d(x,q) \leq d(x,\ell)$, then q not below ℓ.
- $\{x \in \mathbb{R}^2 \mid d(x,q) \leq d(x,\ell)\}$
  is bounded by parabola.



Consequence:

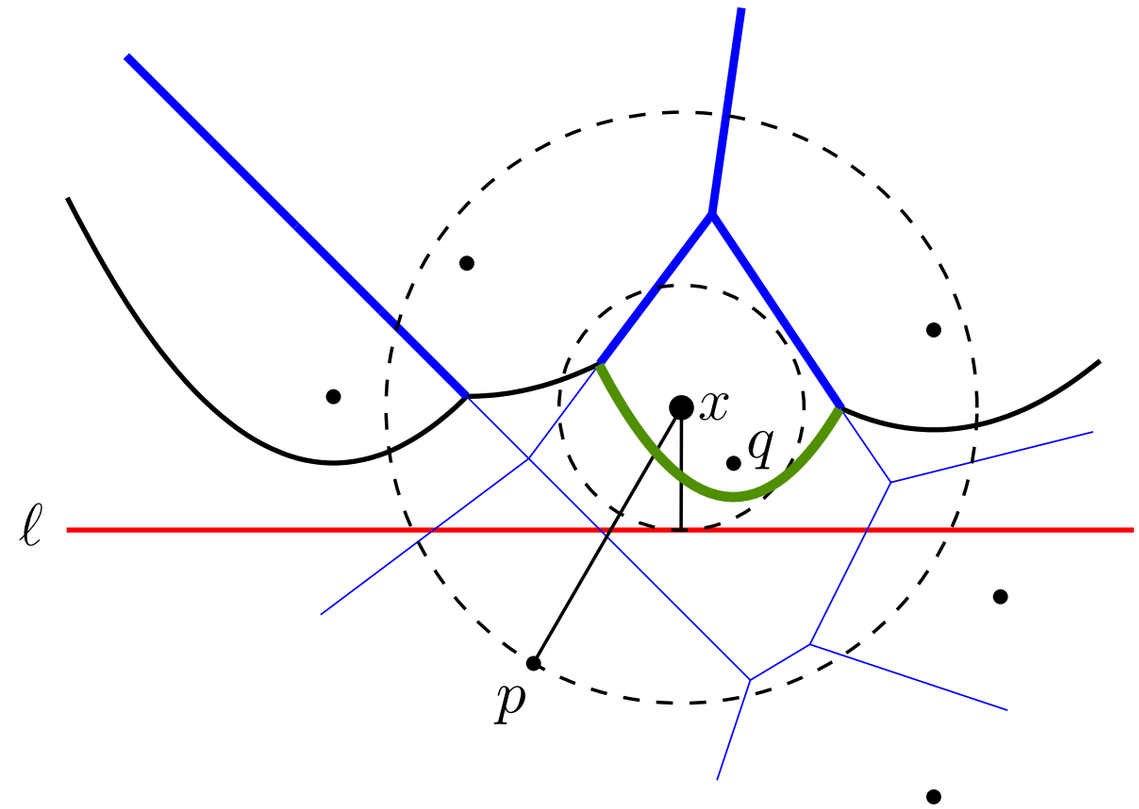
- $\exists$ parabola β: x above β
  $\Rightarrow$ nearest neighbor q not below ℓ.

Beach line:

- $p_1, \ldots, p_k$ above ℓ $\rightarrow$ parabolas $\beta_1, \ldots, \beta_k$
- Beach line: $f : \mathbb{R} \rightarrow \mathbb{R}^2$ with $f(x) :=$
  point on $\beta_1 \cup \cdots \cup \beta_k$ with min. y-coord.
- By construction monotonic

Intuition:

- Let $x \in \mathbb{R}^2$ be above ℓ and $p \in \mathcal{P}$ below ℓ
  $\Rightarrow d(x,p) \geq d(x,\ell)$
- Let $q \in \mathcal{P}$ be nearest site for x.
  If $d(x,q) \leq d(x,\ell)$, then q not below ℓ.
- $\{x \in \mathbb{R}^2 \mid d(x,q) \leq d(x,\ell)\}$
  is bounded by parabola.



Consequence:

- $\exists$ parabola β: x above β
  $\Rightarrow$ nearest neighbor q not below ℓ.

Beach line:

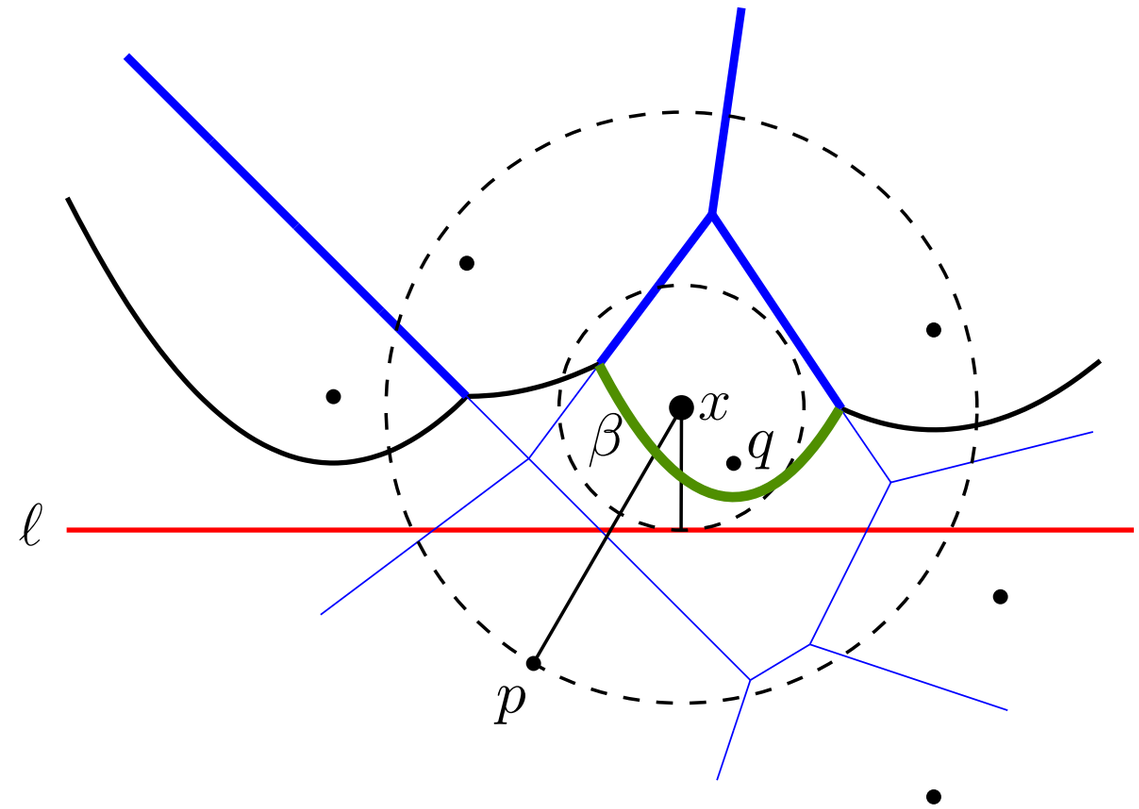
- $p_1, \ldots, p_k$ above ℓ $\rightarrow$ parabolas $\beta_1, \ldots, \beta_k$
- Beach line: $f : \mathbb{R} \rightarrow \mathbb{R}^2$ with $f(x) :=$
  point on $\beta_1 \cup \cdots \cup \beta_k$ with min. y-coord.
- By construction monotonic

Intuition:

- Let $x \in \mathbb{R}^2$ be above ℓ and $p \in \mathcal{P}$ below ℓ
  $\Rightarrow d(x,p) \geq d(x,\ell)$
- Let $q \in \mathcal{P}$ be nearest site for x.
  If $d(x,q) \leq d(x,\ell)$, then q not below ℓ.
- $\{x \in \mathbb{R}^2 \mid d(x,q) \leq d(x,\ell)\}$
  is bounded by parabola.

Consequence:

- $\exists$ parabola β: x above β
  $\Rightarrow$ nearest neighbor q not below ℓ.



Beach line:

- $p_1, \ldots, p_k$ above ℓ $\rightarrow$ parabolas $\beta_1, \ldots, \beta_k$
- Beach line: $f : \mathbb{R} \rightarrow \mathbb{R}^2$ with $f(x) :=$
  point on $\beta_1 \cup \cdots \cup \beta_k$ with min. y-coord.
- By construction monotonic

Intuition:

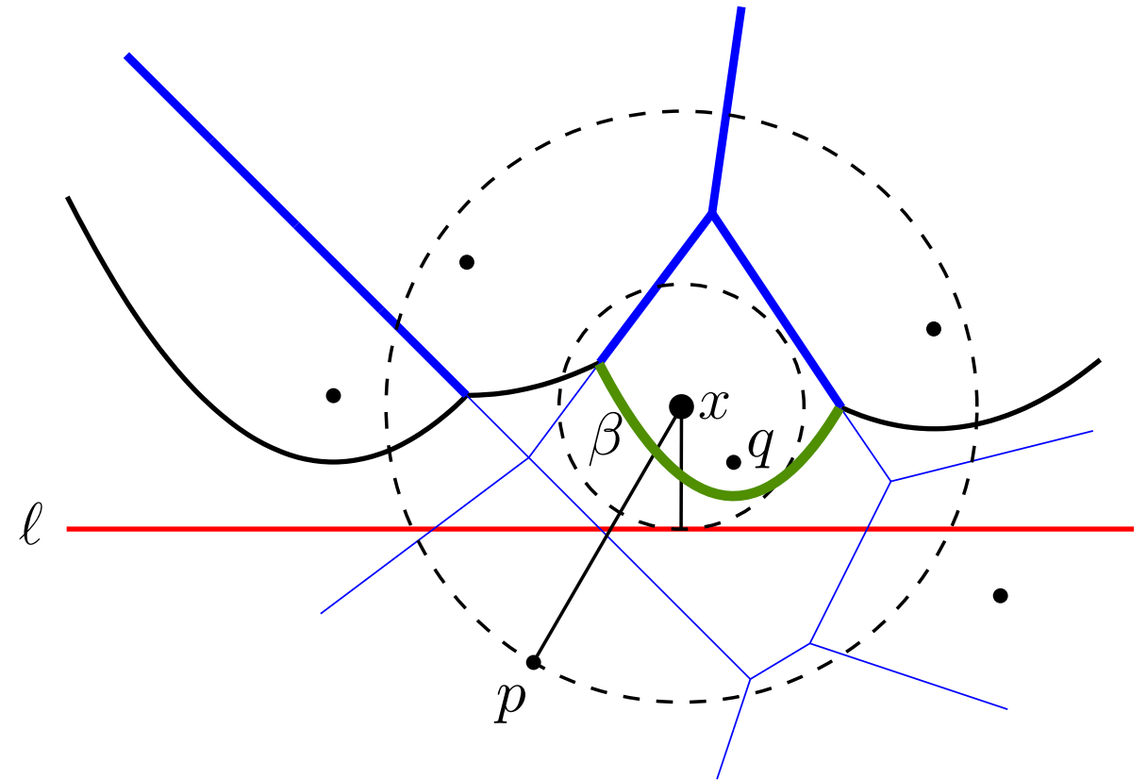
- Let $x \in \mathbb{R}^2$ be above ℓ and $p \in \mathcal{P}$ below ℓ
  $\Rightarrow d(x,p) \geq d(x,\ell)$
- Let $q \in \mathcal{P}$ be nearest site for x.
  If $d(x,q) \leq d(x,\ell)$, then q not below ℓ.
- $\{x \in \mathbb{R}^2 \mid d(x,q) \leq d(x,\ell)\}$
  is bounded by parabola.

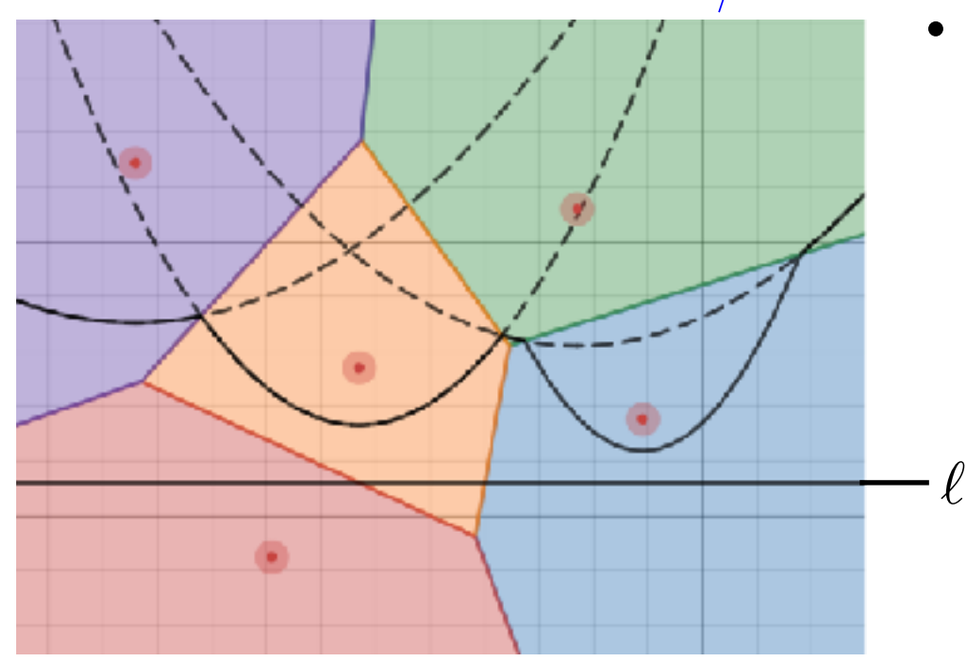
Consequence:

- $\exists$ parabola β: x above β
  $\Rightarrow$ nearest neighbor q not below ℓ.

Beach line:

- $p_1, \ldots, p_k$ above ℓ $\rightarrow$ parabolas $\beta_1, \ldots, \beta_k$
- Beach line: $f : \mathbb{R} \rightarrow \mathbb{R}^2$ with $f(x) :=$
  point on $\beta_1 \cup \cdots \cup \beta_k$ with min. y-coord.
- By construction monotonic





Intuition:

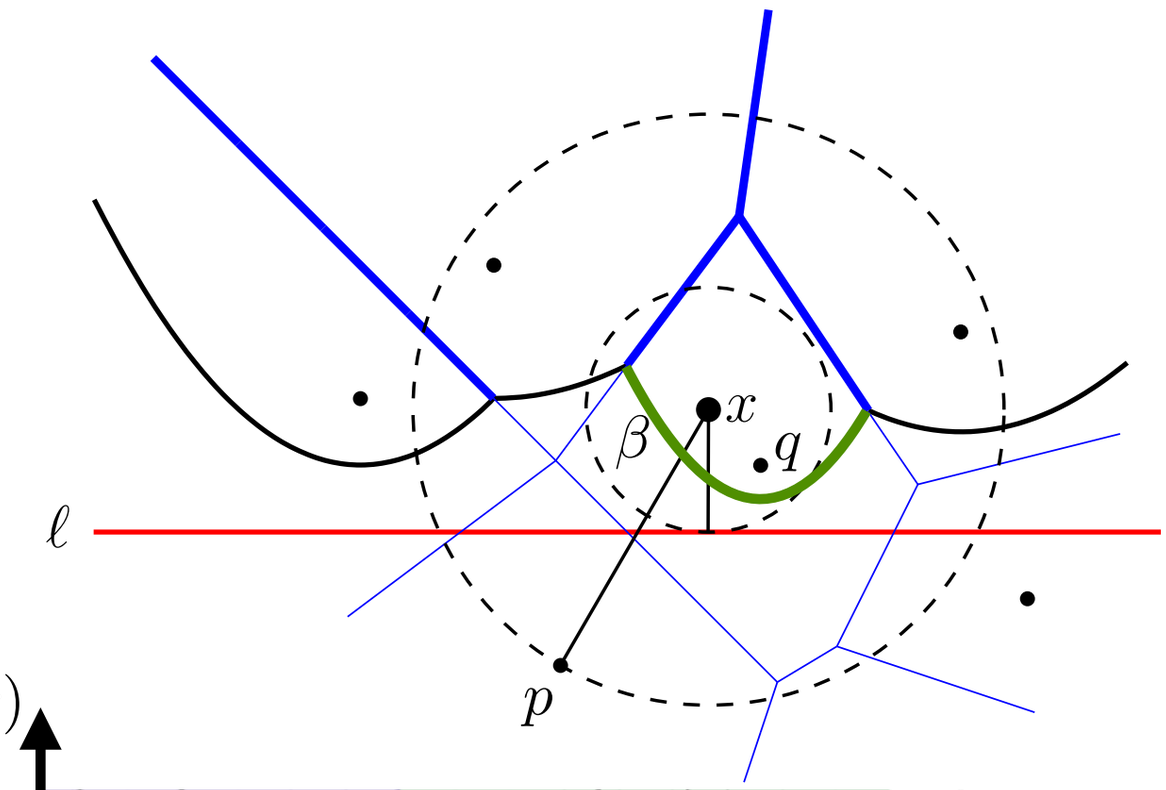
- Let $x \in \mathbb{R}^2$ be above ℓ and $p \in \mathcal{P}$ below ℓ
  $\Rightarrow d(x,p) \geq d(x,\ell)$
- Let $q \in \mathcal{P}$ be nearest site for x.
  If $d(x,q) \leq d(x,\ell)$, then q not below ℓ.
- $\{x \in \mathbb{R}^2 \mid d(x,q) \leq d(x,\ell)\}$
  is bounded by parabola.

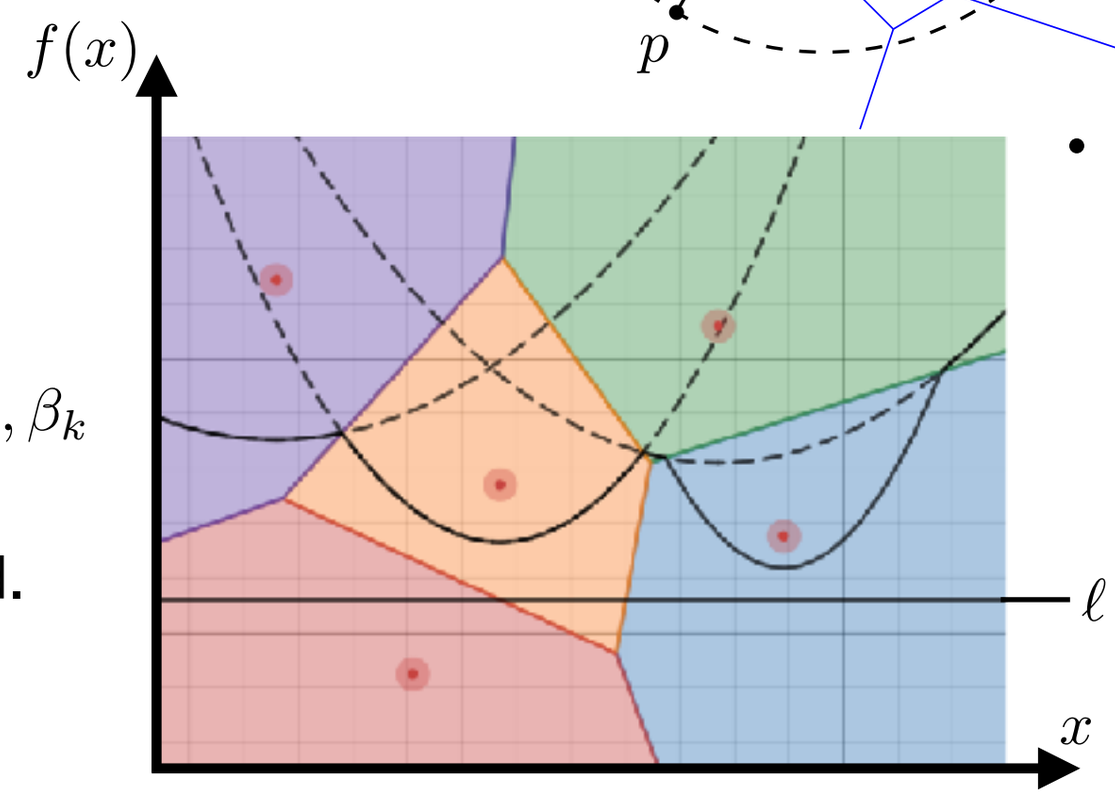
Consequence:

- $\exists$ parabola β : x above β
  $\Rightarrow$ nearest neighbor q not below ℓ.

Beach line:

- $p_1, \ldots, p_k$ above ℓ $\rightarrow$ parabolas $\beta_1, \ldots, \beta_k$
- Beach line: $f : \mathbb{R} \rightarrow \mathbb{R}^2$ with $f(x) :=$
  point on $\beta_1 \cup \cdots \cup \beta_k$ with min. y-coord.
- By construction monotonic





Intuition:

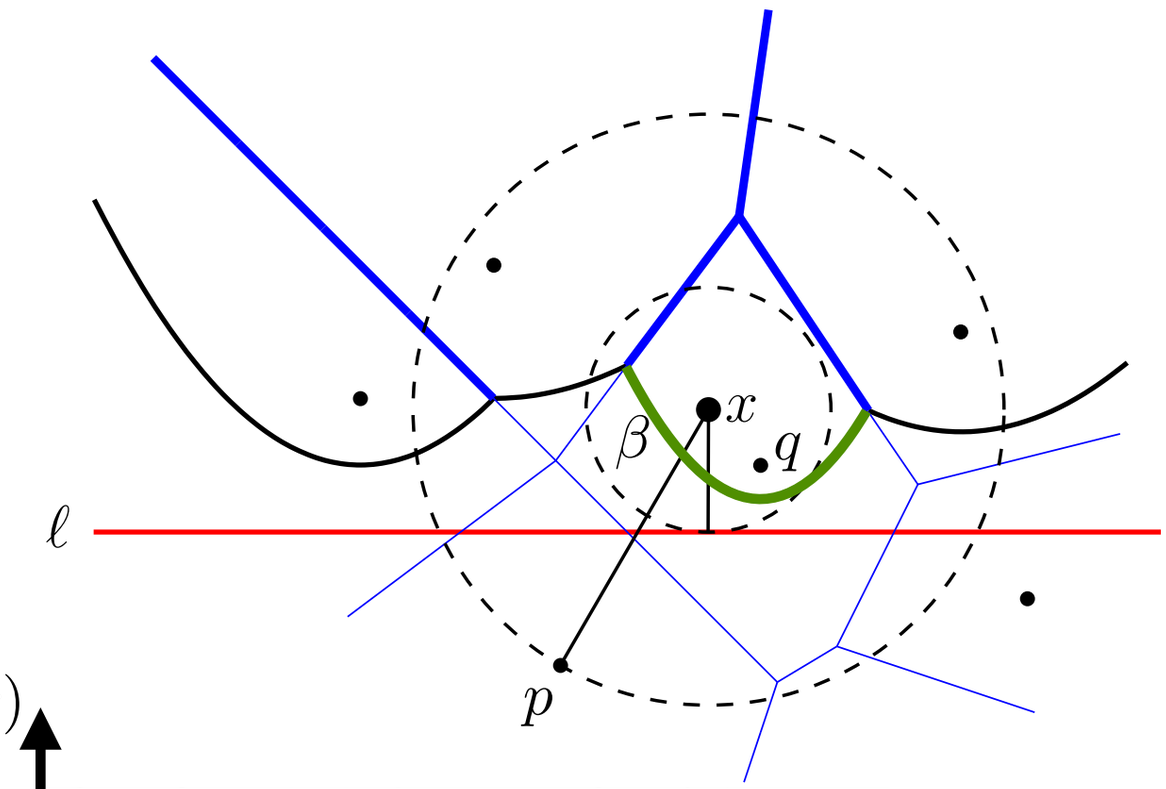
- Let $x \in \mathbb{R}^2$ be above ℓ and $p \in \mathcal{P}$ below ℓ
  $\Rightarrow d(x,p) \geq d(x,\ell)$
- Let $q \in \mathcal{P}$ be nearest site for x.
  If $d(x,q) \leq d(x,\ell)$, then q not below ℓ.
- $\{x \in \mathbb{R}^2 \mid d(x,q) \leq d(x,\ell)\}$
  is bounded by parabola.

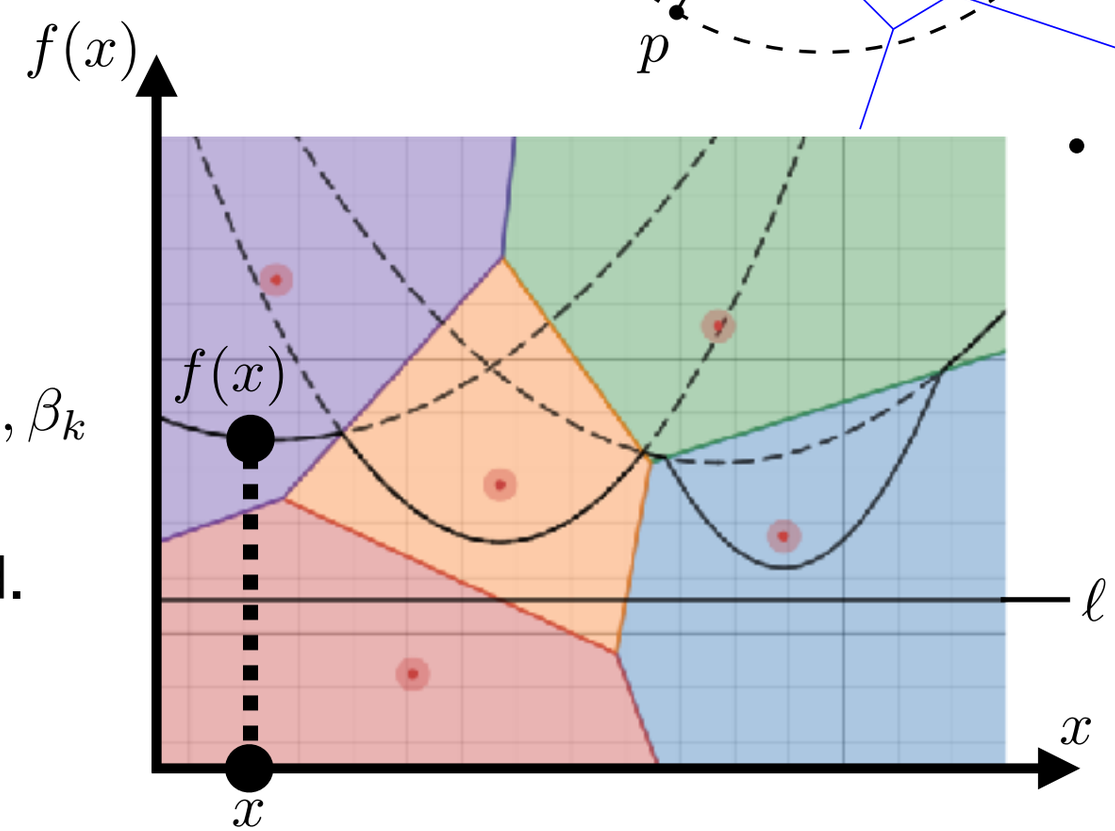
Consequence:

- $\exists$ parabola β : x above β
  $\Rightarrow$ nearest neighbor q not below ℓ.

Beach line:

- $p_1, \ldots, p_k$ above ℓ $\rightarrow$ parabolas $\beta_1, \ldots, \beta_k$
- Beach line: $f : \mathbb{R} \rightarrow \mathbb{R}^2$ with $f(x) :=$
  point on $\beta_1 \cup \cdots \cup \beta_k$ with min. y-coord.
- By construction monotonic





Intuition:

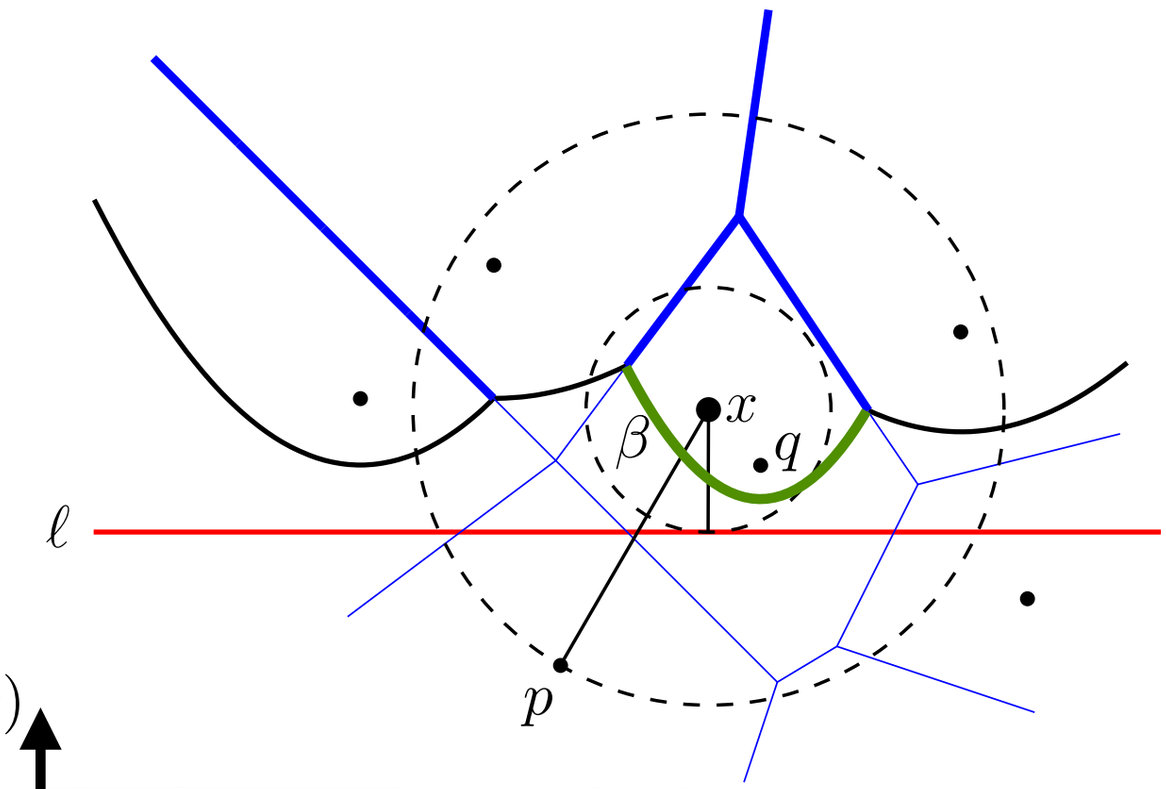
- Let $x \in \mathbb{R}^2$ be above ℓ and $p \in \mathcal{P}$ below ℓ
  $\Rightarrow d(x,p) \geq d(x,\ell)$
- Let $q \in \mathcal{P}$ be nearest site for x.
  If $d(x,q) \leq d(x,\ell)$, then q not below ℓ.
- $\{x \in \mathbb{R}^2 \mid d(x,q) \leq d(x,\ell)\}$
  is bounded by parabola.

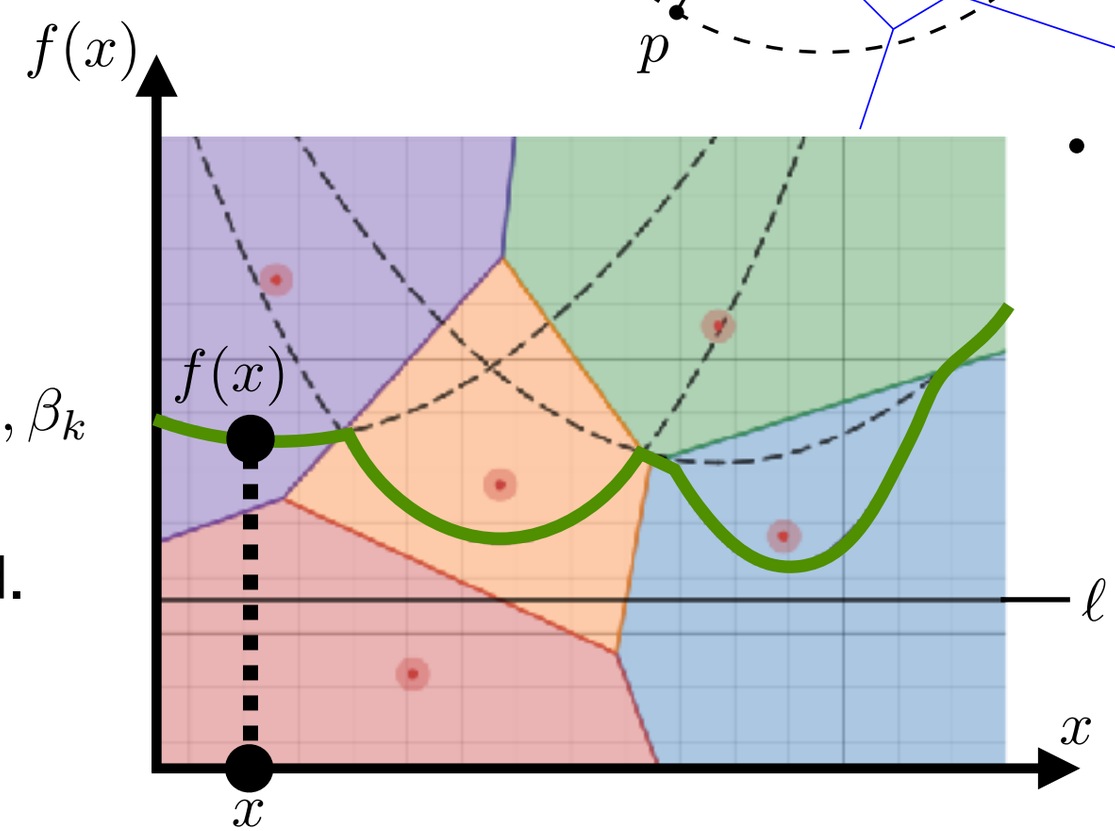
Consequence:

- $\exists$ parabola β : x above β
  $\Rightarrow$ nearest neighbor q not below ℓ.

Beach line:

- $p_1, \ldots, p_k$ above ℓ $\rightarrow$ parabolas $\beta_1, \ldots, \beta_k$
- Beach line: $f : \mathbb{R} \rightarrow \mathbb{R}^2$ with $f(x) :=$
  point on $\beta_1 \cup \cdots \cup \beta_k$ with min. y-coord.
- By construction monotonic





Intuition:

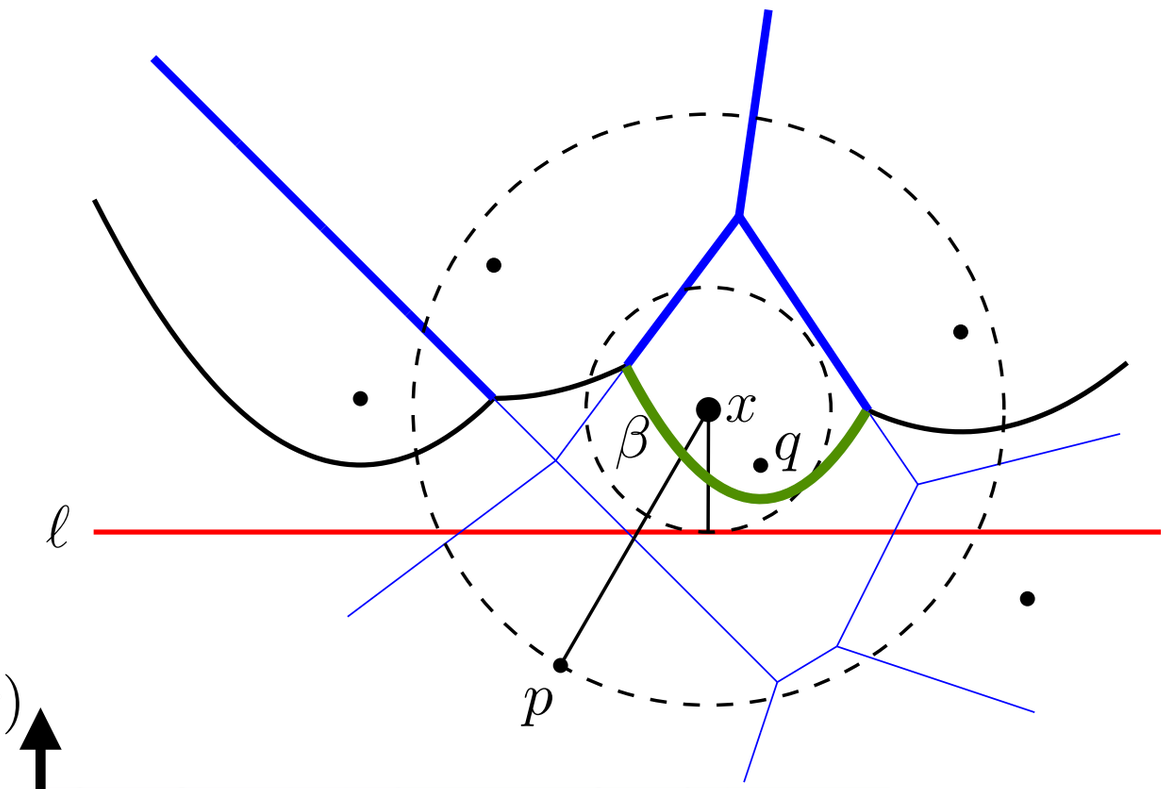
- Let $x \in \mathbb{R}^2$ be above ℓ and $p \in \mathcal{P}$ below ℓ
  $\Rightarrow d(x,p) \geq d(x,\ell)$
- Let $q \in \mathcal{P}$ be nearest site for x.
  If $d(x,q) \leq d(x,\ell)$, then q not below ℓ.
- $\{x \in \mathbb{R}^2 \mid d(x,q) \leq d(x,\ell)\}$
  is bounded by parabola.

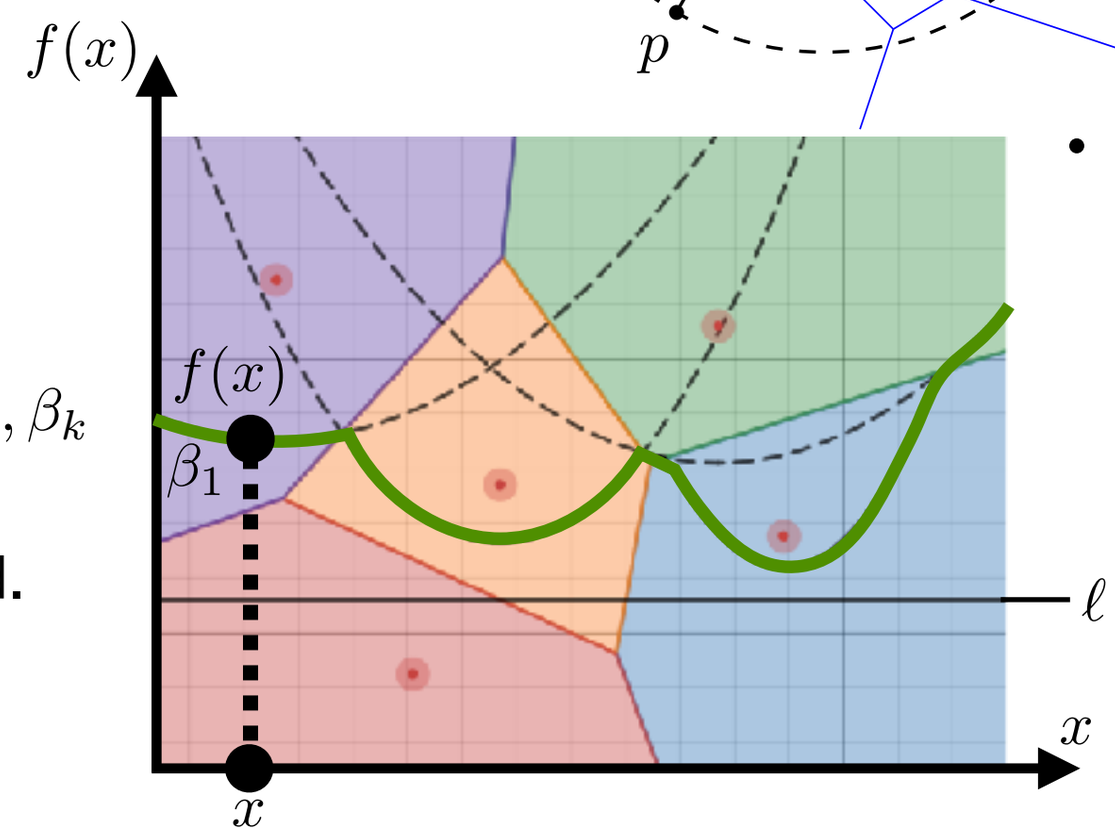
Consequence:

- $\exists$ parabola β : x above β
  $\Rightarrow$ nearest neighbor q not below ℓ.

Beach line:

- $p_1, \ldots, p_k$ above ℓ $\rightarrow$ parabolas $\beta_1, \ldots, \beta_k$
- Beach line: $f : \mathbb{R} \rightarrow \mathbb{R}^2$ with $f(x) :=$
  point on $\beta_1 \cup \cdots \cup \beta_k$ with min. y-coord.
- By construction monotonic





Intuition:

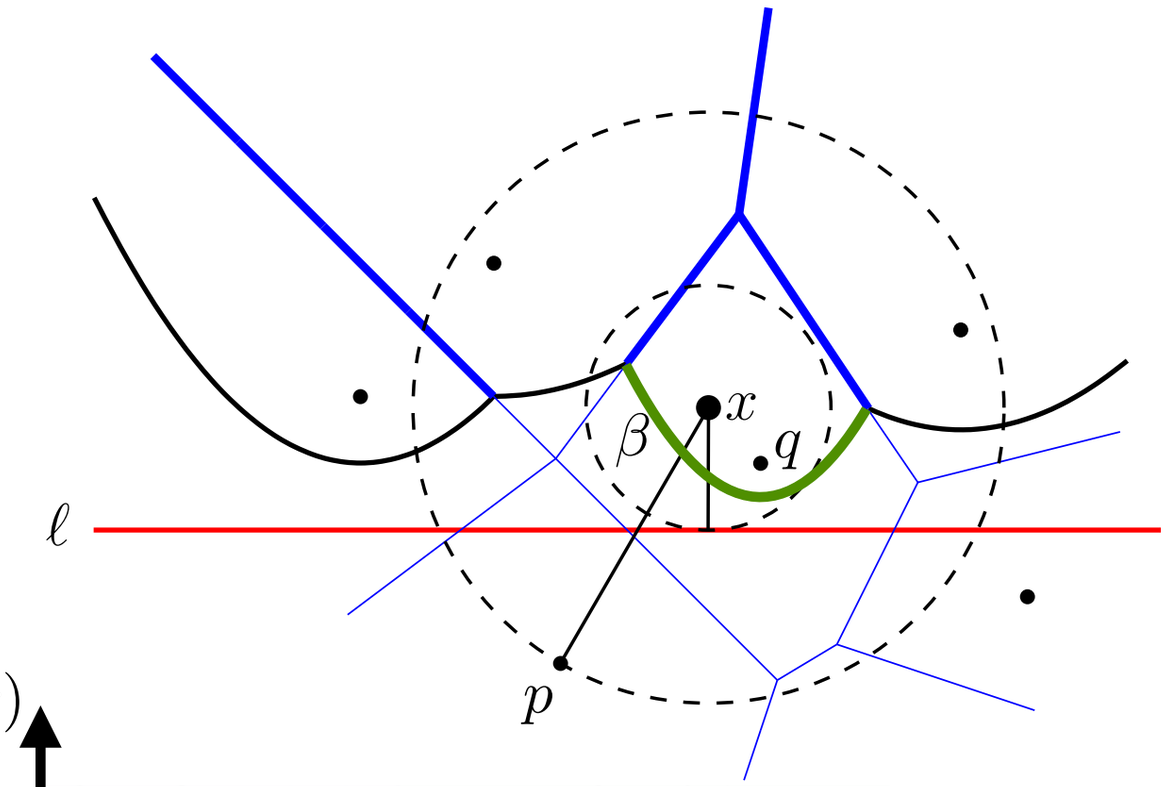
- Let $x \in \mathbb{R}^2$ be above ℓ and $p \in \mathcal{P}$ below ℓ
  $\Rightarrow d(x,p) \geq d(x,\ell)$
- Let $q \in \mathcal{P}$ be nearest site for x.
  If $d(x,q) \leq d(x,\ell)$, then q not below ℓ.
- $\{x \in \mathbb{R}^2 \mid d(x,q) \leq d(x,\ell)\}$
  is bounded by parabola.

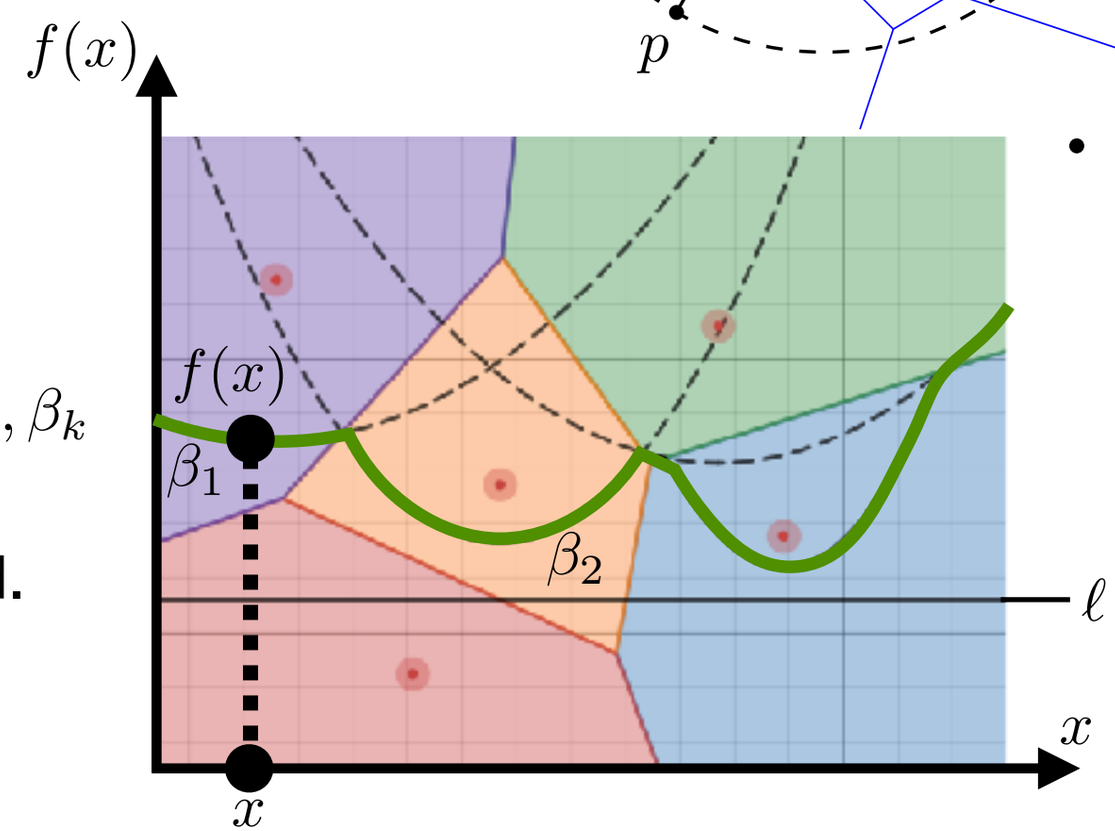
Consequence:

- $\exists$ parabola β : x above β
  $\Rightarrow$ nearest neighbor q not below ℓ.

Beach line:

- $p_1, \ldots, p_k$ above ℓ $\rightarrow$ parabolas $\beta_1, \ldots, \beta_k$
- Beach line: $f : \mathbb{R} \rightarrow \mathbb{R}^2$ with $f(x) :=$
  point on $\beta_1 \cup \cdots \cup \beta_k$ with min. y-coord.
- By construction monotonic





Intuition:

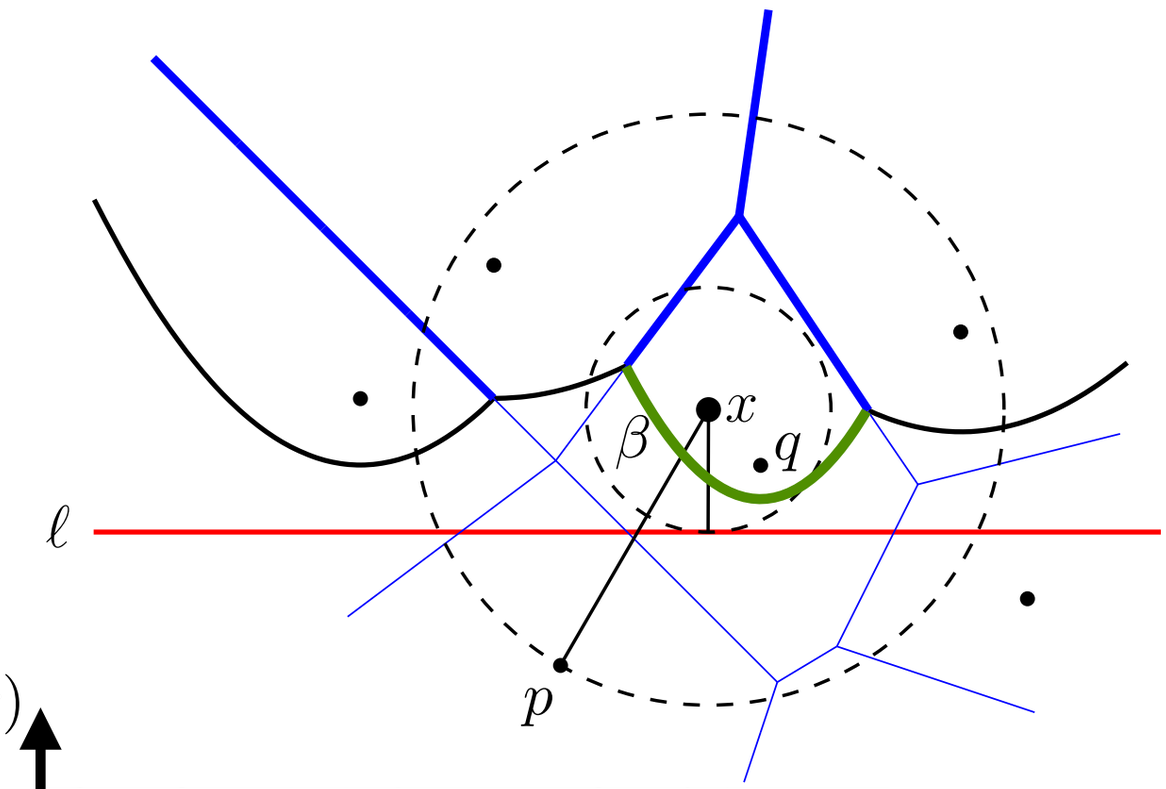
- Let $x \in \mathbb{R}^2$ be above ℓ and $p \in \mathcal{P}$ below ℓ
  $\Rightarrow d(x,p) \geq d(x,\ell)$
- Let $q \in \mathcal{P}$ be nearest site for x.
  If $d(x,q) \leq d(x,\ell)$, then q not below ℓ.
- $\{x \in \mathbb{R}^2 \mid d(x,q) \leq d(x,\ell)\}$
  is bounded by parabola.

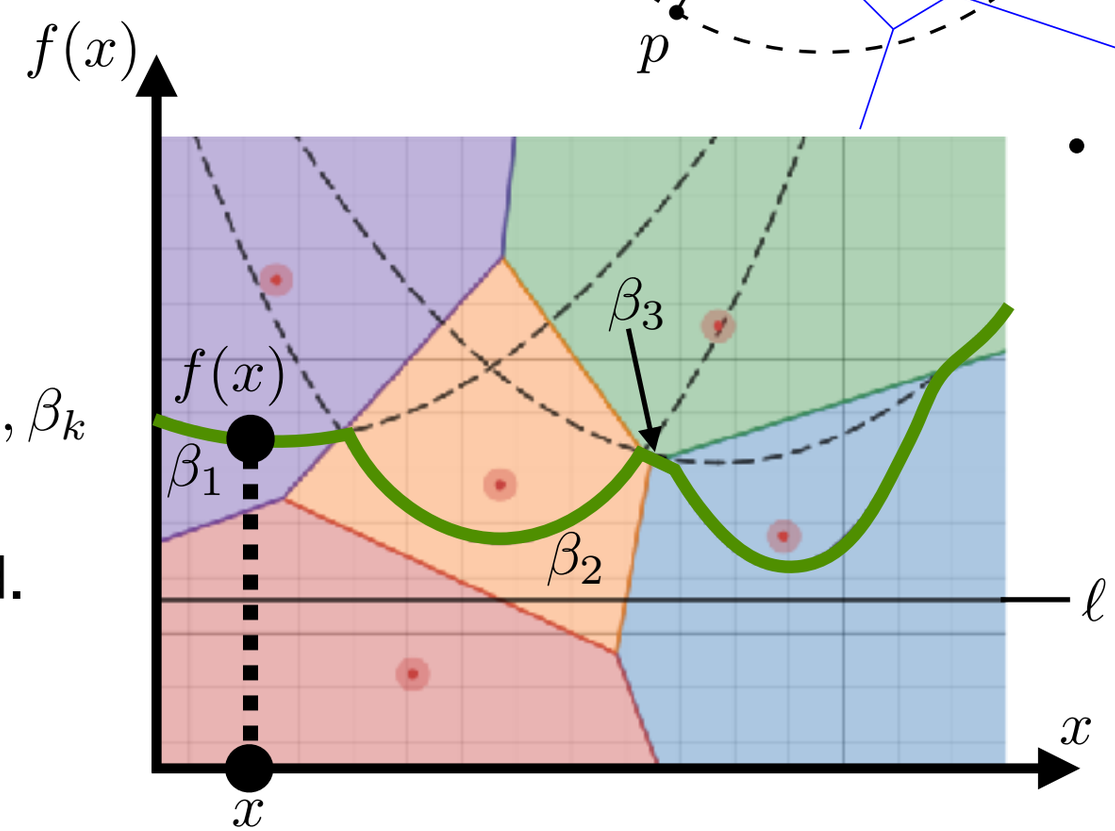
Consequence:

- $\exists$ parabola β : x above β
  $\Rightarrow$ nearest neighbor q not below ℓ.

Beach line:

- $p_1, \ldots, p_k$ above ℓ $\rightarrow$ parabolas $\beta_1, \ldots, \beta_k$
- Beach line: $f : \mathbb{R} \rightarrow \mathbb{R}^2$ with $f(x) :=$
  point on $\beta_1 \cup \cdots \cup \beta_k$ with min. y-coord.
- By construction monotonic





Intuition:

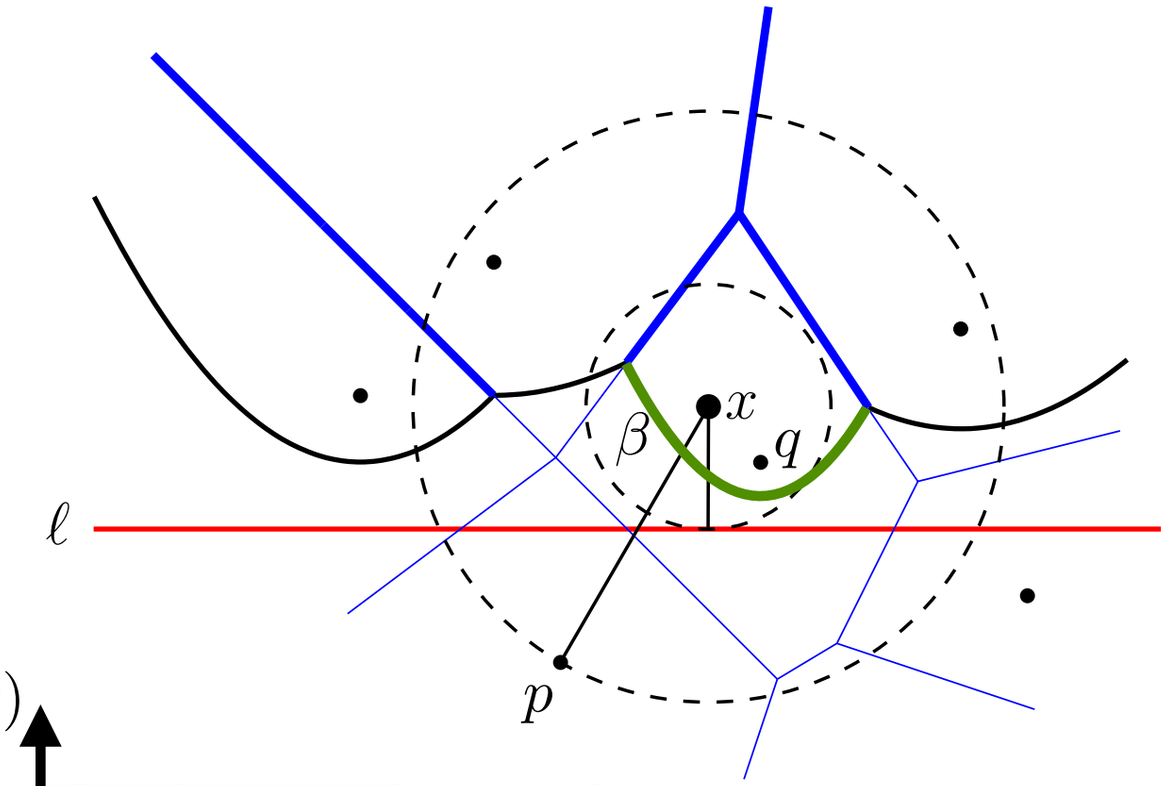
- Let $x \in \mathbb{R}^2$ be above ℓ and $p \in \mathcal{P}$ below ℓ
  $\Rightarrow d(x,p) \geq d(x,\ell)$
- Let $q \in \mathcal{P}$ be nearest site for x.
  If $d(x,q) \leq d(x,\ell)$, then q not below ℓ.
- $\{x \in \mathbb{R}^2 \mid d(x,q) \leq d(x,\ell)\}$
  is bounded by parabola.

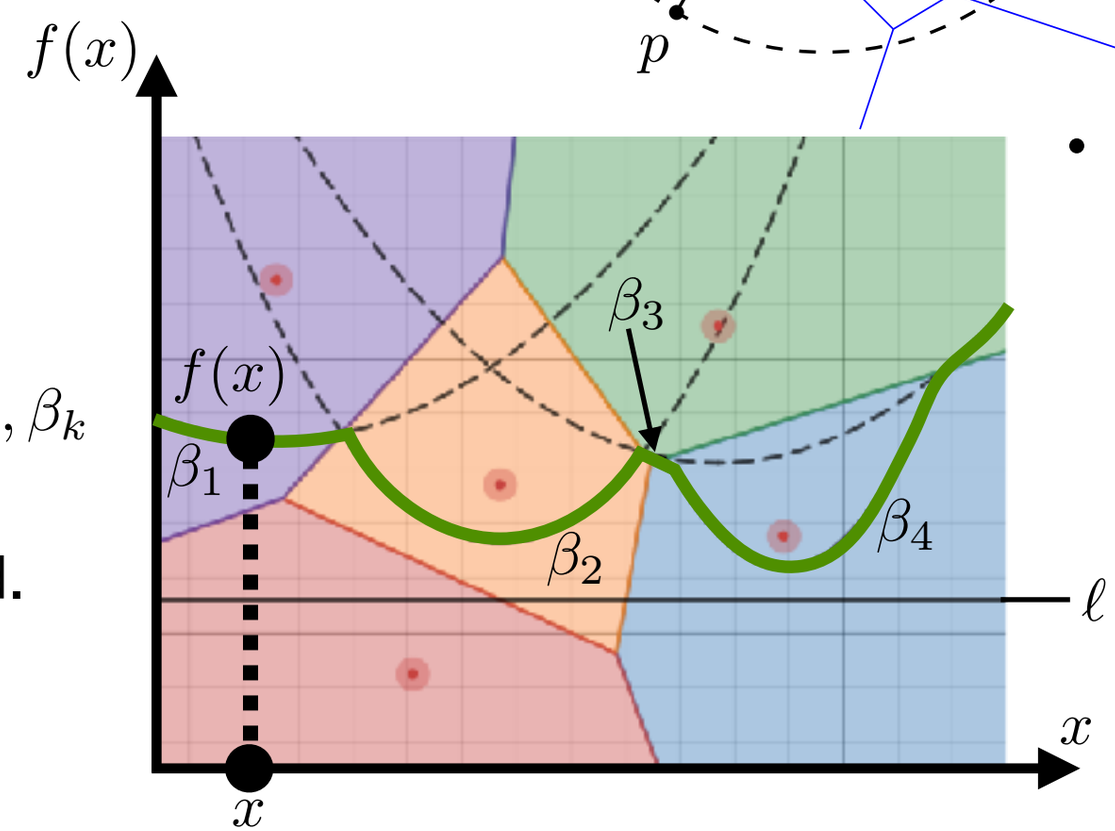
Consequence:

- $\exists$ parabola β : x above β
  $\Rightarrow$ nearest neighbor q not below ℓ.

Beach line:

- $p_1, \ldots, p_k$ above ℓ $\rightarrow$ parabolas $\beta_1, \ldots, \beta_k$
- Beach line: $f : \mathbb{R} \rightarrow \mathbb{R}^2$ with $f(x) :=$ point on $\beta_1 \cup \cdots \cup \beta_k$ with min. y-coord.
- By construction monotonic





Intuition:

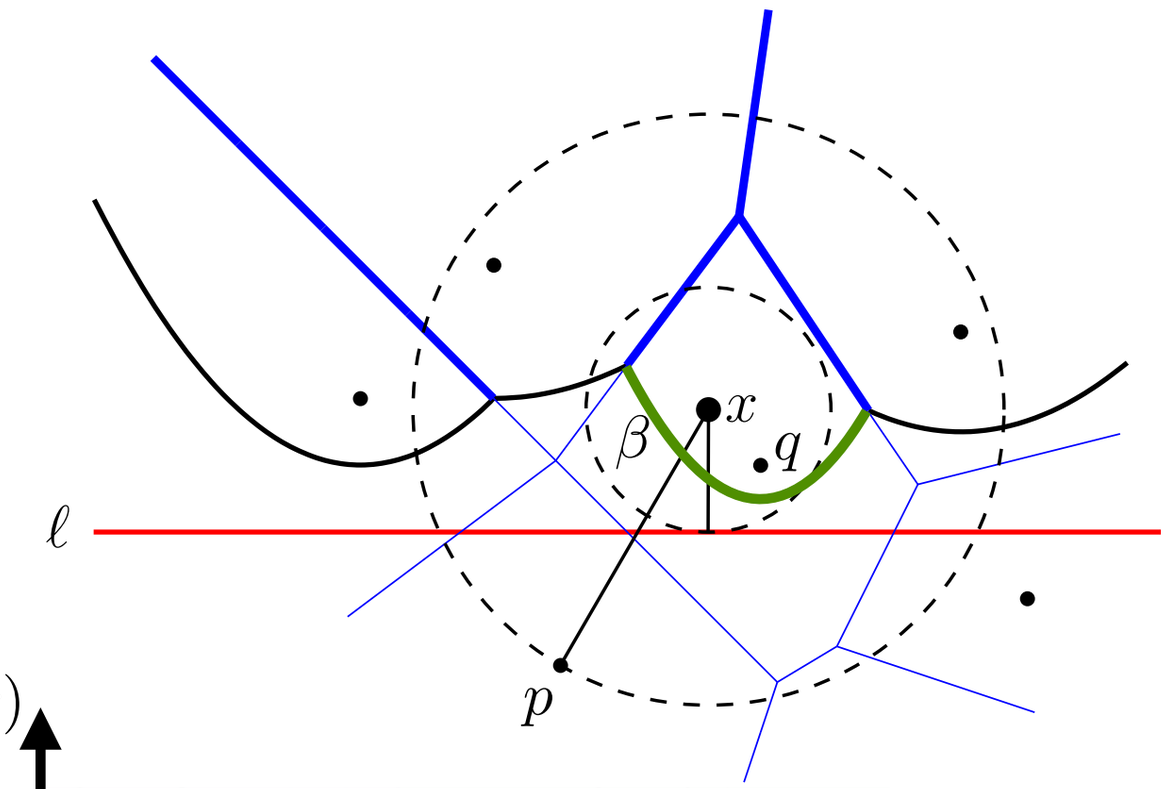
- Let $x \in \mathbb{R}^2$ be above ℓ and $p \in \mathcal{P}$ below ℓ
  $\Rightarrow d(x,p) \geq d(x,\ell)$
- Let $q \in \mathcal{P}$ be nearest site for x.
  If $d(x,q) \leq d(x,\ell)$, then q not below ℓ.
- $\{x \in \mathbb{R}^2 \mid d(x,q) \leq d(x,\ell)\}$
  is bounded by parabola.

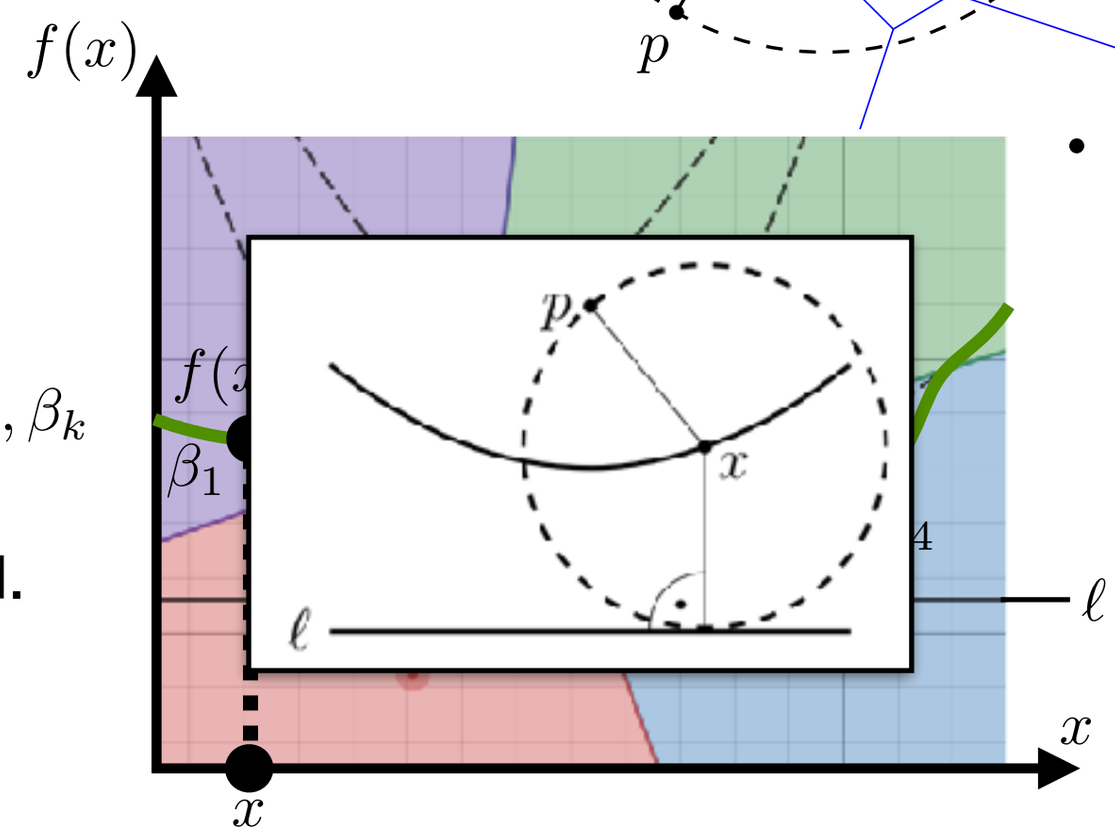
Consequence:

- $\exists$ parabola β : x above β
  $\Rightarrow$ nearest neighbor q not below ℓ.

Beach line:

- $p_1, \ldots, p_k$ above ℓ $\rightarrow$ parabolas $\beta_1, \ldots, \beta_k$
- Beach line: $f : \mathbb{R} \rightarrow \mathbb{R}^2$ with $f(x) :=$
  point on $\beta_1 \cup \cdots \cup \beta_k$ with min. y-coord.
- By construction monotonic





Intuition:

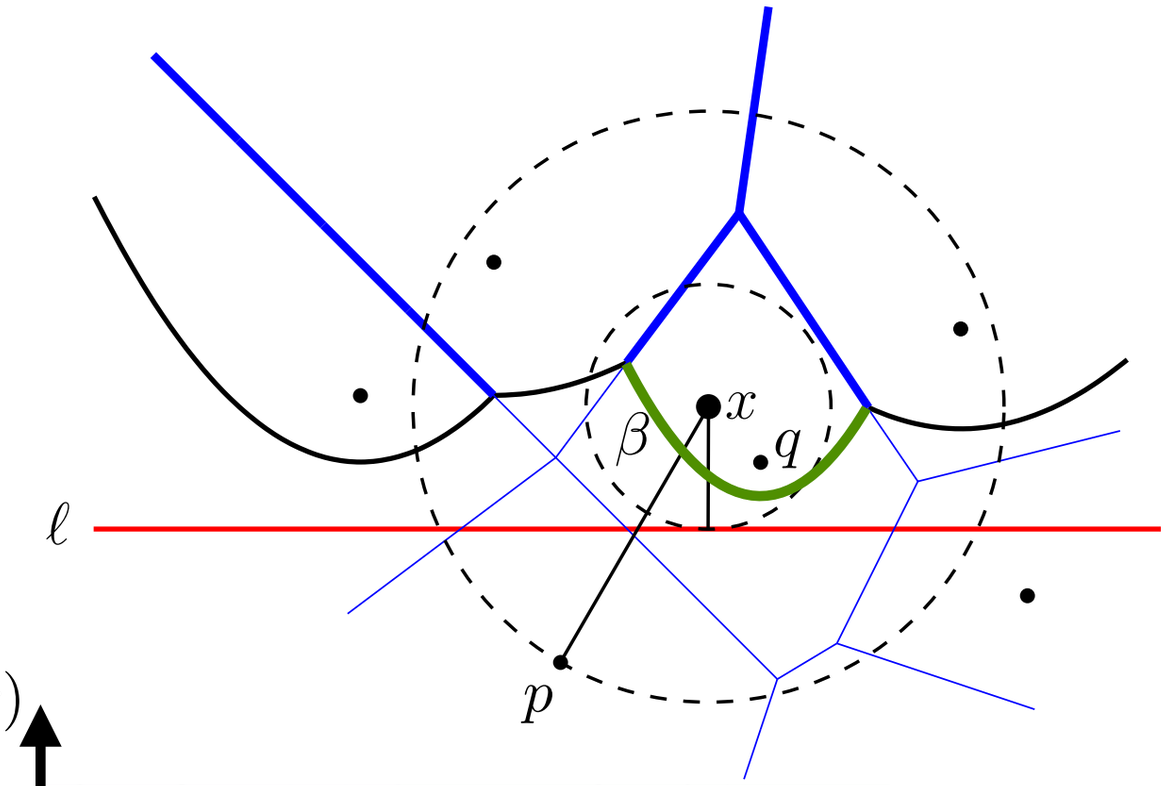
- Let $x \in \mathbb{R}^2$ be above ℓ and $p \in \mathcal{P}$ below ℓ
  $\Rightarrow d(x,p) \geq d(x,\ell)$
- Let $q \in \mathcal{P}$ be nearest site for x.
  If $d(x,q) \leq d(x,\ell)$, then q not below ℓ.
- $\{x \in \mathbb{R}^2 \mid d(x,q) \leq d(x,\ell)\}$
  is bounded by parabola.

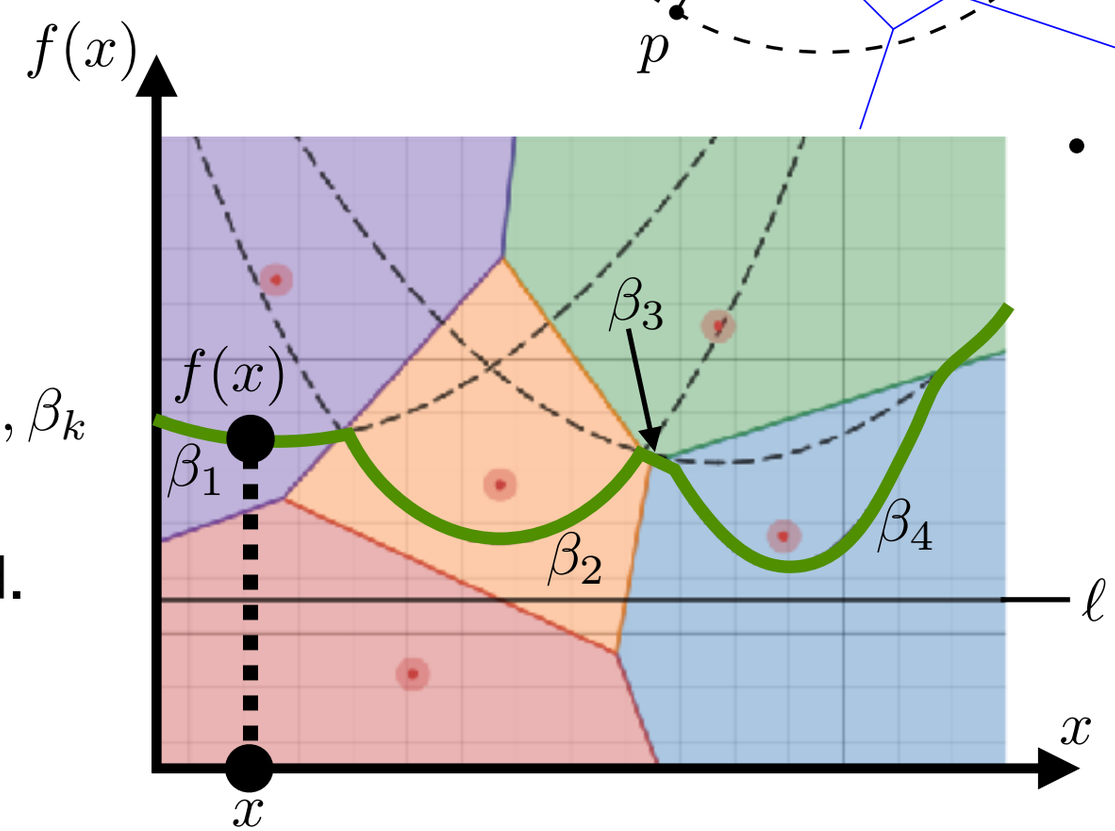
Consequence:

- $\exists$ parabola β : x above β
  $\Rightarrow$ nearest neighbor q not below ℓ.

Beach line:

- $p_1, \ldots, p_k$ above ℓ $\rightarrow$ parabolas $\beta_1, \ldots, \beta_k$
- Beach line: $f : \mathbb{R} \rightarrow \mathbb{R}^2$ with $f(x) :=$ point on $\beta_1 \cup \cdots \cup \beta_k$ with min. y-coord.
- By construction monotonic

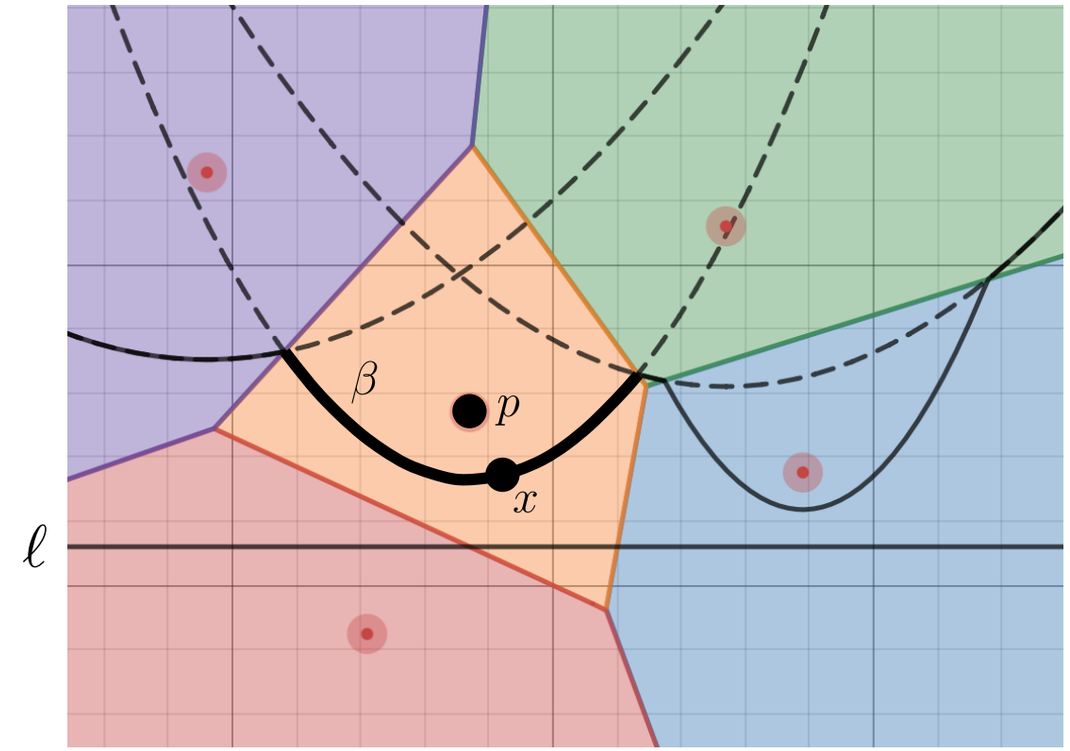




Lemma 4.15

$p \in \mathcal{P}$ defines arc β on beach line

$\Rightarrow$ p is nearest site $\forall x \in \beta$.



Proof:

- Assume: $\exists q \in P : d(x,q) < d(x,p)$
- $d(x,p) = d(x,\ell)$
- Case 1: q not above ℓ
  $\Rightarrow d(x,q) \geq d(x,\ell) = d(x,p)$ ↯
- Case 2: q above ℓ
  $\Rightarrow \exists$arc γ (defined by q)
  $\rightarrow d(x,q) < d(x,p) = d(x,\ell) \rightarrow$ x above γ
- But $x \in \beta$ on beach line. ↯ $\square$

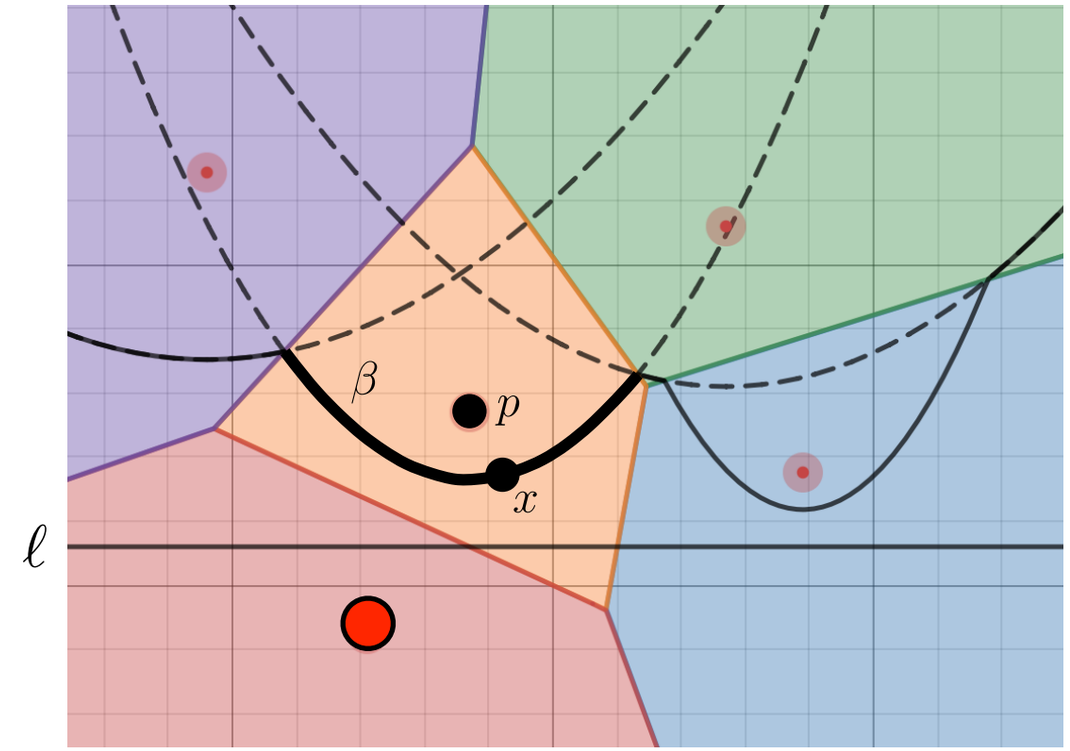
Corollary 4.16

Intersection points of adjacent arcs lie on Voronoi edges.

