

# Farthest-Point Voronoi Diagrams

---

## Outline:

I. Farthest site

II. Voronoi cell

III. Structure of FPVD

IV. Construction

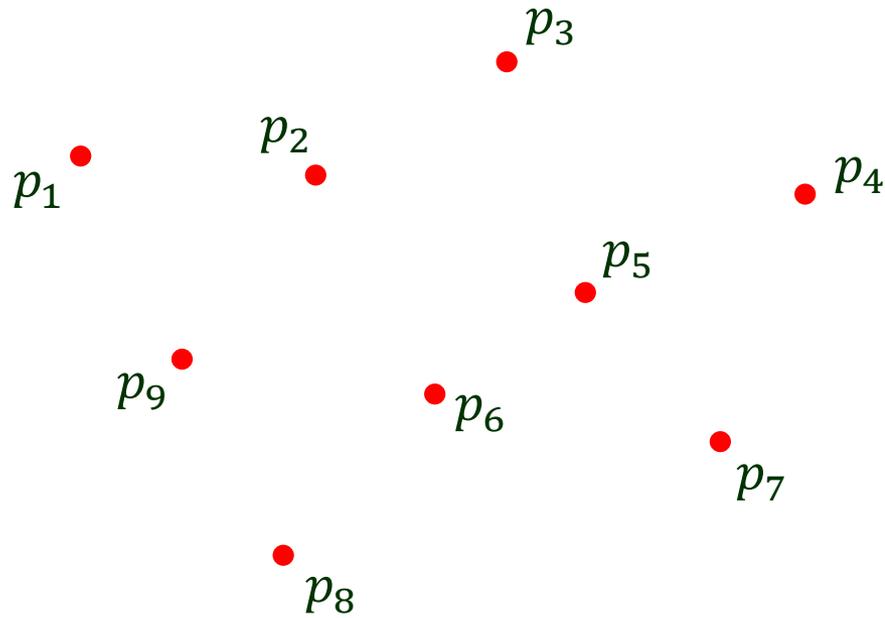
V. Smallest-Width Annulus

# I. Farthest Point

---

$$P = \{p_1, p_2, \dots, p_n\}$$

Sites



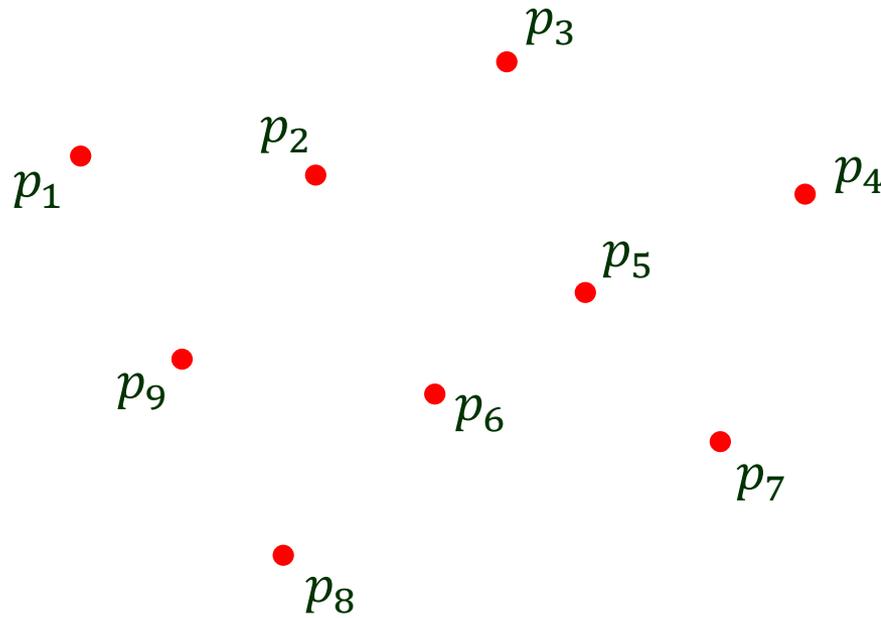
# I. Farthest Point

---

$$P = \underbrace{\{p_1, p_2, \dots, p_n\}}_{\text{Sites}}$$

Given a point  $q$ ,  $p_i \in P$  is its *farthest point* if for all  $1 \leq j \leq n$

$$|q - p_i| \geq |q - p_j|$$



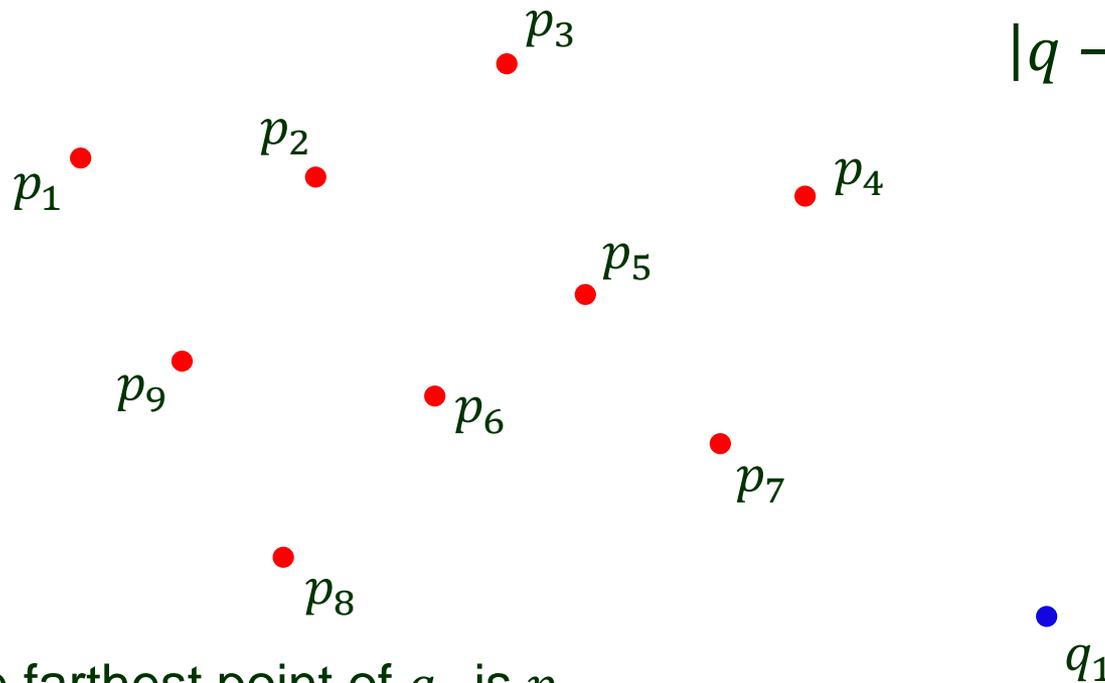
# I. Farthest Point

---

$$P = \underbrace{\{p_1, p_2, \dots, p_n\}}_{\text{Sites}}$$

Given a point  $q$ ,  $p_i \in P$  is its *farthest point* if for all  $1 \leq j \leq n$

$$|q - p_i| \geq |q - p_j|$$



The farthest point of  $q_1$  is  $p_1$ .

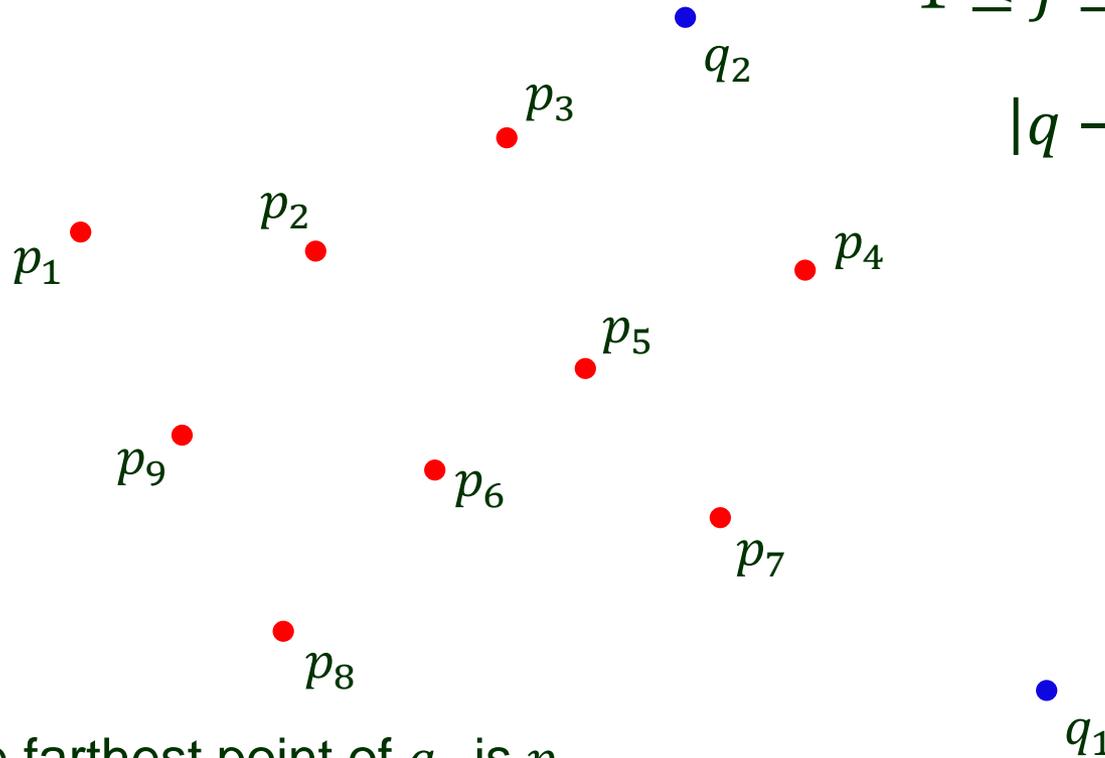
# I. Farthest Point

---

$$P = \underbrace{\{p_1, p_2, \dots, p_n\}}_{\text{Sites}}$$

Given a point  $q$ ,  $p_i \in P$  is its *farthest point* if for all  $1 \leq j \leq n$

$$|q - p_i| \geq |q - p_j|$$



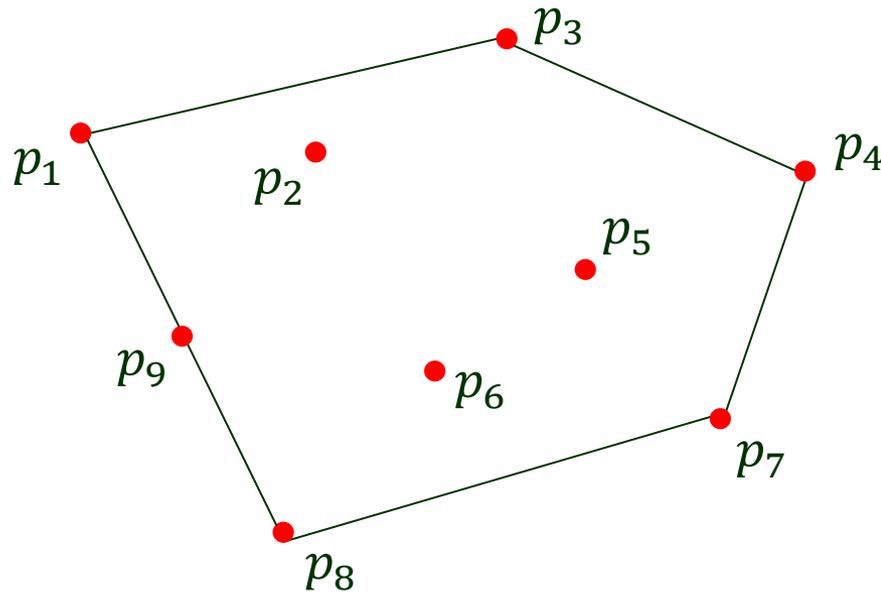
The farthest point of  $q_1$  is  $p_1$ .

The farthest point of  $q_2$  is  $p_8$ .

# Not Every Site Can be the Farthest

---

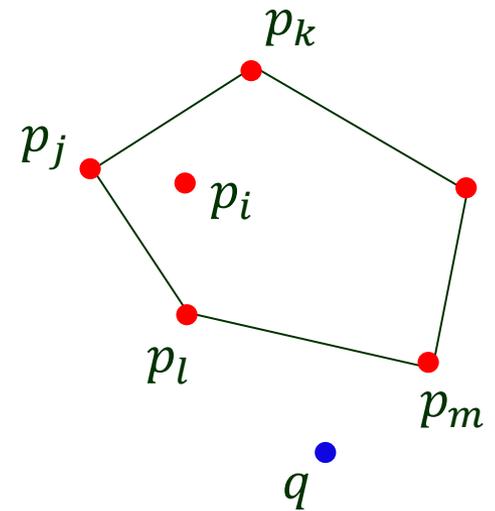
**Claim** A point  $p_i \in P$  is the farthest site of some point  $q$  in the plane if and only if  $p_i$  is a vertex of the convex hull  $\text{CH}(P)$  of  $P$ .



# Proof of Necessity

**Claim** A point  $p_i \in P$  is the farthest site of some point  $q$  in the plane if and only if  $p_i$  is a vertex of the convex hull  $\text{CH}(P)$  of  $P$ .

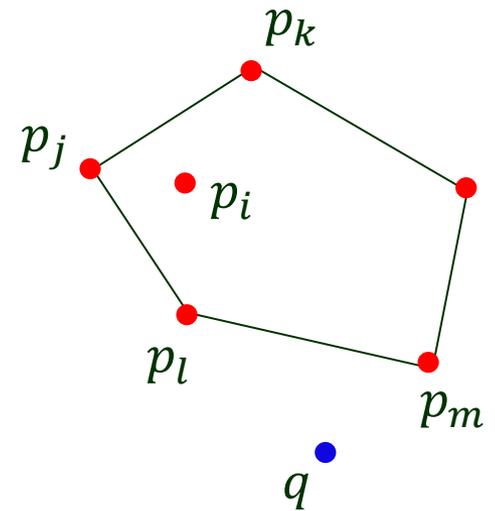
( $\Rightarrow$ ) Suppose there exists some  $q$  such that  $p_i$  is its farthest site.



# Proof of Necessity

**Claim** A point  $p_i \in P$  is the farthest site of some point  $q$  in the plane if and only if  $p_i$  is a vertex of the convex hull  $\text{CH}(P)$  of  $P$ .

( $\Rightarrow$ ) Suppose there exists some  $q$  such that  $p_i$  is its farthest site. Assume that  $p_i$  is not a vertex of  $\text{CH}(P)$ .



# Proof of Necessity

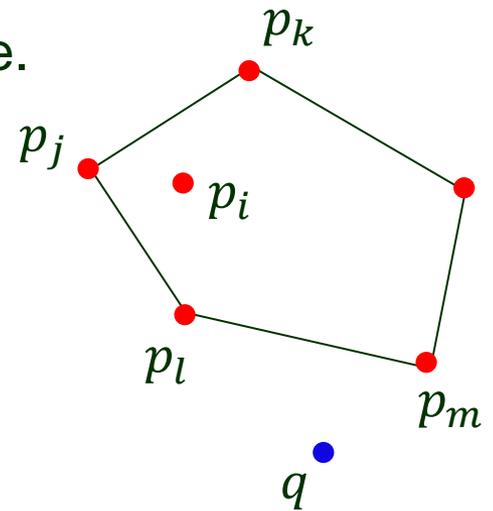
**Claim** A point  $p_i \in P$  is the farthest site of some point  $q$  in the plane if and only if  $p_i$  is a vertex of the convex hull  $\text{CH}(P)$  of  $P$ .

( $\Rightarrow$ ) Suppose there exists some  $q$  such that  $p_i$  is its farthest site.

Assume that  $p_i$  is not a vertex of  $\text{CH}(P)$ .



Then  $p_i$  is either in the interior  $\text{CH}(P)$  or on one of its edge.



# Proof of Necessity

**Claim** A point  $p_i \in P$  is the farthest site of some point  $q$  in the plane if and only if  $p_i$  is a vertex of the convex hull  $\text{CH}(P)$  of  $P$ .

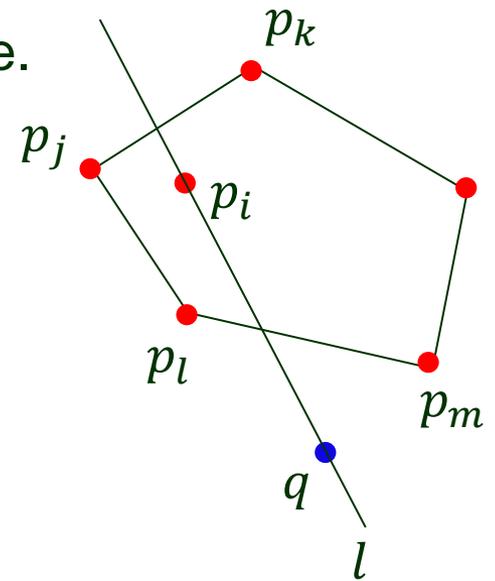
( $\Rightarrow$ ) Suppose there exists some  $q$  such that  $p_i$  is its farthest site.

Assume that  $p_i$  is not a vertex of  $\text{CH}(P)$ .



Then  $p_i$  is either in the interior  $\text{CH}(P)$  or on one of its edge.

Consider the line  $l$  through  $p_i$  and  $q$  (clearly  $p_i \neq q$ ).



# Proof of Necessity

**Claim** A point  $p_i \in P$  is the farthest site of some point  $q$  in the plane if and only if  $p_i$  is a vertex of the convex hull  $\text{CH}(P)$  of  $P$ .

( $\Rightarrow$ ) Suppose there exists some  $q$  such that  $p_i$  is its farthest site.

Assume that  $p_i$  is not a vertex of  $\text{CH}(P)$ .

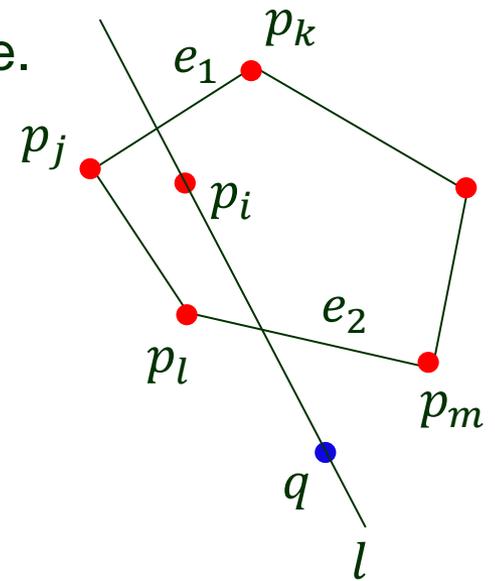


Then  $p_i$  is either in the interior  $\text{CH}(P)$  or on one of its edge.

Consider the line  $l$  through  $p_i$  and  $q$  (clearly  $p_i \neq q$ ).



$l$  intersects  $\text{CH}(P)$  with two of its edges  $e_1$  and  $e_2$ .



# Proof of Necessity

**Claim** A point  $p_i \in P$  is the farthest site of some point  $q$  in the plane if and only if  $p_i$  is a vertex of the convex hull  $\text{CH}(P)$  of  $P$ .

( $\Rightarrow$ ) Suppose there exists some  $q$  such that  $p_i$  is its farthest site.

Assume that  $p_i$  is not a vertex of  $\text{CH}(P)$ .



Then  $p_i$  is either in the interior  $\text{CH}(P)$  or on one of its edge.

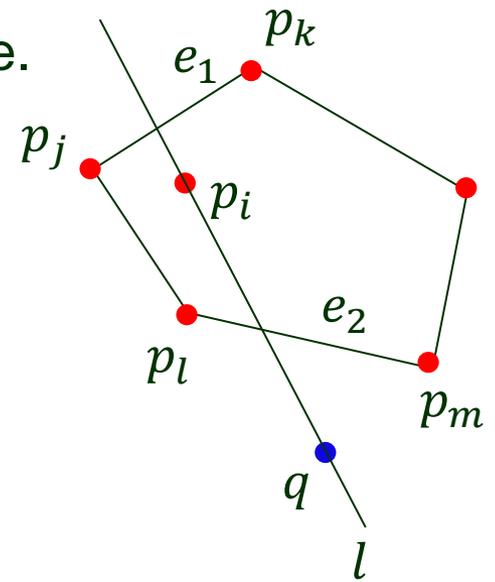
Consider the line  $l$  through  $p_i$  and  $q$  (clearly  $p_i \neq q$ ).



$l$  intersects  $\text{CH}(P)$  with two of its edges  $e_1$  and  $e_2$ .



One of the four endpoints of  $e_1$  and  $e_2$  must be farther from  $q$  than  $p_i$ .



# Proof of Necessity

**Claim** A point  $p_i \in P$  is the farthest site of some point  $q$  in the plane if and only if  $p_i$  is a vertex of the convex hull  $\text{CH}(P)$  of  $P$ .

( $\Rightarrow$ ) Suppose there exists some  $q$  such that  $p_i$  is its farthest site. Assume that  $p_i$  is not a vertex of  $\text{CH}(P)$ .



Then  $p_i$  is either in the interior  $\text{CH}(P)$  or on one of its edge.

Consider the line  $l$  through  $p_i$  and  $q$  (clearly  $p_i \neq q$ ).

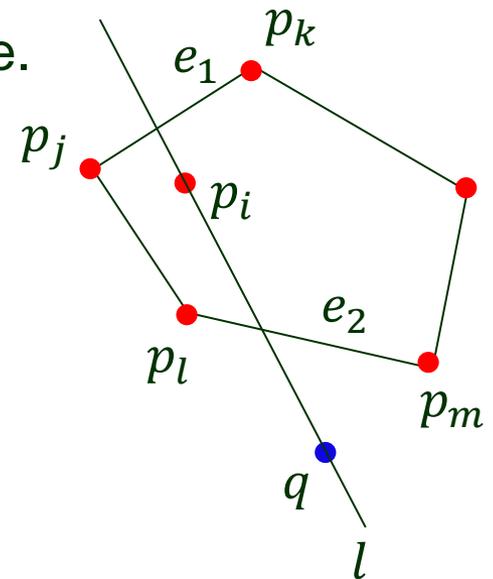


$l$  intersects  $\text{CH}(P)$  with two of its edges  $e_1$  and  $e_2$ .



One of the four endpoints of  $e_1$  and  $e_2$  must be farther from  $q$  than  $p_i$ .

Contradiction.

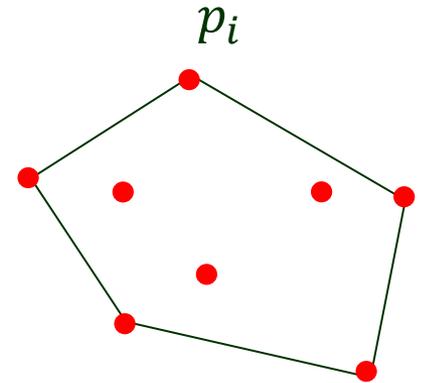


# Proof of Sufficiency

---

**Claim** A point  $p_i \in P$  is the farthest site of some point  $q$  in the plane if and only if  $p_i$  is a vertex of the convex hull  $\text{CH}(P)$  of  $P$ .

( $\Leftarrow$ ) Suppose  $p_i$  is a vertex of  $\text{CH}(P)$ .



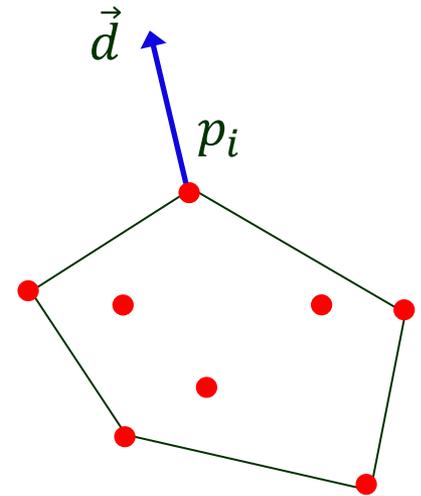
# Proof of Sufficiency

**Claim** A point  $p_i \in P$  is the farthest site of some point  $q$  in the plane if and only if  $p_i$  is a vertex of the convex hull  $\text{CH}(P)$  of  $P$ .

( $\Leftarrow$ ) Suppose  $p_i$  is a vertex of  $\text{CH}(P)$ .



$p_i$  must be extreme in some direction  $\vec{d}$ .



# Proof of Sufficiency

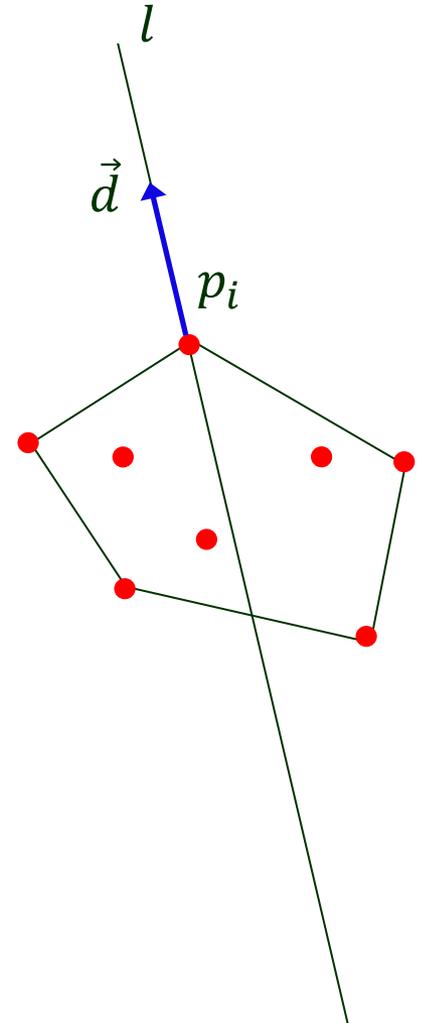
**Claim** A point  $p_i \in P$  is the farthest site of some point  $q$  in the plane if and only if  $p_i$  is a vertex of the convex hull  $\text{CH}(P)$  of  $P$ .

( $\Leftarrow$ ) Suppose  $p_i$  is a vertex of  $\text{CH}(P)$ .



$p_i$  must be extreme in some direction  $\vec{d}$ .

Let  $l$  be the line through  $p_i$  in  $\vec{d}$ .



# Proof of Sufficiency

**Claim** A point  $p_i \in P$  is the farthest site of some point  $q$  in the plane if and only if  $p_i$  is a vertex of the convex hull  $\text{CH}(P)$  of  $P$ .

