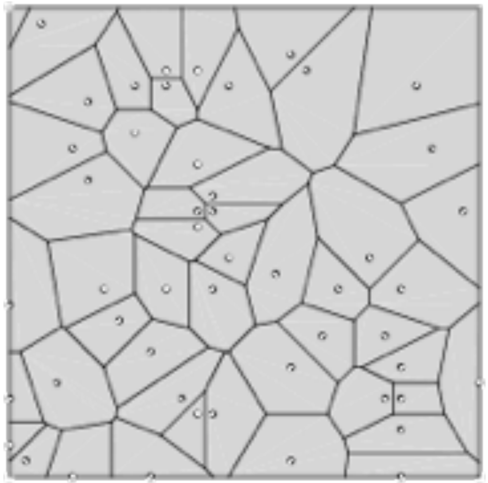
Voronoi Diagram



Sasanka Roy

Chennai Mathematical Institute

Organization of the Talk

Organization of the Talk

1. Preliminaries
2. Generic Definition
3. Some Technical Details
4. Conclusion

Organization of the Talk

1. Preliminaries

2. Generic Definition

3. Some Technical Details

4. Conclusion

What are we going to talk about?

What are we going to talk about?

We have some data

What are we going to talk about?

We have some data

Geometric Data

What are we going to talk about?

We have some data

Geometric Data ????

Geometric Data

What are we going to talk about?

We have some data

Geometric Data ????

Geometric Data

What do I mean ????

What are we going to talk about?

We have some data

Geometric Data ????

I mean: we have

Geometric Data

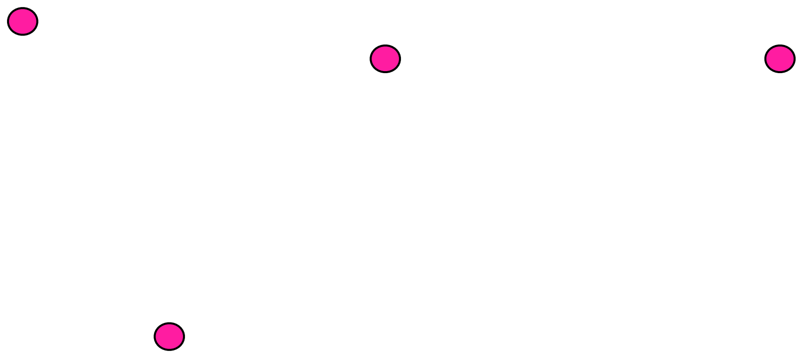
What do I mean ????

What are we going to talk about?

We have some data

Geometric Data ????

I mean: we have points,



Geometric Data

What do I mean ????

What are we going to talk about?

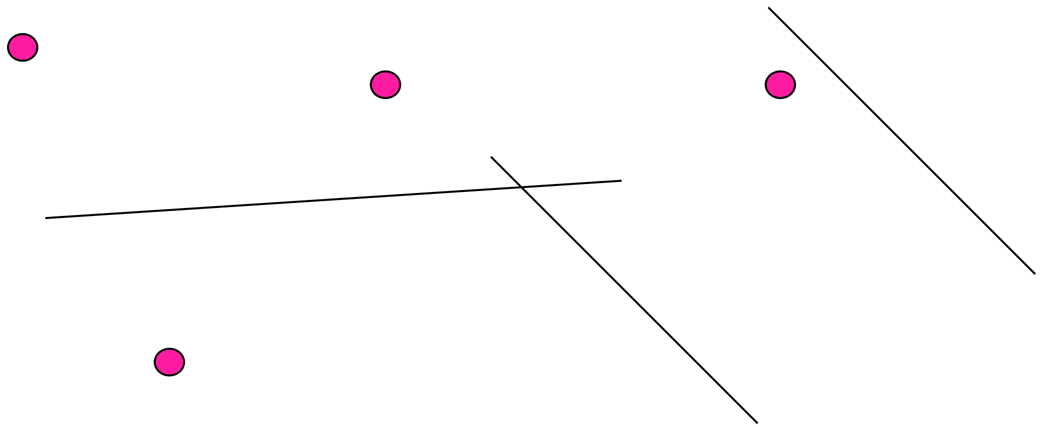
We have some data

Geometric Data ????

Geometric Data

What do I mean ????

I mean: we have points, line segments,



What are we going to talk about?