Lemma 4.15

$p \in \mathcal{P}$ defines arc β on beach line

$\Rightarrow$ p is nearest site $\forall x \in \beta$.



Proof:

- Assume: $\exists q \in P : d(x,q) < d(x,p)$
- $d(x,p) = d(x,\ell)$
- Case 1: q not above ℓ
  $\Rightarrow d(x,q) \geq d(x,\ell) = d(x,p)$ ↯
- Case 2: q above ℓ
  $\Rightarrow \exists$arc γ (defined by q)
  $\rightarrow d(x,q) < d(x,p) = d(x,\ell) \rightarrow x$ above γ
- But $x \in \beta$ on beach line. ↯ $\square$

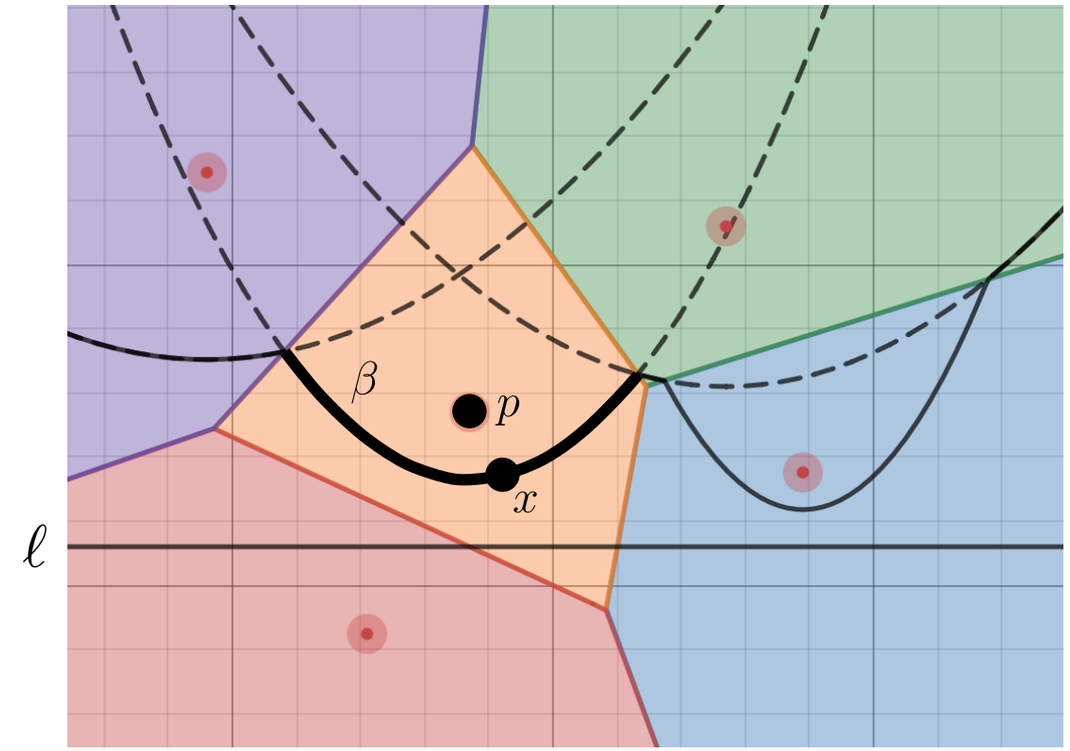
Corollary 4.16

Intersection points of adjacent arcs lie on Voronoi edges.

Lemma 4.15

$p \in \mathcal{P}$ defines arc β on beach line

$\Rightarrow$ p is nearest site $\forall x \in \beta$.



Proof:

- Assume: $\exists q \in P : d(x,q) < d(x,p)$
- $d(x,p) = d(x,\ell)$
- Case 1: q not above ℓ
  $\Rightarrow d(x,q) \geq d(x,\ell) = d(x,p)$ ↯
- Case 2: q above ℓ
  $\Rightarrow \exists$arc γ (defined by q)
  $\rightarrow d(x,q) < d(x,p) = d(x,\ell) \rightarrow$ x above γ
- But $x \in \beta$ on beach line. ↯ $\square$

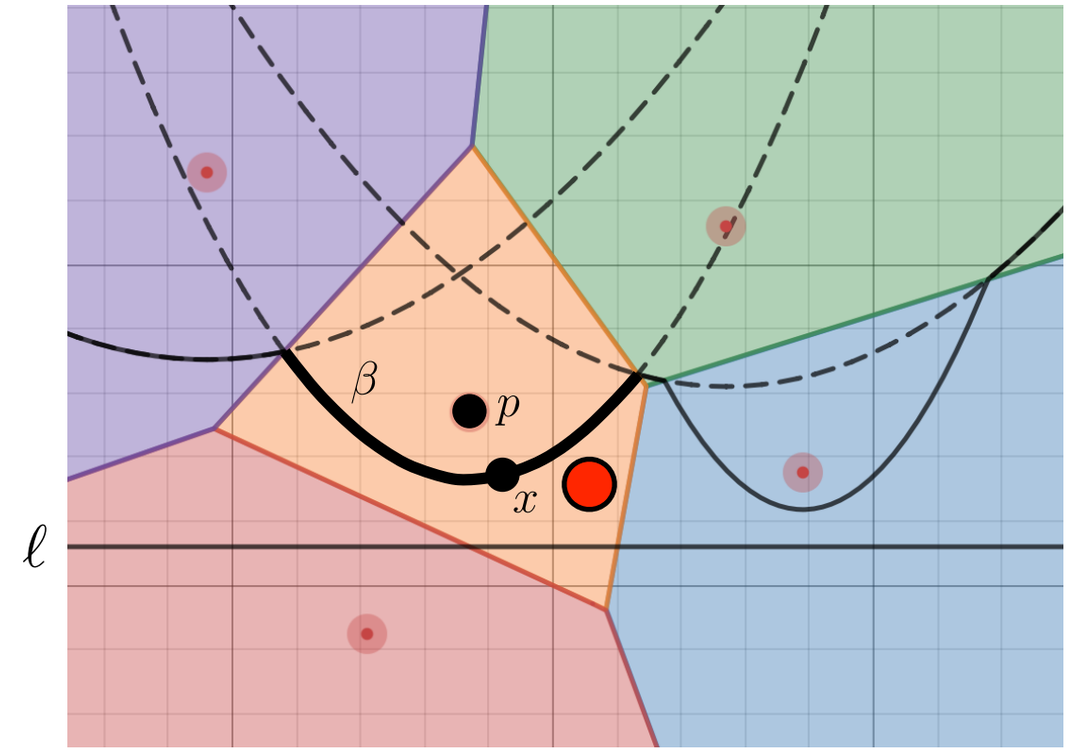
Corollary 4.16

Intersection points of adjacent arcs lie on Voronoi edges.

Lemma 4.15

$p \in \mathcal{P}$ defines arc β on beach line

$\Rightarrow$ p is nearest site $\forall x \in \beta$.



Proof:

- Assume: $\exists q \in P : d(x,q) < d(x,p)$
- $d(x,p) = d(x,\ell)$
- Case 1: q not above ℓ
  $\Rightarrow d(x,q) \geq d(x,\ell) = d(x,p)$ ↯
- Case 2: q above ℓ
  $\Rightarrow \exists$arc γ (defined by q)
  $\rightarrow d(x,q) < d(x,p) = d(x,\ell) \rightarrow x$ above γ
- But $x \in \beta$ on beach line. ↯ $\square$

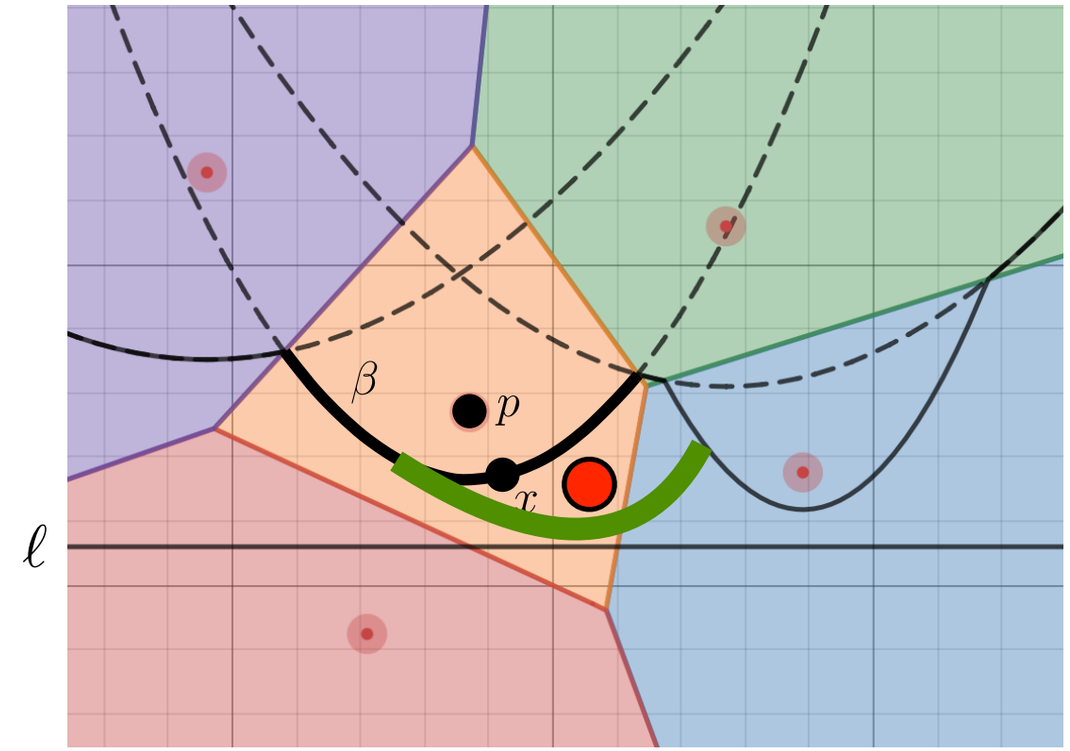
Corollary 4.16

Intersection points of adjacent arcs lie on Voronoi edges.

Lemma 4.15

$p \in \mathcal{P}$ defines arc β on beach line

$\Rightarrow$ p is nearest site $\forall x \in \beta$.



Proof:

- Assume: $\exists q \in P : d(x,q) < d(x,p)$
- $d(x,p) = d(x,\ell)$
- Case 1: q not above ℓ
  $\Rightarrow d(x,q) \geq d(x,\ell) = d(x,p)$ ↯
- Case 2: q above ℓ
  $\Rightarrow \exists$arc γ (defined by q)
  $\rightarrow d(x,q) < d(x,p) = d(x,\ell) \rightarrow x$ above γ
- But $x \in \beta$ on beach line. ↯ $\square$

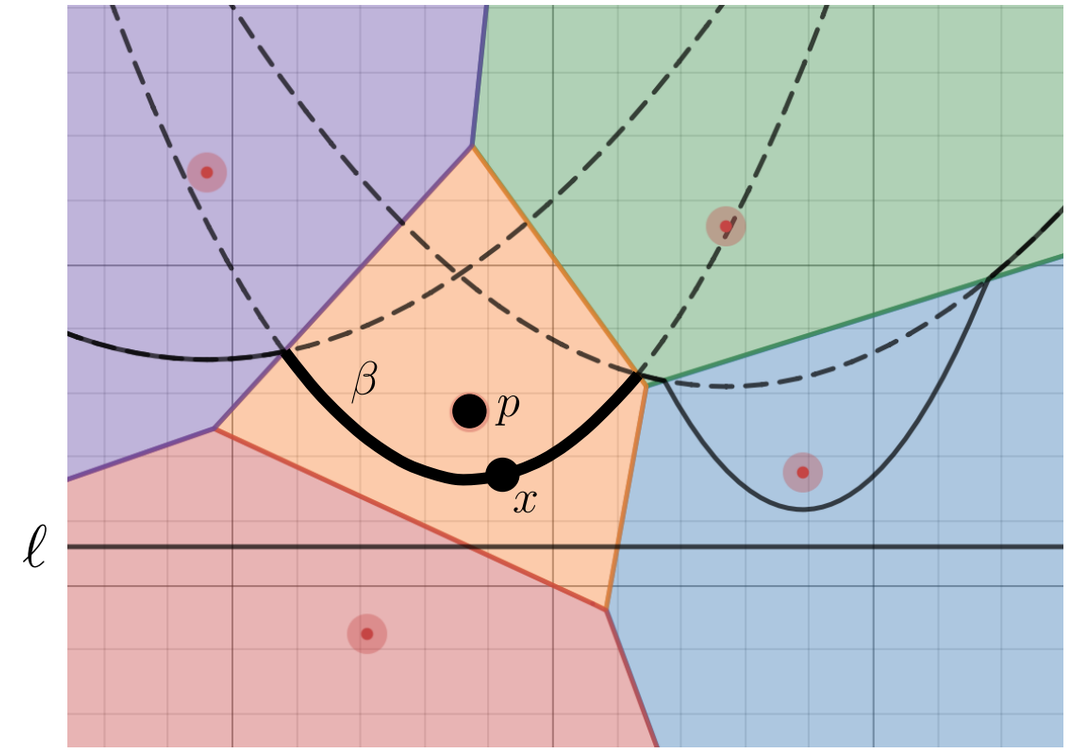
Corollary 4.16

Intersection points of adjacent arcs lie on Voronoi edges.

Lemma 4.15

$p \in \mathcal{P}$ defines arc β on beach line

$\Rightarrow$ p is nearest site $\forall x \in \beta$.



Proof:

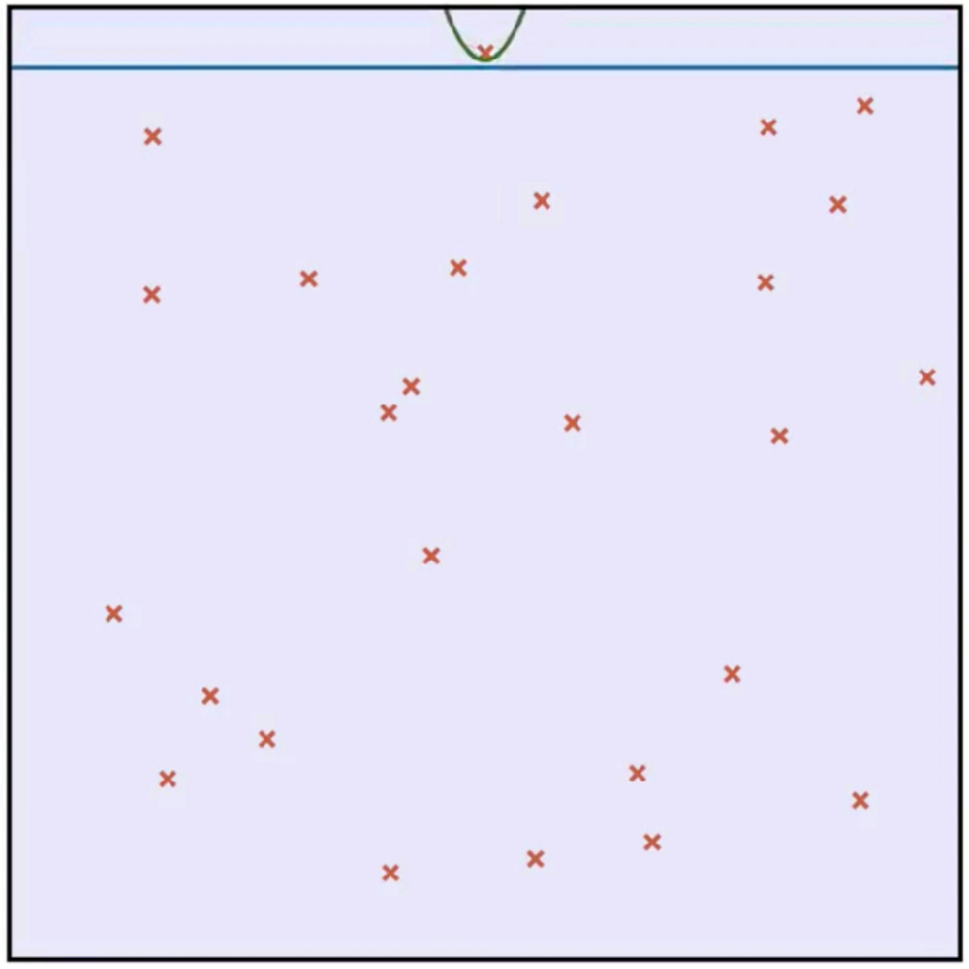
- Assume: $\exists q \in P : d(x,q) < d(x,p)$
- $d(x,p) = d(x,\ell)$
- Case 1: q not above ℓ
  $\Rightarrow d(x,q) \geq d(x,\ell) = d(x,p)$ ↯
- Case 2: q above ℓ
  $\Rightarrow \exists$arc γ (defined by q)
  $\rightarrow d(x,q) < d(x,p) = d(x,\ell) \rightarrow$ x above γ
- But $x \in \beta$ on beach line. ↯ $\square$

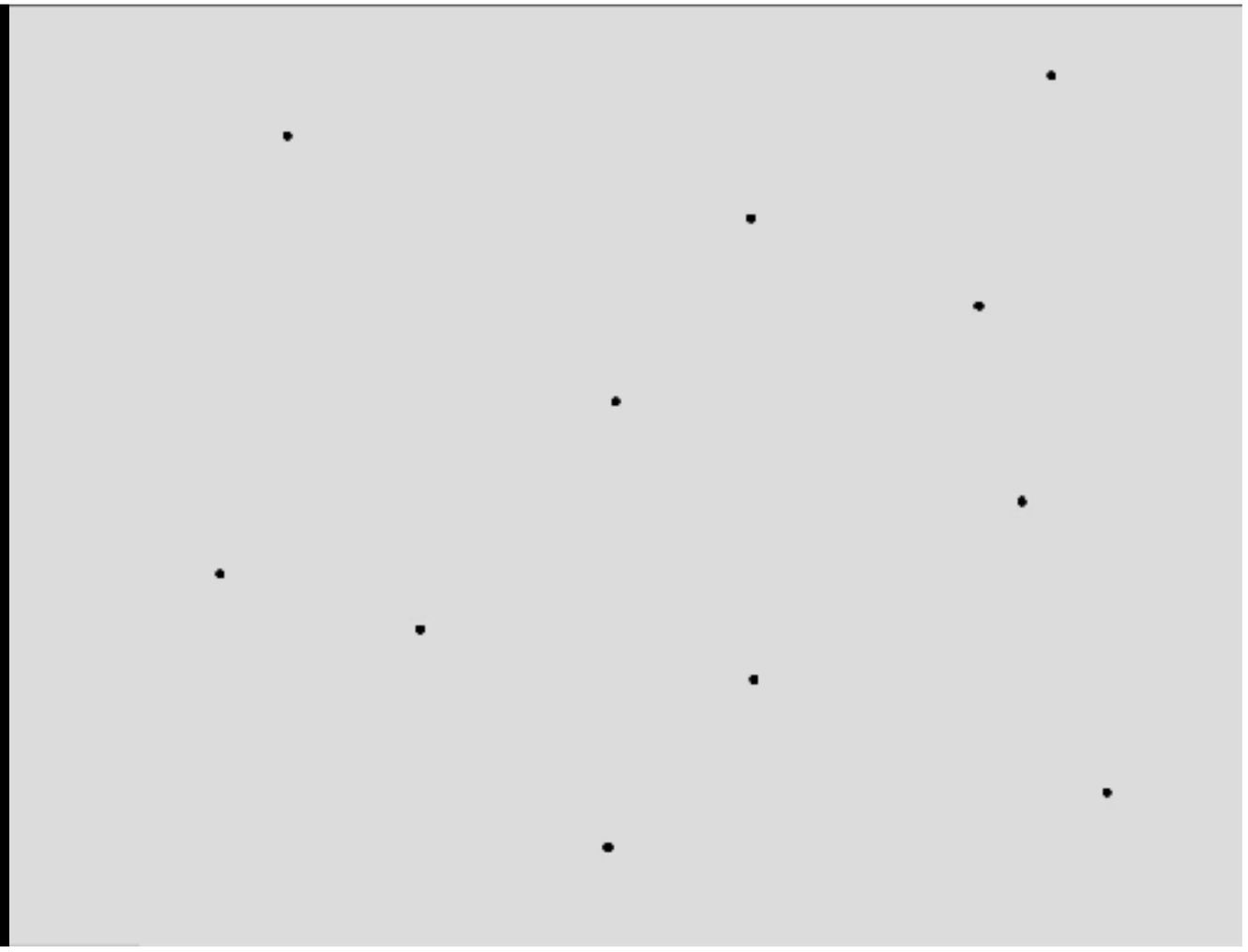
Corollary 4.16

Intersection points of adjacent arcs lie on Voronoi edges.

Approach:

- Plane sweep
- Compute $Vor(p)$ in guaranteed regions $\rightarrow$ draw Voronoi edges.

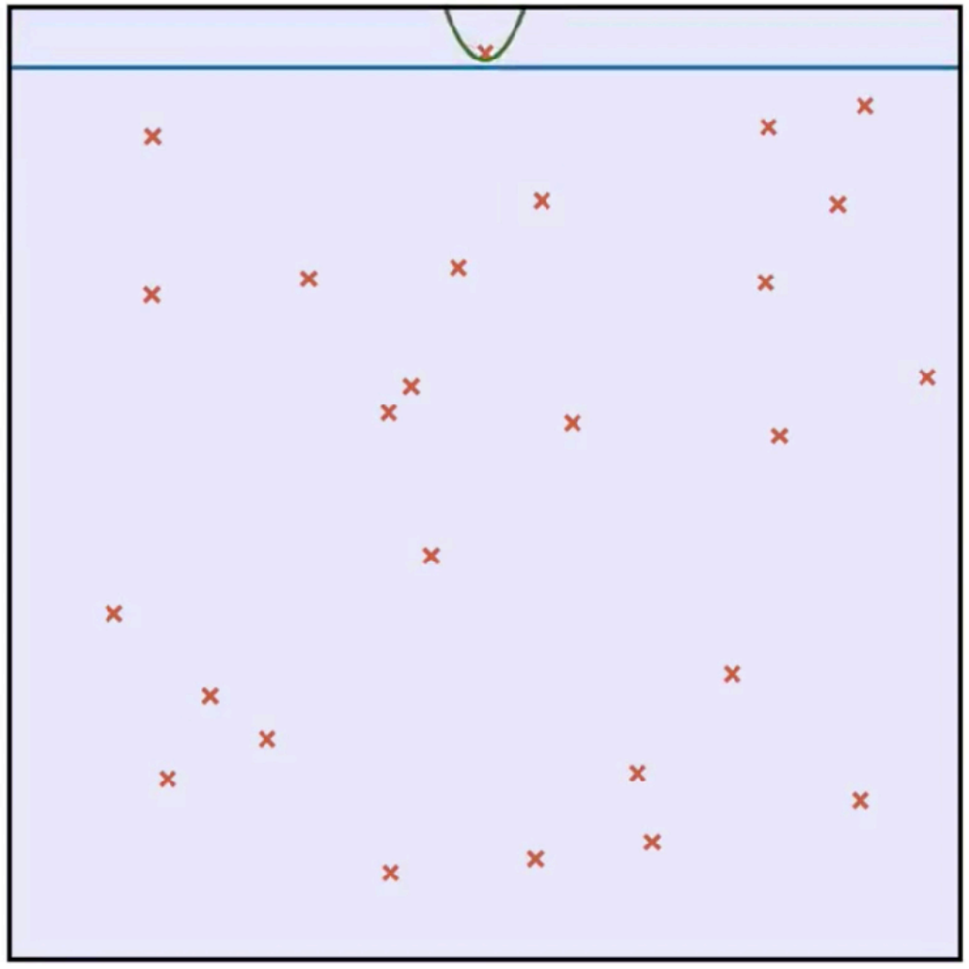


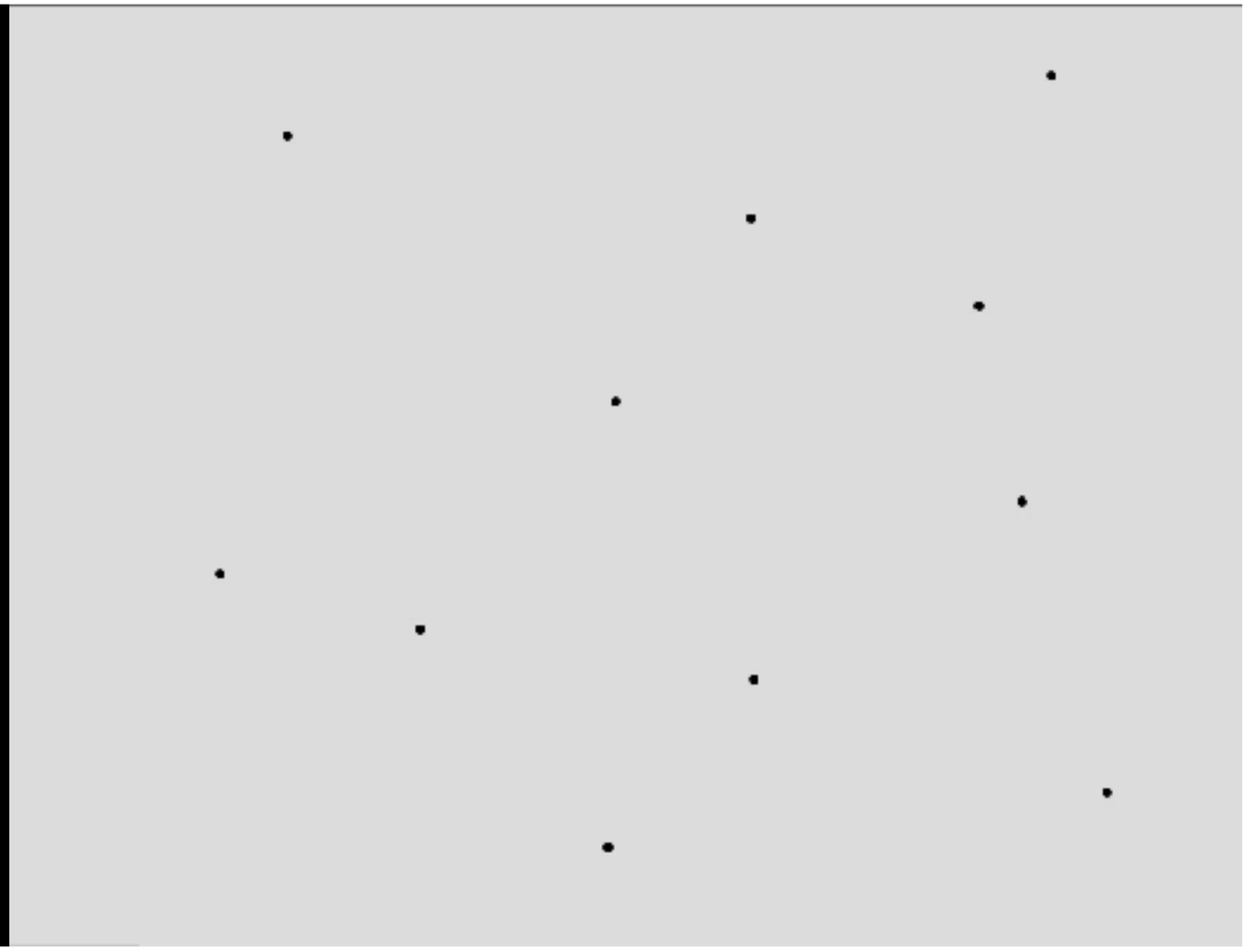


- Discretization

Approach:

- Plane sweep
- Compute $Vor(p)$ in guaranteed regions $\rightarrow$ draw Voronoi edges.

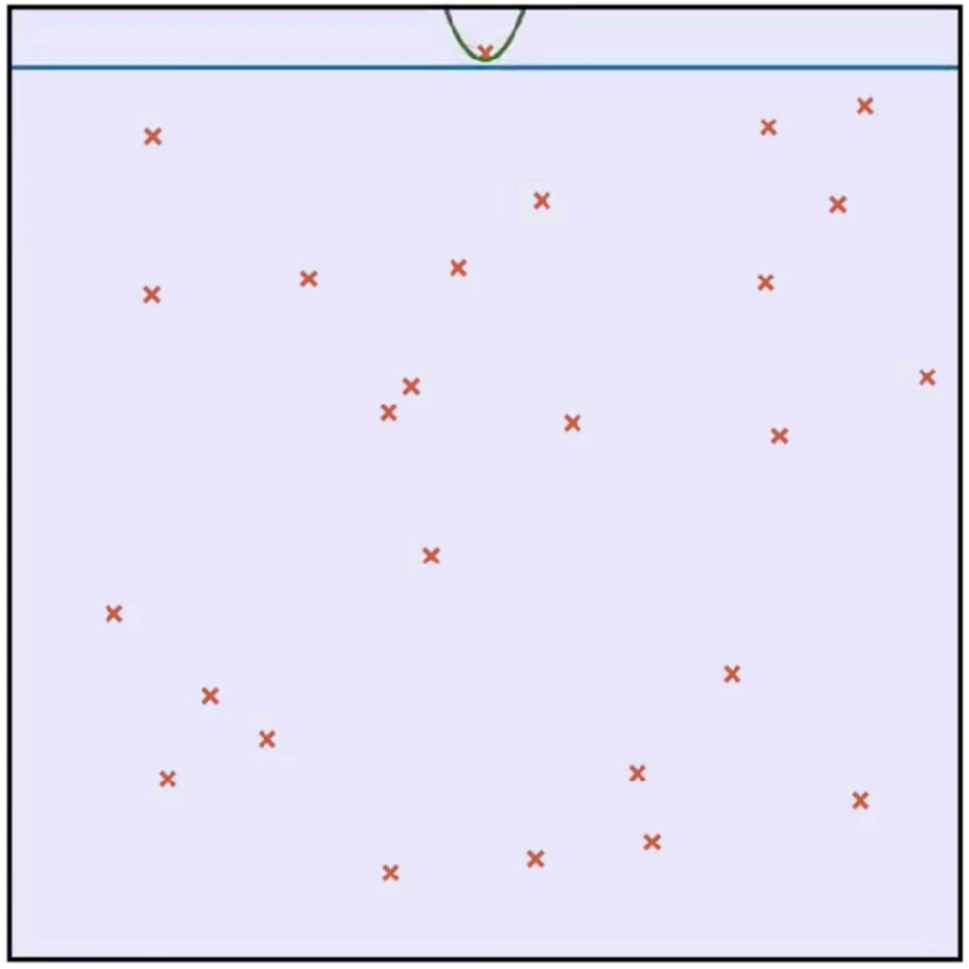


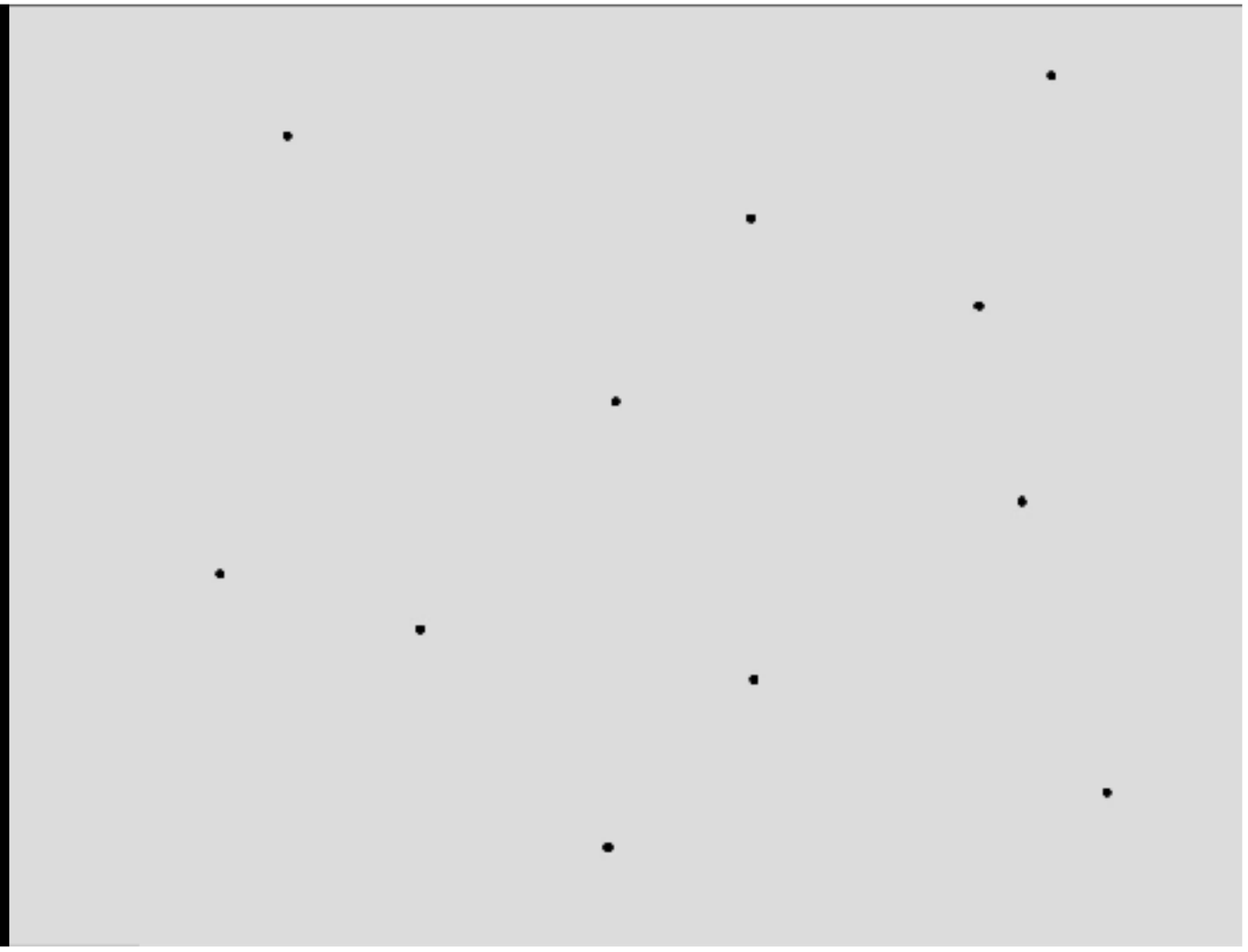


- Discretization

Approach:

- Plane sweep
- Compute $Vor(p)$ in guaranteed regions $\rightarrow$ draw Voronoi edges.





- Discretization

Point events:

- Sweep line ℓ reaches $p \in \mathcal{P}$

Issue

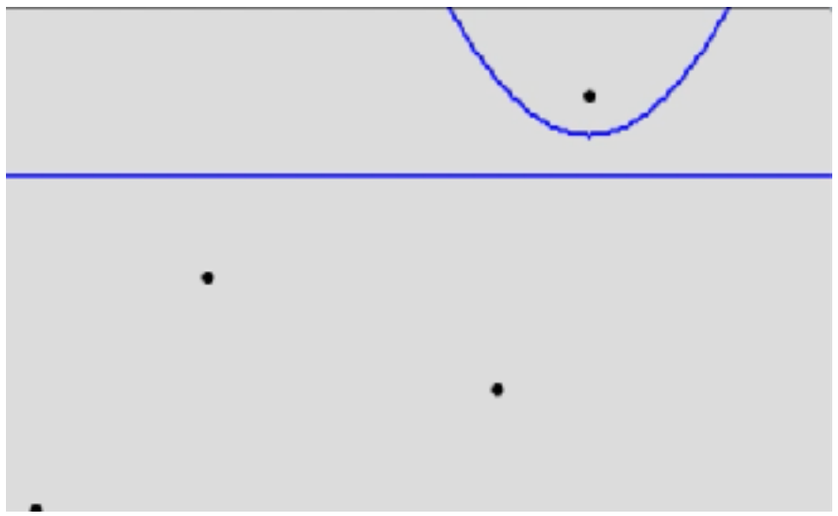
- Complexity of beach line?

Lemma 4.17

New parabolic arcs can only occur at point events.

Point events:

- Sweep line ℓ reaches $p \in \mathcal{P}$



Issue

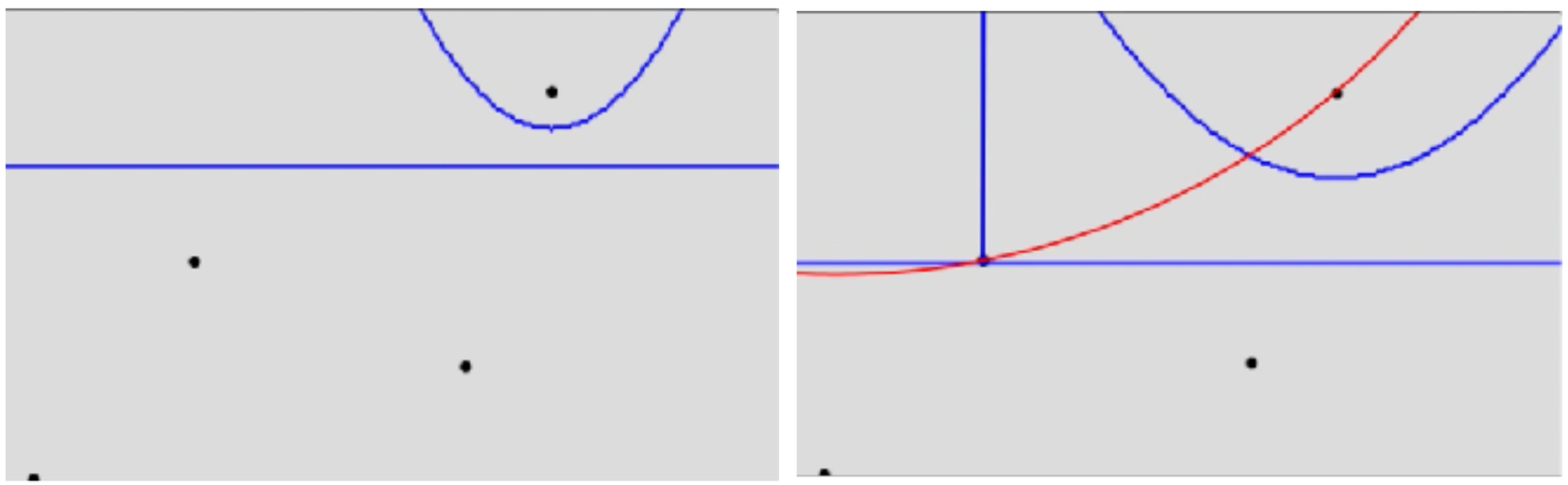
- Complexity of beach line?

Lemma 4.17

New parabolic arcs can only occur at point events.

Point events:

- Sweep line ℓ reaches $p \in \mathcal{P}$



Issue

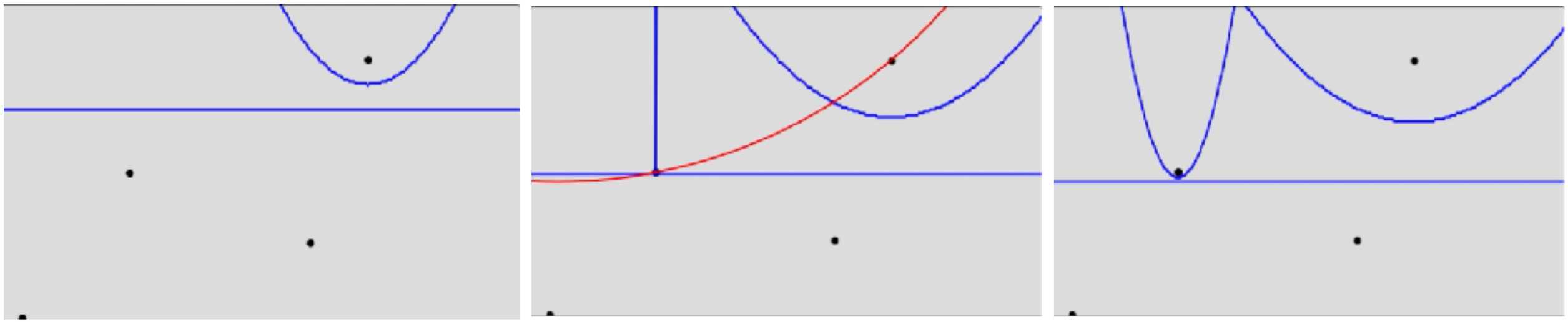
- Complexity of beach line?

Lemma 4.17

New parabolic arcs can only occur at point events.

Point events:

- Sweep line ℓ reaches $p \in \mathcal{P}$



Issue

- Complexity of beach line?

Lemma 4.17

New parabolic arcs can only occur at point events.

Lemma 4.17

New parabolic arcs can only occur at point events.

Lemma 4.17

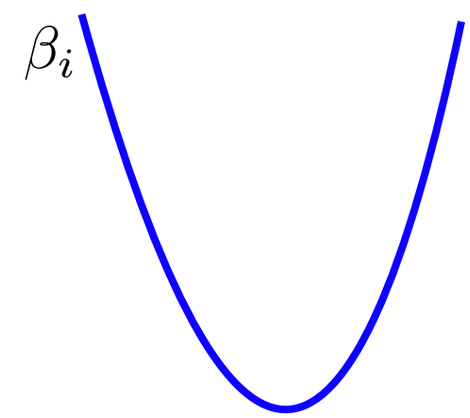
New parabolic arcs can only occur at point events.

Proof:

Lemma 4.17

New parabolic arcs can only occur at point events.

Proof:

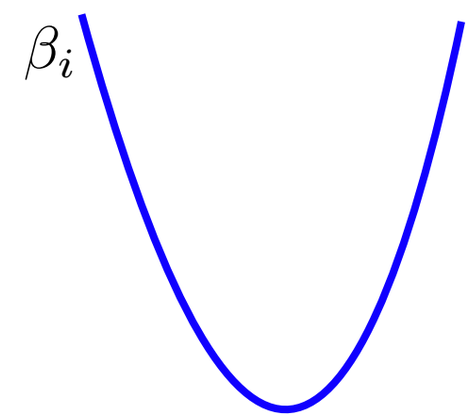


Lemma 4.17

New parabolic arcs can only occur at point events.

Proof:

- In general, parabolas correspond to sites, so:
  $\rightarrow$ Can an existing arc β_j (defined by $p_j \in \mathcal{P}$) pierce through the beach line?

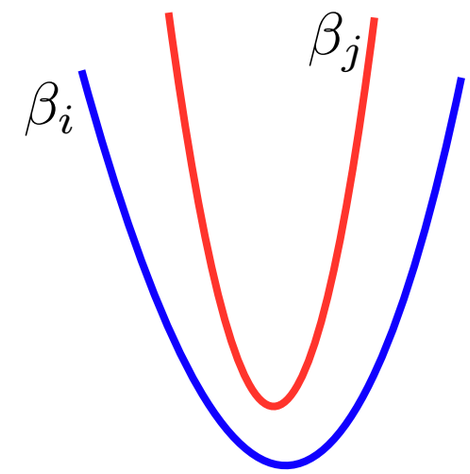


Lemma 4.17

New parabolic arcs can only occur at point events.

Proof:

- In general, parabolas correspond to sites, so:
  $\rightarrow$ Can an existing arc β_j (defined by $p_j \in \mathcal{P}$) pierce through the beach line?

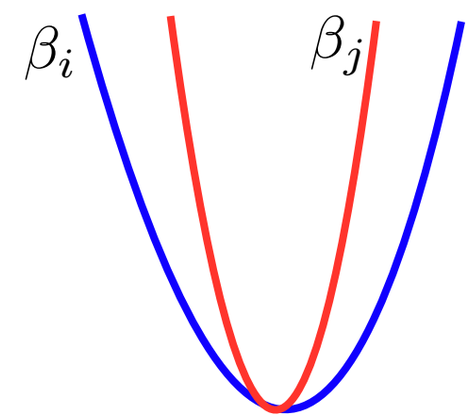


Lemma 4.17

New parabolic arcs can only occur at point events.

Proof:

- In general, parabolas correspond to sites, so:
  $\rightarrow$ Can an existing arc β_j (defined by $p_j \in \mathcal{P}$) pierce through the beach line?



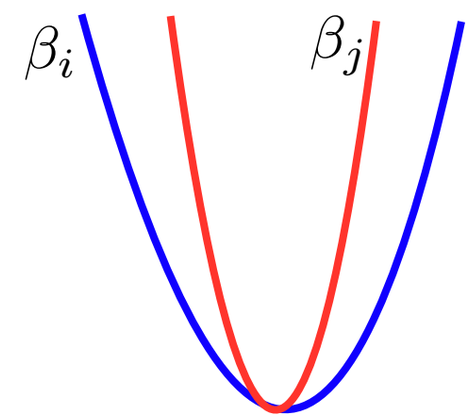
Lemma 4.17

New parabolic arcs can only occur at point events.

Proof:

- In general, parabolas correspond to sites, so:
  $\rightarrow$ Can an existing arc β_j (defined by $p_j \in \mathcal{P}$) pierce through the beach line?

- Option 1: β_j pierces β_i (defined by $p_i \in \mathcal{P}$).
  $\Rightarrow \forall x \in \mathbb{R} : \beta_i(x) \leq \beta_j(x) (\star)$



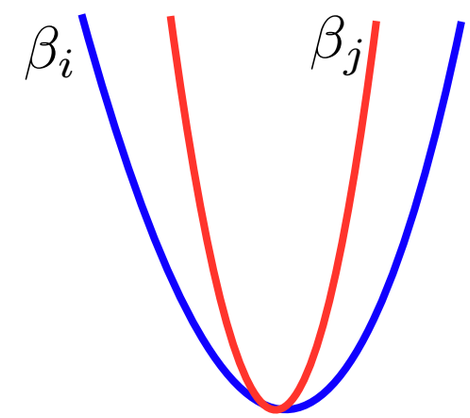
Lemma 4.17

New parabolic arcs can only occur at point events.

Proof:

- In general, parabolas correspond to sites, so:
  $\rightarrow$ Can an existing arc β_j (defined by $p_j \in \mathcal{P}$) pierce through the beach line?

- Option 1: β_j pierces β_i (defined by $p_i \in \mathcal{P}$).
  $\Rightarrow \forall x \in \mathbb{R} : \beta_i(x) \leq \beta_j(x) (\star)$
- $\ell.y$:= y-coordinate of ℓ at „piercing event“.
  $\Rightarrow$ At moment $\ell.y$: One joint point



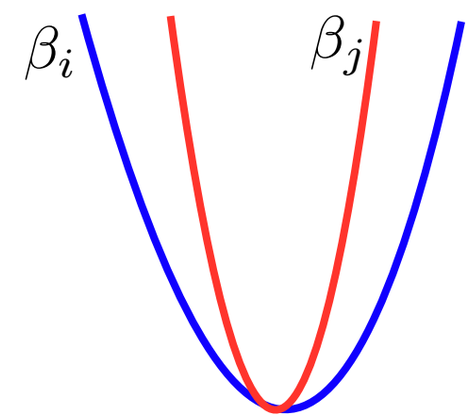
Lemma 4.17

New parabolic arcs can only occur at point events.

Proof:

- In general, parabolas correspond to sites, so:
  $\rightarrow$ Can an existing arc β_j (defined by $p_j \in \mathcal{P}$) pierce through the beach line?

- Option 1: β_j pierces β_i (defined by $p_i \in \mathcal{P}$).
  $\Rightarrow \forall x \in \mathbb{R} : \beta_i(x) \leq \beta_j(x) (\star)$



- $\ell.y$:= y-coordinate of ℓ at „piercing event“.
  $\Rightarrow$ At moment $\ell.y$: One joint point

- $p_i.y \neq p_j.y$, otherwise
  contradiction to $(\star)$

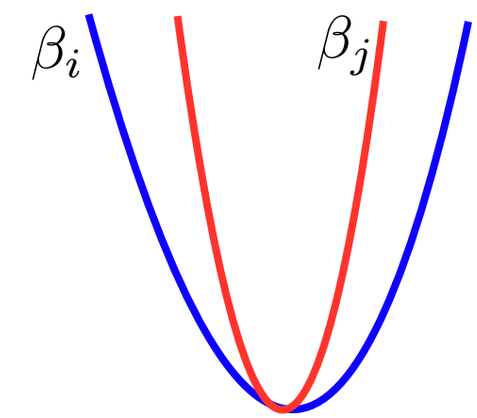
Lemma 4.17

New parabolic arcs can only occur at point events.

Proof:

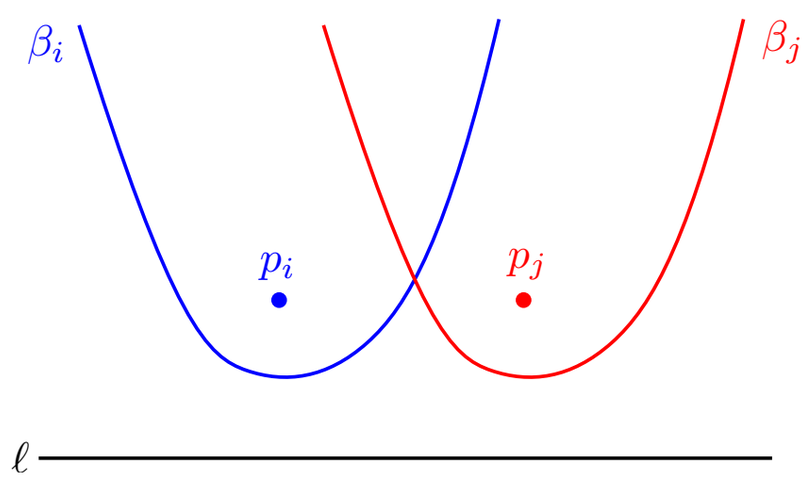
- In general, parabolas correspond to sites, so:
  $\rightarrow$ Can an existing arc β_j (defined by $p_j \in \mathcal{P}$) pierce through the beach line?

- Option 1: β_j pierces β_i (defined by $p_i \in \mathcal{P}$).
  $\Rightarrow \forall x \in \mathbb{R} : \beta_i(x) \leq \beta_j(x) (\star)$



- $\ell.y$:= y-coordinate of ℓ at „piercing event“.
  $\Rightarrow$ At moment $\ell.y$: One joint point

- $p_i.y \neq p_j.y$, otherwise contradiction to $(\star)$



Lemma 4.17

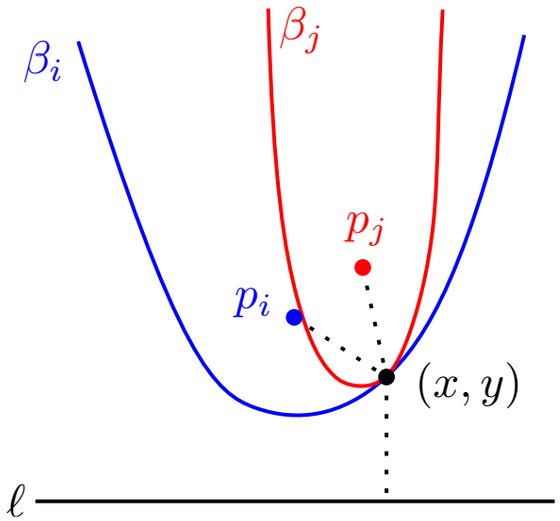
New parabolic arcs on the beach line can only occur by point events.

Proof (cont.):

Lemma 4.17

New parabolic arcs on the beach line can only occur by point events.

Proof (cont.):



Lemma 4.17

New parabolic arcs on the beach line can only occur by point events.

Proof (cont.):

- Equation for β_j:

$$(\ell.y - y)^2 = (p_j.x - x)^2 + (p_j.x - y)^2$$

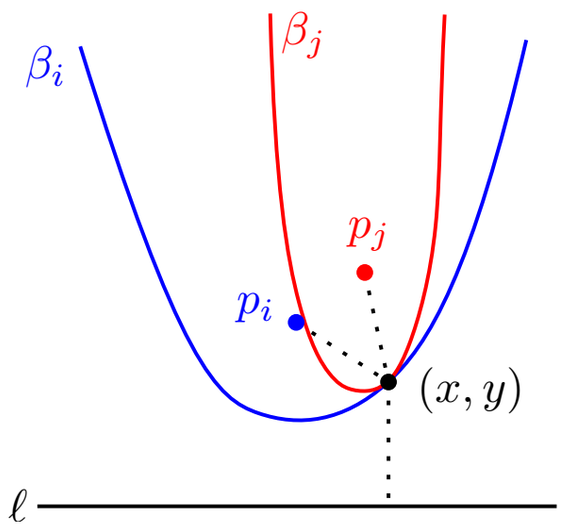
$$\Leftrightarrow \ell.y^2 - 2 \cdot \ell.y \cdot y + y^2$$

$$= p_j.x^2 - 2 \cdot x \cdot p_j.x + x^2 + p_j.y^2 - 2 \cdot y \cdot p_j.y + y^2$$

$$\Leftrightarrow y \cdot 2\,(p_j.y - \ell.y)$$

$$= x^2 - 2 \cdot x \cdot p_j.x + p_j.x^2 + p_j.y^2 - \ell.y^2$$

$$\Leftrightarrow y = \frac{x^2 - 2 \cdot x \cdot p_j.x + p_j.x^2 + p_j.y^2 - \ell.y^2}{2\,(p_j.y - \ell.y)}$$



Lemma 4.17

New parabolic arcs on the beach line can only occur by point events.

Proof (cont.):

- Equation for β_j:

$$(\ell.y - y)^2 = (p_j.x - x)^2 + (p_j.x - y)^2$$

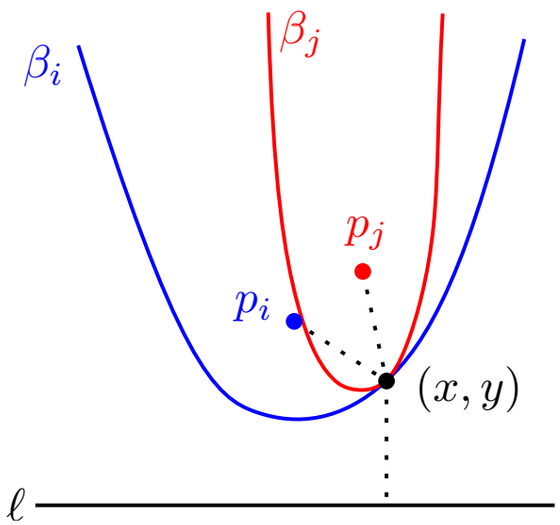
$$\Leftrightarrow \ell.y^2 - 2 \cdot \ell.y \cdot y + y^2$$

$$= p_j.x^2 - 2 \cdot x \cdot p_j.x + x^2 + p_j.y^2 - 2 \cdot y \cdot p_j.y + y^2$$

$$\Leftrightarrow y \cdot 2\,(p_j.y - \ell.y)$$

$$= x^2 - 2 \cdot x \cdot p_j.x + p_j.x^2 + p_j.y^2 - \ell.y^2$$

$$\Leftrightarrow y = \frac{x^2 - 2 \cdot x \cdot p_j.x + p_j.x^2 + p_j.y^2 - \ell.y^2}{2\,(p_j.y - \ell.y)}$$



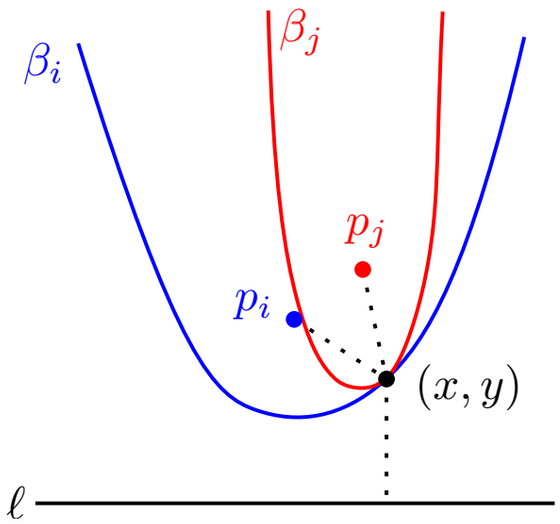
- Analogously for β_i

Lemma 4.17

New parabolic arcs on the beach line can only occur by point events.

Proof:

Analogously for β_i

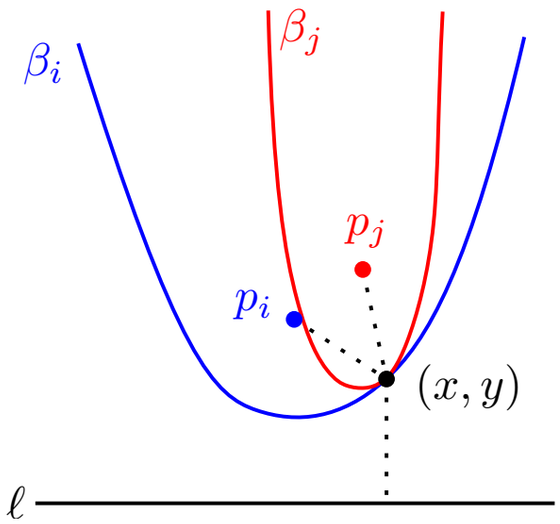


Lemma 4.17

New parabolic arcs on the beach line can only occur by point events.

Proof:

- Analogously for β_i



Lemma 4.17

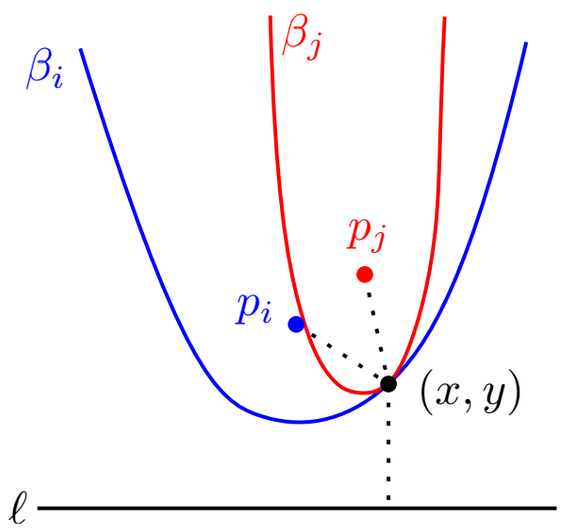
New parabolic arcs on the beach line can only occur by point events.

Proof:

- Analogously for β_i
- Equate:

$$\Rightarrow \quad \frac{x^2 - 2 \cdot x \cdot p_j.x + p_j.x^2 + p_j.y^2 - \ell.y^2}{2\,(p_j.y - \ell.y)}$$

$$= \frac{x^2 - 2 \cdot x \cdot p_i.x + p_i.x^2 + p_i.y^2 - \ell.y^2}{2\,(p_i.y - \ell.y)}$$



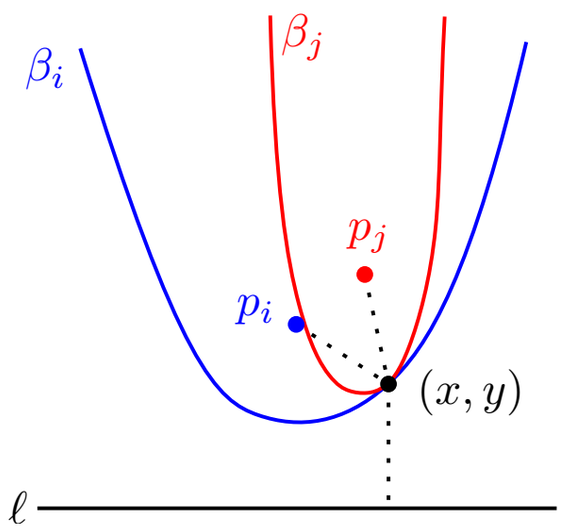
Lemma 4.17

New parabolic arcs on the beach line can only occur by point events.

Proof:

- Analogously for β_i
- Equate:

$$\Rightarrow \quad \frac{x^2 - 2 \cdot x \cdot p_j.x + p_j.x^2 + p_j.y^2 - \ell.y^2}{2\,(p_j.y - \ell.y)} = \frac{x^2 - 2 \cdot x \cdot p_i.x + p_i.x^2 + p_i.y^2 - \ell.y^2}{2\,(p_i.y - \ell.y)}$$



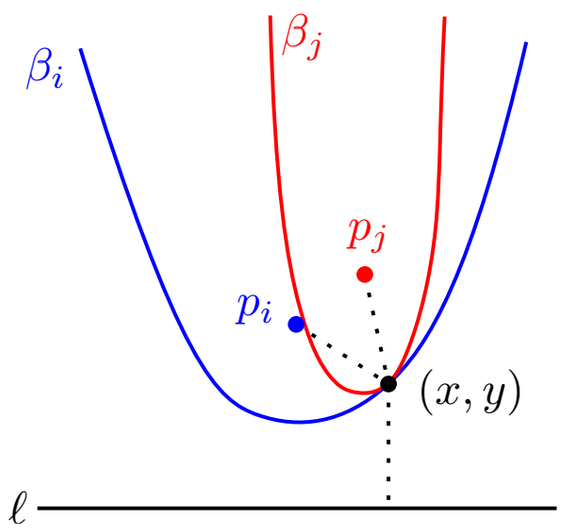
- $p_j.y \neq p_i.y$ and $p_i.y, p_j.y > \ell.y$

  $\Rightarrow \exists c_1, c_2 \in \mathbb{R} : 1 \cdot x^2 + c_1 \cdot x + c_2 = 0$ with $c_2 \neq 0$

Lemma 4.17

New parabolic arcs on the beach line can only occur by point events.

Proof:



- Analogously for β_i
- Equate:

$$\Rightarrow \quad \frac{x^2 - 2 \cdot x \cdot p_j.x + p_j.x^2 + p_j.y^2 - \ell.y^2}{2\,(p_j.y - \ell.y)} = \frac{x^2 - 2 \cdot x \cdot p_i.x + p_i.x^2 + p_i.y^2 - \ell.y^2}{2\,(p_i.y - \ell.y)}$$

- $p_j.y \neq p_i.y$ and $p_i.y, p_j.y > \ell.y$

  $\Rightarrow \exists c_1, c_2 \in \mathbb{R} : 1 \cdot x^2 + c_1 \cdot x + c_2 = 0$ with $c_2 \neq 0$

  $\Rightarrow$ Two intersection points ↯

Lemma 4.17

New parabolic arcs on the beach line can only occur by point events.

Proof:

Lemma 4.17

New parabolic arcs on the beach line can only occur by point events.

Proof:

- In general, parabolas correspond to sites, so:
  $\rightarrow$ Can an existing arc β_j (defined by $p_j \in \mathcal{P}$) pierce through the beach line?

Lemma 4.17

New parabolic arcs on the beach line can only occur by point events.

Proof:

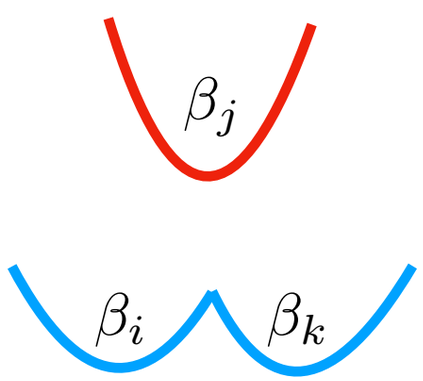
- In general, parabolas correspond to sites, so:
  $\rightarrow$ Can an existing arc β_j (defined by $p_j \in \mathcal{P}$) pierce through the beach line?
- Option 2: β_j pierces intersection point q of β_i, β_k (defined by $p_i, p_k \in \mathcal{P}$).

Lemma 4.17

New parabolic arcs on the beach line can only occur by point events.

Proof:

- In general, parabolas correspond to sites, so:
  $\rightarrow$ Can an existing arc β_j (defined by $p_j \in \mathcal{P}$) pierce through the beach line?
- Option 2: β_j pierces intersection point q of β_i, β_k (defined by $p_i, p_k \in \mathcal{P}$).

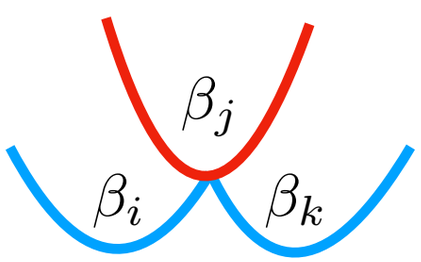


Lemma 4.17

New parabolic arcs on the beach line can only occur by point events.

Proof:

- In general, parabolas correspond to sites, so:
  $\rightarrow$ Can an existing arc β_j (defined by $p_j \in \mathcal{P}$) pierce through the beach line?
- Option 2: β_j pierces intersection point q of β_i, β_k (defined by $p_i, p_k \in \mathcal{P}$).

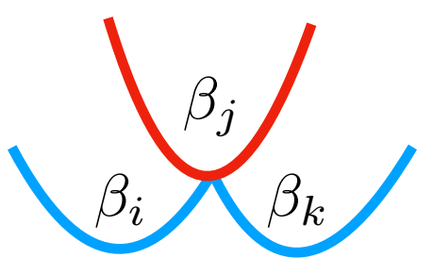


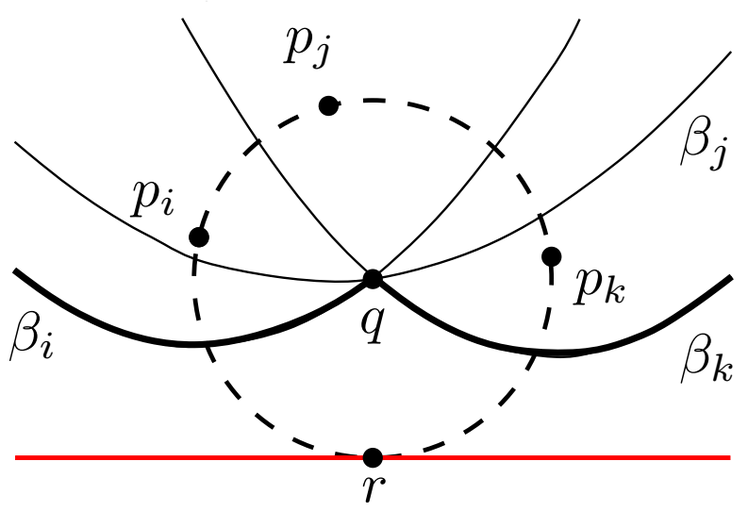
Lemma 4.17

New parabolic arcs on the beach line can only occur by point events.

Proof:

- In general, parabolas correspond to sites, so:
  $\rightarrow$ Can an existing arc β_j (defined by $p_j \in \mathcal{P}$) pierce through the beach line?
- Option 2: β_j pierces intersection point q of β_i, β_k (defined by $p_i, p_k \in \mathcal{P}$).



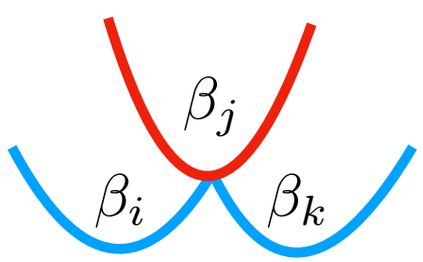


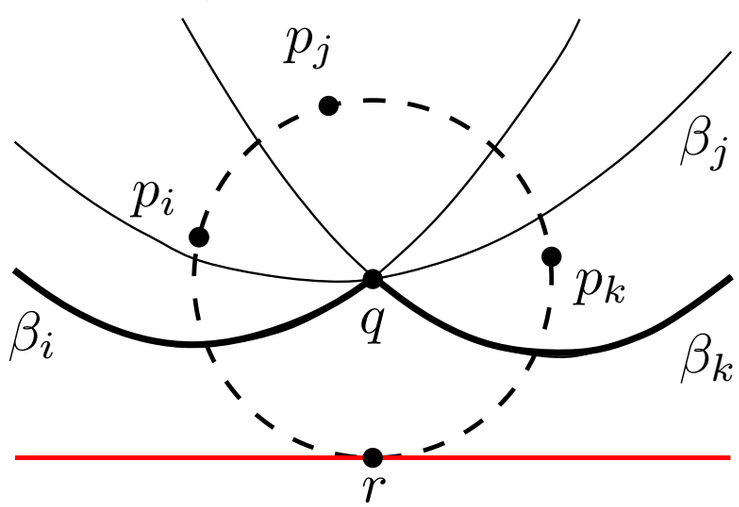
Lemma 4.17

New parabolic arcs on the beach line can only occur by point events.

Proof:

- In general, parabolas correspond to sites, so:
  $\rightarrow$ Can an existing arc β_j (defined by $p_j \in \mathcal{P}$) pierce through the beach line?
- Option 2: β_j pierces intersection point q of β_i, β_k (defined by $p_i, p_k \in \mathcal{P}$).
- $d(q, p_i) = d(q, p_j) = d(q, p_k) = d(q, r)$



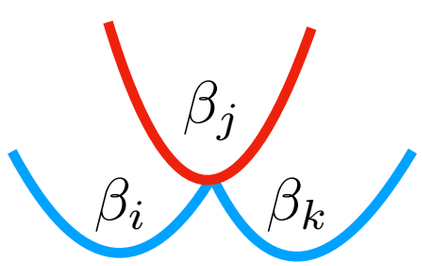


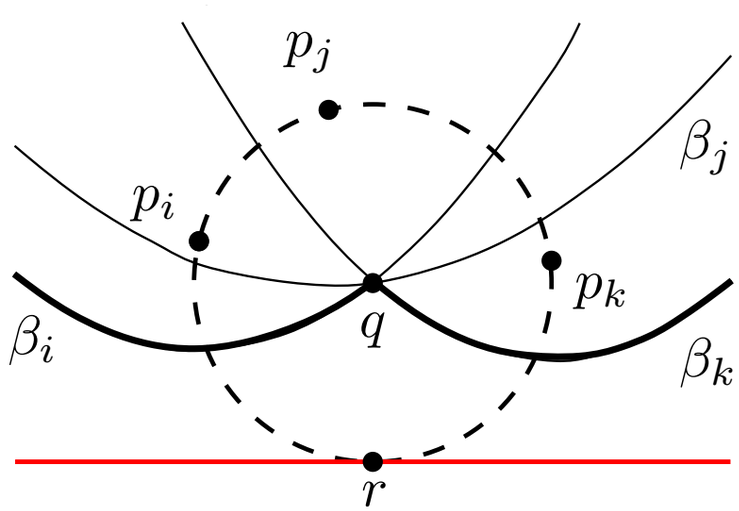
Lemma 4.17

New parabolic arcs on the beach line can only occur by point events.

Proof:

- In general, parabolas correspond to sites, so:
  $\rightarrow$ Can an existing arc β_j (defined by $p_j \in \mathcal{P}$) pierce through the beach line?
- Option 2: β_j pierces intersection point q of β_i, β_k (defined by $p_i, p_k \in \mathcal{P}$).
- $d(q, p_i) = d(q, p_j) = d(q, p_k) = d(q, r)$
- Infinitesimal perturbation of ℓ $\rightarrow$ circle C with center point $u := \beta_i \cap \beta_j$ that touches ℓ.



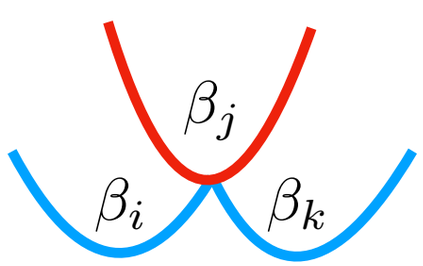


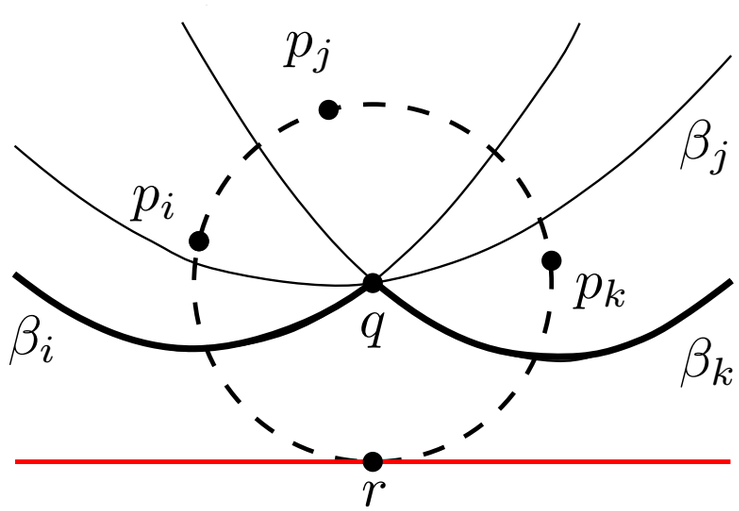
Lemma 4.17

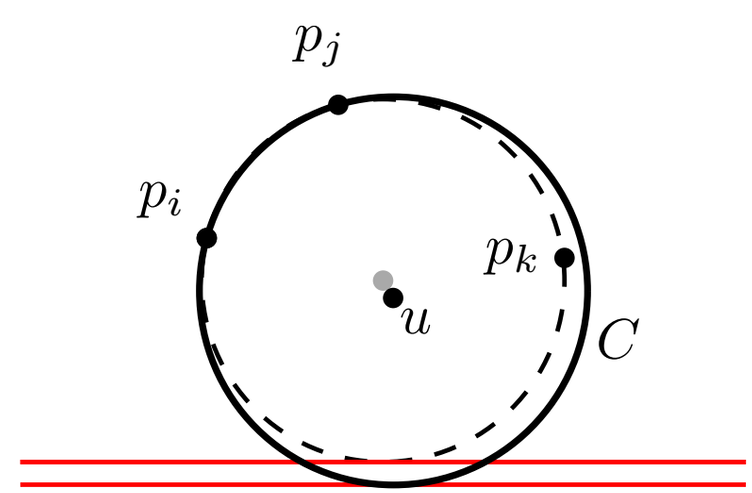
New parabolic arcs on the beach line can only occur by point events.

Proof:

- In general, parabolas correspond to sites, so:
  $\rightarrow$ Can an existing arc β_j (defined by $p_j \in \mathcal{P}$) pierce through the beach line?
- Option 2: β_j pierces intersection point q of β_i, β_k (defined by $p_i, p_k \in \mathcal{P}$).
- $d(q, p_i) = d(q, p_j) = d(q, p_k) = d(q, r)$
- Infinitesimal perturbation of ℓ $\rightarrow$ circle C with center point $u := \beta_i \cap \beta_j$ that touches ℓ.





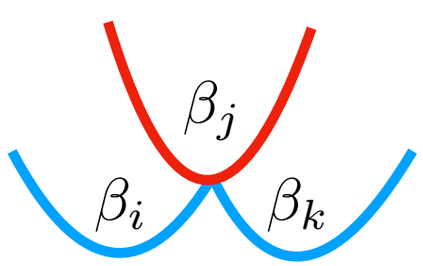


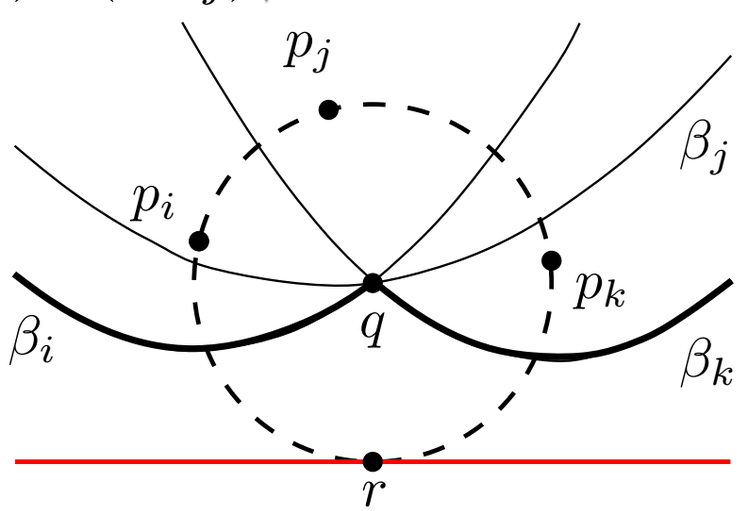
Lemma 4.17

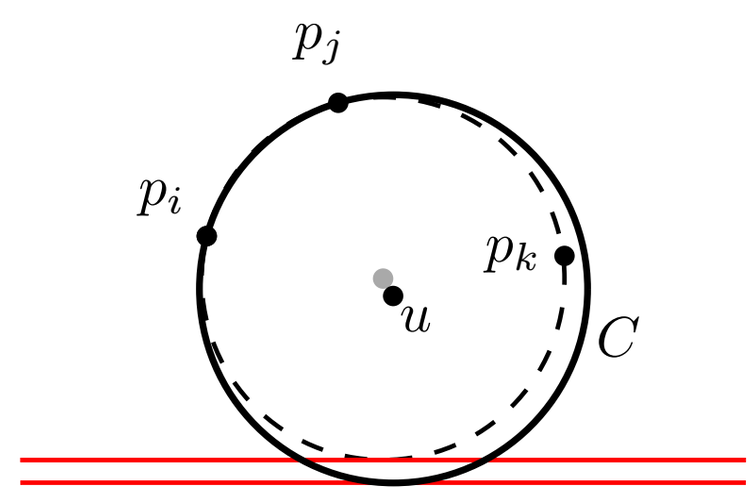
New parabolic arcs on the beach line can only occur by point events.

Proof:

- In general, parabolas correspond to sites, so:
  $\rightarrow$ Can an existing arc β_j (defined by $p_j \in \mathcal{P}$) pierce through the beach line?
- Option 2: β_j pierces intersection point q of β_i, β_k (defined by $p_i, p_k \in \mathcal{P}$).
- $d(q, p_i) = d(q, p_j) = d(q, p_k) = d(q, r)$
- Infinitesimal perturbation of ℓ $\rightarrow$ circle C with center point $u := \beta_i \cap \beta_j$ that touches ℓ.
- $p_k \in C^\circ$
  $\Rightarrow d(u, p_k) < d(u, p_i), d(q, p_j)$ ↯







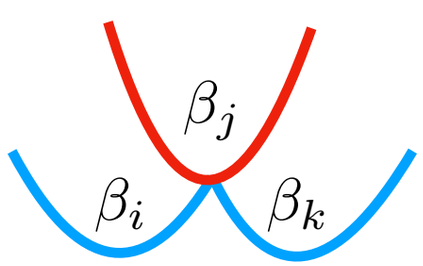
Lemma 4.17

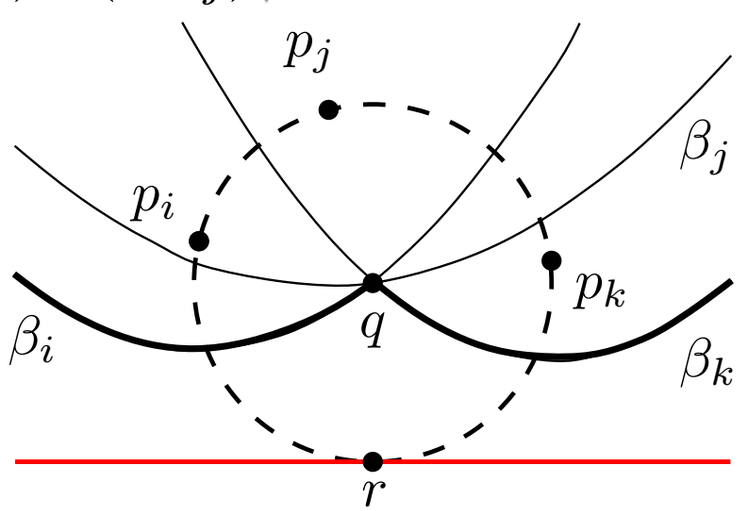
New parabolic arcs on the beach line can only occur by point events.