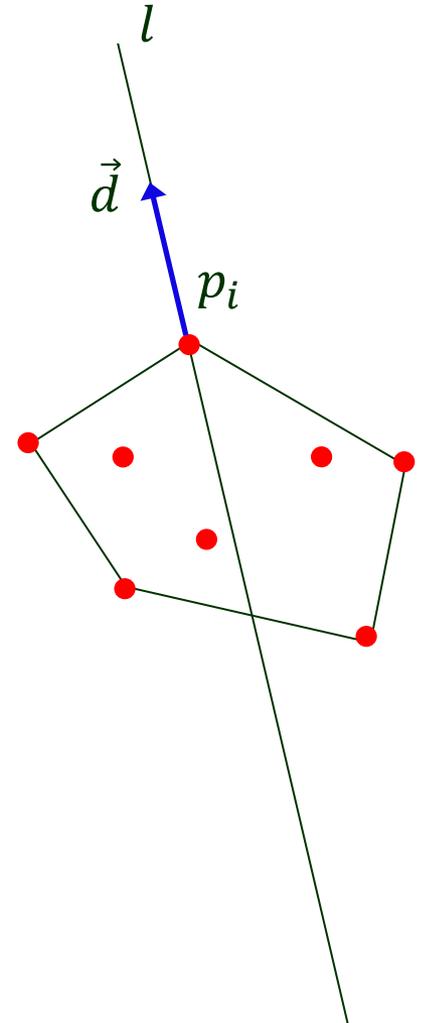
( $\Leftarrow$ ) Suppose  $p_i$  is a vertex of  $\text{CH}(P)$ .



$p_i$  must be extreme in some direction  $\vec{d}$ .

Let  $l$  be the line through  $p_i$  in  $\vec{d}$ .

- Start at  $p_i$ .
- Move on  $l$  in the direction  $-\vec{d}$ .



# Proof of Sufficiency

**Claim** A point  $p_i \in P$  is the farthest site of some point  $q$  in the plane if and only if  $p_i$  is a vertex of the convex hull  $\text{CH}(P)$  of  $P$ .

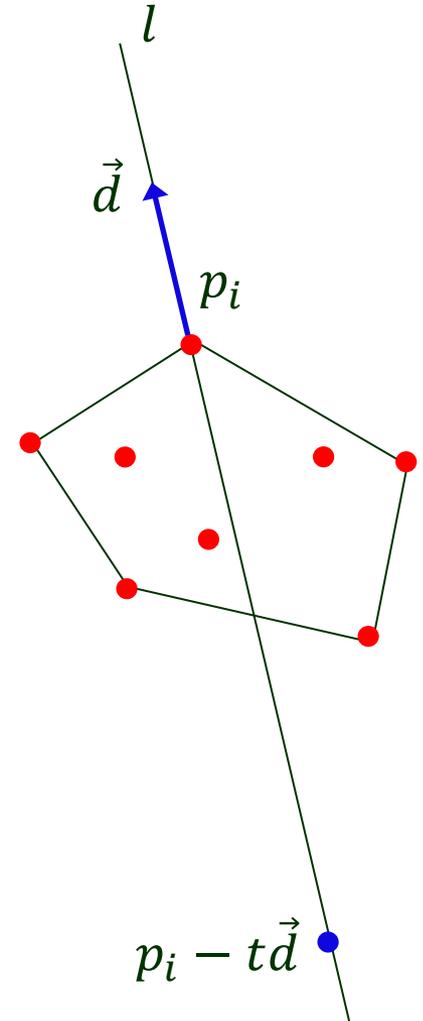
( $\Leftarrow$ ) Suppose  $p_i$  is a vertex of  $\text{CH}(P)$ .



$p_i$  must be extreme in some direction  $\vec{d}$ .

Let  $l$  be the line through  $p_i$  in  $\vec{d}$ .

- Start at  $p_i$ .
- Move on  $l$  in the direction  $-\vec{d}$ .



# Proof of Sufficiency

**Claim** A point  $p_i \in P$  is the farthest site of some point  $q$  in the plane if and only if  $p_i$  is a vertex of the convex hull  $\text{CH}(P)$  of  $P$ .

( $\Leftarrow$ ) Suppose  $p_i$  is a vertex of  $\text{CH}(P)$ .



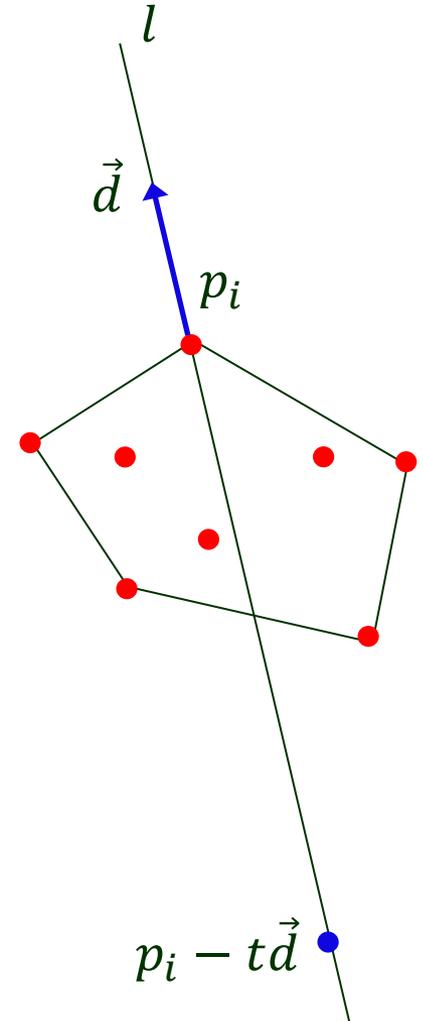
$p_i$  must be extreme in some direction  $\vec{d}$ .

Let  $l$  be the line through  $p_i$  in  $\vec{d}$ .

- Start at  $p_i$ .
- Move on  $l$  in the direction  $-\vec{d}$ .

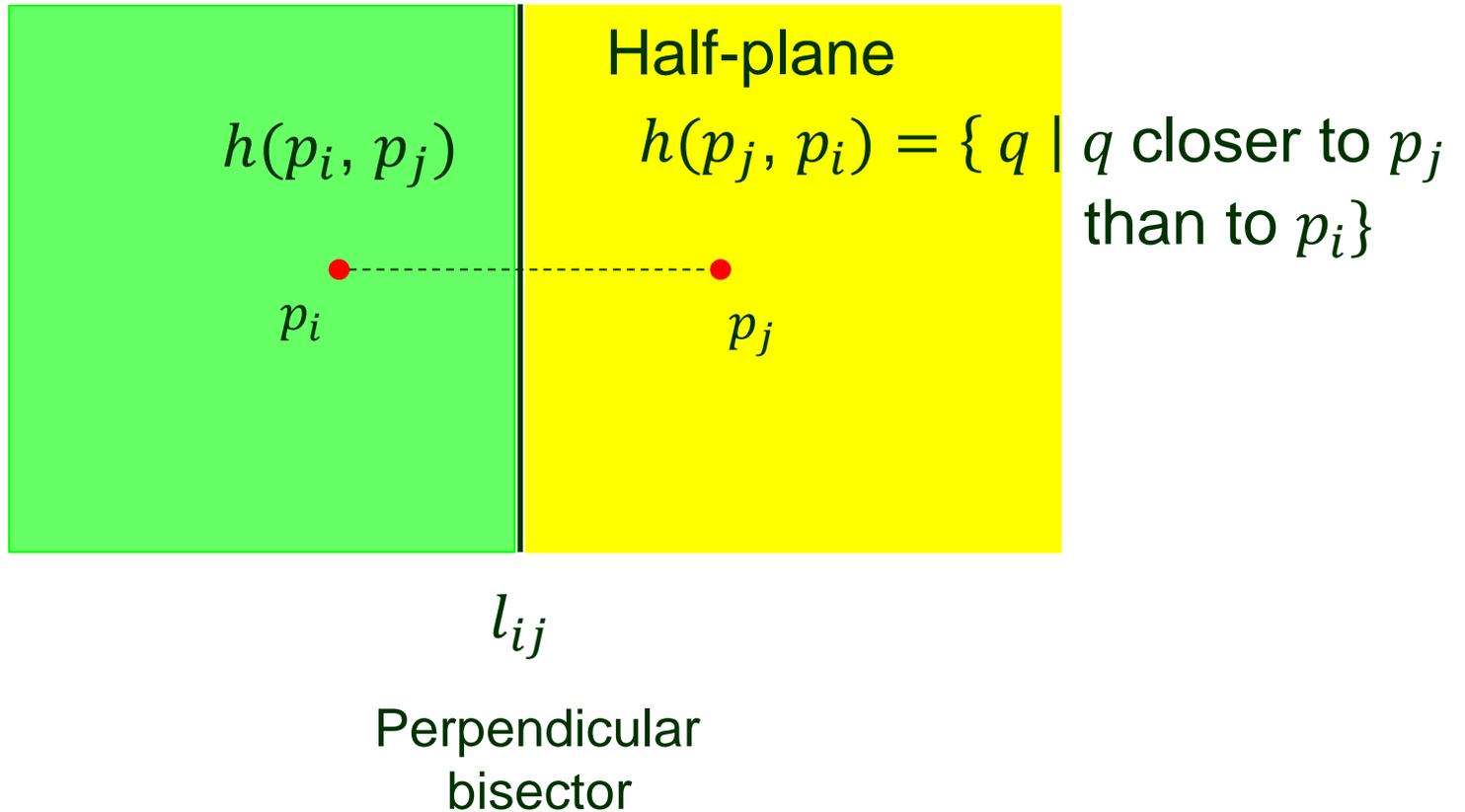


The point  $p_i - t\vec{d}$ , for large enough  $t$ , is farther from  $p_i$  than any other site.



# II. Two Sites

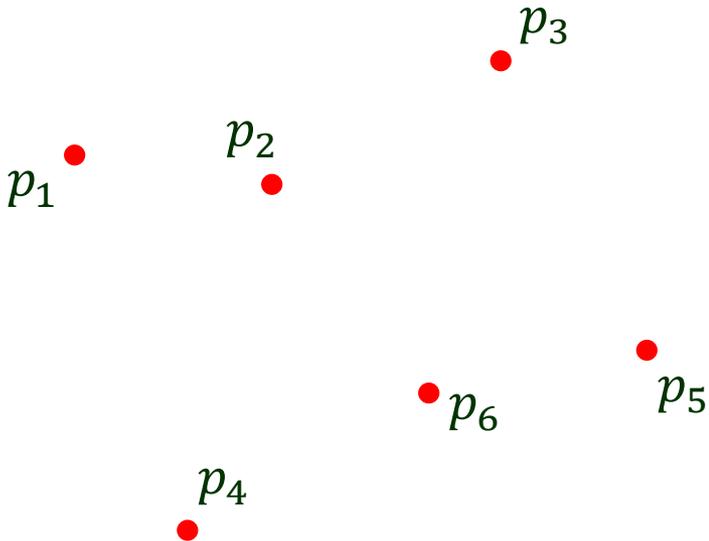
---



# Voronoi Cell

---

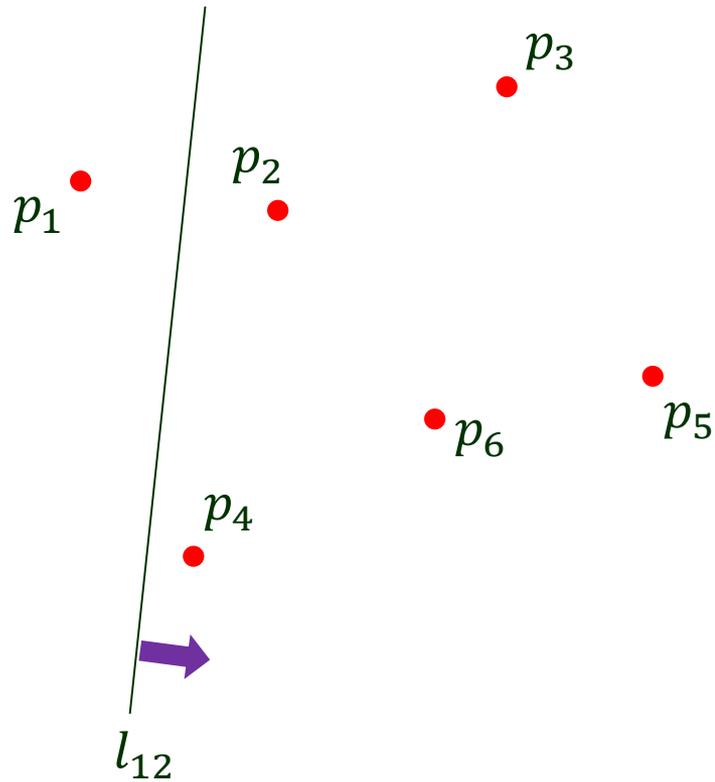
$$V'(p_i) = \bigcap_{\substack{1 \leq j \leq n \\ j \neq i}} h(p_j, p_i)$$



# Voronoi Cell

---

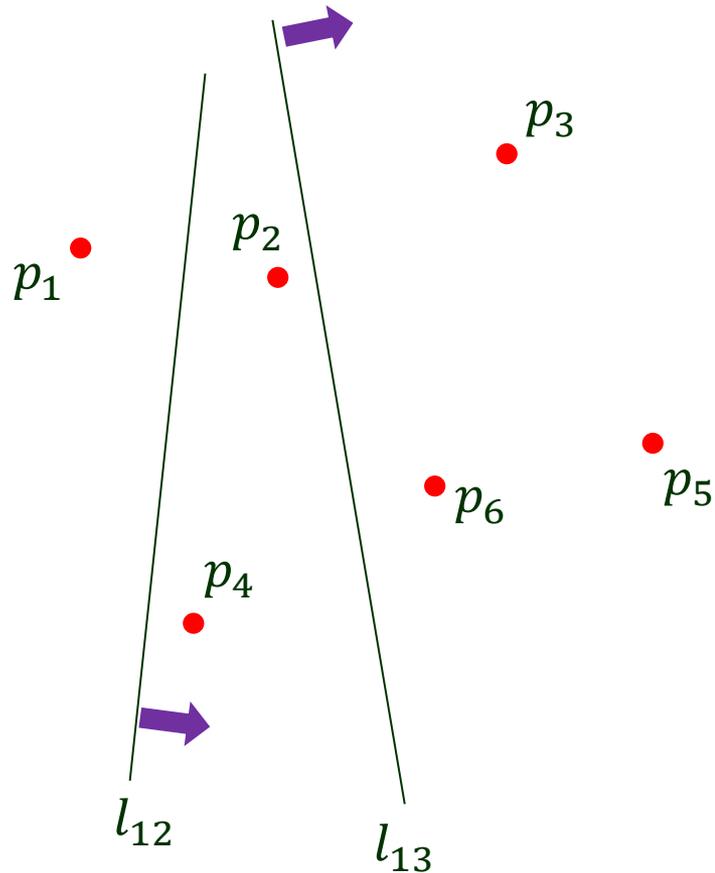
$$V'(p_i) = \bigcap_{\substack{1 \leq j \leq n \\ j \neq i}} h(p_j, p_i)$$