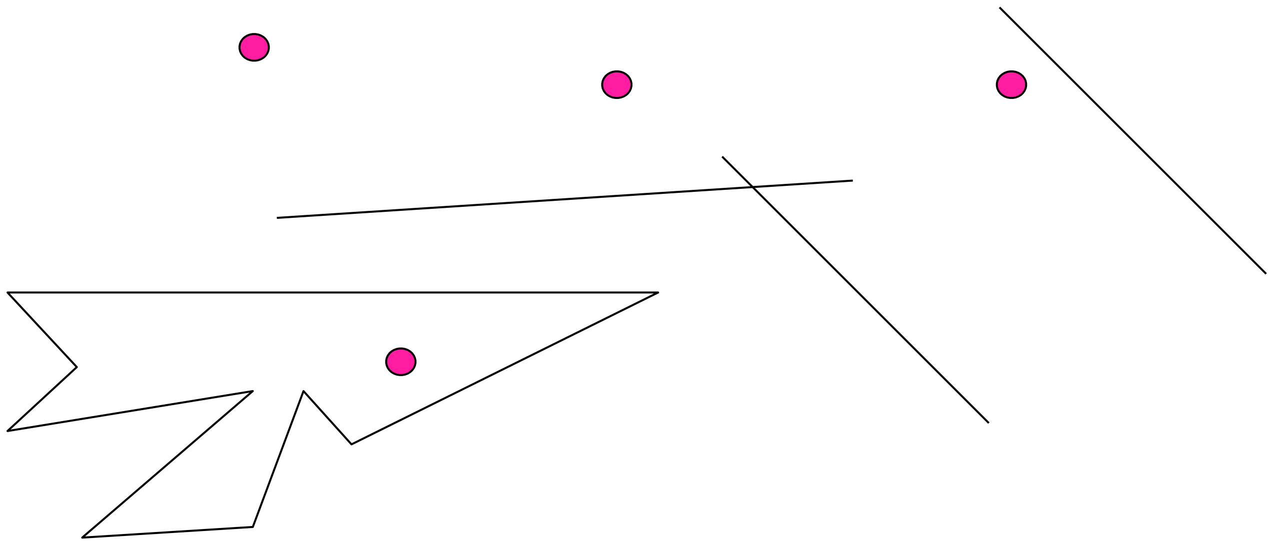
We have some data

Geometric Data ????

Geometric Data

What do I mean ????

I mean: we have points, line segments, polygons etc.



What are we going to talk about?

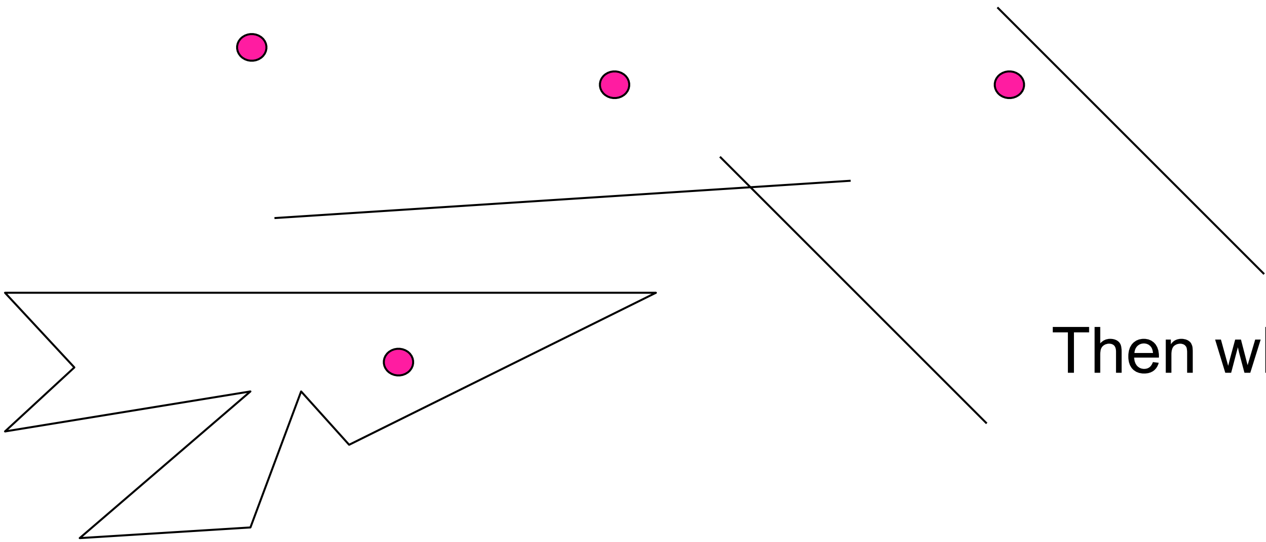
We have some data

Geometric Data ????

Geometric Data

What do I mean ????

I mean: we have points, line segments, polygons etc.



Then what?????

What are we going to talk about?