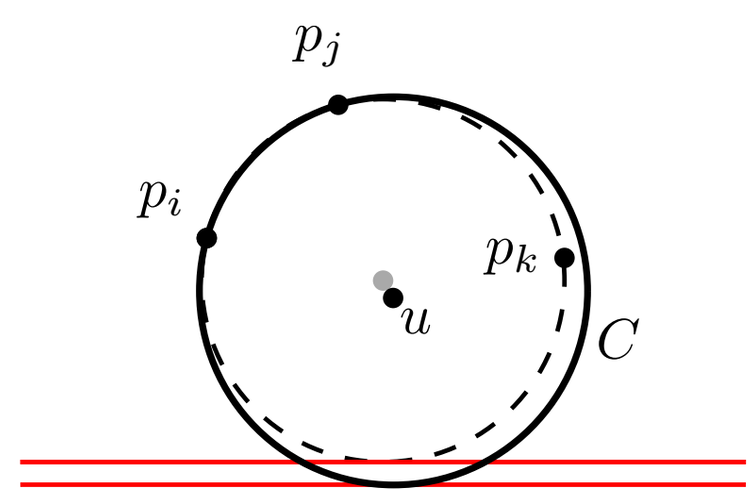
Proof:

- In general, parabolas correspond to sites, so:
  $\rightarrow$ Can an existing arc β_j (defined by $p_j \in \mathcal{P}$) pierce through the beach line?
- Option 2: β_j pierces intersection point q of β_i, β_k (defined by $p_i, p_k \in \mathcal{P}$).
- $d(q, p_i) = d(q, p_j) = d(q, p_k) = d(q, r)$
- Infinitesimal perturbation of ℓ $\rightarrow$ circle C with center point $u := \beta_i \cap \beta_j$ that touches ℓ.
- $p_k \in C^\circ$
  $\Rightarrow d(u, p_k) < d(u, p_i), d(q, p_j)$ ↯

$\square$







Complexity of the Beach Line - V

Lemma 4.18

The beach line has at most $2n - 1$ parabolic arcs.

Lemma 4.18

The beach line has at most $2n - 1$ parabolic arcs.

Proof:

Lemma 4.18

The beach line has at most $2n-1$ parabolic arcs.

Proof:

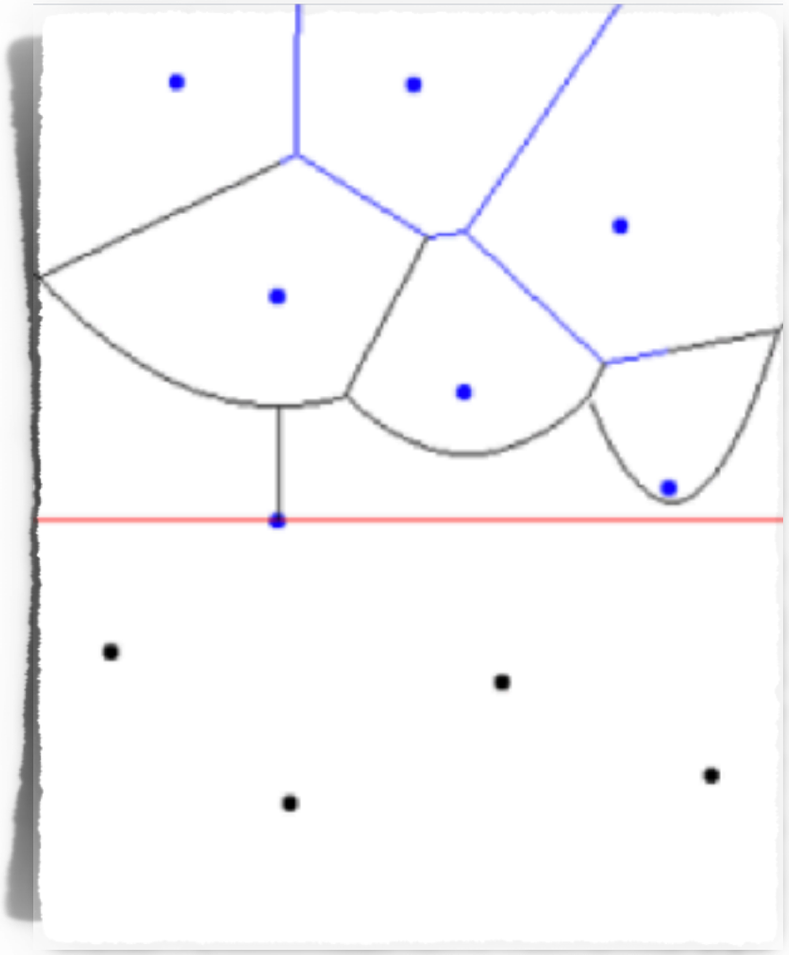
- Each $p \in \mathcal{P}$ generates a new parabolic arc.

Lemma 4.18

The beach line has at most $2n-1$ parabolic arcs.

Proof:

- Each $p \in \mathcal{P}$ generates a new parabolic arc.

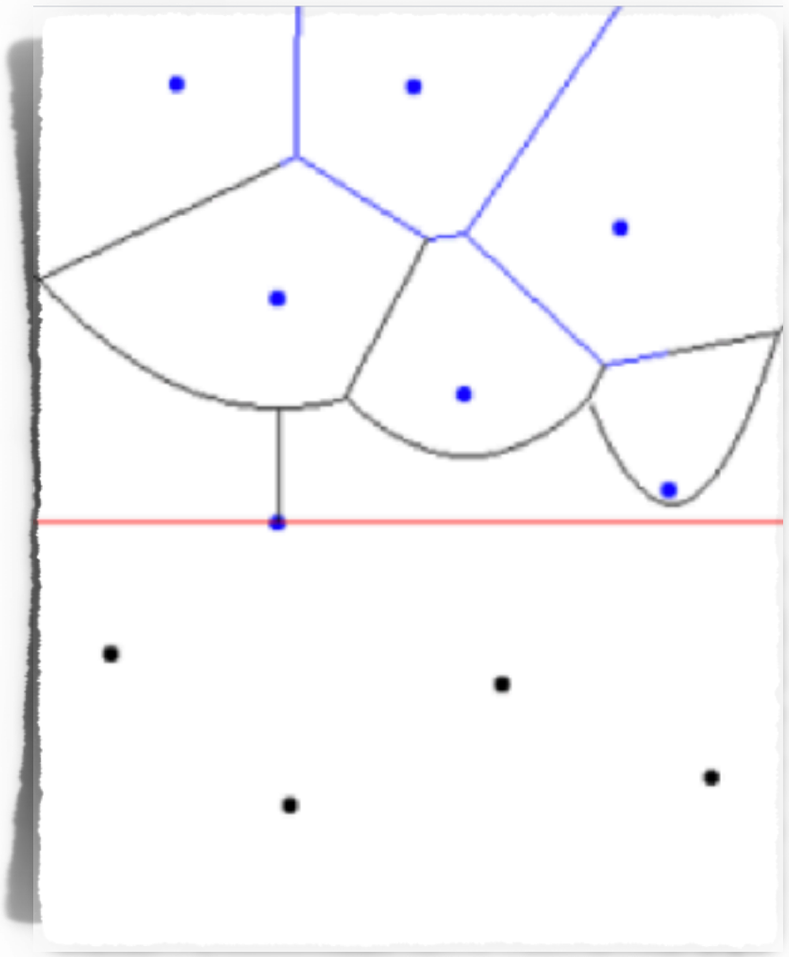


Lemma 4.18

The beach line has at most $2n-1$ parabolic arcs.

Proof:

- Each $p \in \mathcal{P}$ generates a new parabolic arc.
- Each new arc (with the exception of the first arc) can split at most one other arc into two pieces.

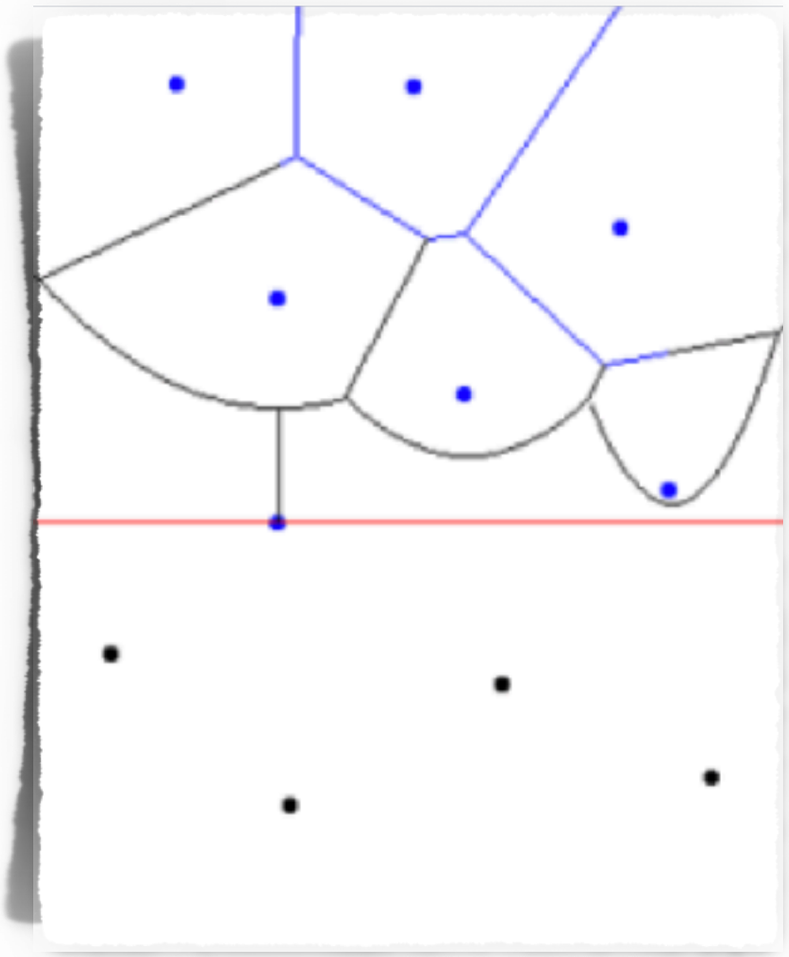


Lemma 4.18

The beach line has at most $2n-1$ parabolic arcs.

Proof:

- Each $p \in \mathcal{P}$ generates a new parabolic arc.
- Each new arc (with the exception of the first arc) can split at most one other arc into two pieces.
- Lemma 4.17: There is no other way to generate new arcs.

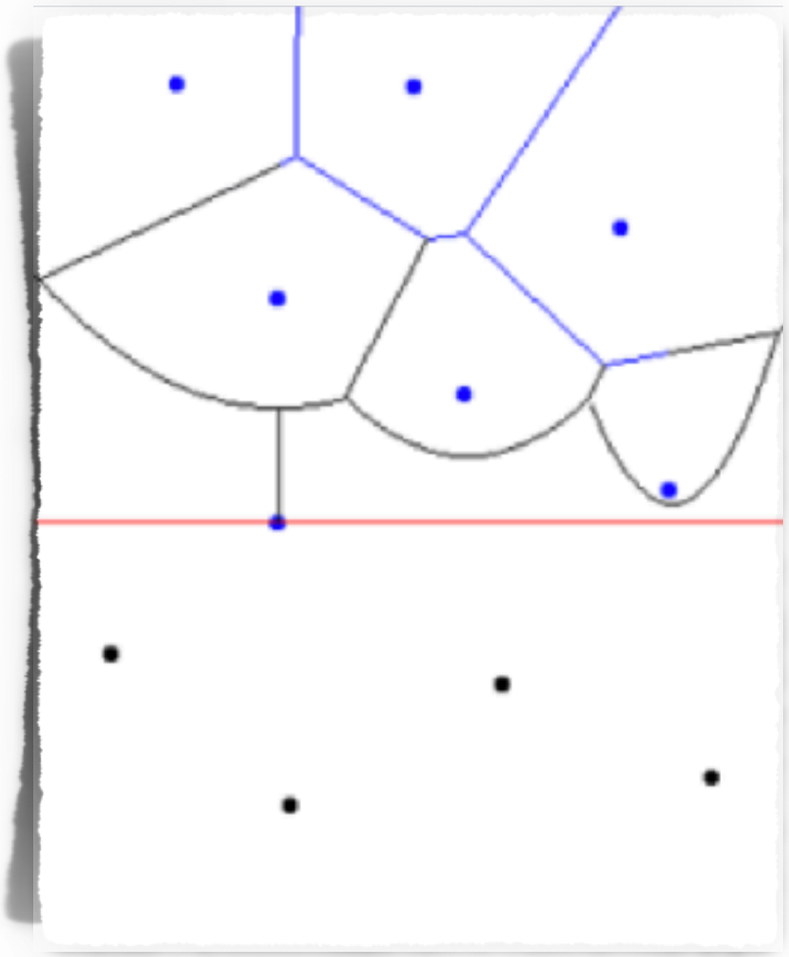


Lemma 4.18

The beach line has at most $2n-1$ parabolic arcs.

Proof:

- Each $p \in \mathcal{P}$ generates a new parabolic arc.
- Each new arc (with the exception of the first arc) can split at most one other arc into two pieces.
- Lemma 4.17: There is no other way to generate new arcs.



Lemma 4.17

New parabolic arcs on the beach line can only occur by point events.

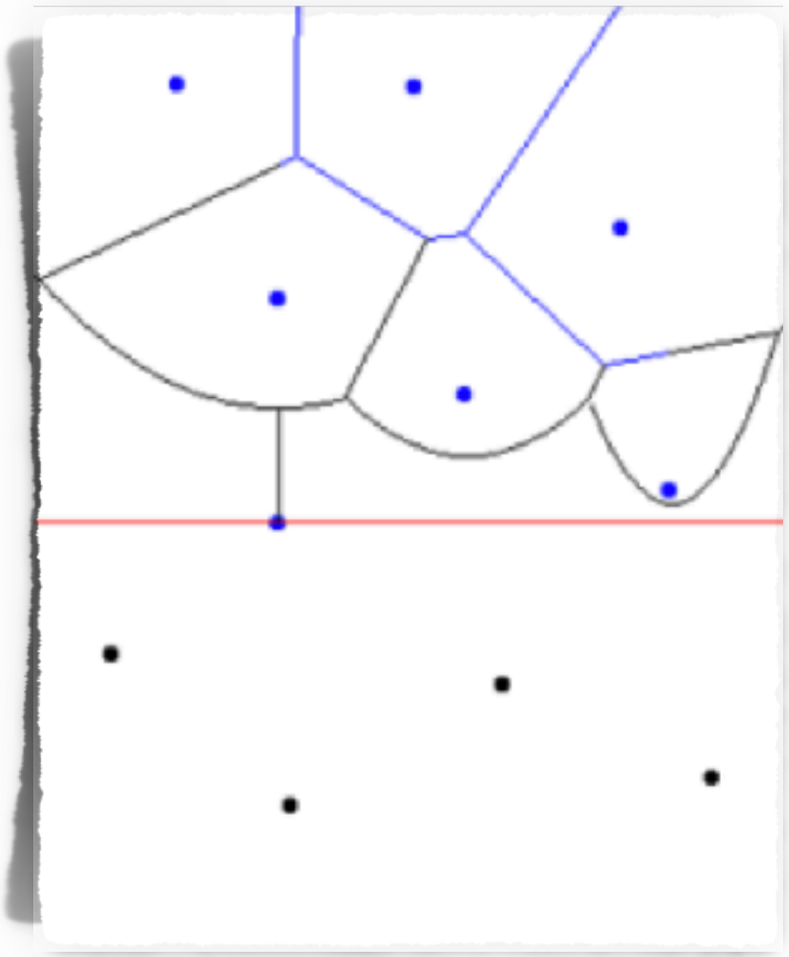
Lemma 4.18

The beach line has at most $2n-1$ parabolic arcs.

Proof:

- Each $p \in \mathcal{P}$ generates a new parabolic arc.
- Each new arc (with the exception of the first arc) can split at most one other arc into two pieces.
- Lemma 4.17: There is no other way to generate new arcs.

$\square$



Lemma 4.17

New parabolic arcs on the beach line can only occur by point events.

Implications of a Point Event

Observation:

Observation:

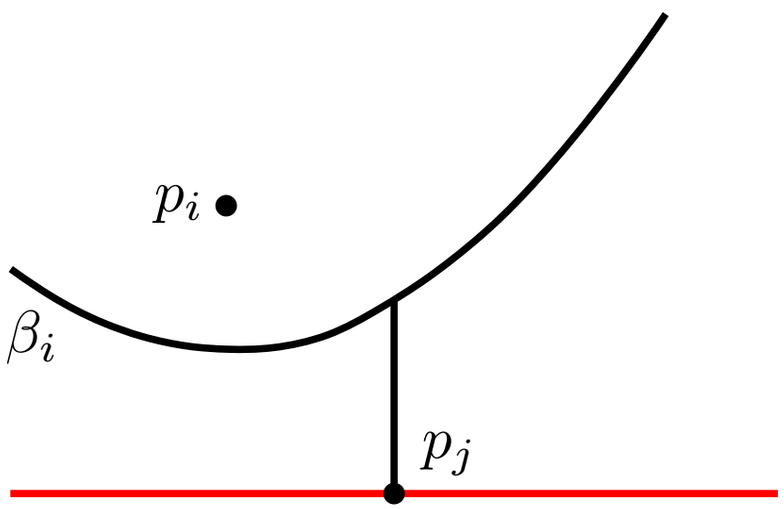
- Point event $p_j \rightarrow \geq 1$ Voronoi edges are discovered.

Observation:

- Point event $p_j \rightarrow \geq 1$ Voronoi edges are discovered.
- More precisely: Arc β_j (defined by p_j) splits β_i (defined by p_i) into $\beta_{i,1}, \beta_{i,2}$.

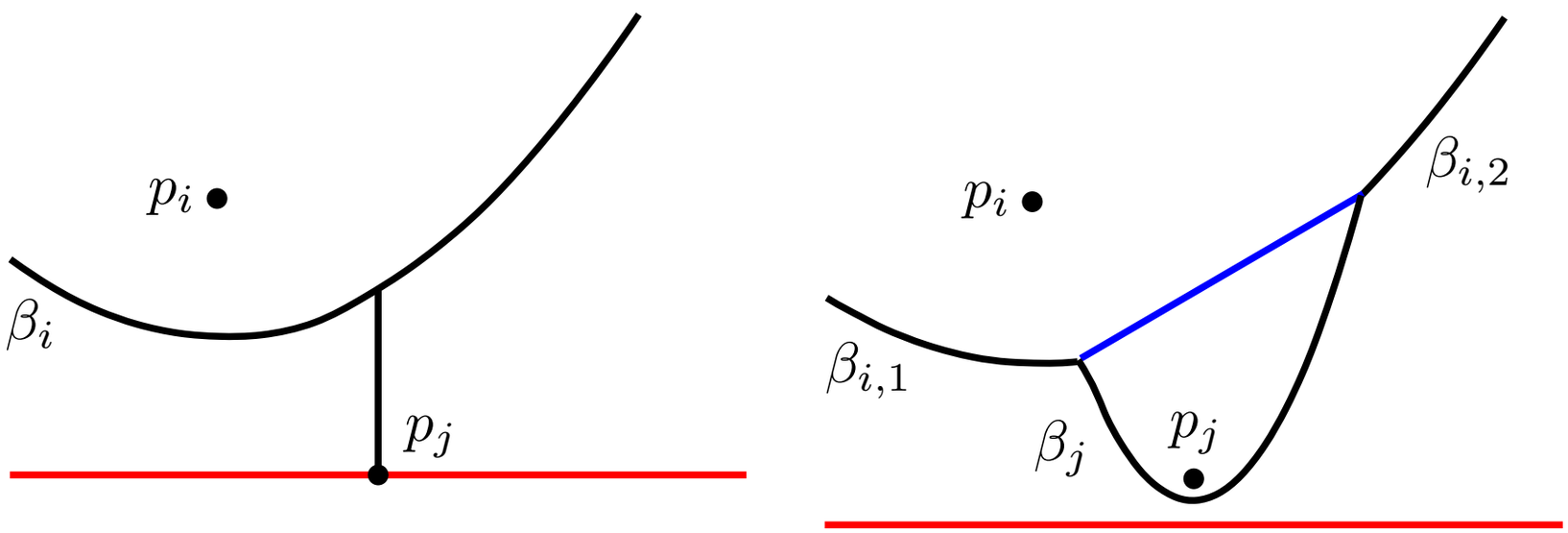
Observation:

- Point event $p_j \rightarrow \geq 1$ Voronoi edges are discovered.
- More precisely: Arc β_j (defined by p_j) splits β_i (defined by p_i) into $\beta_{i,1}, \beta_{i,2}$.



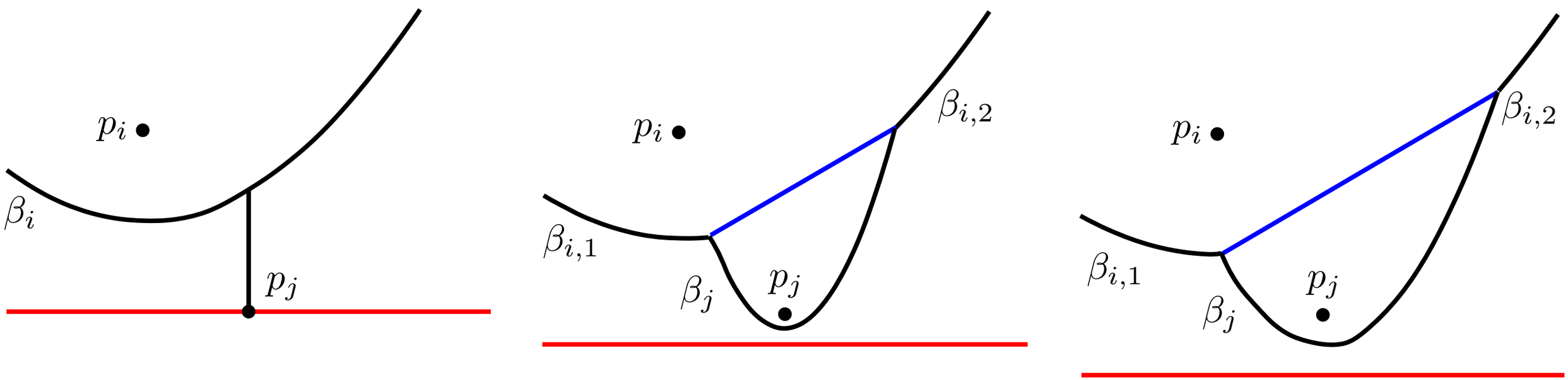
Observation:

- Point event $p_j \rightarrow \geq 1$ Voronoi edges are discovered.
- More precisely: Arc β_j (defined by p_j) splits β_i (defined by p_i) into $\beta_{i,1}, \beta_{i,2}$.



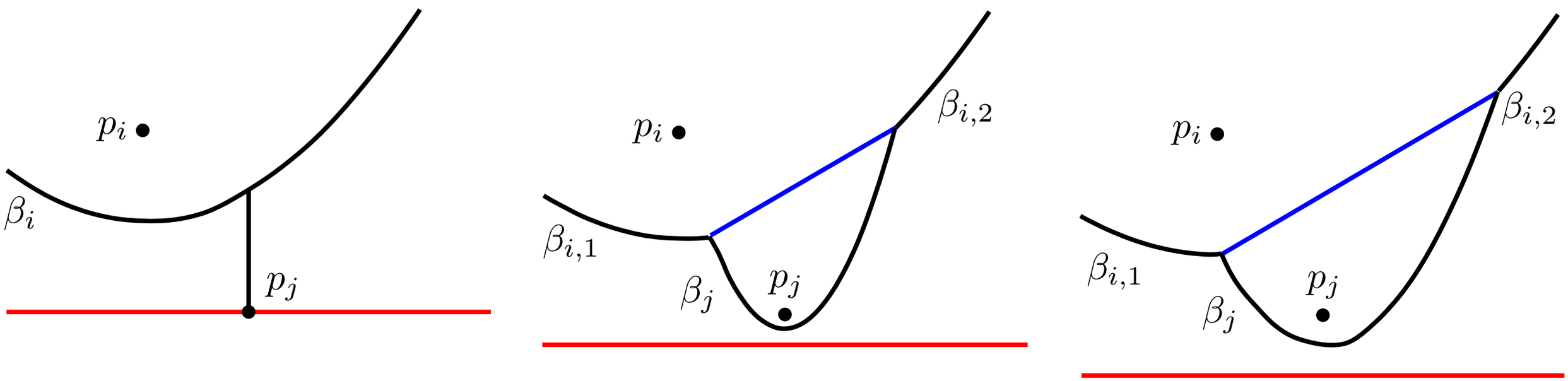
Observation:

- Point event $p_j \rightarrow \geq 1$ Voronoi edges are discovered.
- More precisely: Arc β_j (defined by p_j) splits β_i (defined by p_i) into $\beta_{i,1}, \beta_{i,2}$.



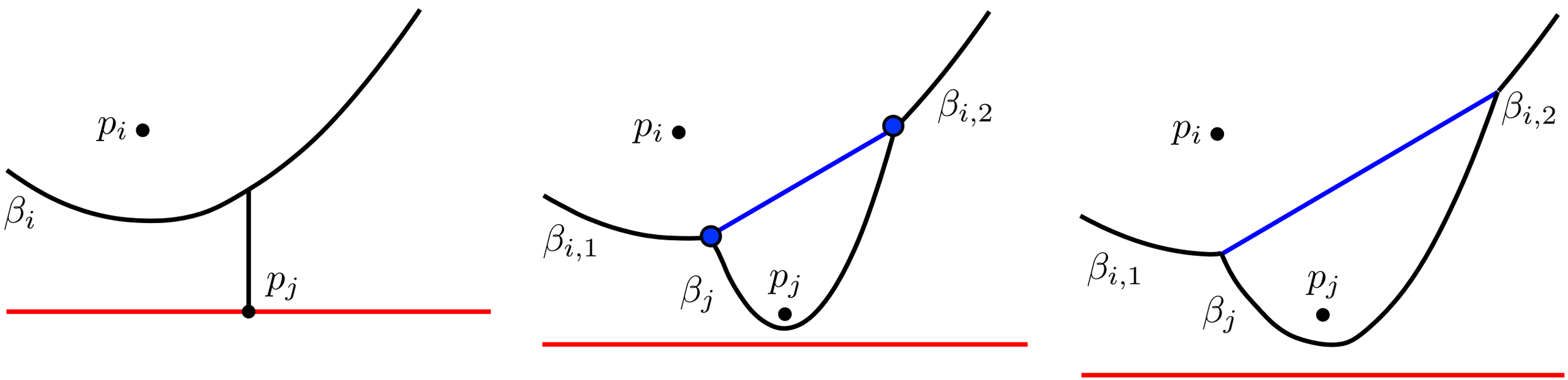
Observation:

- Point event $p_j \rightarrow$ ≥ 1 Voronoi edges are discovered.
- More precisely: Arc β_j (defined by p_j) splits β_i (defined by p_i) into $\beta_{i,1}, \beta_{i,2}$.
- Lemma 4.15/Corollary 4.16: Intersection points $x_1 := \beta_j \cap \beta_{i,1}$ and $x_2 := \beta_j \cap \beta_{i,2}$ lie on a Voronoi edge $e \subseteq B(p_i, p_j)$.



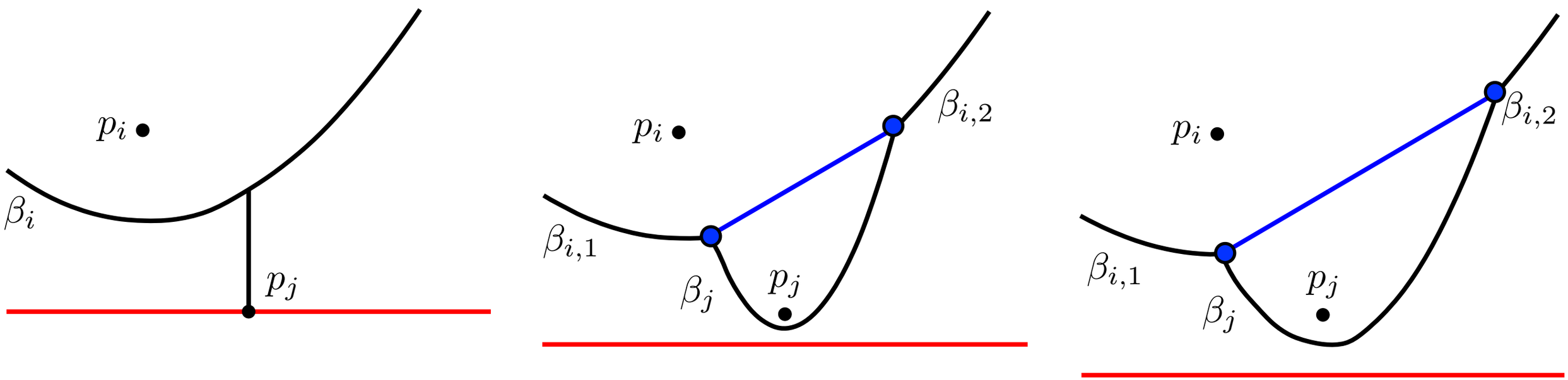
Observation:

- Point event $p_j \rightarrow$ ≥ 1 Voronoi edges are discovered.
- More precisely: Arc β_j (defined by p_j) splits β_i (defined by p_i) into $\beta_{i,1}, \beta_{i,2}$.
- Lemma 4.15/Corollary 4.16: Intersection points $x_1 := \beta_j \cap \beta_{i,1}$ and $x_2 := \beta_j \cap \beta_{i,2}$ lie on a Voronoi edge $e \subseteq B(p_i, p_j)$.



Observation:

- Point event $p_j \rightarrow$ ≥ 1 Voronoi edges are discovered.
- More precisely: Arc β_j (defined by p_j) splits β_i (defined by p_i) into $\beta_{i,1}, \beta_{i,2}$.
- Lemma 4.15/Corollary 4.16: Intersection points $x_1 := \beta_j \cap \beta_{i,1}$ and $x_2 := \beta_j \cap \beta_{i,2}$ lie on a Voronoi edge $e \subseteq B(p_i, p_j)$.

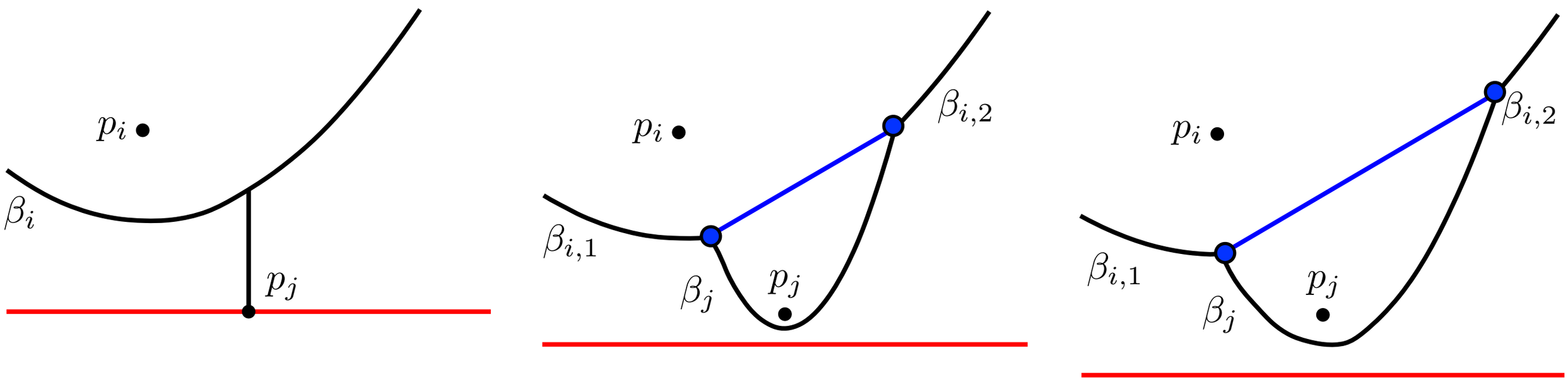


Lemma 4.15
$p \in \mathcal{P}$ defines arc β on beach line
$\Rightarrow$ p is nearest neighbor $\forall x \in \beta$.

Implications of a Point Event

Observation:

- Point event $p_j \rightarrow$ ≥ 1 Voronoi edges are discovered.
- More precisely: Arc β_j (defined by p_j) splits β_i (defined by p_i) into $\beta_{i,1}, \beta_{i,2}$.
- Lemma 4.15/Corollary 4.16: Intersection points $x_1 := \beta_j \cap \beta_{i,1}$ and $x_2 := \beta_j \cap \beta_{i,2}$ lie on a Voronoi edge $e \subseteq B(p_i, p_j)$.

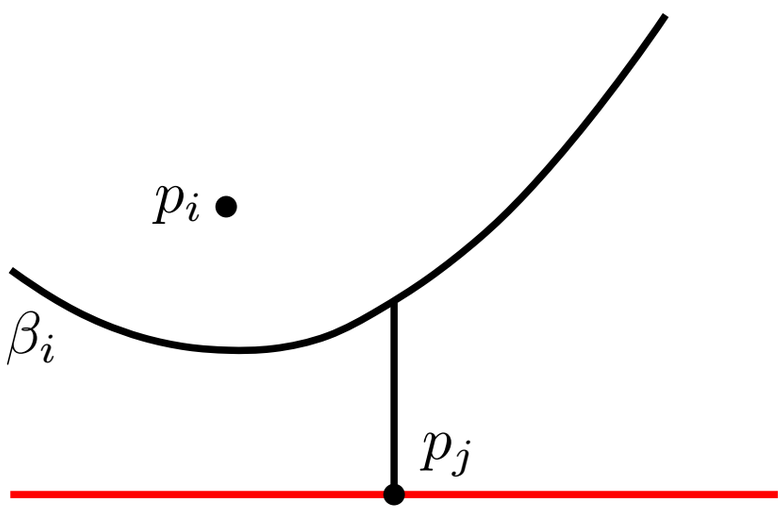


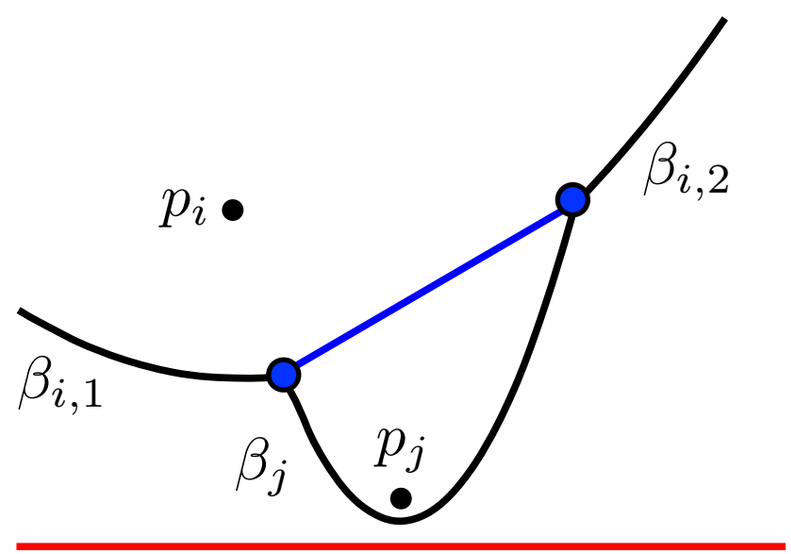
Lemma 4.15
$p \in \mathcal{P}$ defines arc β on beach line
$\Rightarrow$ p is nearest neighbor $\forall x \in \beta$.

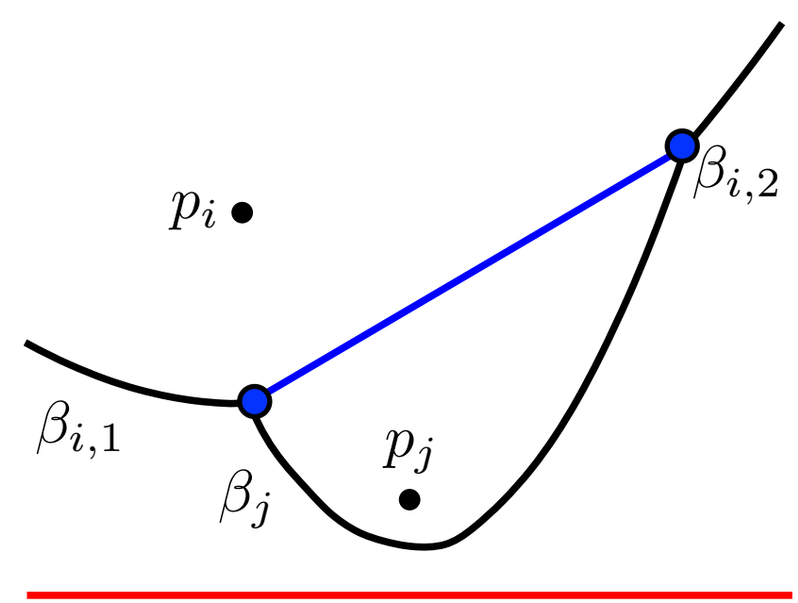
Corollary 4.16
Intersection points of adjacent arcs lie on Voronoi edges.

Observation:

- Point event $p_j \rightarrow \geq 1$ Voronoi edges are discovered.
- More precisely: Arc β_j (defined by p_j) splits β_i (defined by p_i) into $\beta_{i,1}, \beta_{i,2}$.
- Lemma 4.15/Corollary 4.16: Intersection points $x_1 := \beta_j \cap \beta_{i,1}$ and $x_2 := \beta_j \cap \beta_{i,2}$ lie on a Voronoi edge $e \subseteq B(p_i, p_j)$.





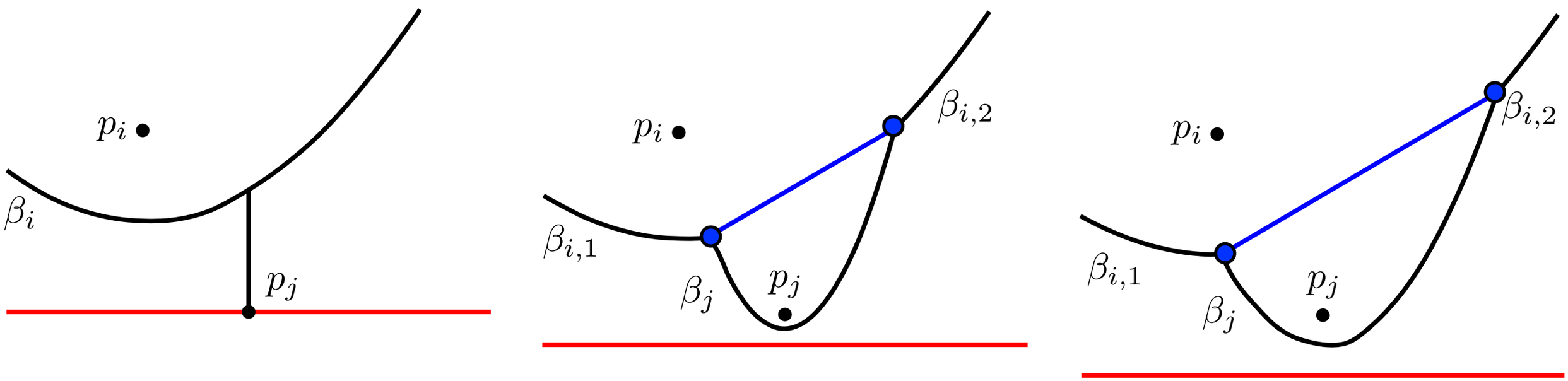


Lemma 4.15
$p \in \mathcal{P}$ defines arc β on beach line
$\Rightarrow$ p is nearest neighbor $\forall x \in \beta$.

Corollary 4.16
Intersection points of adjacent arcs lie on Voronoi edges.

Observation:

- Point event $p_j \rightarrow \geq 1$ Voronoi edges are discovered.
- More precisely: Arc β_j (defined by p_j) splits β_i (defined by p_i) into $\beta_{i,1}, \beta_{i,2}$.
- Lemma 4.15/Corollary 4.16: Intersection points $x_1 := \beta_j \cap \beta_{i,1}$ and $x_2 := \beta_j \cap \beta_{i,2}$ lie on a Voronoi edge $e \subseteq B(p_i, p_j)$.



- Edge i.g. not connected to already computed part of $Vor(\mathcal{P})$

Circle Events - I

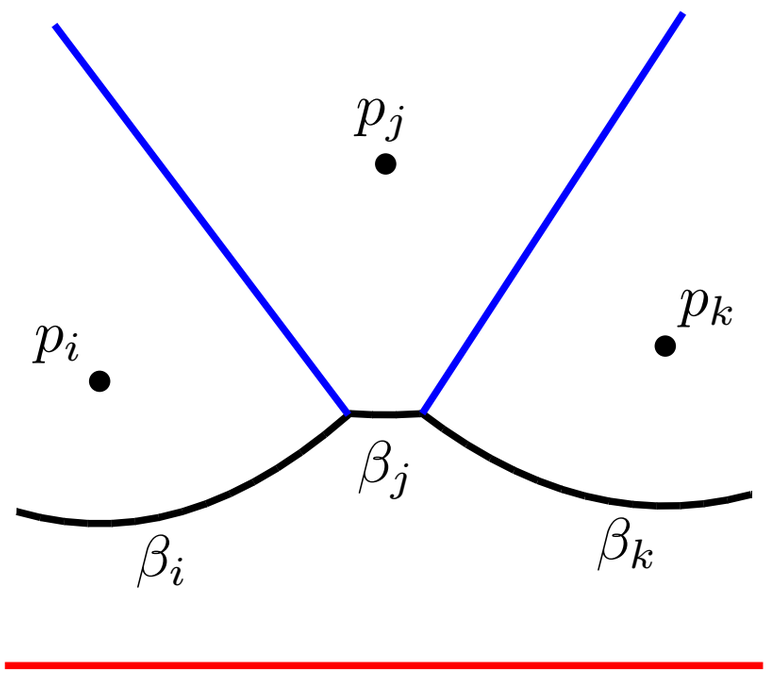
Circle events:

Circle events:

- Arc β_j shrinks to a point q.

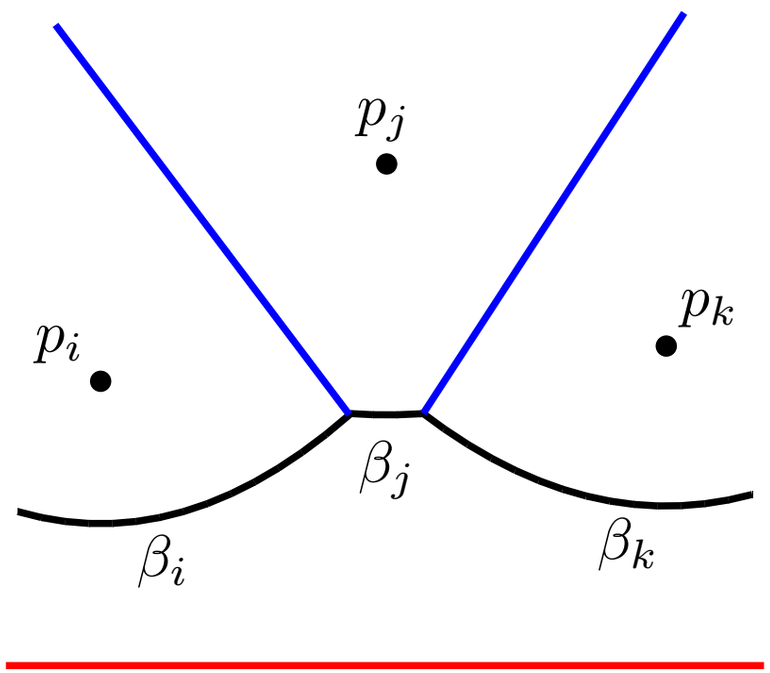
Circle events:

- Arc β_j shrinks to a point q.



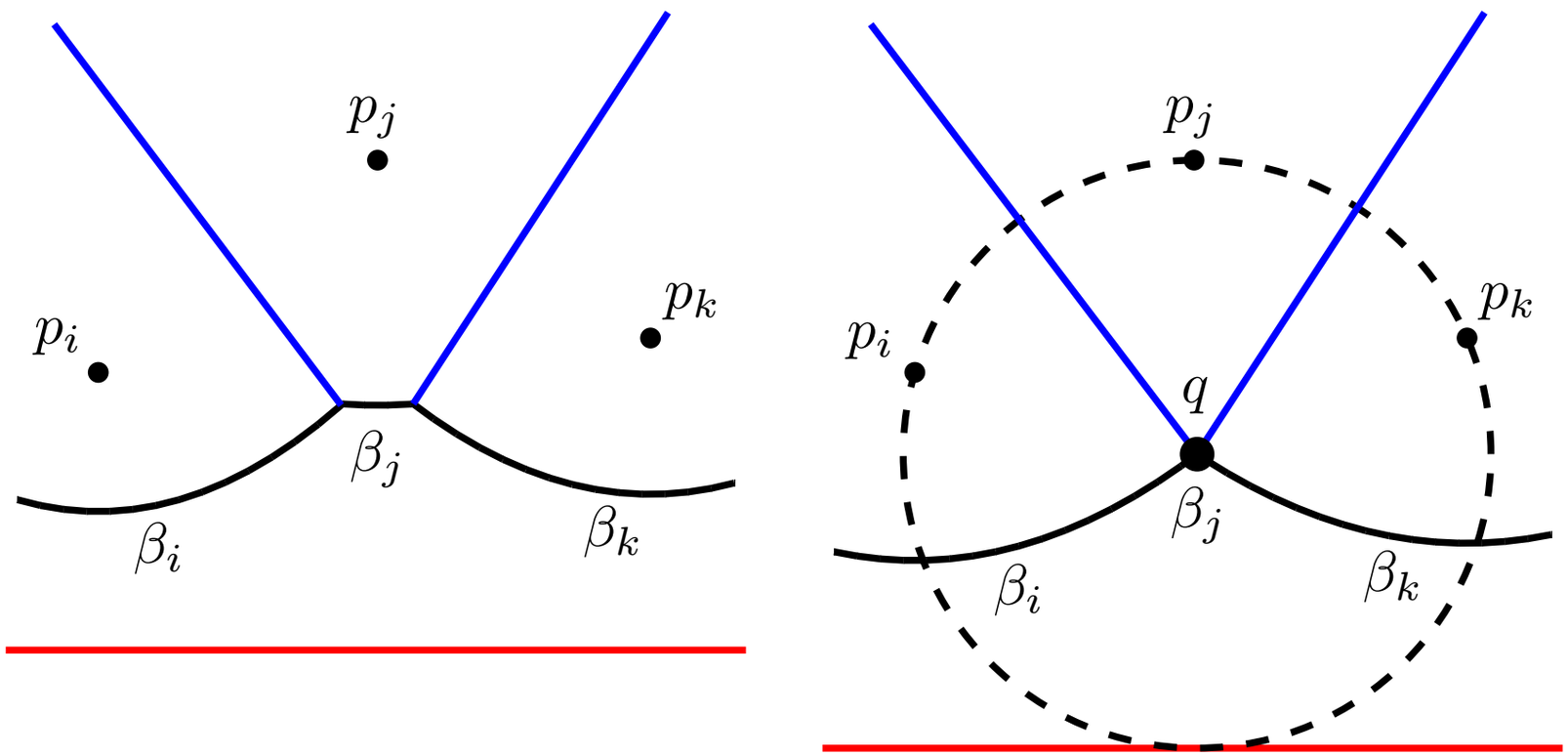
Circle events:

- Arc β_j shrinks to a point q.
- β_i / β_k left/right neighbor of β_j.



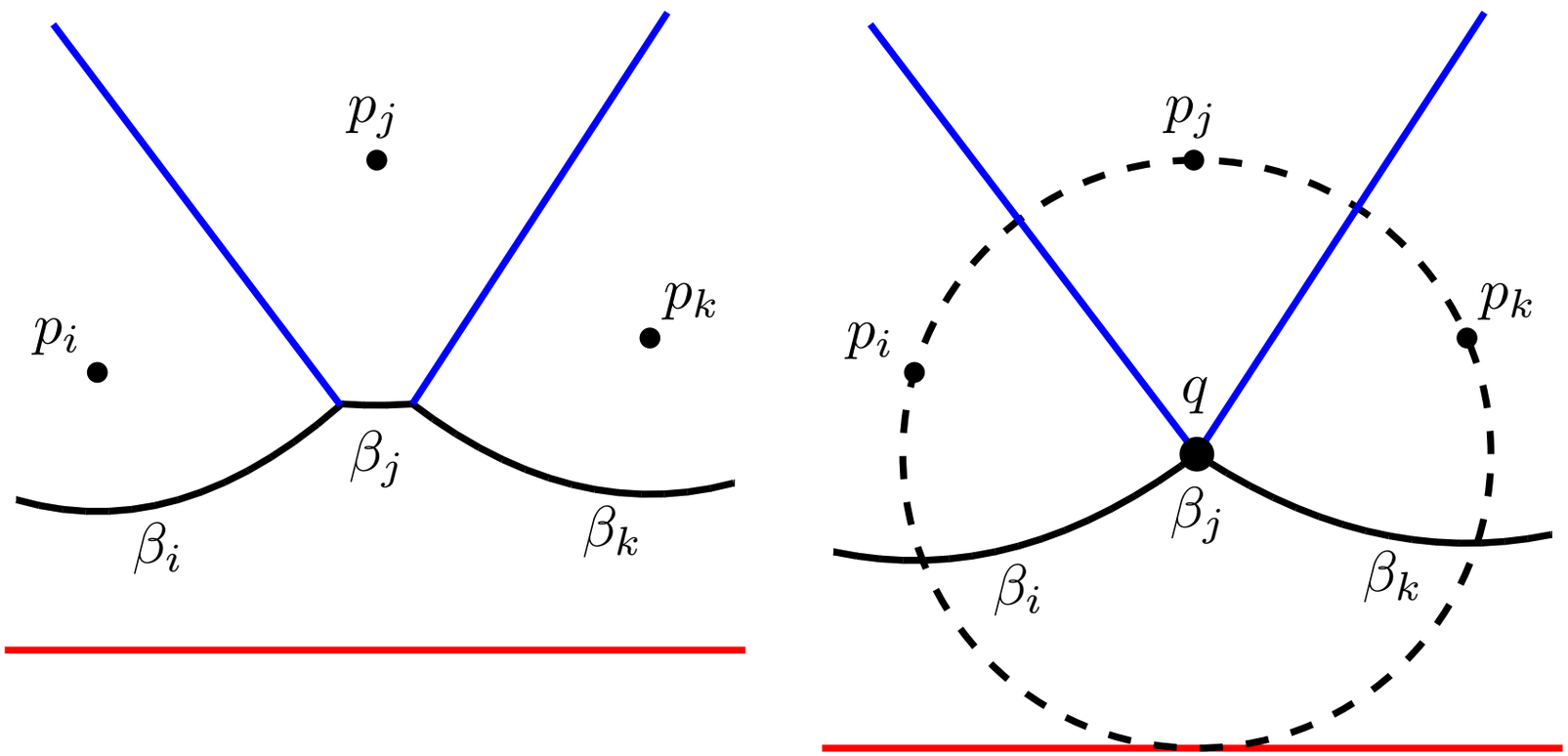
Circle events:

- Arc β_j shrinks to a point q.
- β_i / β_k left/right neighbor of β_j.



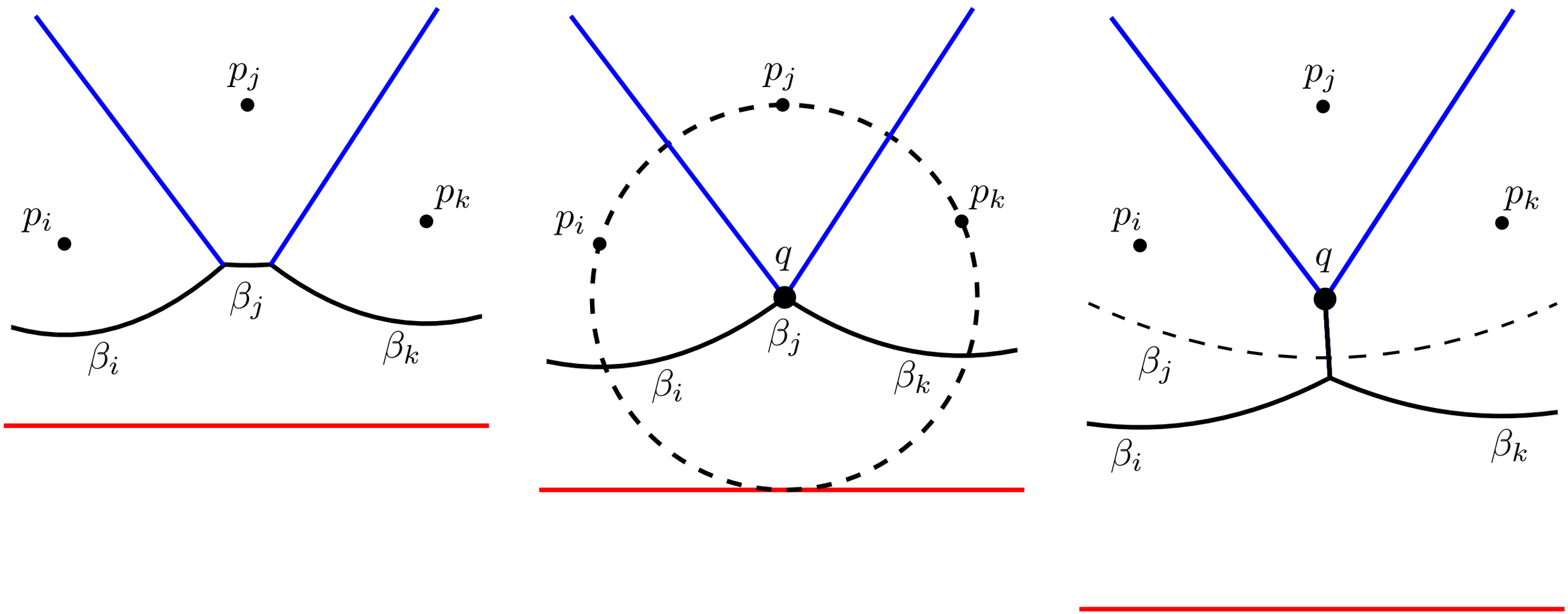
Circle events:

- Arc β_j shrinks to a point q.
- β_i / β_k left/right neighbor of β_j.
- $d(q, \ell) = d(q, p_i) = d(q, p_j) = d(q, p_k) \Rightarrow q$ Voronoi vertex. $(*)$



Circle events:

- Arc β_j shrinks to a point q.
- β_i / β_k left/right neighbor of β_j.
- $d(q, \ell) = d(q, p_i) = d(q, p_j) = d(q, p_k) \Rightarrow q$ Voronoi vertex. $(*)$



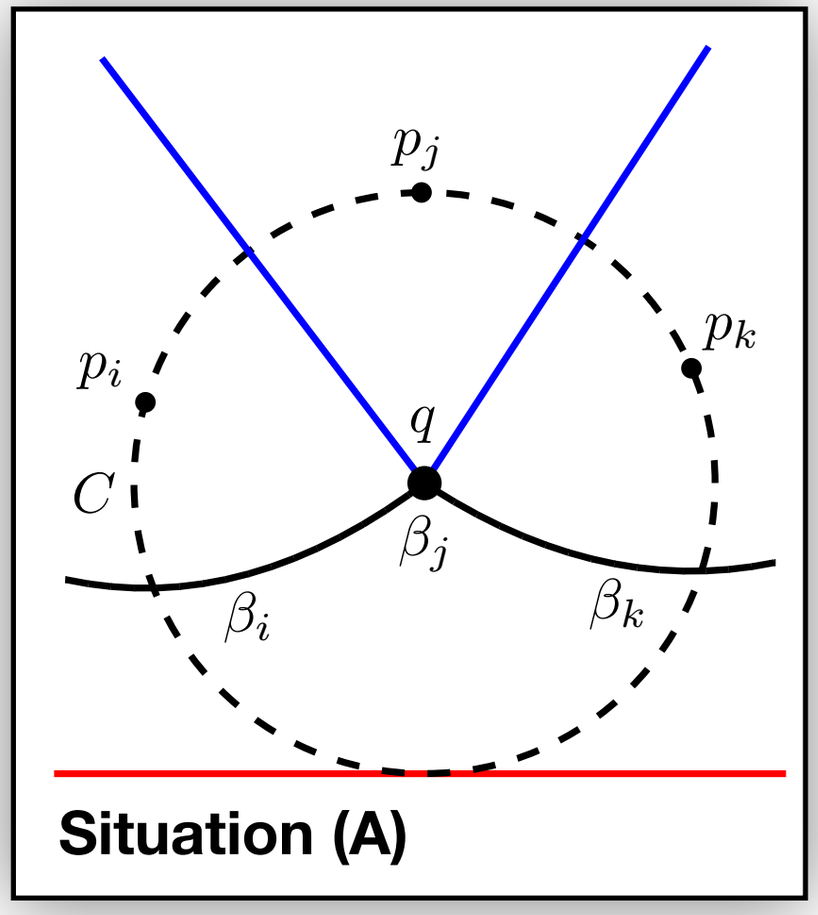
Circle Events - II

Lemma 4.19

In situation (A), C does not contain a $r \in \mathcal{P}$.

Lemma 4.19

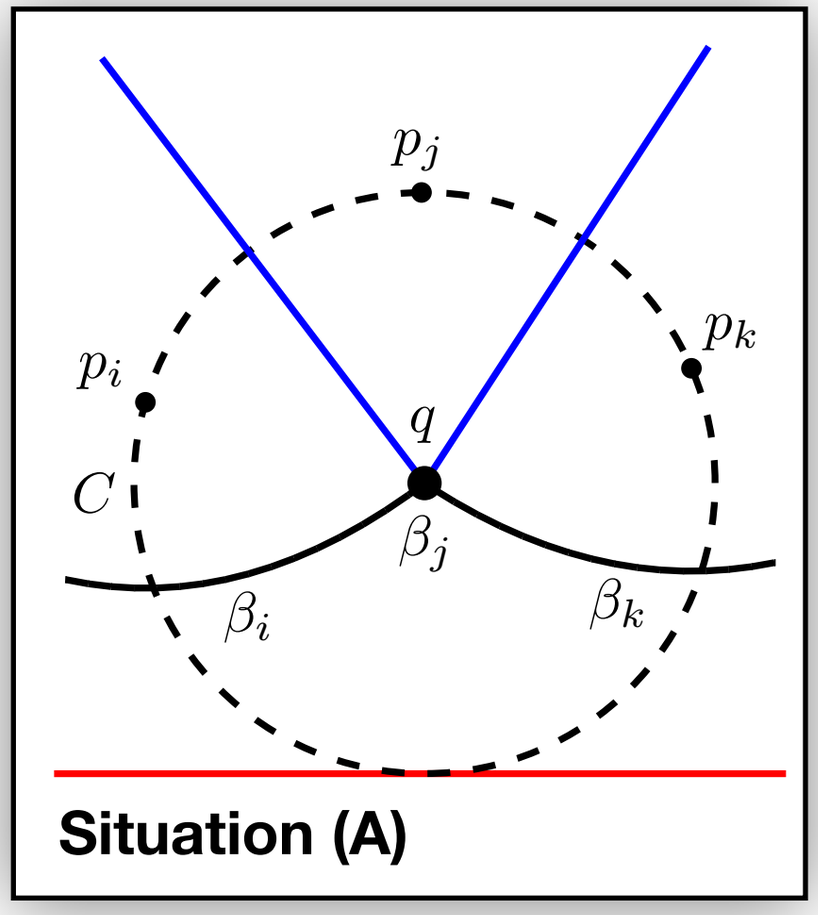
In situation (A), C does not contain a $r \in \mathcal{P}$.



Lemma 4.19

In situation (A), C does not contain a $r \in \mathcal{P}$.

Proof:

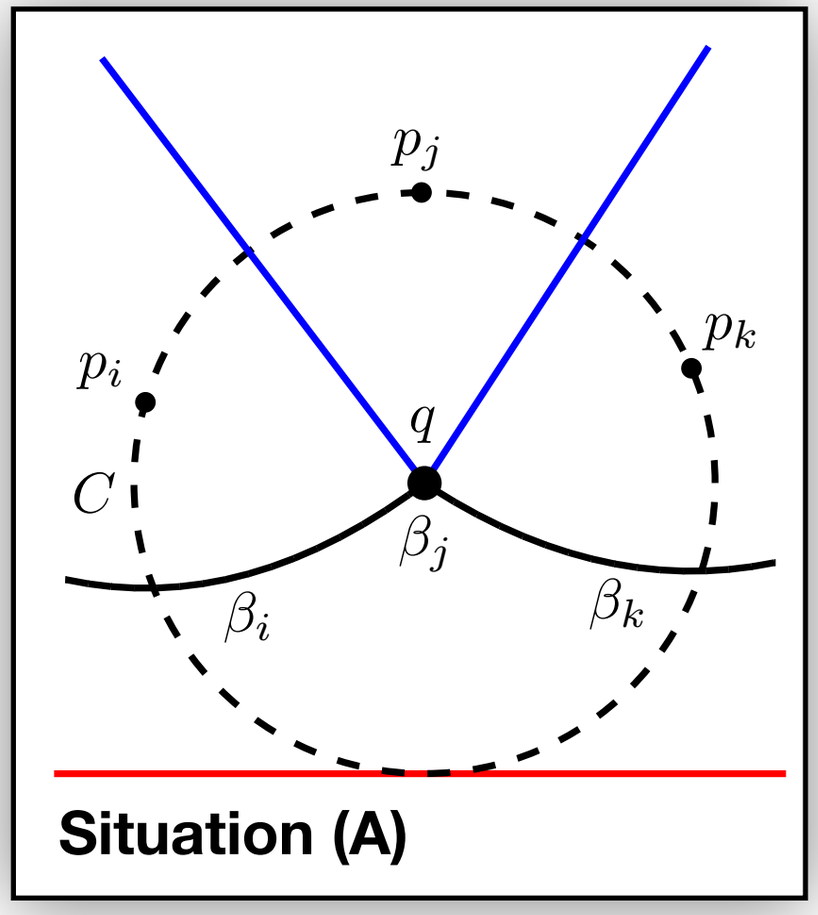


Lemma 4.19

In situation (A), C does not contain a $r \in \mathcal{P}$.

Proof:

- Assumption: $r \in C^\circ$.
  $\Rightarrow d(q,r) < d(q,\ell)$

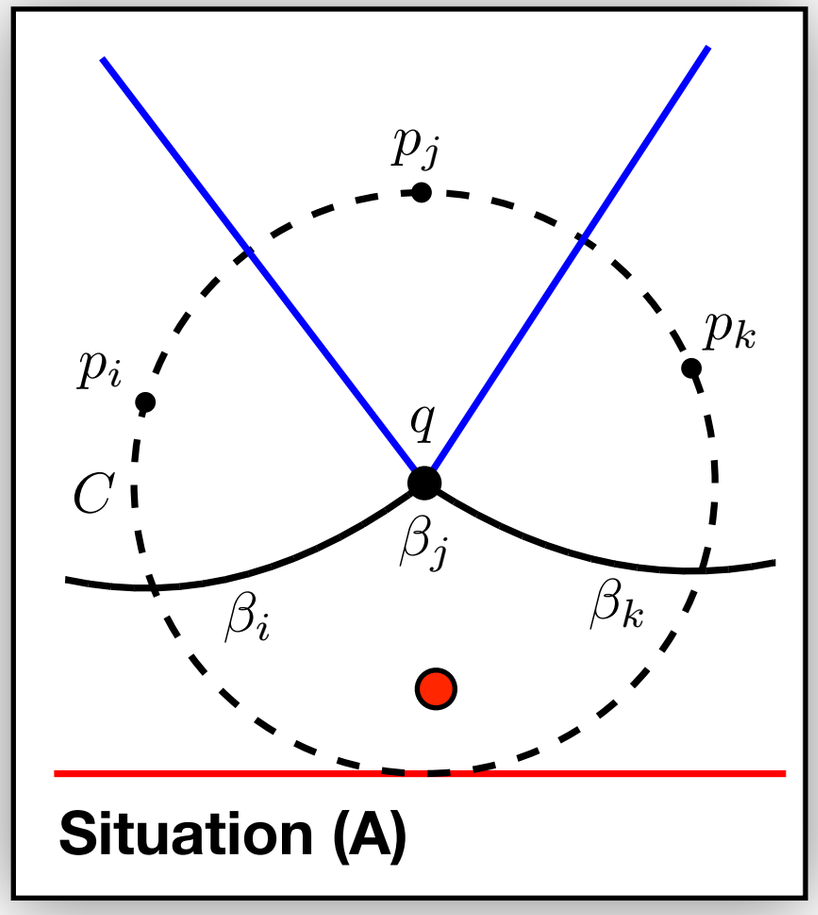


Lemma 4.19

In situation (A), C does not contain a $r \in \mathcal{P}$.

Proof:

- Assumption: $r \in C^\circ$.
  $\Rightarrow d(q,r) < d(q,\ell)$



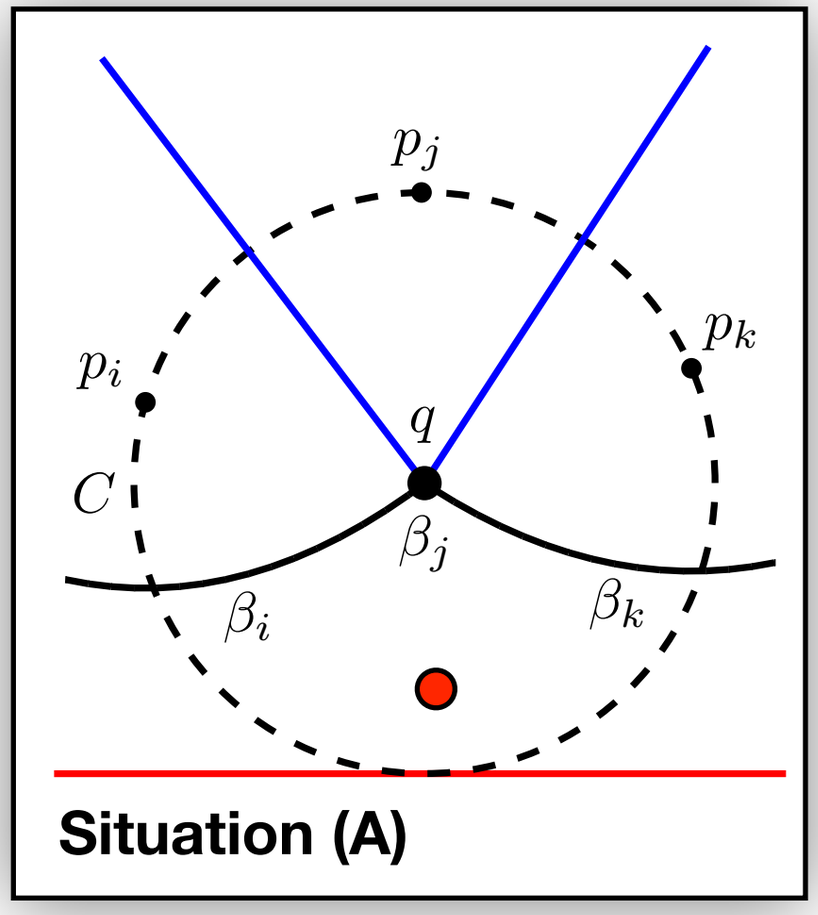
Lemma 4.19

In situation (A), C does not contain a $r \in \mathcal{P}$.

Proof:

- Assumption: $r \in C^\circ$.
  $\Rightarrow d(q,r) < d(q,\ell)$

- Lemma 4.15: Nearest neighbor s of q on ℓ or one of p_i, p_j, p_k.



Lemma 4.19

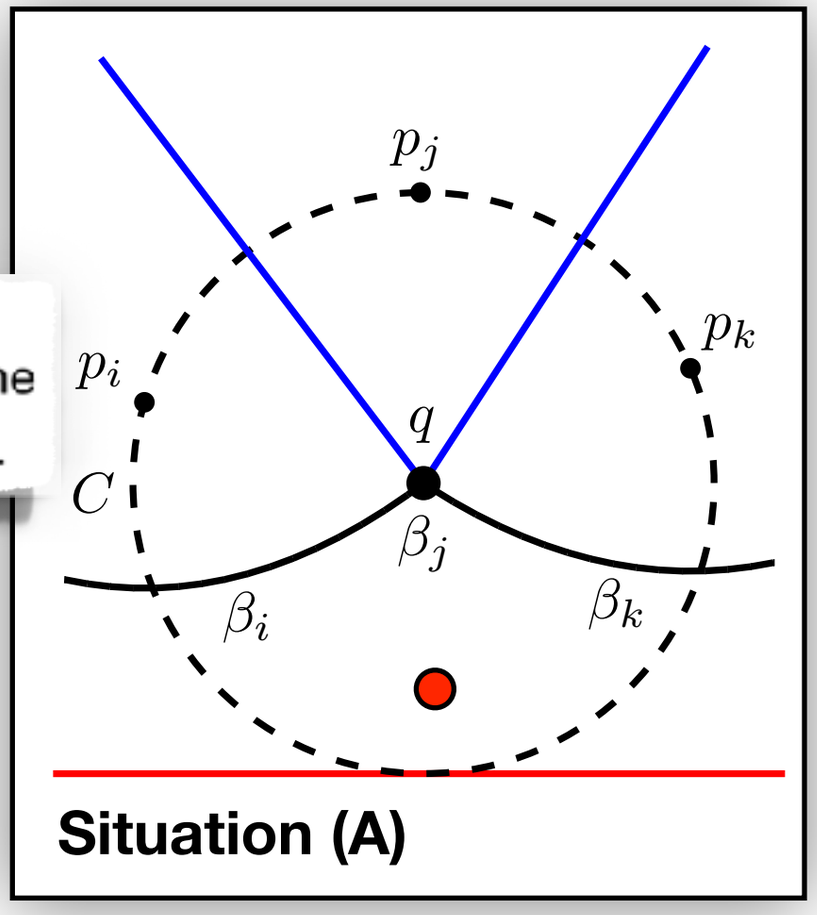
In situation (A), C does not contain a $r \in \mathcal{P}$.

Proof:

- Assumption: $r \in C^\circ$.
  $\Rightarrow d(q,r) < d(q,\ell)$

Lemma 4.15
$p \in \mathcal{P}$ defines arc β on beach line
$\Rightarrow p$ is nearest neighbor $\forall x \in \beta$.

- Lemma 4.15: Nearest neighbor s of q on ℓ or one of p_i, p_j, p_k.



Lemma 4.19

In situation (A), C does not contain a $r \in \mathcal{P}$.

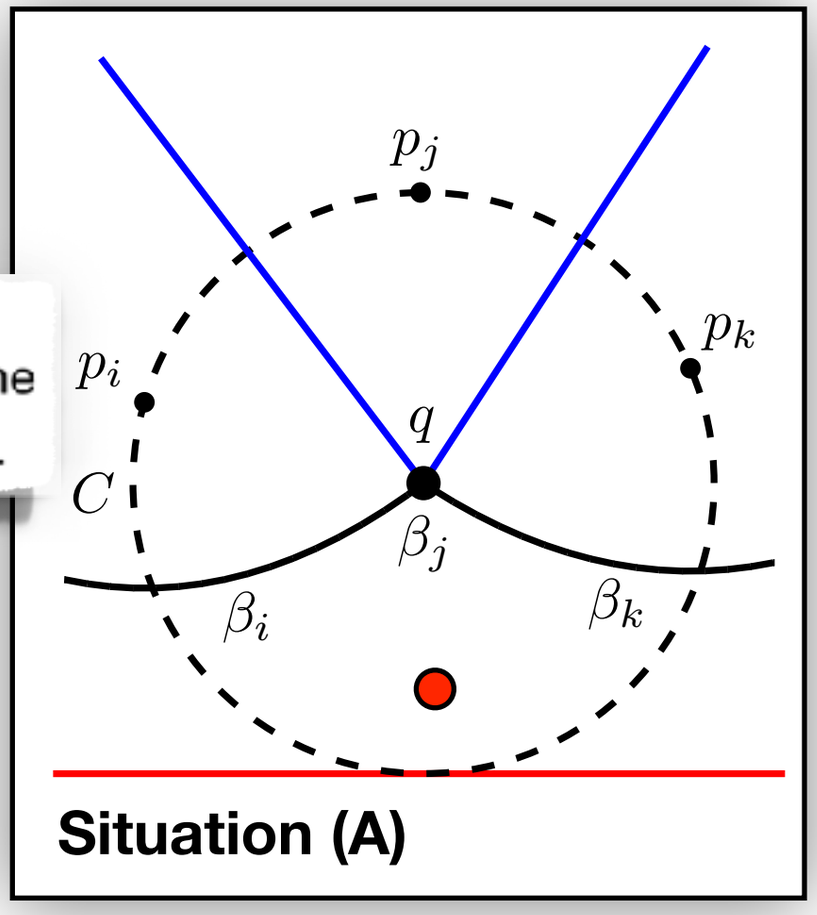
Proof:

- Assumption: $r \in C^\circ$.
  $\Rightarrow d(q,r) < d(q,\ell)$

Lemma 4.15
$p \in \mathcal{P}$ defines arc β on beach line
$\Rightarrow p$ is nearest neighbor $\forall x \in \beta$.

- Lemma 4.15: Nearest neighbor s of q on ℓ or one of p_i, p_j, p_k.

- We have: $d(q,s) = d(q,\ell) > d(q,r)$ ↯



Lemma 4.19

In situation (A), C does not contain a $r \in \mathcal{P}$.

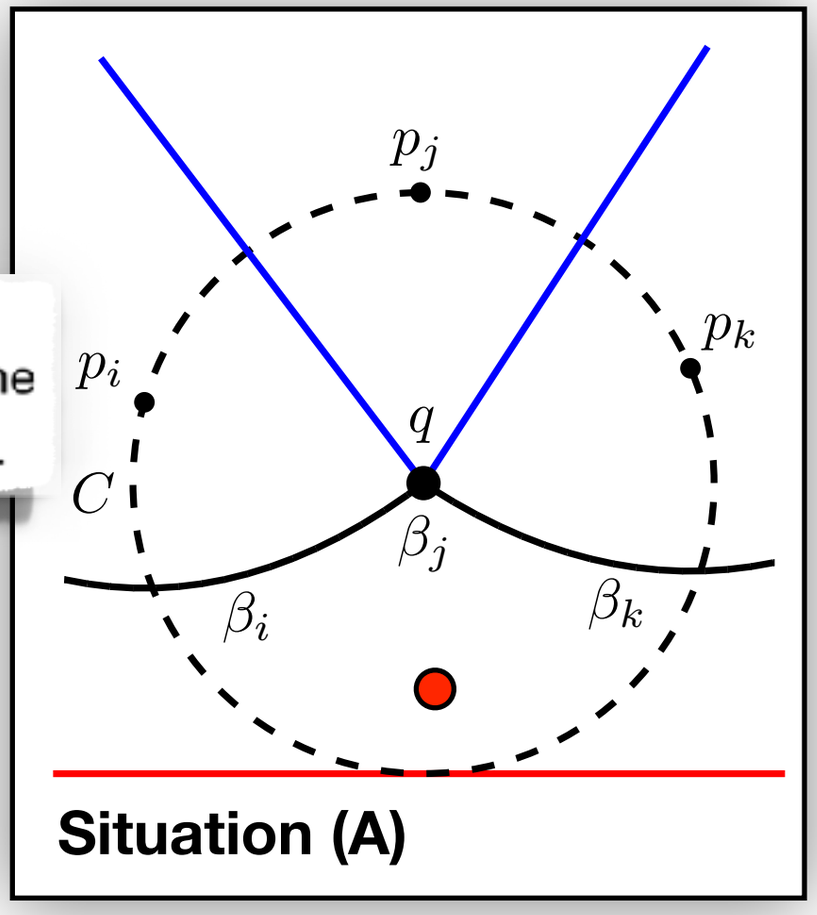
Proof:

- Assumption: $r \in C^\circ$.
  $\Rightarrow d(q,r) < d(q,\ell)$

Lemma 4.15
$p \in \mathcal{P}$ defines arc β on beach line
$\Rightarrow p$ is nearest neighbor $\forall x \in \beta$.

- Lemma 4.15: Nearest neighbor s of q on ℓ or one of p_i, p_j, p_k.

- We have: $d(q,s) = d(q,\ell) > d(q,r)$ ↯



Lemma 4.19

In situation (A), C does not contain a $r \in \mathcal{P}$.

Proof:

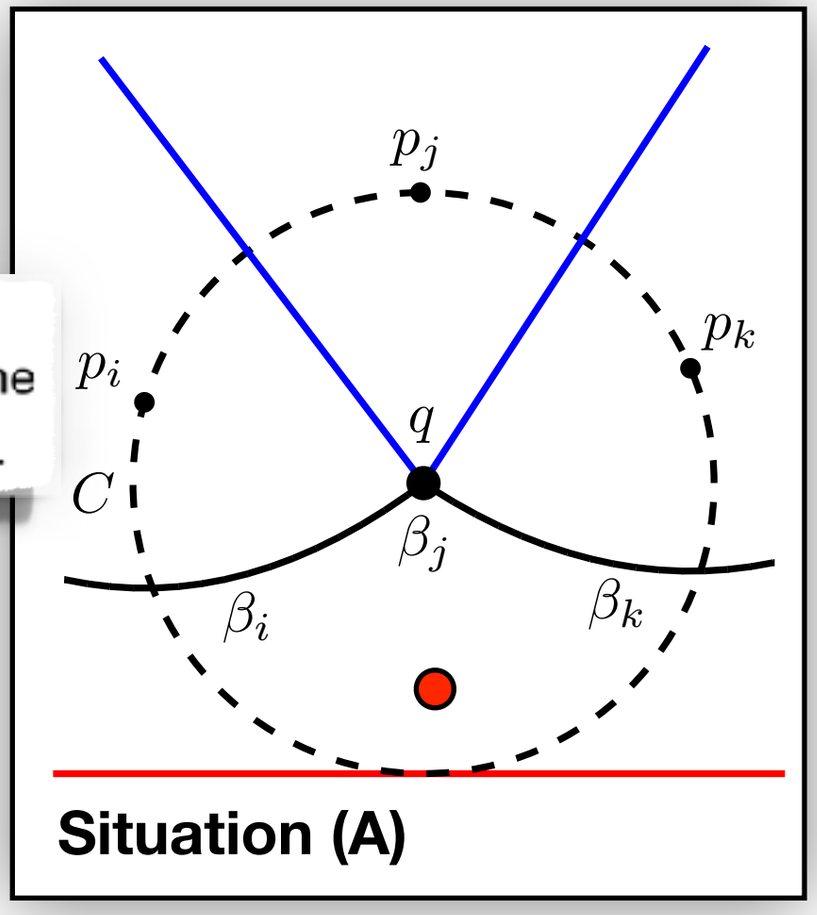
- Assumption: $r \in C^\circ$.
  $\Rightarrow d(q,r) < d(q,\ell)$

Lemma 4.15
$p \in \mathcal{P}$ defines arc β on beach line
$\Rightarrow p$ is nearest neighbor $\forall x \in \beta$.

- Lemma 4.15: Nearest neighbor s of q on ℓ or one of p_i, p_j, p_k.

- We have: $d(q,s) = d(q,\ell) > d(q,r)$ ↯

$\square$



Lemma 4.19

In situation (A), C does not contain a $r \in \mathcal{P}$.