# Voronoi Cell

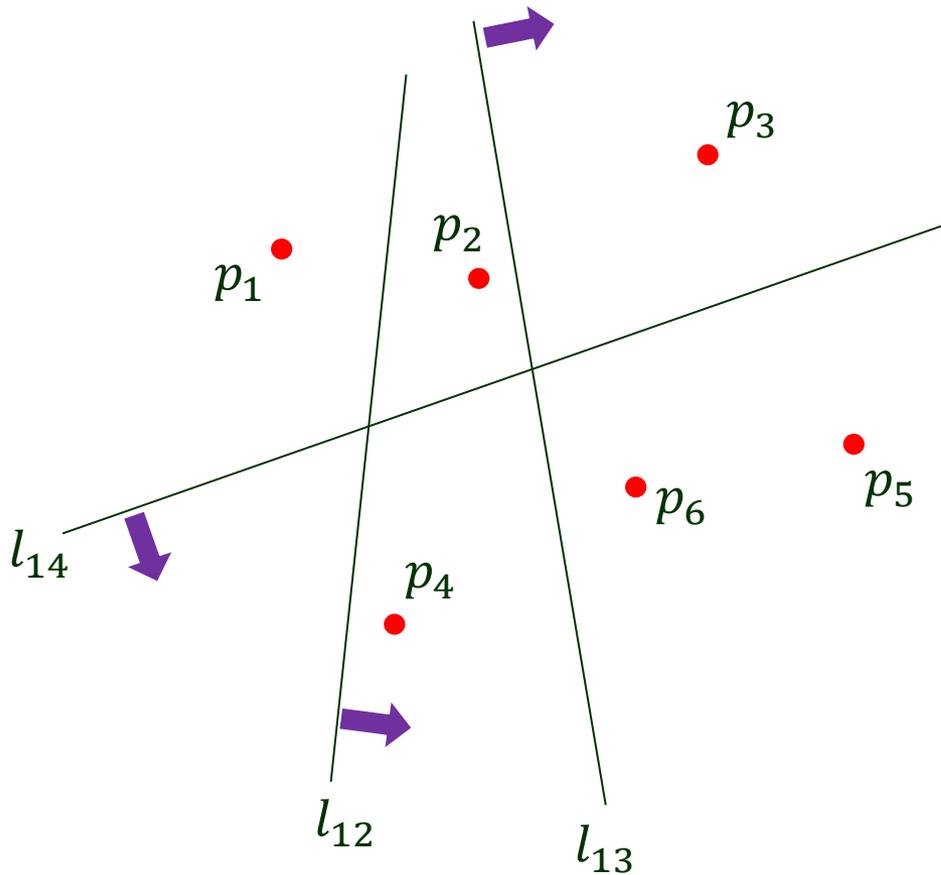
---

$$V'(p_i) = \bigcap_{\substack{1 \leq j \leq n \\ j \neq i}} h(p_j, p_i)$$



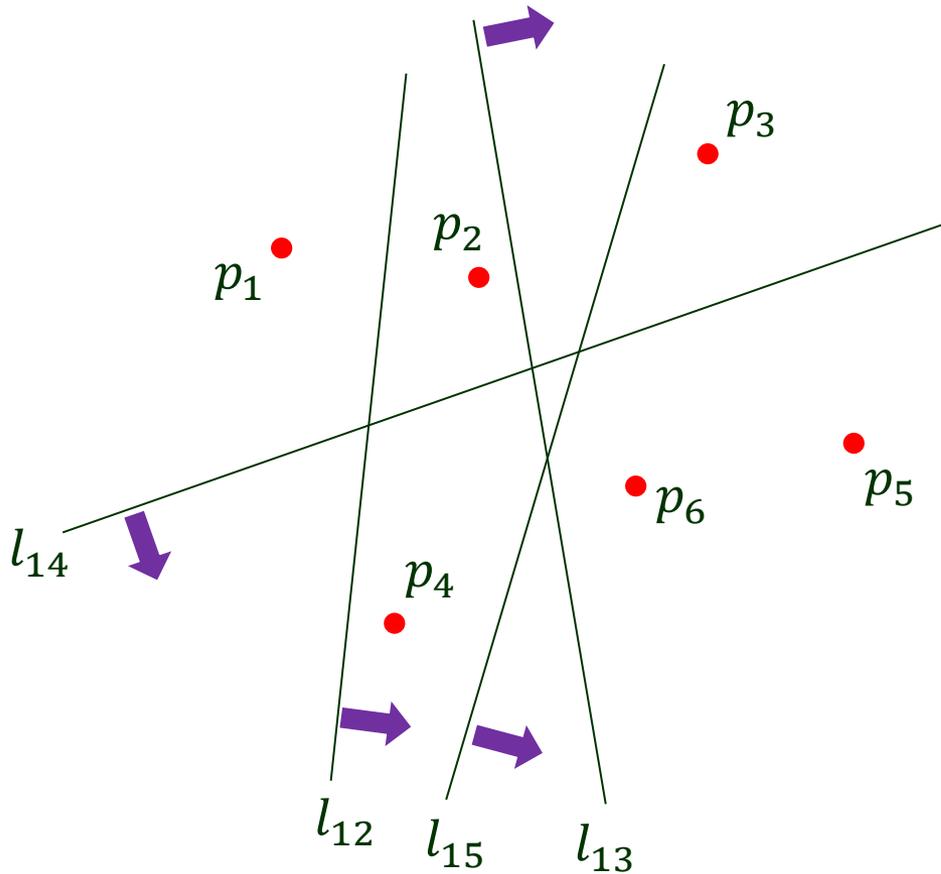
# Voronoi Cell

$$V'(p_i) = \bigcap_{\substack{1 \leq j \leq n \\ j \neq i}} h(p_j, p_i)$$



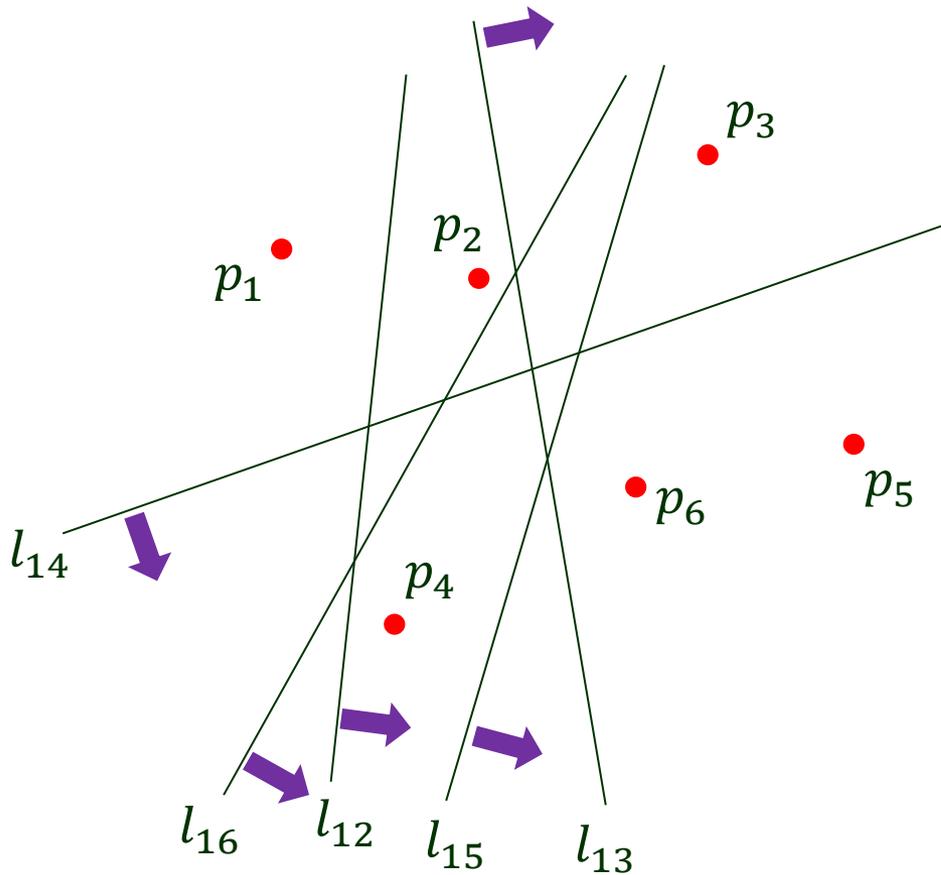
# Voronoi Cell

$$V'(p_i) = \bigcap_{\substack{1 \leq j \leq n \\ j \neq i}} h(p_j, p_i)$$



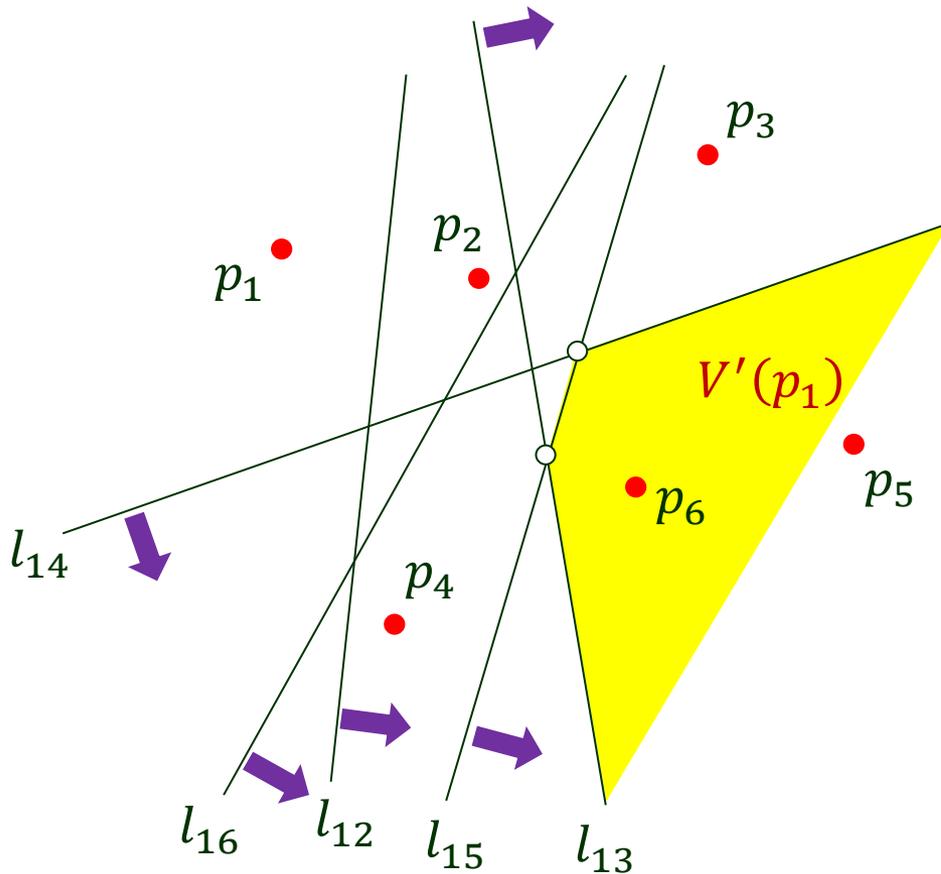
# Voronoi Cell

$$V'(p_i) = \bigcap_{\substack{1 \leq j \leq n \\ j \neq i}} h(p_j, p_i)$$



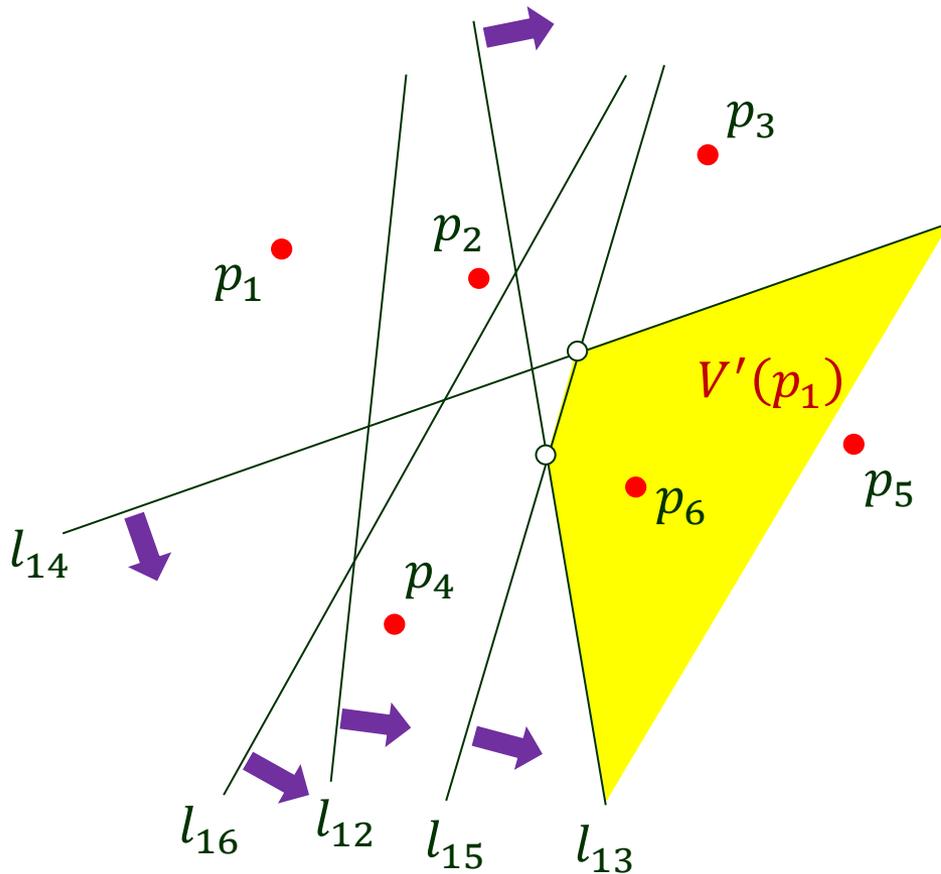
# Voronoi Cell

$$V'(p_i) = \bigcap_{\substack{1 \leq j \leq n \\ j \neq i}} h(p_j, p_i)$$



# Voronoi Cell

$$V'(p_i) = \bigcap_{\substack{1 \leq j \leq n \\ j \neq i}} h(p_j, p_i)$$

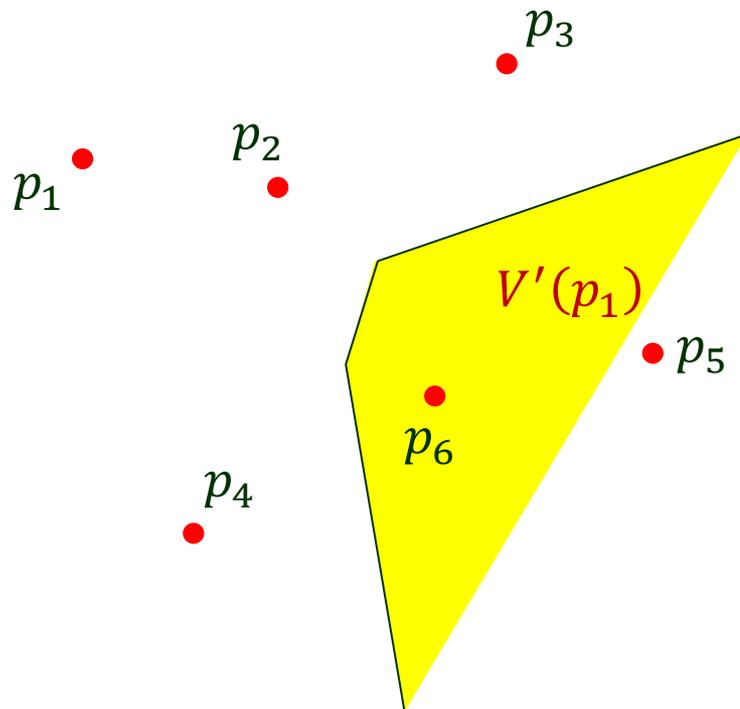


- ◆ Open convex region
- ◆  $\leq n - 1$  vertices
- ◆  $\leq n - 1$  edges

# Unboundedness

---

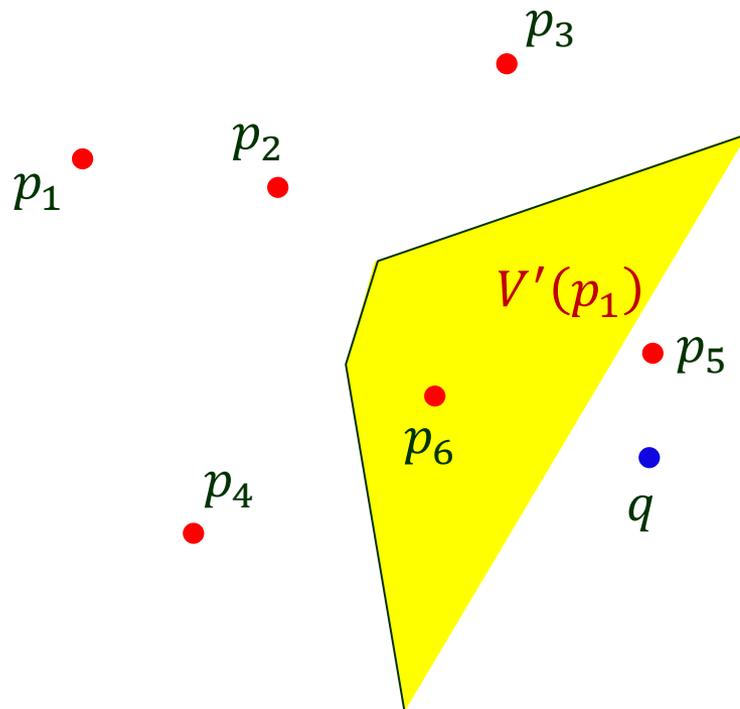
The cell contains a ray  $r$  collinear with  $p_i$ .



# Unboundedness

---

The cell contains a ray  $r$  collinear with  $p_i$ .

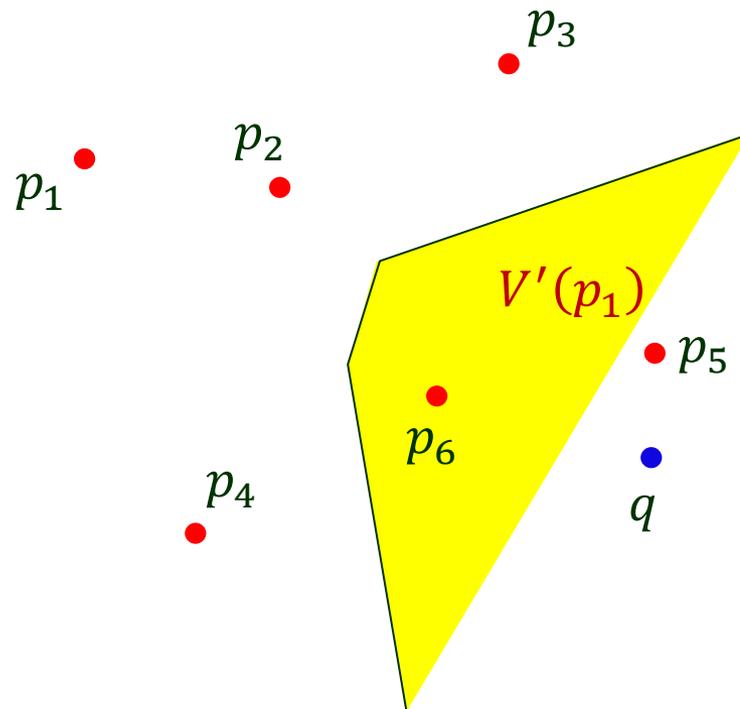


# Unboundedness

---

The cell contains a ray  $r$  collinear with  $p_i$ .

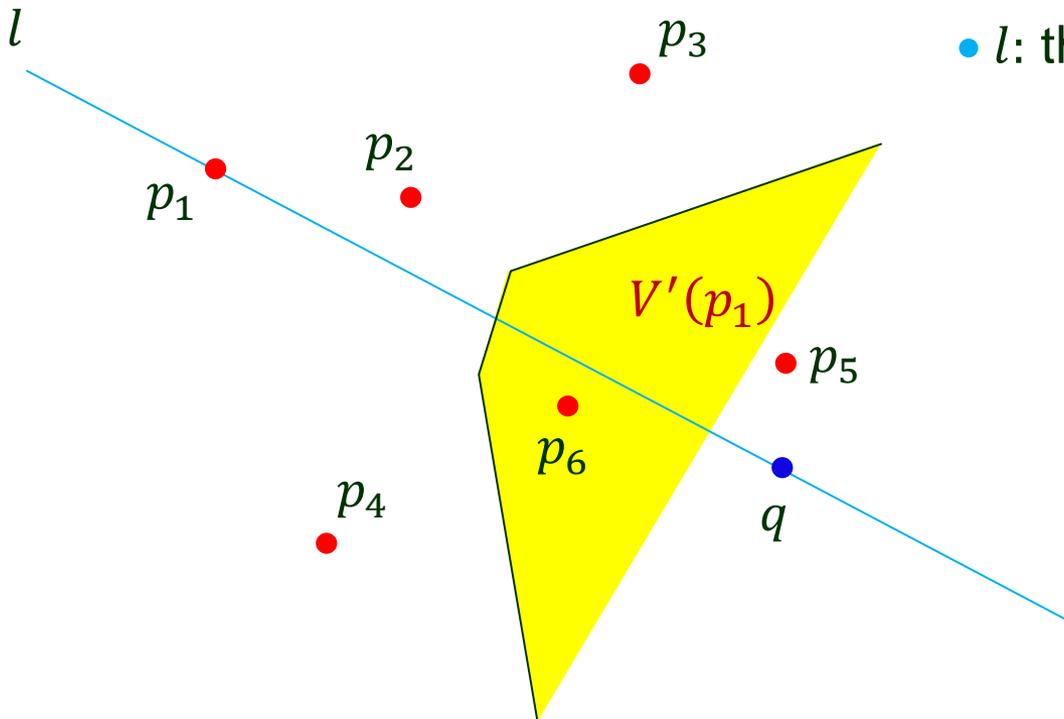
- $p_i$ : farthest point from  $q$ .



# Unboundedness

---

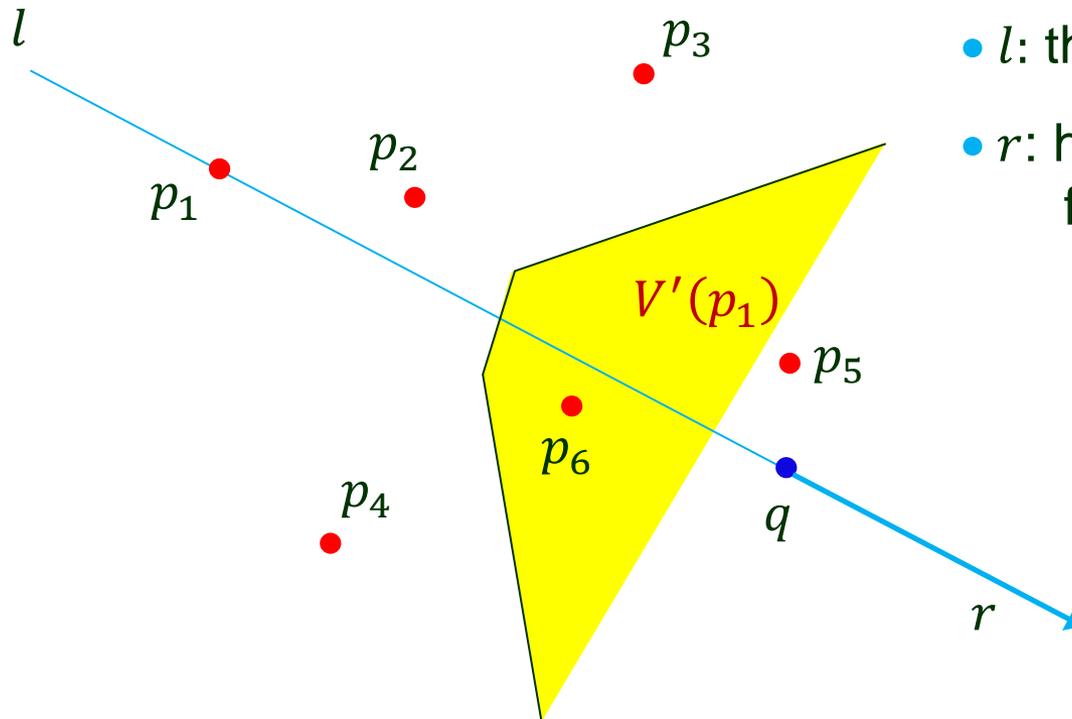
The cell contains a ray  $r$  collinear with  $p_i$ .



- $p_i$ : farthest point from  $q$ .
- $l$ : the line through  $p_i$  and  $q$ .

# Unboundedness

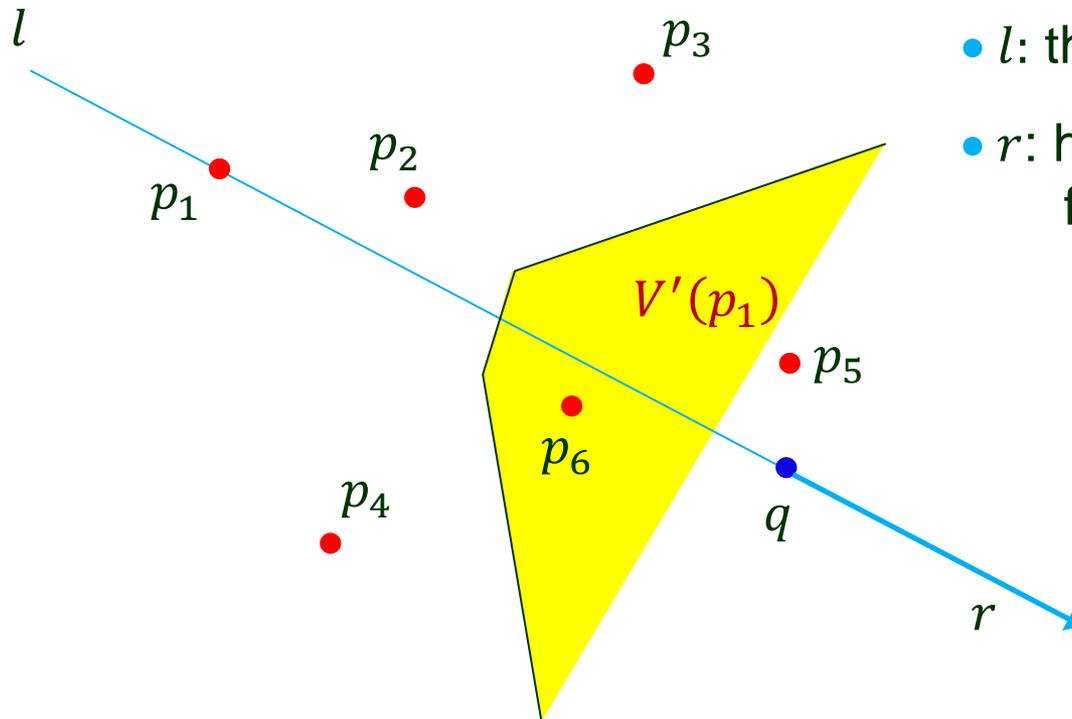
The cell contains a ray  $r$  collinear with  $p_i$ .



- $p_i$ : farthest point from  $q$ .
- $l$ : the line through  $p_i$  and  $q$ .
- $r$ : half-line starting at  $q$  and away from  $p_i$ .

# Unboundedness

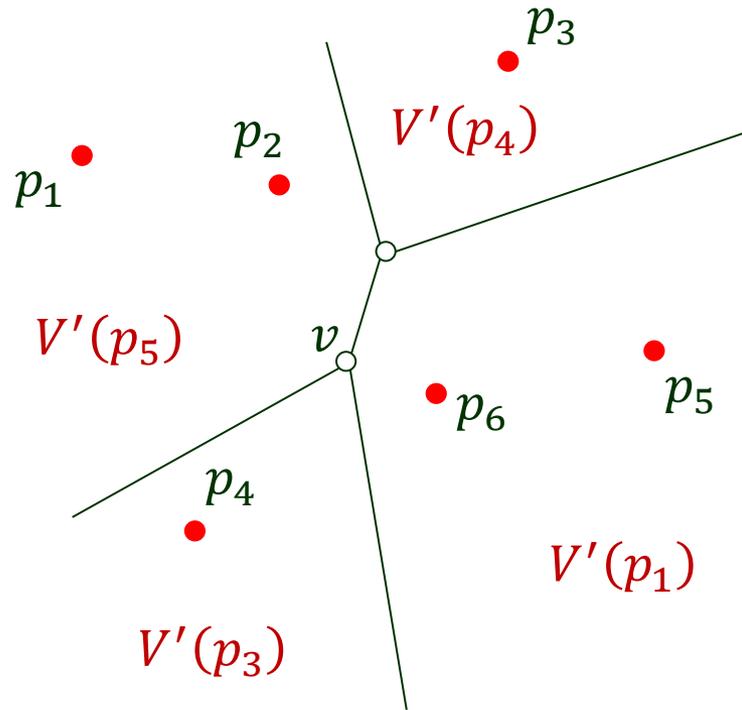
The cell contains a ray  $r$  collinear with  $p_i$ .



- $p_i$ : farthest point from  $q$ .
- $l$ : the line through  $p_i$  and  $q$ .
- $r$ : half-line starting at  $q$  and away from  $p_i$ .

All the points on  $r$  have  $p_i$  as the farthest point!

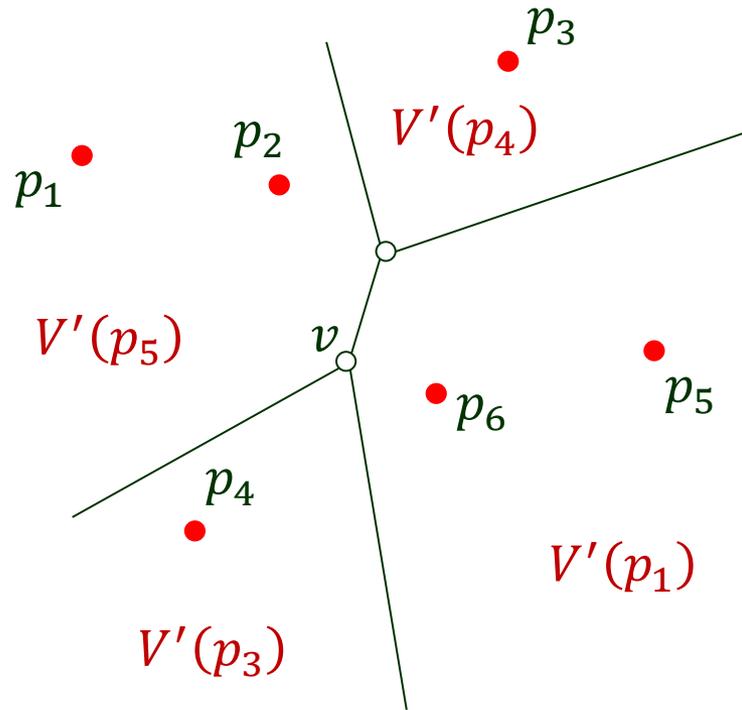
# III. Farthest-Point Voronoi Diagram



## Tree-like structure

- ◆ Edges include segments and half-infinite lines.

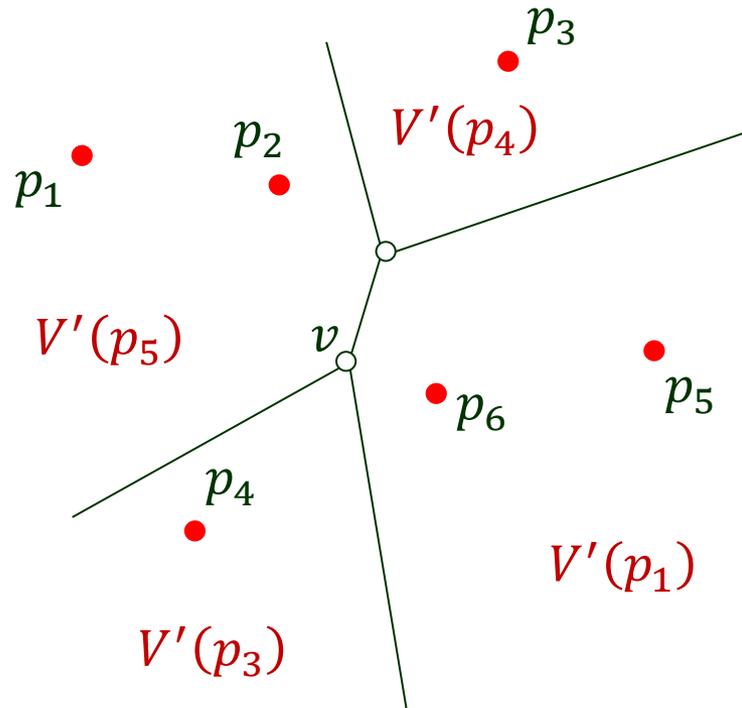
# III. Farthest-Point Voronoi Diagram



## Tree-like structure

- ◆ Edges include segments and half-infinite lines.
- ◆ No cycles

# III. Farthest-Point Voronoi Diagram

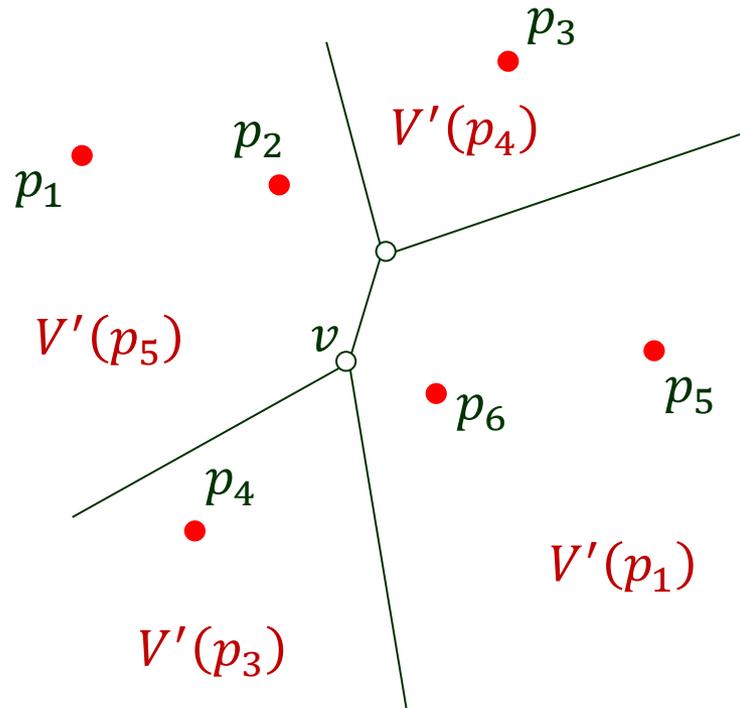


## Tree-like structure

- ◆ Edges include segments and half-infinite lines.
- ◆ No cycles

A cycle would imply a bounded cell.

# III. Farthest-Point Voronoi Diagram



## Tree-like structure

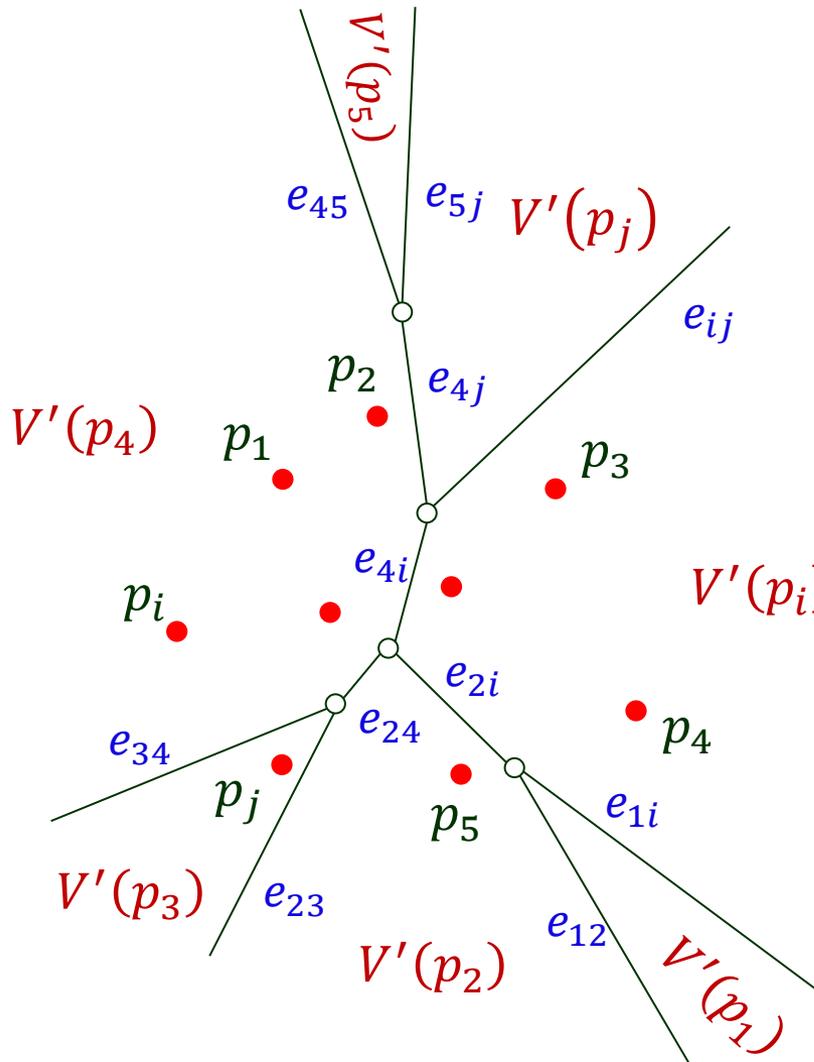
- ◆ Edges include segments and half-infinite lines.