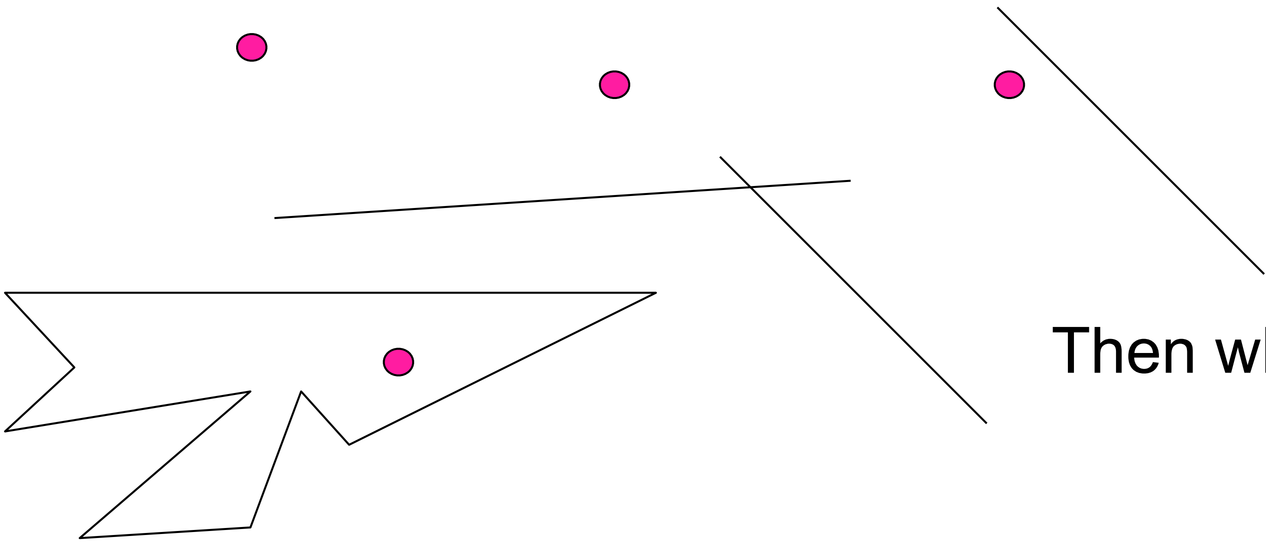
We have some data

Geometric Data ????

Geometric Data

What do I mean ????

I mean: we have points, line segments, polygons etc.



Then what?????

We want to get answers to the specific questions

What are we going to talk about?

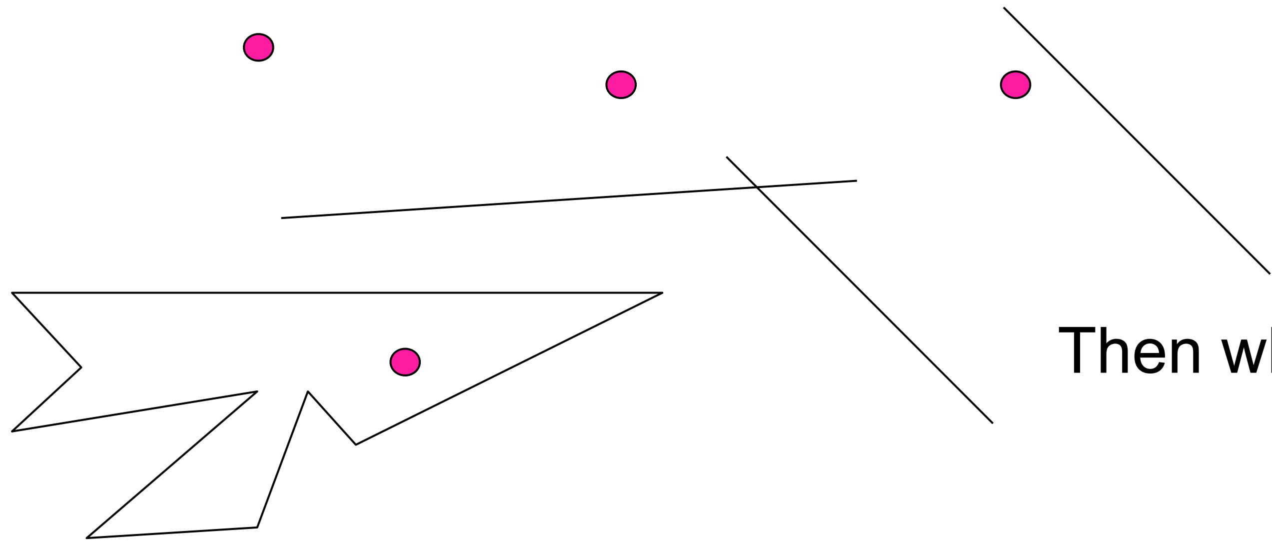
We have some data

Geometric Data ????

Geometric Data

What do I mean ????

I mean: we have points, line segments, polygons etc.



Then what?????

We want to get answers to the specific questions

Closest points to the line segments

What are we going to talk about?