Proof:

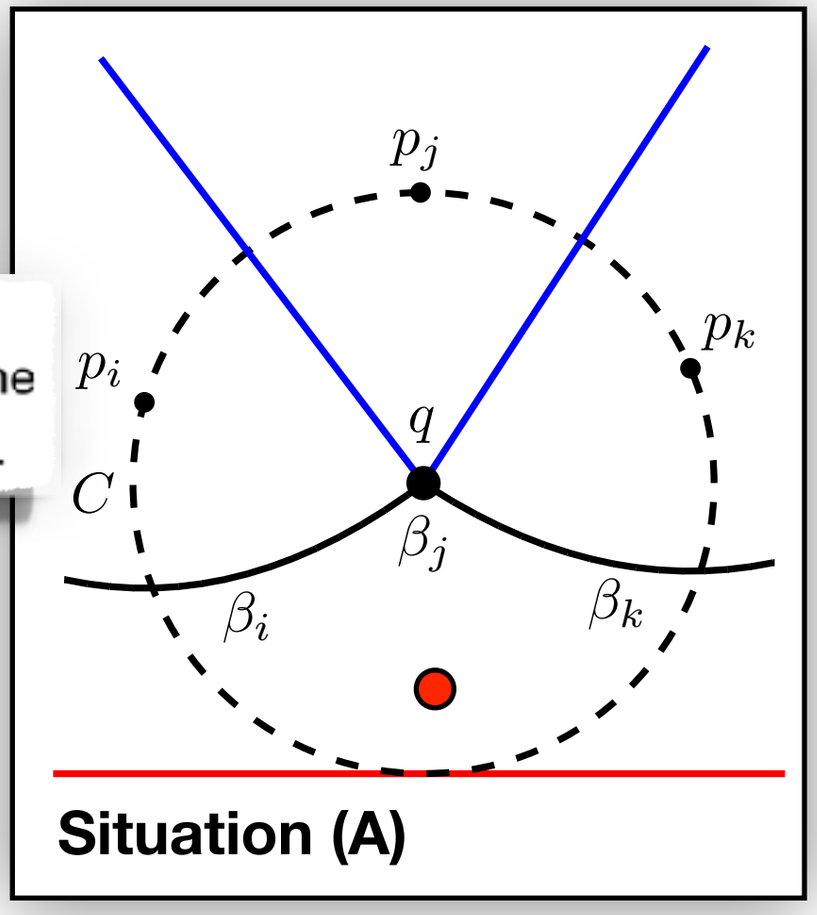
- Assumption: $r \in C^\circ$.
  $\Rightarrow d(q,r) < d(q,\ell)$

Lemma 4.15
$p \in \mathcal{P}$ defines arc β on beach line
$\Rightarrow p$ is nearest neighbor $\forall x \in \beta$.

- Lemma 4.15: Nearest neighbor s of q on ℓ or one of p_i, p_j, p_k.

- We have: $d(q,s) = d(q,\ell) > d(q,r)$ ↯

$\square$



Observation:

- Arc disappears $\Leftrightarrow \ell$ reaches lowest point of circle C.

Lemma 4.19

In situation (A), C does not contain a $r \in \mathcal{P}$.

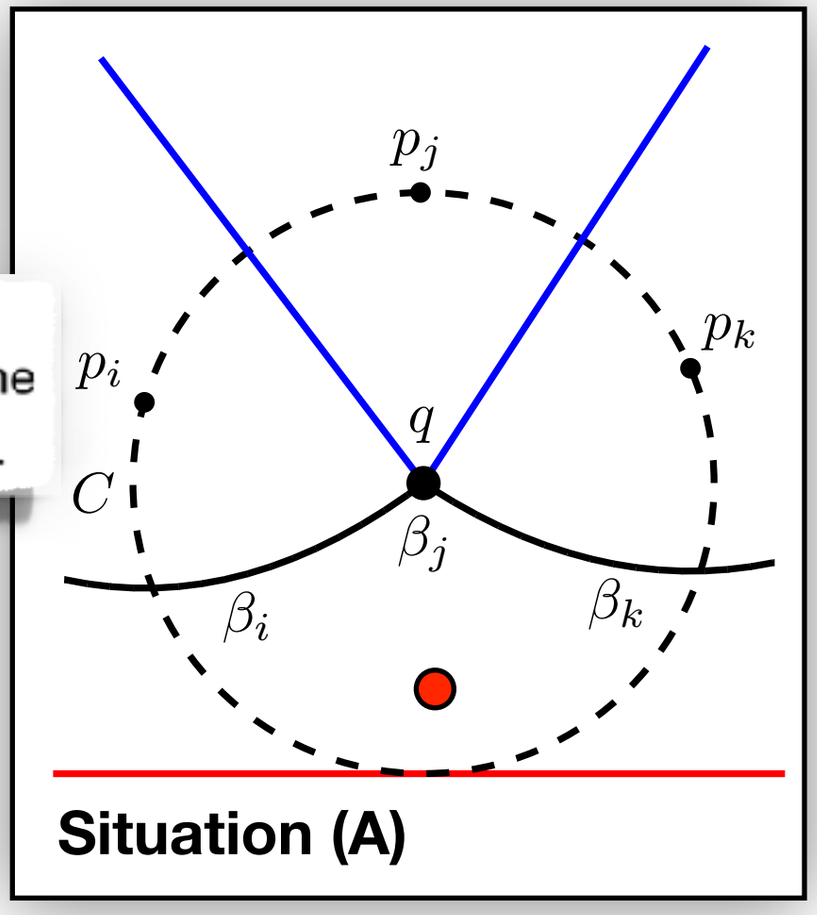
Proof:

- Assumption: $r \in C^\circ$.
  $\Rightarrow d(q,r) < d(q,\ell)$

Lemma 4.15
$p \in \mathcal{P}$ defines arc β on beach line
$\Rightarrow p$ is nearest neighbor $\forall x \in \beta$.

- Lemma 4.15: Nearest neighbor s of q on ℓ or one of p_i, p_j, p_k.
- We have: $d(q,s) = d(q,\ell) > d(q,r)$ ↯

$\square$



Observation:

- Arc disappears $\Leftrightarrow \ell$ reaches lowest point of circle C.
- $p_i, p_j, p_k \in \partial C$
- $C^\circ \cap \mathcal{P} = \emptyset$

$\Rightarrow$ Algorithmic recognition of Voronoi vertices

Lemma 4.19

In situation (A), C does not contain a $r \in \mathcal{P}$.

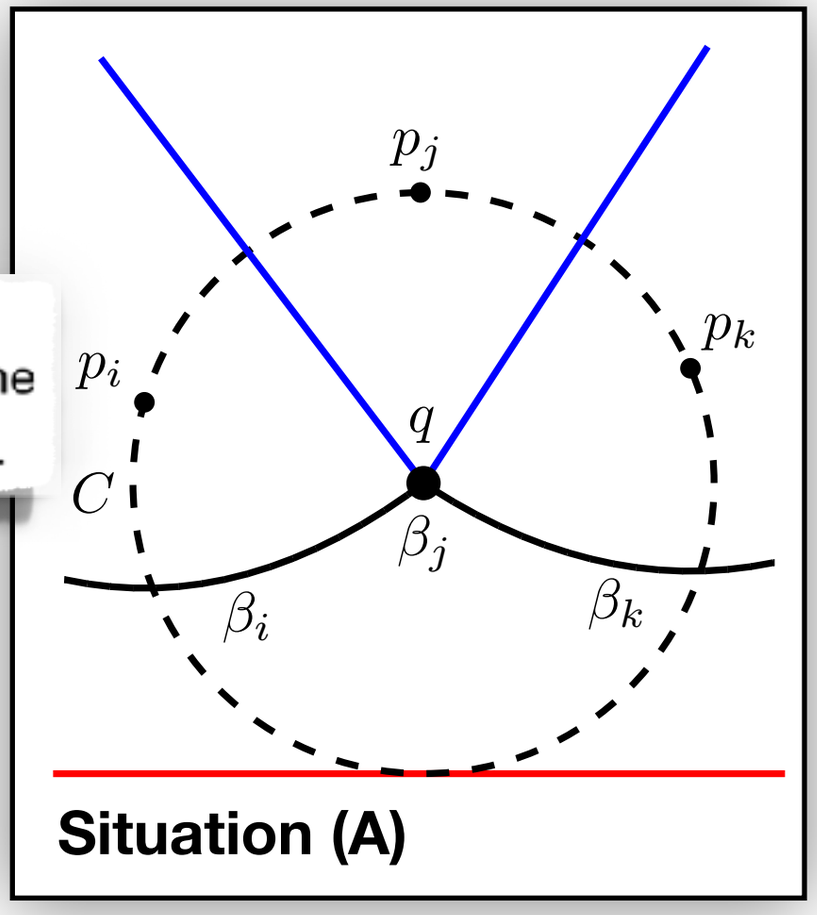
Proof:

- Assumption: $r \in C^\circ$.
  $\Rightarrow d(q,r) < d(q,\ell)$

Lemma 4.15
$p \in \mathcal{P}$ defines arc β on beach line
$\Rightarrow p$ is nearest neighbor $\forall x \in \beta$.

- Lemma 4.15: Nearest neighbor s of q on ℓ or one of p_i, p_j, p_k.
- We have: $d(q,s) = d(q,\ell) > d(q,r)$ ↯

$\square$



Observation:

- Arc disappears $\Leftrightarrow \ell$ reaches lowest point of circle C.
- $p_i, p_j, p_k \in \partial C$
- $C^\circ \cap \mathcal{P} = \emptyset$

$\Rightarrow$ Algorithmic recognition of Voronoi vertices

Properties of the Voronoi Diagram

Theorem 4.20:

Theorem 4.20:

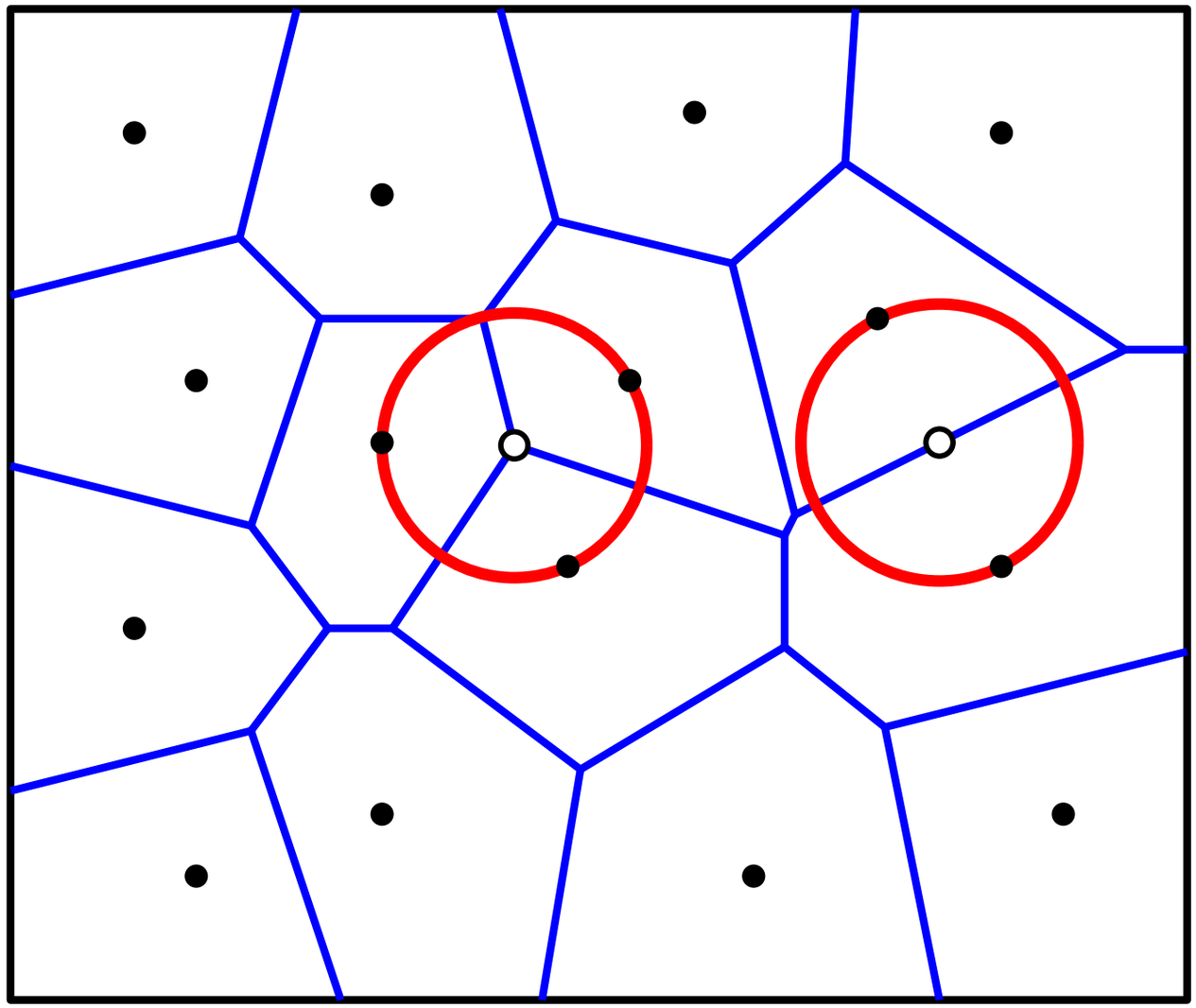
1. $x \in \mathbb{R}^2$ Voronoi vertex

$$\Updownarrow$$

Theorem 4.20:

1. $x \in \mathbb{R}^2$ Voronoi vertex

$$\Updownarrow$$

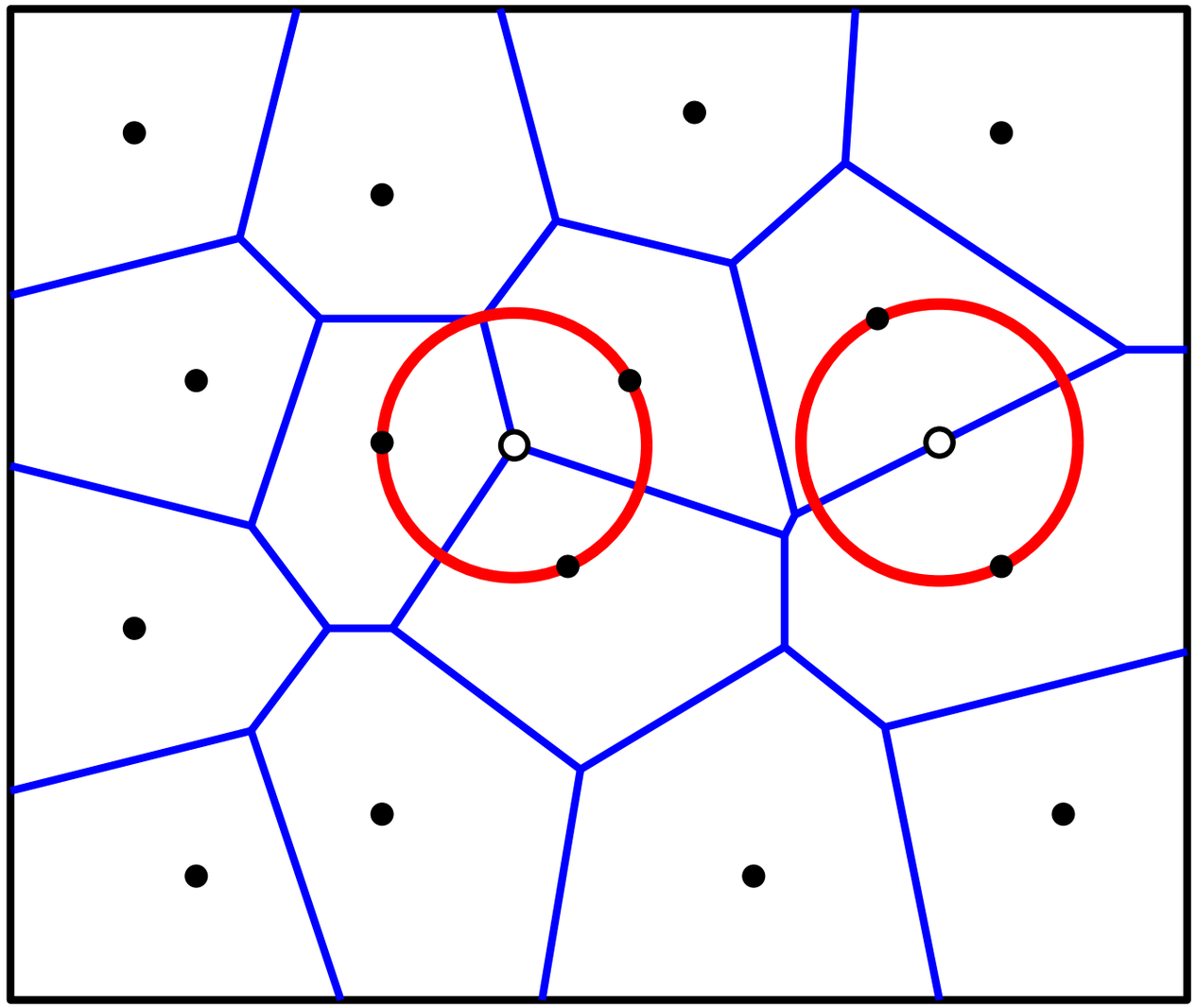


Theorem 4.20:

1. $x \in \mathbb{R}^2$ Voronoi vertex

$$\Updownarrow$$

Largest circle C with $\mathcal{P} \cap C^\circ = \emptyset$ and center x has three points on its boundary

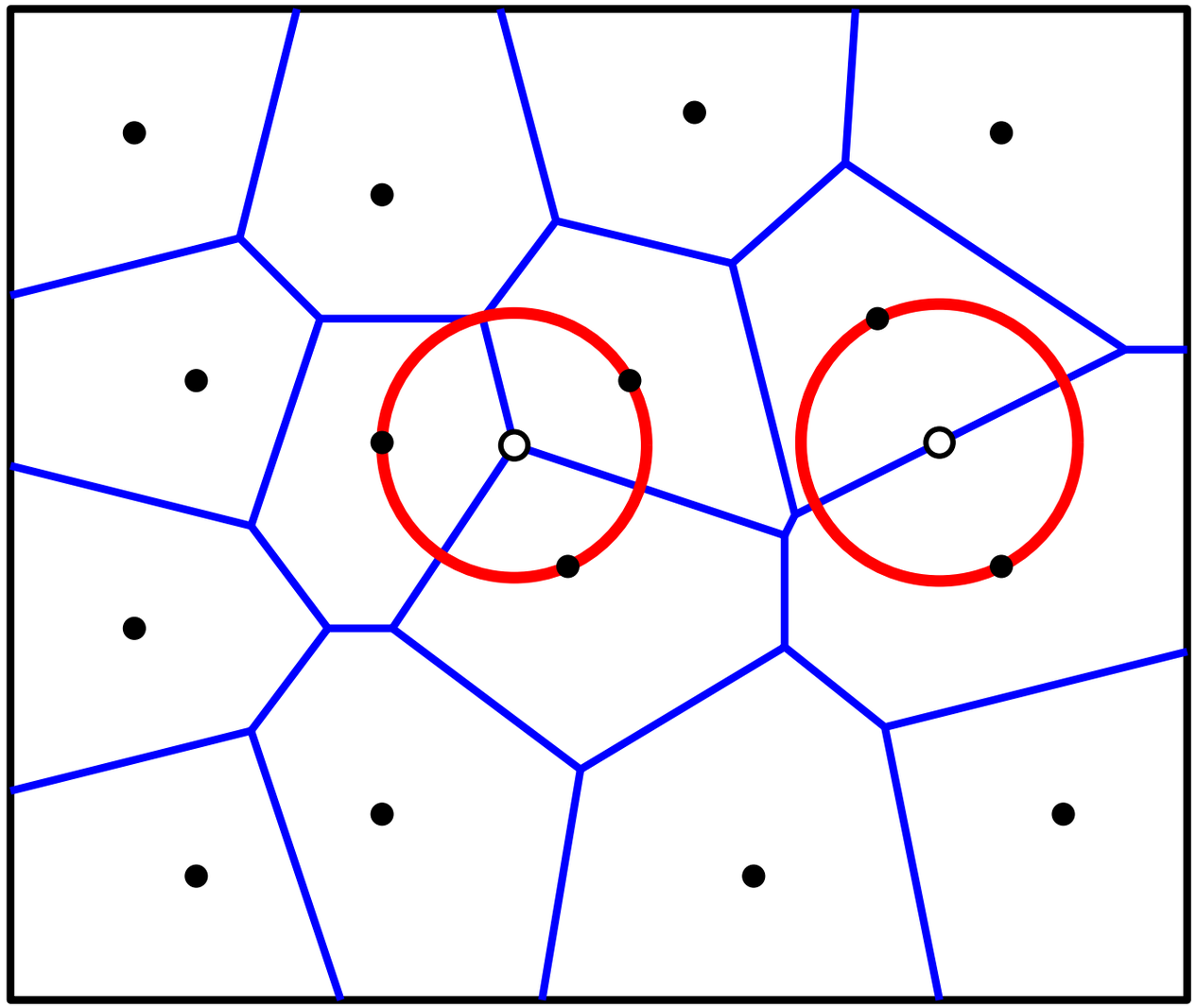


Theorem 4.20:

1. $x \in \mathbb{R}^2$ Voronoi vertex

$$\Updownarrow$$

Largest circle C with $\mathcal{P} \cap C^\circ = \emptyset$ and center x has three points on its boundary and



Theorem 4.20:

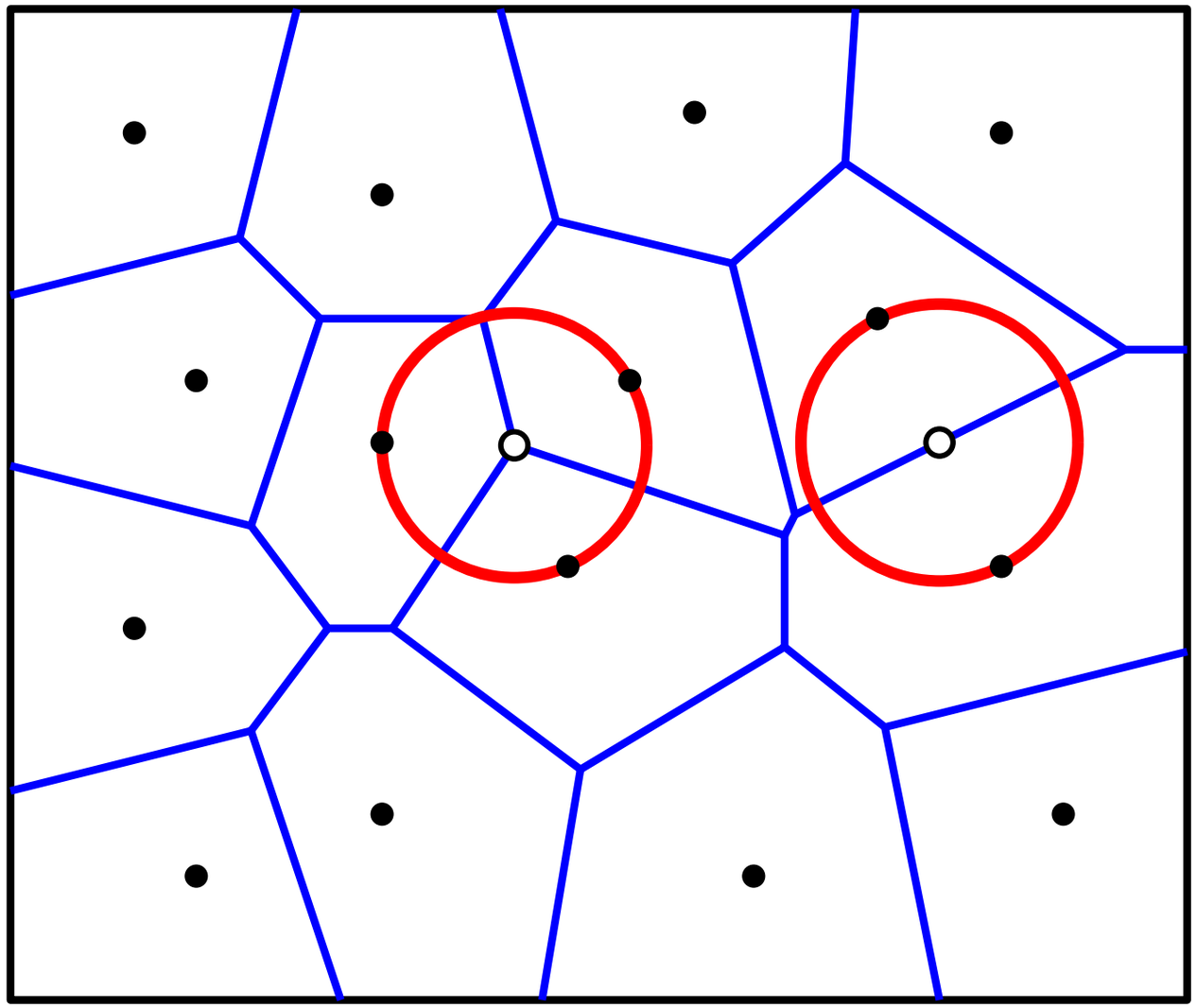
1. $x \in \mathbb{R}^2$ Voronoi vertex

   $\Updownarrow$

   Largest circle C with $\mathcal{P} \cap C^\circ = \emptyset$ and center x has three points on its boundary and

2. $p_i, p_j \in \mathcal{P}$ define Voronoi edge $e \subseteq B(p_i, p_j)$

   $\Updownarrow$



Theorem 4.20:

1. $x \in \mathbb{R}^2$ Voronoi vertex

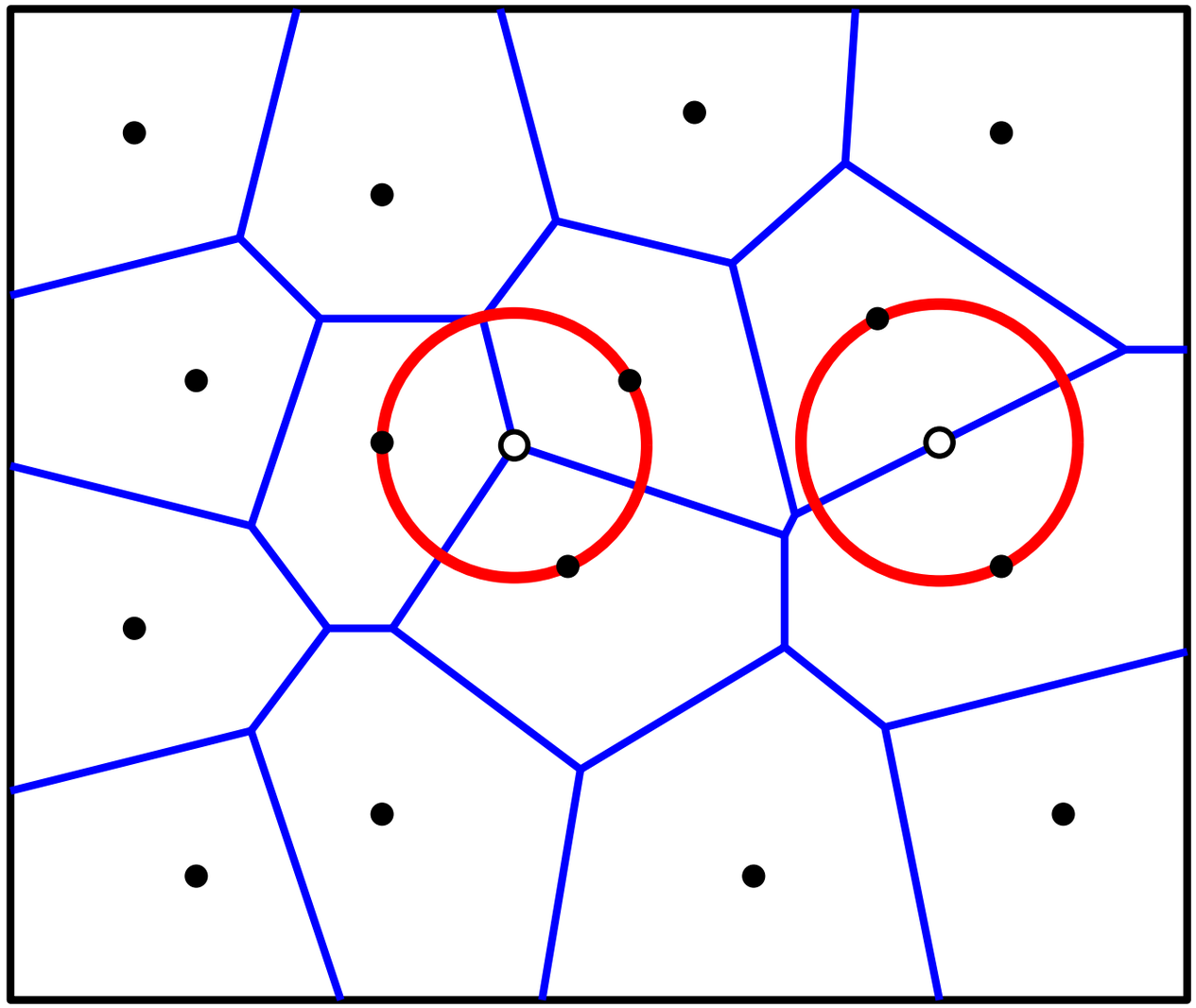
   $\Updownarrow$

   Largest circle C with
   $\mathcal{P} \cap C^\circ = \emptyset$ and
   center x has
   three points on its boundary
   and

2. $p_i, p_j \in \mathcal{P}$ define
   Voronoi edge $e \subseteq B(p_i, p_j)$

   $\Updownarrow$

   $\exists$ Circle C with
   - only p_i, p_j on boundary and
   - no point in its interior.



Theorem 4.20:

1. $x \in \mathbb{R}^2$ Voronoi vertex

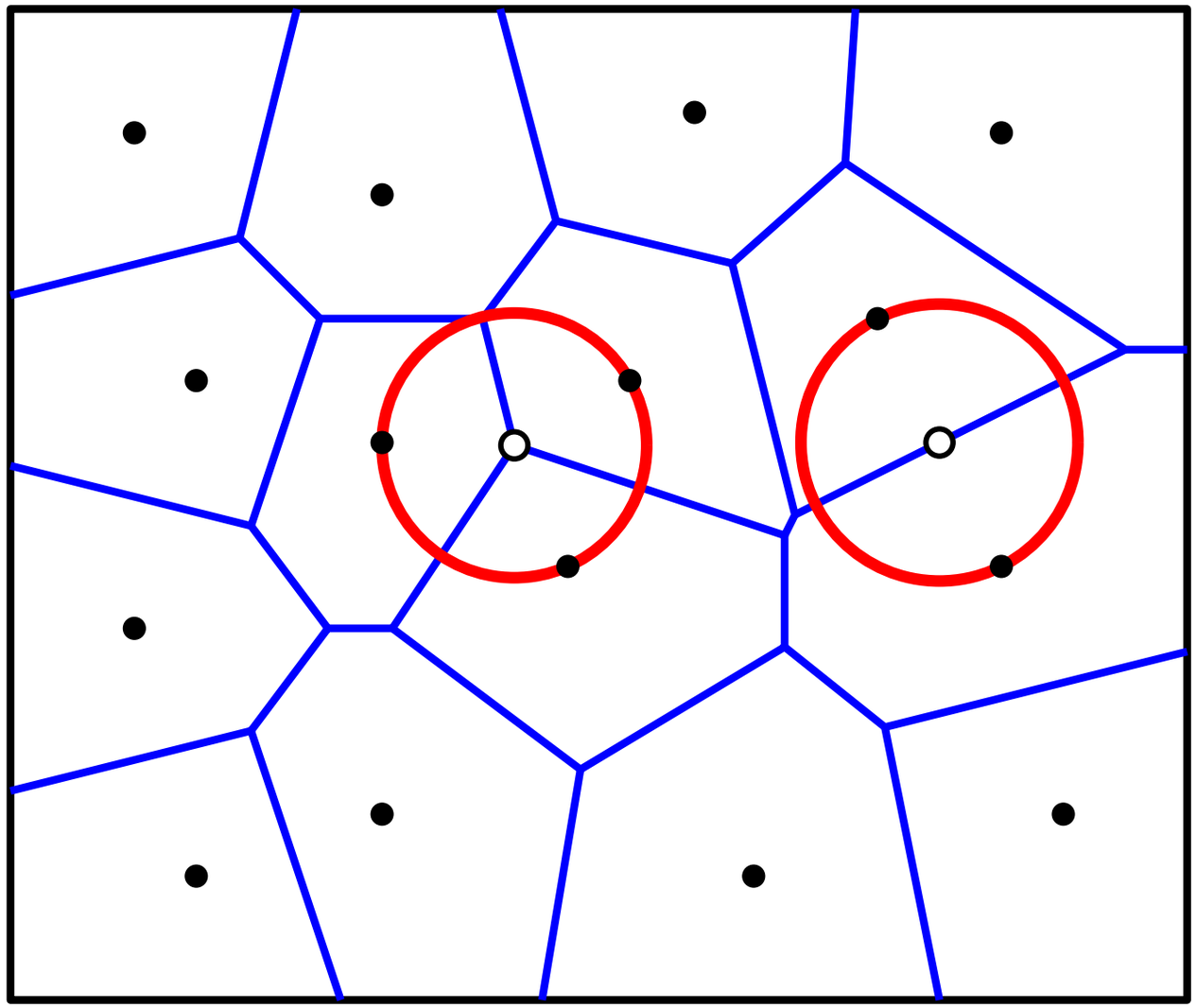
   $\Updownarrow$

   Largest circle C with
   $\mathcal{P} \cap C^\circ = \emptyset$ and
   center x has
   three points on its boundary
   and

2. $p_i, p_j \in \mathcal{P}$ define
   Voronoi edge $e \subseteq B(p_i, p_j)$

   $\Updownarrow$

   $\exists$ Circle C with
   - only p_i, p_j on boundary and
   - no point in its interior.



Proof:

Straightforward, because nearest neighbor to center lies on circle.

Fortune’s Algorithm - I

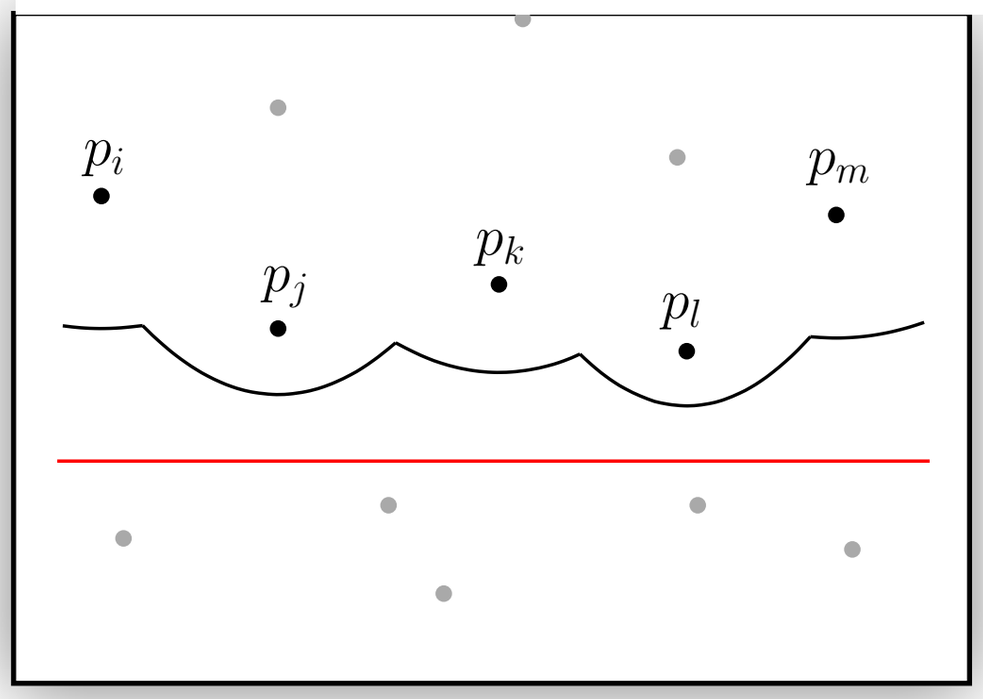
Approach [Fortune, 1987]:

Approach [Fortune, 1987]:

- *Plane sweep*: Beach line in x-structure.

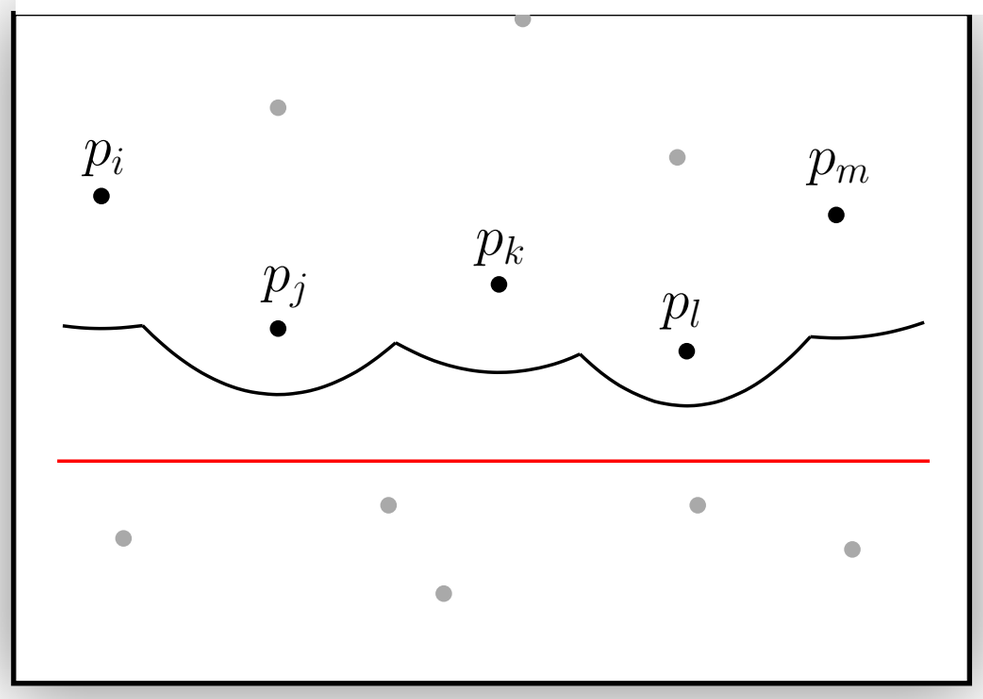
Approach [Fortune, 1987]:

- *Plane sweep*: Beach line in x-structure.



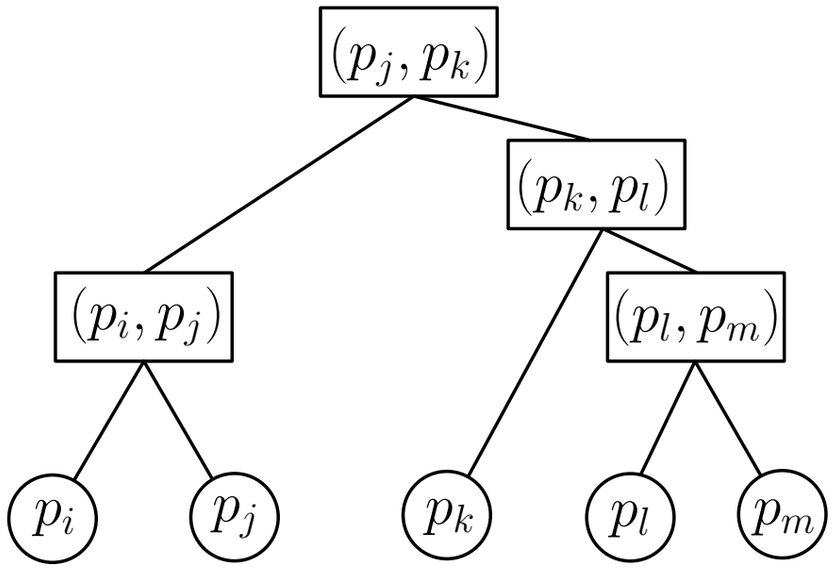
Approach [Fortune, 1987]:

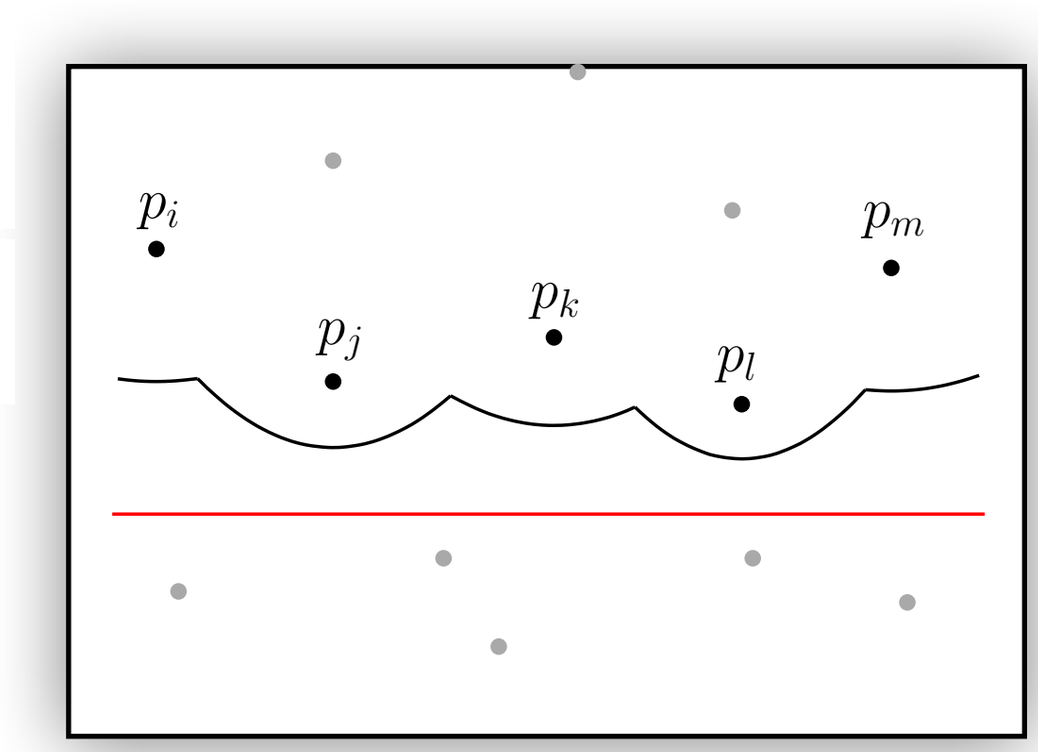
- *Plane sweep*: Beach line in x-structure.
- x-structure: balanced binary search tree B.



Approach [Fortune, 1987]:

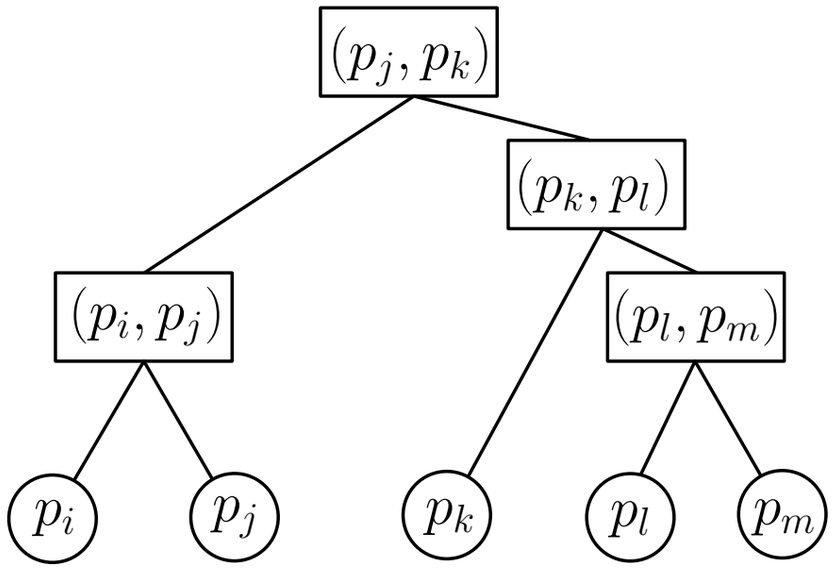
- *Plane sweep*: Beach line in x-structure.
- x-structure: balanced binary search tree B.

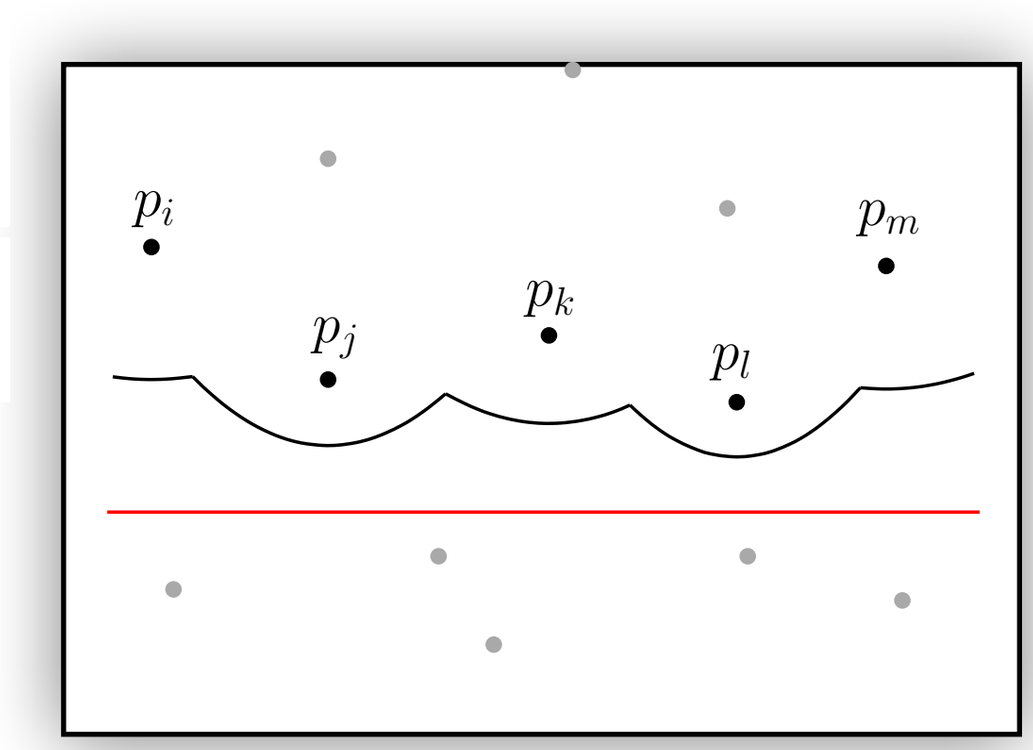




Approach [Fortune, 1987]:

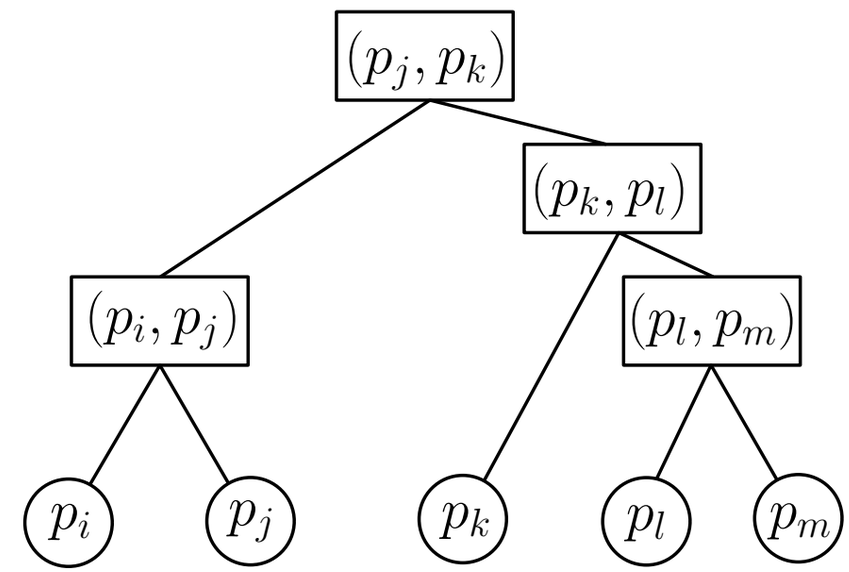
- *Plane sweep*: Beach line in x-structure.
- x-structure: balanced binary search tree B.
- Leaves = parabolic arcs.

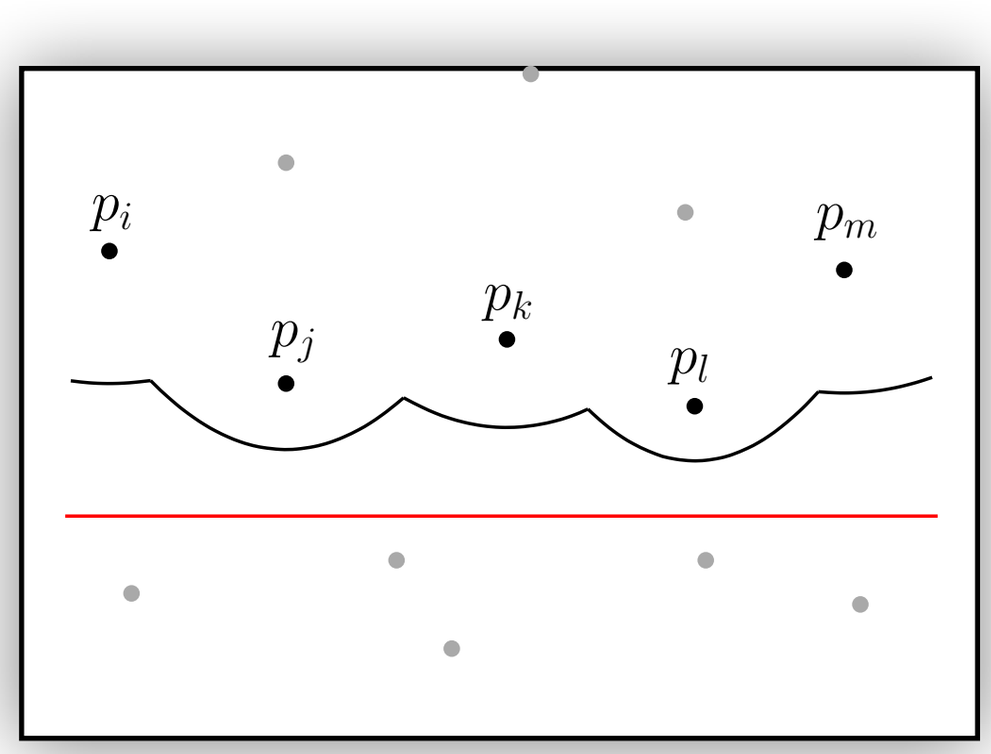




Approach [Fortune, 1987]:

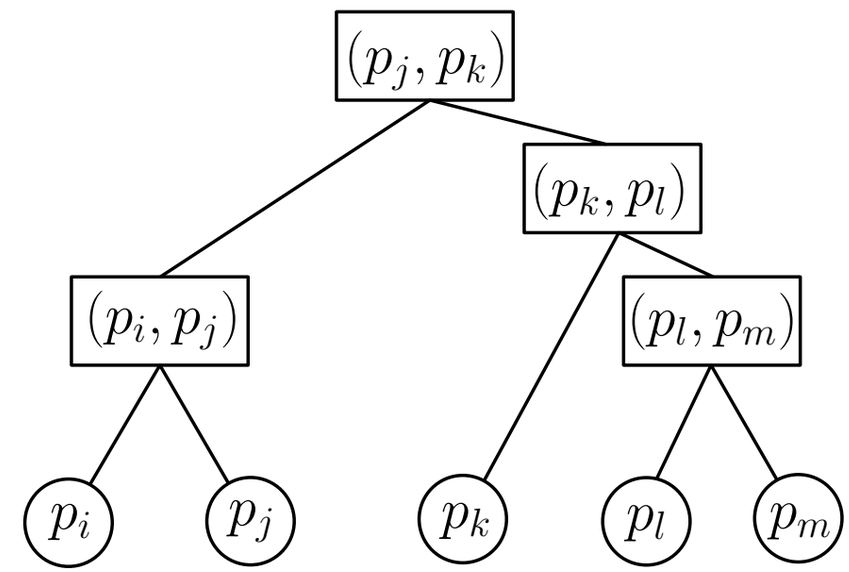
- *Plane sweep*: Beach line in x-structure.
- x-structure: balanced binary search tree B.
- Leaves = parabolic arcs.
- Order: left to right

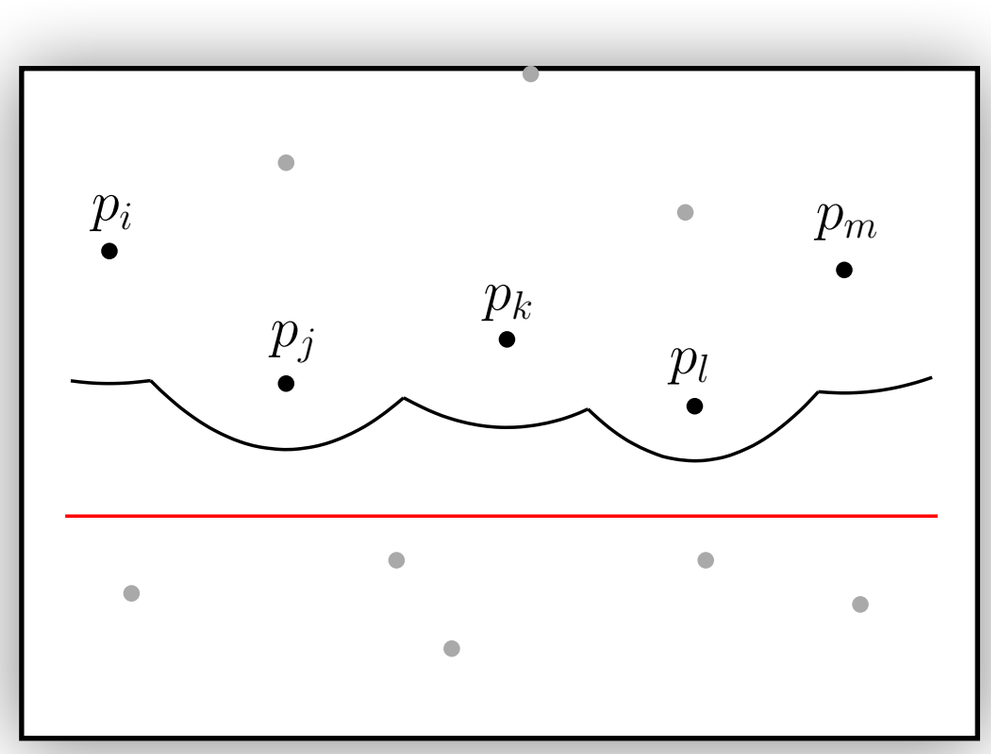




Approach [Fortune, 1987]:

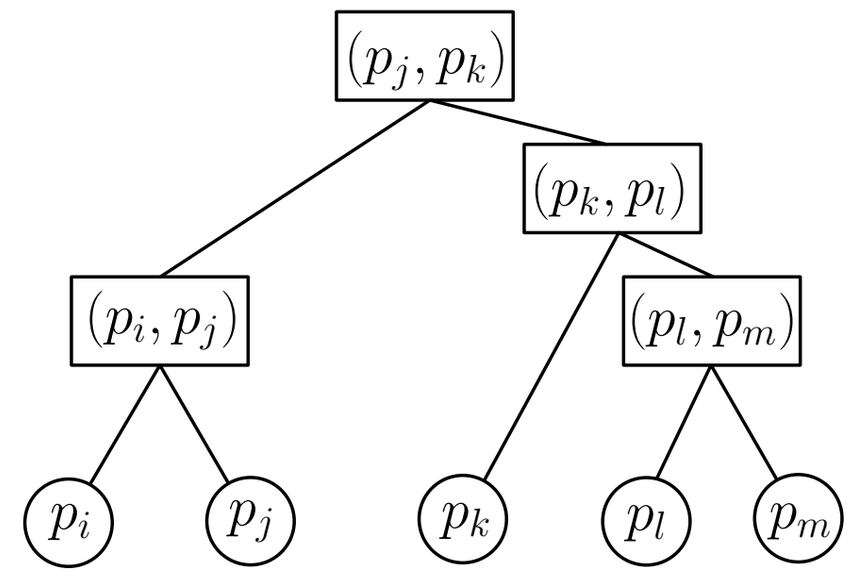
- *Plane sweep*: Beach line in x-structure.
- x-structure: balanced binary search tree B.
- Leaves = parabolic arcs.
- Order: left to right
- Inner nodes: intersection points between adjacent arcs

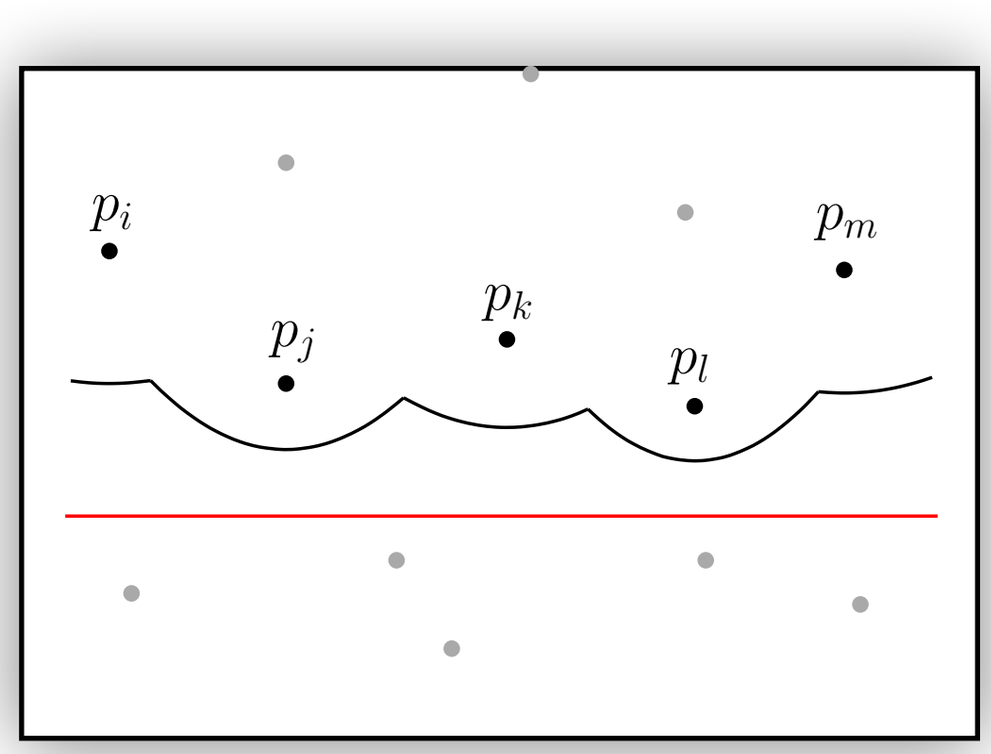




Approach [Fortune, 1987]:

- *Plane sweep*: Beach line in x-structure.
- x-structure: balanced binary search tree B.
- Leaves = parabolic arcs.
- Order: left to right
- Inner nodes: intersection points between adjacent arcs
- Representation of arcs: implicitly by defining points





Fortune’s Algorithm - II

Storing Events:

Storing Events:

- Point events known (and sorted) in advance

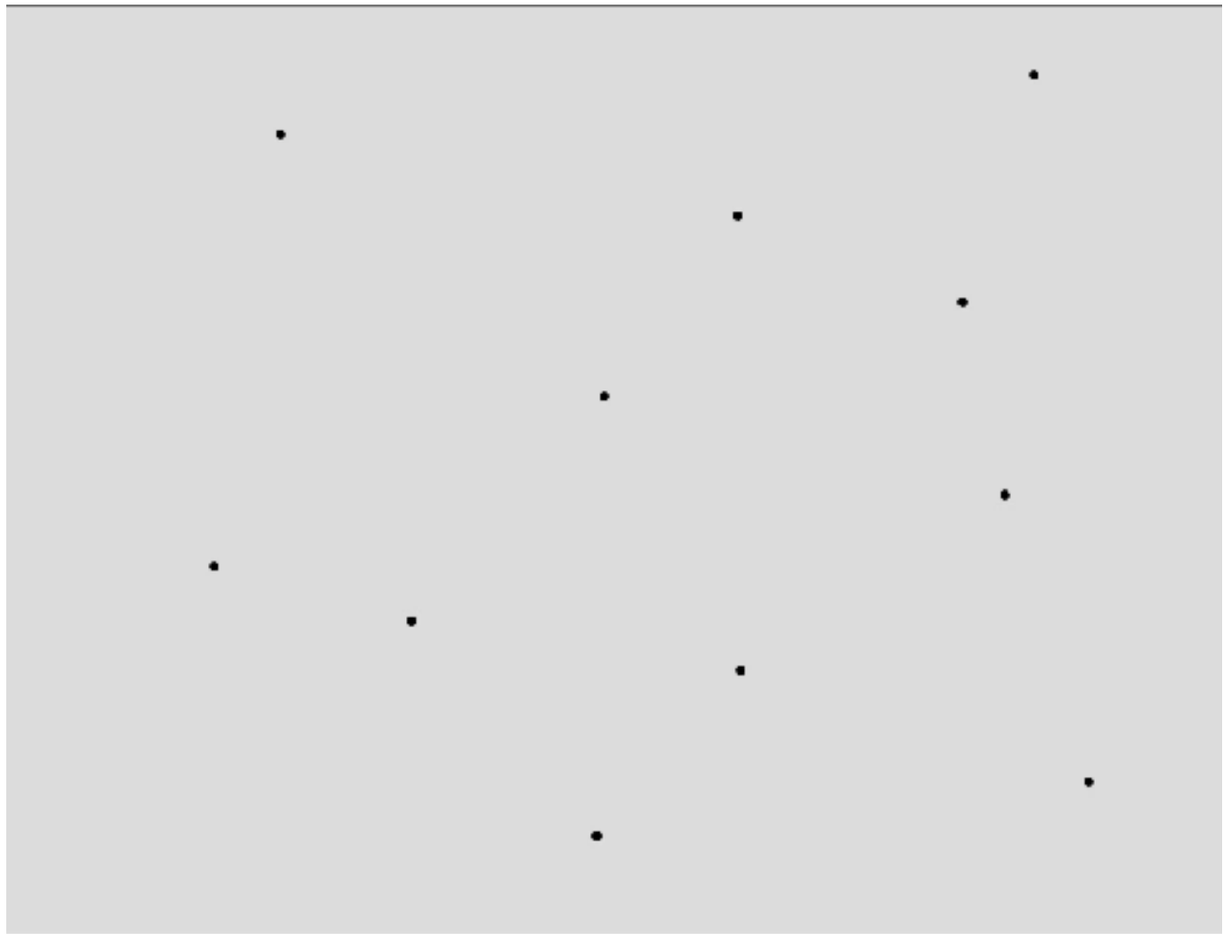
Storing Events:

- Point events known (and sorted) in advance
- Circle events: recognized during runtime

Fortune’s Algorithm - II

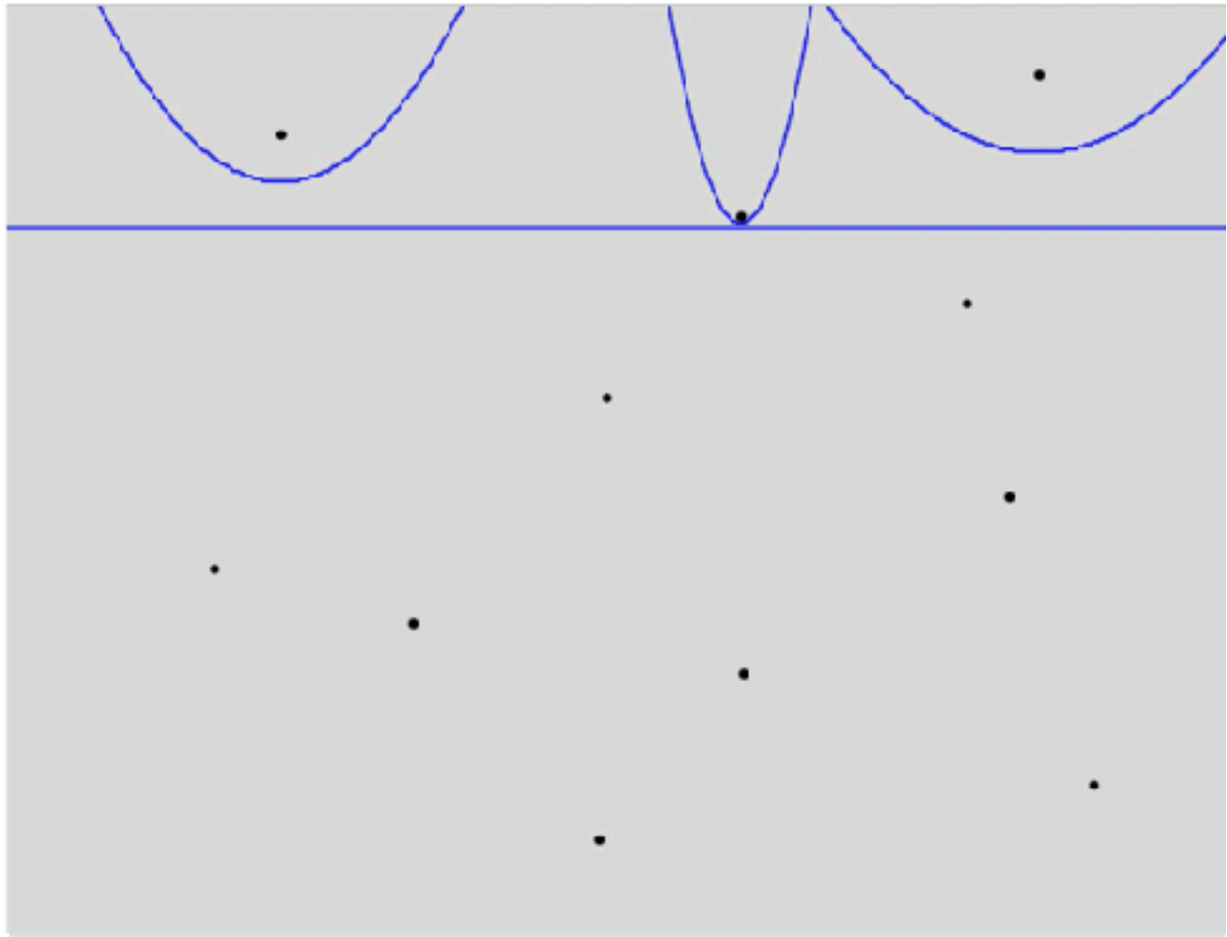
Storing Events:

- Point events known (and sorted) in advance
- Circle events: recognized during runtime



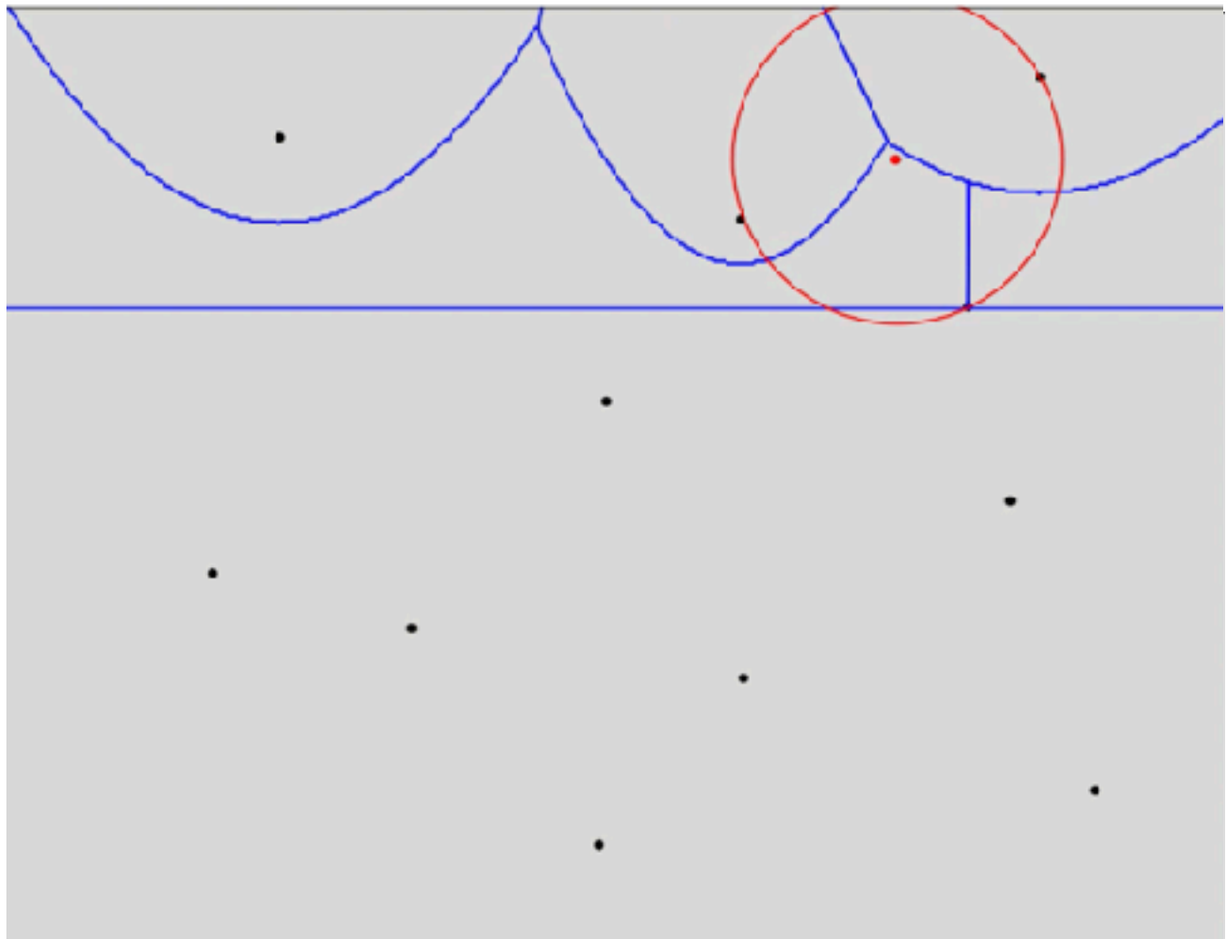
Storing Events:

- Point events known (and sorted) in advance
- Circle events: recognized during runtime



Storing Events:

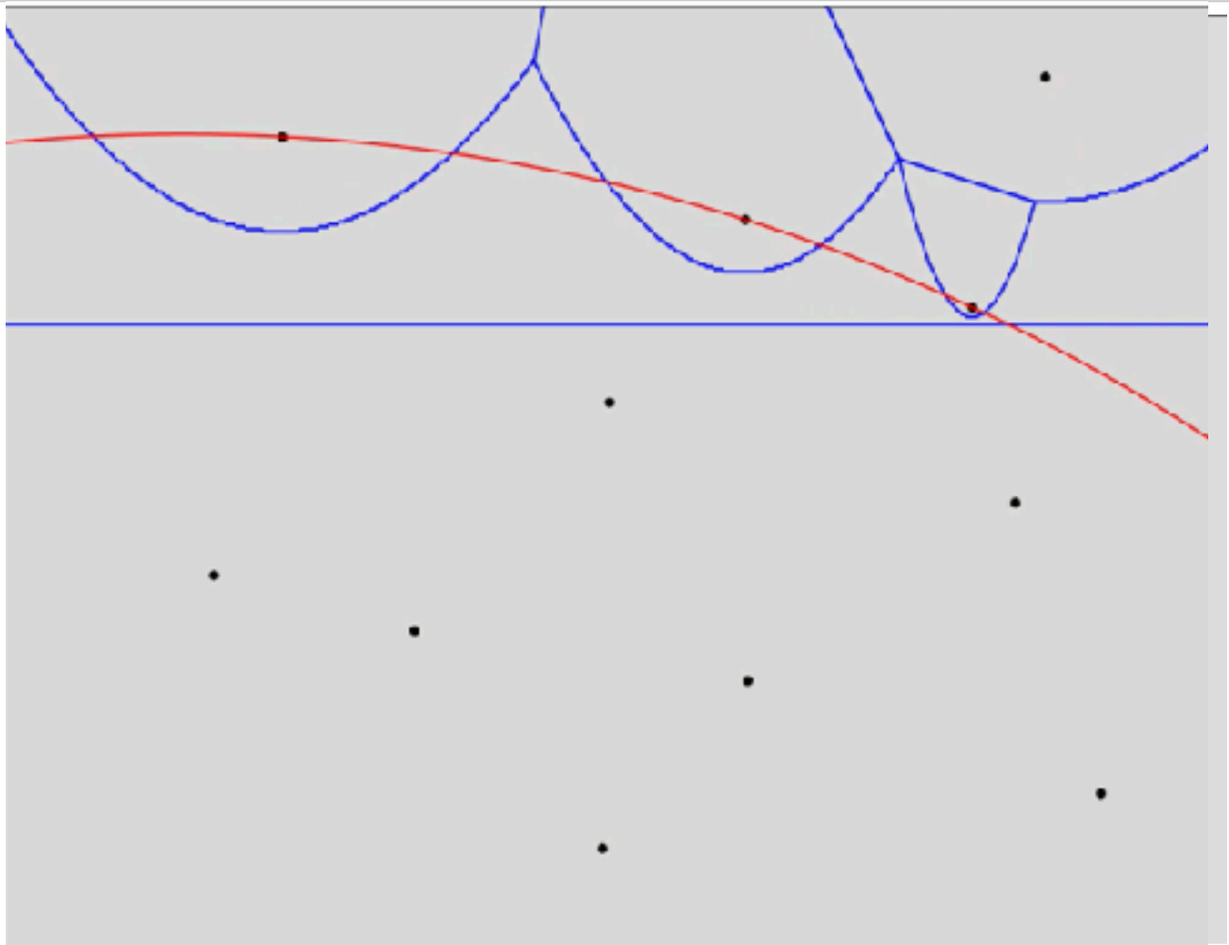
- Point events known (and sorted) in advance
- Circle events: recognized during runtime



Fortune's Algorithm - II

Storing Events:

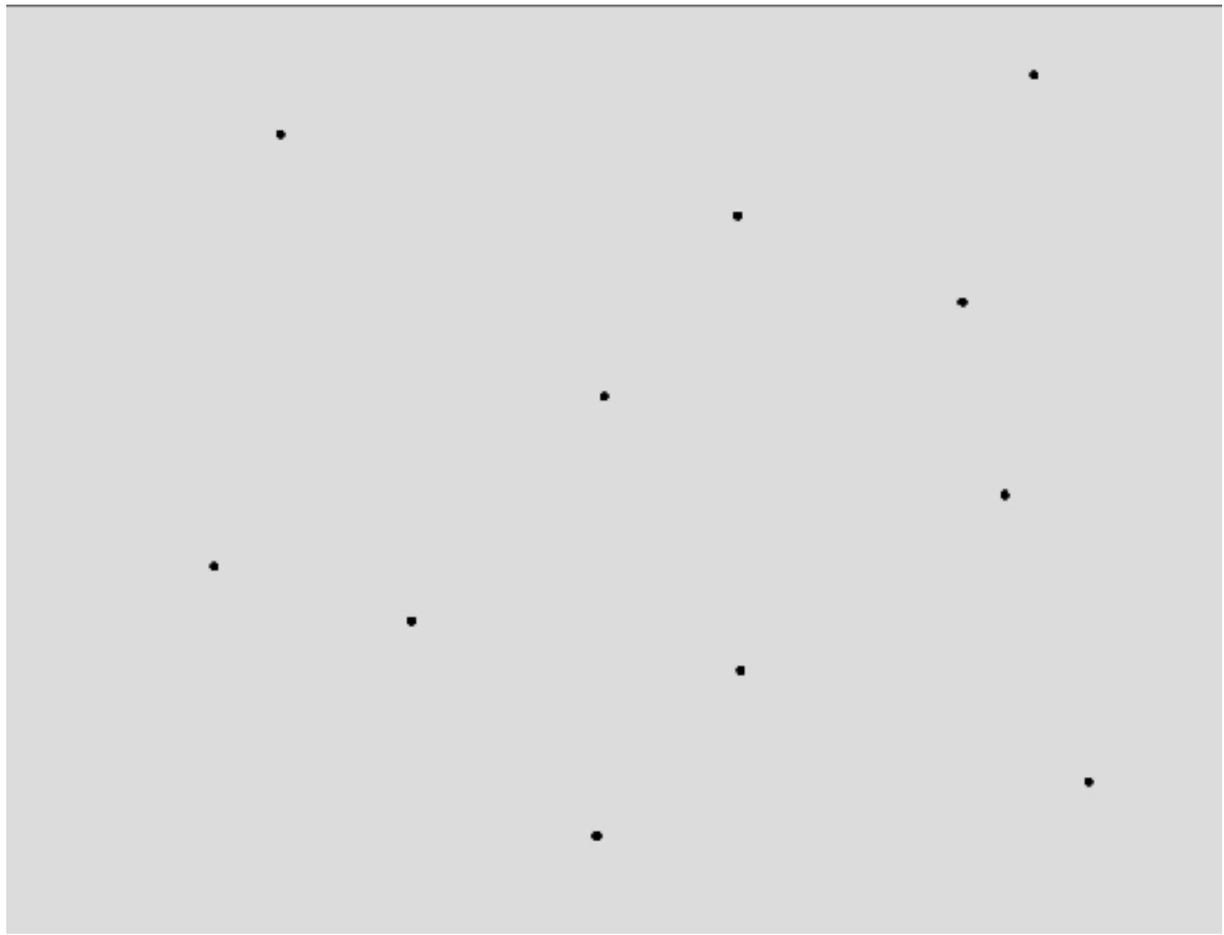
- Point events known (and sorted) in advance
- Circle events: recognized during runtime



Fortune’s Algorithm - II

Storing Events:

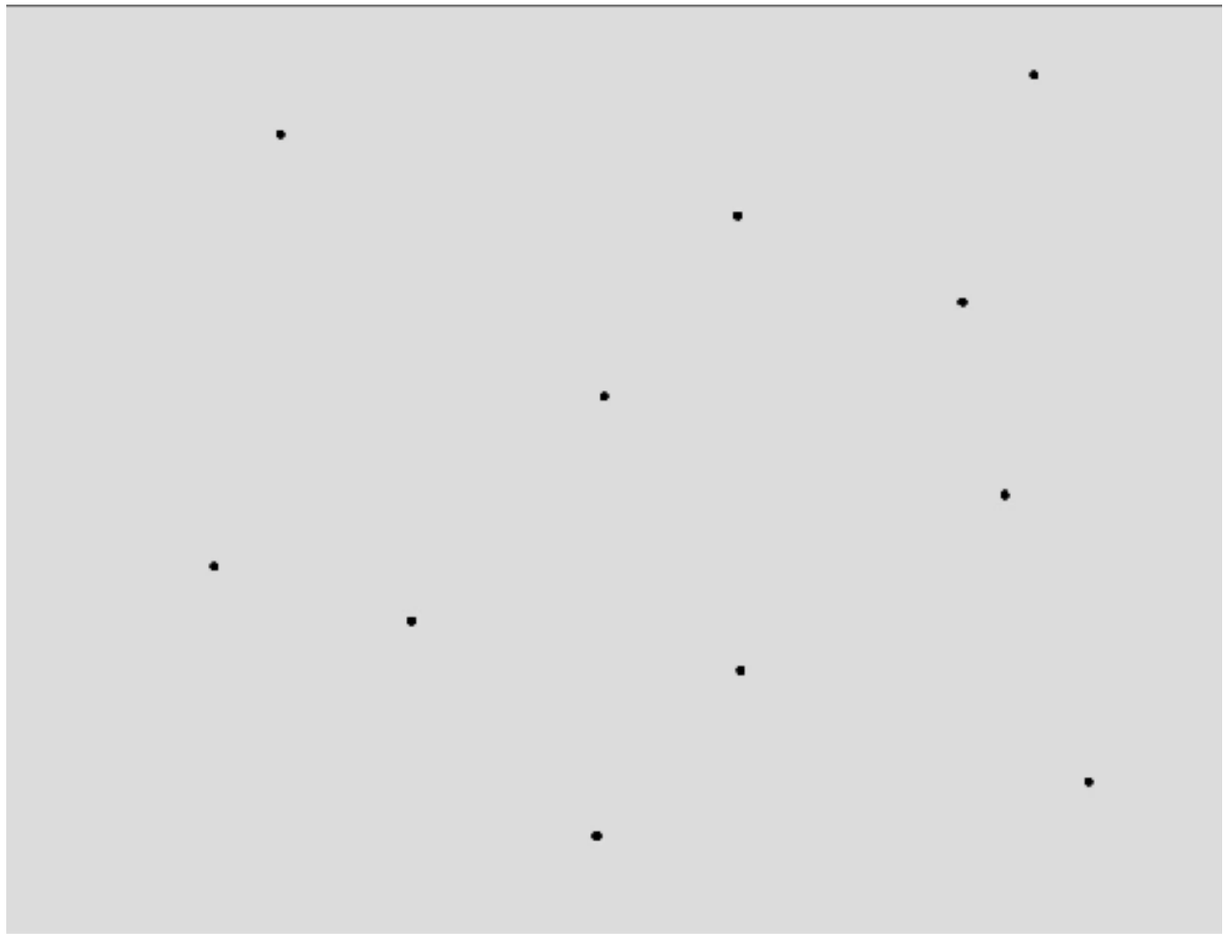
- Point events known (and sorted) in advance
- Circle events: recognized during runtime



Fortune’s Algorithm - II

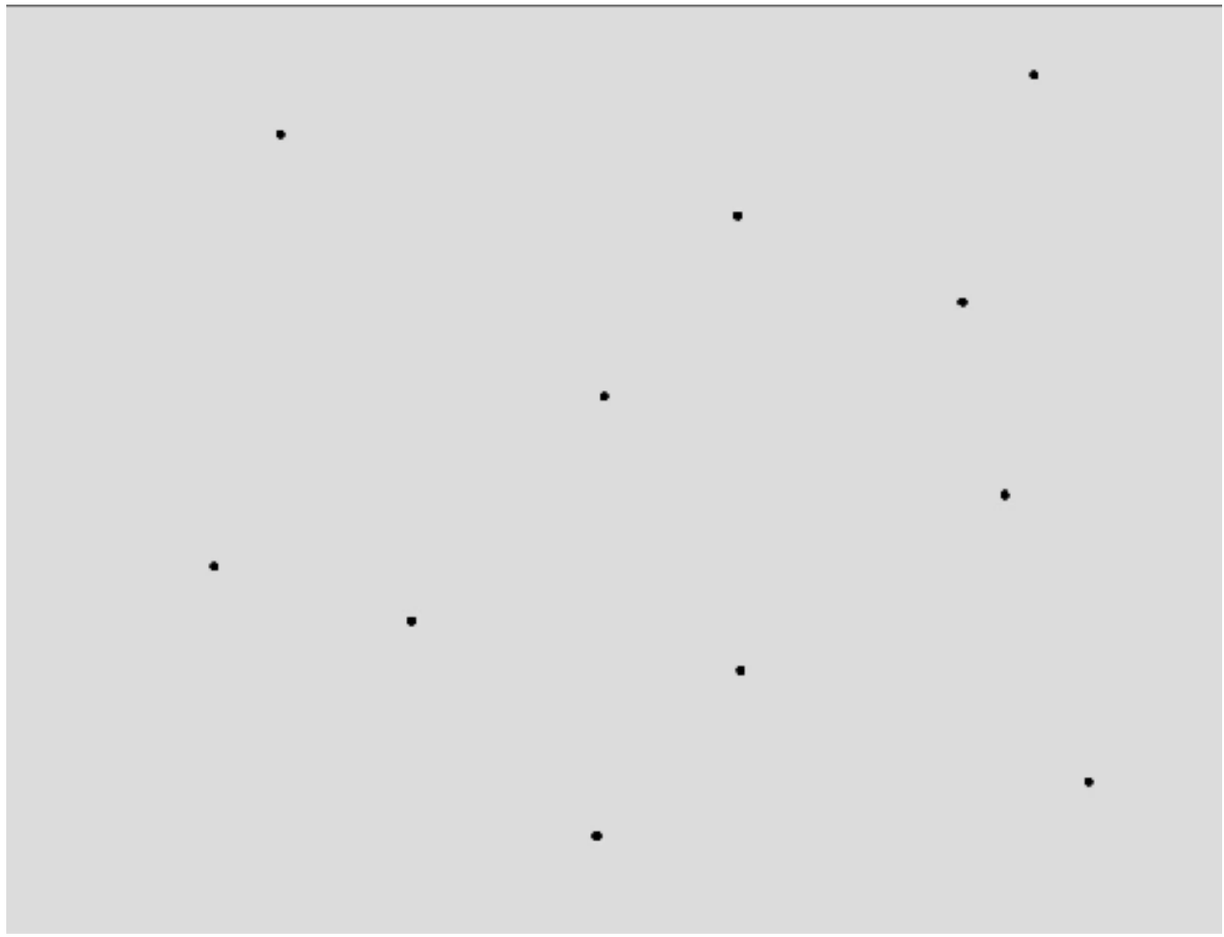
Storing Events:

- Point events known (and sorted) in advance
- Circle events: recognized during runtime

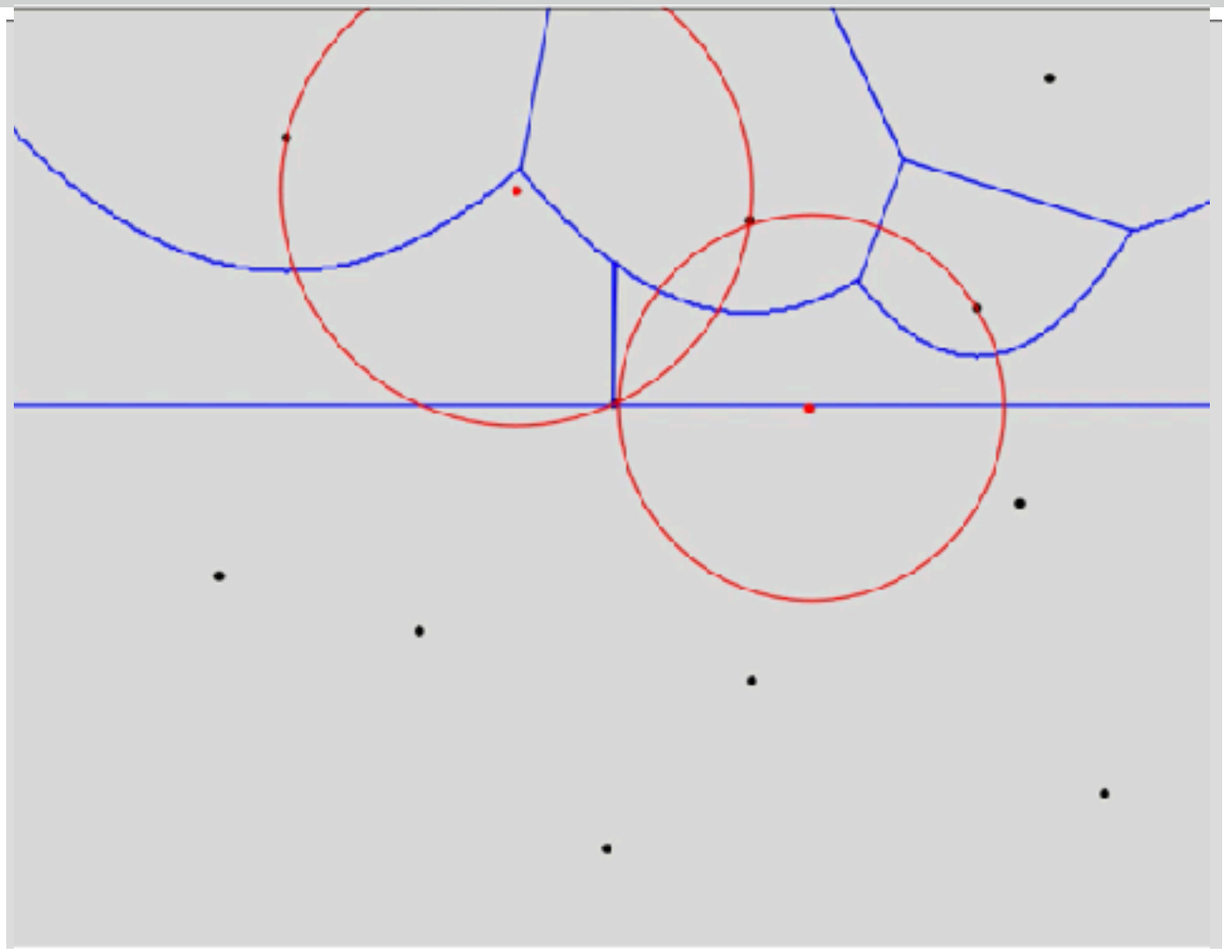


Storing Events:

- Point events known (and sorted) in advance
- Circle events: recognized during runtime
- Each event (inserting or deleting an arc into/from the beach line) creates at most three new consecutive triples of arcs, so update in constant time!