- ◆ No cycles

A cycle would imply a bounded cell.

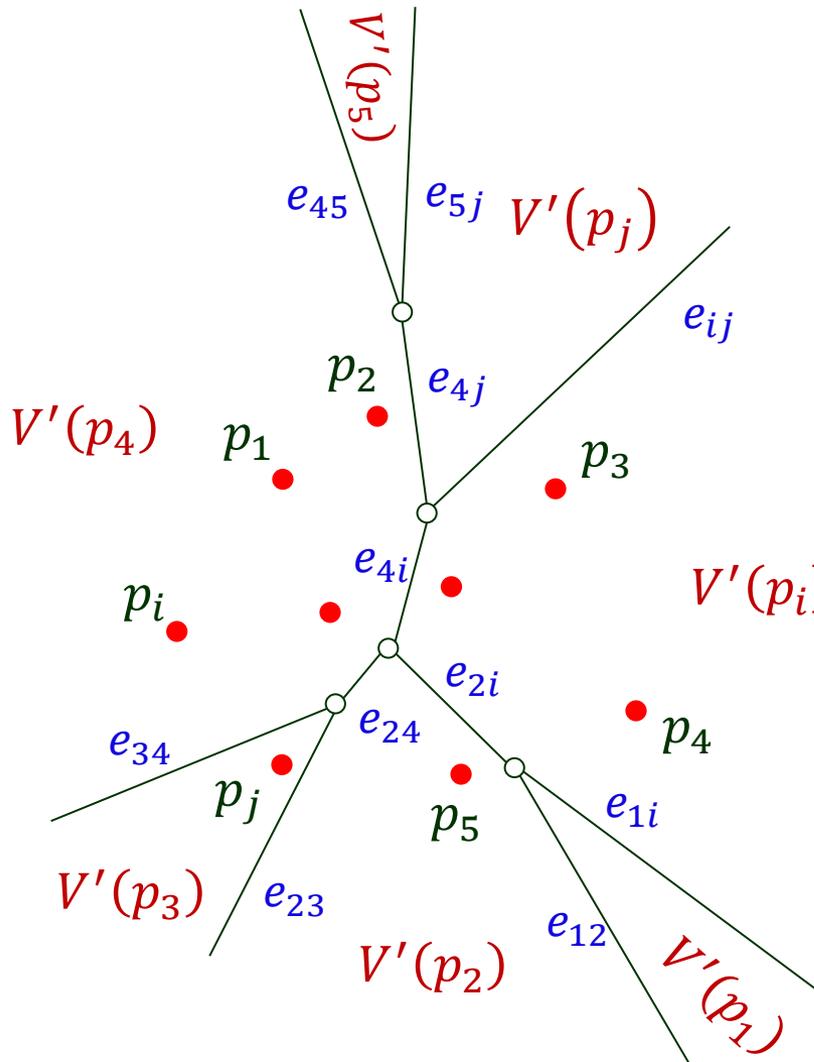
- ◆ A vertex has  $\geq 3$  farthest sites.

# More Properties



- ◆ Any site that is not a vertex of the convex hull has *no Voronoi cell*.

# More Properties

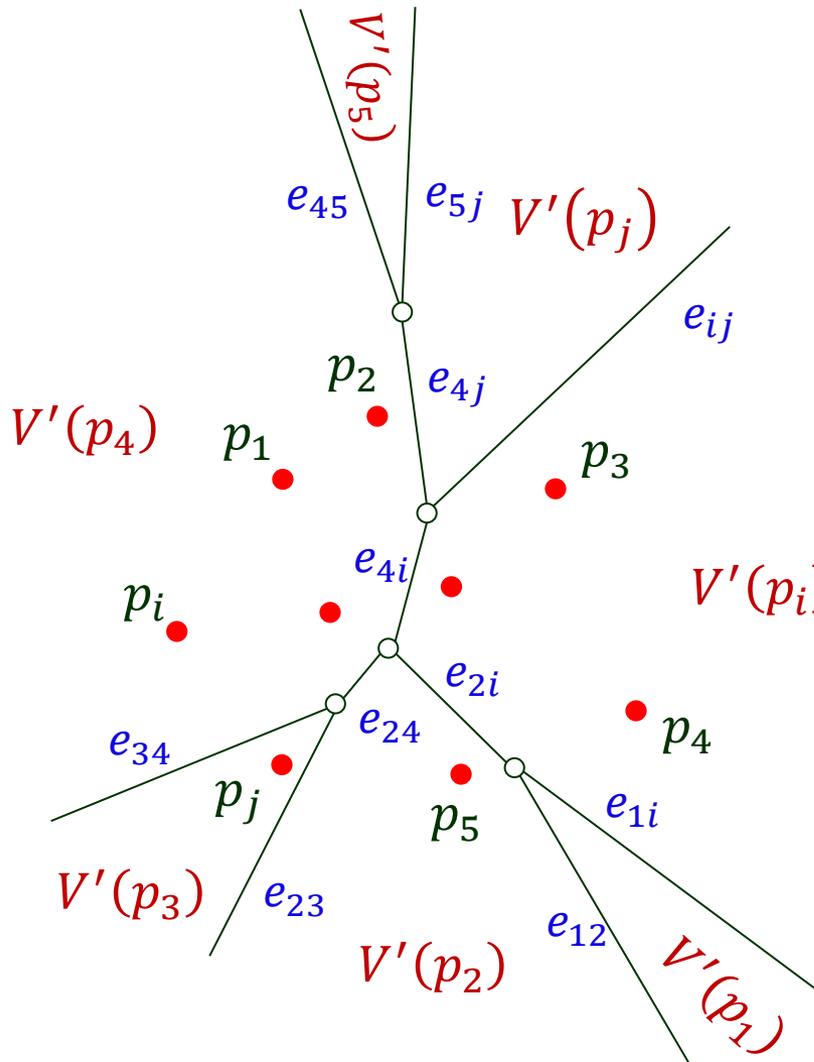


- ◆ Any site that is not a vertex of the convex hull has *no Voronoi cell*.



It contributes no Voronoi edge.

# More Properties



- ◆ Any site that is not a vertex of the convex hull has *no Voronoi cell*.

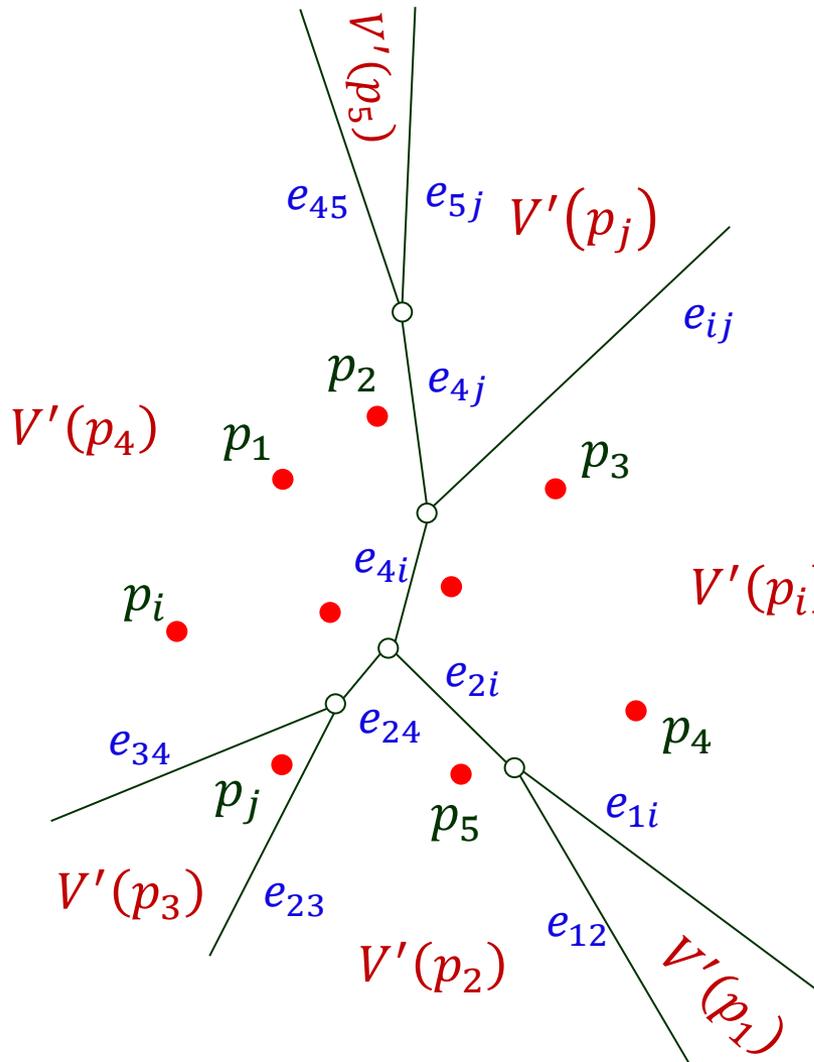


It contributes no Voronoi edge.



- ◆ Every Voronoi edge is part of a bisector of two convex hull vertices.

# More Properties



- ◆ Any site that is not a vertex of the convex hull has *no Voronoi cell*.



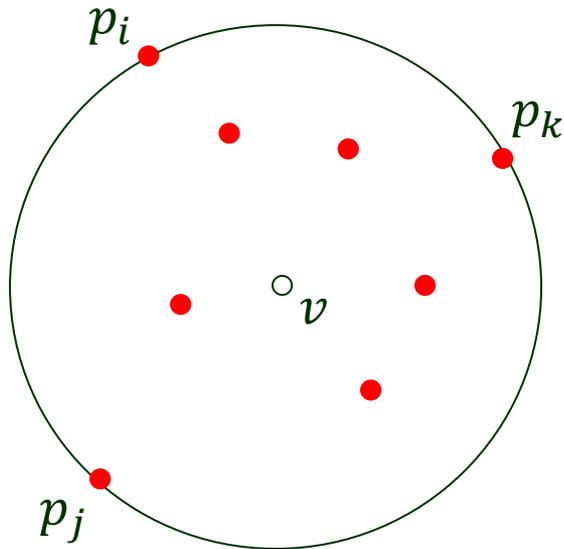
It contributes no Voronoi edge.



- ◆ Every Voronoi edge is part of a bisector of two convex hull vertices.
- ◆  $O(n)$  vertices, edges and cells

# Center of Smallest Enclosing Disk

---

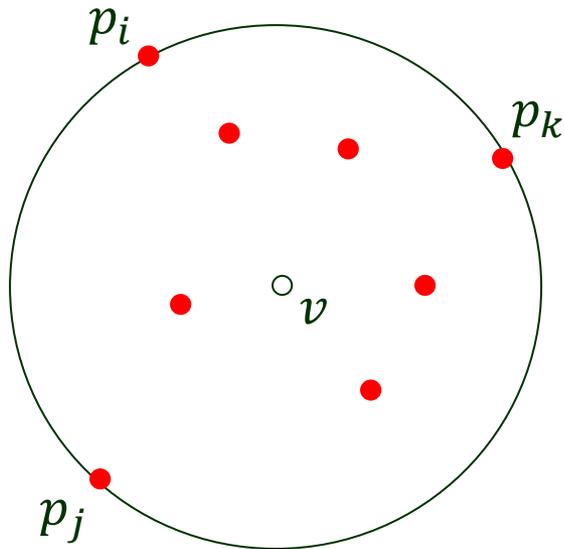


Two possibilities:

- ◆ Vertex  $\Rightarrow \geq 3$  equidistant farthest sites

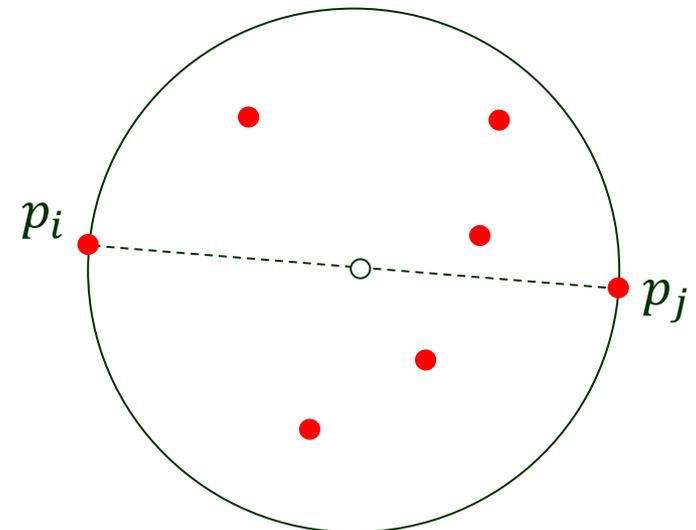
# Center of Smallest Enclosing Disk

---



Two possibilities:

- ◆ Vertex  $\Rightarrow \geq 3$  equidistant farthest sites
- ◆ Midpoint of two sites defining an edge  $\Rightarrow$  two equidistant farthest sites



# Storage

---

Doubly-connected edge list (DCEL) with modifications

Half-infinite edge  $v \circ \overleftrightarrow{\hspace{1.5cm}} \vec{d} = (-1, 0)$   
 $e$

- ♣ If no origin, stores the direction of the edge ( $\vec{d}$ ) instead of coordinates.
- ♣ Either  $\text{next}(e)$  or  $\text{prev}(e)$  is undefined.

# IV. Preprocessing for Construction

---

1. Compute the convex hull  $\text{CH}(P)$  with  $h$  vertices.

# IV. Preprocessing for Construction

---

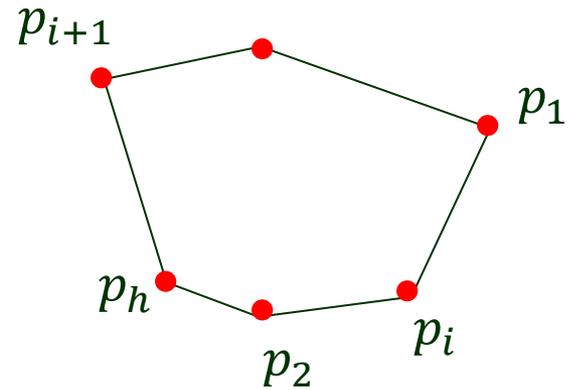
1. Compute the convex hull  $CH(P)$  with  $h$  vertices.
2. Order vertices of the hull randomly.

# IV. Preprocessing for Construction

---

1. Compute the convex hull  $CH(P)$  with  $h$  vertices.
2. Order vertices of the hull randomly.

$p_1, p_2, \dots, p_h$  (new indices)

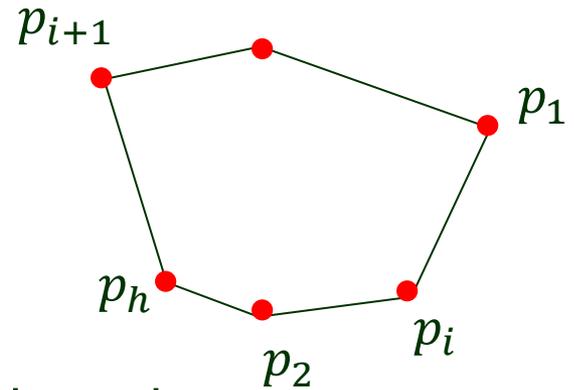


# IV. Preprocessing for Construction

---

1. Compute the convex hull  $CH(P)$  with  $h$  vertices.
2. Order vertices of the hull randomly.

$p_1, p_2, \dots, p_h$  (new indices)



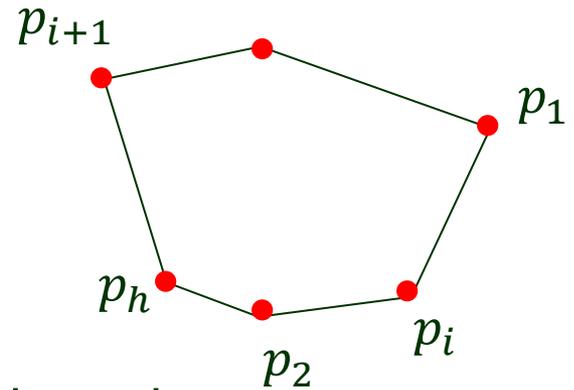
3. Remove  $p_h, p_{h-1}, \dots, p_4$  one by one in the order.

# IV. Preprocessing for Construction

---

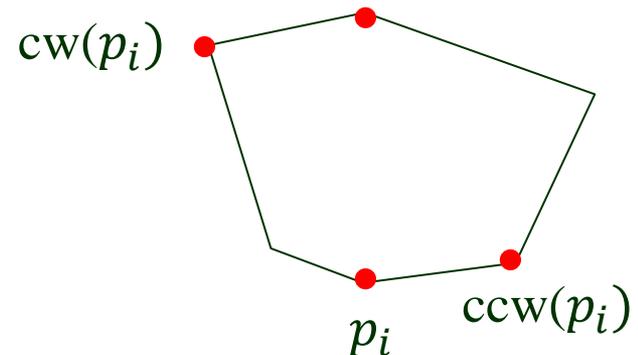
1. Compute the convex hull  $CH(P)$  with  $h$  vertices.
2. Order vertices of the hull randomly.

$p_1, p_2, \dots, p_h$  (new indices)



3. Remove  $p_h, p_{h-1}, \dots, p_4$  one by one in the order.

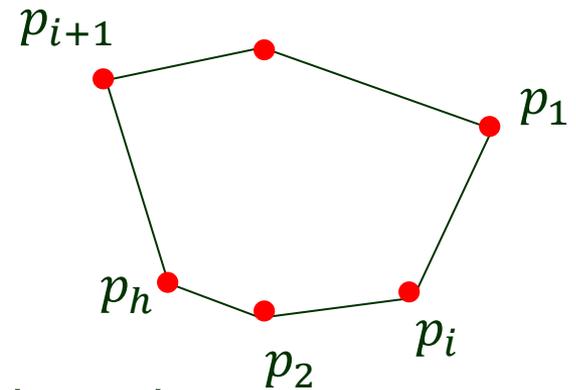
- For each  $p_i$ , store its clockwise neighbor  $cw(p_i)$  and counterclockwise neighbor  $ccw(p_i)$  at the time of removal.



# IV. Preprocessing for Construction

1. Compute the convex hull  $CH(P)$  with  $h$  vertices.
2. Order vertices of the hull randomly.

$p_1, p_2, \dots, p_h$  (new indices)

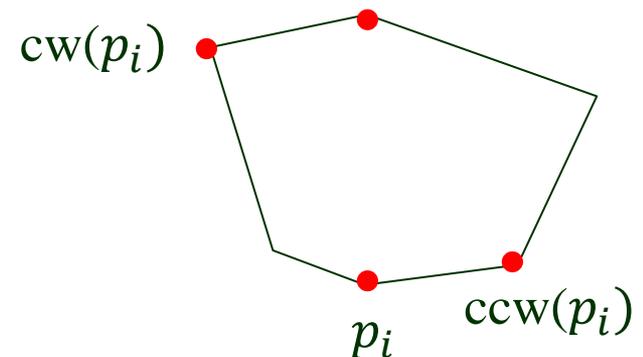


3. Remove  $p_h, p_{h-1}, \dots, p_4$  one by one in the order.

- For each  $p_i$ , store its clockwise neighbor  $cw(p_i)$  and counterclockwise neighbor  $ccw(p_i)$  at the time of removal.



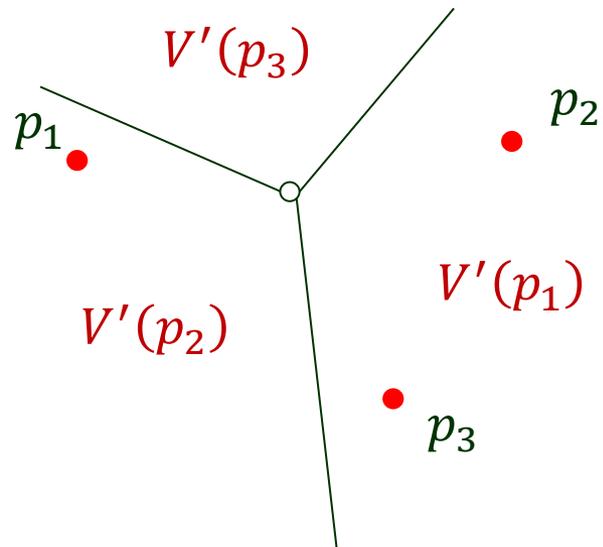
$p_i$  cannot be a neighbor of any point removed later.



# Construction

---

1. Initialize with the FPVD of  $p_1, p_2, p_3$ .

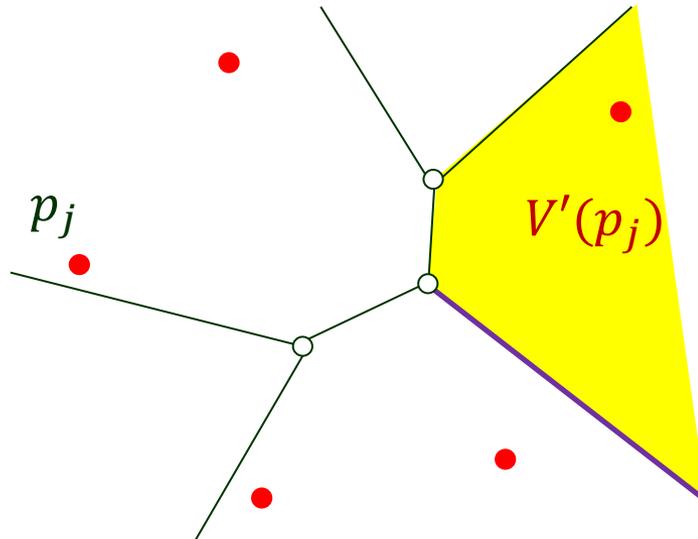


# Construction (cont'd)

---

2. Insert  $p_4, p_5, \dots, p_h$  one by one in the order.

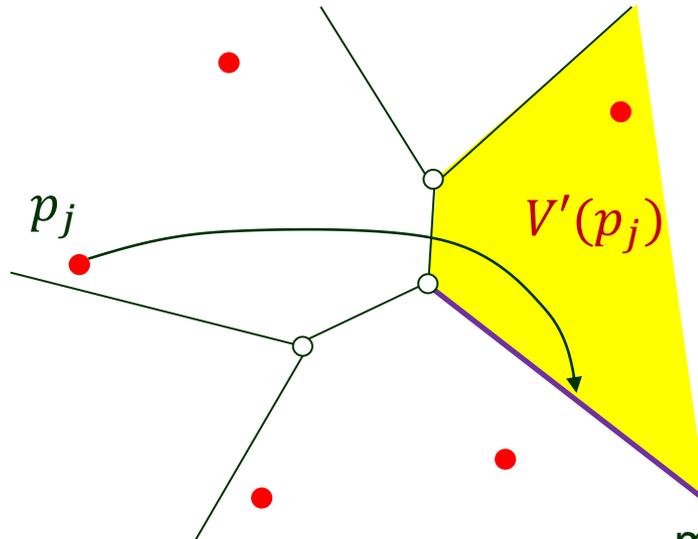
$FPVD_{i-1}$  for  $\{p_1, \dots, p_{i-1}\}$ :



# Construction (cont'd)

2. Insert  $p_4, p_5, \dots, p_h$  one by one in the order.

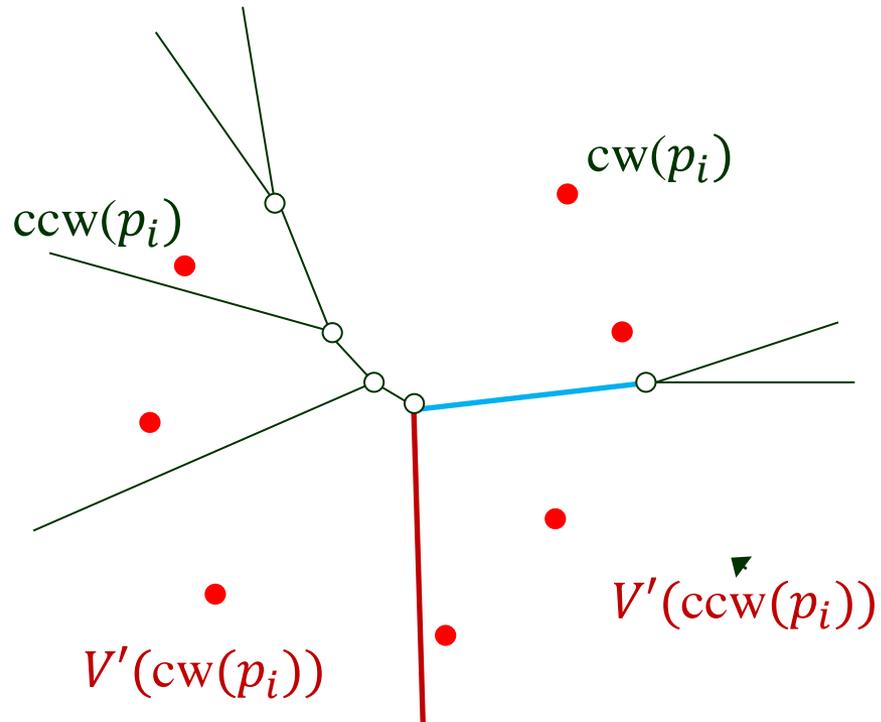
$FPVD_{i-1}$  for  $\{p_1, \dots, p_{i-1}\}$ :