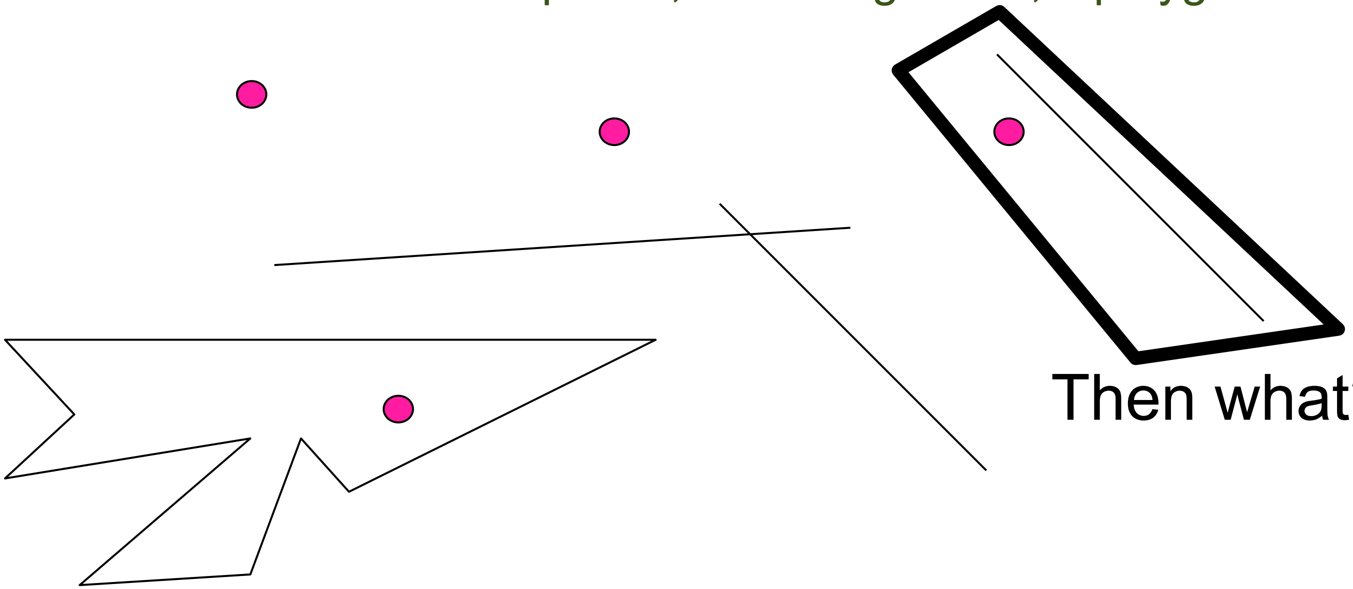
We have some data

Geometric Data ????

Geometric Data

What do I mean ????

I mean: we have points, line segments, polygons etc.



Then what?????

We want to get answers to the specific questions

Closest points to the line segments

What are we going to talk about?

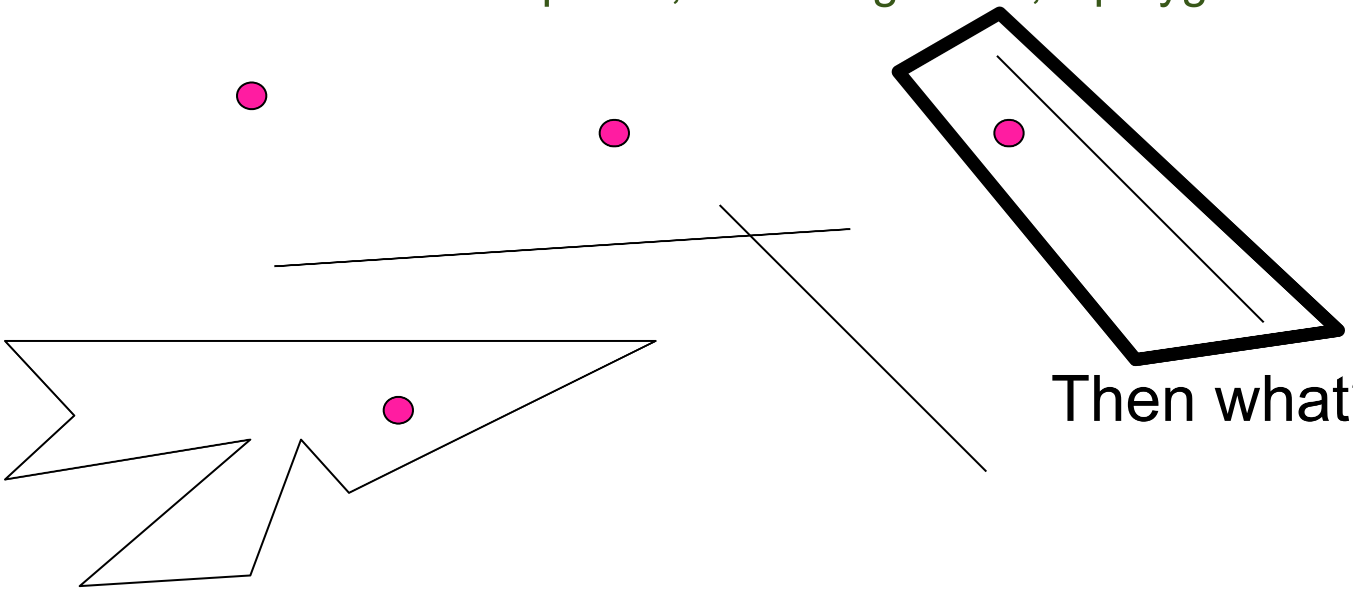
We have some data

Geometric Data ????

Geometric Data

What do I mean ????

I mean: we have points, line segments, polygons etc.



Then what?????

We want to get answers to the specific questions

Closest points to the line segments

Point inside the simple polygon

What are we going to talk about?