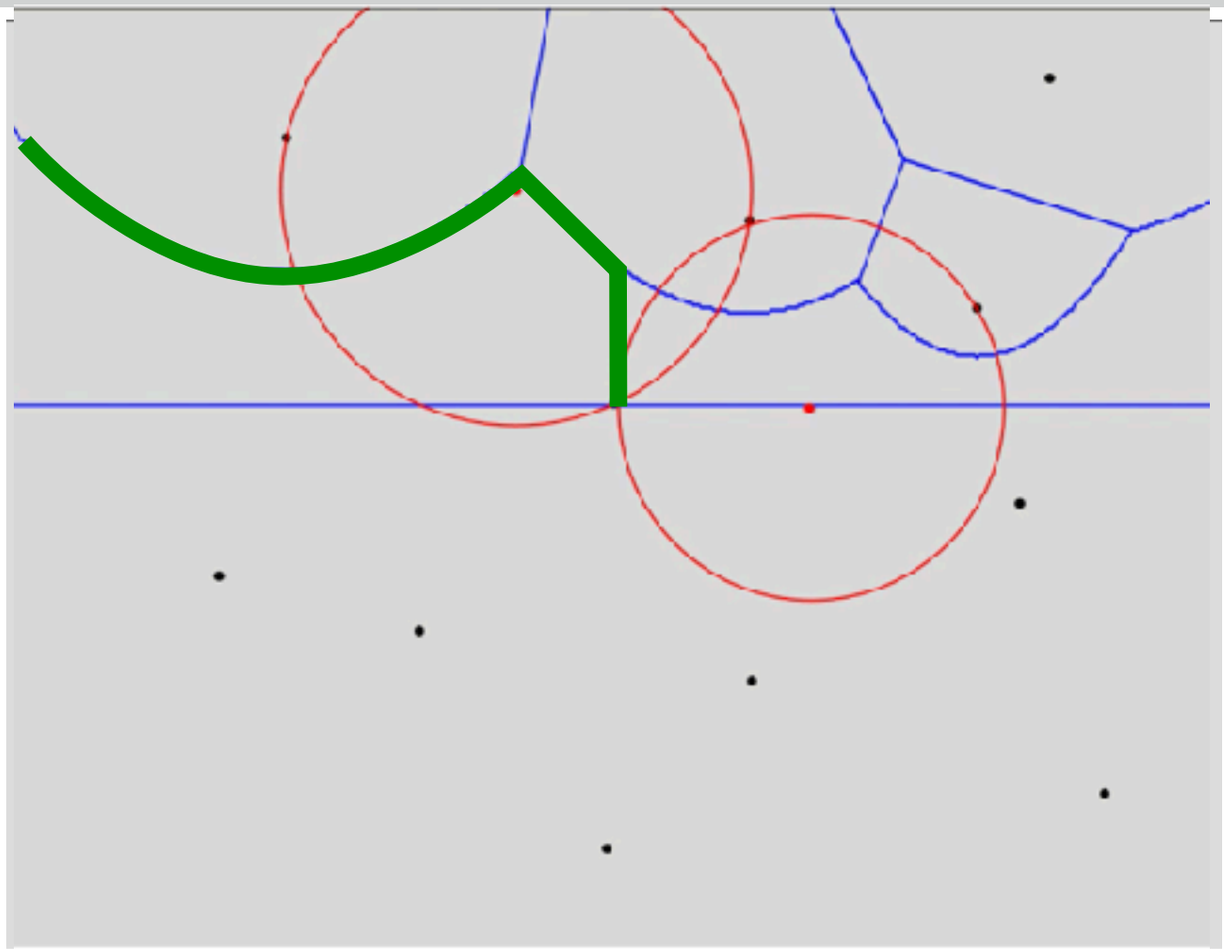
Fortune's Algorithm - II



Storing Events:

- Point events known (and sorted) in advance
- Circle events: recognized during runtime
- Each event (inserting or deleting an arc into/from the beach line) creates at most three new consecutive triples of arcs, so update in constant time!

Fortune's Algorithm - II

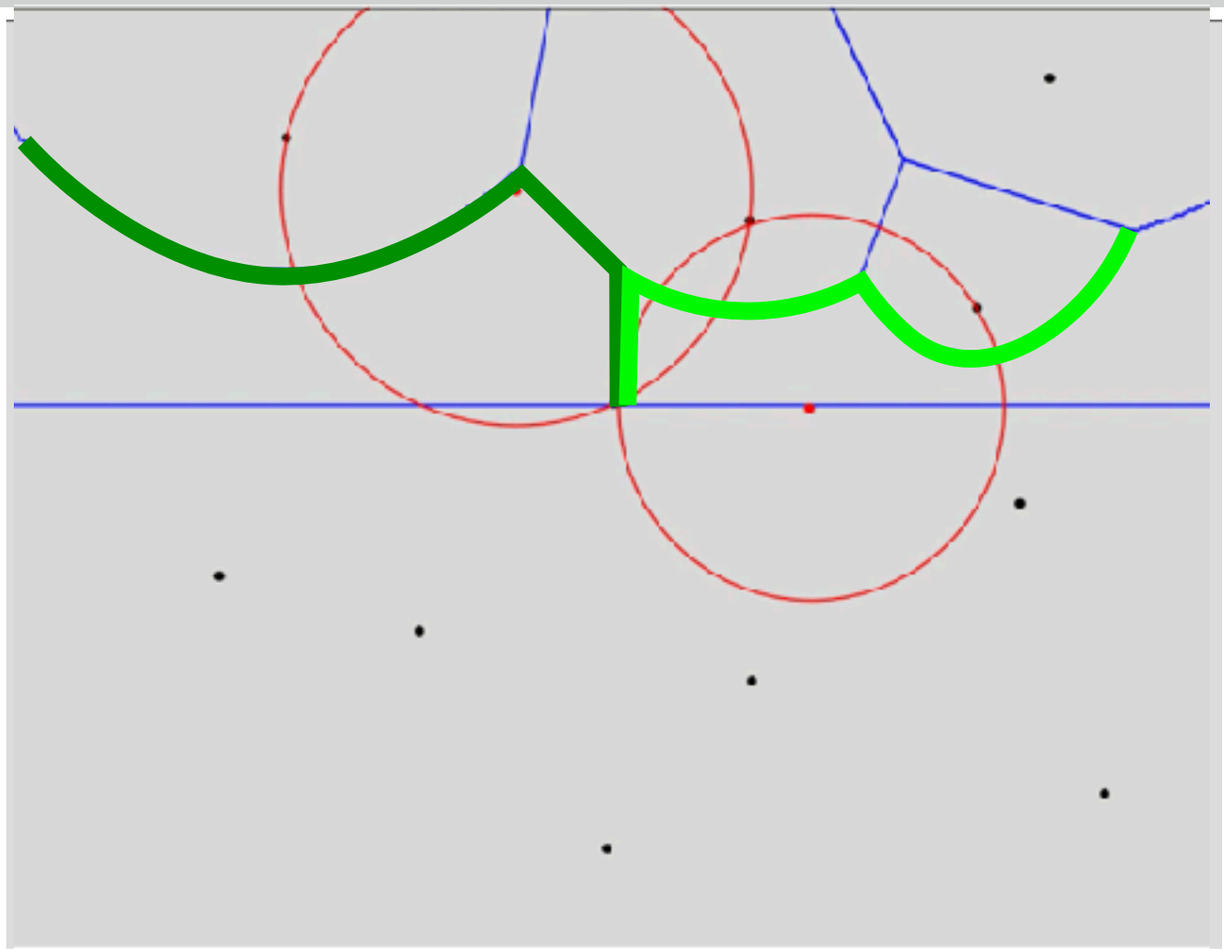


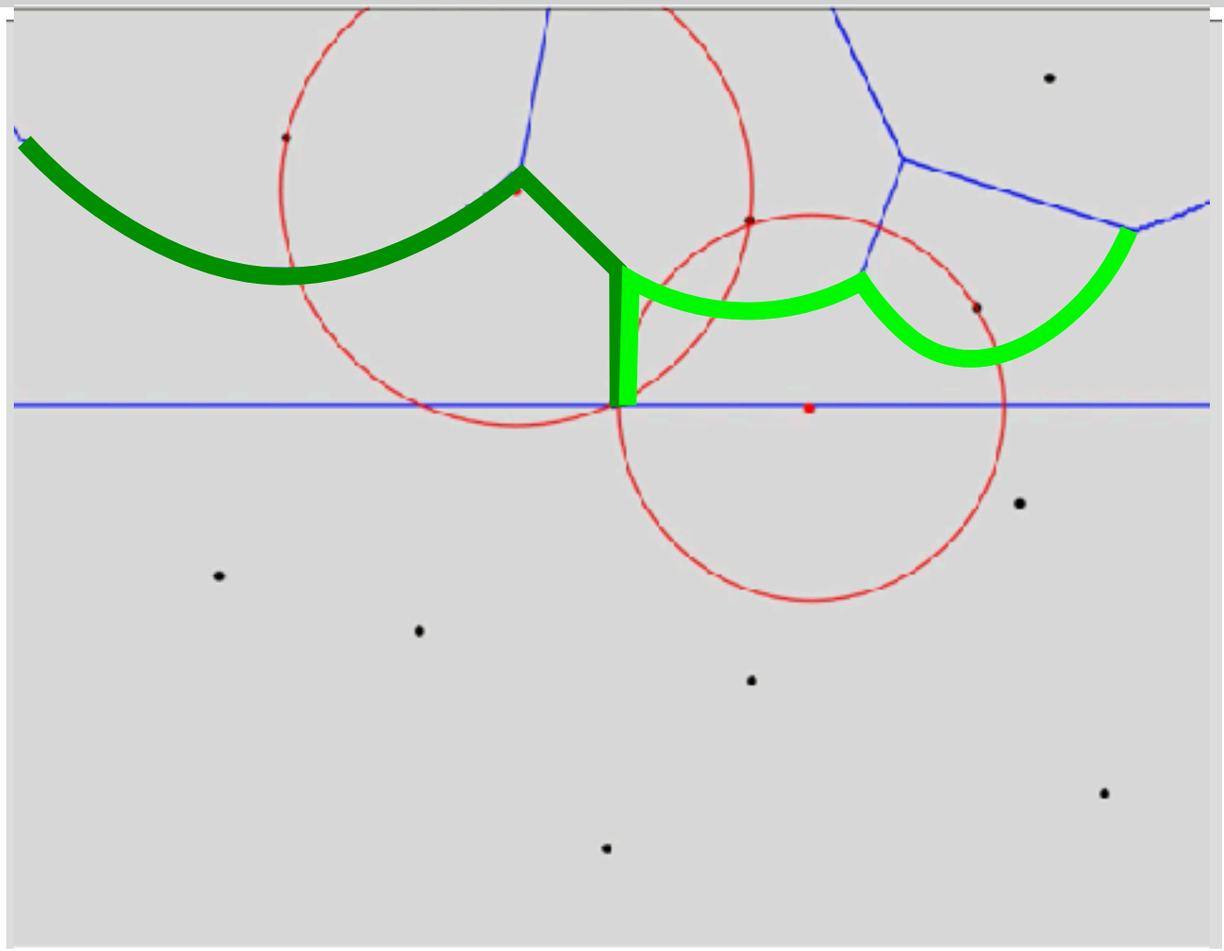
Storing Events:

- Point events known (and sorted) in advance
- Circle events: recognized during runtime
- Each event (inserting or deleting an arc into/from the beach line) creates at most three new consecutive triples of arcs, so update in constant time!

Storing Events:

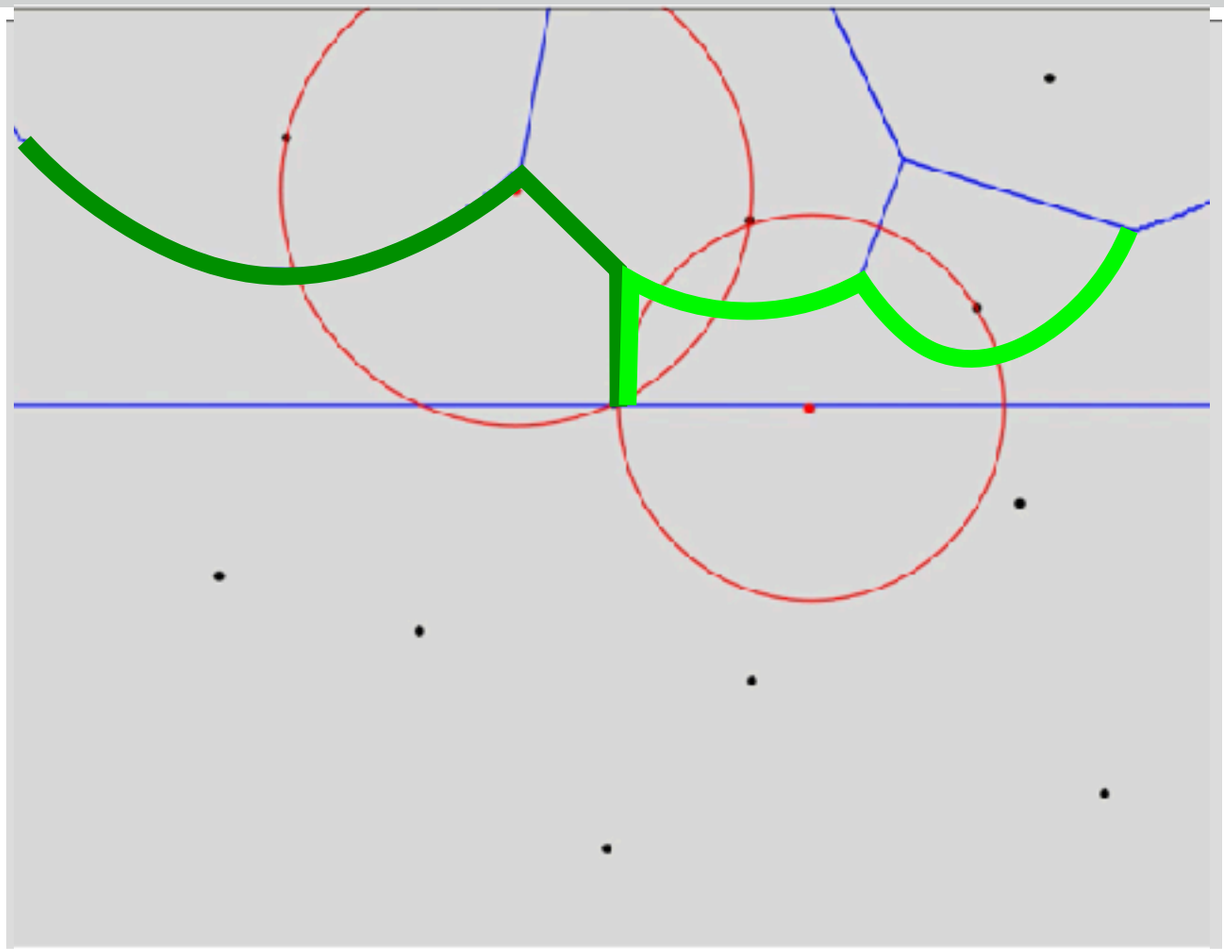
- Point events known (and sorted) in advance
- Circle events: recognized during runtime
- Each event (inserting or deleting an arc into/from the beach line) creates at most three new consecutive triples of arcs, so update in constant time!





Storing Events:

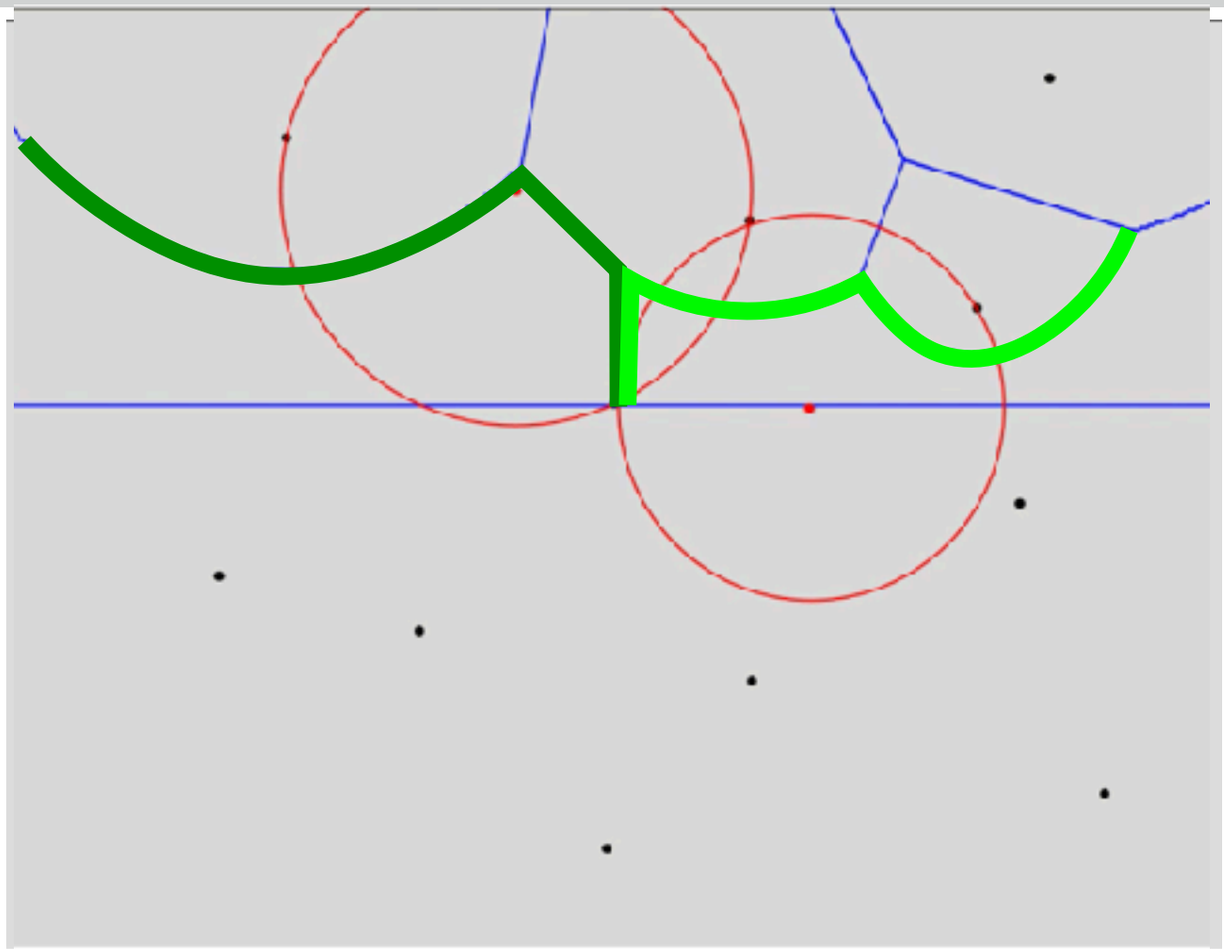
- Point events known (and sorted) in advance
- Circle events: recognized during runtime
- Each event (inserting or deleting an arc into/from the beach line) creates at most three new consecutive triples of arcs, so update in constant time!
- Priority queue: decending wrt. $\leq_y$



Storing Events:

- Point events known (and sorted) in advance
- Circle events: recognized during runtime
- Each event (inserting or deleting an arc into/from the beach line) creates at most three new consecutive triples of arcs, so update in constant time!
- Priority queue: decending wrt. $\leq_y$

  Point events $p_i \rightarrow$ Priority by $p_i.y$.

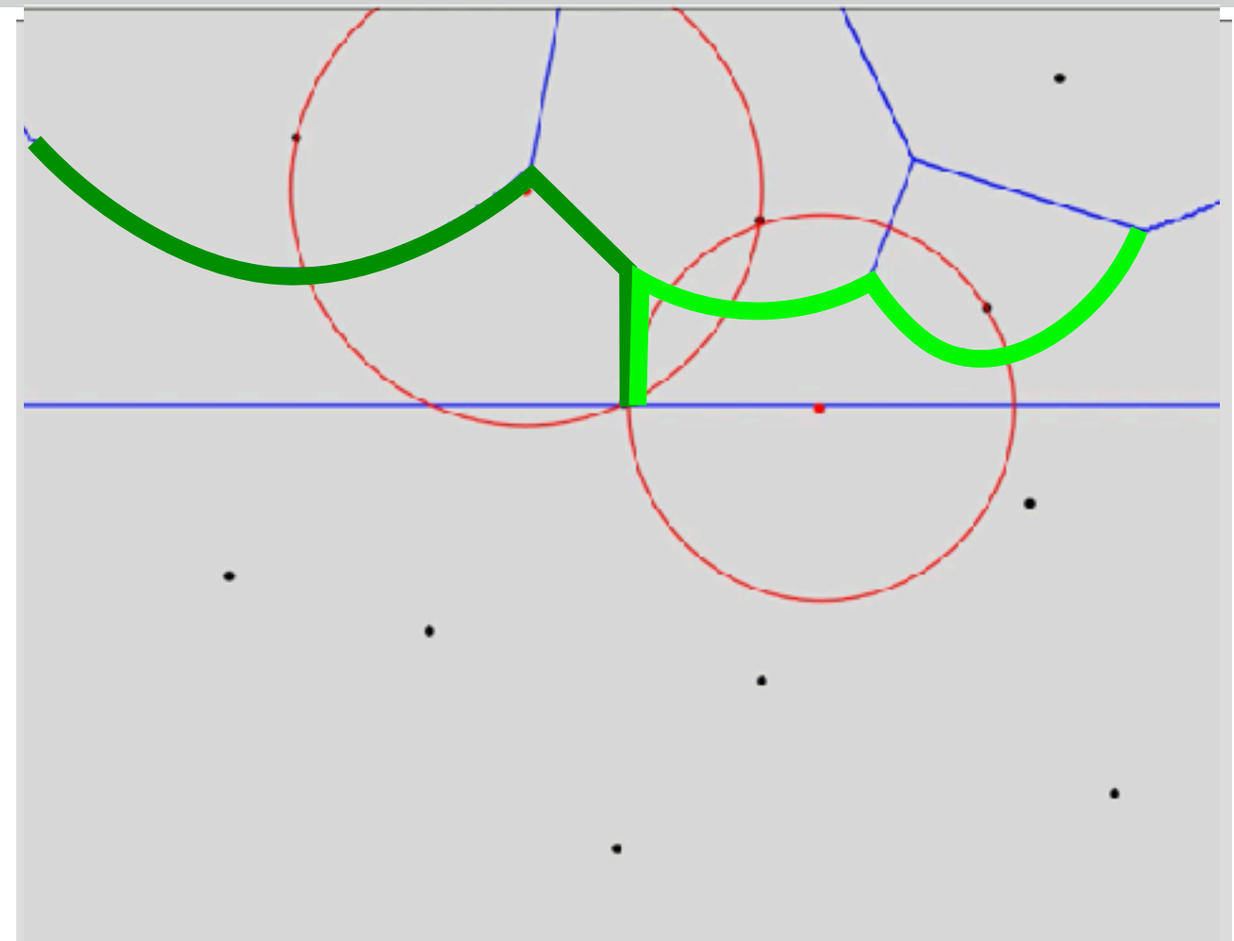


Storing Events:

- Point events known (and sorted) in advance
- Circle events: recognized during runtime
- Each event (inserting or deleting an arc into/from the beach line) creates at most three new consecutive triples of arcs, so update in constant time!
- Priority queue: decending wrt. $\leq_y$

  Point events $p_i \rightarrow$ Priority by $p_i.y$.

  Circle events C .

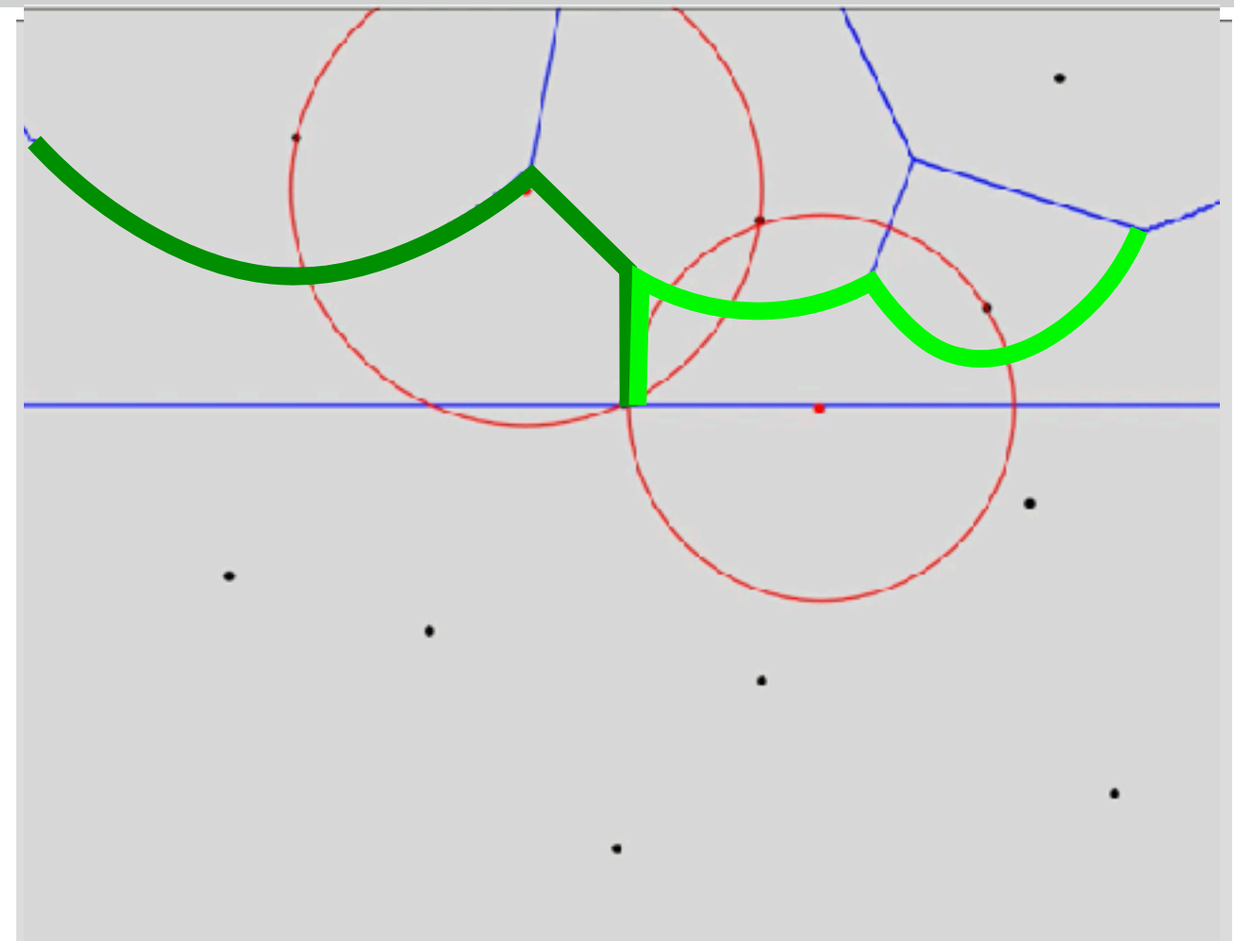


Storing Events:

- Point events known (and sorted) in advance
- Circle events: recognized during runtime
- Each event (inserting or deleting an arc into/from the beach line) creates at most three new consecutive triples of arcs, so update in constant time!
- Priority queue: decending wrt. $\leq_y$

  Point events $p_i \rightarrow$ Priority by $p_i.y$.

  Circle events $C \rightarrow$ Priority by y-coordinate of lowest point of C.

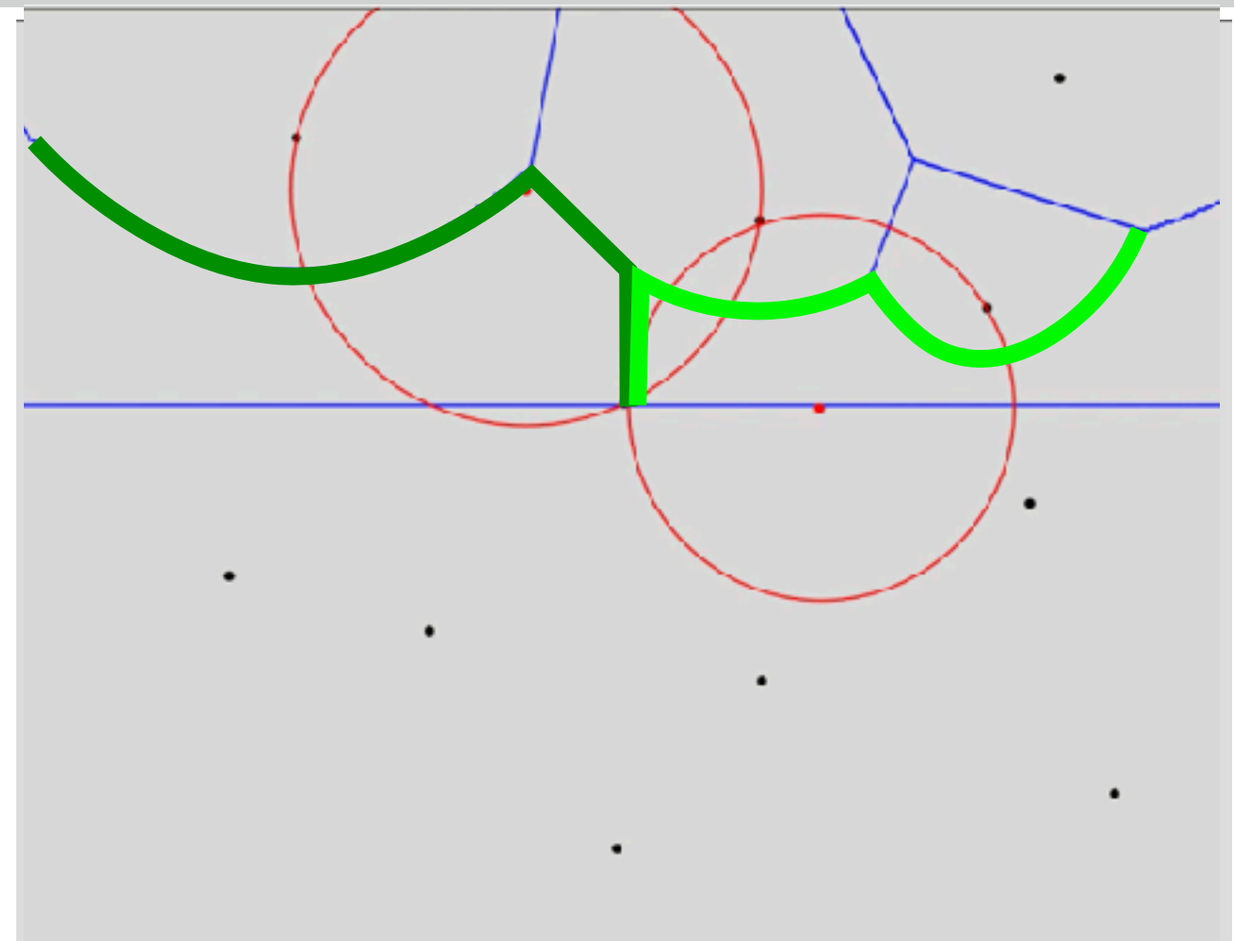


Storing Events:

- Point events known (and sorted) in advance
- Circle events: recognized during runtime
- Each event (inserting or deleting an arc into/from the beach line) creates at most three new consecutive triples of arcs, so update in constant time!
- Priority queue: decending wrt. $\leq_y$

  Point events $p_i \rightarrow$ Priority by $p_i.y$.

  Circle events $C \rightarrow$ Priority by y-coordinate of lowest point of C.
  $\rightarrow$ Pointer to arc (representing leaf $b \in B$) which may disappear.



Storing Events:

- Point events known (and sorted) in advance
- Circle events: recognized during runtime
- Each event (inserting or deleting an arc into/from the beach line) creates at most three new consecutive triples of arcs, so update in constant time!
- Priority queue: decending wrt. $\leq_y$

  Point events $p_i \rightarrow$ Priority by $p_i.y$.

  Circle events $C \rightarrow$ Priority by y-coordinate of lowest point of C.
  $\rightarrow$ Pointer to arc (representing leaf $b \in B$) which may disappear.

- Leaf $b \in B$ points to circle event $C \in Q$ for which arc β of b may disappear.

Validity of Circle Events

Observation 1:

Observation 1:

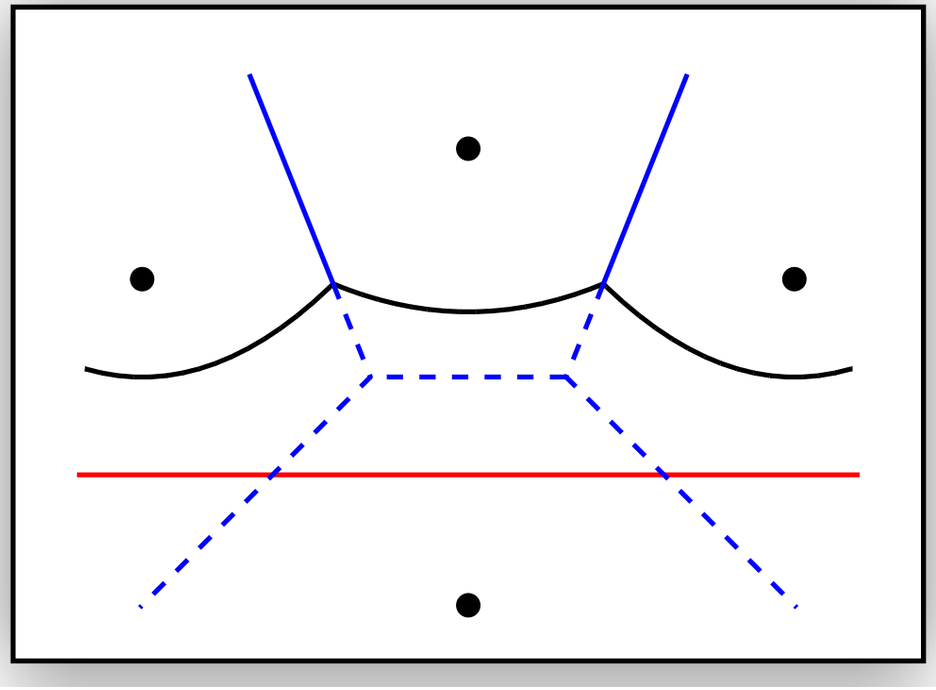
- Discovering $p \in C^\circ$

Observation 1:

- Discovering $p \in C^\circ \Rightarrow$ Circle event C becomes obsolete

Observation 1:

- Discovering $p \in C^\circ \Rightarrow$ Circle event C becomes obsolete

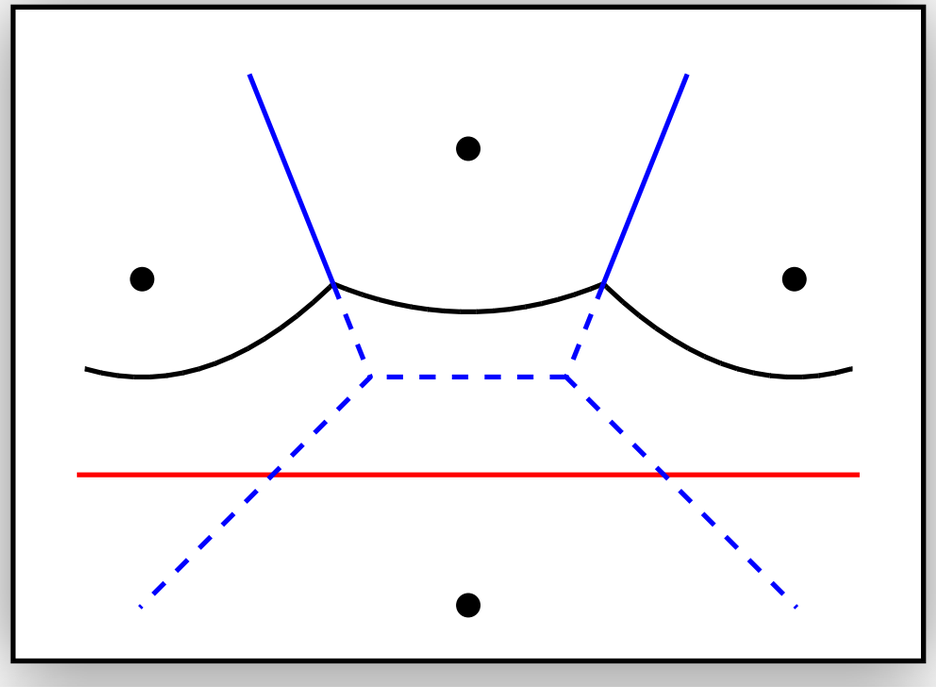


Validity of Circle Events

Observation 1:

- Discovering $p \in C^\circ \Rightarrow$ Circle event C becomes obsolete

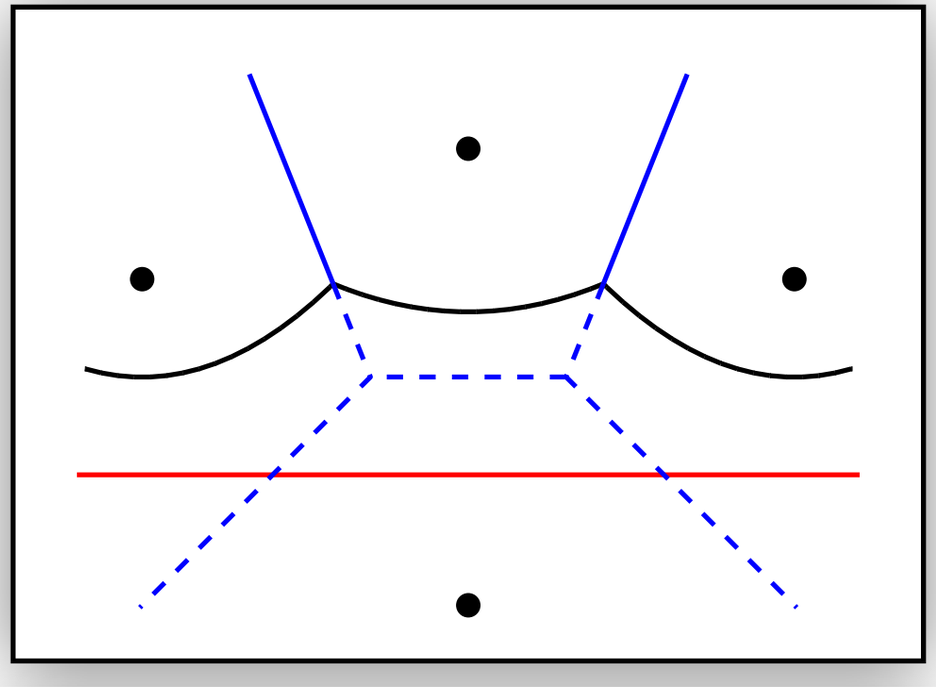
  $\rightarrow$ Parabolic arc must know associated circle event.



Observation 1:

- Discovering $p \in C^\circ \Rightarrow$ Circle event C becomes obsolete

  $\rightarrow$ Parabolic arc must know associated circle event.

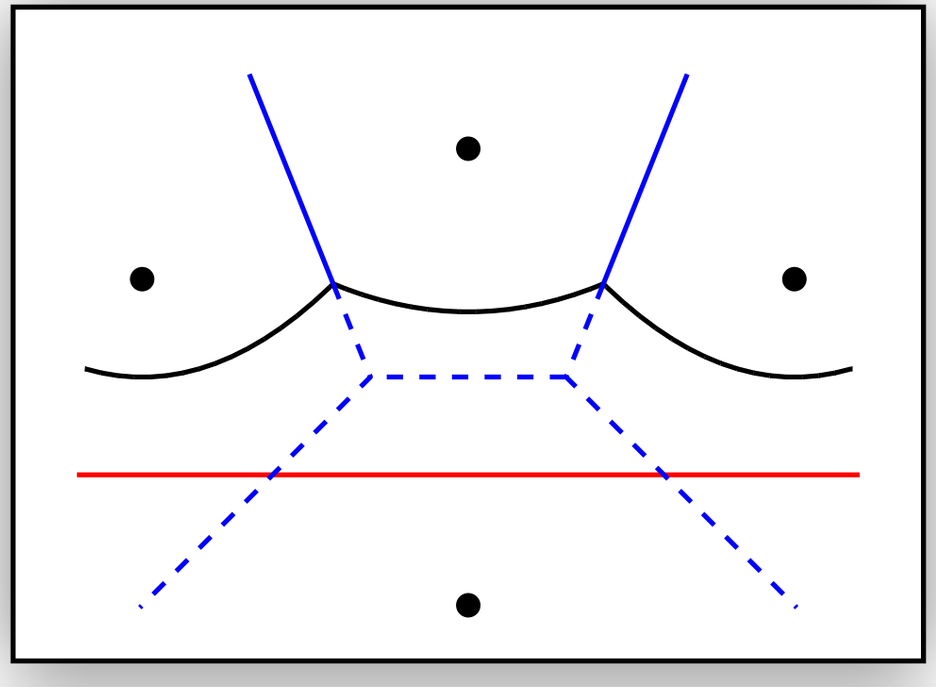


Observation 2:

Observation 1:

- Discovering $p \in C^\circ \Rightarrow$ Circle event C becomes obsolete

  $\rightarrow$ Parabolic arc must know associated circle event.



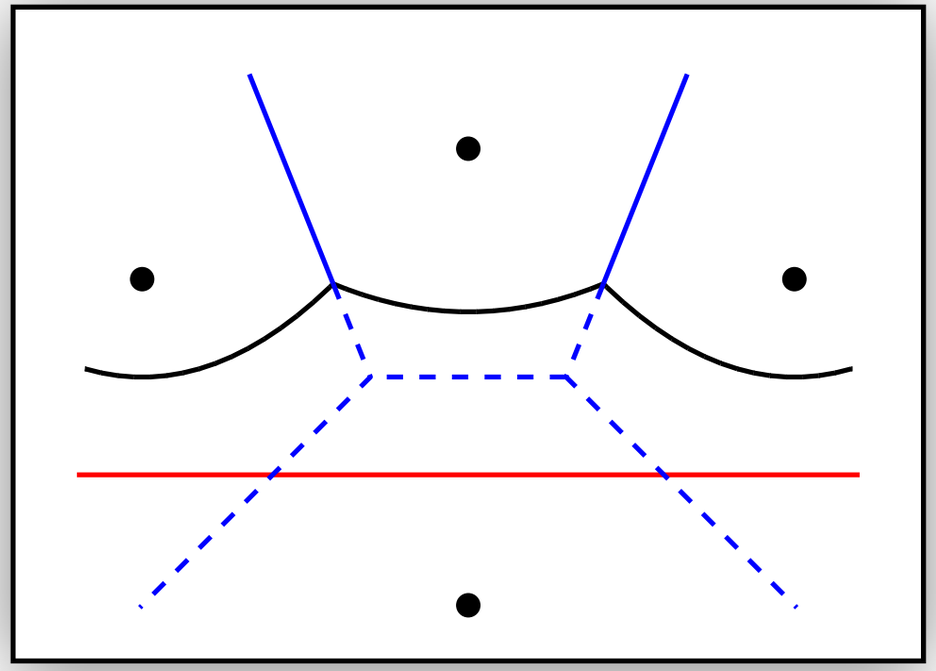
Observation 2:

- Triple of adjacent arcs do not always define a circle event.

Observation 1:

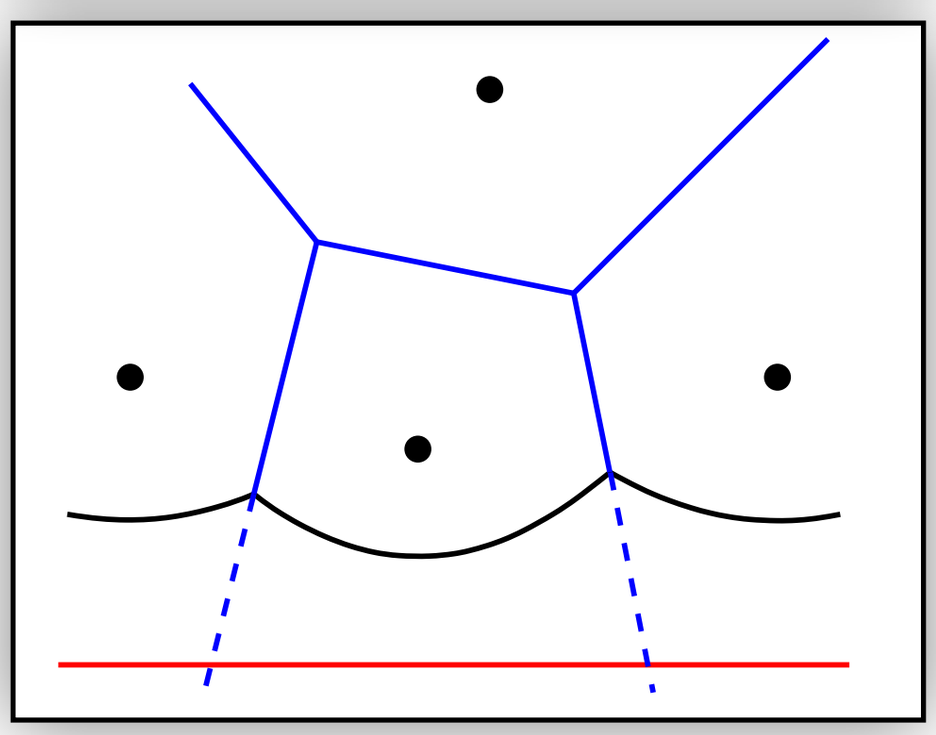
- Discovering $p \in C^\circ \Rightarrow$ Circle event C becomes obsolete

  $\rightarrow$ Parabolic arc must know associated circle event.



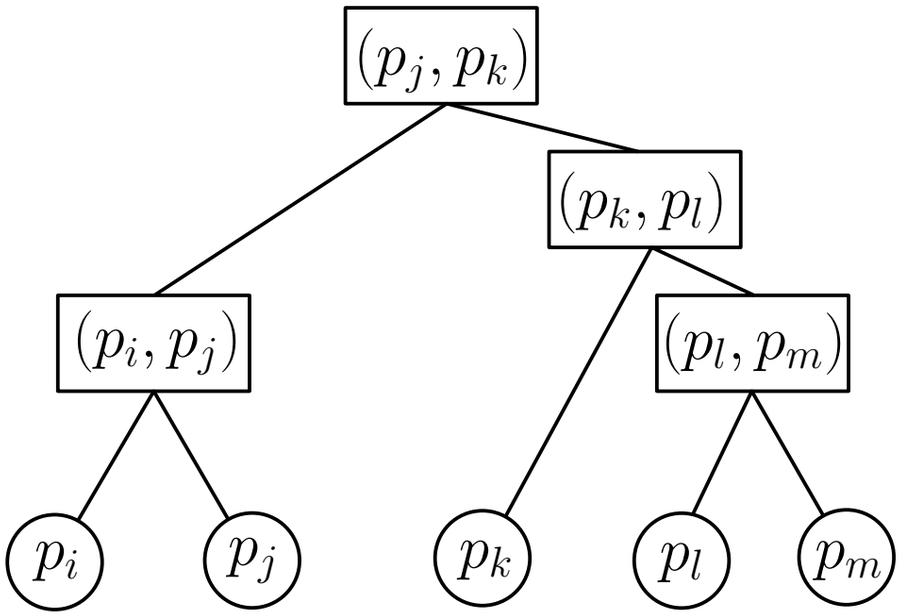
Observation 2:

- Triple of adjacent arcs do not always define a circle event.

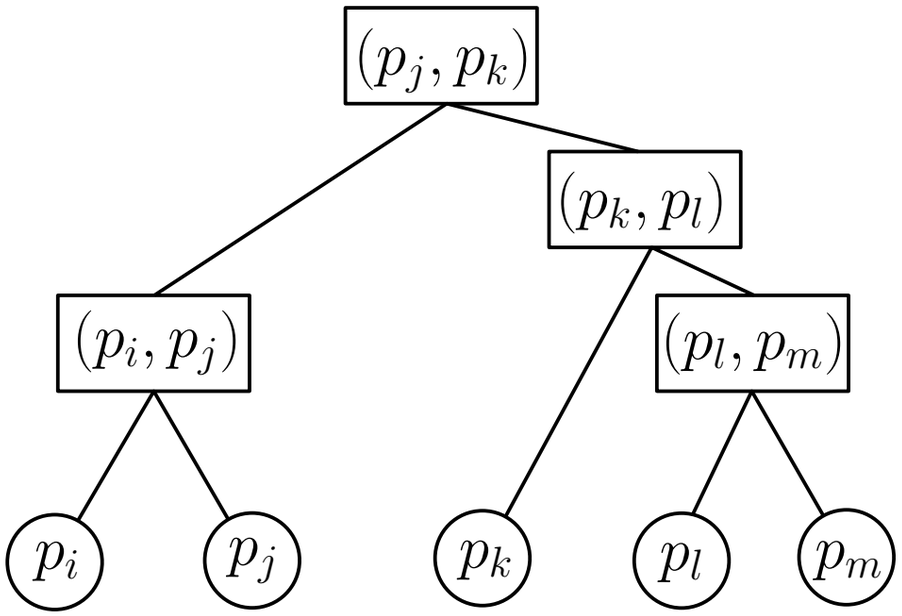


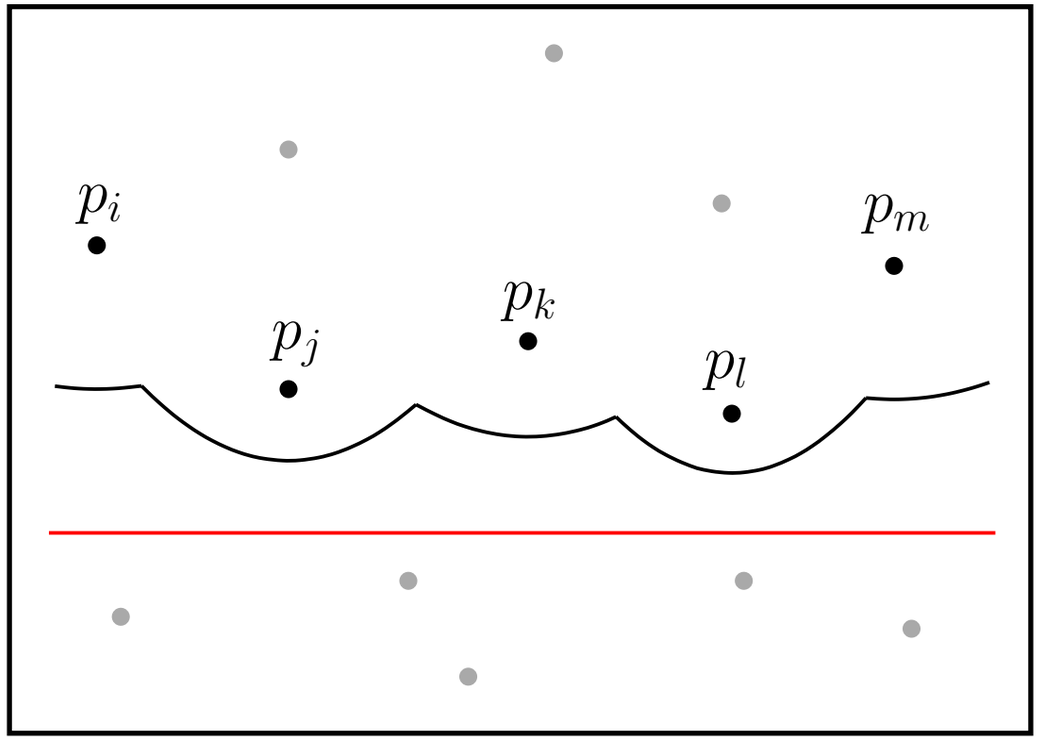
Processing Point Events - I

Processing Point Events - I

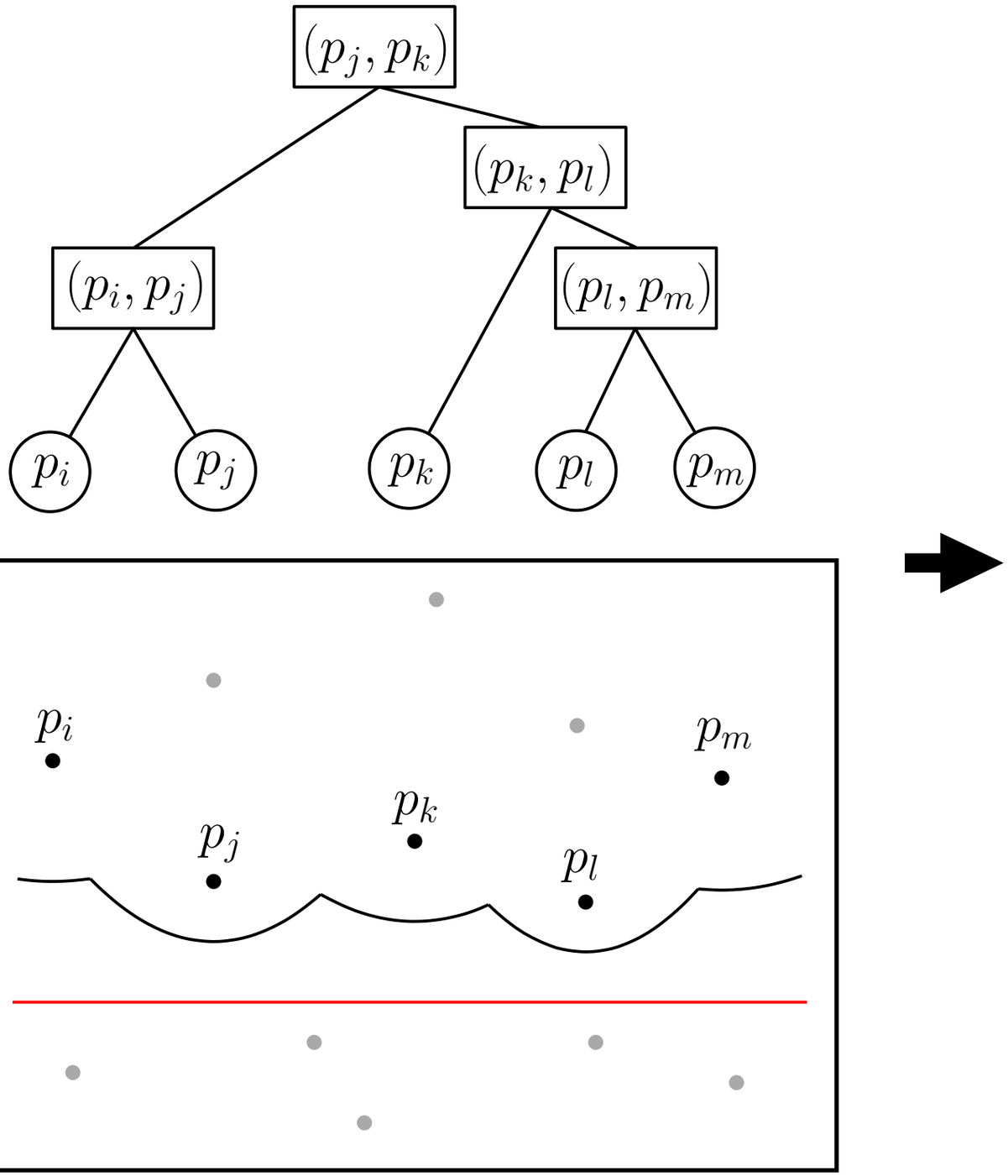


Processing Point Events - I

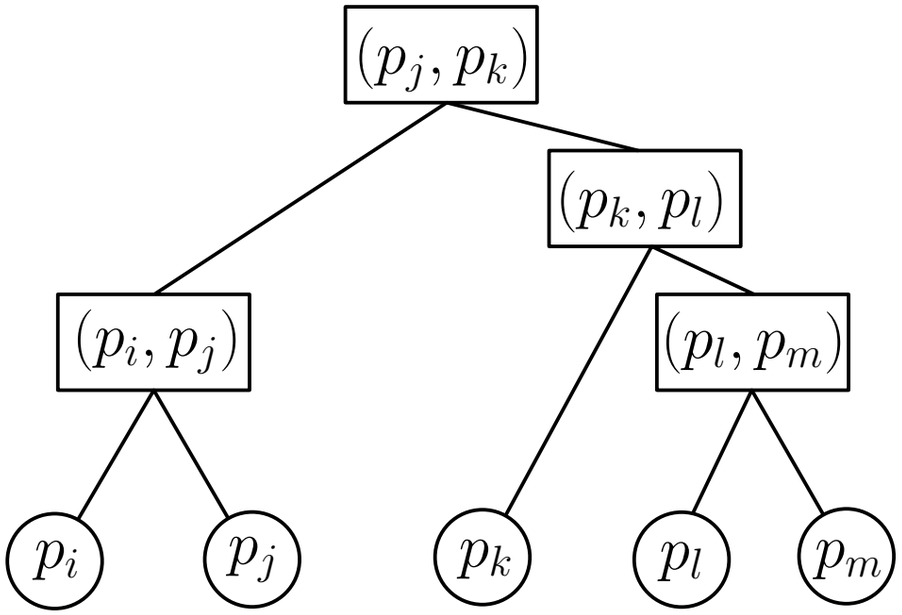


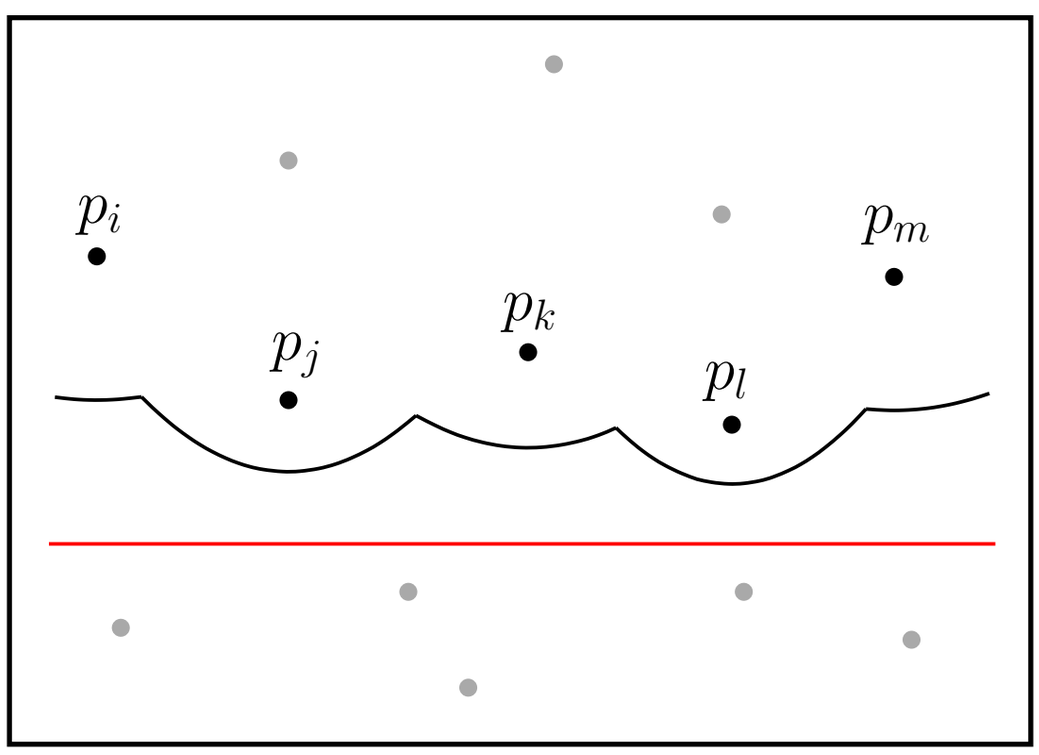


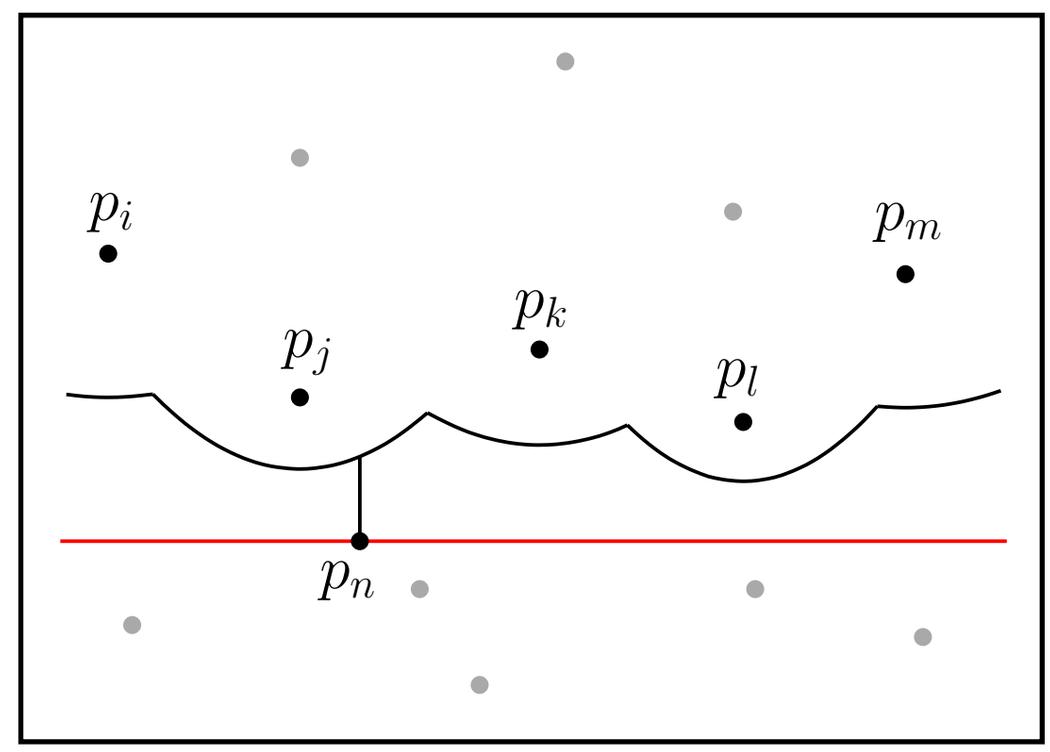
Processing Point Events - I



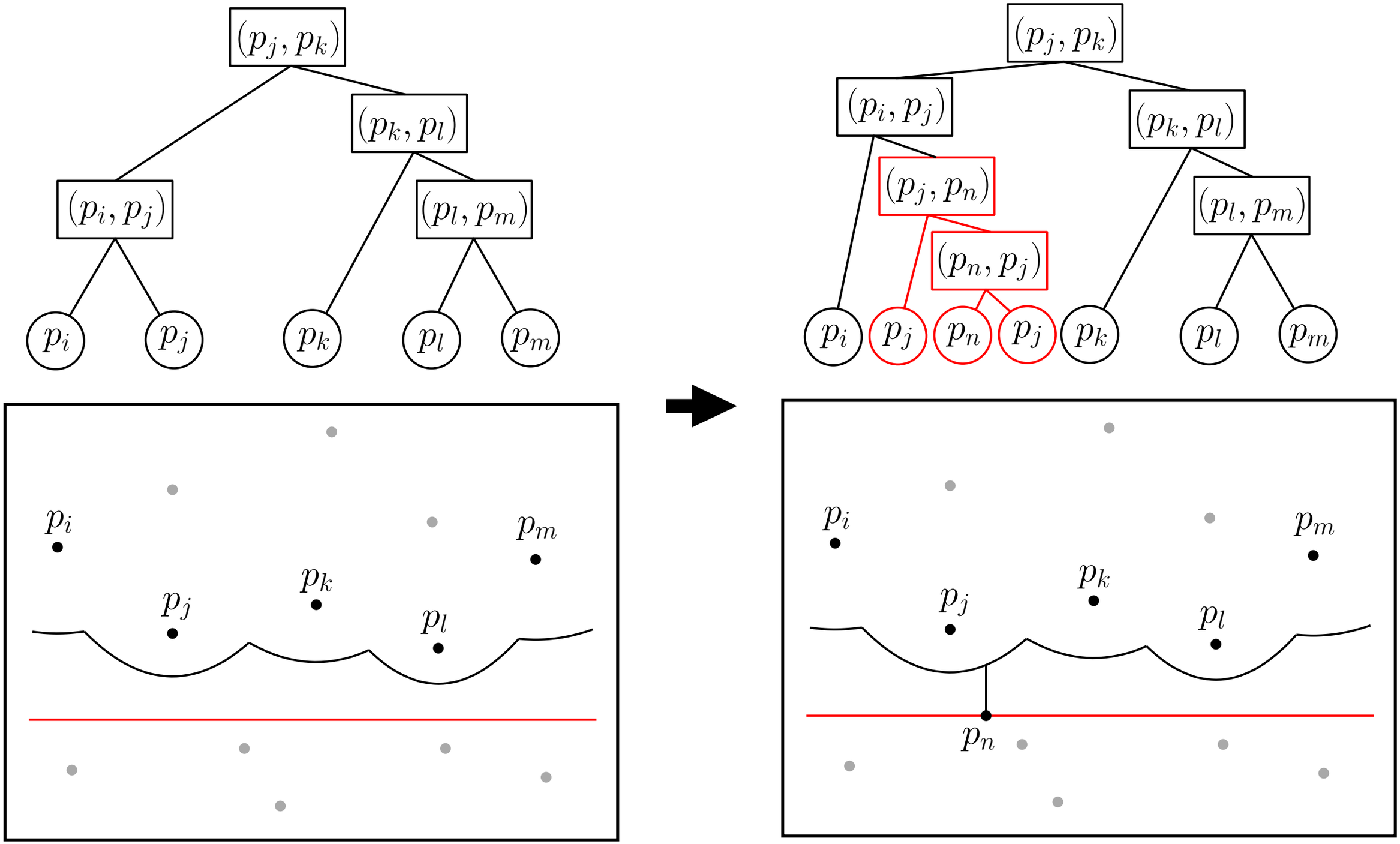
Processing Point Events - I







Processing Point Events - I



Processing Point Events - II

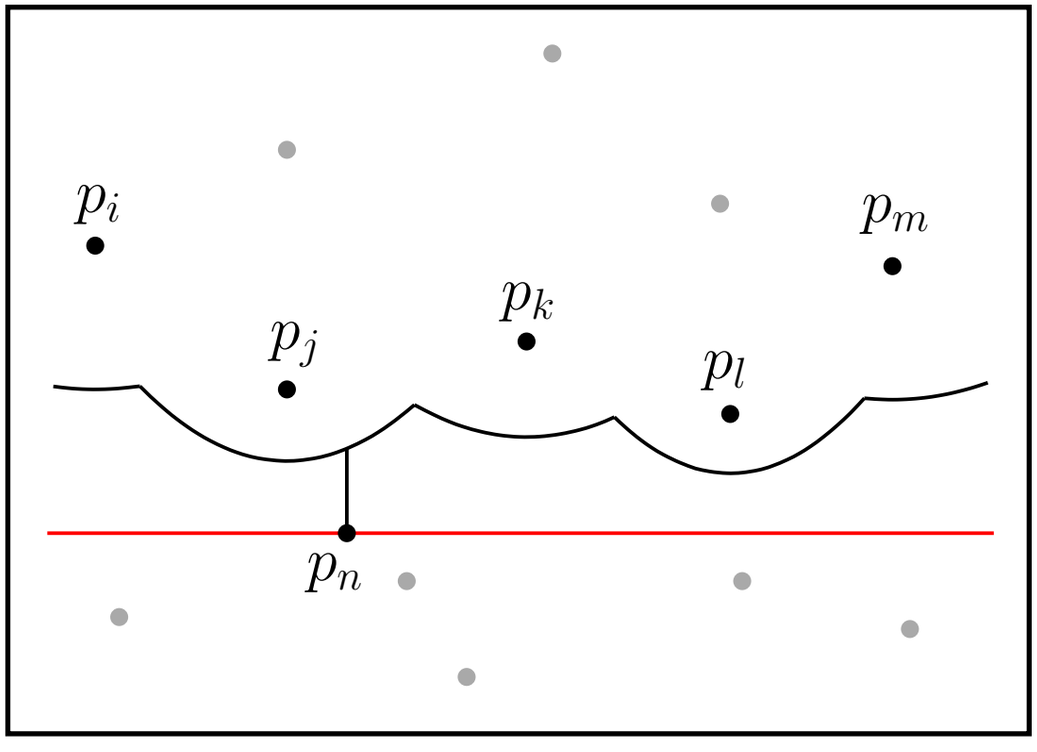
Introducing an arc

Introducing an arc

- Defined by point p_n

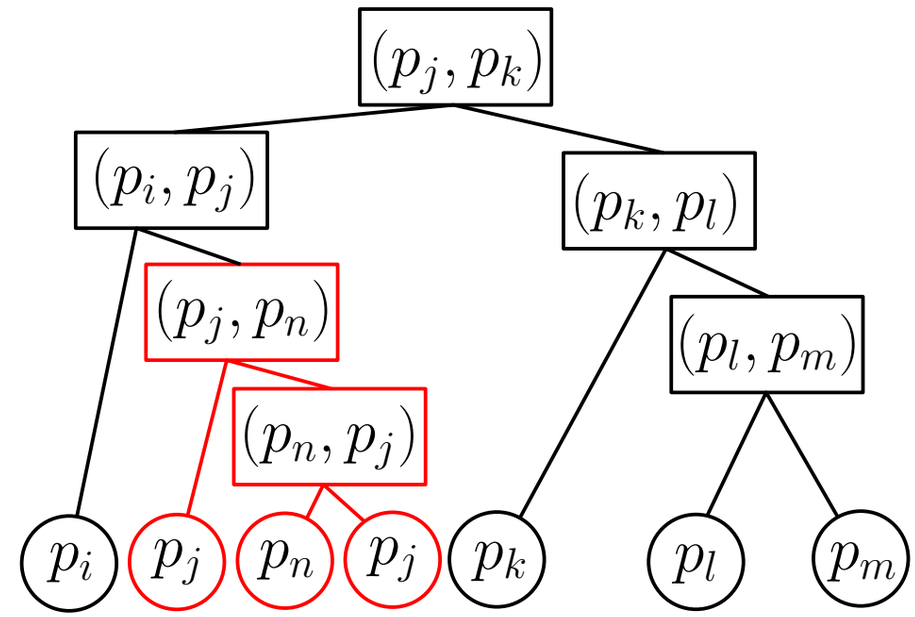
Introducing an arc

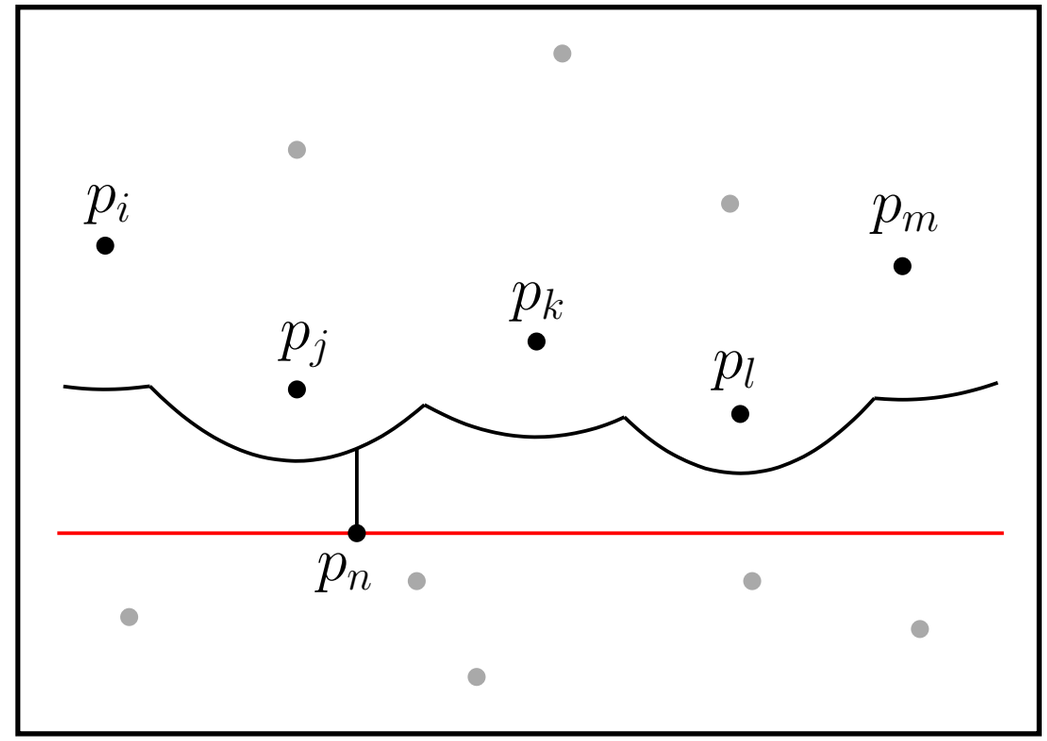
- Defined by point p_n



Introducing an arc

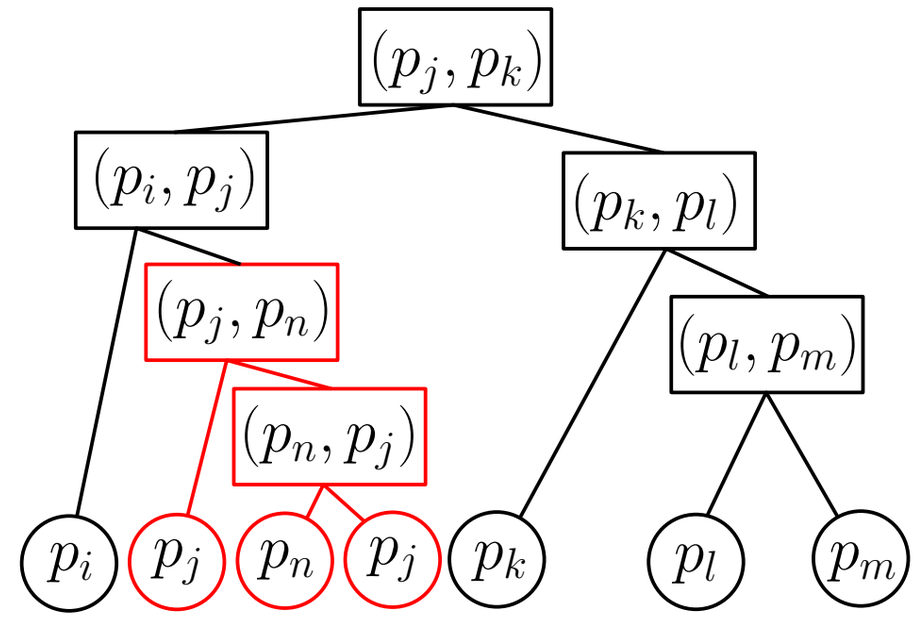
- Defined by point p_n

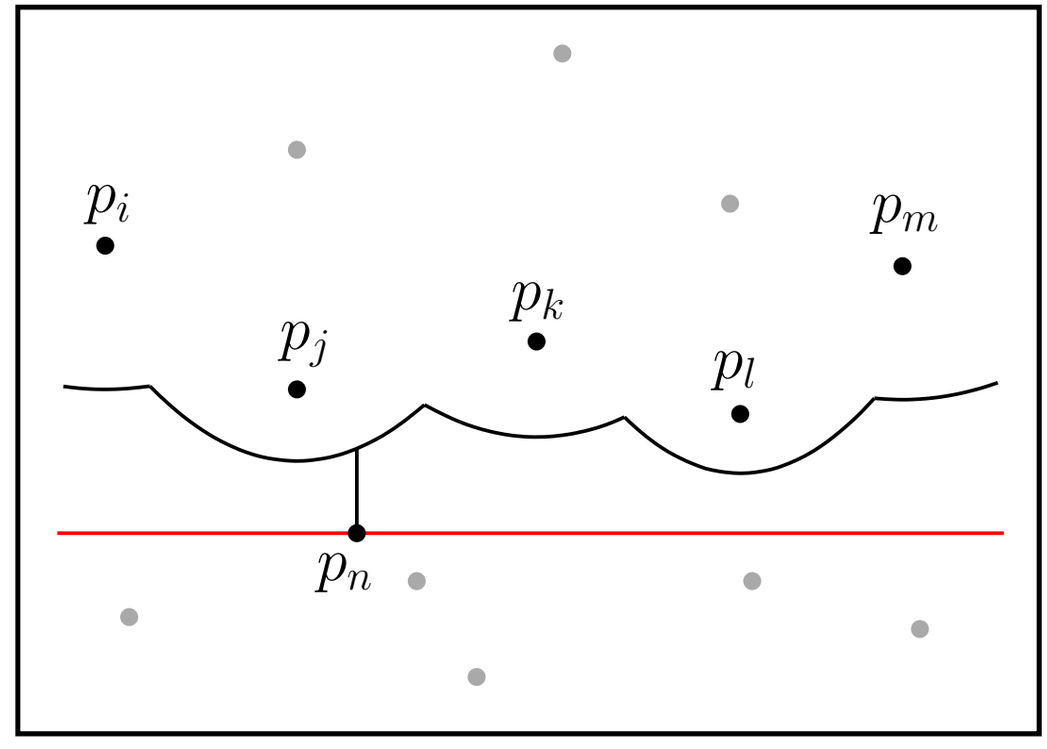




Introducing an arc

- Defined by point p_n
- Search for B at insert position.