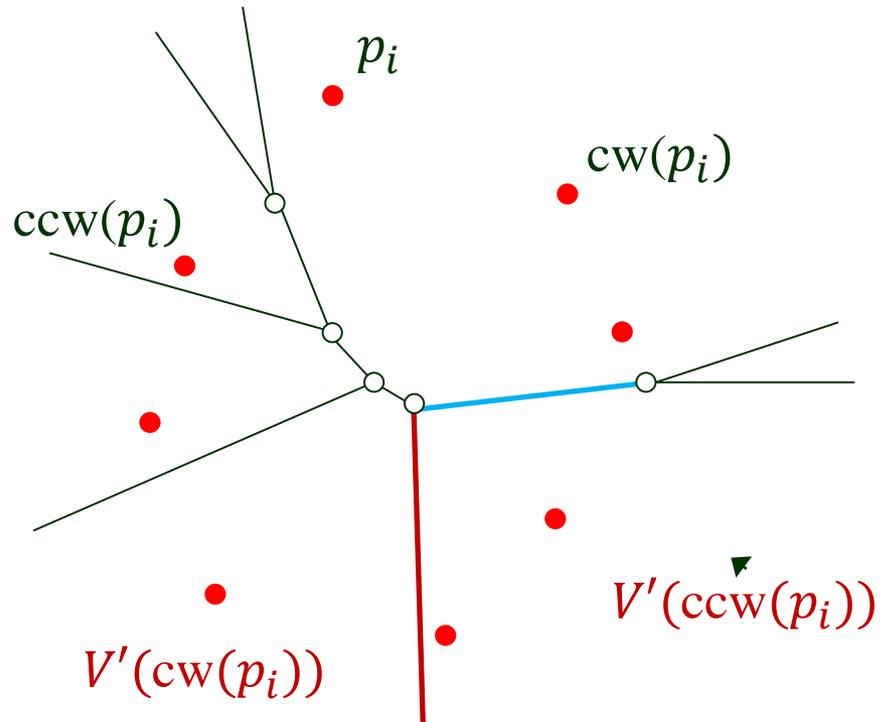
most counterclockwise  
half-edge in a traversal of  
the boundary of  $V'(p_j)$

# How to Add $p_i$ ?

---

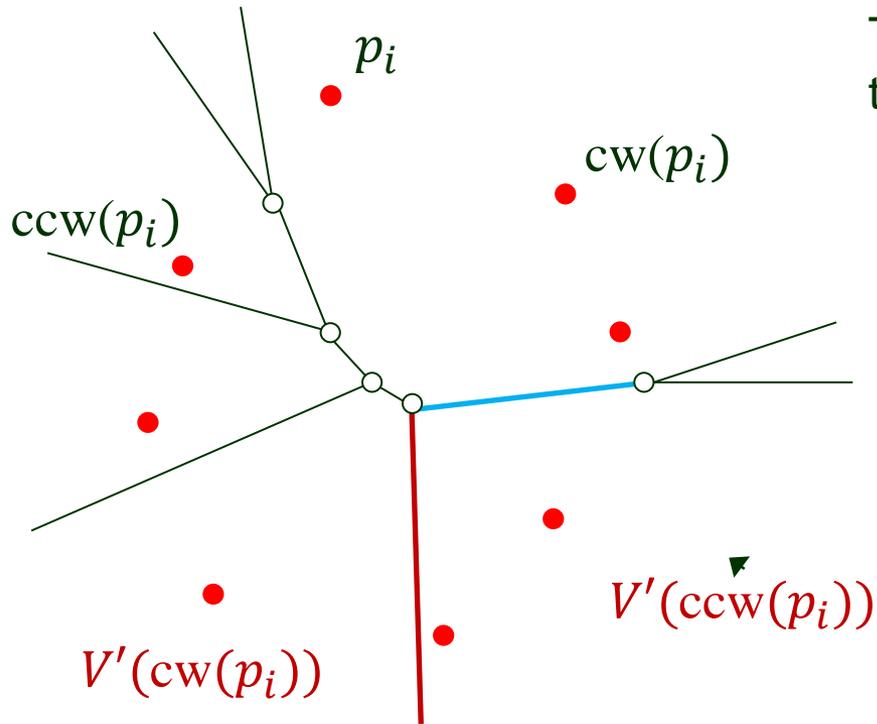


# How to Add $p_i$ ?



# How to Add $p_i$ ?

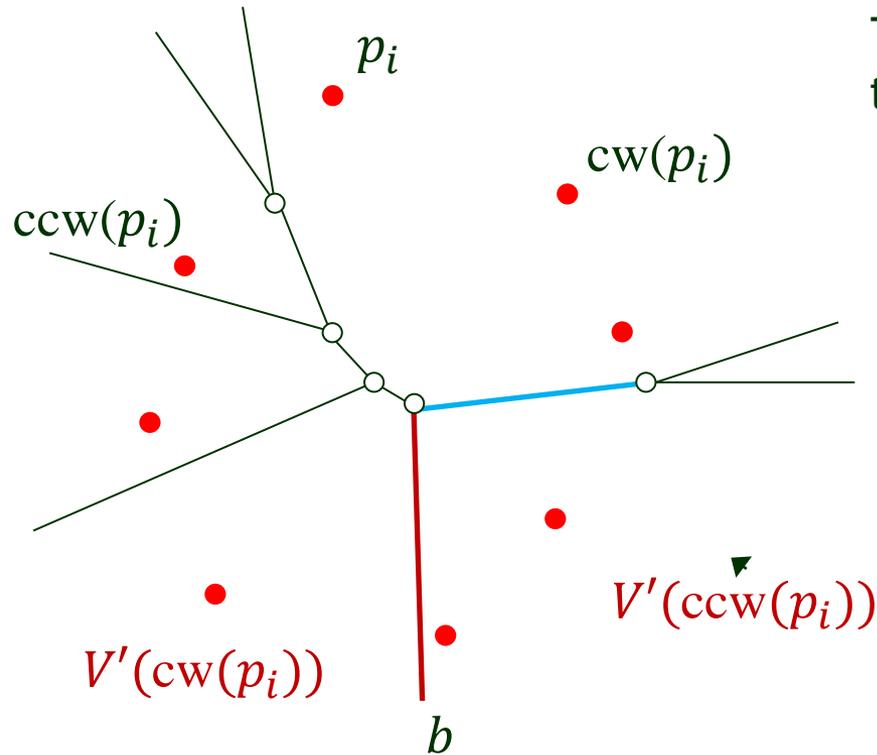
The cell  $V'(p_i)$  of  $p_i$  will come in between the adjacent cells  $V'(cw(p_i))$  and  $V'(ccw(p_i))$ .



# How to Add $p_i$ ?

The cell  $V'(p_i)$  of  $p_i$  will come in between the adjacent cells  $V'(\text{cw}(p_i))$  and  $V'(\text{ccw}(p_i))$ .

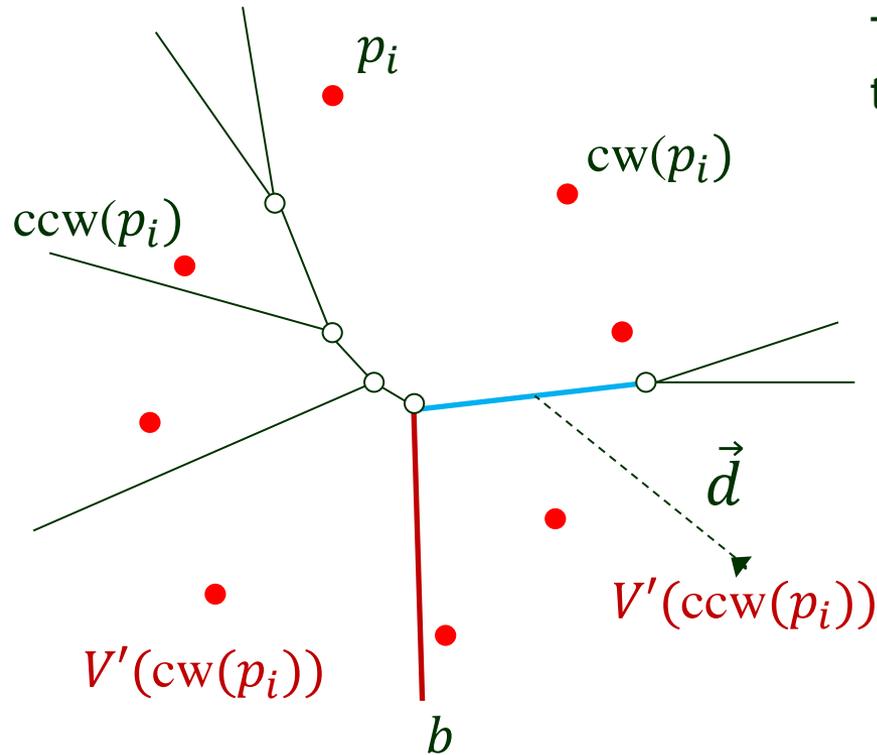
- $\text{ccw}(p_i)$  has a pointer to bisector  $b$  (most counterclockwise edge in its cell).



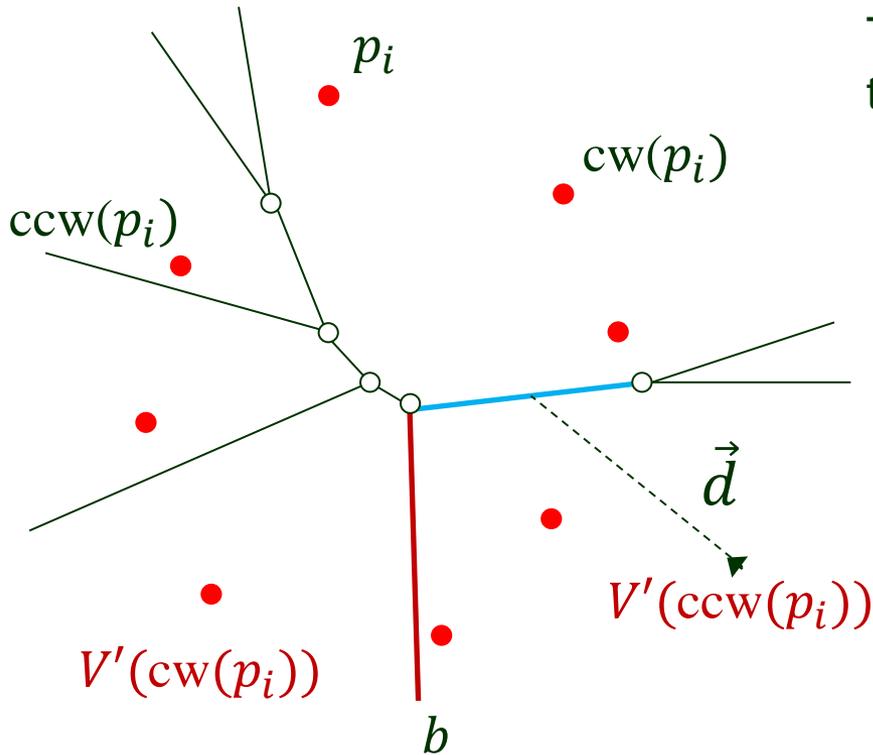
# How to Add $p_i$ ?

The cell  $V'(p_i)$  of  $p_i$  will come in between the adjacent cells  $V'(\text{cw}(p_i))$  and  $V'(\text{ccw}(p_i))$ .

- $\text{ccw}(p_i)$  has a pointer to bisector  $b$  (most counterclockwise edge in its cell).
- Bisector of  $\text{ccw}(p_i)$  and  $p_i$  will contribute a half-edge  $\vec{d}$  to  $V'(p_i)$ .



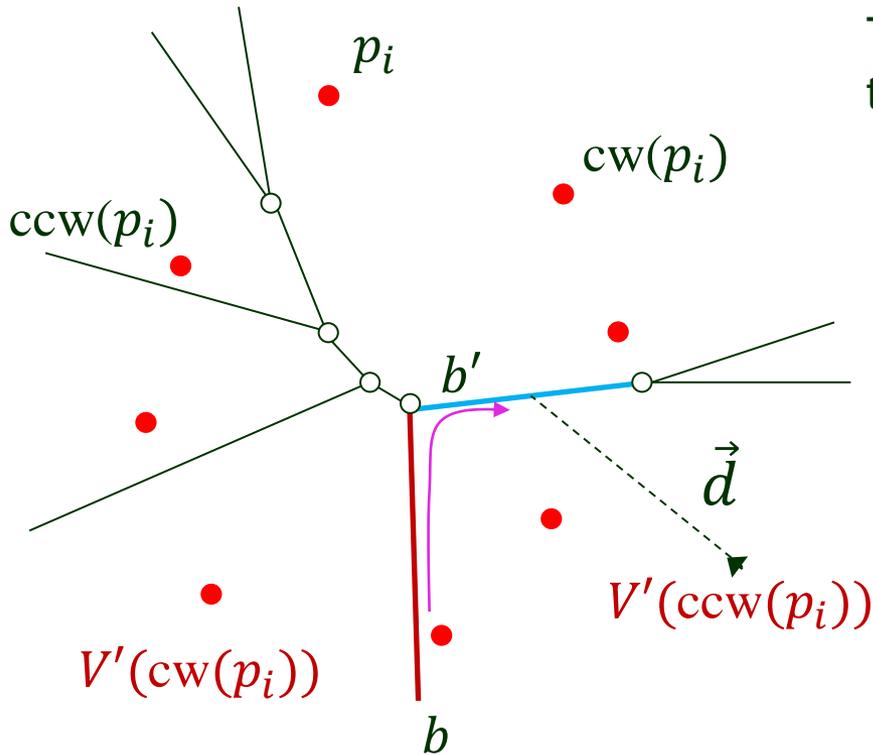
# How to Add $p_i$ ?



The cell  $V'(p_i)$  of  $p_i$  will come in between the adjacent cells  $V'(cw(p_i))$  and  $V'(ccw(p_i))$ .

- $ccw(p_i)$  has a pointer to bisector  $b$  (most counterclockwise edge in its cell).
- Bisector of  $ccw(p_i)$  and  $p_i$  will contribute a half-edge  $\vec{d}$  to  $V'(p_i)$ .
- Traverse the boundary of  $V'(ccw(p_i))$ , starting at  $b$ , in a clockwise way to find the intersection  $q$  of  $\vec{d}$  with a boundary edge  $b'$  between  $V'(ccw(p_i))$  and, say,  $V'(p_j)$  of another site  $p_j$ .

# How to Add $p_i$ ?

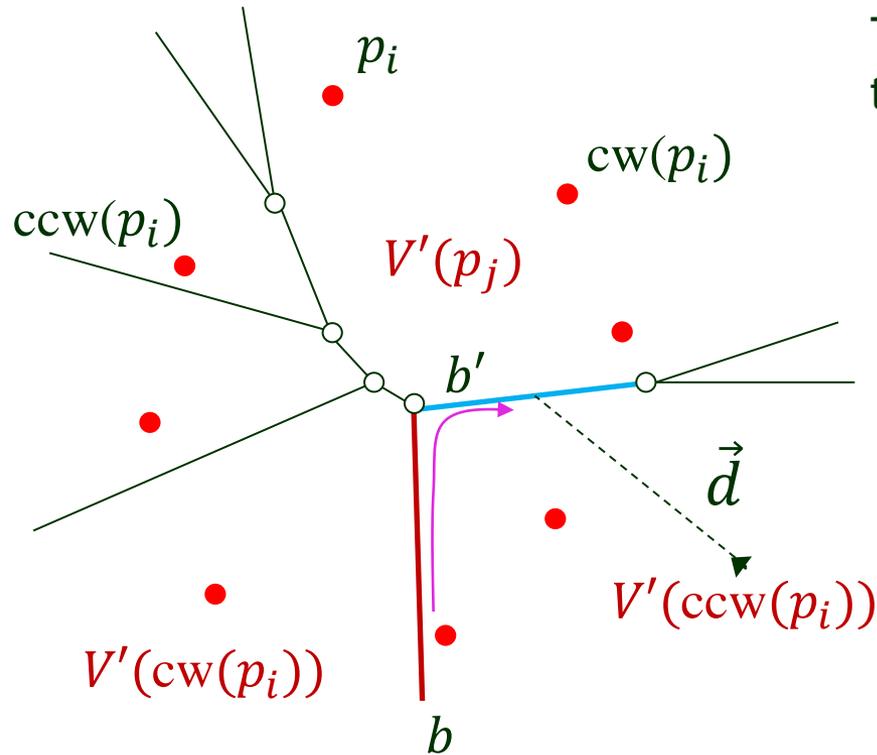


The cell  $V'(p_i)$  of  $p_i$  will come in between the adjacent cells  $V'(cw(p_i))$  and  $V'(ccw(p_i))$ .

- $ccw(p_i)$  has a pointer to bisector  $b$  (most counterclockwise edge in its cell).
- Bisector of  $ccw(p_i)$  and  $p_i$  will contribute a half-edge  $\vec{d}$  to  $V'(p_i)$ .
- Traverse the boundary of  $V'(ccw(p_i))$ , starting at  $b$ , in a clockwise way to find the intersection  $q$  of  $\vec{d}$  with a boundary edge  $b'$  between  $V'(ccw(p_i))$  and, say,  $V'(p_j)$  of another site  $p_j$ .

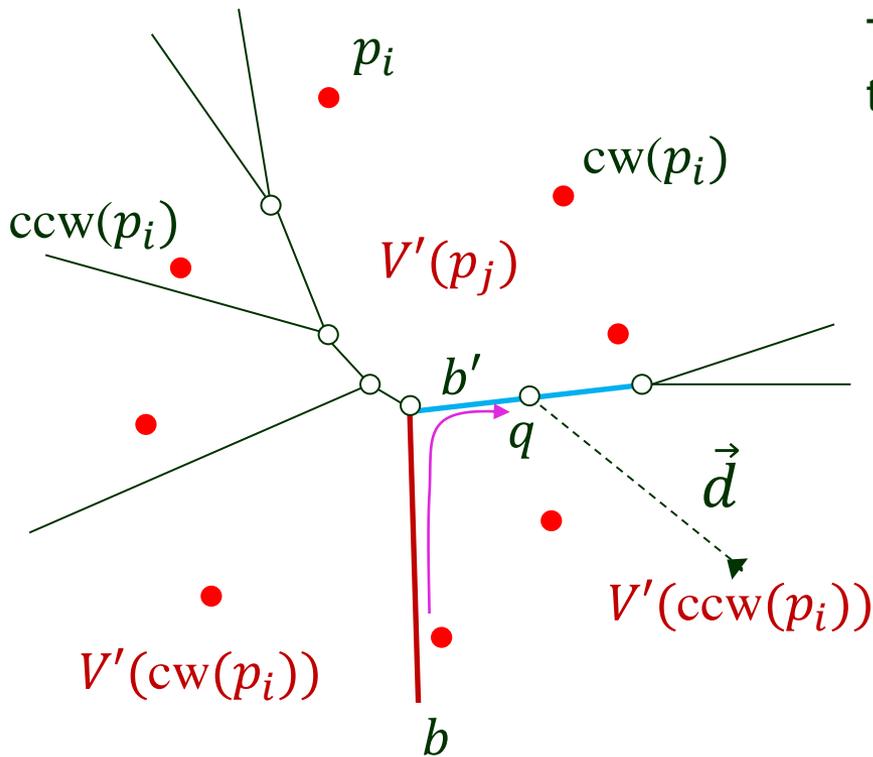
# How to Add $p_i$ ?

The cell  $V'(p_i)$  of  $p_i$  will come in between the adjacent cells  $V'(\text{cw}(p_i))$  and  $V'(\text{ccw}(p_i))$ .



- $\text{ccw}(p_i)$  has a pointer to bisector  $b$  (most counterclockwise edge in its cell).
- Bisector of  $\text{ccw}(p_i)$  and  $p_i$  will contribute a half-edge  $\vec{d}$  to  $V'(p_i)$ .
- Traverse the boundary of  $V'(\text{ccw}(p_i))$ , starting at  $b$ , in a clockwise way to find the intersection  $q$  of  $\vec{d}$  with a boundary edge  $b'$  between  $V'(\text{ccw}(p_i))$  and, say,  $V'(p_j)$  of another site  $p_j$ .

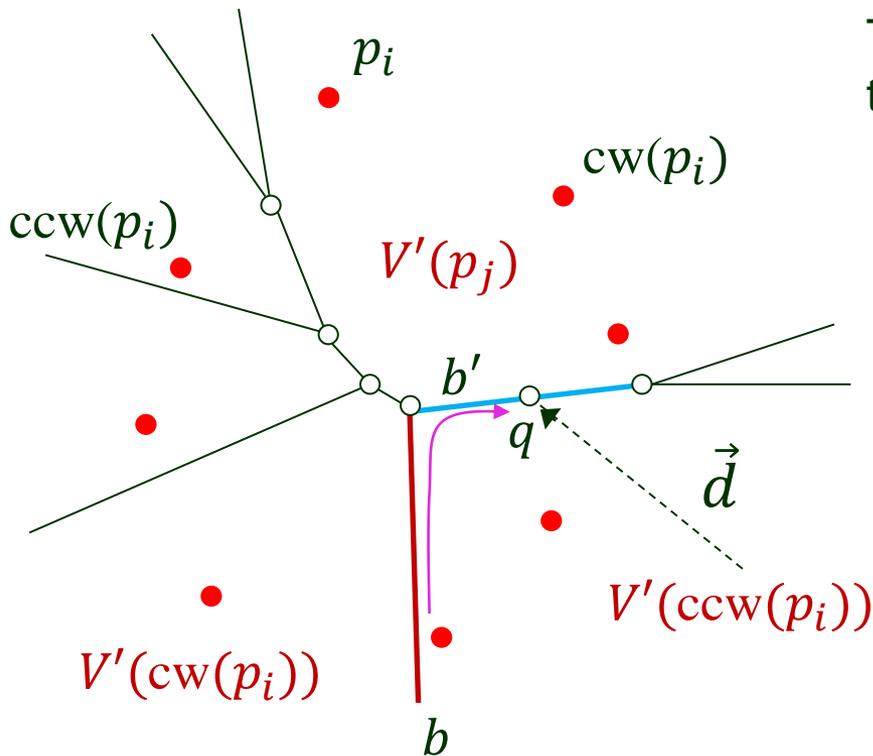
# How to Add $p_i$ ?



The cell  $V'(p_i)$  of  $p_i$  will come in between the adjacent cells  $V'(cw(p_i))$  and  $V'(ccw(p_i))$ .

- $ccw(p_i)$  has a pointer to bisector  $b$  (most counterclockwise edge in its cell).
- Bisector of  $ccw(p_i)$  and  $p_i$  will contribute a half-edge  $\vec{d}$  to  $V'(p_i)$ .
- Traverse the boundary of  $V'(ccw(p_i))$ , starting at  $b$ , in a clockwise way to find the intersection  $q$  of  $\vec{d}$  with a boundary edge  $b'$  between  $V'(ccw(p_i))$  and, say,  $V'(p_j)$ .
- Move along  $\vec{d}$  to  $q$  and cross into the cell of  $p_j$ .

# How to Add $p_i$ ?

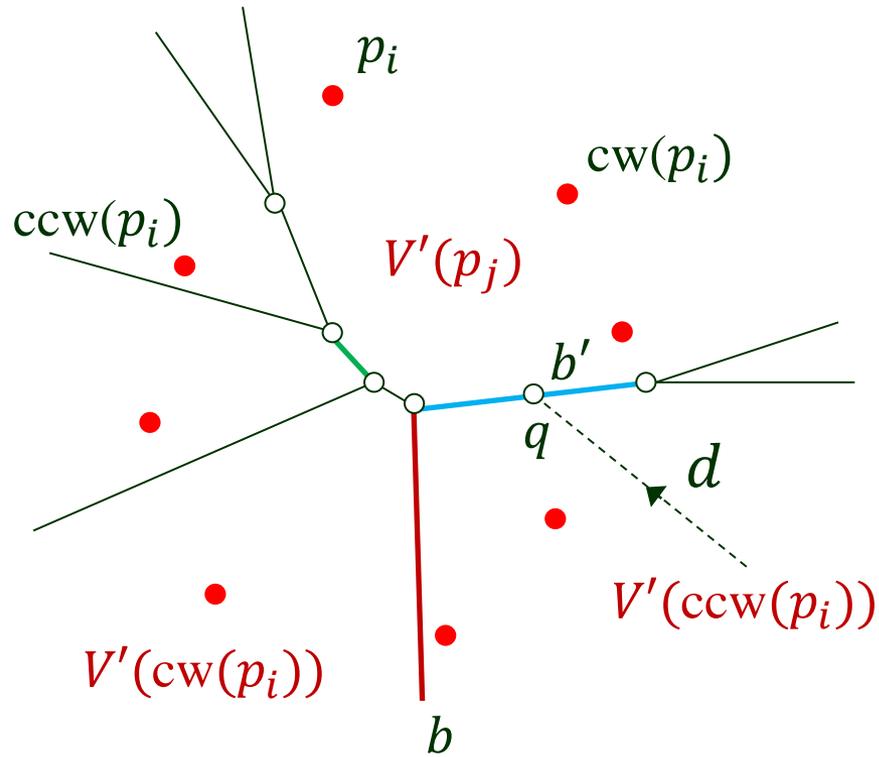


The cell  $V'(p_i)$  of  $p_i$  will come in between the adjacent cells  $V'(cw(p_i))$  and  $V'(ccw(p_i))$ .

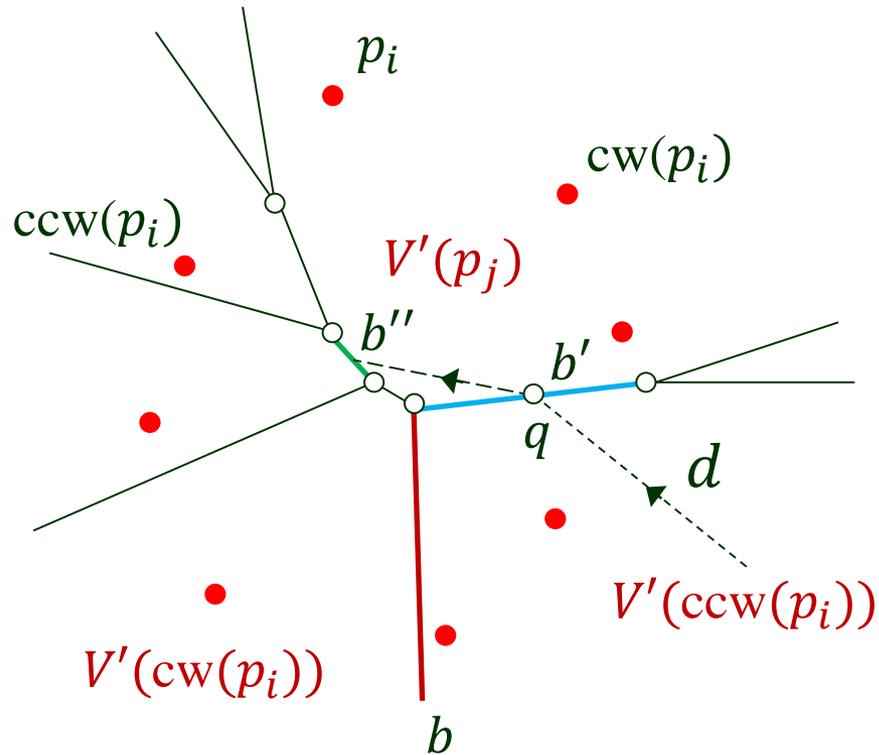
- $ccw(p_i)$  has a pointer to bisector  $b$  (most counterclockwise edge in its cell).
- Bisector of  $ccw(p_i)$  and  $p_i$  will contribute a half-edge  $\vec{d}$  to  $V'(p_i)$ .
- Traverse the boundary of  $V'(ccw(p_i))$ , starting at  $b$ , in a clockwise way to find the intersection  $q$  of  $\vec{d}$  with a boundary edge  $b'$  between  $V'(ccw(p_i))$  and, say,  $V'(p_j)$ .
- Move along  $\vec{d}$  to  $q$  and cross into the cell of  $p_j$ .