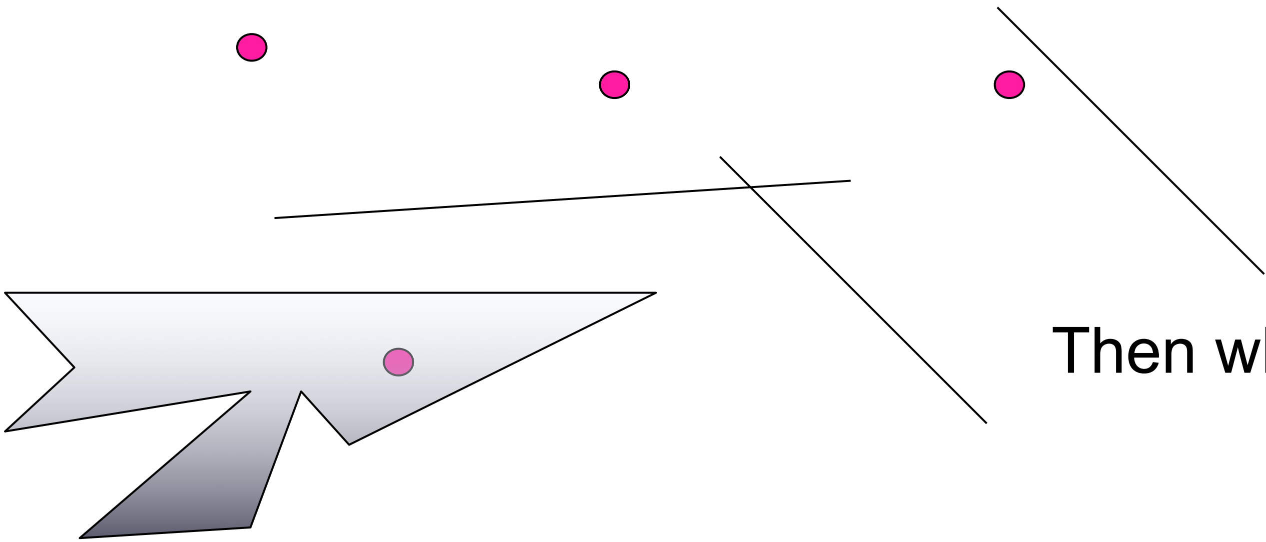
We have some data

Geometric Data ????

Geometric Data

What do I mean ????

I mean: we have points, line segments, polygons etc.



Then what?????

We want to get answers to the specific questions

Closest points to the line segments

Point inside the simple polygon

What are we going to talk about?

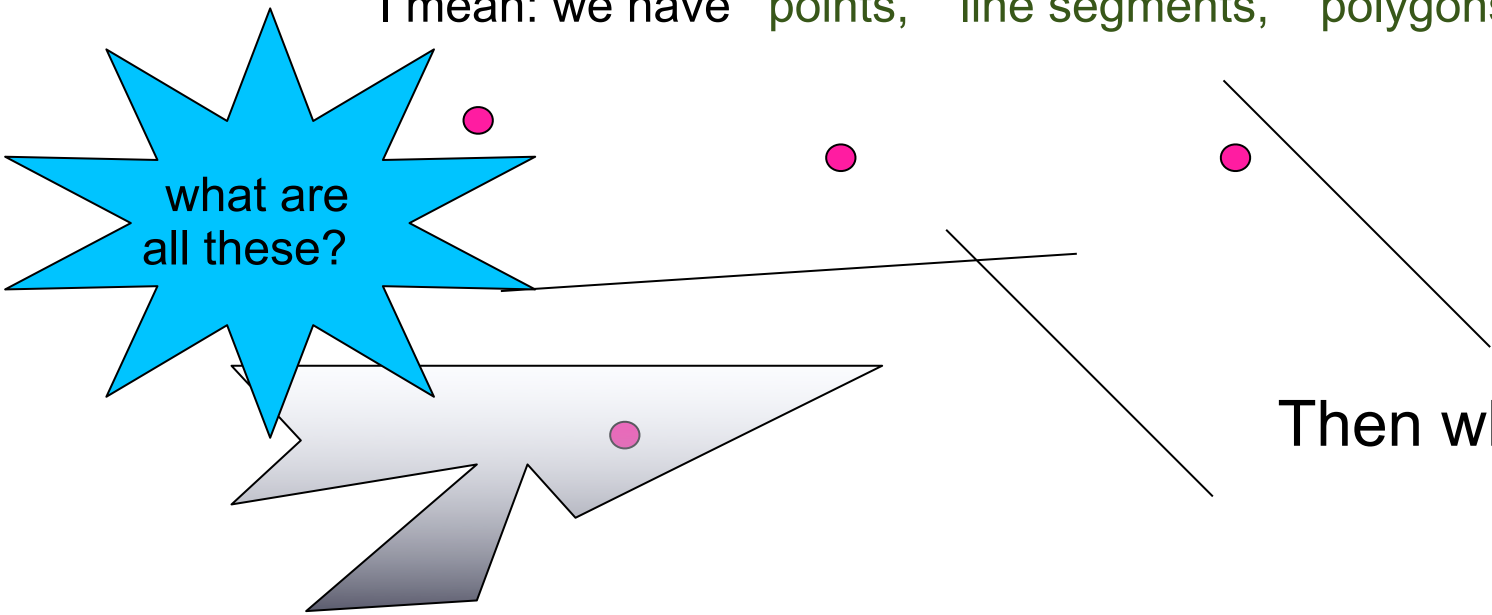
We have some data

Geometric Data ????