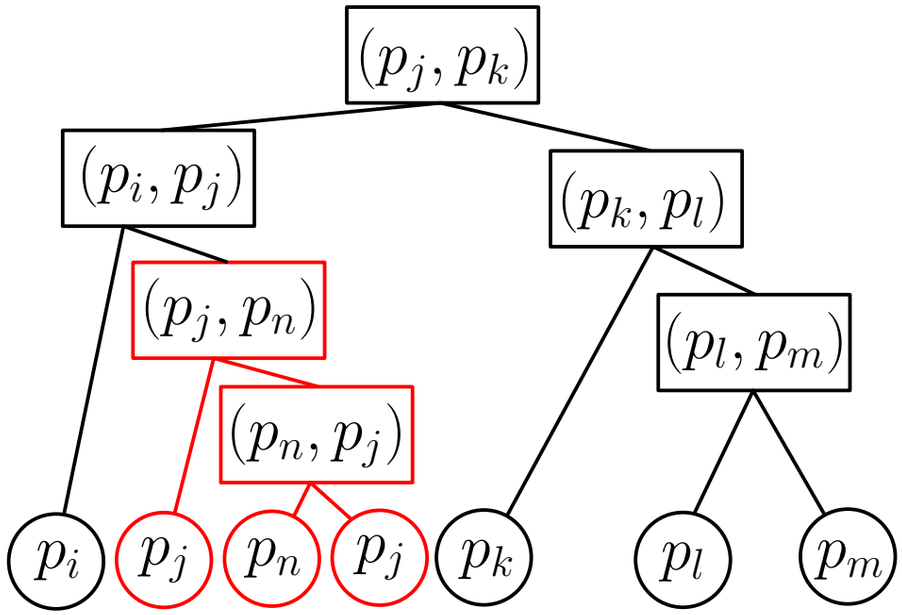


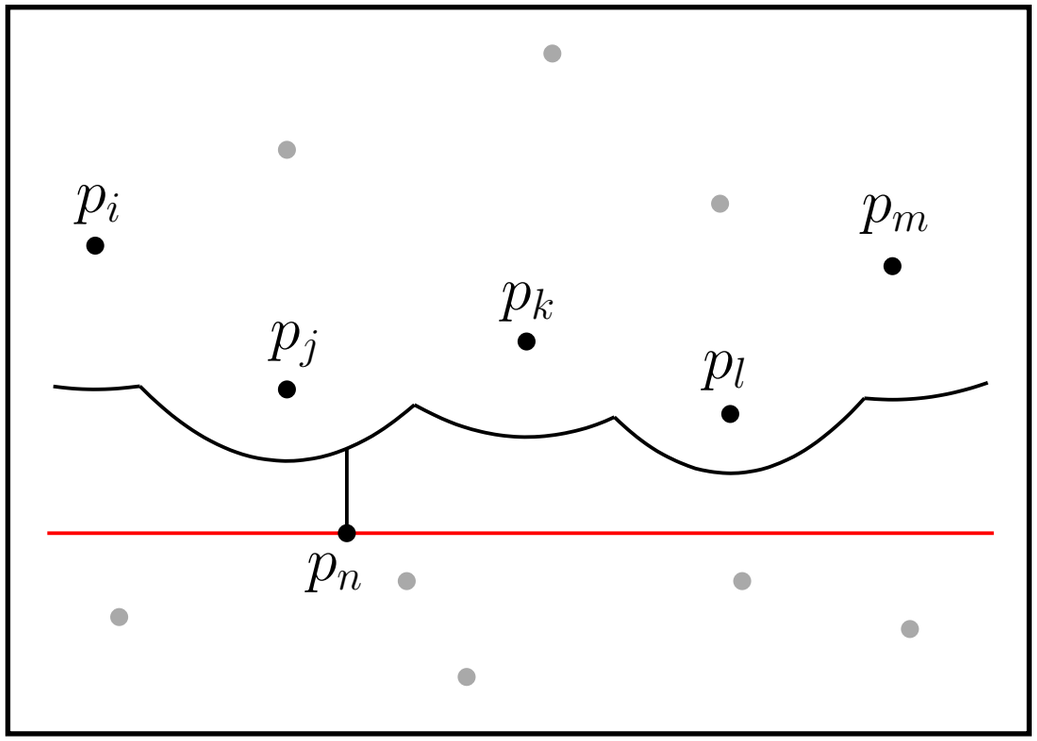


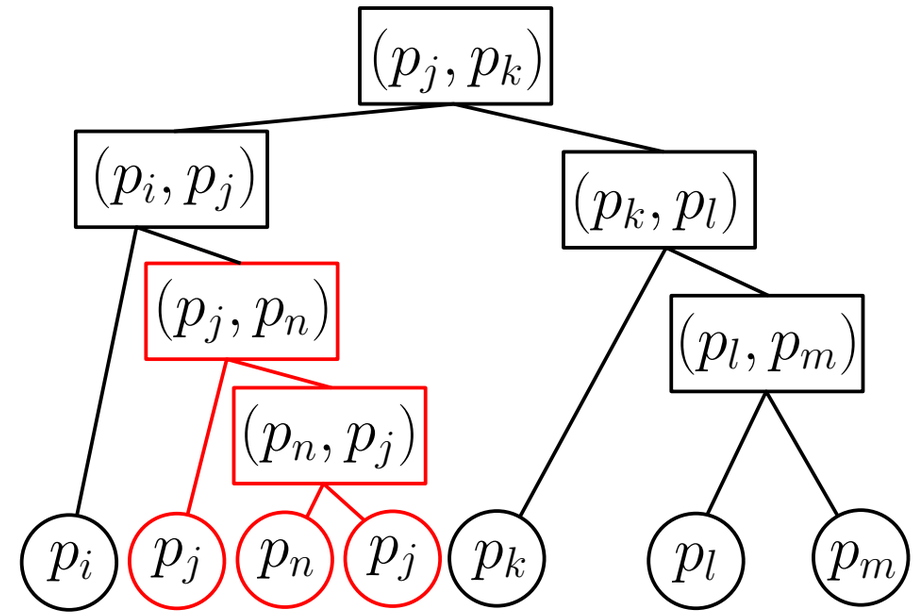
Introducing an arc

- Defined by point p_n
- Search for B at insert position.

  $\rightarrow$ Intersection points stored implicitly



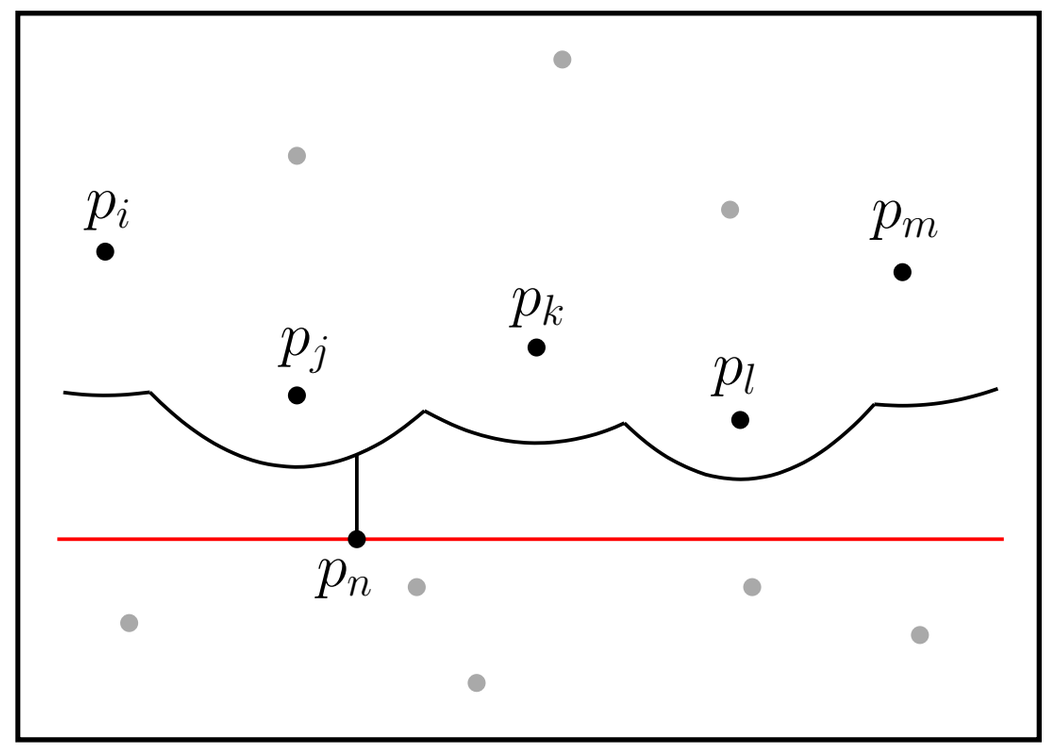


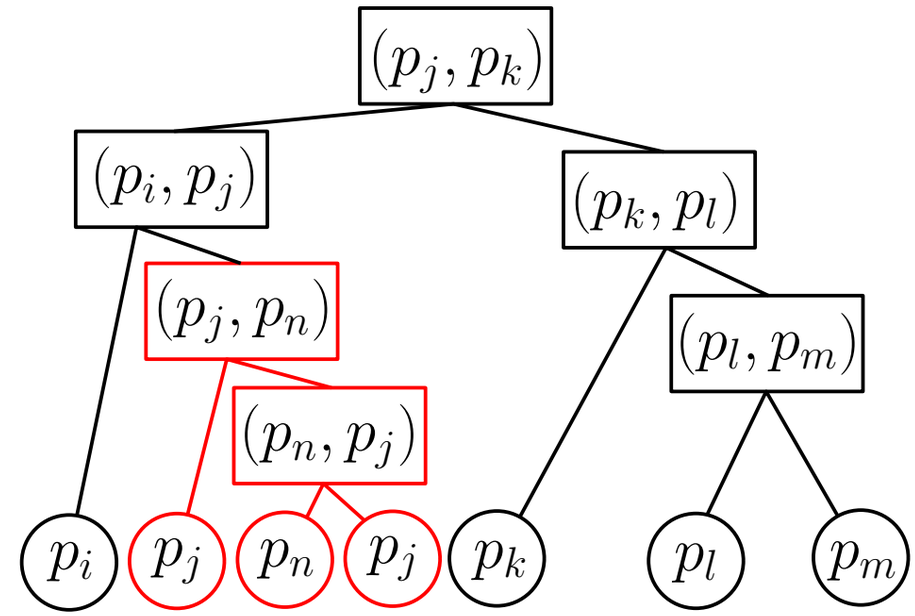


Introducing an arc

- Defined by point p_n
- Search for B at insert position.

  $\rightarrow$ Intersection points stored implicitly
- Splitting an arc β
  (Special case: p_n below arc intersection)

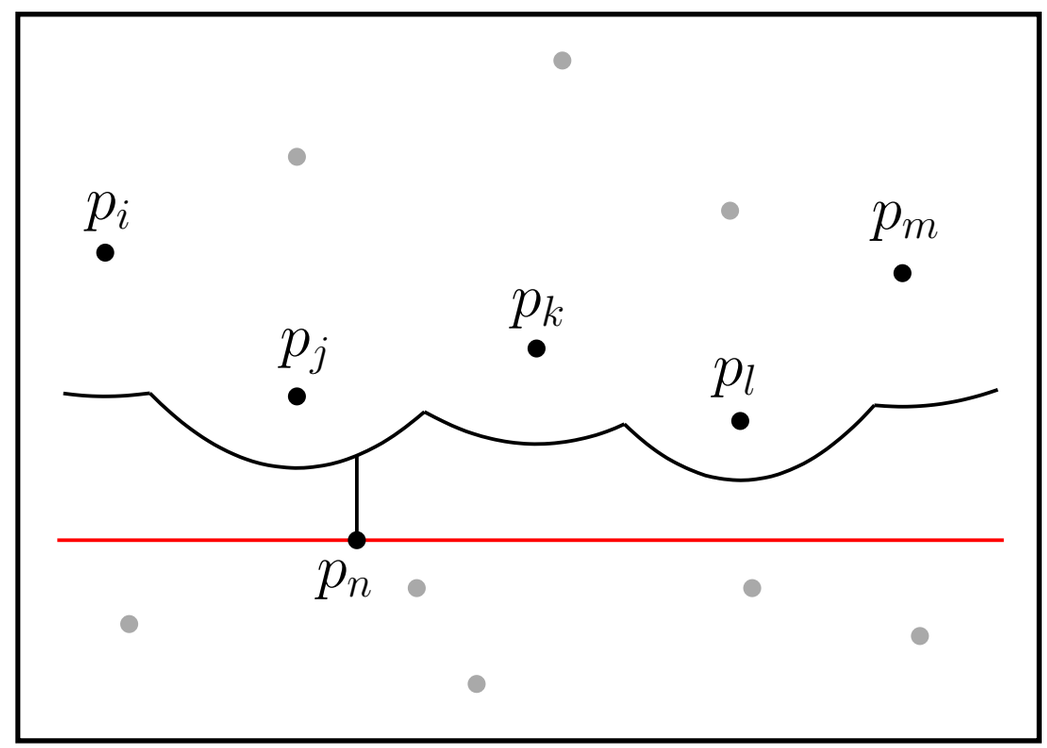




Introducing an arc

- Defined by point p_n
- Search for B at insert position.

  $\rightarrow$ Intersection points stored implicitly
- Splitting an arc β
  (Special case: p_n below arc intersection)
- Splitting: $\beta \rightarrow \beta_1, \beta_n, \beta_2$
  (rebalancing if necessary)



Processing Point Events - III

Generating circle events:

Generating circle events:

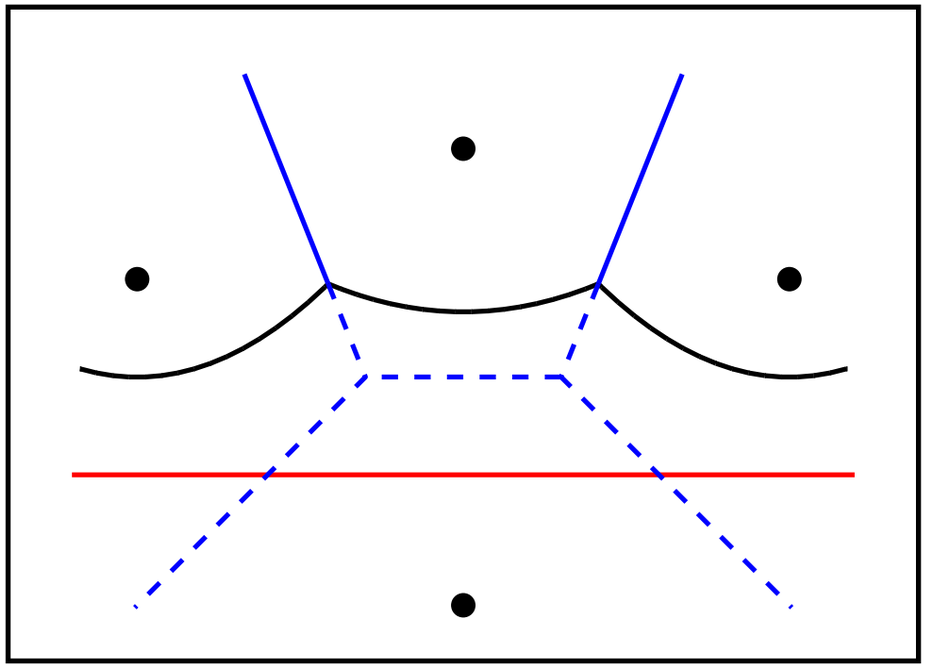
- Before insertion of β_n(defined byp_n):

$$\ldots \beta_i \beta_j \beta_k \beta_l \ldots$$

Generating circle events:

- Before insertion of β_n(defined by p_n):

$$\ldots \beta_i \beta_j \beta_k \beta_l \ldots$$



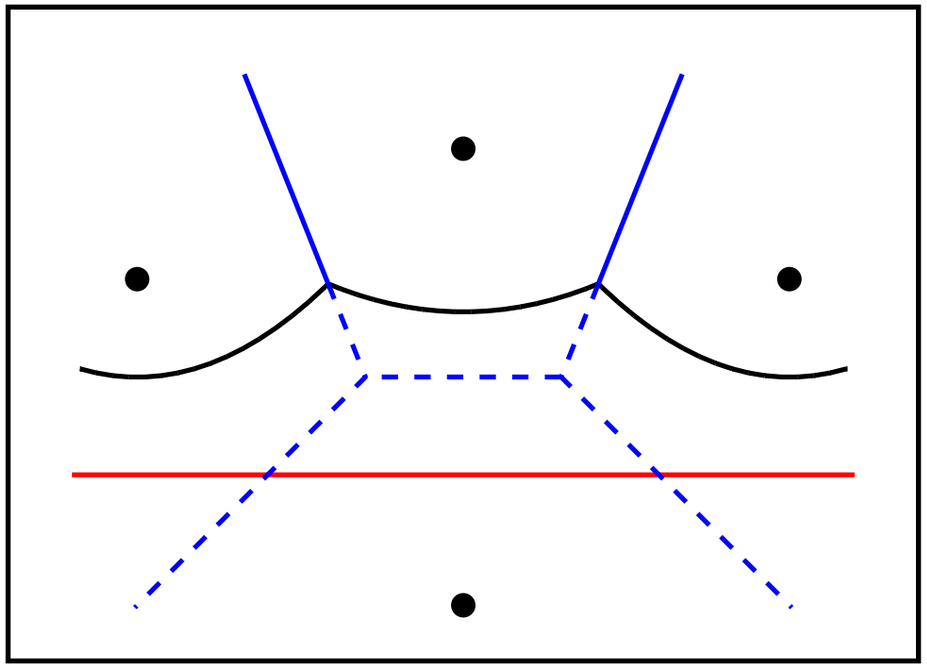
Generating circle events:

- Before insertion of β_n(defined byp_n):

$$\ldots \beta_i \beta_j \beta_k \beta_l \ldots$$

- After insertion:

$$\ldots \beta_i \beta_{j,1} \beta_n \beta_{j,2} \beta_k \beta_l \ldots$$



Generating circle events:

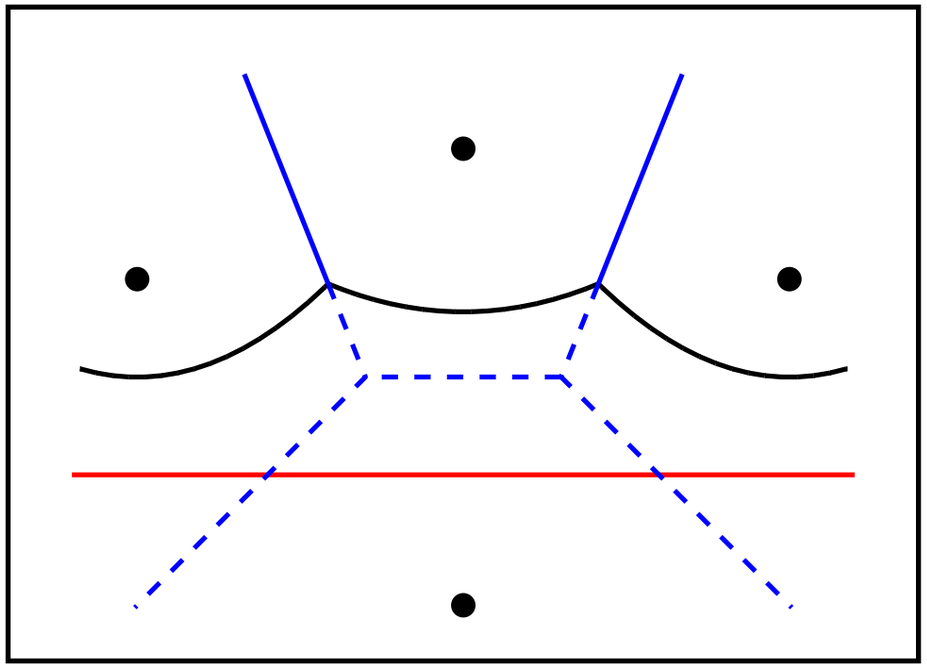
- Before insertion of β_n(defined by p_n):

$$\ldots \beta_i \beta_j \beta_k \beta_l \ldots$$

- After insertion:

$$\ldots \beta_i \beta_{j,1} \beta_n \beta_{j,2} \beta_k \beta_l \ldots$$

- Possibly deletion of circle events (e.g., defined by $(\beta_i, \beta_j, \beta_k)$).



Generating circle events:

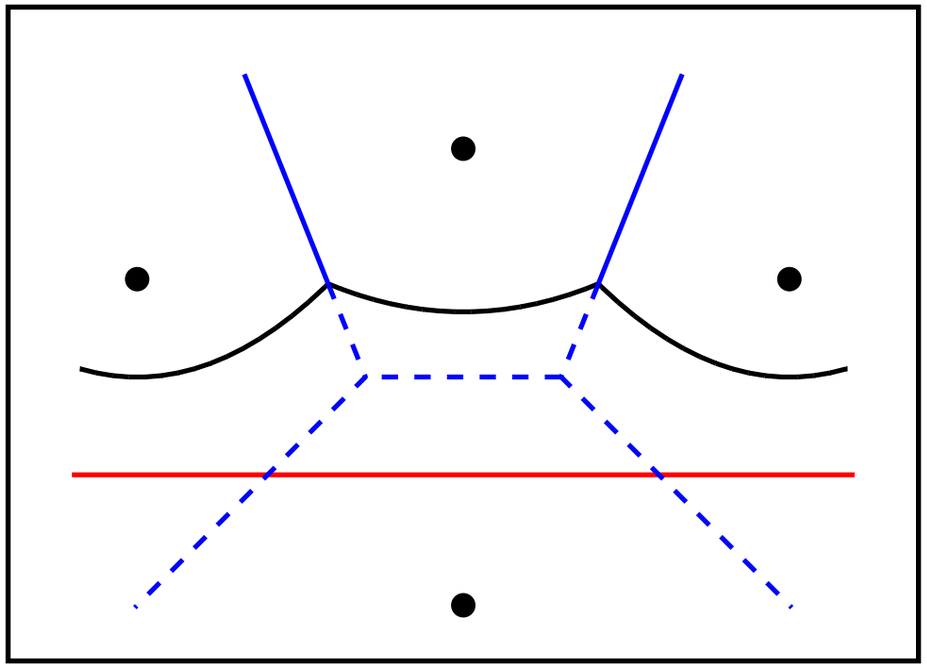
- Before insertion of β_n(defined by p_n):

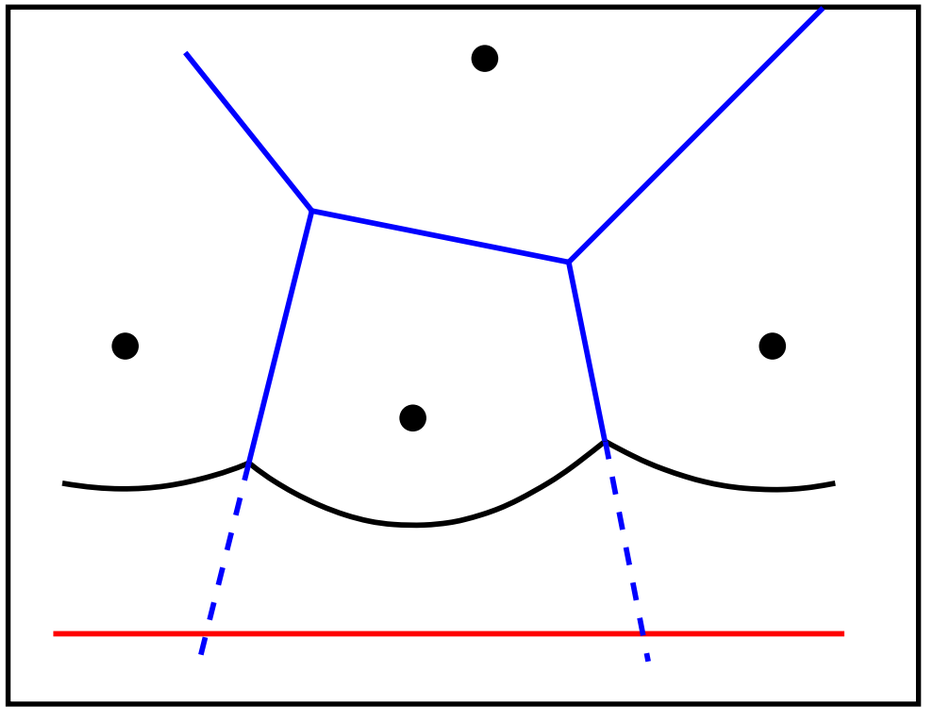
$$\ldots \beta_i \beta_j \beta_k \beta_l \ldots$$

- After insertion:

$$\ldots \beta_i \beta_{j,1} \beta_n \beta_{j,2} \beta_k \beta_l \ldots$$

- Possibly deletion of circle events (e.g., defined by $(\beta_i, \beta_j, \beta_k)$).





Generating circle events:

- Before insertion of β_n(defined by p_n):

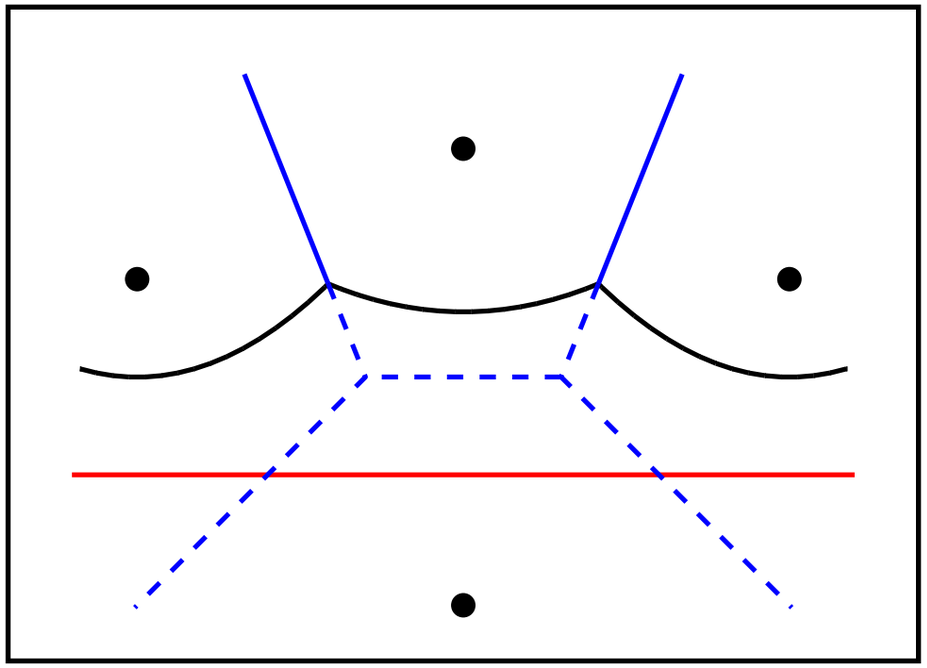
$$\ldots \beta_i \beta_j \beta_k \beta_l \ldots$$

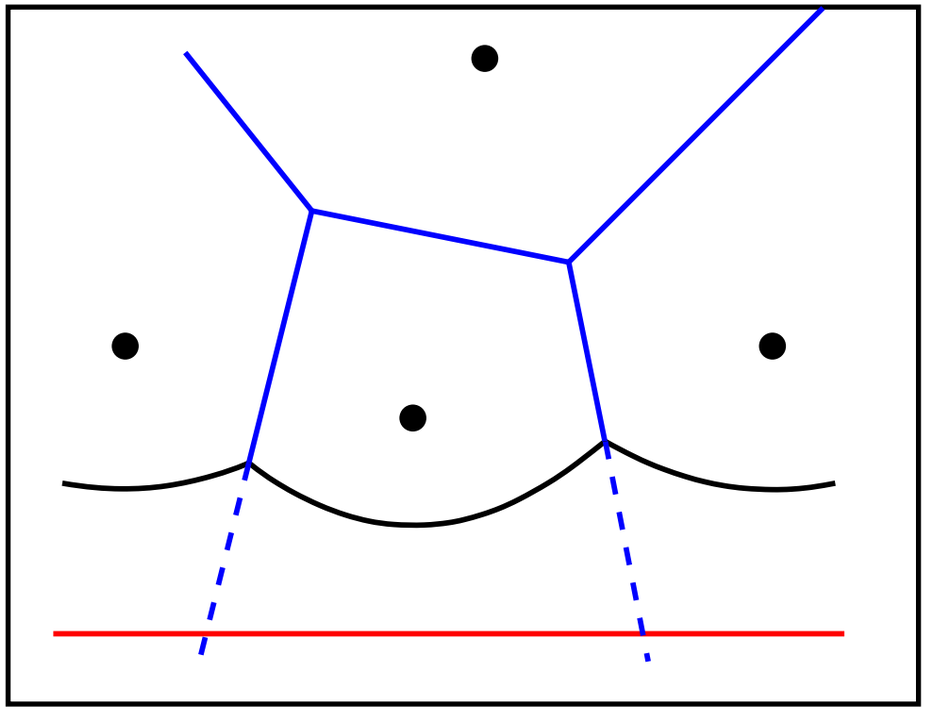
- After insertion:

$$\ldots \beta_i \beta_{j,1} \beta_n \beta_{j,2} \beta_k \beta_l \ldots$$

- Possibly deletion of circle events (e.g., defined by $(\beta_i, \beta_j, \beta_k)$).

- Test all newly adjacent triples.





Generating circle events:

- Before insertion of β_n(defined by p_n):

$$\ldots \beta_i \beta_j \beta_k \beta_l \ldots$$

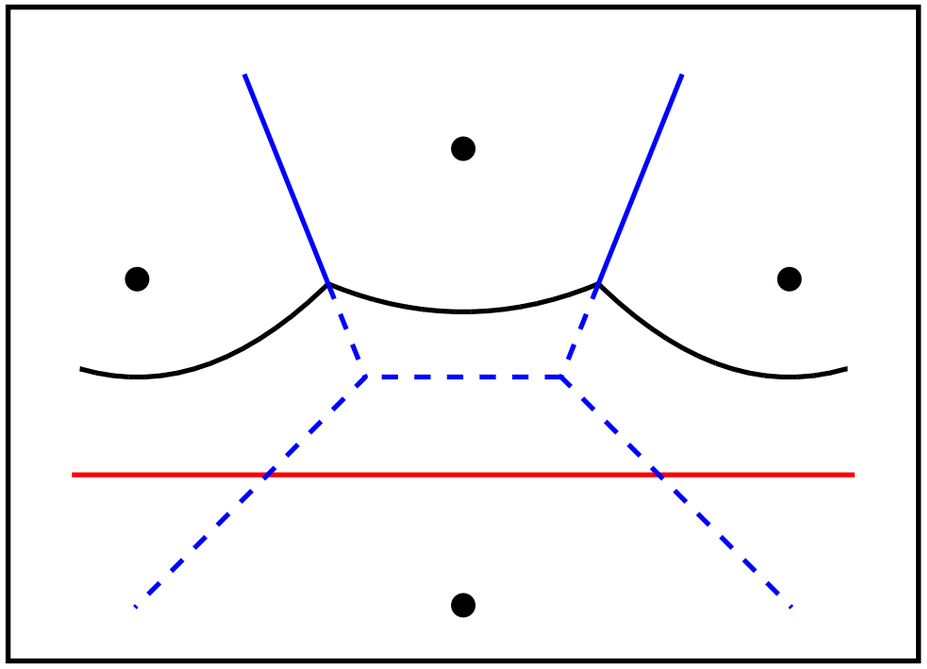
- After insertion:

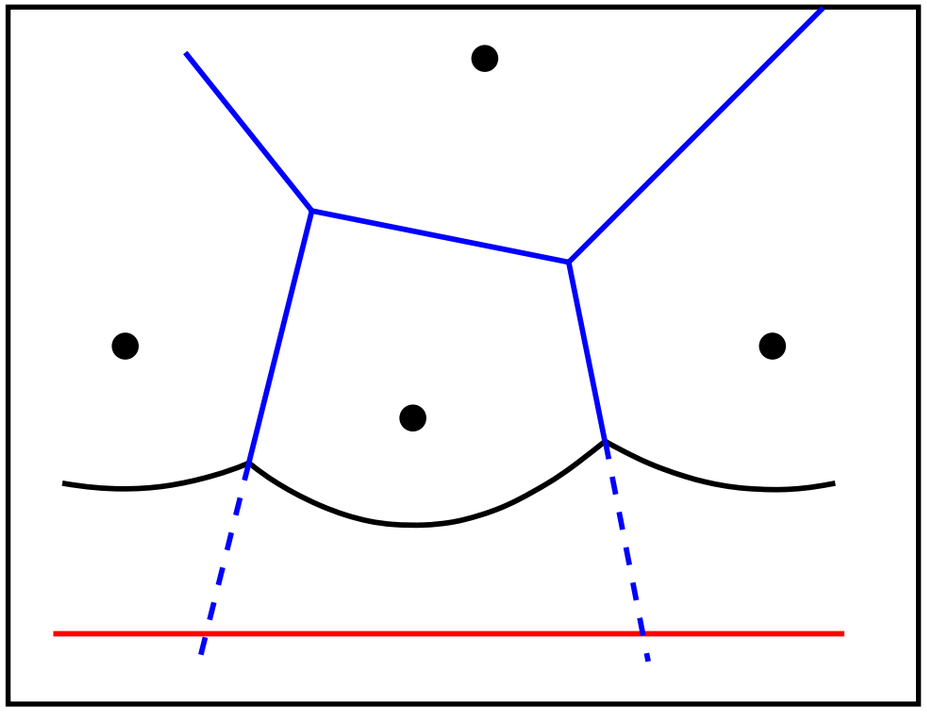
$$\ldots \beta_i \beta_{j,1} \beta_n \beta_{j,2} \beta_k \beta_l \ldots$$

- Possibly deletion of circle events (e.g., defined by $(\beta_i, \beta_j, \beta_k)$).

- Test all newly adjacent triples.

- Insert new events into Q.

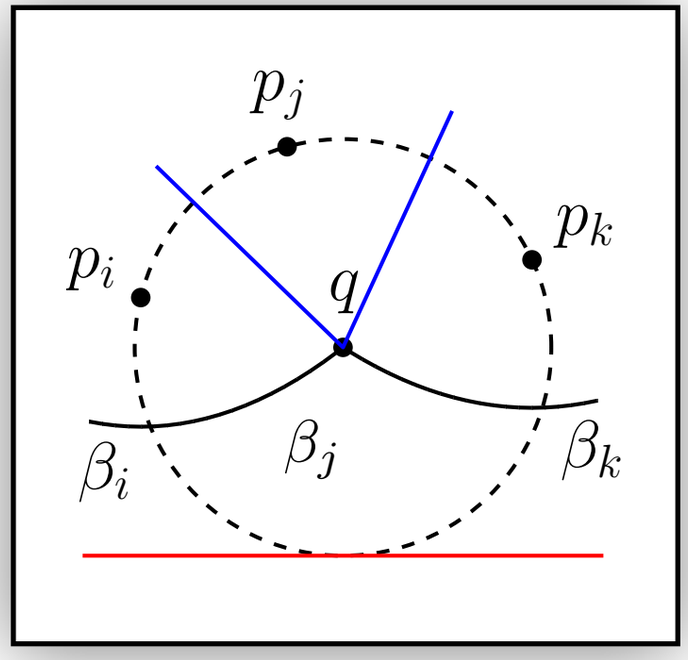




Processing Circle Events - I

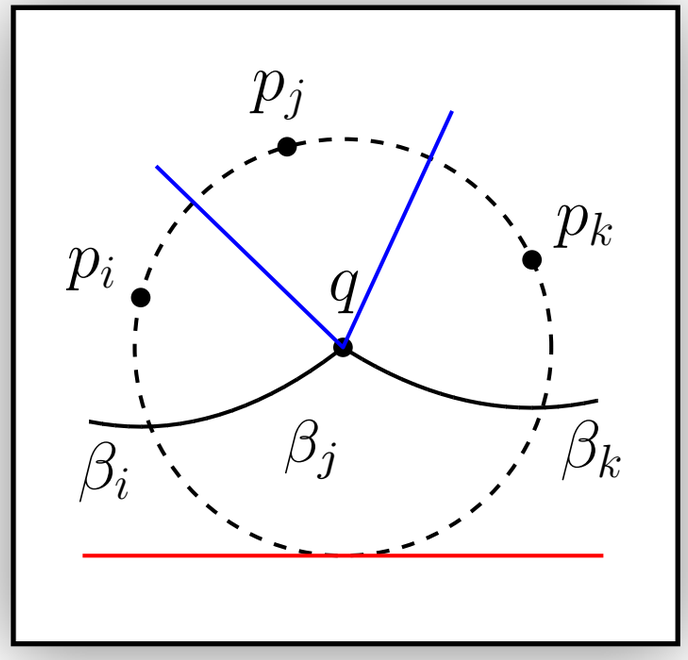
Updates

Updates



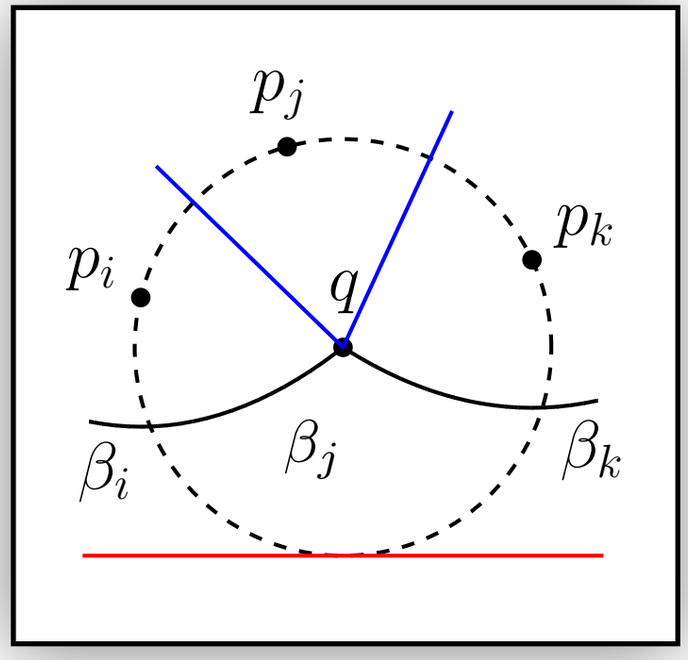
Updates

- Delete β_j from B.



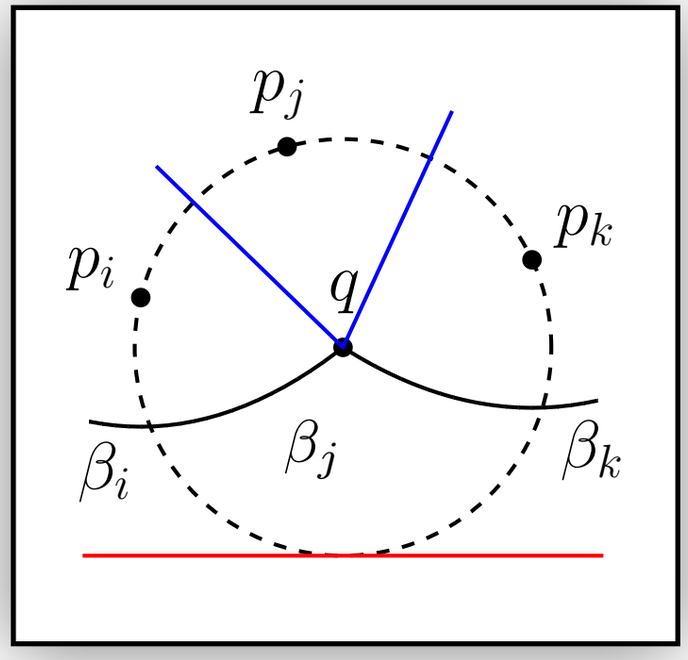
Updates

- Delete β_j from B.
  $\rightarrow$ Rebalance if necessary



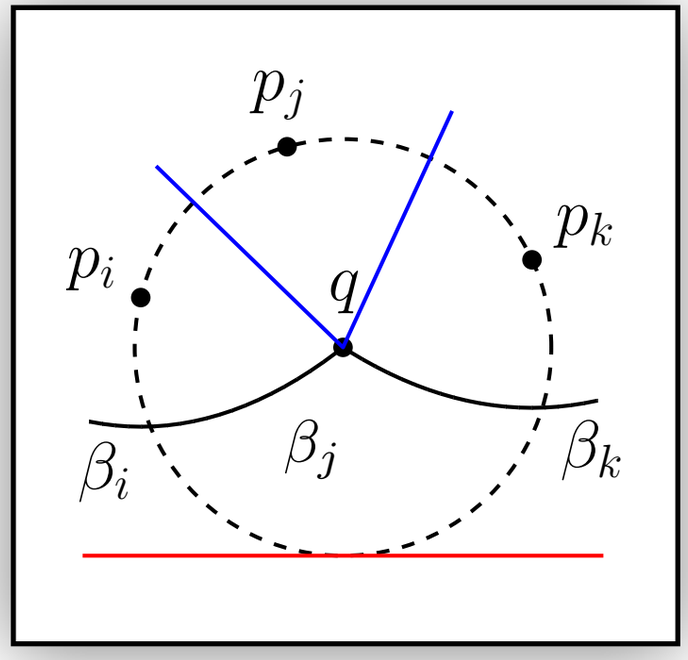
Updates

- Delete β_j from B.
  $\rightarrow$ Rebalance if necessary
- Delete circle events defined by β_j



Updates

- Delete β_j from B.
  $\rightarrow$ Rebalance if necessary
- Delete circle events defined by β_j
- New adjacent triples
  $\rightarrow$ Possibly insert new circle events



Correctness - I

Lemma 4.21

Voronoi vertices $q \in Vor(\mathcal{P})$ correspond to circle events.

Lemma 4.21

Voronoi vertices $q \in Vor(\mathcal{P})$ correspond to circle events.

Proof:

Lemma 4.21

Voronoi vertices $q \in Vor(\mathcal{P})$ correspond to circle events.

Proof:

- Theorem 4.20 $\Rightarrow \exists p_i, p_j, p_k \in \mathcal{P}$:

$$p_i, p_j, p_k \in \partial C$$

$$C \cap \mathcal{P} = \{p_i, p_j, p_k\}$$

for circle C with center q and radius $d(p_i, q) = d(p_j, q) = d(p_k, q)$.

Lemma 4.21

Voronoi vertices $q \in Vor(\mathcal{P})$ correspond to circle events.

Proof:

- Theorem 4.20 $\Rightarrow \exists p_i, p_j, p_k \in \mathcal{P}$:

$$p_i, p_j, p_k \in \partial C$$

$$C \cap \mathcal{P} = \{p_i, p_j, p_k\}$$

for circle C with center q and radius $d(p_i, q) = d(p_j, q) = d(p_k, q)$.

Theorem 4.20:

1. $x \in \mathbb{R}^2$ Voronoi vertex

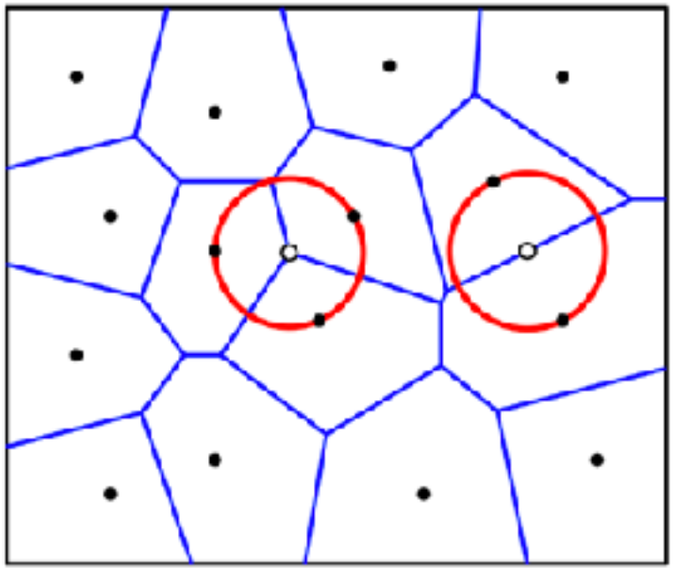
   $\Updownarrow$

   Largest circle C with $\mathcal{P} \cap C^\circ = \emptyset$ and center x has three points on its boundary and

2. $p_i, p_j \in \mathcal{P}$ define Voronoi edge $e \subseteq B(p_i, p_j)$

   $\Updownarrow$

   $\exists$Circle C with
   - only p_i, p_j on boundary and
   - no point in its interior.



Lemma 4.21

Voronoi vertices $q \in Vor(\mathcal{P})$ correspond to circle events.

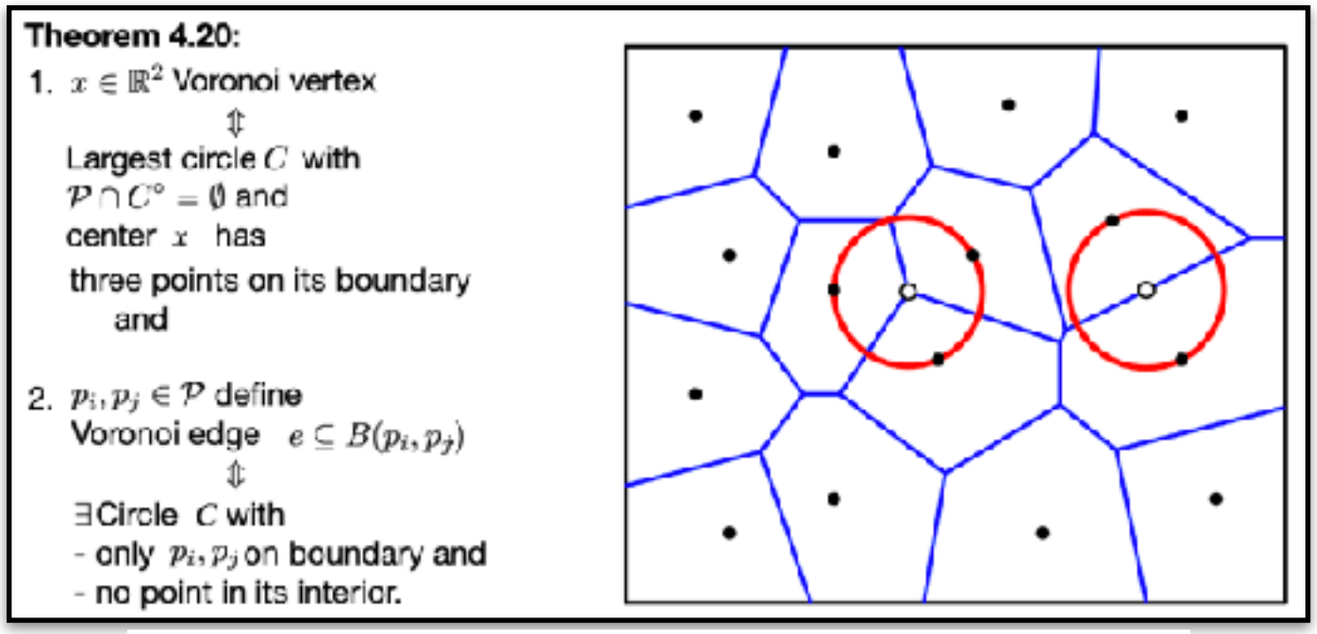
Proof:

- Theorem 4.20 $\Rightarrow \exists p_i, p_j, p_k \in \mathcal{P}$:

$$p_i, p_j, p_k \in \partial C$$

$$C \cap \mathcal{P} = \{p_i, p_j, p_k\}$$

for circle C with center q and radius $d(p_i, q) = d(p_j, q) = d(p_k, q)$.



- $C \cap \mathcal{P} = \{p_i, p_j, p_k\}$
  $\Rightarrow \beta_i, \beta_j, \beta_k$ form adjacent triple
  when ℓ reaches lowest point of C
  $\Rightarrow$ Circle event is processed.

Lemma 4.21

Voronoi vertices $q \in Vor(\mathcal{P})$ correspond to circle events.

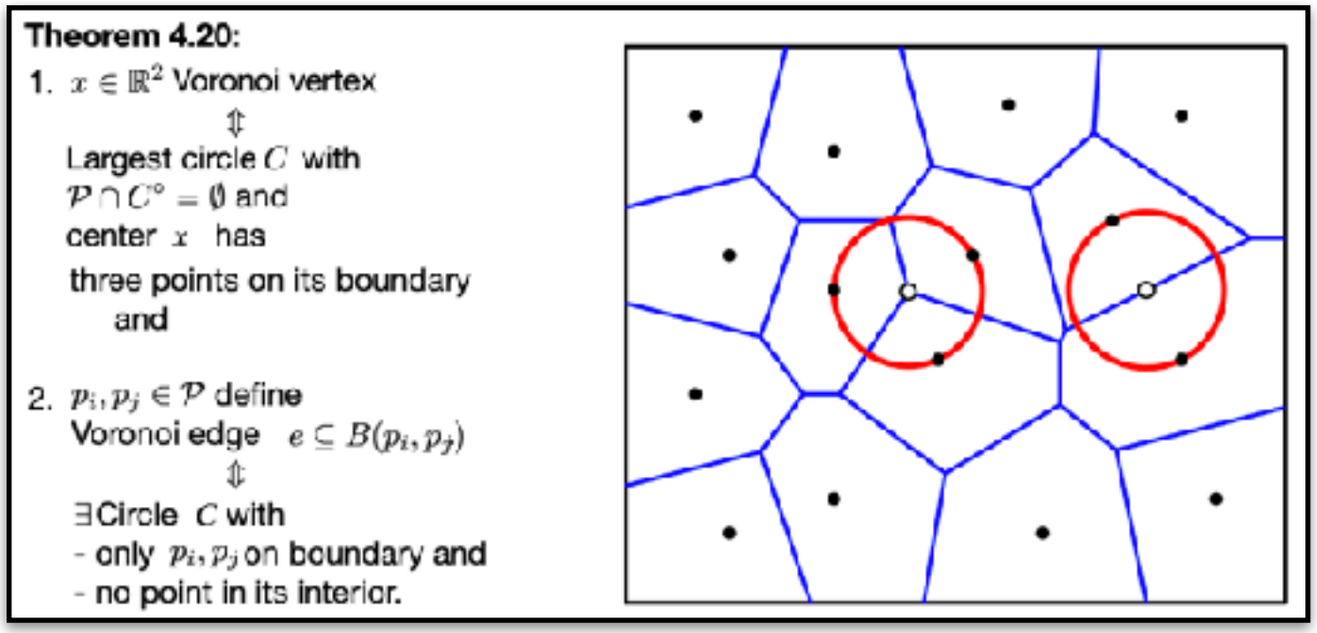
Proof:

- Theorem 4.20 $\Rightarrow \exists p_i, p_j, p_k \in \mathcal{P}$:

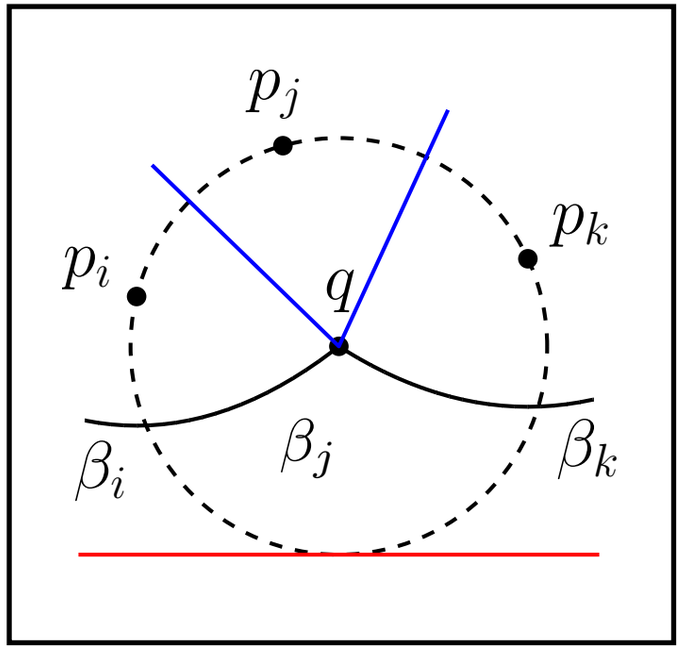
$$p_i, p_j, p_k \in \partial C$$

$$C \cap \mathcal{P} = \{p_i, p_j, p_k\}$$

for circle C with center q and radius $d(p_i, q) = d(p_j, q) = d(p_k, q)$.



- $C \cap \mathcal{P} = \{p_i, p_j, p_k\}$
  $\Rightarrow \beta_i, \beta_j, \beta_k$ form adjacent triple
  when ℓ reaches lowest point of C
  $\Rightarrow$ Circle event is processed.



Lemma 4.21

Voronoi vertices $q \in Vor(\mathcal{P})$ correspond to circle events.

Proof:

- Theorem 4.20 $\Rightarrow \exists p_i, p_j, p_k \in \mathcal{P}$:

$$p_i, p_j, p_k \in \partial C$$

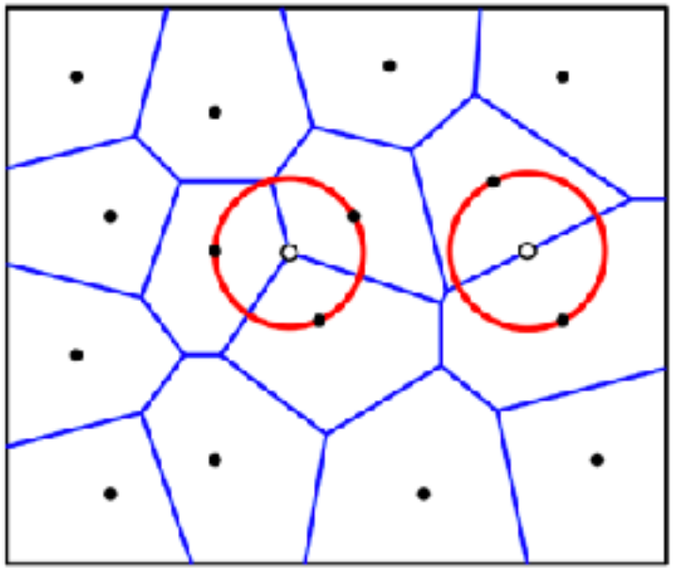
$$C \cap \mathcal{P} = \{p_i, p_j, p_k\}$$

for circle C with center q and radius $d(p_i, q) = d(p_j, q) = d(p_k, q)$.

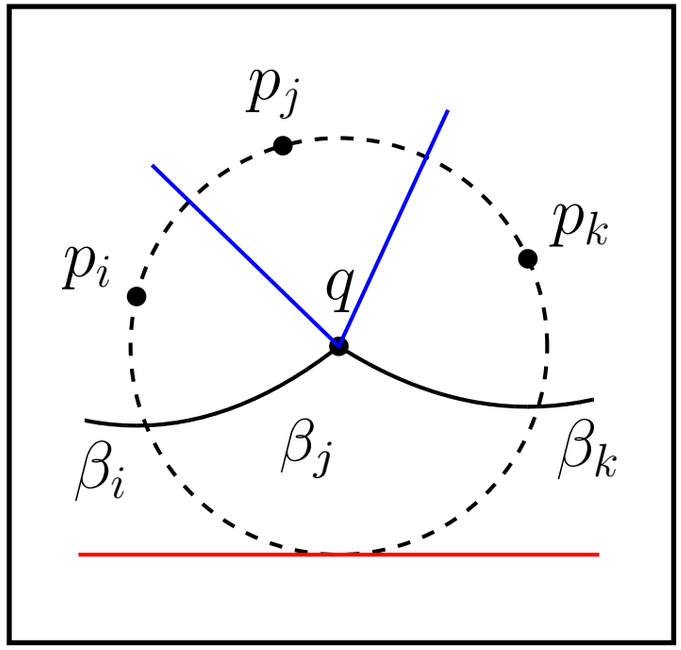
Theorem 4.20:

1. $x \in \mathbb{R}^2$ Voronoi vertex
   $\Updownarrow$
   Largest circle C with $\mathcal{P} \cap C^\circ = \emptyset$ and center x has three points on its boundary and

2. $p_i, p_j \in \mathcal{P}$ define Voronoi edge $e \subseteq B(p_i, p_j)$
   $\Updownarrow$
   $\exists$Circle C with
   - only p_i, p_j on boundary and
   - no point in its interior.



- $C \cap \mathcal{P} = \{p_i, p_j, p_k\}$
  $\Rightarrow \beta_i, \beta_j, \beta_k$ form adjacent triple
  when ℓ reaches lowest point of C
  $\Rightarrow$ Circle event is processed.



$\square$

Constructing the Voronoi Diagram - I

Goal

Goal

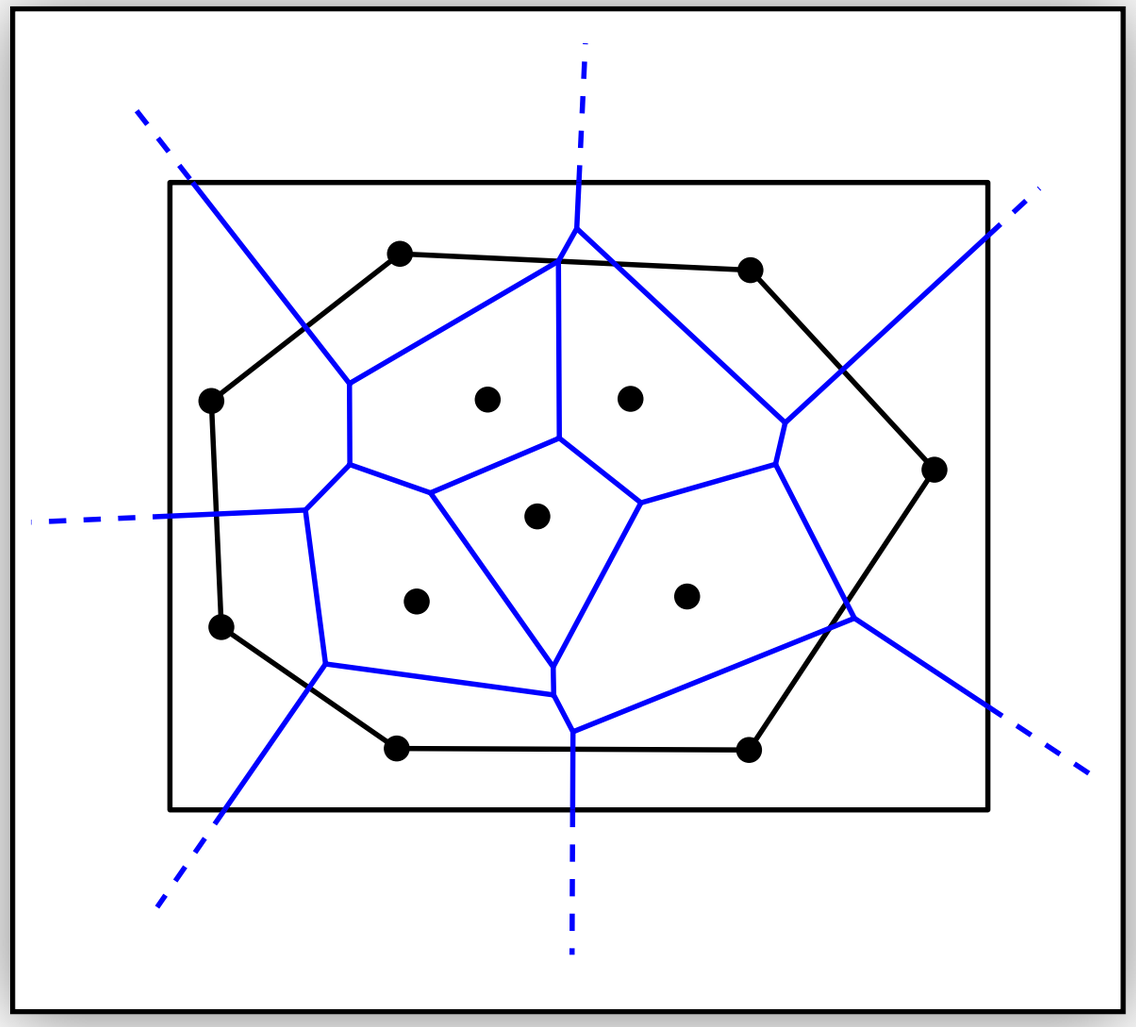
- Voronoi diagram: DCEL.

Goal

- Voronoi diagram: DCEL.
- Necessary: bounding box

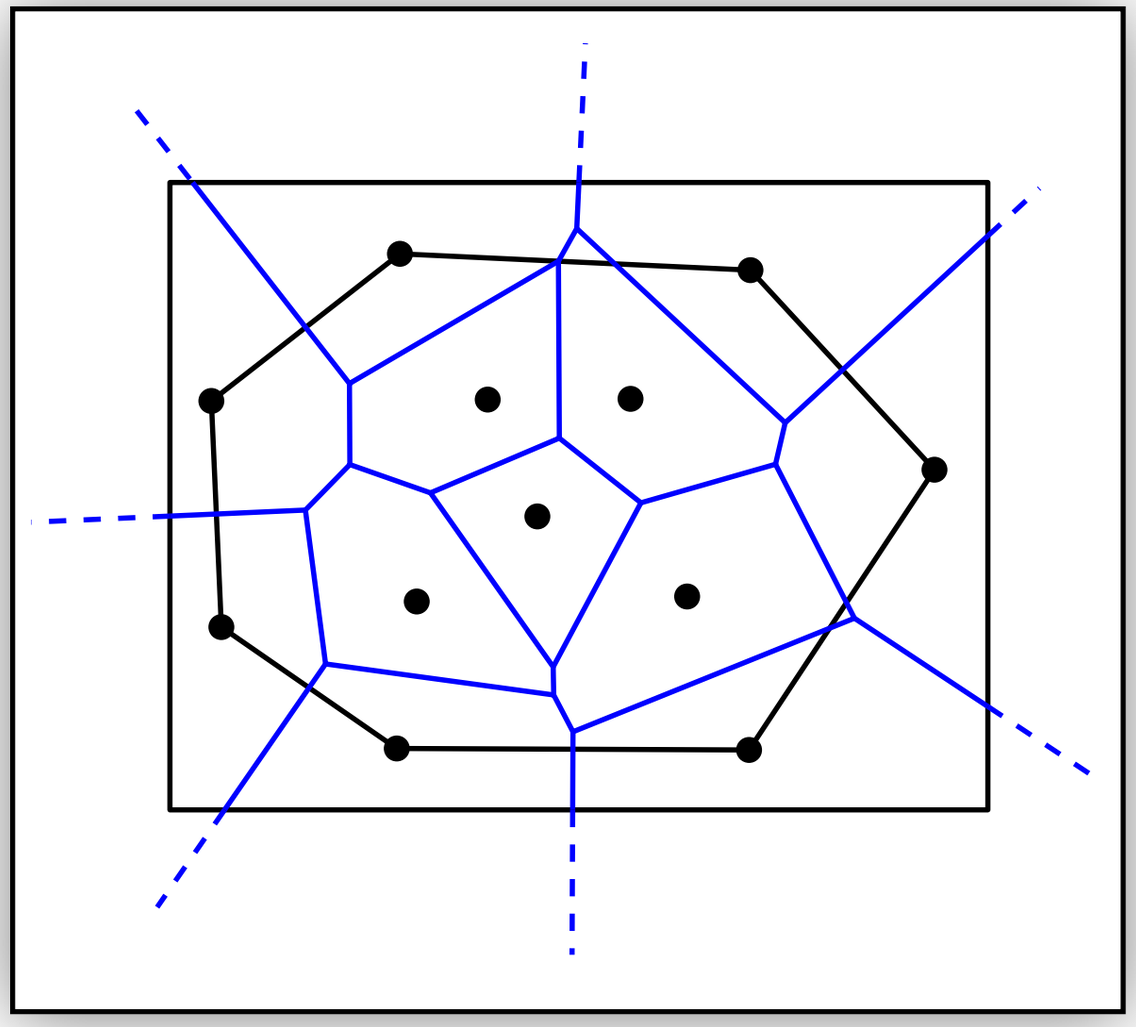
Goal

- Voronoi diagram: DCEL.
- Necessary: bounding box



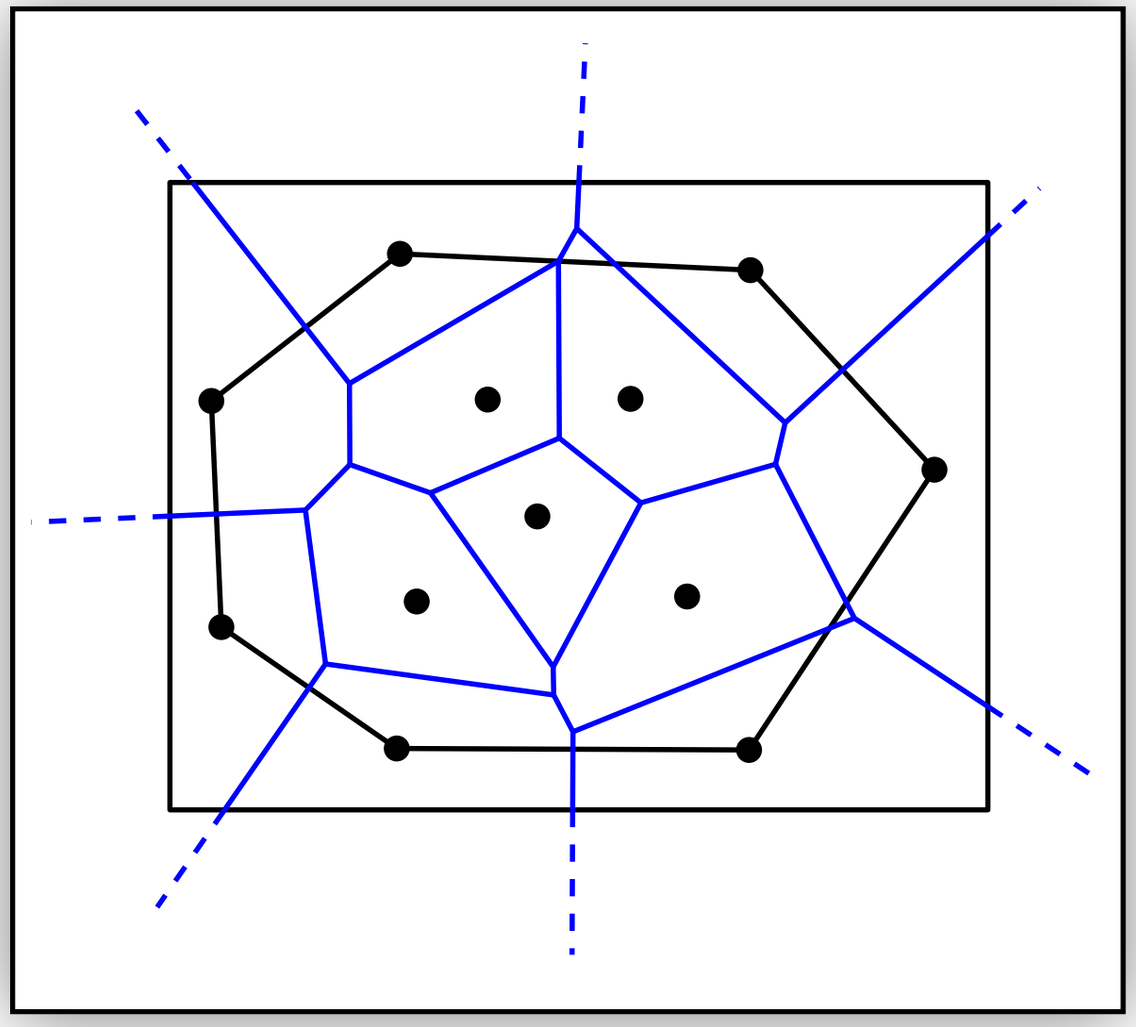
Goal

- Voronoi diagram: DCEL.
- Necessary: bounding box
- Possible: Voronoi vertices outside $conv(\mathcal{P})$.



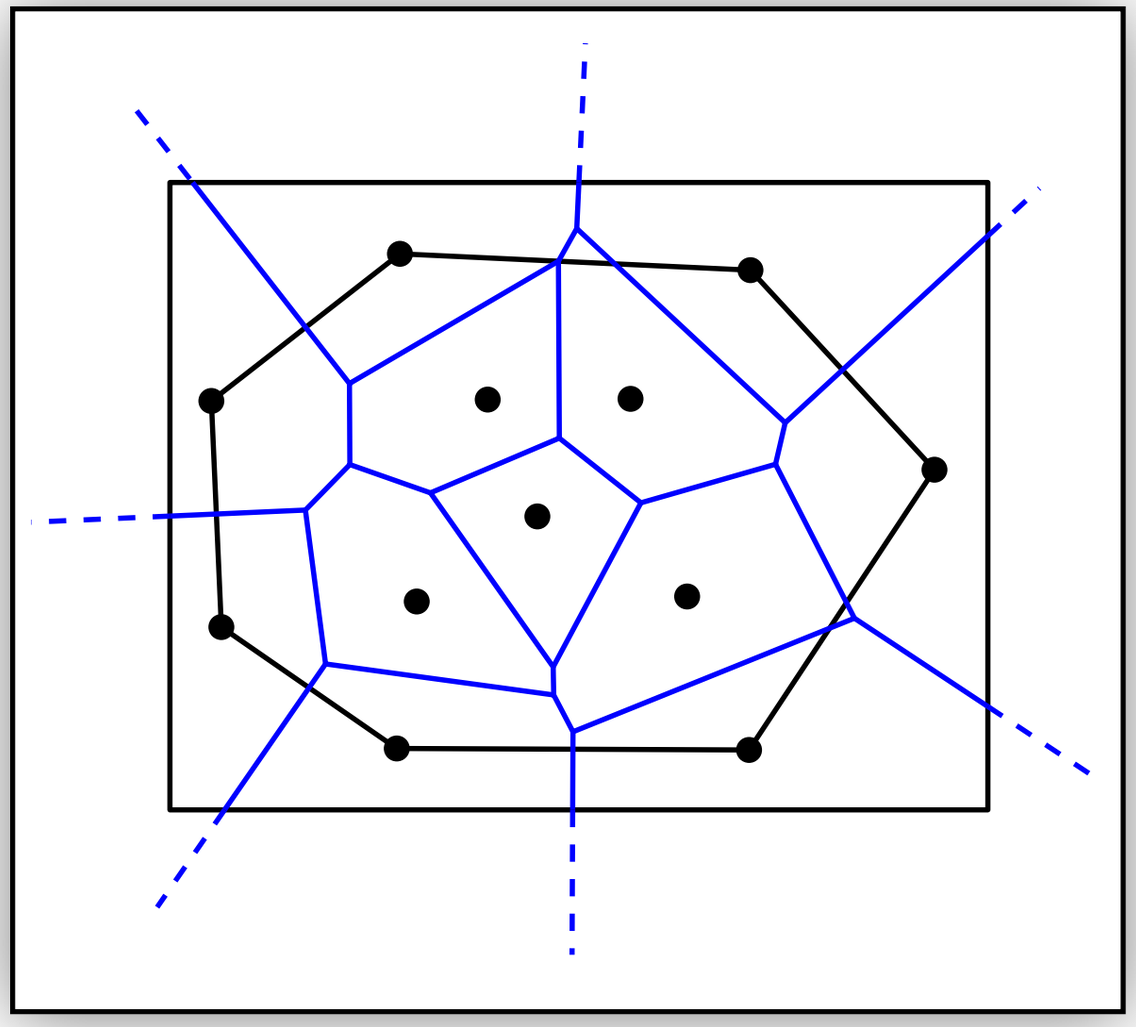
Goal

- Voronoi diagram: DCEL.
- Necessary: bounding box
- Possible: Voronoi vertices outside $conv(\mathcal{P})$.
- Preprocessing: $conv(\mathcal{P})$ + topologically enclosing box $\rightarrow$ outer face.