# Moving on ...

- At  $q$  start a clockwise traversal of the boundary of the cell  $V'(p_j)$ .

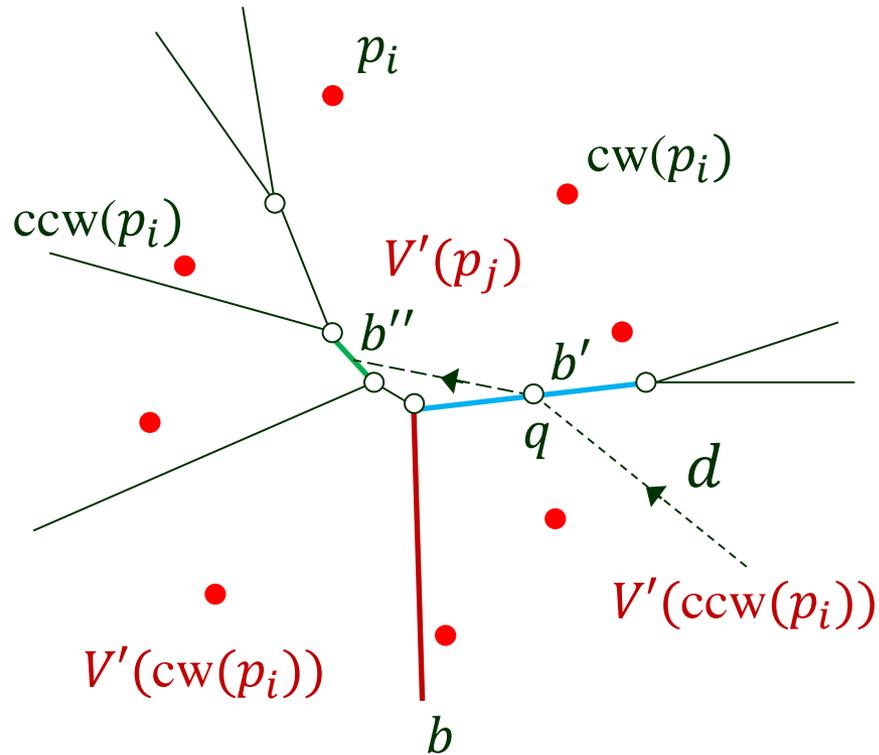


# Moving on ...



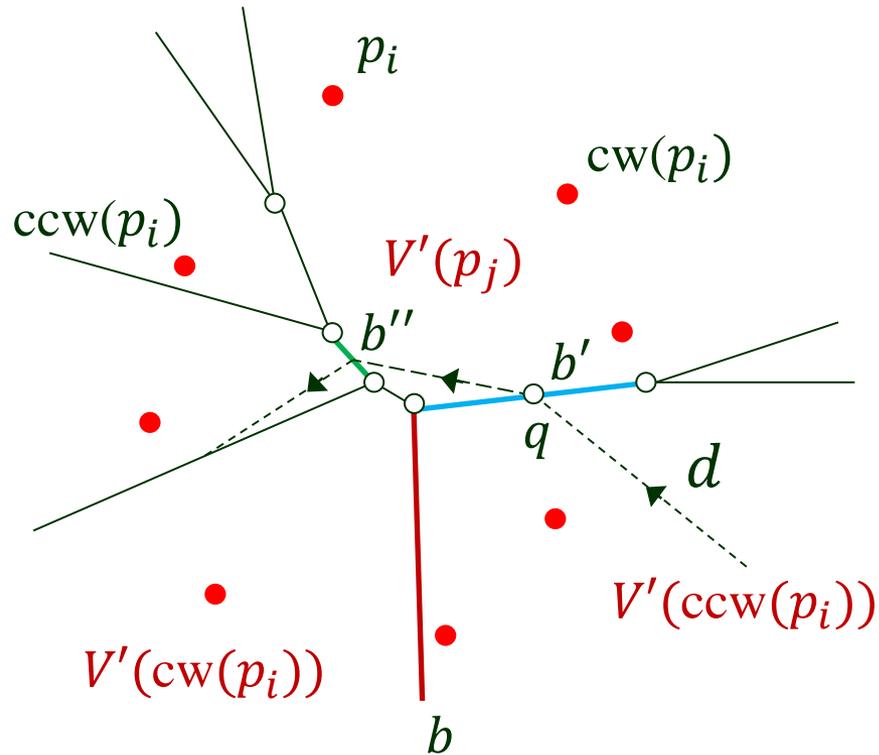
- At  $q$  start a clockwise traversal of the boundary of the cell  $V'(p_j)$ .
- Traversal stops at an edge  $b''$  that intersects the bisector of  $p_i$  and  $p_j$ .

# Moving on ...



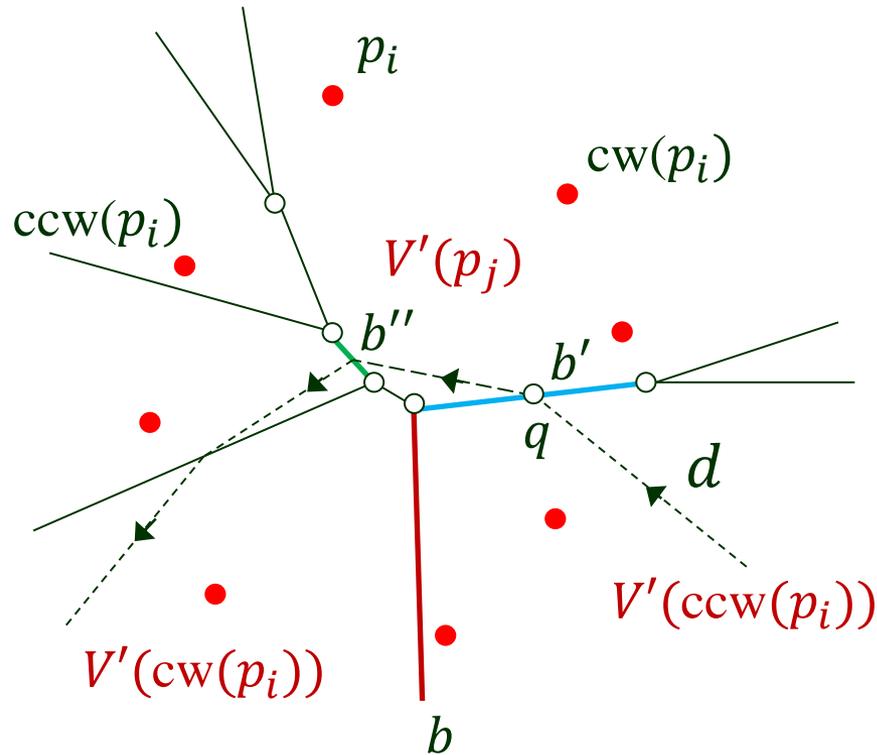
- At  $q$  start a clockwise traversal of the boundary of the cell  $V'(p_j)$ .
- Traversal stops at an edge  $b''$  that intersects the bisector of  $p_i$  and  $p_j$ .
- Exit the cell  $V'(p_j)$ , and so on.

# Moving on ...



- At  $q$  start a clockwise traversal of the boundary of the cell  $V'(p_j)$ .
- Traversal stops at an edge  $b''$  that intersects the bisector of  $p_i$  and  $p_j$ .
- Exit the cell  $V'(p_j)$ , and so on.

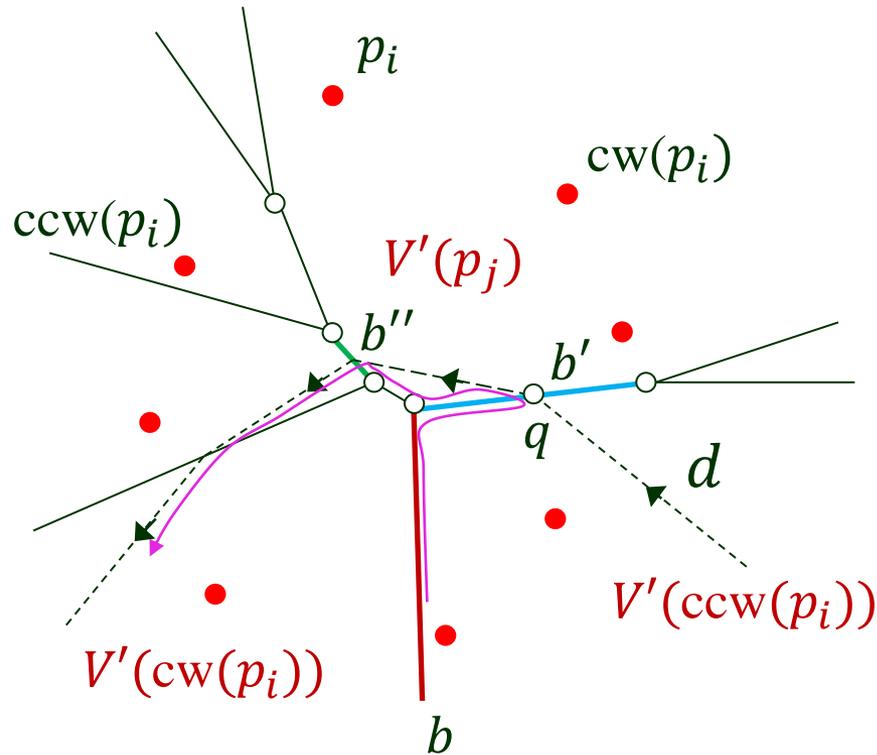
# Moving on ...



- At  $q$  start a clockwise traversal of the boundary of the cell  $V'(p_j)$ .
- Traversal stops at an edge  $b''$  that intersects the bisector of  $p_i$  and  $p_j$ .
- Exit the cell  $V'(p_j)$ , and so on.

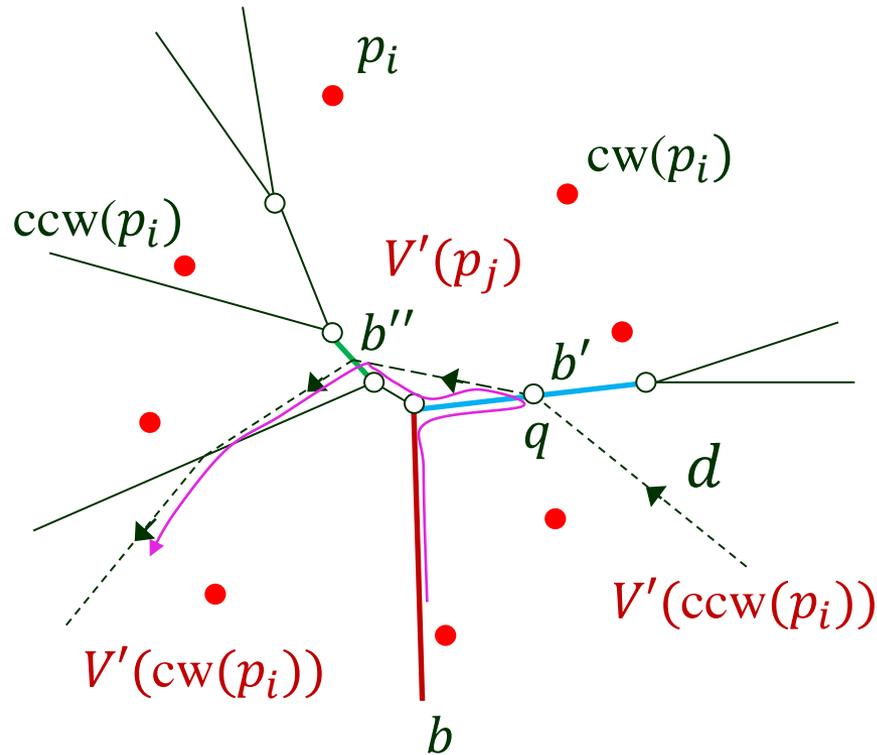


# Moving on ...



- At  $q$  start a clockwise traversal of the boundary of the cell  $V'(p_j)$ .
- Traversal stops at an edge  $b''$  that intersects the bisector of  $p_i$  and  $p_j$ .
- Exit the cell  $V'(p_j)$ , and so on.
- Last bisector will be between  $p_i$  and  $cw(p_i)$ .

# Moving on ...

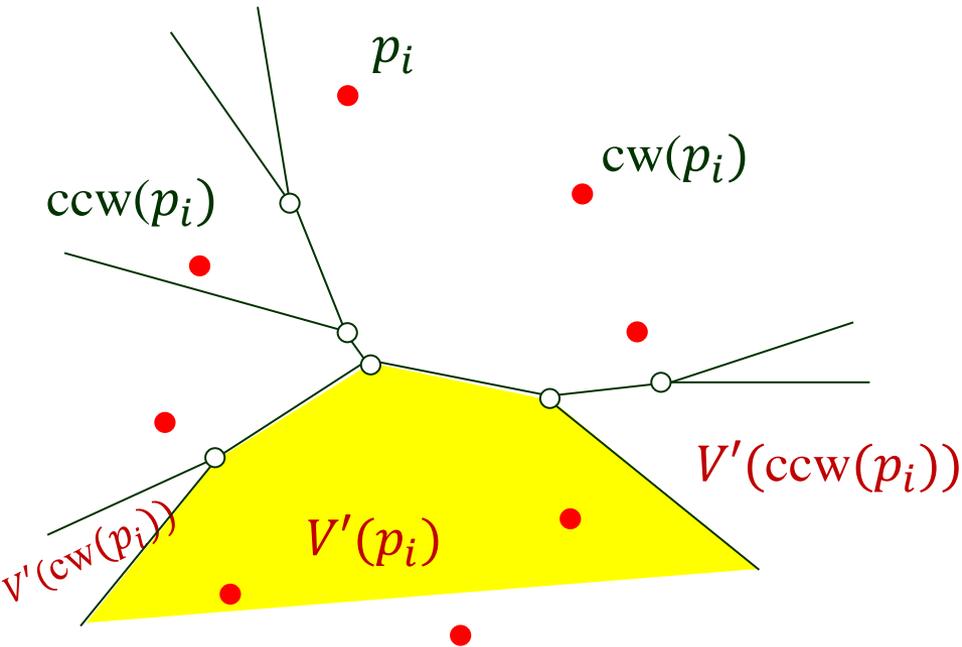


- At  $q$  start a clockwise traversal of the boundary of the cell  $V'(p_j)$ .
- Traversal stops at an edge  $b''$  that intersects the bisector of  $p_i$  and  $p_j$ .
- Exit the cell  $V'(p_j)$ , and so on.
- Last bisector will be between  $p_i$  and  $ccw(p_i)$ .

Trace out the boundary of  $V'(p_i)$  by traversing a sequence of cells, each in a clockwise way.

# Summary

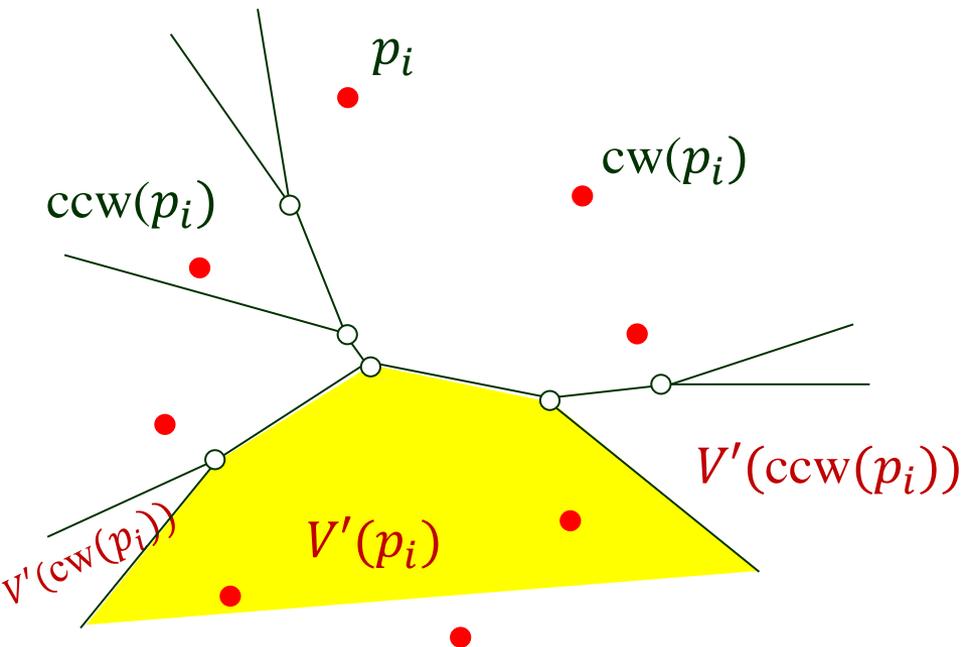
---



- ◆ All new edges are added to DCEL.
- ◆ Afterward, all the edges lying inside the cell of  $p_i$  are removed.

# Summary

---

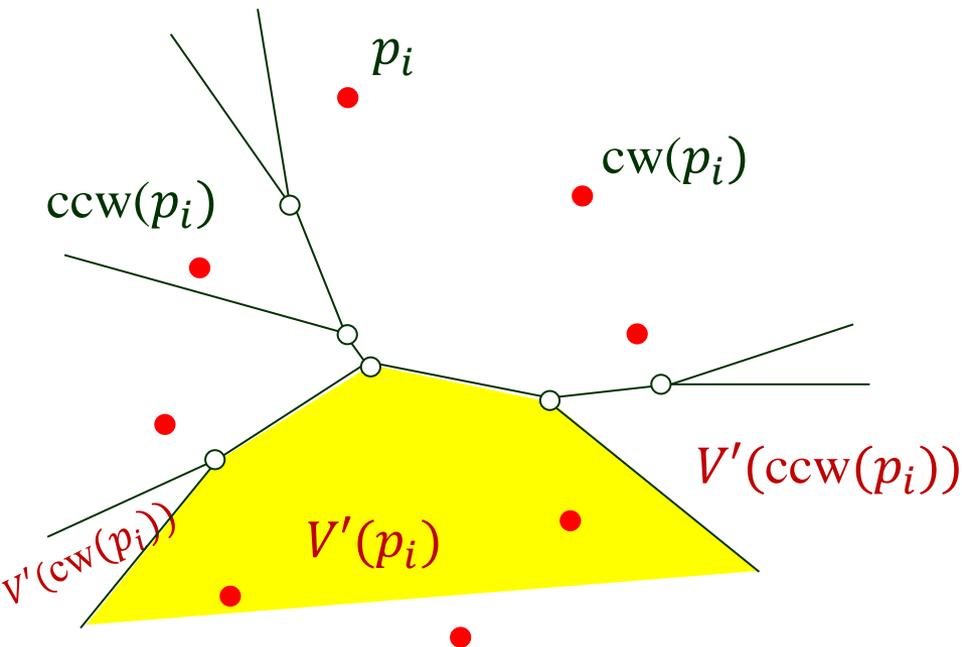


- ◆ All new edges are added to DCEL.
- ◆ Afterward, all the edges lying inside the cell of  $p_i$  are removed.

**Theorem** FPVD can be constructed in  $O(n \log n)$  expected time using  $O(n)$  storage.

# Summary

---



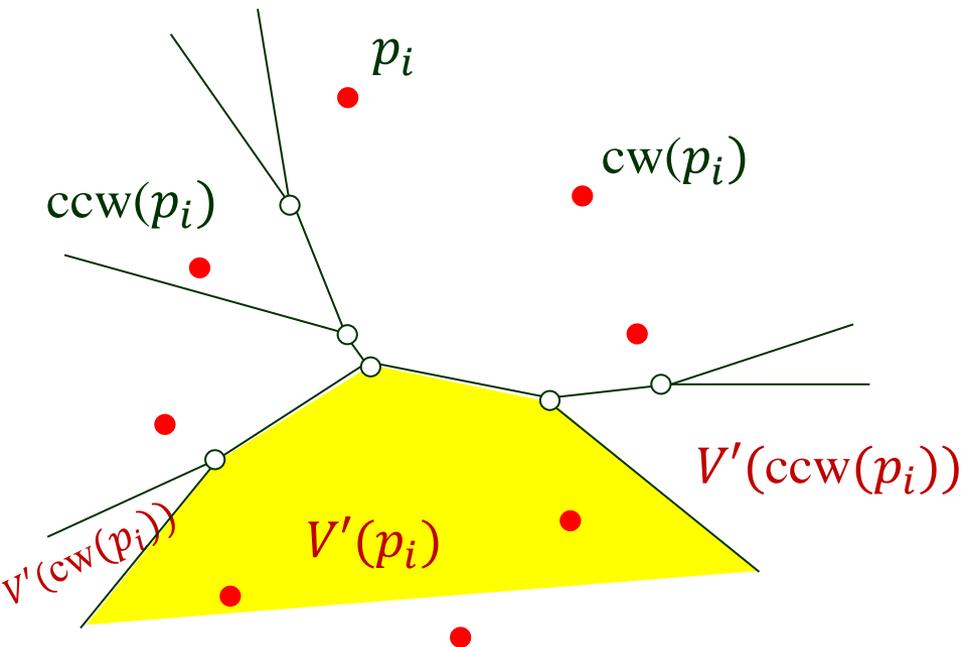
- ◆ All new edges are added to DCEL.
- ◆ Afterward, all the edges lying inside the cell of  $p_i$  are removed.

**Theorem** FPVD can be constructed in  $O(n \log n)$  expected time using  $O(n)$  storage.

- $O(n \log n)$  time to compute the convex hull.

# Summary

---



- ◆ All new edges are added to DCEL.
- ◆ Afterward, all the edges lying inside the cell of  $p_i$  are removed.

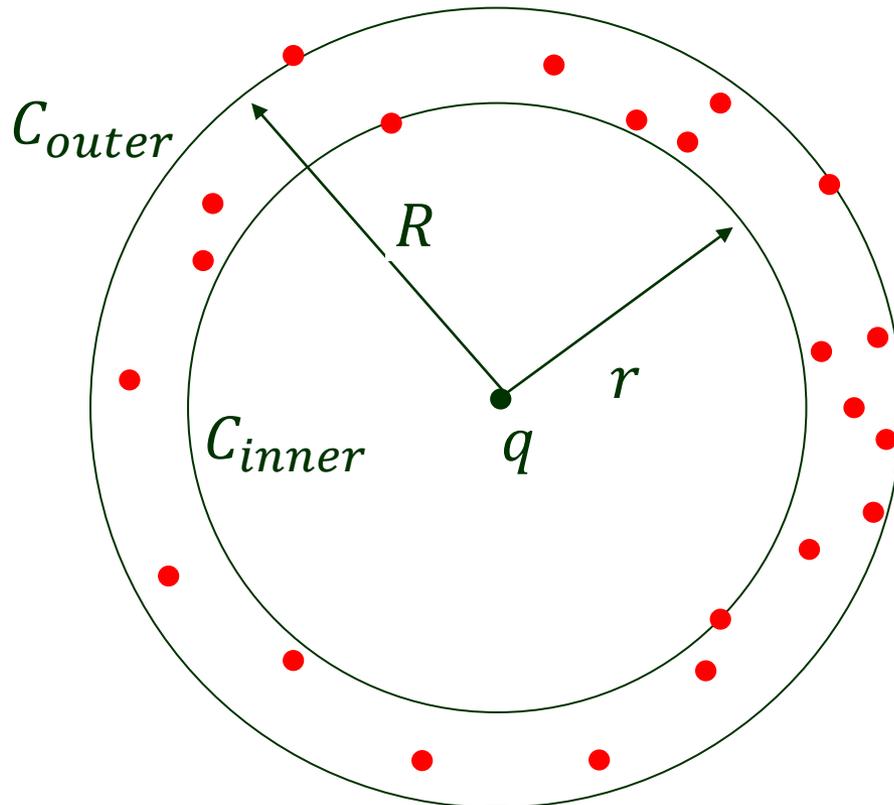
**Theorem** FPVD can be constructed in  $O(n \log n)$  expected time using  $O(n)$  storage.

- $O(n \log n)$  time to compute the convex hull.
- $O(n)$  time to construct FPVD (backward analysis).

# V. Roundness of a Point Set

---

The **roundness** of a set of points is measured by the **minimum width** of any annulus that contains the points.



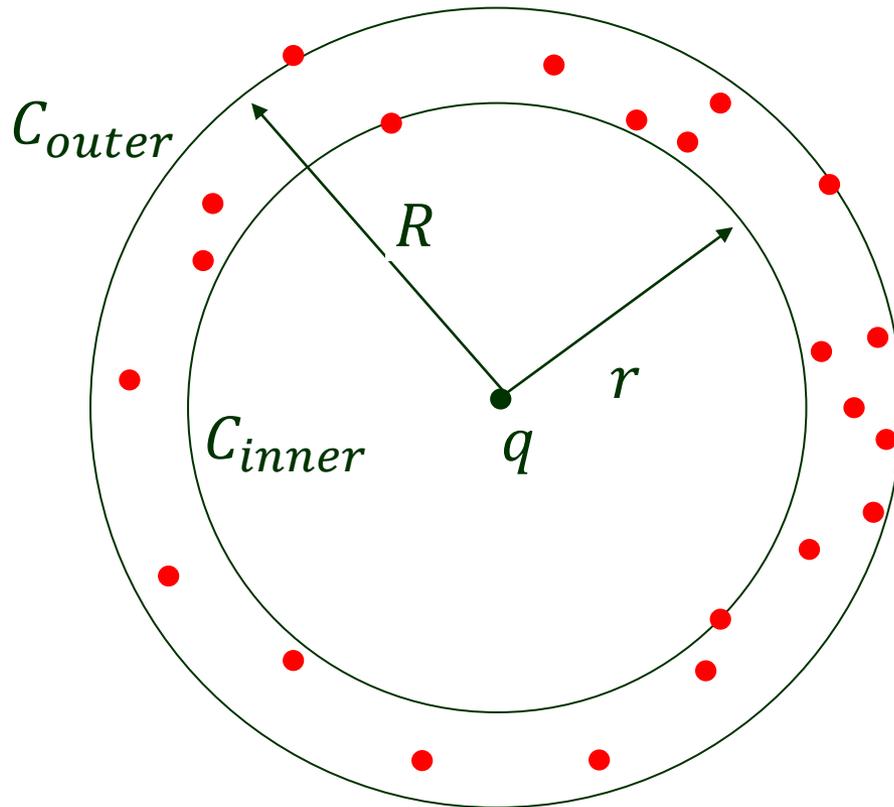
# V. Roundness of a Point Set

---

The **roundness** of a set of points is measured by the *minimum width* of any annulus that contains the points.