Geometric Data

What do I mean ????

I mean: we have points, line segments, polygons etc.



Then what?????

We want to get answers to the specific questions

Closest points to the line segments

Point inside the simple polygon

Can you be a bit Practical??

Planar Point Location

Planar Point Location

Which state has the site/point with

Latitude= 11° 0' 0" N

Longitude= 77° 0' 0" E

Planar Point Location

Which state has the site/point with

Latitude= 11° 0' 0" N

Longitude= 77° 0' 0" E

Coimbatore in Tamil Nadu

Planar Point Location

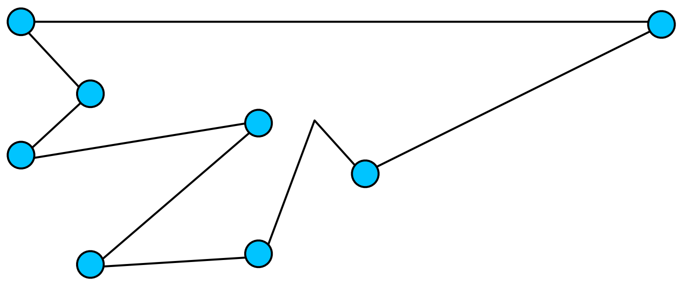
Which state has the site/point with

Latitude= 11° 0' 0" N

Longitude= 77° 0' 0" E

Coimbatore in Tamil Nadu

Can we view States as simple polygon?



Planar Point Location

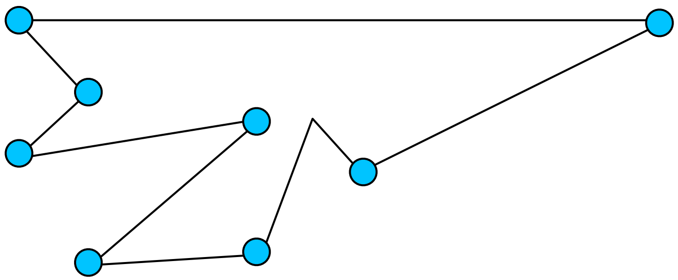
Which state has the site/point with

Latitude= 11° 0' 0" N

Longitude= 77° 0' 0" E

Coimbatore in Tamil Nadu

Can we view States as simple polygon?



simple polygon: Closed region whose boundary is formed by non-intersecting line segments

Planar Point Location

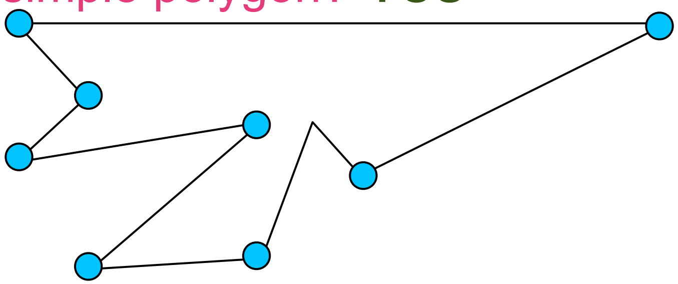
Which state has the site/point with

Latitude= 11° 0' 0" N

Longitude= 77° 0' 0" E

Coimbatore in Tamil Nadu

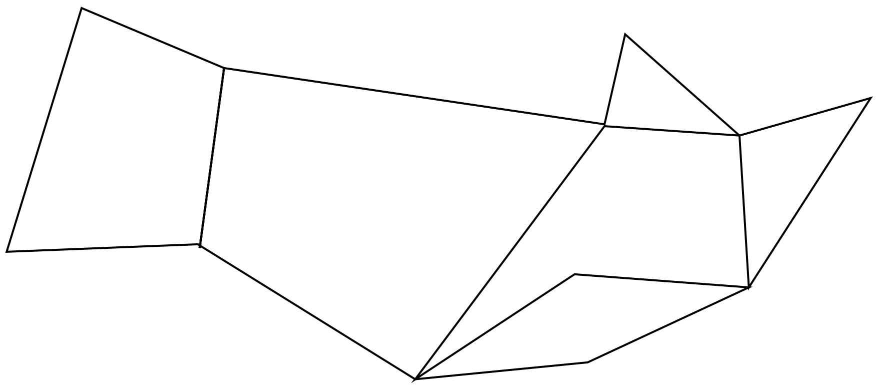
Can we view States as simple polygon? Yes



simple polygon: Closed region whose boundary is formed by non-intersecting line segments