Goal

- Voronoi diagram: DCEL.
- Necessary: bounding box
- Possible: Voronoi vertices outside $conv(\mathcal{P})$.
- Preprocessing: $conv(\mathcal{P})$ + topologically enclosing box $\rightarrow$ outer face.
- Constructing $Vor(\mathcal{P})$: second run



Constructing the Voronoi Diagram - II

Constructing the DCEL:

Constructing the DCEL:

- Initially: Construct all regions

Constructing the DCEL:

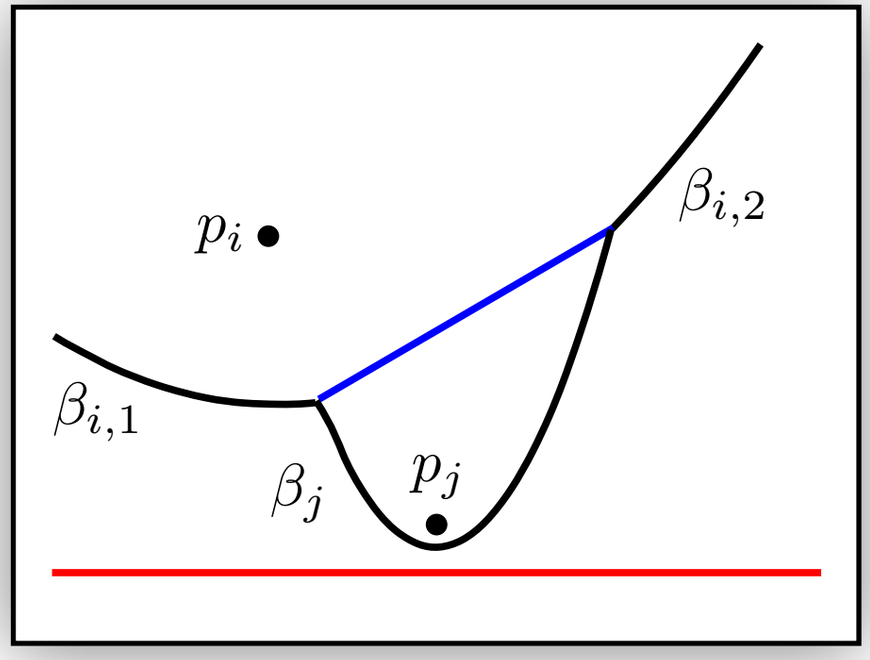
- Initially: Construct all regions

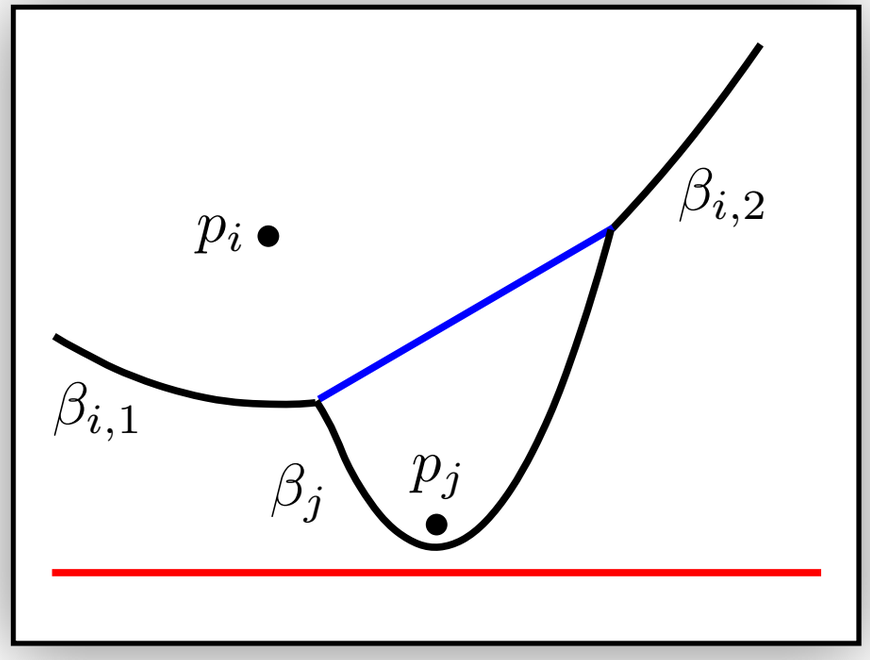
- Point event: Generate $e \subset B(p_i, p_j)$

Constructing the DCEL:

- Initially: Construct all regions

- Point event: Generate $e \subset B(p_i, p_j)$

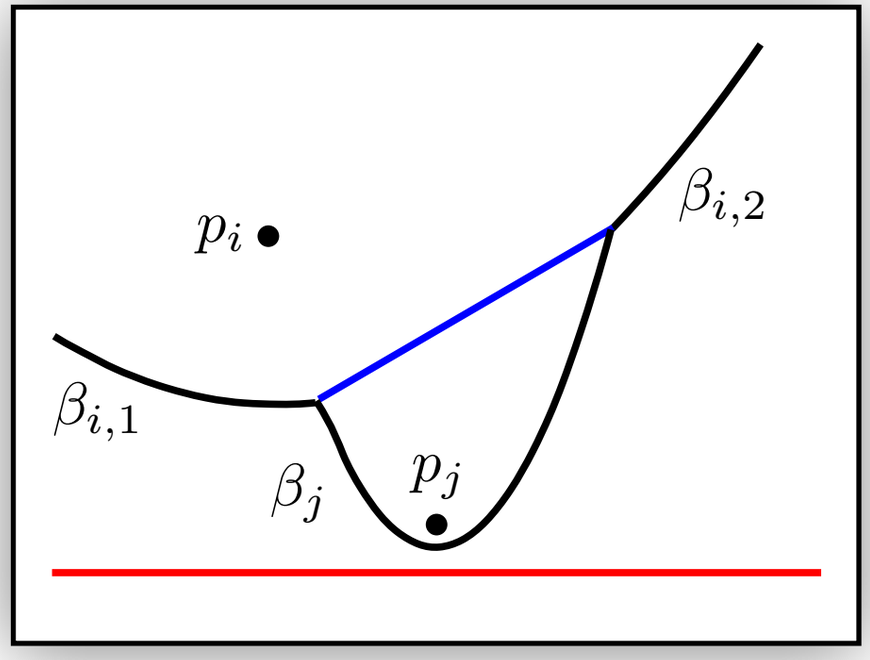




Constructing the DCEL:

- Initially: Construct all regions

- Point event: Generate $e \subset B(p_i, p_j)$ (first without end points)

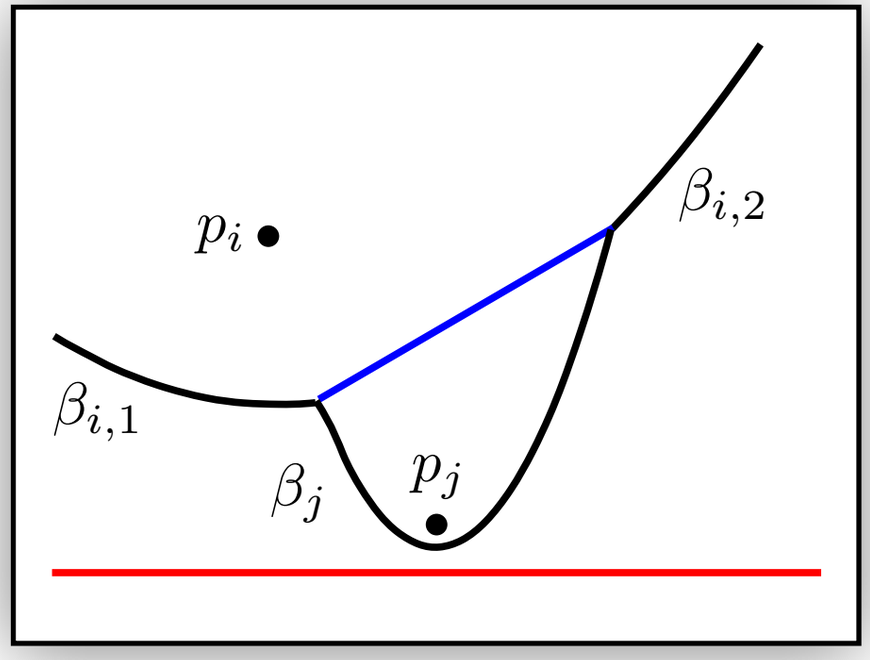


Constructing the DCEL:

- Initially: Construct all regions

- Point event: Generate $e \subset B(p_i, p_j)$ (first without end points)

  $\rightarrow e$ separates regions for p_i, p_j.



Constructing the DCEL:

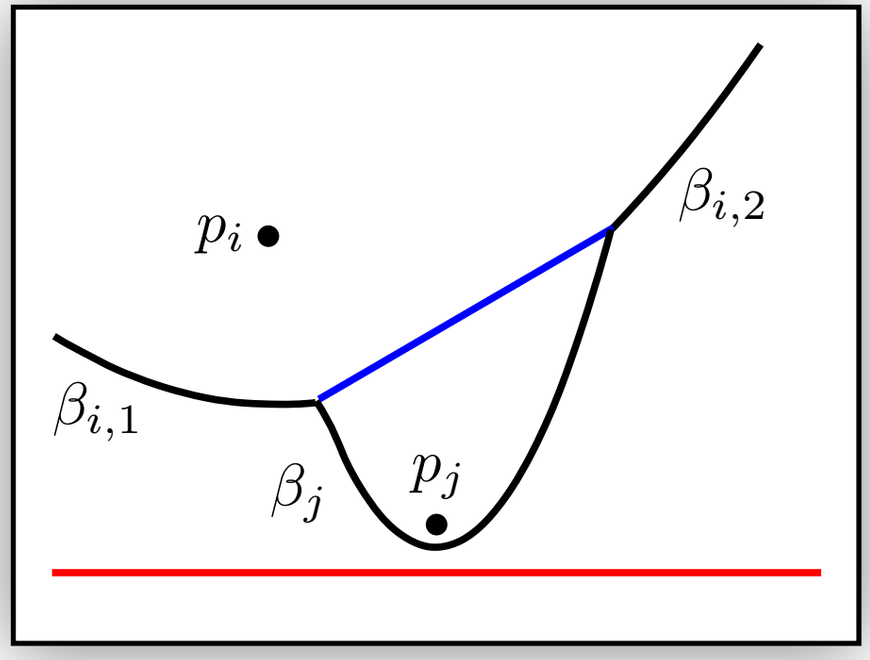
- Initially: Construct all regions

- Point event: Generate $e \subset B(p_i, p_j)$ (first without end points)
  $\rightarrow e$ separates regions for p_i, p_j.

- Circle event: Merge $e_1 \subset B(p_i, p_j), e_2 \subset B(p_j, p_k)$ at q.

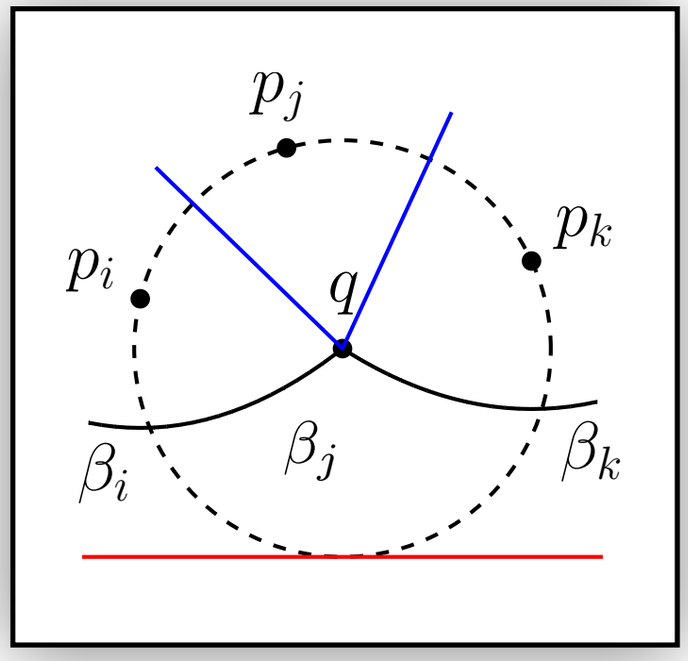
Constructing the DCEL:

- Initially: Construct all regions



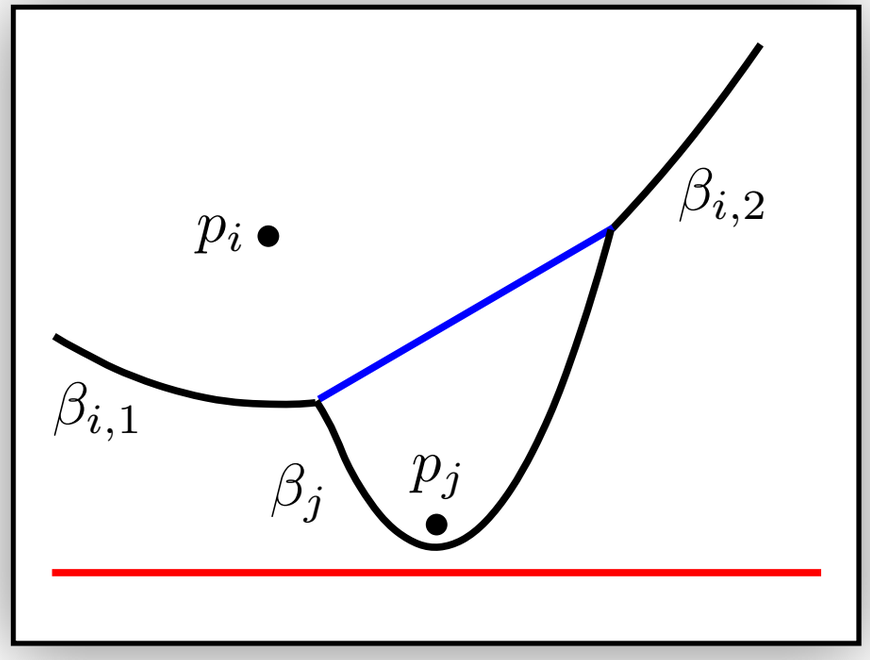
- Point event: Generate $e \subset B(p_i, p_j)$ (first without end points)
  $\rightarrow e$ separates regions for p_i, p_j.

- Circle event: Merge $e_1 \subset B(p_i, p_j), e_2 \subset B(p_j, p_k)$ at q.



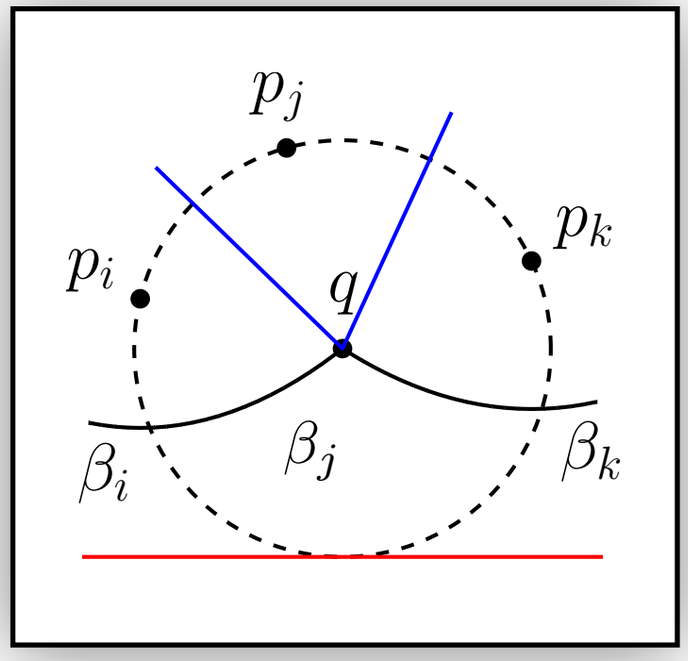
Constructing the DCEL:

- Initially: Construct all regions



- Point event: Generate $e \subset B(p_i, p_j)$ (first without end points)
  $\rightarrow e$ separates regions for p_i, p_j.

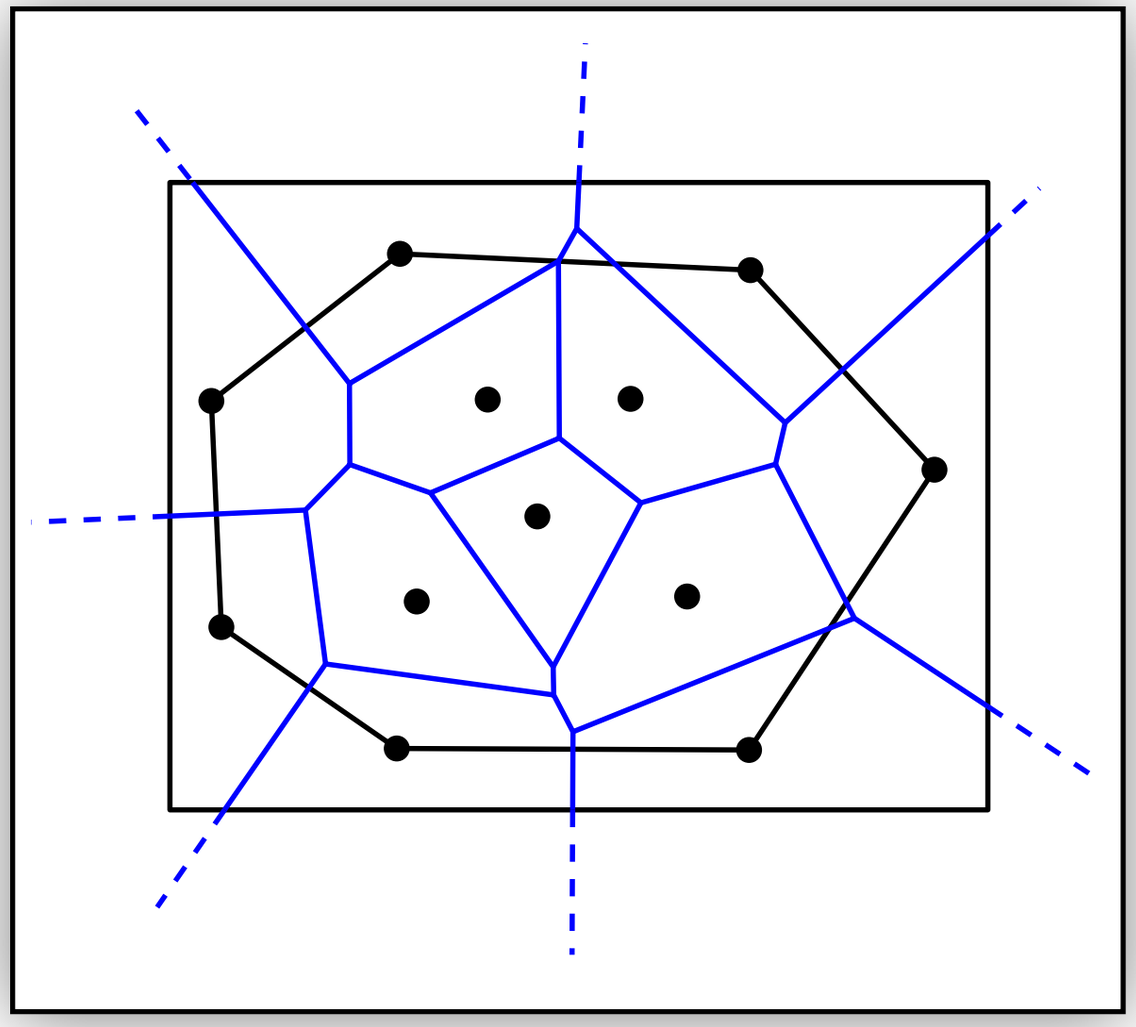
- Circle event: Merge $e_1 \subset B(p_i, p_j), e_2 \subset B(p_j, p_k)$ at q.
  $\rightarrow$ new edge $e_3 \subset B(p_i, p_k)$ incident to q.



Constructing the Voronoi Diagram - III

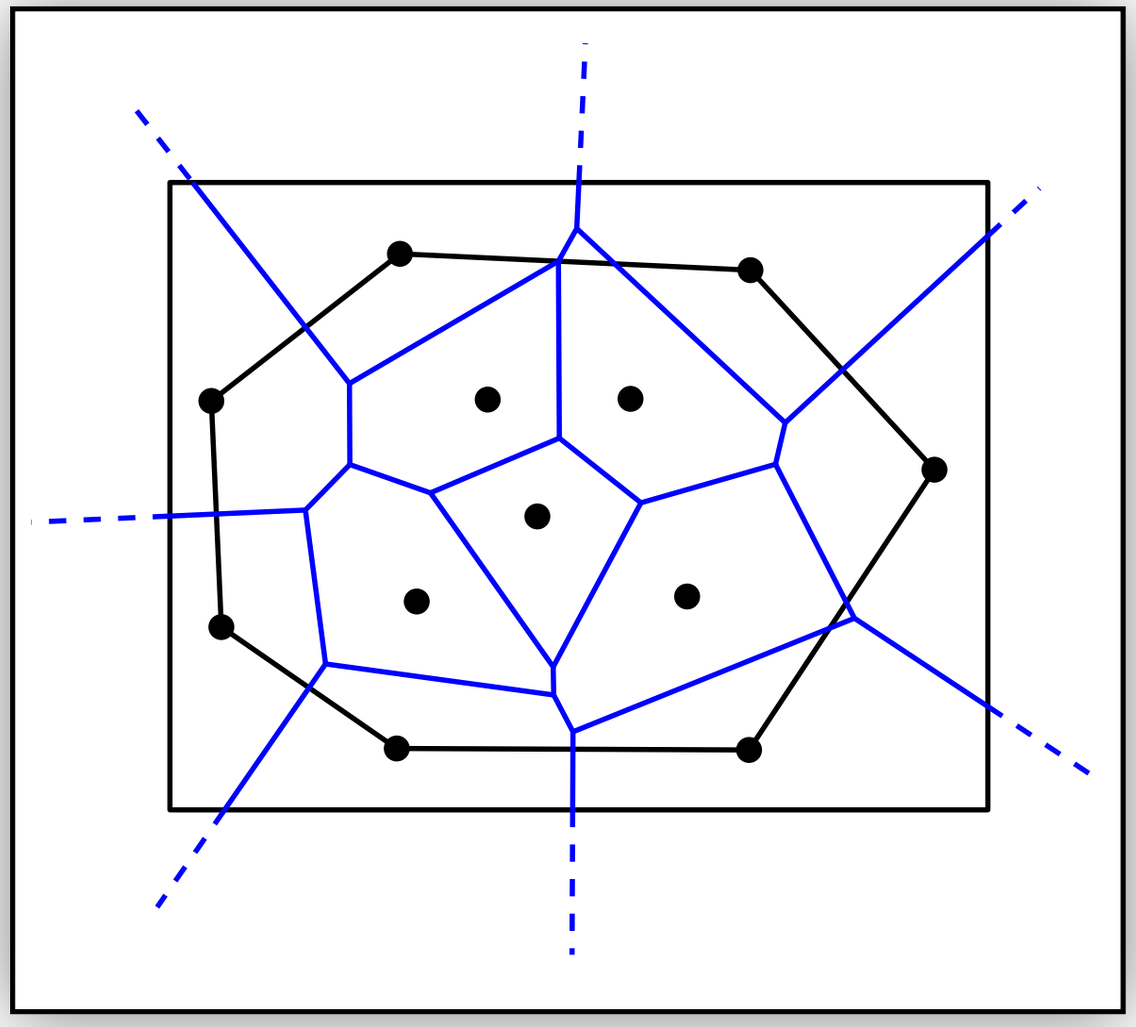
Unbounded edges:

Unbounded edges:



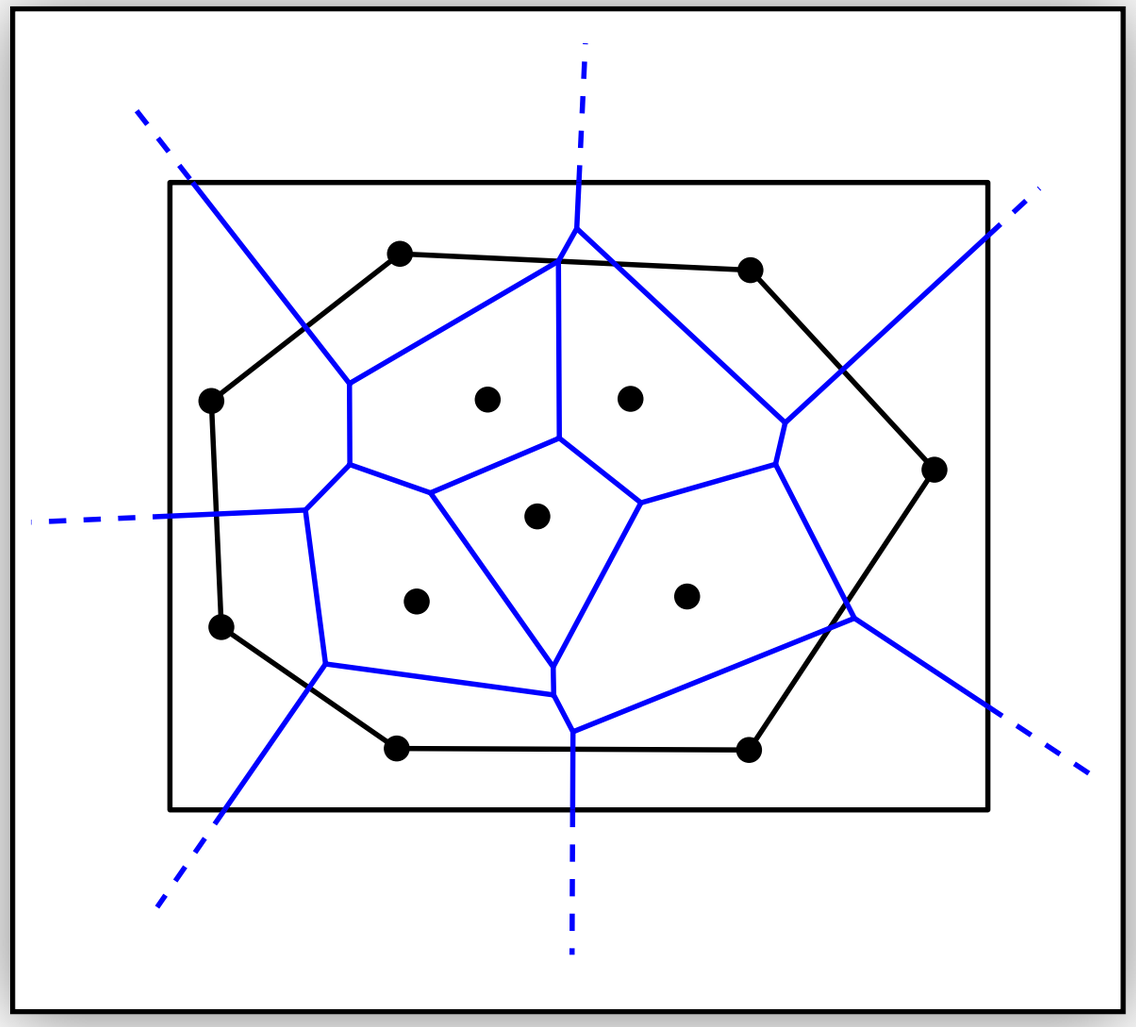
Unbounded edges:

- After last event in Q: $|B| \geq 2$



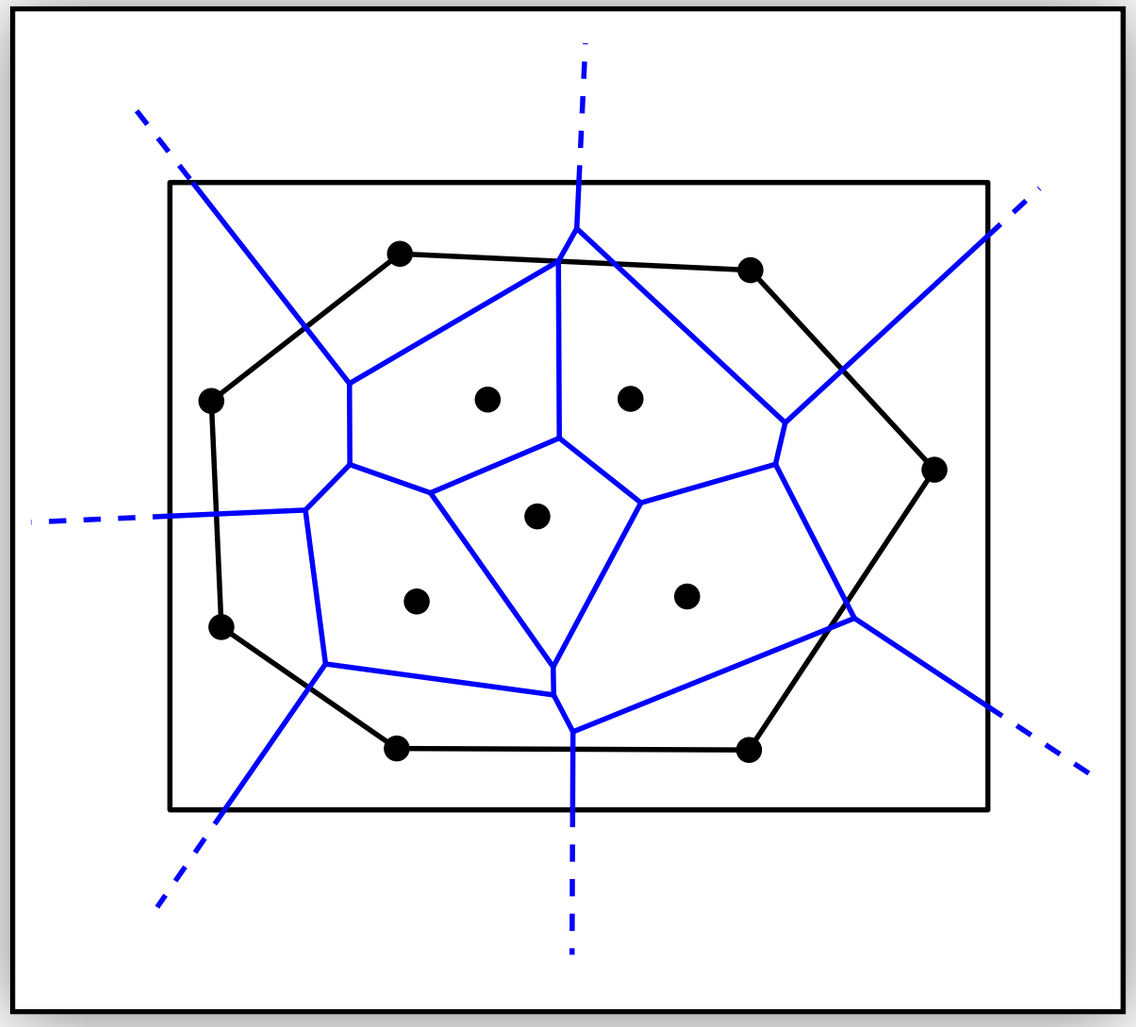
Unbounded edges:

- After last event in Q: $|B| \geq 2$
- $\rightarrow$ Two $p_1, p_2 \in Q$ form unbounded $e \subset B(p_1, p_2)$



Unbounded edges:

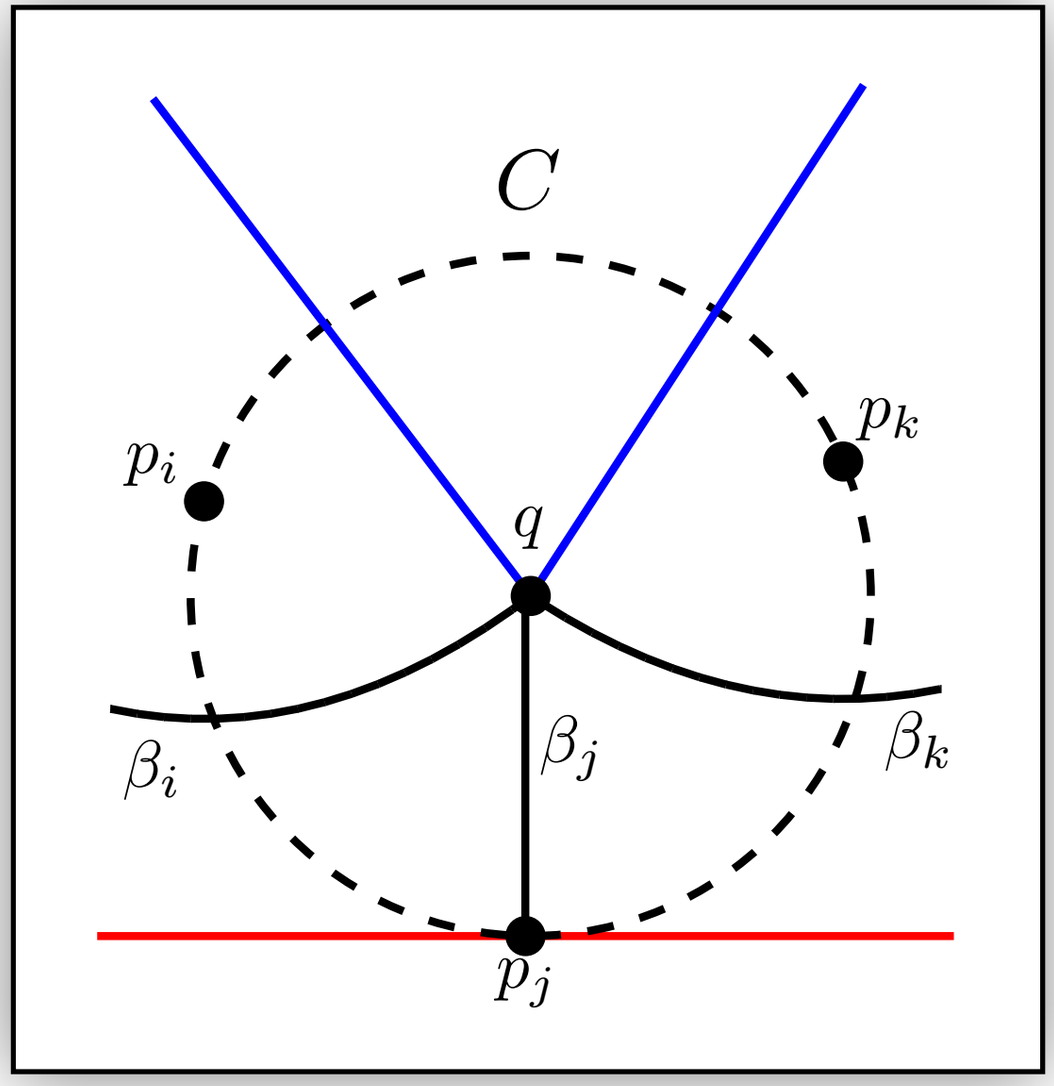
- After last event in Q: $|B| \geq 2$
- $\rightarrow$ Two $p_1, p_2 \in Q$ form unbounded $e \subset B(p_1, p_2)$
- Connect such edges with the bounding box.



Degeneracy

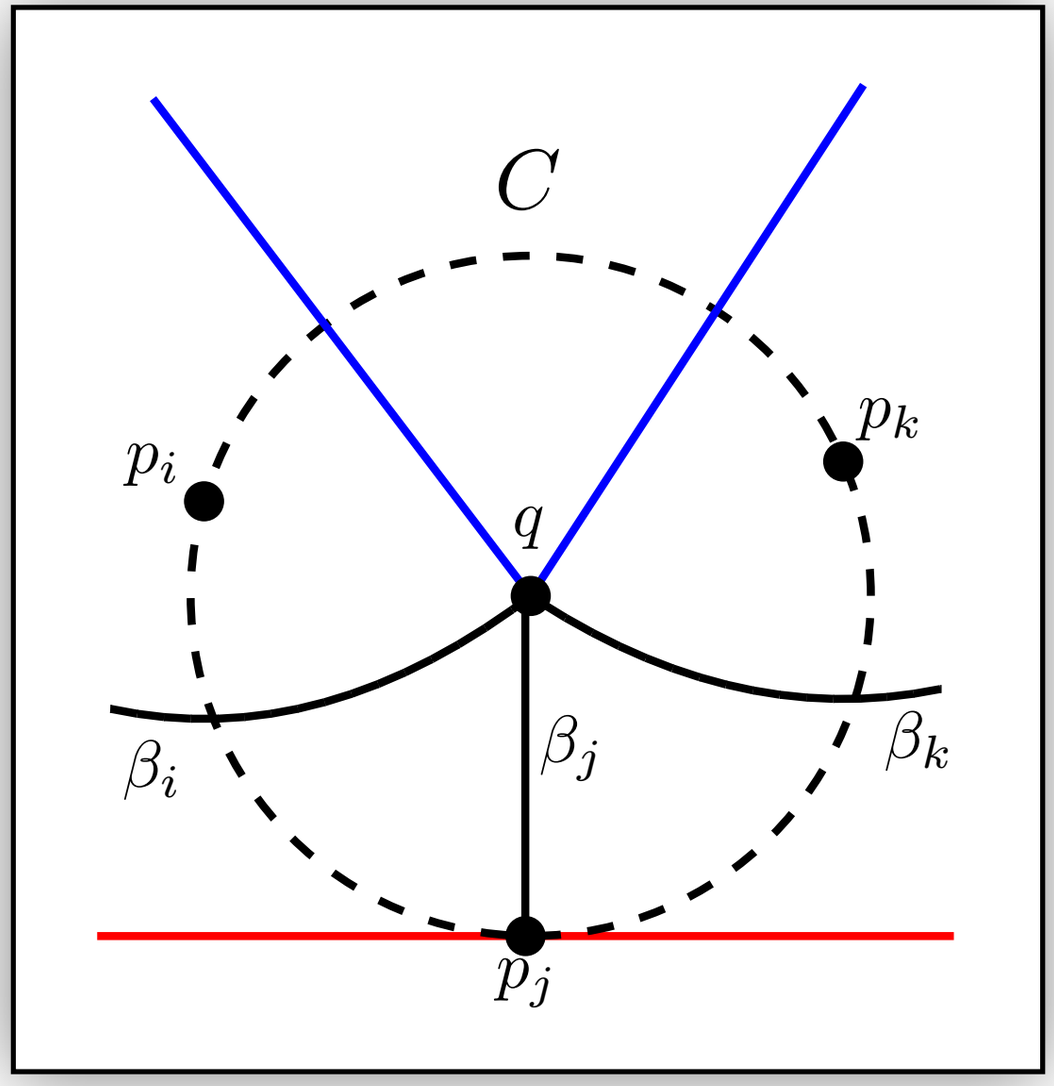
Degenerate situation:

Degenerate situation:



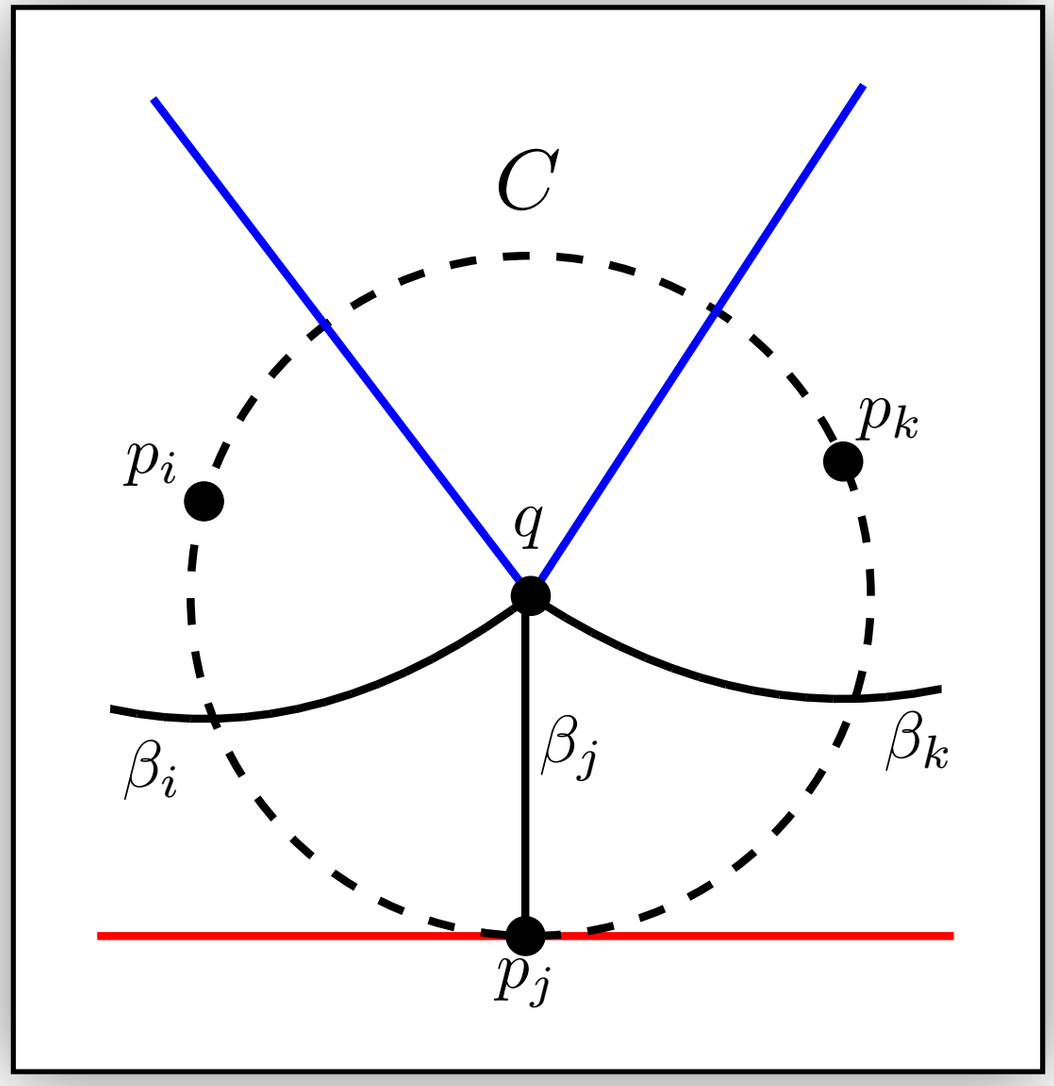
Degenerate situation:

- Point event p_j below q.



Degenerate situation:

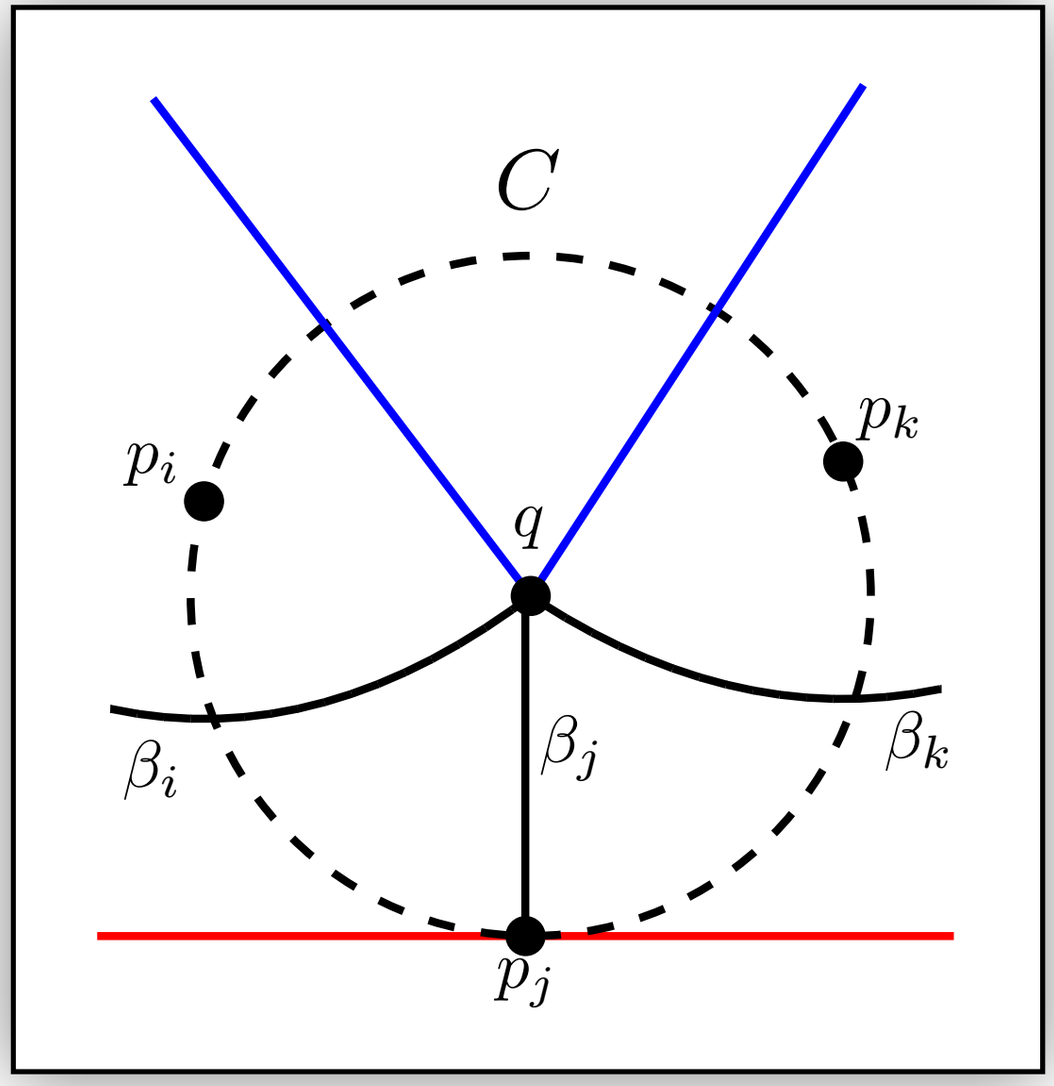
- Point event p_j below q.
  $\rightarrow p_j$ lowest point
  of $C := \bigcirc(p_i, p_j, p_k)$



Degenerate situation:

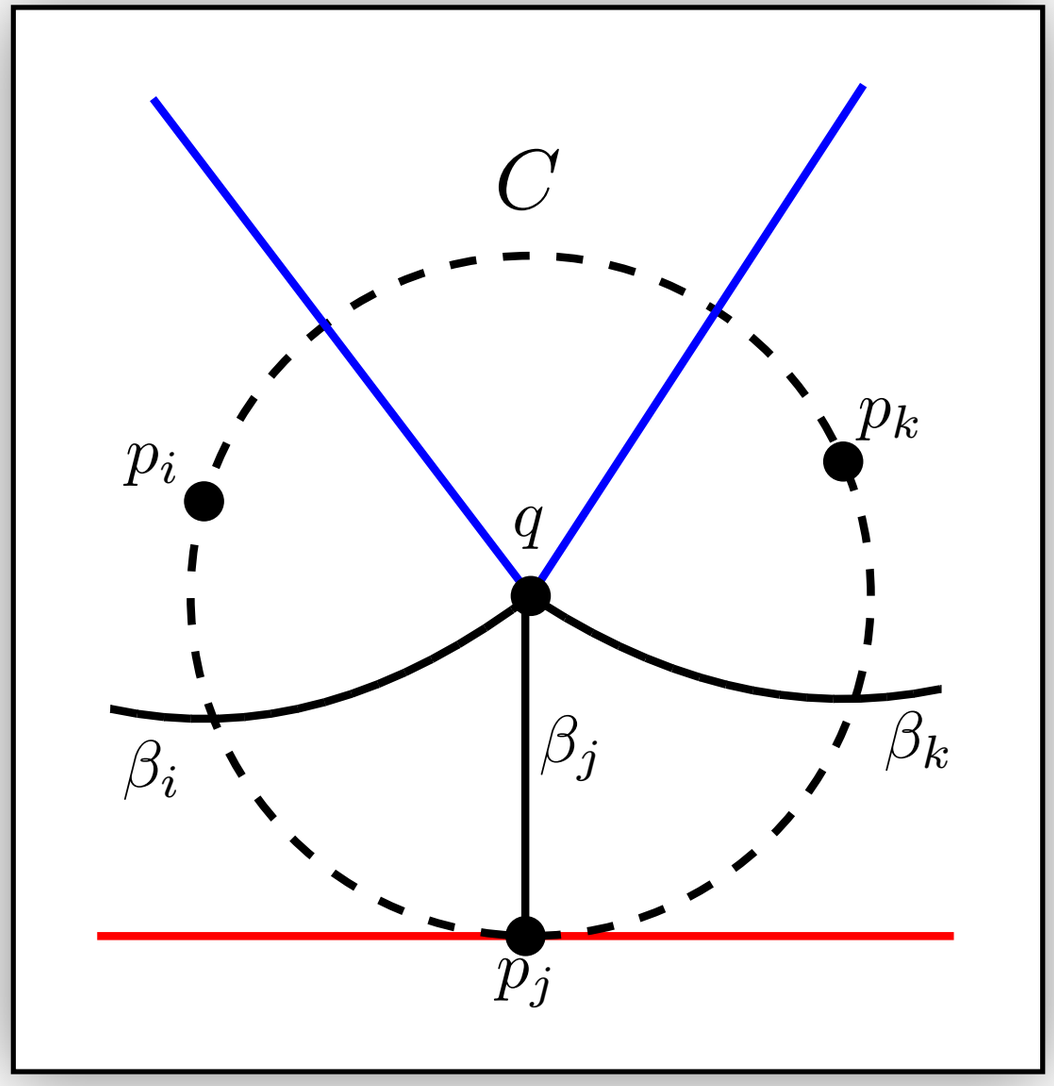
- Point event p_j below q.
  $\rightarrow p_j$ lowest point
  of $C := \bigcirc(p_i, p_j, p_k)$

- Simultaneously: Generate and reach circle events C.



Degenerate situation:

- Point event p_j below q.
  $\rightarrow p_j$ lowest point
  of $C := \bigcirc(p_i, p_j, p_k)$
- Simultaneously: Generate and reach circle events C.
- Generate Voronoi vertex q.



Overview

Algorithm 1: Computation of $V^*(S)$.
Input: S is a set of $n \geq 1$ points with unique bottommost point.
Output: The bisectors and vertices of V^*.
Data structures: Q: a priority queue of points in the plane, ordered lexicographically. Each point is labeled as a site, or labeled as the intersection of a pair of boundaries of a single region. Q may contain duplicate instances of the same point with distinct labels; the ordering of duplicates is irrelevant.
L: a sequence $(r_1, e_1, r_2, \ldots, r_k)$ of regions (labeled by site) and boundaries (labeled by a pair of sites). Note that a region can appear many times on L.

1. initialize Q with all sites
2. $p \leftarrow$ extract_min(Q)
3. $L \leftarrow$ the list containing R_p.
4. **while** Q is not empty **begin**
5. $p \leftarrow$ extract_min(Q)
6. **case**
7. p is a site:
8. Find an occurrence of a region R_q^* on L containing p.
9. Create bisector B_{pq}^*.
10. Update list L so that it contains $\ldots, R_q^*, C_{pq}^-, R_p^*, C_{pq}^+, R_q^*, \ldots$ in place of R_q^*.
11. Delete from Q the intersection between the left and right boundary of R_q^*, if any.
12. Insert into Q the intersection between C_{pq}^- and its neighbor to the left on L, if any, and the intersection between C_{pq}^+ and its neighbor to the right, if any.
13. p is an intersection:
14. Let p be the intersection of boundaries C_{qr} and C_{rs}.
15. Create the bisector B_{qs}^*.
16. Update list L so it contains $C_{qs} = C_{qs}^-$ or C_{qs}^+, as appropriate, instead of C_{qr}, R_r^*, C_{rs}.
17. Delete from Q any intersection between C_{qr} and its neighbor to the left and between C_{rs} and its neighbor to the right.
18. Insert any intersections between C_{qs} and its neighbors to the left or right into Q.
19. Mark p as a vertex and as an endpoint of B_{qr}^*, B_{rs}^*, and B_{qs}^*.
20. **end**

Fig. 2.4. Algorithm 1: computation of $V^*(s)$.

Analysis

Analysis

- x-structure B, event queue Q: cost of $\mathcal{O}(\log n)$ per operation

Analysis

- x-structure B, event queue Q: cost of $\mathcal{O}(\log n)$ per operation
- $\mathcal{O}(n)$ point events

Analysis

- x-structure B, event queue Q: cost of $\mathcal{O}(\log n)$ per operation
- $\mathcal{O}(n)$ point events
- $\mathcal{O}(n)$ Voronoi vertices

Analysis

- x-structure B, event queue Q: cost of $\mathcal{O}(\log n)$ per operation
- $\mathcal{O}(n)$ point events
- $\mathcal{O}(n)$ Voronoi vertices

  $\Rightarrow \mathcal{O}(n)$ processed circle events

Analysis

- x-structure B, event queue Q: cost of $\mathcal{O}(\log n)$ per operation
- $\mathcal{O}(n)$ point events
- $\mathcal{O}(n)$ Voronoi vertices

  $\Rightarrow \mathcal{O}(n)$ processed circle events

  $\Rightarrow \mathcal{O}(n)$ total circle events
  (processed circle event generates
  $\mathcal{O}(1)$ new circle events)

Analysis

- x-structure B, event queue Q: cost of $\mathcal{O}(\log n)$ per operation
- $\mathcal{O}(n)$ point events
- $\mathcal{O}(n)$ Voronoi vertices

  $\Rightarrow \mathcal{O}(n)$ processed circle events

  $\Rightarrow \mathcal{O}(n)$ total circle events
  (processed circle event generates
  $\mathcal{O}(1)$ new circle events)

Theorem 4.23

Fortune's algorithm computes the Voronoi diagram of n points in time $\Theta(n \log n)$.

Analysis

- x-structure B, event queue Q: cost of $\mathcal{O}(\log n)$ per operation
- $\mathcal{O}(n)$ point events
- $\mathcal{O}(n)$ Voronoi vertices

  $\Rightarrow \mathcal{O}(n)$ processed circle events

  $\Rightarrow \mathcal{O}(n)$ total circle events
  (processed circle event generates
  $\mathcal{O}(1)$ new circle events)

Theorem 4.23

Fortune's algorithm computes the Voronoi diagram of n points in time $\Theta(n \log n)$.

THEOREM 2.8. *Algorithm 1 can be implemented to run in time $O(n \log n)$ and space $O(n)$.*

Steve’s Questions

Steve's Questions

Steven Fortune 4. Dezember 2020 um 14:48

Aw: Computational Geometry - video message?

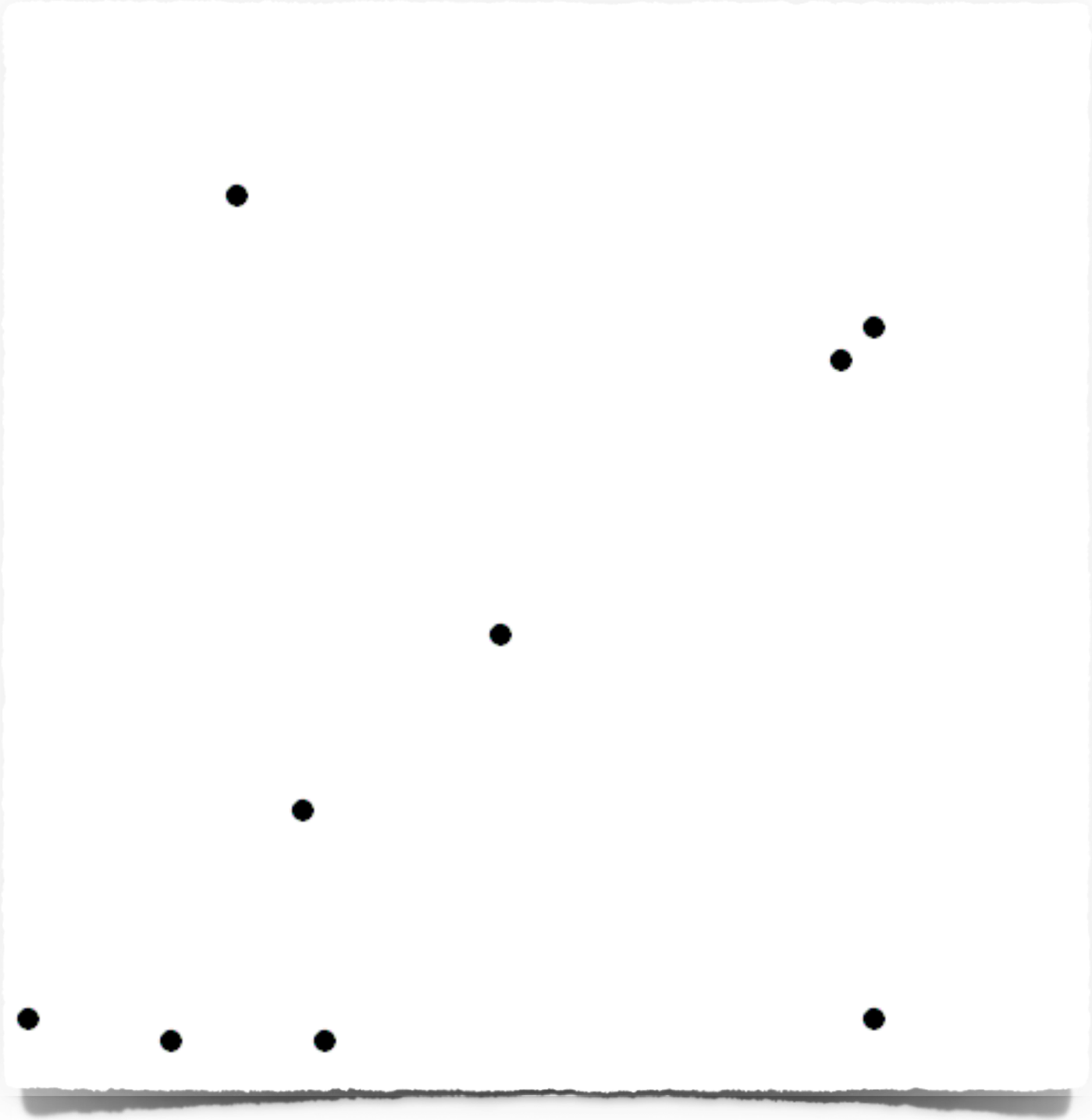
An: Sándor Fekete,

Umgeleitet von: fekete@tu-braunschweig.de

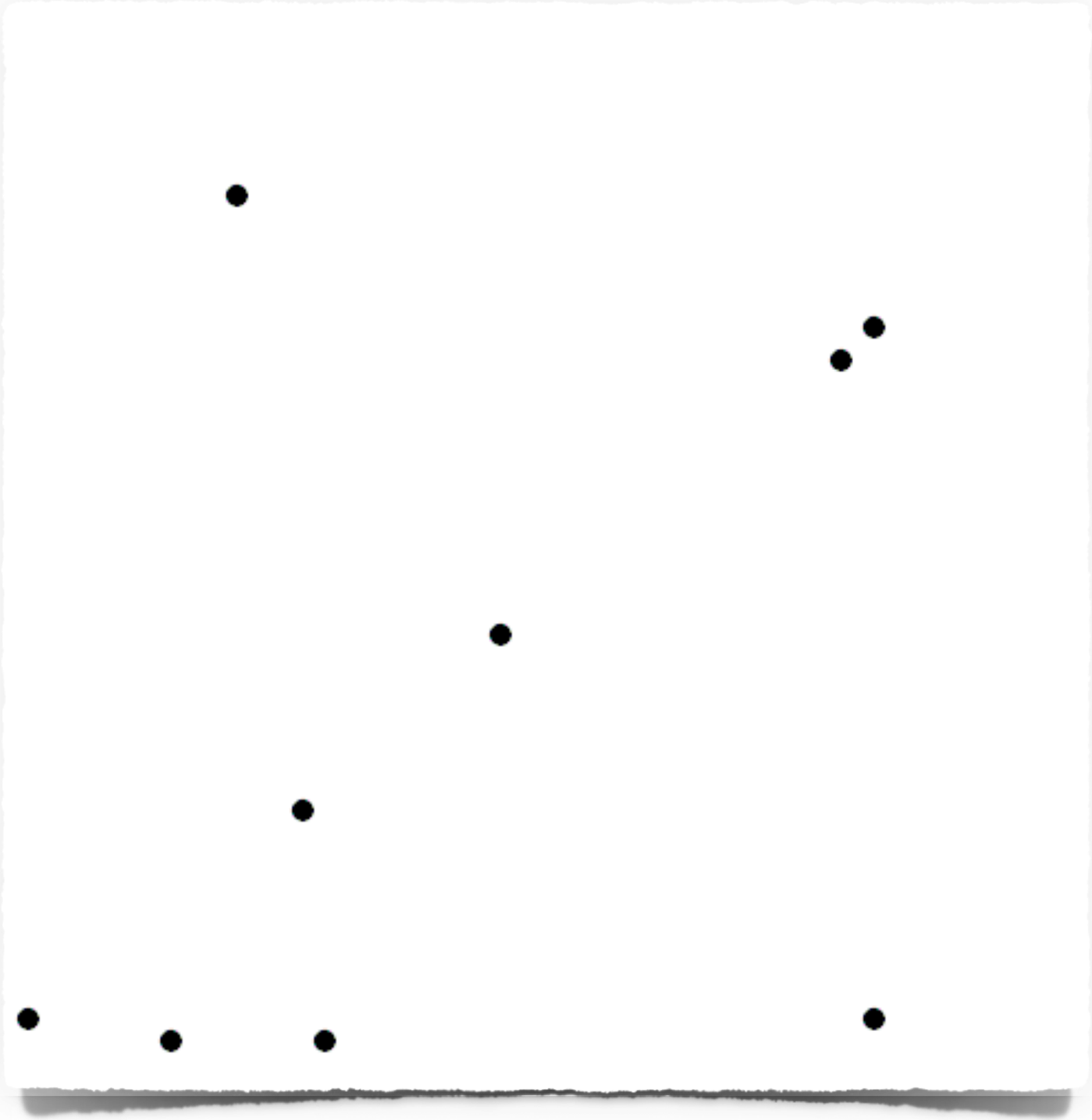
The ways I know:

1. original paper, transforming the plane
2. beach line (maintaining the "known" part of the VD)
3. sweep line maintains the top of circles
4. in 3d, sweep a plane at 45 degrees to xy, watch it intersect 45 degree cones centered at siteas

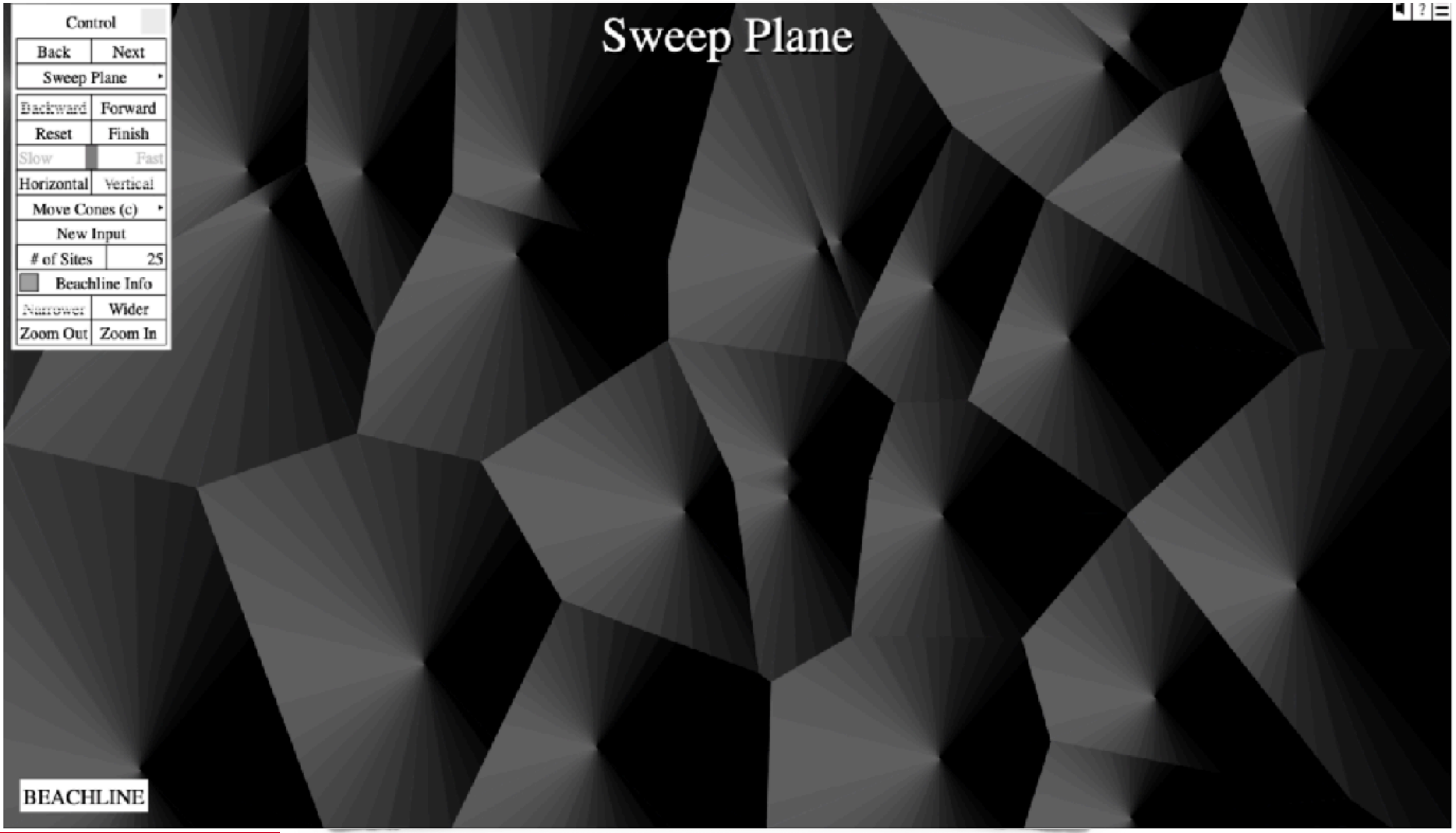
Steve’s Questions

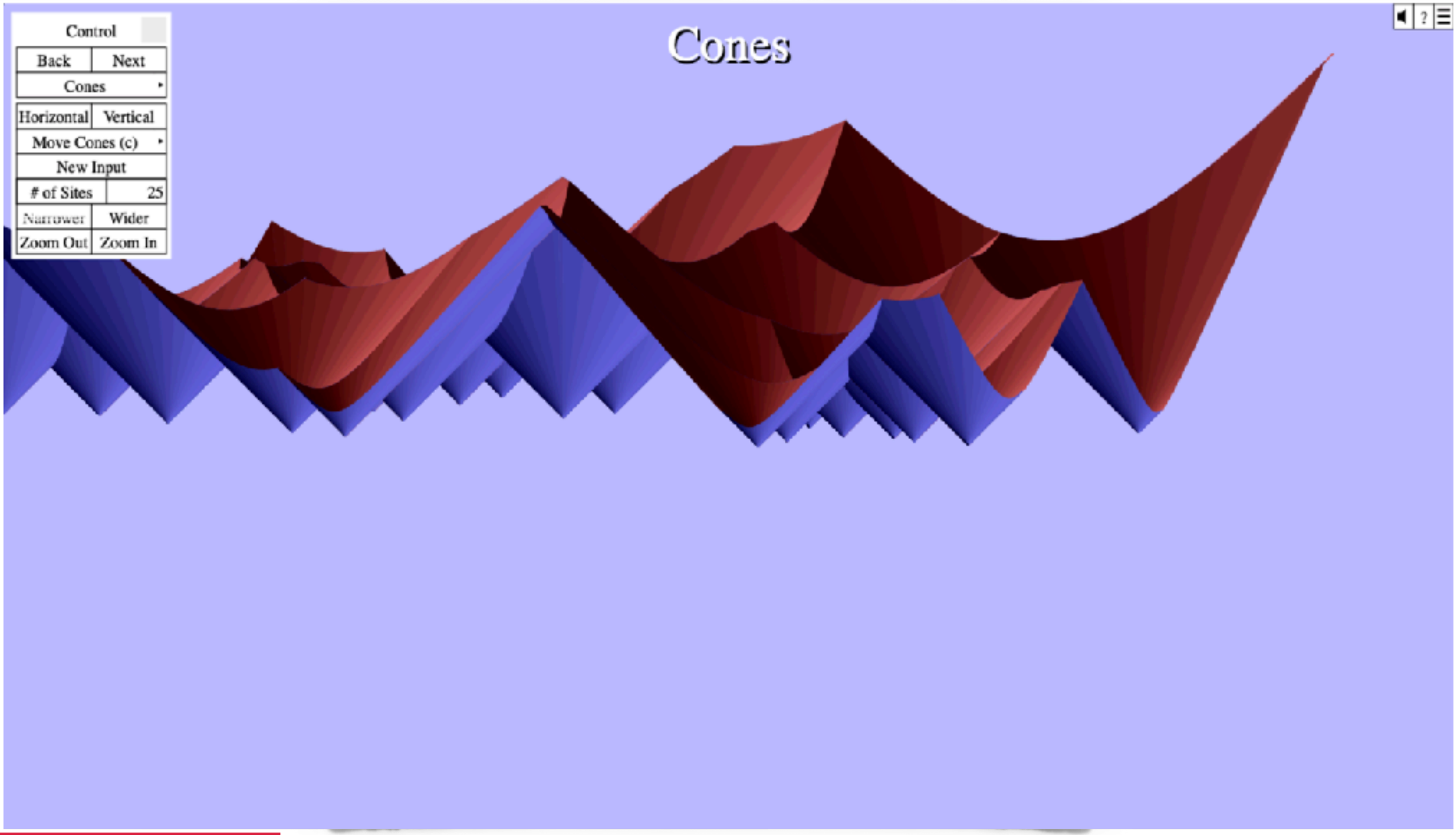


Steve’s Questions



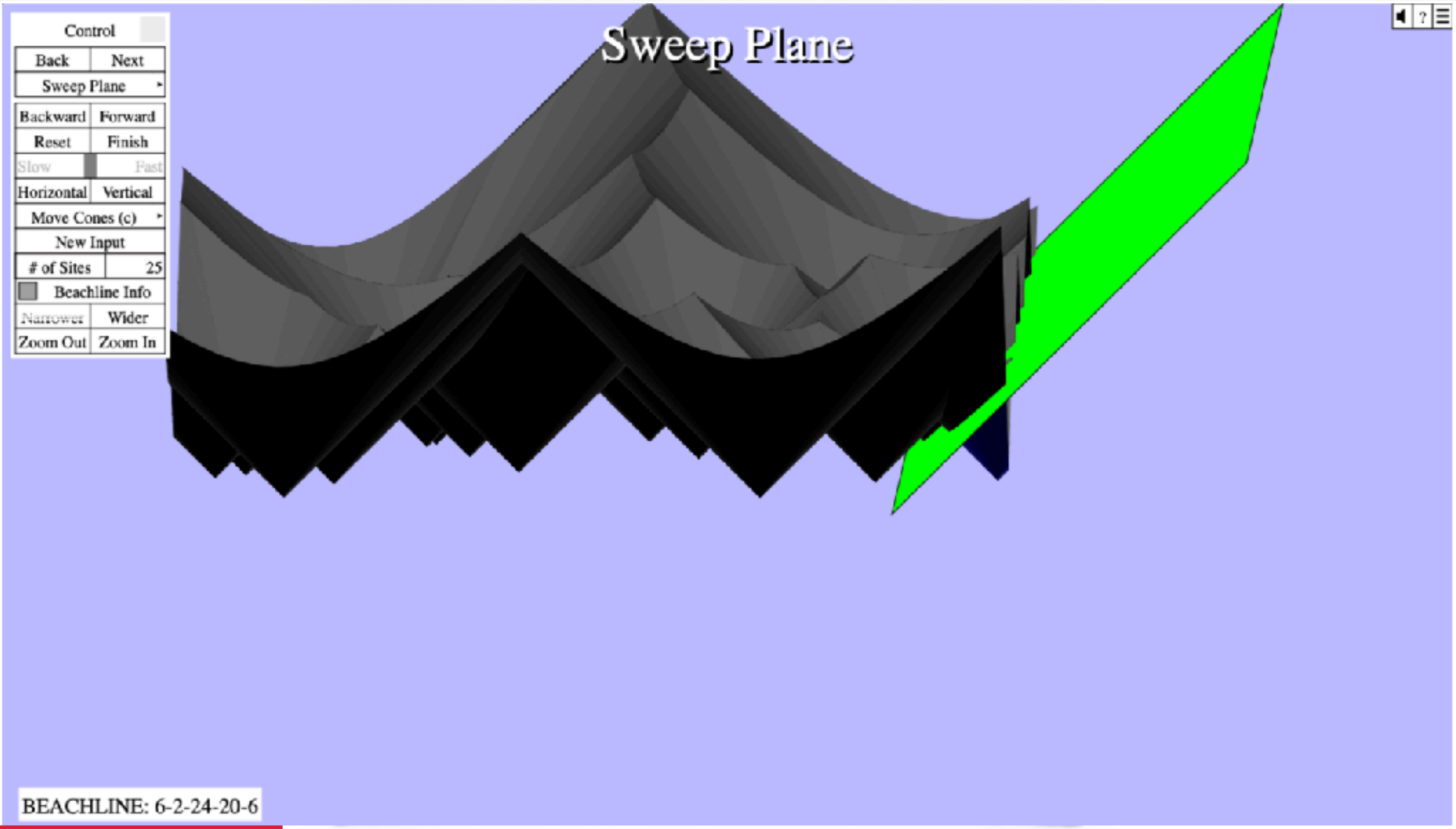
Steve's Questions



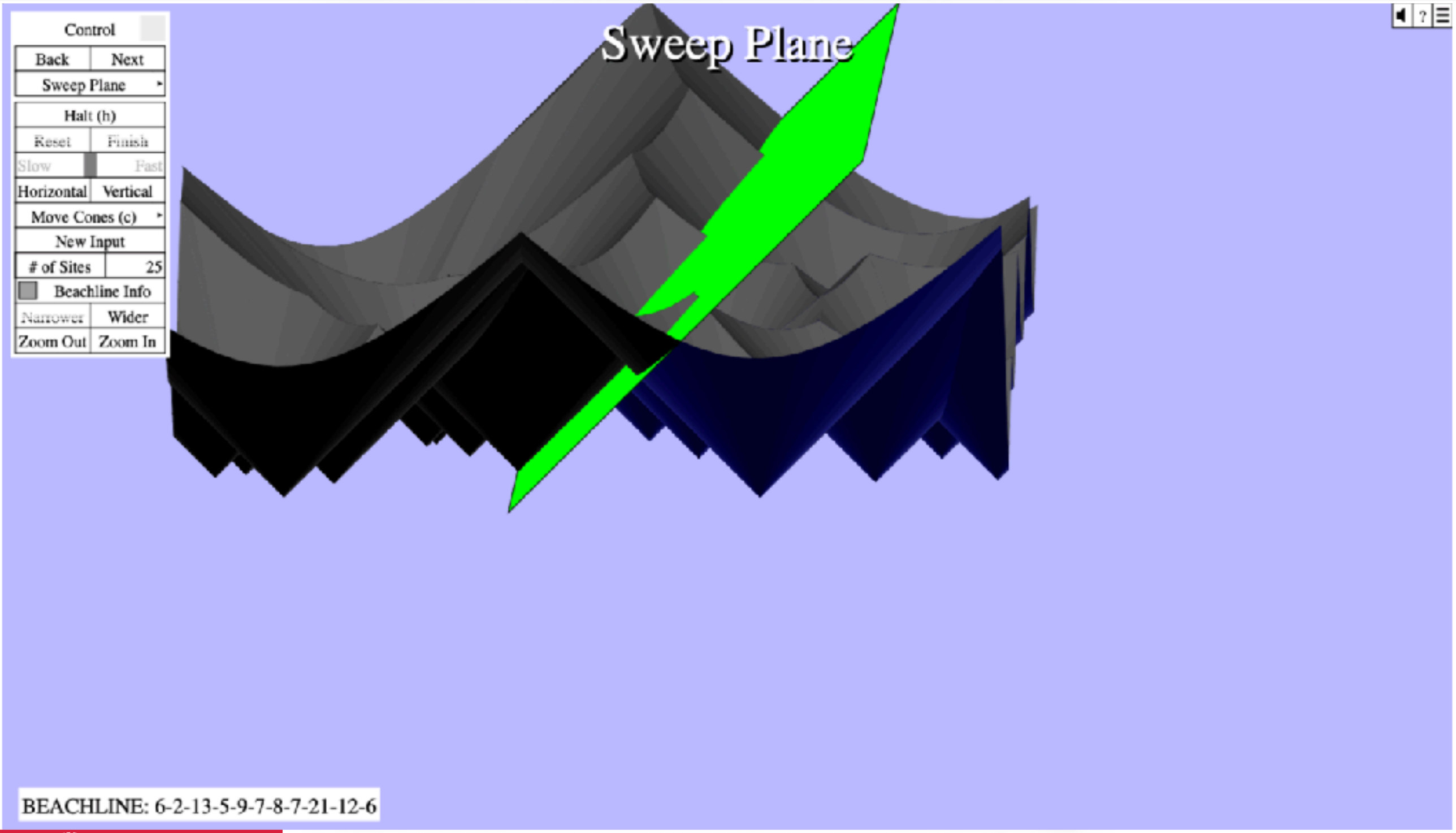

Control
Back
Next
Cones
Horizontal
Vertical
Move Cones (c)
New Input
# of Sites
25
Narrower
Wider
Zoom Out
Zoom In
Cones

Steve's Questions

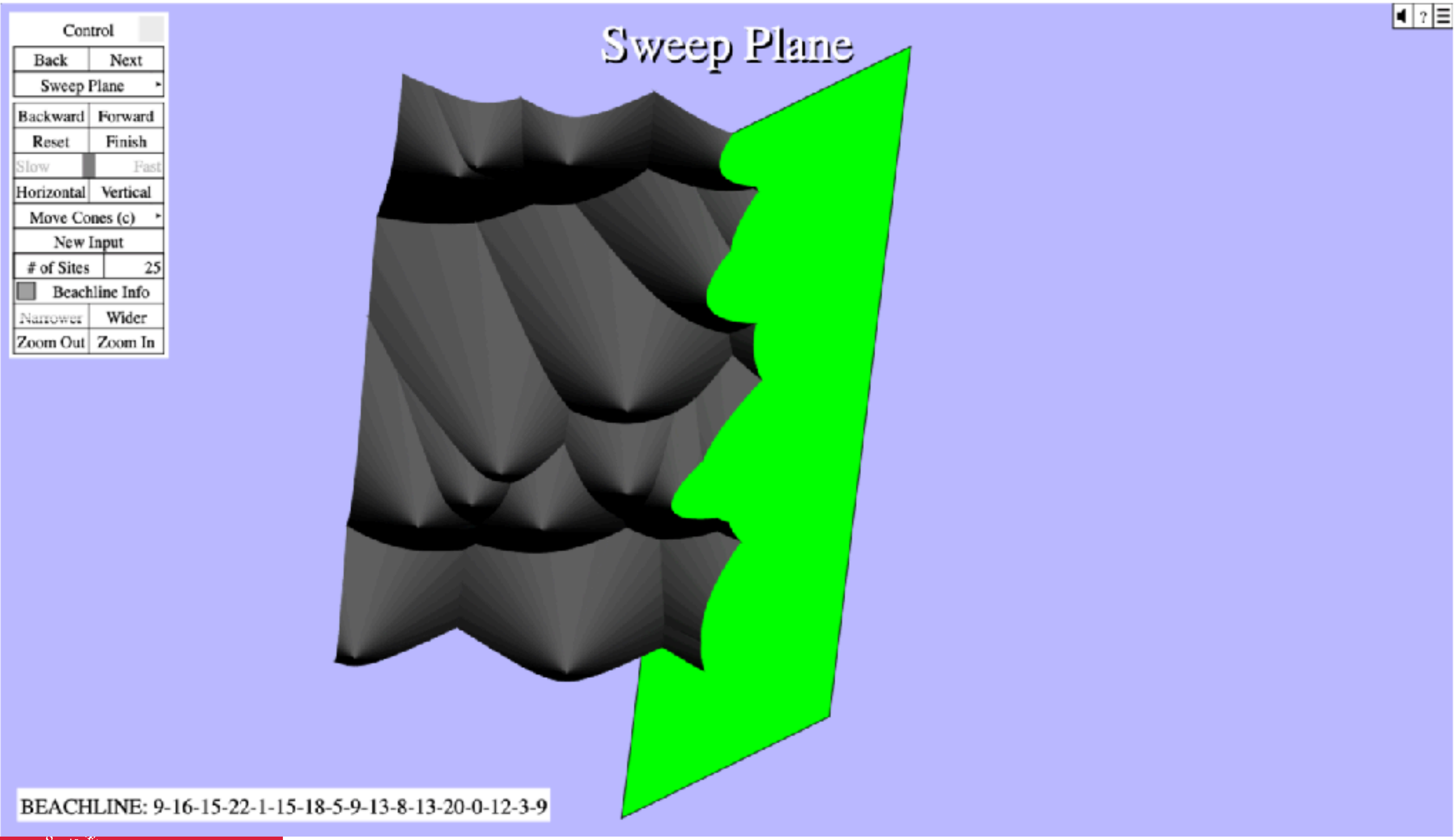
Steve's Questions



Steve's Questions



Steve's Questions



1. Introduction and Motivation
2. Definitions
3. Representing planar partitions
4. Properties
5. Fortune's algorithm
6. Variations
7. The Voronoi Game
8. Summary and conclusions