Observation:

There must be one point each on  $C_{outer}$  and  $C_{inner}$ .



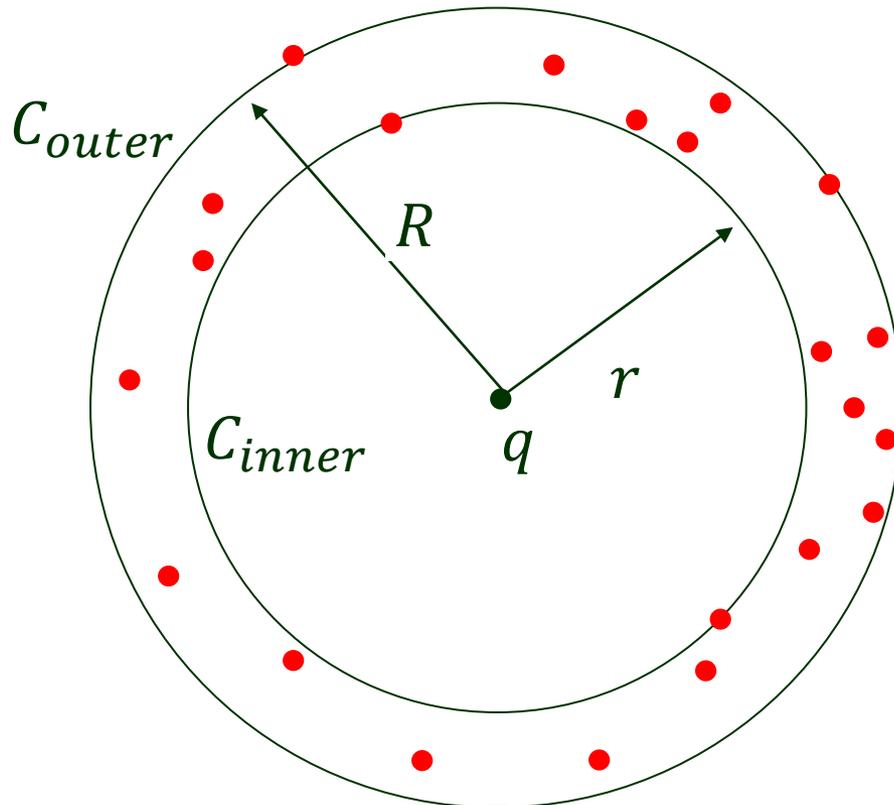
# V. Roundness of a Point Set

The **roundness** of a set of points is measured by the *minimum width* of any annulus that contains the points.

Observation:

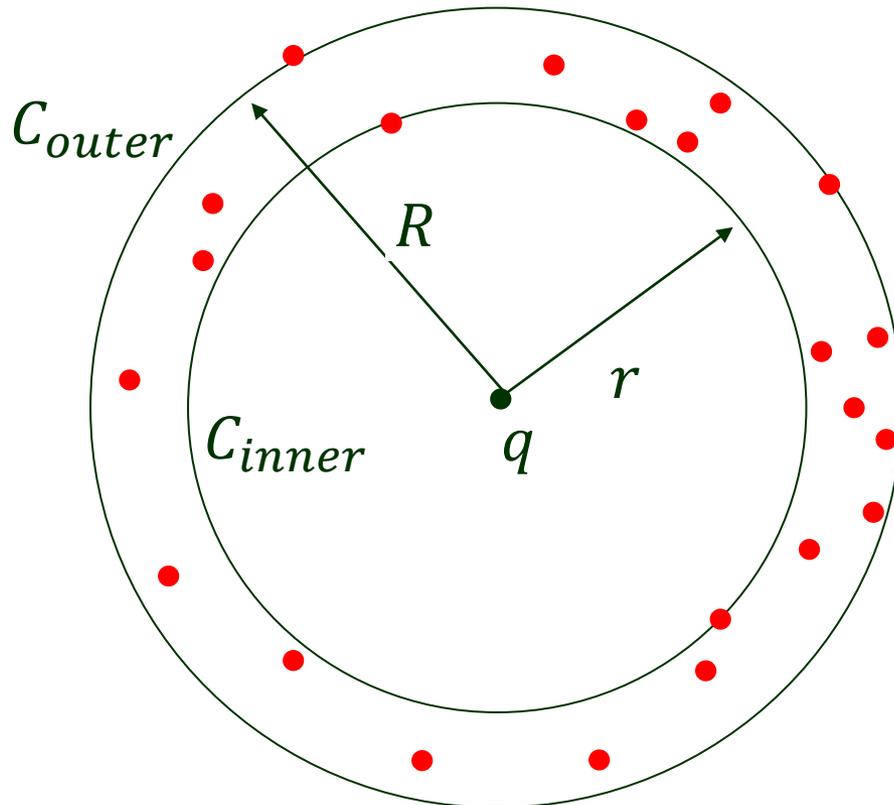
There must be one point each on  $C_{outer}$  and  $C_{inner}$ .

Otherwise, we can always reduce the size of  $C_{outer}$ , or increase that of  $C_{inner}$ .



# V. Roundness of a Point Set

The **roundness** of a set of points is measured by the *minimum width* of any annulus that contains the points.



Observation:

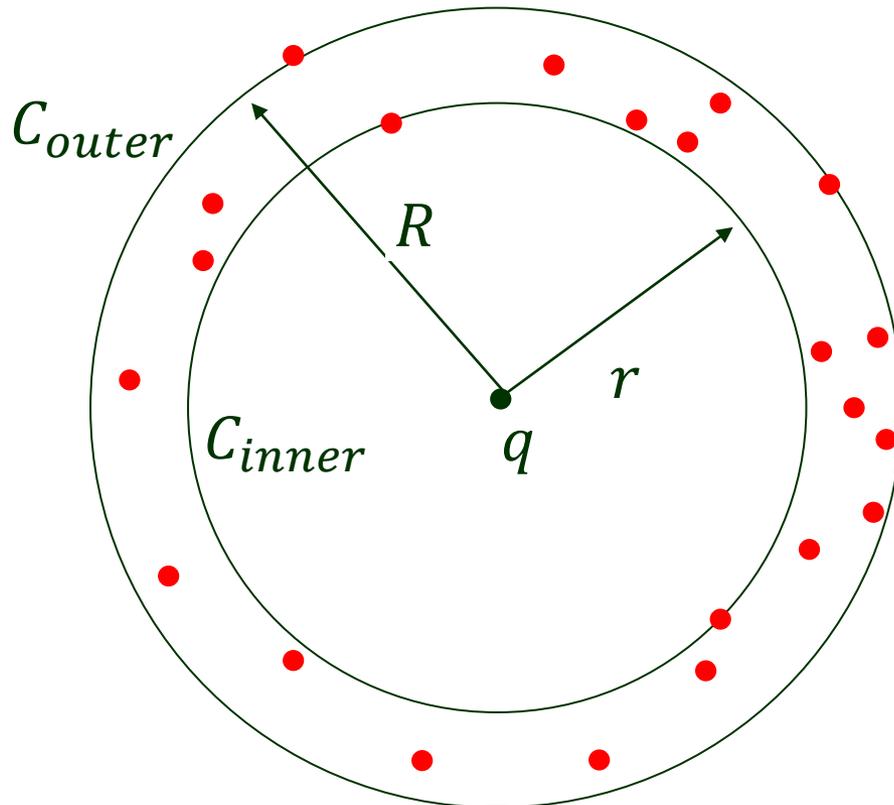
There must be one point each on  $C_{outer}$  and  $C_{inner}$ .

Otherwise, we can always reduce the size of  $C_{outer}$ , or increase that of  $C_{inner}$ .

But one point on each bounding circle does not yield the smallest-width annulus.

# V. Roundness of a Point Set

The **roundness** of a set of points is measured by the **minimum width** of any annulus that contains the points.



Observation:

There must be one point each on  $C_{outer}$  and  $C_{inner}$ .

Otherwise, we can always reduce the size of  $C_{outer}$ , or increase that of  $C_{inner}$ .

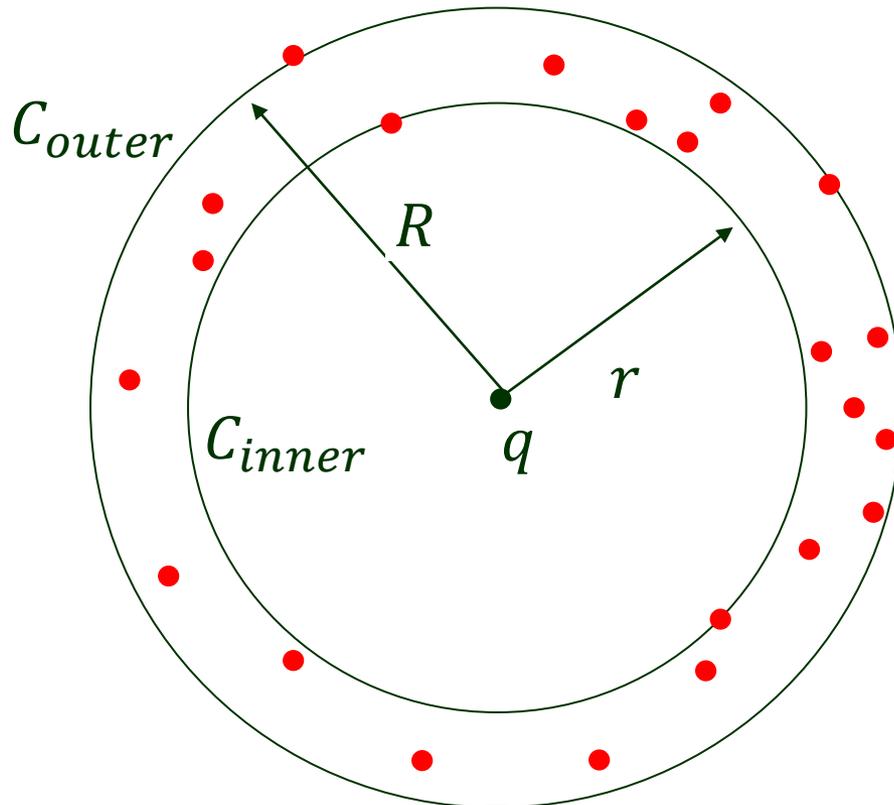
But one point on each bounding circle does not yield the smallest-width annulus.

Four degrees of freedom:

- ◆ Center  $q = (q_x, q_y)$
- ◆ Outer radius  $R$
- ◆ Inner radius  $r$

# V. Roundness of a Point Set

The **roundness** of a set of points is measured by the **minimum width** of any annulus that contains the points.



Observation:

There must be one point each on  $C_{outer}$  and  $C_{inner}$ .

Otherwise, we can always reduce the size of  $C_{outer}$ , or increase that of  $C_{inner}$ .

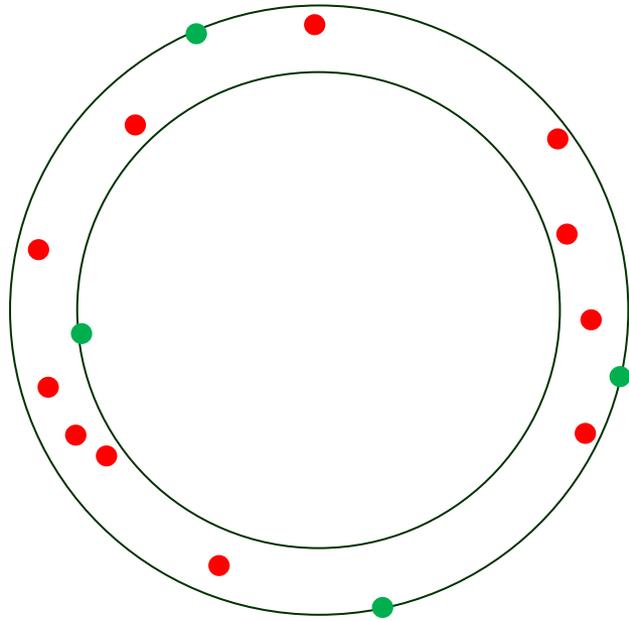
But one point on each bounding circle does not yield the smallest-width annulus.

Four degrees of freedom:

- ◆ Center  $q = (q_x, q_y)$
- ◆ Outer radius  $R$
- ◆ Inner radius  $r$     **Need 4 constraints!**

# Only Three Different Cases

---



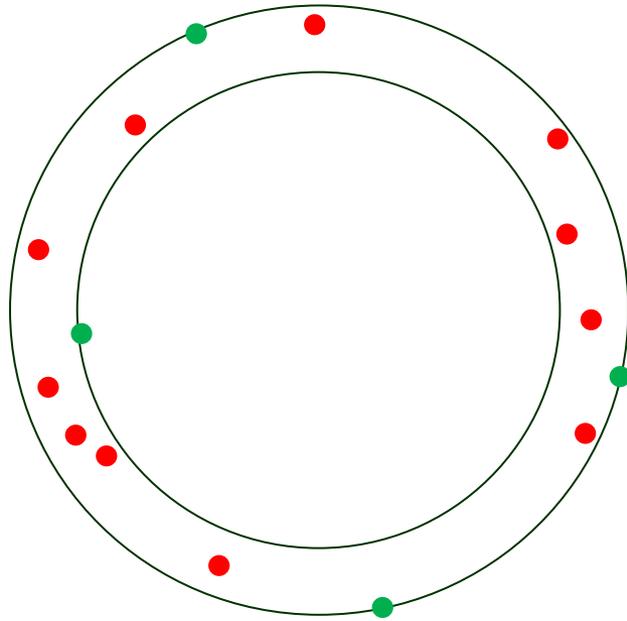
(a)

$\geq 3$  points on  $C_{outer}$

$\geq 1$  point on  $C_{inner}$

# Only Three Different Cases

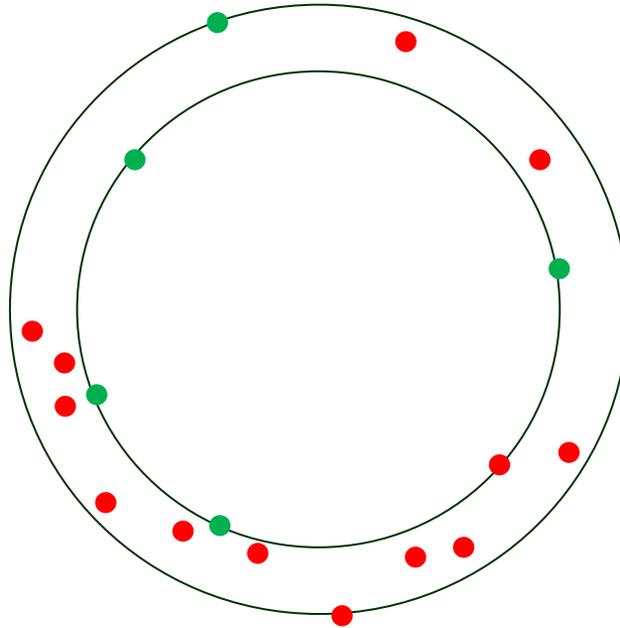
---



(a)

$\geq 3$  points on  $C_{outer}$

$\geq 1$  point on  $C_{inner}$



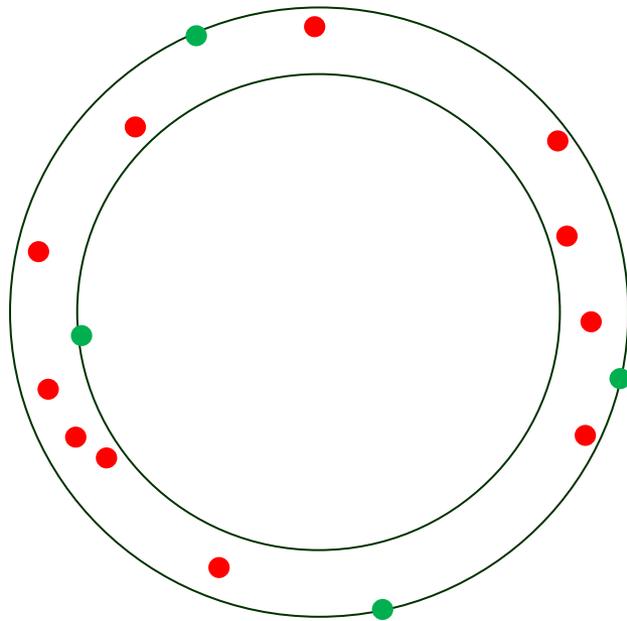
(b)

$\geq 1$  point on  $C_{outer}$

$\geq 3$  points on  $C_{inner}$

# Only Three Different Cases

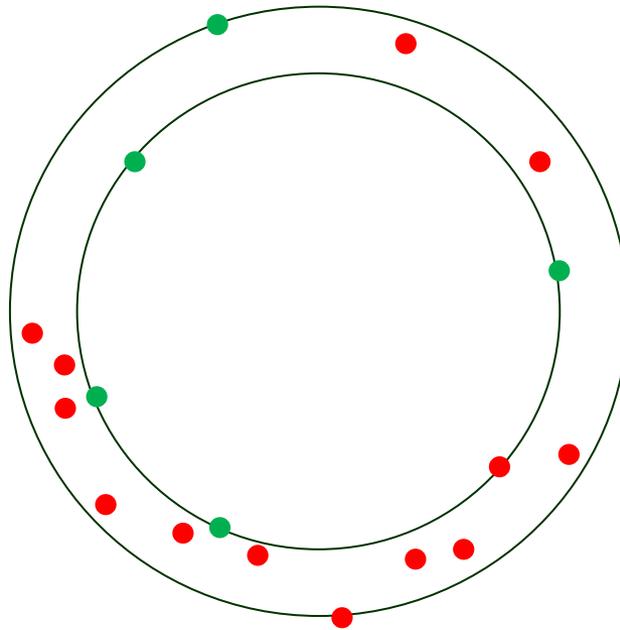
---



(a)

$\geq 3$  points on  $C_{outer}$

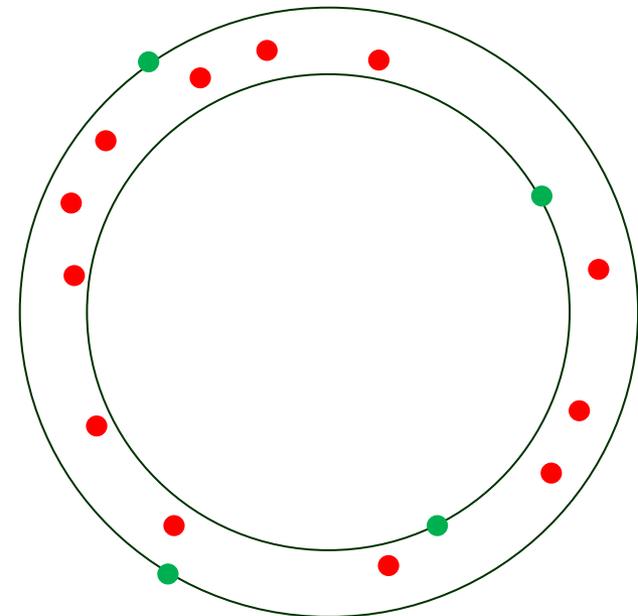
$\geq 1$  point on  $C_{inner}$



(b)

$\geq 1$  point on  $C_{outer}$

$\geq 3$  points on  $C_{inner}$



(c)

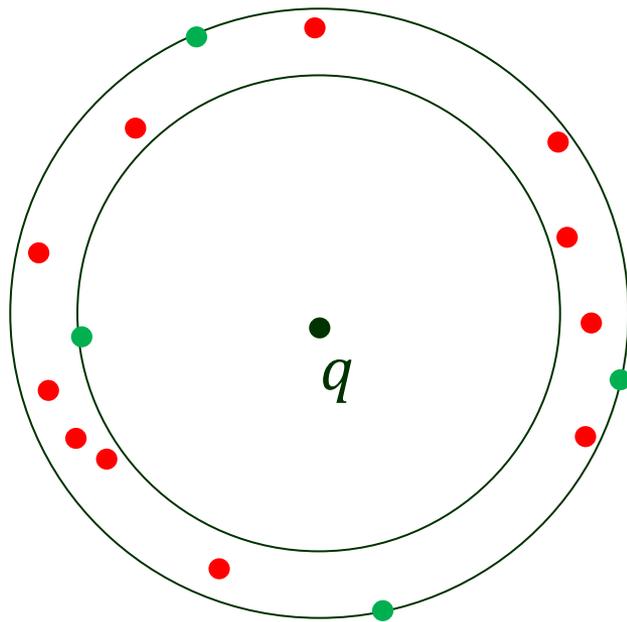
2 points on  $C_{outer}$

2 points on  $C_{inner}$

# Smallest-Width Annulus – Case (a)

---

The problem is equivalent to finding the center point  $q$  of the annulus.



In case (a)

(a)

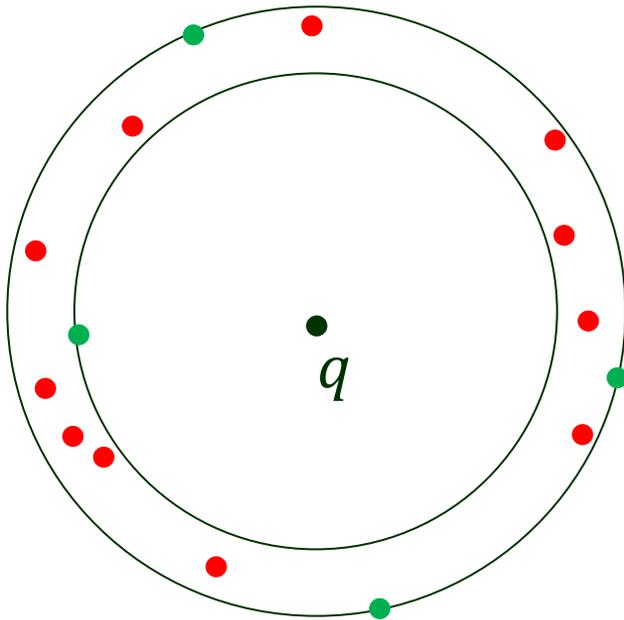
$\geq 3$  points on  $C_{outer}$

$\geq 1$  point on  $C_{inner}$

# Smallest-Width Annulus – Case (a)

---

The problem is equivalent to finding the center point  $q$  of the annulus.



In case (a)

$\geq 3$  points on  $C_{outer}$

(a)

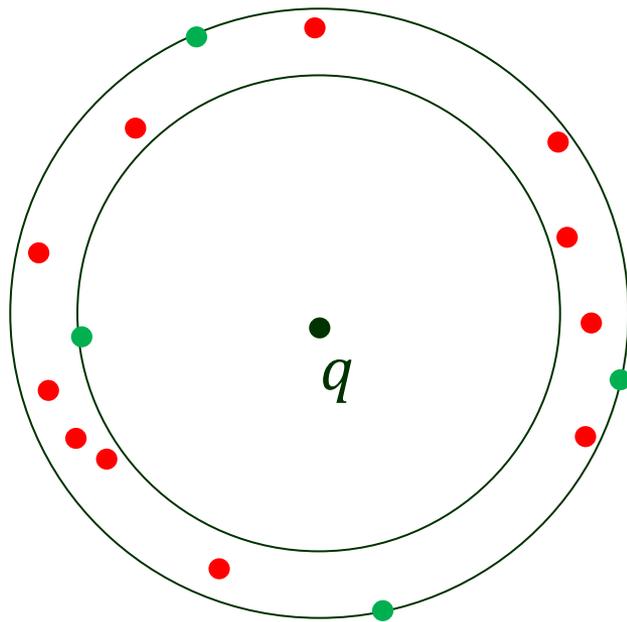
$\geq 3$  points on  $C_{outer}$

$\geq 1$  point on  $C_{inner}$

# Smallest-Width Annulus – Case (a)

---

The problem is equivalent to finding the center point  $q$  of the annulus.



(a)

$\geq 3$  points on  $C_{outer}$

$\geq 1$  point on  $C_{inner}$

In case (a)

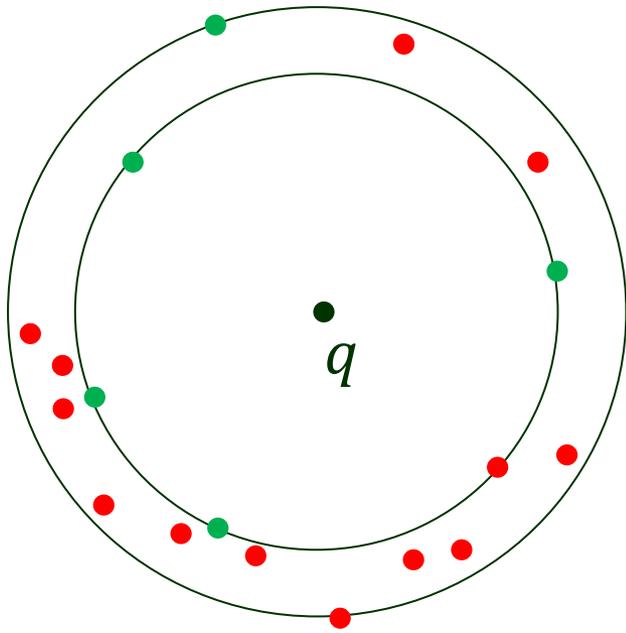
$\geq 3$  points on  $C_{outer}$



$q$  must be a vertex of the farthest-point Voronoi diagram.