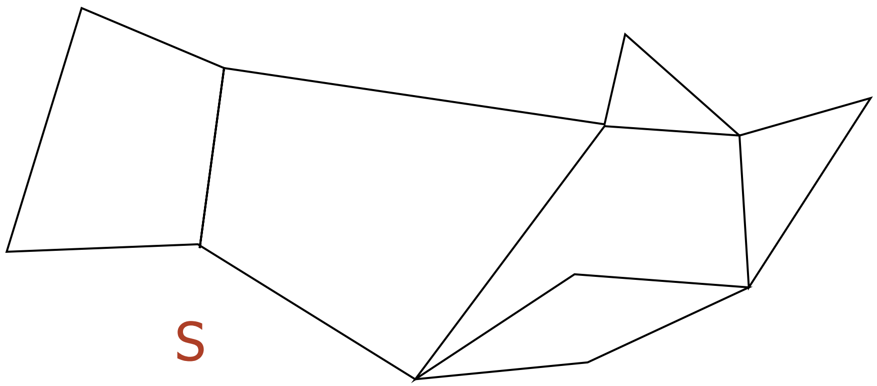
Formally Planar Point Location

Given a planar subdivision S of O(n) vertices/faces/edges



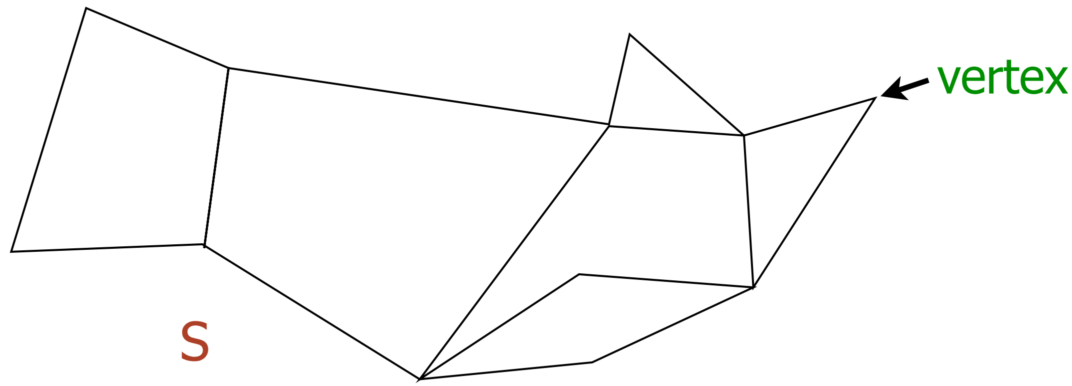
Formally Planar Point Location

Given a planar subdivision S of O(n) vertices/faces/edges



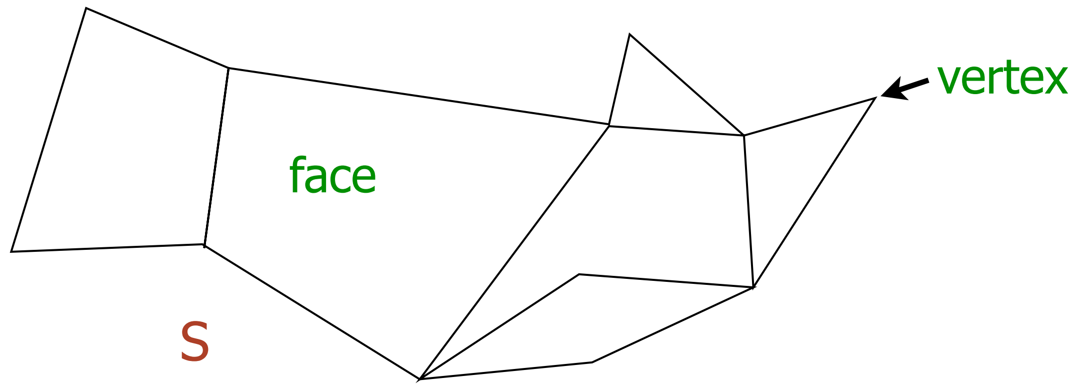
Formally Planar Point Location

Given a planar subdivision S of O(n) vertices/faces/edges



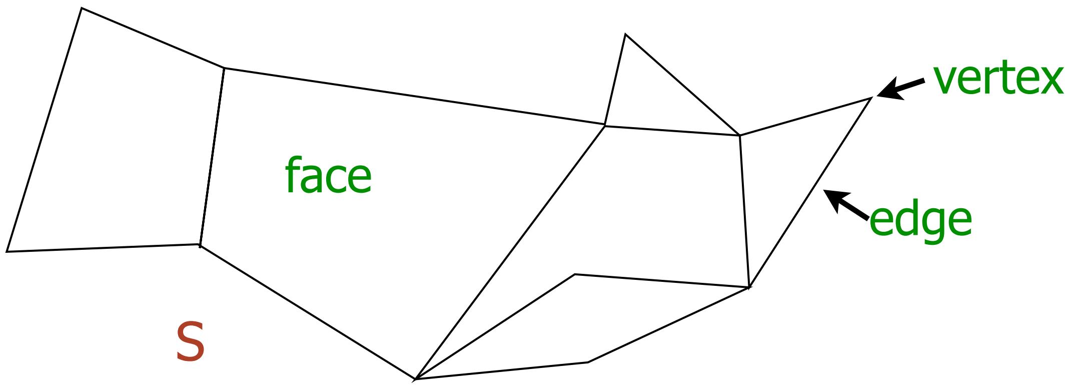
Formally Planar Point Location

Given a planar subdivision S of O(n) vertices/faces/edges



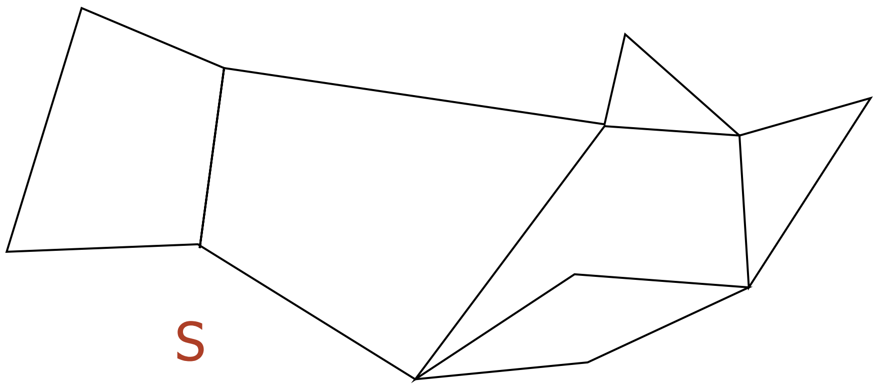
Formally Planar Point Location