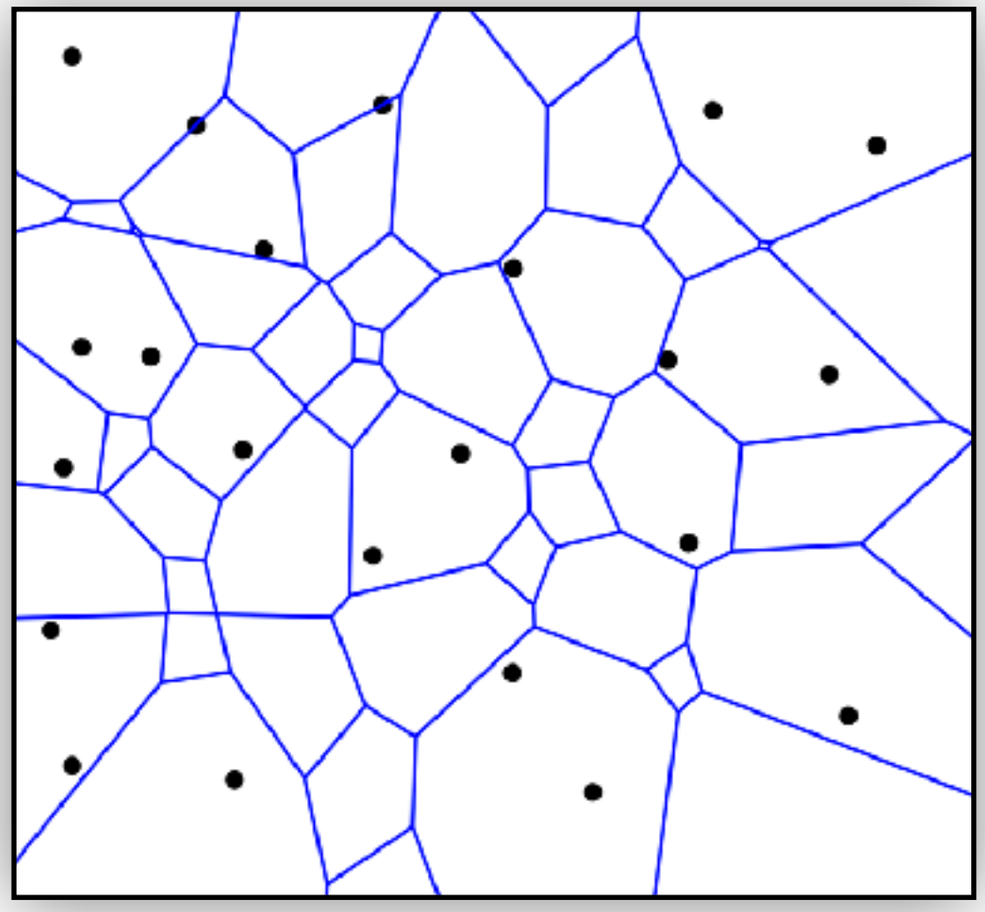
Higher-order Voronoi diagrams:

Higher-order Voronoi diagrams:

- Voronoi region := point set with same set of k nearest neighbors

Higher-order Voronoi diagrams:

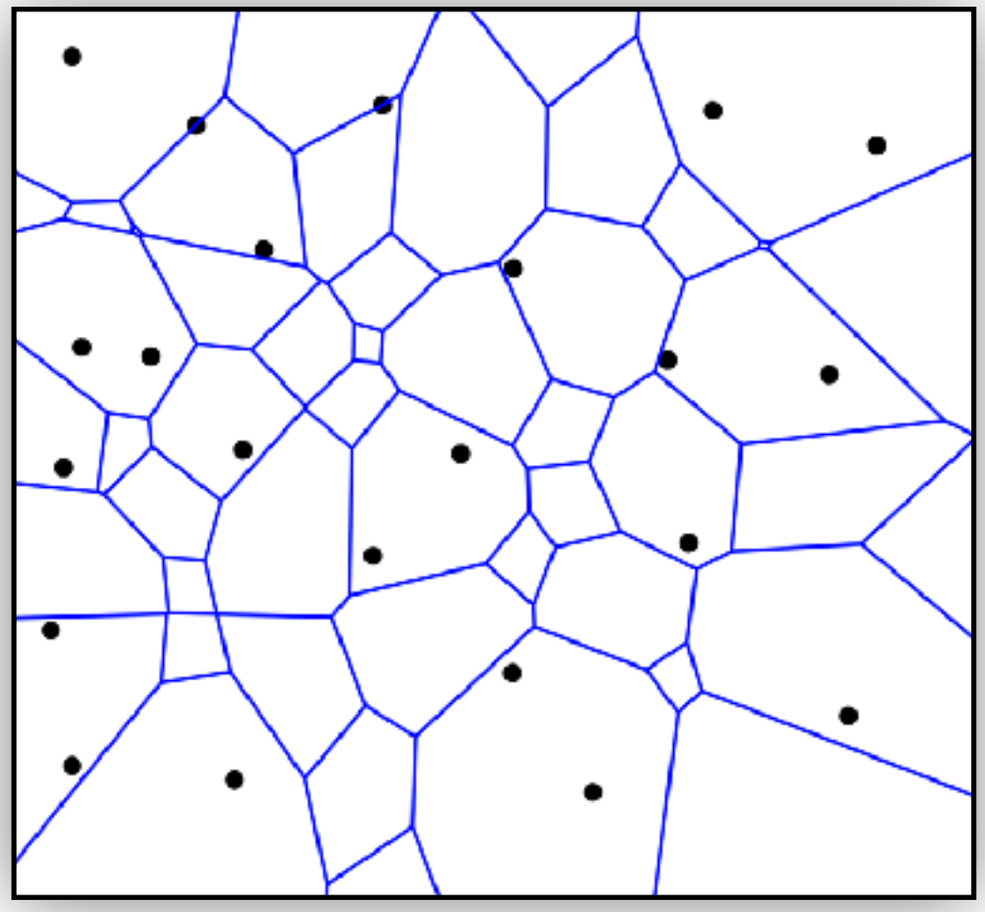
- Voronoi region := point set with same set of k nearest neighbors



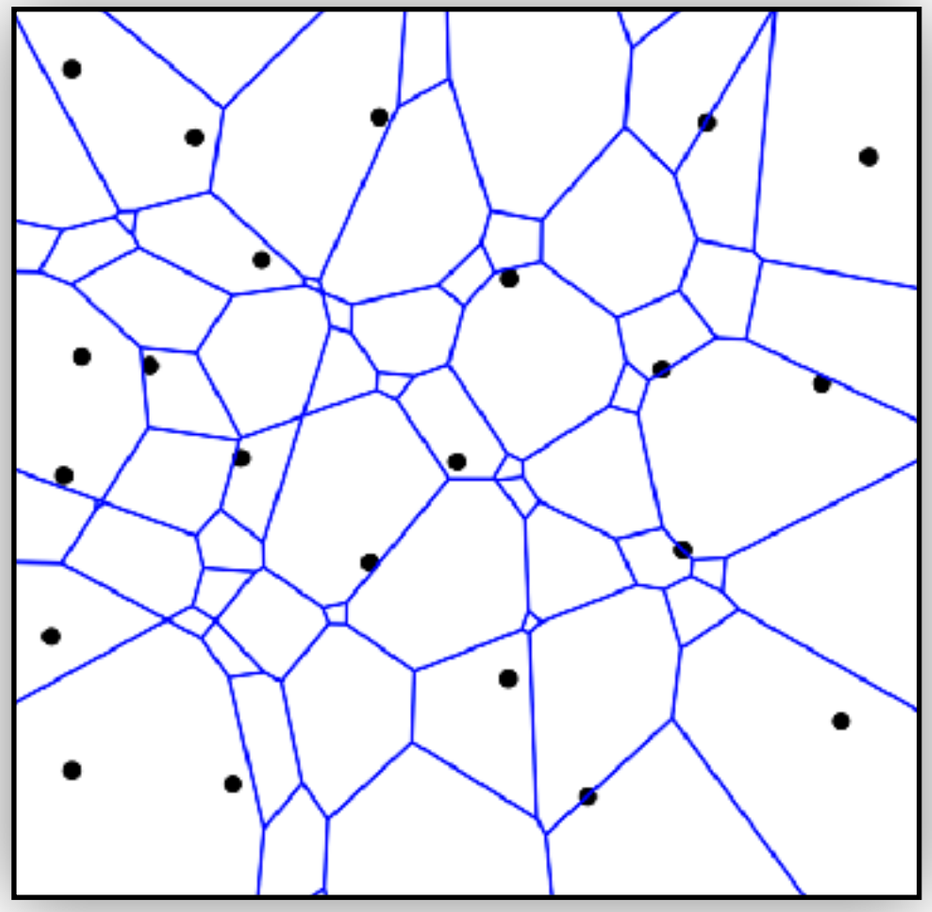
$k = 2$

Higher-order Voronoi diagrams:

- Voronoi region := point set with same set of k nearest neighbors



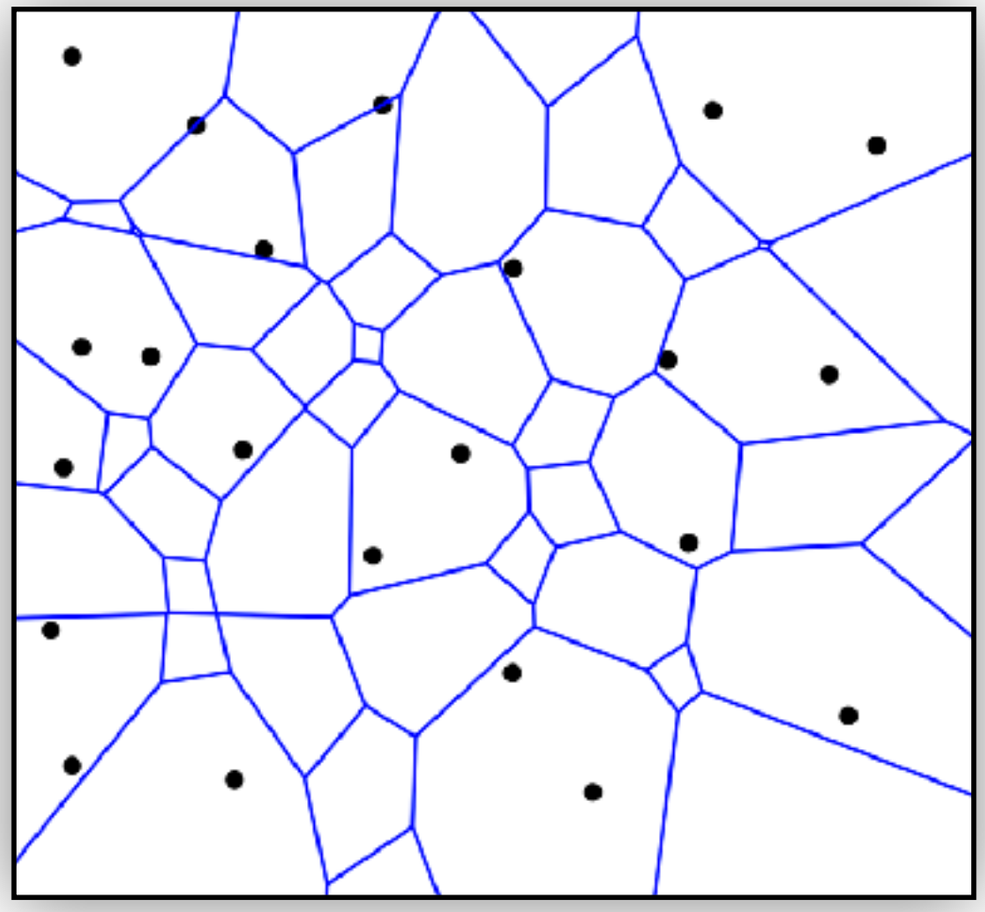
$k = 2$



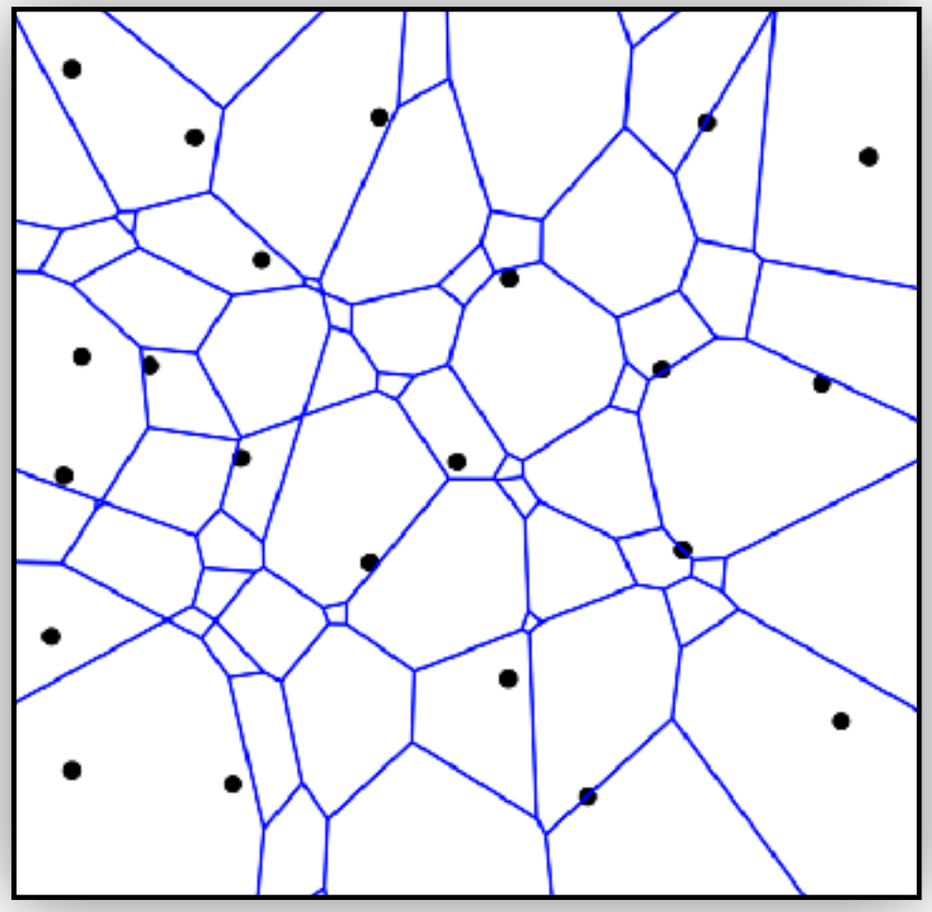
$k = 3$

Higher-order Voronoi diagrams:

- Voronoi region := point set with same set of k nearest neighbors



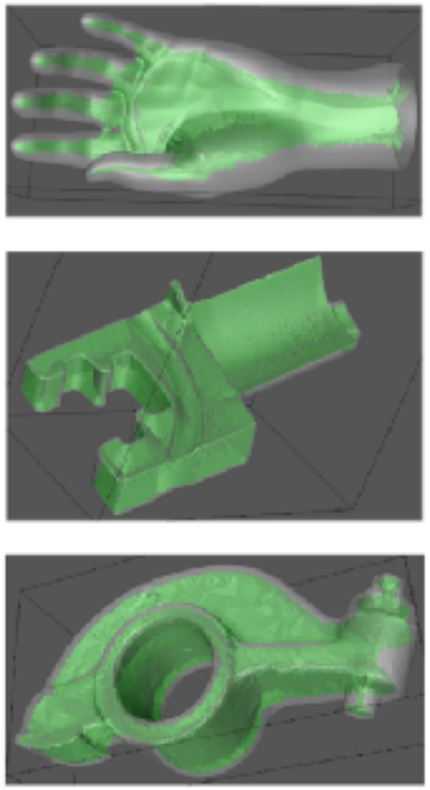
$k = 2$



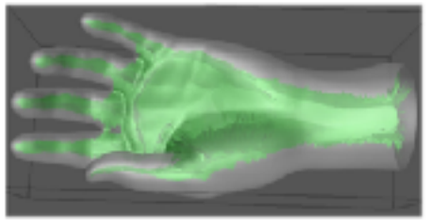
$k = 3$

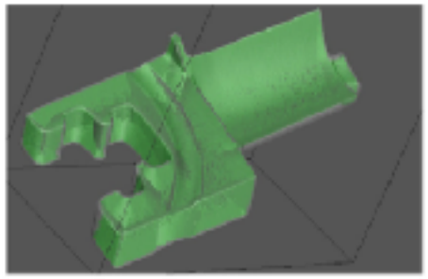
- *Worst-Case*-optimal: $\mathcal{O}(n^3)$ for arbitrary (but fixed) $k \geq 2$
  [Edelsbrunner and Seidel, 1986].

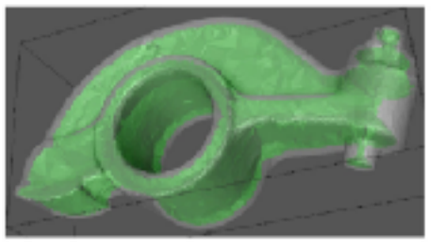
Medial Axis



Medial axis of a simple polygon

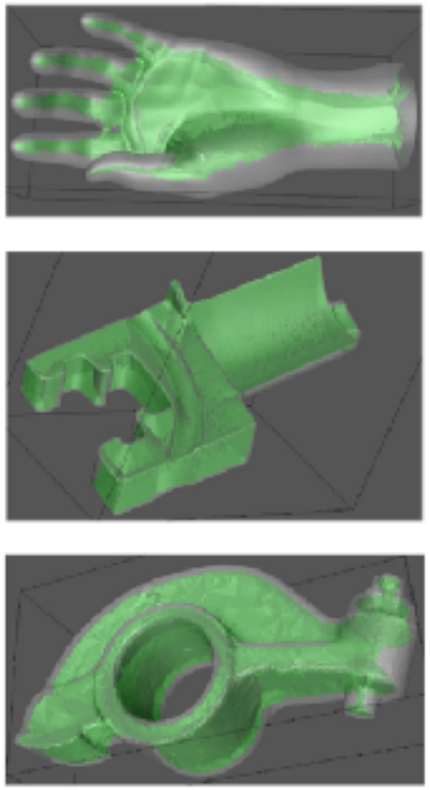






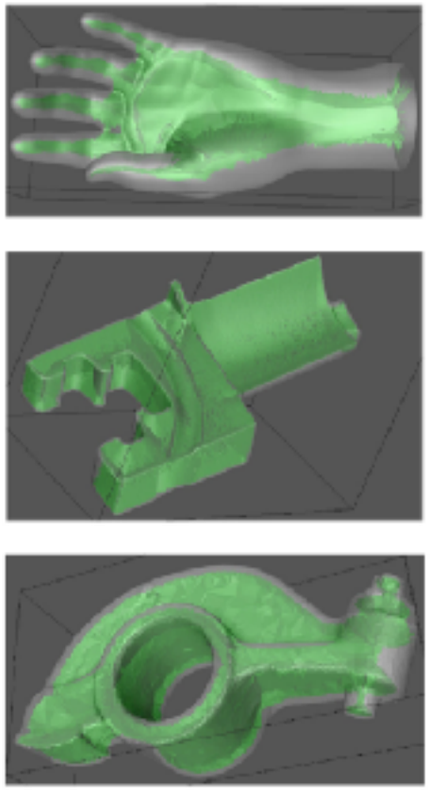
Medial axis of a simple polygon

- Analogously: polygon instead of $\mathcal{P}$



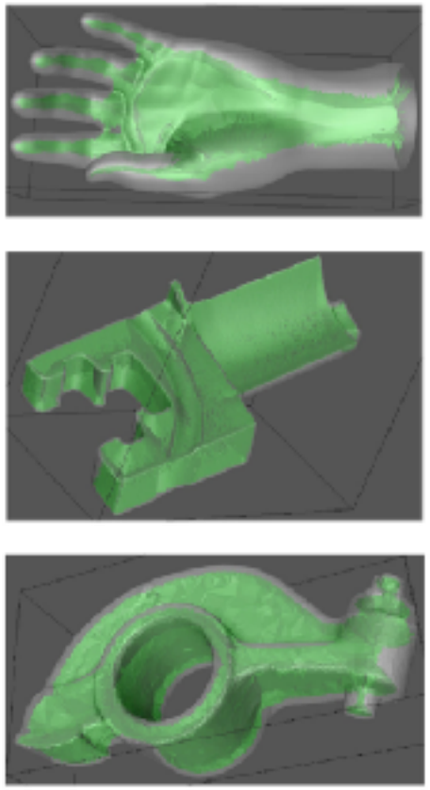
Medial axis of a simple polygon

- Analogously: polygon instead of $\mathcal{P}$
- Voronoi edges and vertices: two or three nearest neighbors on polygon boundary



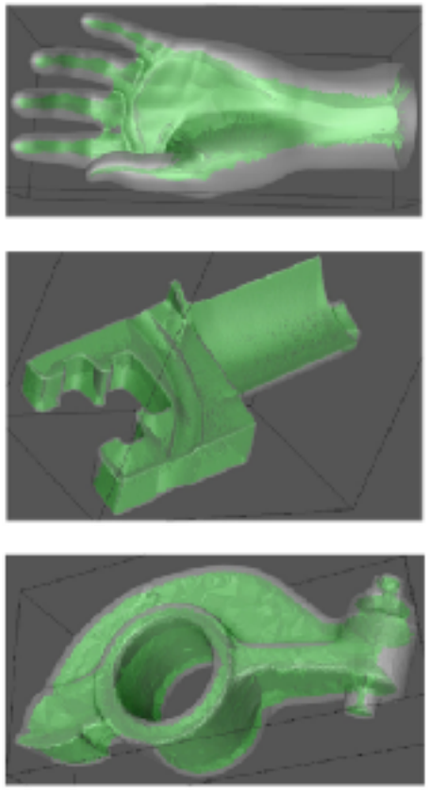
Medial axis of a simple polygon

- Analogously: polygon instead of $\mathcal{P}$
- Voronoi edges and vertices: two or three nearest neighbors on polygon boundary
- Computation for simple polygons in linear(!) time [Chin *et al.,* 1995].



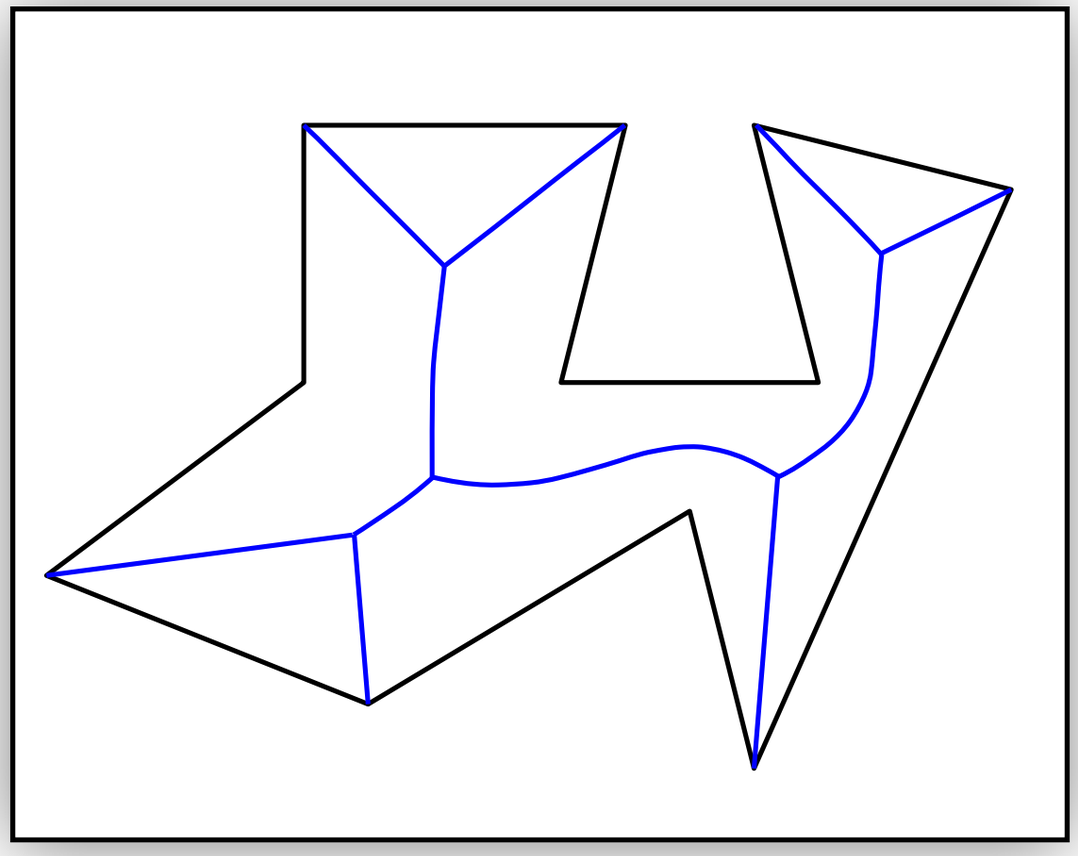
Medial axis of a simple polygon

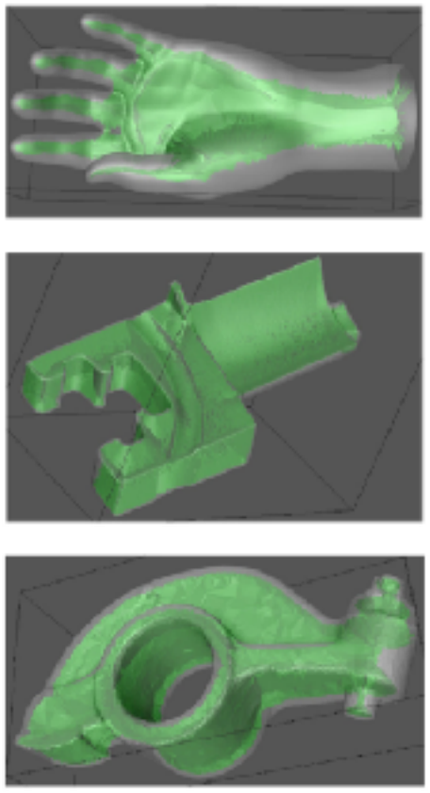
- Analogously: polygon instead of $\mathcal{P}$
- Voronoi edges and vertices: two or three nearest neighbors on polygon boundary
- Computation for simple polygons in linear(!) time [Chin *et al.,* 1995].
- Application: Surface reconstruction



Medial axis of a simple polygon

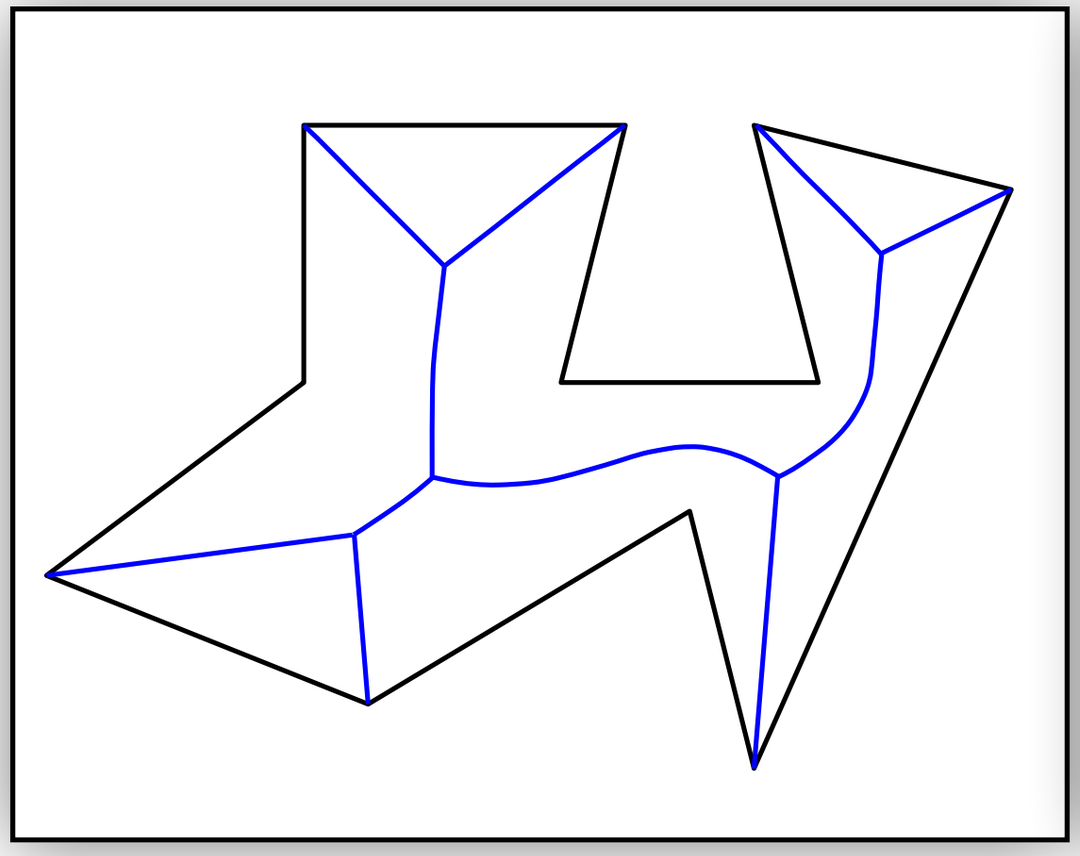
- Analogously: polygon instead of $\mathcal{P}$
- Voronoi edges and vertices: two or three nearest neighbors on polygon boundary
- Computation for simple polygons in linear(!) time [Chin *et al.,* 1995].
- Application: Surface reconstruction

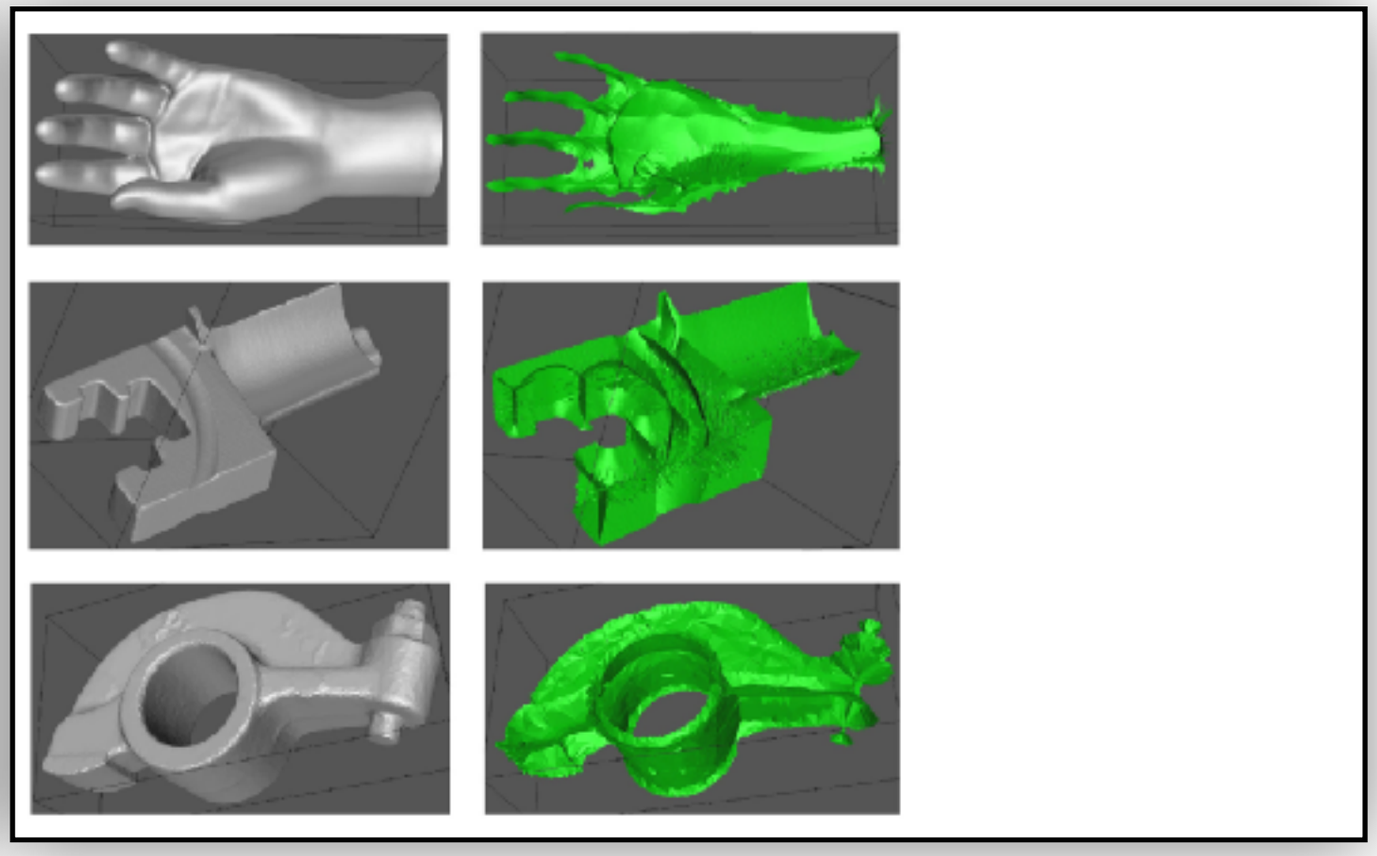




Medial axis of a simple polygon

- Analogously: polygon instead of $\mathcal{P}$
- Voronoi edges and vertices: two or three nearest neighbors on polygon boundary
- Computation for simple polygons in linear(!) time [Chin *et al.,* 1995].
- Application: Surface reconstruction





Straight Skeleton

Straight skeleton of a simple polygon:

Straight skeleton of a simple polygon:

- Voronoi edges and vertices: Intersection points of parallel wave fronts [Aichholzer *et al.,* 1995].

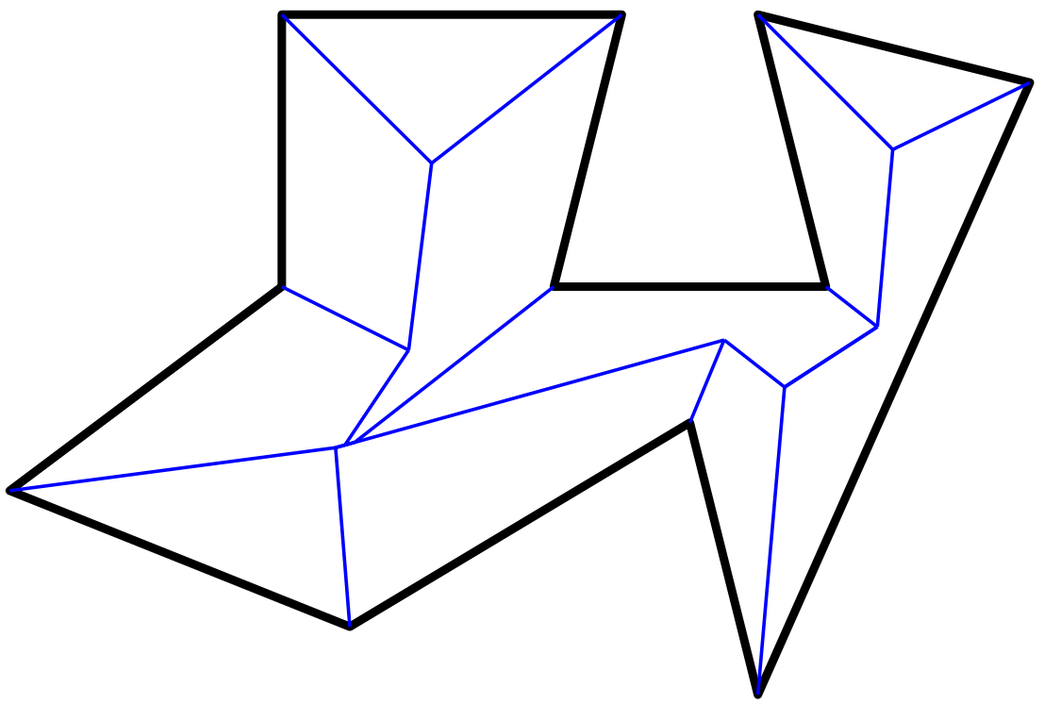
Straight skeleton of a simple polygon:

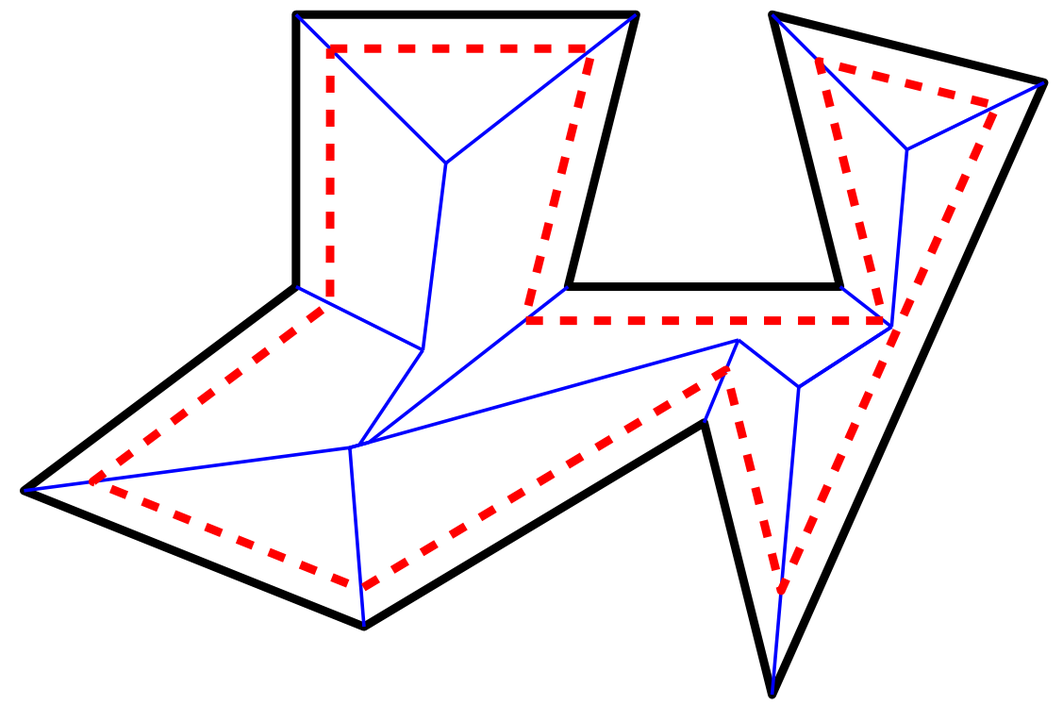
- Voronoi edges and vertices: Intersection points of parallel wave fronts [Aichholzer *et al.,* 1995].

- Computation in $\mathcal{O}(mn + n \log n)$ with $m =$: # reflex vertices [Felkel and Obdrzalek, 1998].

Straight skeleton of a simple polygon:

- Voronoi edges and vertices: Intersection points of parallel wave fronts [Aichholzer *et al.,* 1995].

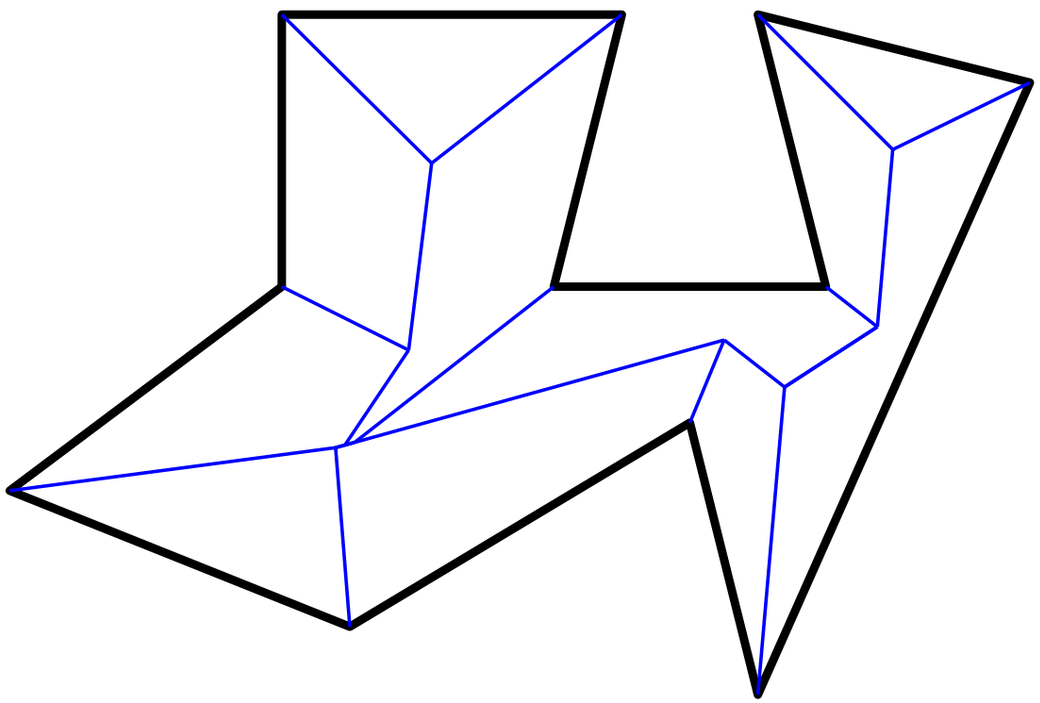




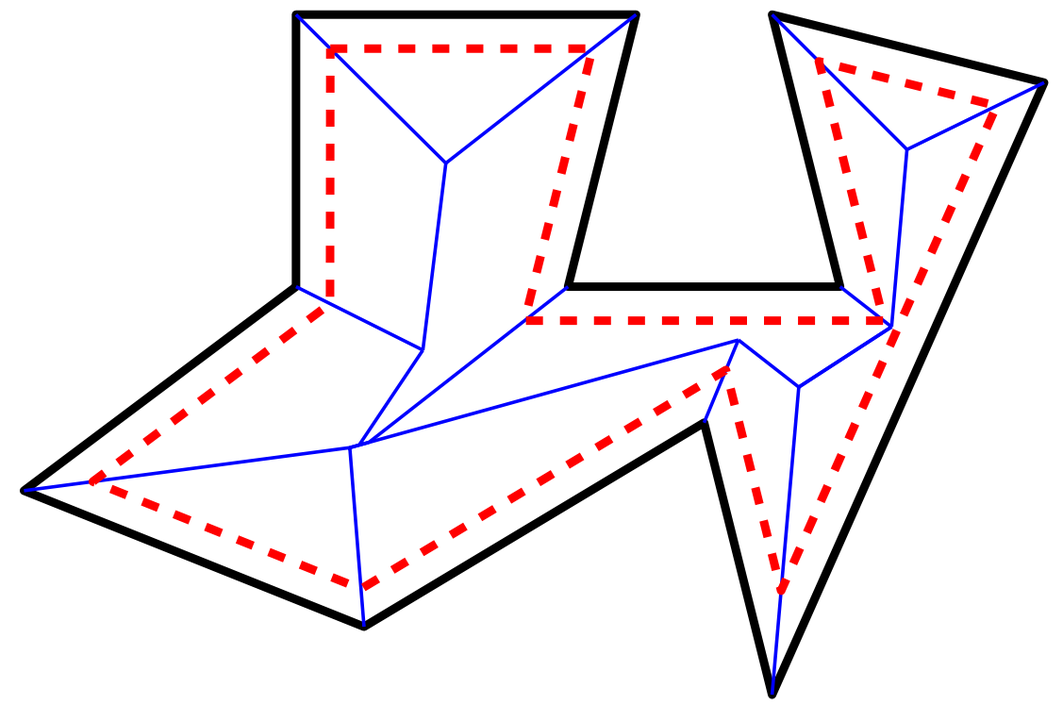
- Computation in $\mathcal{O}(mn + n \log n)$ with $m =$: # reflex vertices [Felkel and Obdrzalek, 1998].

Straight skeleton of a simple polygon:

- Voronoi edges and vertices: Intersection points of parallel wave fronts [Aichholzer *et al.,* 1995].



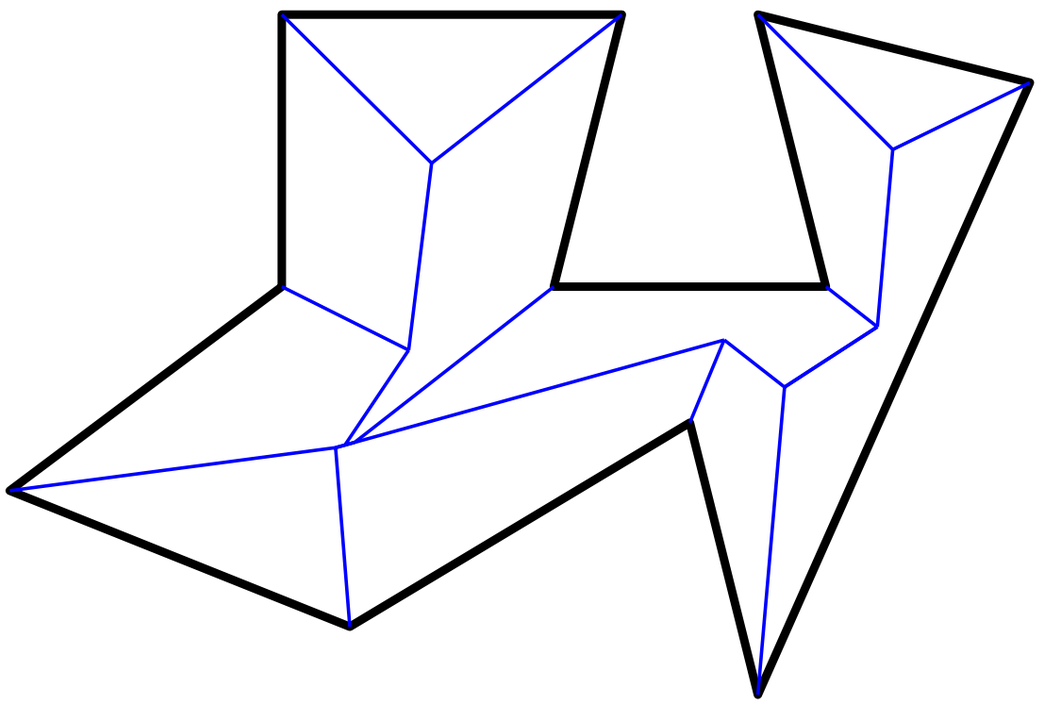
Straight Skeleton



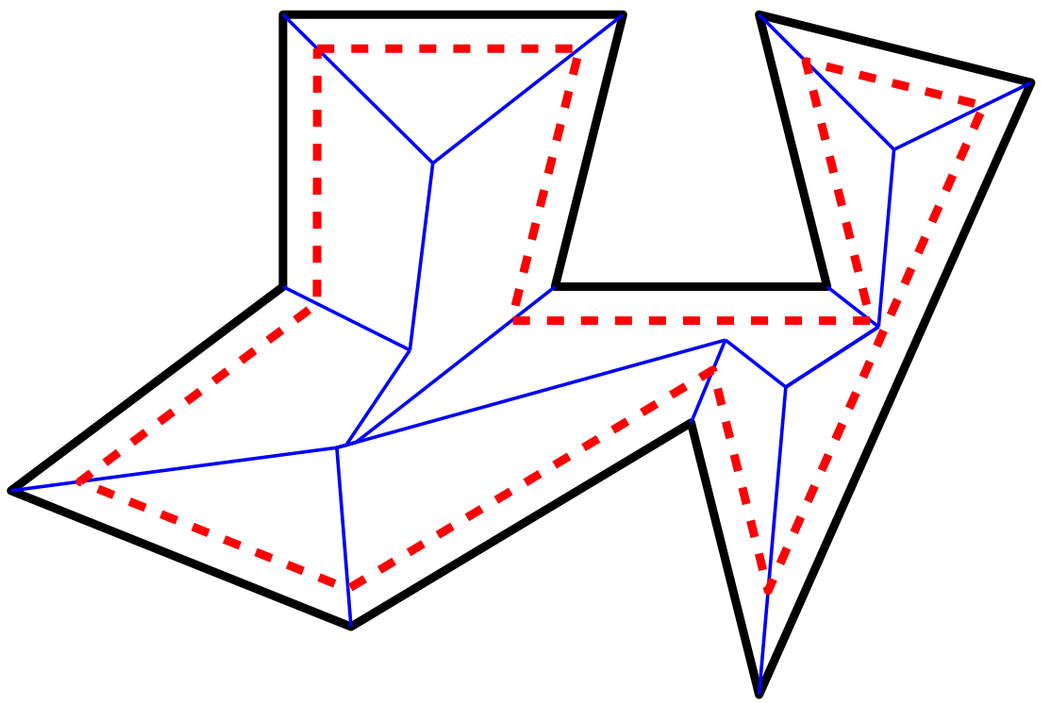
- Computation in $\mathcal{O}(mn + n \log n)$ with $m =$: # reflex vertices [Felkel and Obdrzalek, 1998].

Straight skeleton of a simple polygon:

- Voronoi edges and vertices: Intersection points of parallel wave fronts [Aichholzer *et al.,* 1995].



Straight Skeleton



Parallel wave front

- Computation in $\mathcal{O}(mn + n \log n)$ with $m =:$ # reflex vertices [Felkel and Obdrzalek, 1998].

Furthest-Point Voronoi Diagram

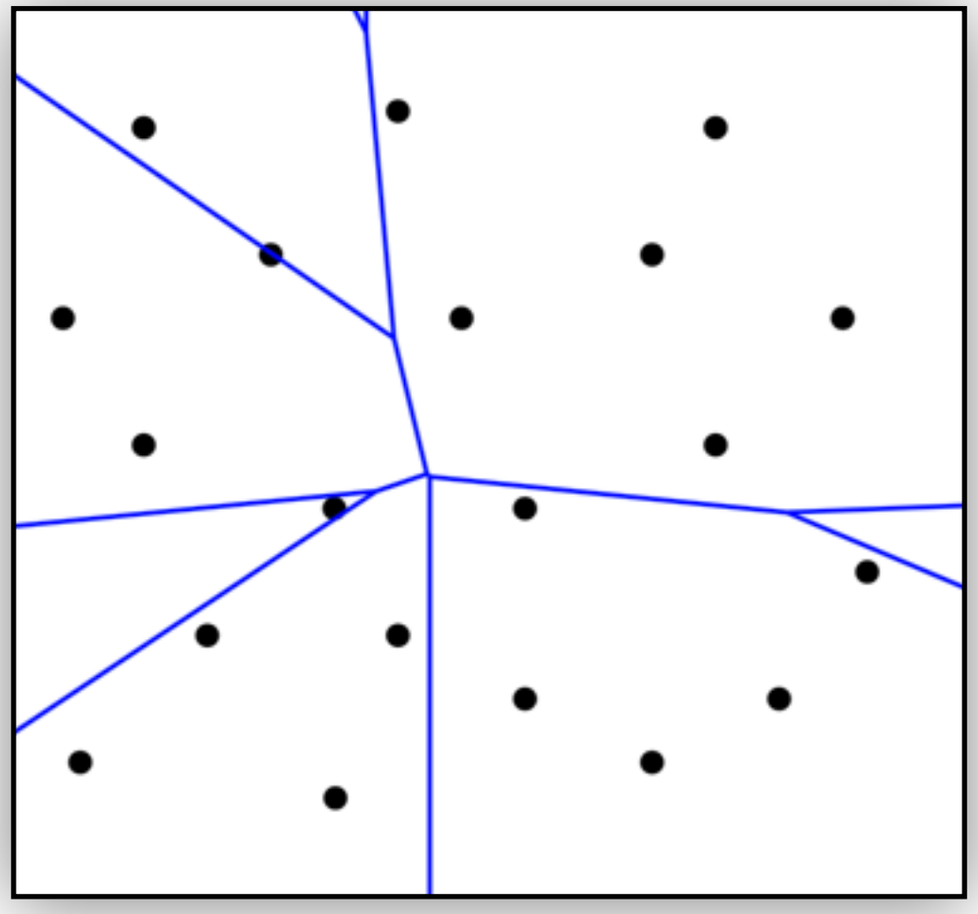
Furthest-point Voronoi diagram:

Furthest-point Voronoi diagram:

- Voronoi region: Set of points with same furthest site $p \in \mathcal{P}$.

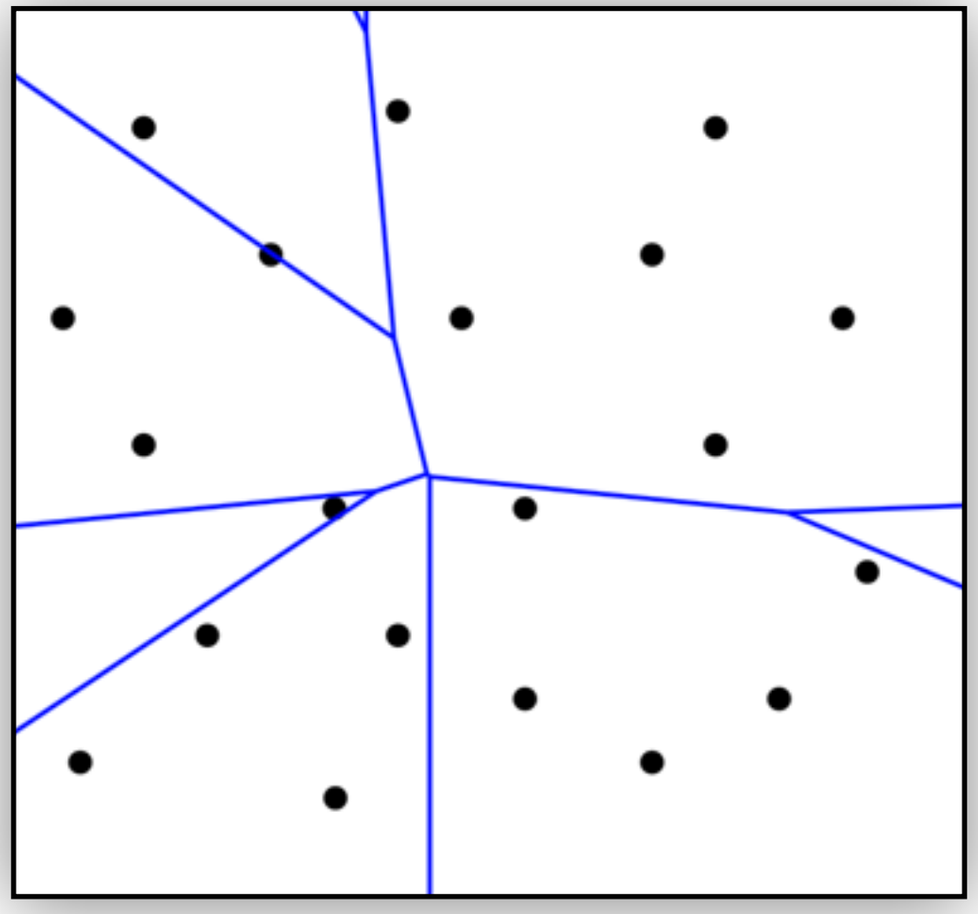
Furthest-point Voronoi diagram:

- Voronoi region: Set of points with same furthest site $p \in \mathcal{P}$.



Furthest-point Voronoi diagram:

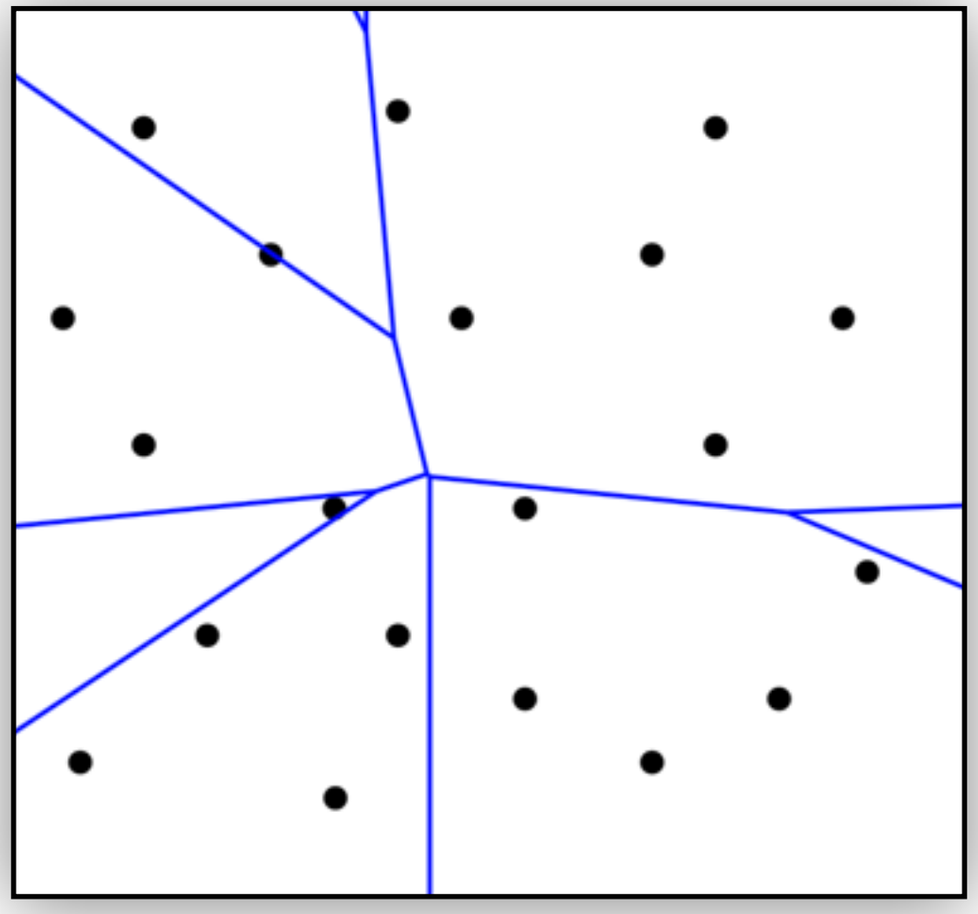
- Voronoi region: Set of points with same furthest site $p \in \mathcal{P}$.

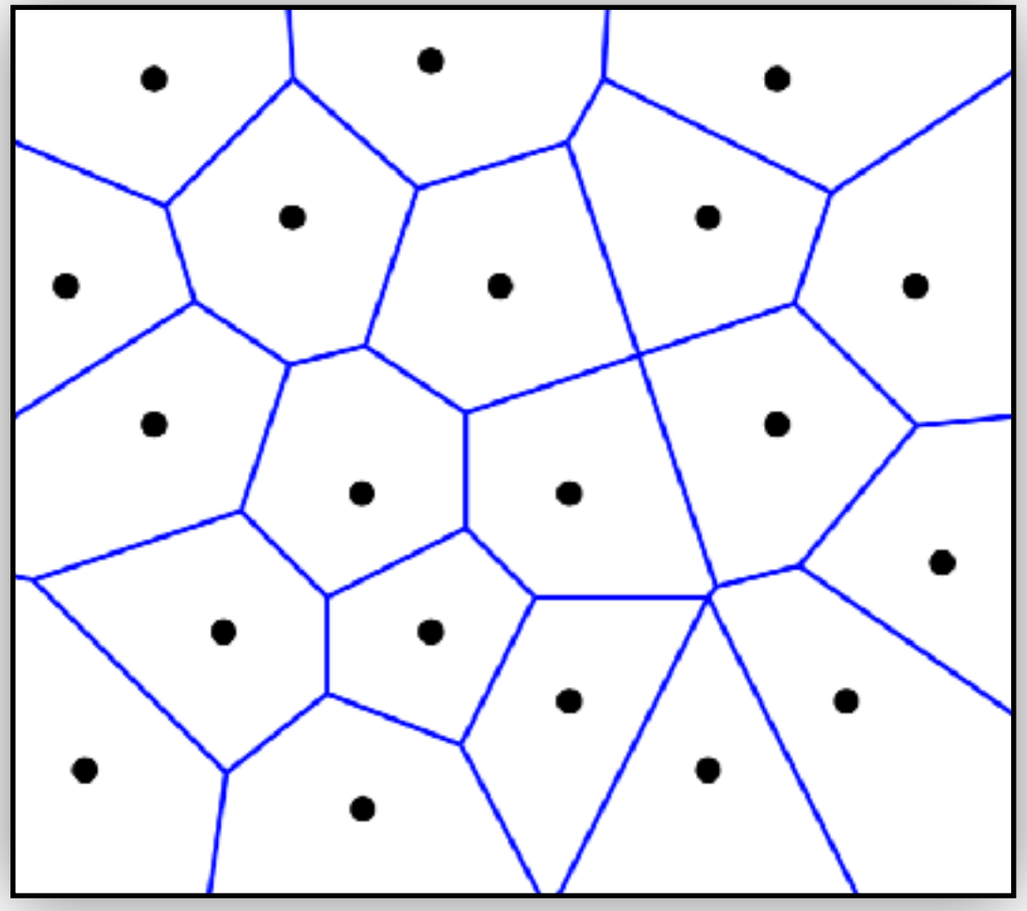


Furthtest-point Voronoi diagram

Furthest-point Voronoi diagram:

- Voronoi region: Set of points with same furthest site $p \in \mathcal{P}$.

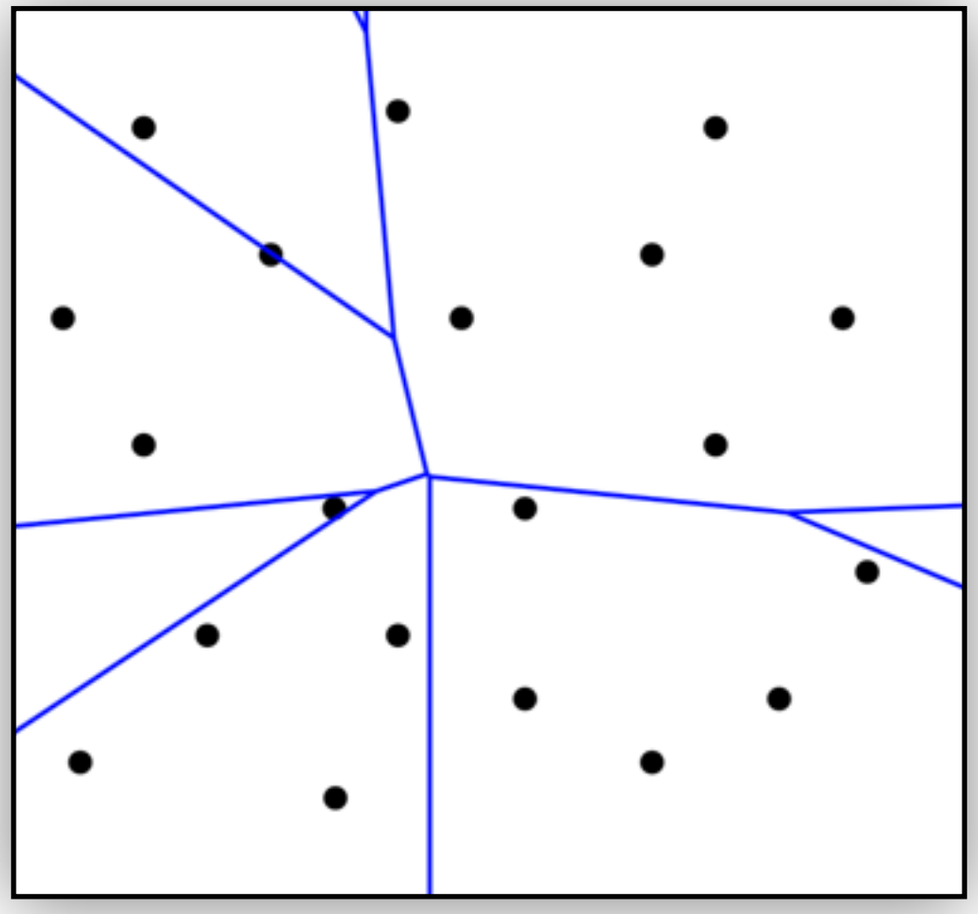




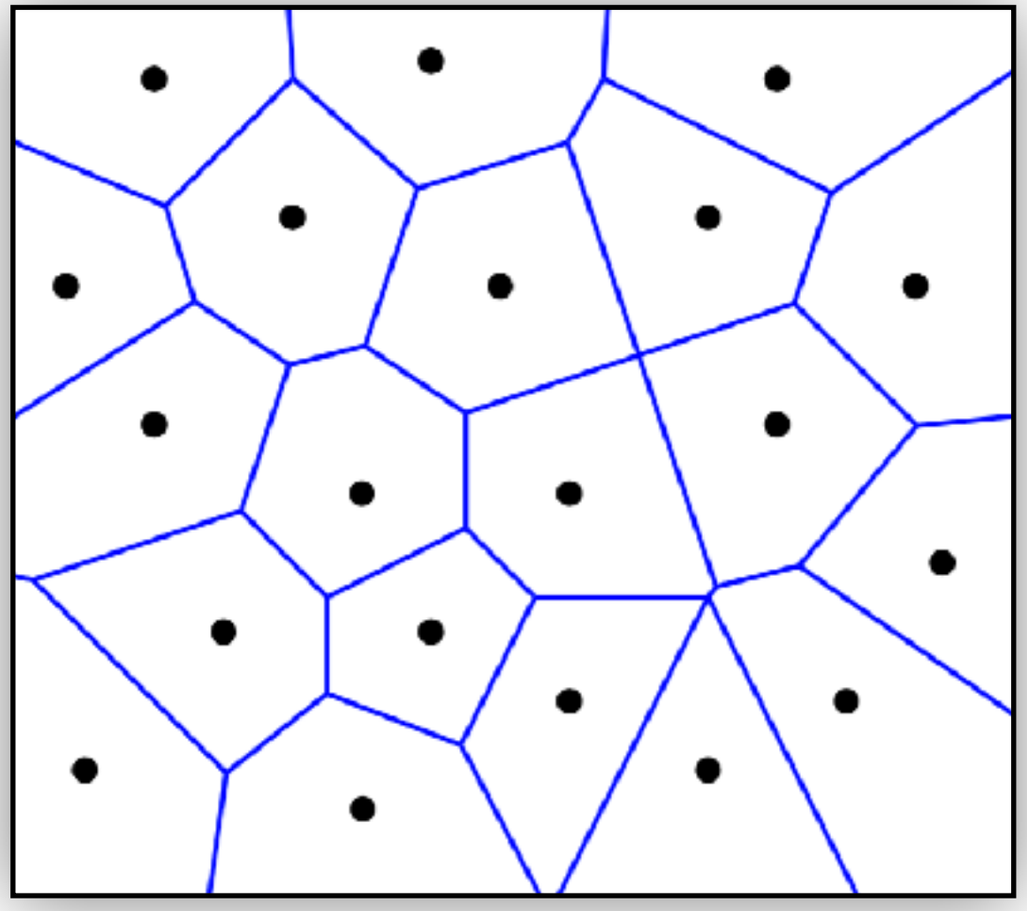
Furthtest-point Voronoi diagram

Furthest-point Voronoi diagram:

- Voronoi region: Set of points with same furthest site $p \in \mathcal{P}$.



Furthtest-point Voronoi diagram

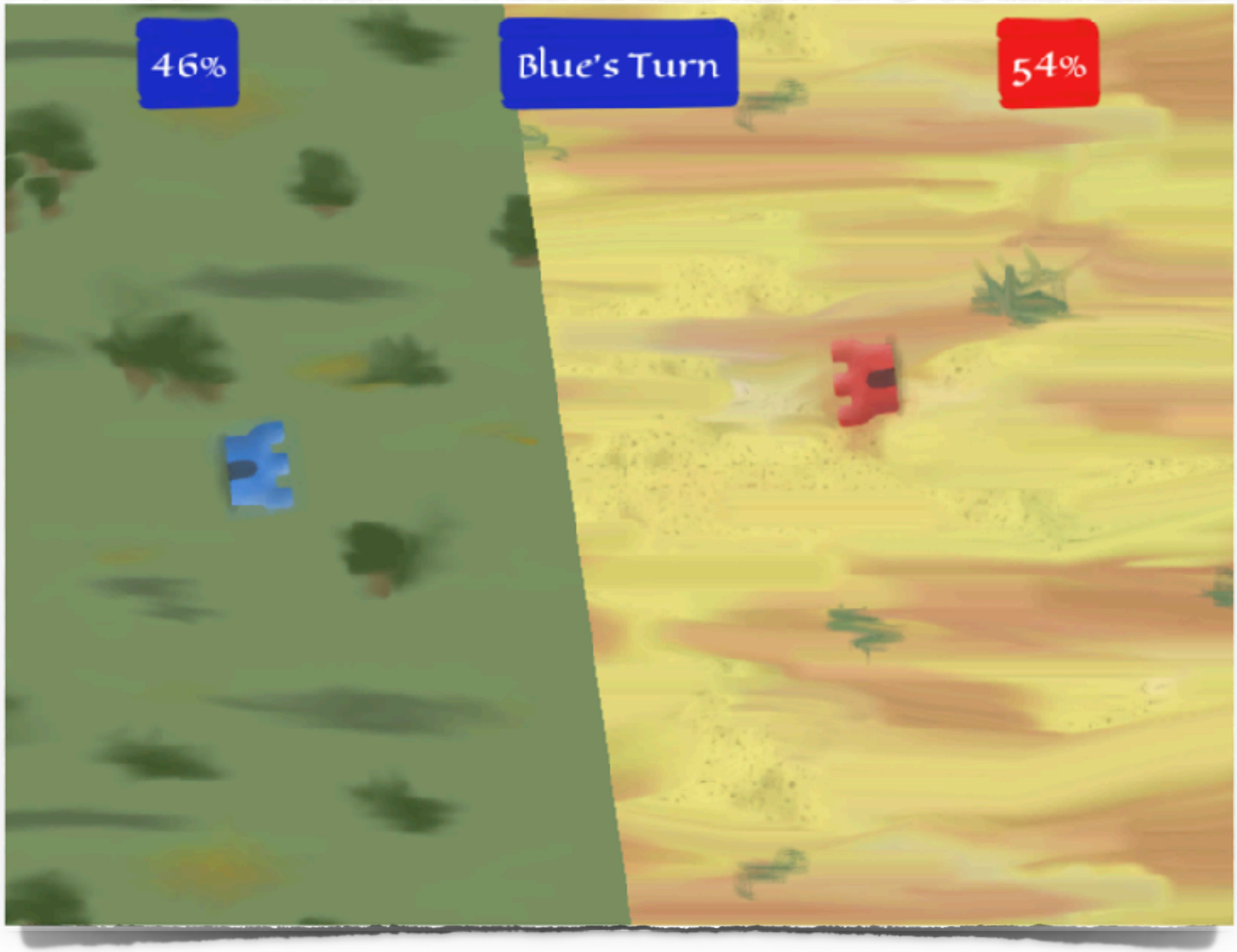


Voronoi diagram

1. Introduction and Motivation
2. Definitions
3. Representing planar partitions
4. Properties
5. Fortune's algorithm
6. Variations
7. The Voronoi game
8. Summary and conclusions

Ruler of the Plane

The Voronoi Game



ELSEVIER

Available online at www.sciencedirect.com
SCIENCE DIRECT®

Computational Geometry 30 (2005) 81–96

Computational Geometry
Theory and Applications

www.elsevier.com/locate/comgeo

The one-round Voronoi game replayed ☆

Sándor P. Fekete [a,*], Henk Meijer [b,1]

[a] *Abteilung für Mathematische Optimierung, Braunschweig University of Technology, D-38106 Braunschweig, Germany*
[b] *School of Computing, Queen's University, Kingston, Ontario K7L 3N6, Canada*

Received 6 June 2003; received in revised form 29 April 2004; accepted 29 May 2004

Available online 7 October 2004

Communicated by R. Klein

Abstract

We consider the one-round Voronoi game, where the first player ("White", called "Wilma") places a set of n points in a rectangular area of aspect ratio $\rho \leqslant 1$, followed by the second player ("Black", called "Barney"), who places the same number of points. Each player wins the fraction of the board closest to one of his points, and the goal is to win more than half of the total area. This problem has been studied by Cheong et al. who showed that for large enough n and $\rho = 1$, Barney has a strategy that guarantees a fraction of $1/2 + \alpha$, for some small fixed α.

We resolve a number of open problems raised by that paper. In particular, we give a precise characterization of the outcome of the game for optimal play: we show that Barney has a winning strategy for $n \geqslant 3$ and $\rho > \sqrt{2}/n$, and for $n = 2$ and $\rho > \sqrt{3}/2$. Wilma wins in all remaining cases, i.e., for $n \geqslant 3$ and $\rho \leqslant \sqrt{2}/n$, for $n = 2$ and $\rho \leqslant \sqrt{3}/2$, and for $n = 1$. We also discuss complexity aspects of the game on more general boards, by proving that for a polygon with holes, it is NP-hard to maximize the area Barney can win against a given set of points by Wilma.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Computational geometry; Voronoi diagram; Voronoi game; Competitive facility location; 2-person games; NP-hardness

☆ A previous extended abstract version appears in [Proceedings of the 8th Workshop on Algorithms and Data Structures, Springer, Berlin, 2003, pp. 150–161] [8].

* Corresponding author.
E-mail addresses: s.fekete@tu-bs.de (S.P. Fekete), henk@cs.queensu.ca (H. Meijer).
URLs: http://www.mathi.tu-bs.de/~fekete (S.P. Fekete), http://www.cs.queensu.ca/~henk/ (H. Meijer).

[1] Partially supported by NSERC. This work was done while visiting TU Braunschweig.

0925-7721/$ – see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.comgeo.2004.05.005

The Voronoi Game

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computational Geometry 30 (2005) 81–96
Computational Geometry
Theory and Applications
www.elsevier.com/locate/comgeo

The one-round Voronoi game replayed ☆

Sándor P. Fekete [a,*], Henk Meijer [b,1]

[a] Abteilung für Mathematische Optimierung, Braunschweig University of Technology, D-38106 Braunschweig, Germany
[b] School of Computing, Queen's University, Kingston, Ontario K7L 3N6, Canada

Received 6 June 2003; received in revised form 29 April 2004; accepted 29 May 2004

Available online 7 October 2004

Communicated by R. Klein

Abstract

We consider the one-round Voronoi game, where the first player ("White", called "Wilma") places a set of n points in a rectangular area of aspect ratio $\rho \leqslant 1$, followed by the second player ("Black", called "Barney"), who places the same number of points. Each player wins the fraction of the board closest to one of his points, and the goal is to win more than half of the total area. This problem has been studied by Cheong et al. who showed that for large enough n and $\rho = 1$, Barney has a strategy that guarantees a fraction of $1/2 + \alpha$, for some small fixed α.

We resolve a number of open problems raised by that paper. In particular, we give a precise characterization of the outcome of the game for optimal play: we show that Barney has a winning strategy for $n \geqslant 3$ and $\rho > \sqrt{2}/n$, and for $n = 2$ and $\rho > \sqrt{3}/2$. Wilma wins in all remaining cases, i.e., for $n \geqslant 3$ and $\rho \leqslant \sqrt{2}/n$, for $n = 2$ and $\rho \leqslant \sqrt{3}/2$, and for $n = 1$. We also discuss complexity aspects of the game on more general boards, by proving that for a polygon with holes, it is NP-hard to maximize the area Barney can win against a given set of points by Wilma.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Computational geometry; Voronoi diagram; Voronoi game; Competitive facility location; 2-person games; NP-hardness

☆ A previous extended abstract version appears in [Proceedings of the 8th Workshop on Algorithms and Data Structures, Springer, Berlin, 2003, pp. 150–161] [8].

* Corresponding author.
E-mail addresses: s.fekete@tu-bs.de (S.P. Fekete), henk@cs.queensu.ca (H. Meijer).
URLs: http://www.math.tu-bs.de/~fekete (S.P. Fekete), http://www.cs.queensu.ca/~henk/ (H. Meijer).

[1] Partially supported by NSERC. This work was done while visiting TU Braunschweig.

0925-7721/$ – see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.comgeo.2004.05.005

Competitive Location Problems: Balanced Facility Location and the One-Round Manhattan Voronoi Game *

Thomas Byrne[1[0000−0003−3548−4086]], Sándor P. Fekete[2[0000−0002−9062−4241]], Jörg Kalcsics[1[0000−0002−5313−3449]], and Linda Kleist[2[0000−0002−3796−916X]]

[1] School of Mathematics, University of Edinburgh, United Kingdom
{tbyrne,joerg.kalcsics}@ed.ac.uk
[2] Department of Computer Science, TU Braunschweig, Germany
{s.fekete,l.kleist}@tu-bs.de

Abstract. We study competitive location problems in a continuous setting, in which facilities have to be placed in a rectangular domain R of normalized dimensions of 1 and $\rho \geq 1$, and distances are measured according to the Manhattan metric. We show that the family of *balanced* configurations (in which the Voronoi cells of individual facilities are equalized with respect to geometric properties) is richer in this metric than for Euclidean distances. Our main result considers the *One-Round Voronoi Game* with Manhattan distances, in which first player White and then player Black each place n points in R; each player scores the area for which one of its facilities is closer than the facilities of the opponent. We give a tight characterization: White has a winning strategy if and only if $\rho \geq n$; for all other cases, we present a winning strategy for Black.

Keywords: Facility location · competitive location · Manhattan distances · Voronoi game · geometric optimization.

1 Introduction

Problems of optimal location are arguably among the most important in a wide range of areas, such as economics, engineering, and biology, as well as in mathematics and computer science. In recent years, they have gained importance through clustering problems in artificial intelligence. In all scenarios, the task is to choose a set of positions from a given domain, such that some optimality criteria for the resulting distances to a set of demand points are satisfied; in a geometric setting, Euclidean or Manhattan distances are natural choices. Another challenge is that facility location problems often happen in a *competitive* setting, in which two or more players contend for the best locations. This change to competitive, multi-player versions can have a serious impact on the algorithmic difficulty of optimization problems: e.g., the classic Travelling Salesman Problem is NP-hard, while the competitive two-player variant is even PSPACE-complete [10].

* A full version can be found at arXiv: 2011.13275 [6].

1. Introduction and Motivation
2. Definitions
3. Representing planar partitions
4. Properties
5. Fortune's algorithm
6. Variations
7. The Voronoi game
8. Summary and conclusions

Summary

Voronoi diagrams:

Summary

Voronoi diagrams:

- Definition for point sets

Summary

Voronoi diagrams:

- Definition for point sets
- Lower bound for runtime: $\Omega(n \log n)$

Summary

Voronoi diagrams:

- Definition for point sets
- Lower bound for runtime: $\Omega(n \log n)$
- Related to several previous problems

Summary

Voronoi diagrams:

- Definition for point sets
- Lower bound for runtime: $\Omega(n \log n)$
- Related to several previous problems

Algorithm for 2D

Summary

Voronoi diagrams:

- Definition for point sets
- Lower bound for runtime: $\Omega(n \log n)$
- Related to several previous problems

Algorithm for 2D

- *Plane sweep* [Fortune, 1987]

Voronoi diagrams:

- Definition for point sets
- Lower bound for runtime: $\Omega(n \log n)$
- Related to several previous problems

Algorithm for 2D

- *Plane sweep* [Fortune, 1987]

Further reading:

Summary

Voronoi diagrams:

- Definition for point sets
- Lower bound for runtime: $\Omega(n \log n)$
- Related to several previous problems

Algorithm for 2D

- *Plane sweep* [Fortune, 1987]

Further reading:

- [Aurenhammer, 1991] (Survey article)

Summary

Voronoi diagrams:

- Definition for point sets
- Lower bound for runtime: $\Omega(n \log n)$
- Related to several previous problems

Algorithm for 2D

- *Plane sweep* [Fortune, 1987]

Further reading:

- [Aurenhammer, 1991] (Survey article)
- [Fortune, 2004] (Survey article)

Summary

Voronoi diagrams:

- Definition for point sets
- Lower bound for runtime: $\Omega(n \log n)$
- Related to several previous problems

Algorithm for 2D

- *Plane sweep* [Fortune, 1987]

Further reading:

- [Aurenhammer, 1991] (Survey article)
- [Fortune, 2004] (Survey article)
- [Akabe *et al.,* 2000] (Survey book).

Thank you for today!