# Case (b)

---



$\geq 3$  points on  $C_{inner}$

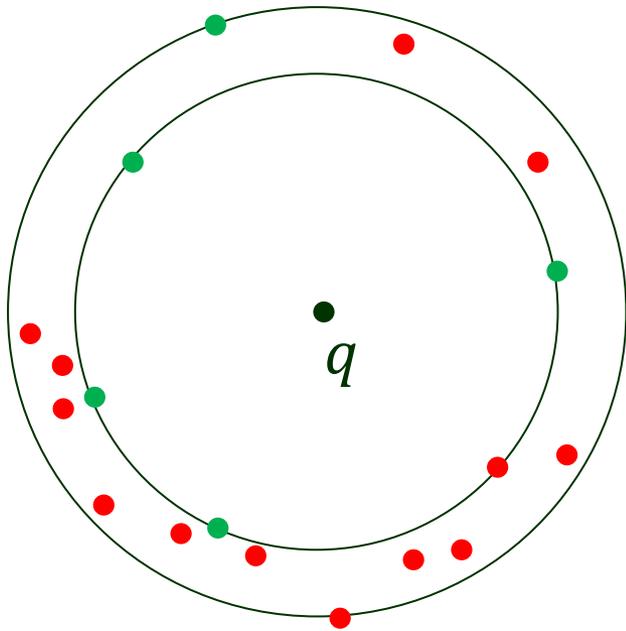
(b)

$\geq 1$  point on  $C_{outer}$

$\geq 3$  points on  $C_{inner}$

# Case (b)

---



(b)

$\geq 1$  point on  $C_{outer}$

$\geq 3$  points on  $C_{inner}$

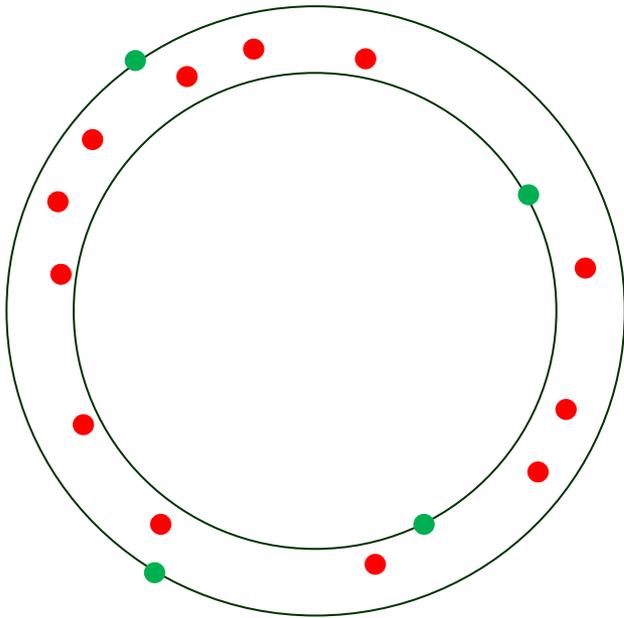
$\geq 3$  points on  $C_{inner}$



$q$  must be a vertex of the  
(nearest-point) Voronoi diagram.

# Case (c)

---



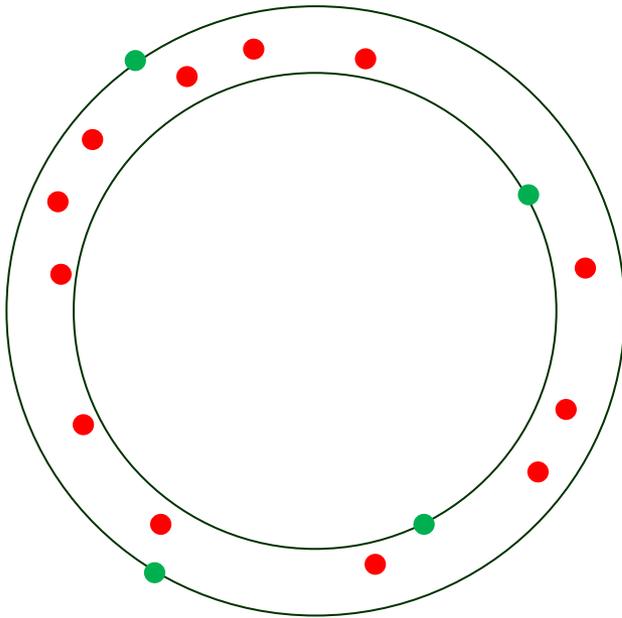
(c)

2 points on  $C_{outer}$

2 points on  $C_{inner}$

# Case (c)

---



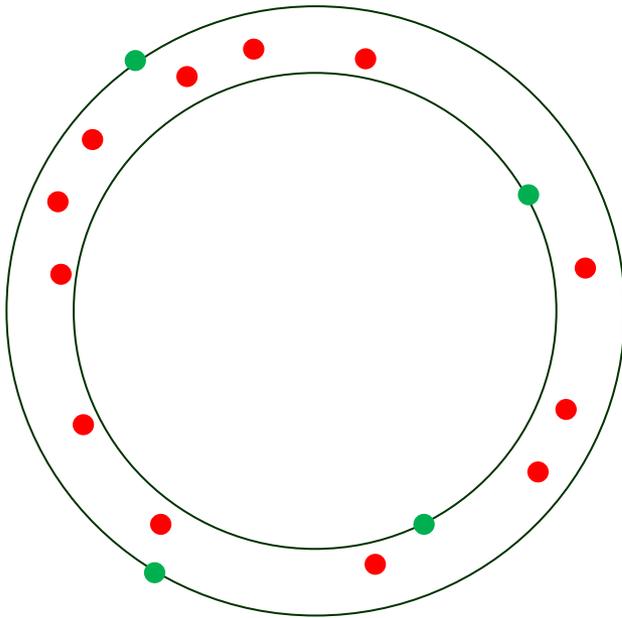
(c)

2 points on  $C_{outer}$   $\implies$   $q$  must be on an edge of the FPVD.

2 points on  $C_{inner}$

# Case (c)

---



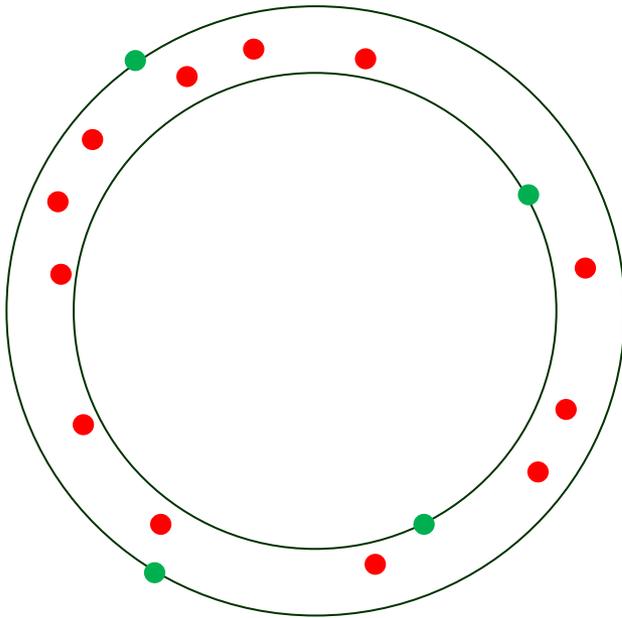
(c)

2 points on  $C_{outer}$   $\implies$   $q$  must be on an edge of the FPVD.

2 points on  $C_{inner}$   $\implies$   $q$  must be on an edge of the VD.

# Case (c)

---

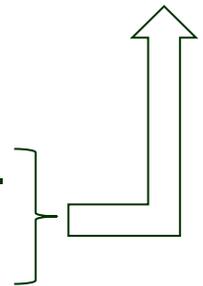


$q$  must be at the intersection of an VD edge with an FPVD edge.

(c)

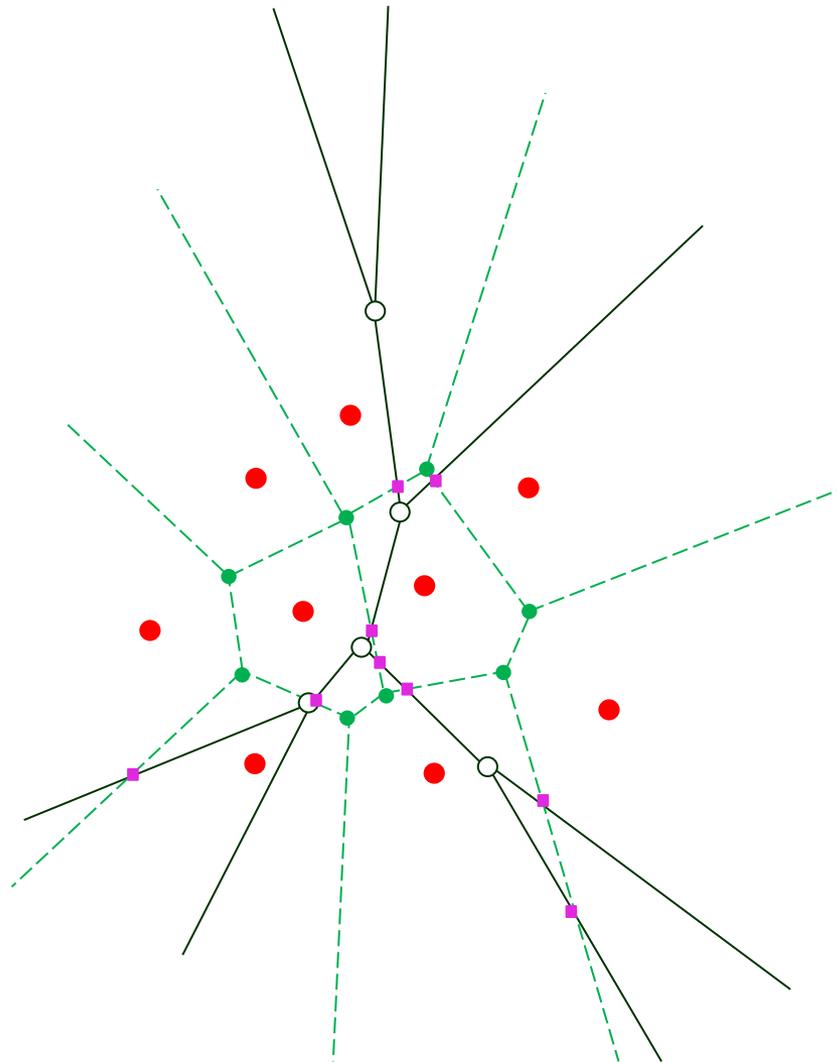
2 points on  $C_{outer}$   $\implies$   $q$  must be on an edge of the FPVD.

2 points on  $C_{inner}$   $\implies$   $q$  must be on an edge of the VD.



# Overlay of VD and FPVD

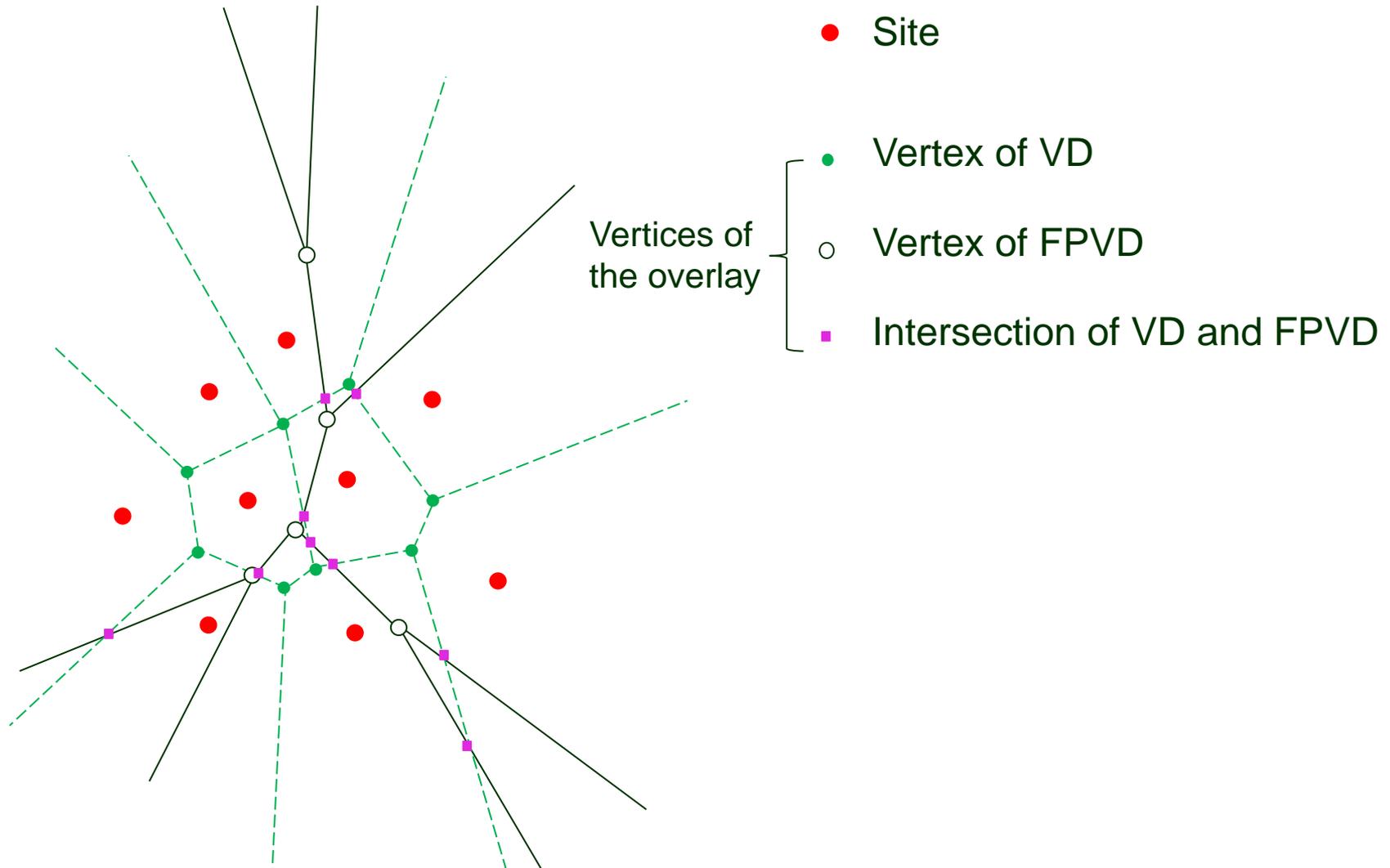
---



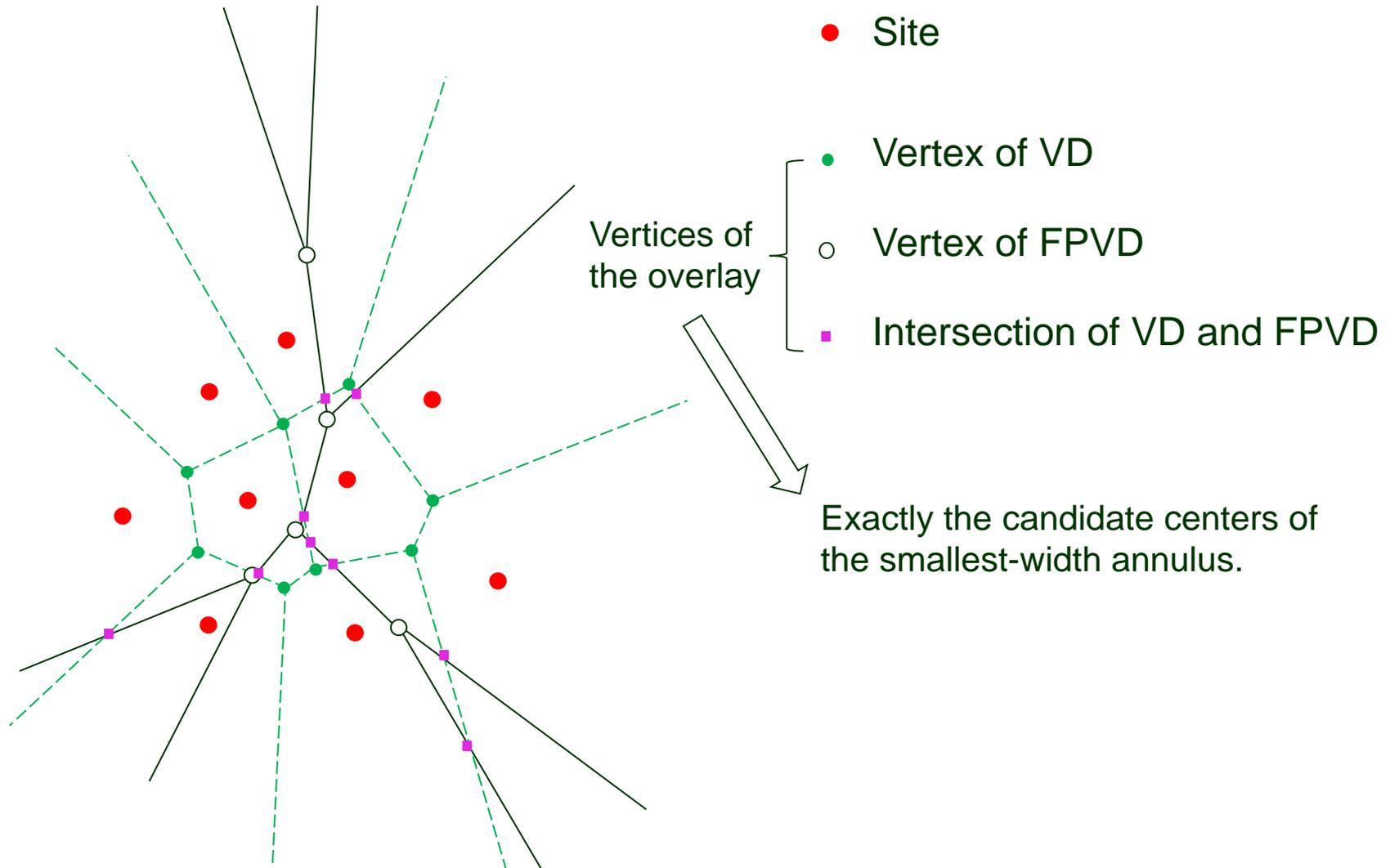
- Site
- Vertex of VD
- Vertex of FPVD
- Intersection of VD and FPVD

# Overlay of VD and FPVD

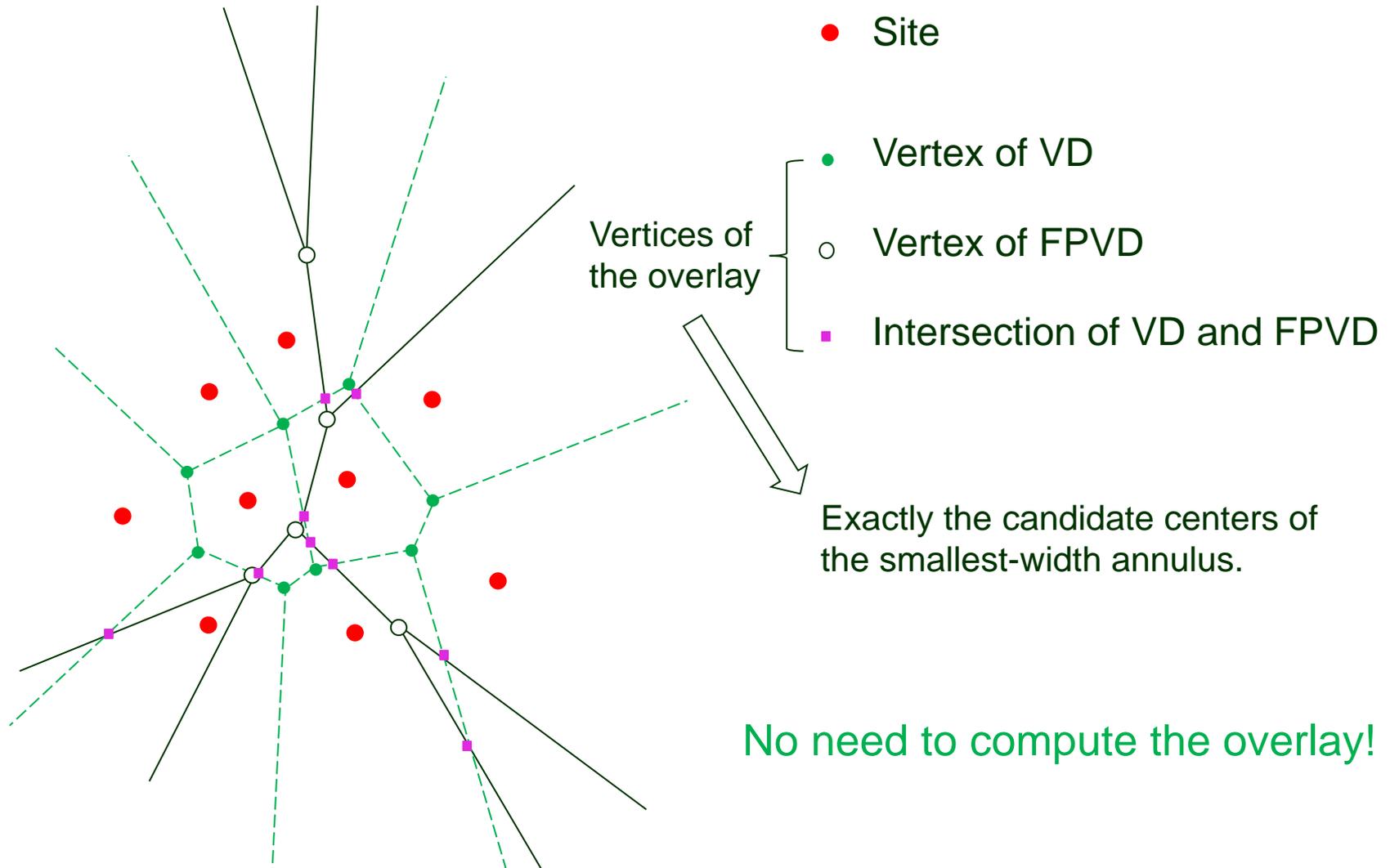
---



# Overlay of VD and FPVD



# Overlay of VD and FPVD



# Smallest-Width Annulus Algorithm

---

1. Construct the Voronoi diagram and farthest-point Voronoi diagram.

# Smallest-Width Annulus Algorithm

---

1. Construct the Voronoi diagram and farthest-point Voronoi diagram.
2. For every vertex  $v$  of the FPVD ( $O(n)$  vertices)

# Smallest-Width Annulus Algorithm

---

1. Construct the Voronoi diagram and farthest-point Voronoi diagram.
2. For every vertex  $v$  of the FPVD ( $O(n)$  vertices)
  - Its farthest sites  $p_i, p_j, p_k$  (equidistant) are known ( $C_{outer}$ ).

# Smallest-Width Annulus Algorithm

---

1. Construct the Voronoi diagram and farthest-point Voronoi diagram.
2. For every vertex  $v$  of the FPVD ( $O(n)$  vertices)
  - Its farthest sites  $p_i, p_j, p_k$  (equidistant) are known ( $C_{outer}$ ).
  - Determine its closest site  $p_l$  using VD ( $C_{inner}$  in  $O(\log n)$  time).

# Smallest-Width Annulus Algorithm

---

1. Construct the Voronoi diagram and farthest-point Voronoi diagram.
2. For every vertex  $v$  of the FPVD ( $O(n)$  vertices)
  - Its farthest sites  $p_i, p_j, p_k$  (equidistant) are known ( $C_{outer}$ ).
  - Determine its closest site  $p_l$  using VD ( $C_{inner}$  in  $O(\log n)$  time).
  - This yields a candidate annulus.

# Smallest-Width Annulus Algorithm

---

1. Construct the Voronoi diagram and farthest-point Voronoi diagram.
2. For every vertex  $v$  of the FPVD ( $O(n)$  vertices)
  - Its farthest sites  $p_i, p_j, p_k$  (equidistant) are known ( $C_{outer}$ ).
  - Determine its closest site  $p_l$  using VD ( $C_{inner}$  in  $O(\log n)$  time).
  - This yields a candidate annulus.
3. For every vertex  $v$  of the VD ( $O(n)$  vertices)

# Smallest-Width Annulus Algorithm

---

1. Construct the Voronoi diagram and farthest-point Voronoi diagram.
2. For every vertex  $v$  of the FPVD ( $O(n)$  vertices)
  - Its farthest sites  $p_i, p_j, p_k$  (equidistant) are known ( $C_{outer}$ ).
  - Determine its closest site  $p_l$  using VD ( $C_{inner}$  in  $O(\log n)$  time).
  - This yields a candidate annulus.
3. For every vertex  $v$  of the VD ( $O(n)$  vertices)
  - Its closest sites  $p_i, p_j, p_k$  (equidistant) are known ( $C_{inner}$ ).
  - Determine its farthest site  $p_l$  using FPVD ( $C_{outer}$  in  $O(\log n)$  time).
  - This yields a candidate annulus.

# Algorithm (cont'd)

---

4. For every pair of edges, one from VD and the other from FPVD ( $O(n^2)$  pairs)

# Algorithm (cont'd)

---

4. For every pair of edges, one from VD and the other from FPVD ( $O(n^2)$  pairs)
  - Test if they intersect.

# Algorithm (cont'd)

---

4. For every pair of edges, one from VD and the other from FPVD ( $O(n^2)$  pairs)
  - Test if they intersect.
  - If so, the two closest sites  $p_i, p_j$  and two farthest sites  $p_k, p_l$  are known. Construct the annulus in  $O(1)$  time.

# Algorithm (cont'd)

---

4. For every pair of edges, one from VD and the other from FPVD ( $O(n^2)$  pairs)
  - Test if they intersect.
  - If so, the two closest sites  $p_i, p_j$  and two farthest sites  $p_k, p_l$  are known. Construct the annulus in  $O(1)$  time.
5. Choose the smallest-width annulus of all constructed annuli.

# Algorithm (cont'd)

---

4. For every pair of edges, one from VD and the other from FPVD ( $O(n^2)$  pairs)
  - Test if they intersect.
  - If so, the two closest sites  $p_i, p_j$  and two farthest sites  $p_k, p_l$  are known. Construct the annulus in  $O(1)$  time.
5. Choose the smallest-width annulus of all constructed annuli.

**Theorem** Given a set of  $n$  points in the plane, the smallest-width annulus can be determined in  $O(n^2)$  time using  $O(n)$  storage.