Given a planar subdivision S of O(n) vertices/faces/edges



Formally Planar Point Location

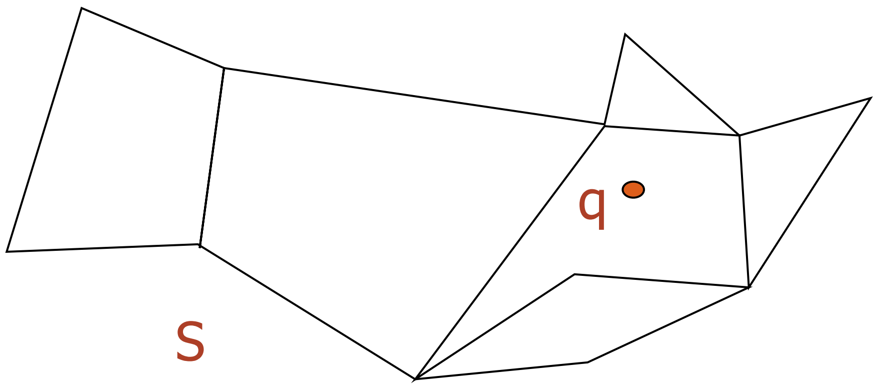
Given a planar subdivision S



Preprocess S such that:

Formally Planar Point Location

Given a planar subdivision S

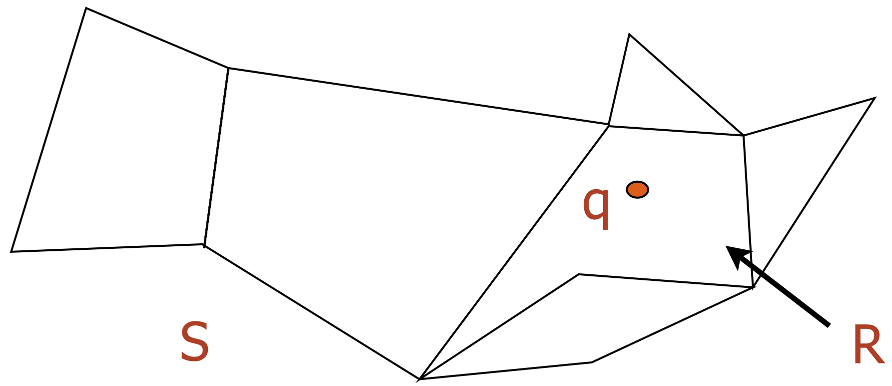


Preprocess S such that:

For any query point q,

Formally Planar Point Location

Given a planar subdivision S



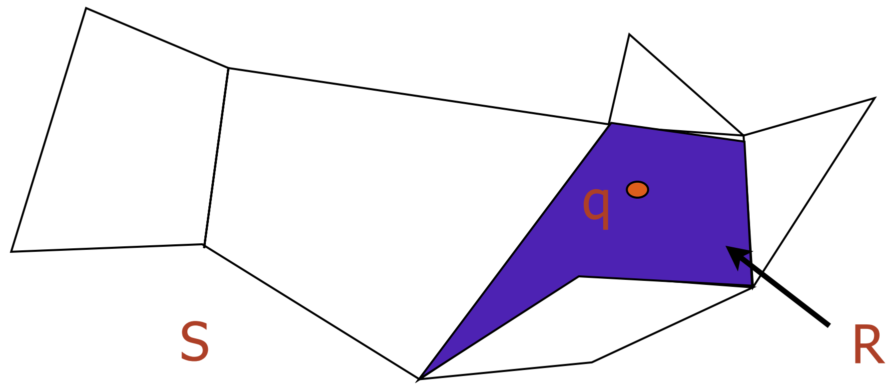
Preprocess S such that:

For any query point q

The region/face R containing q can be reported efficiently.

Formally Planar Point Location

Given a planar subdivision S

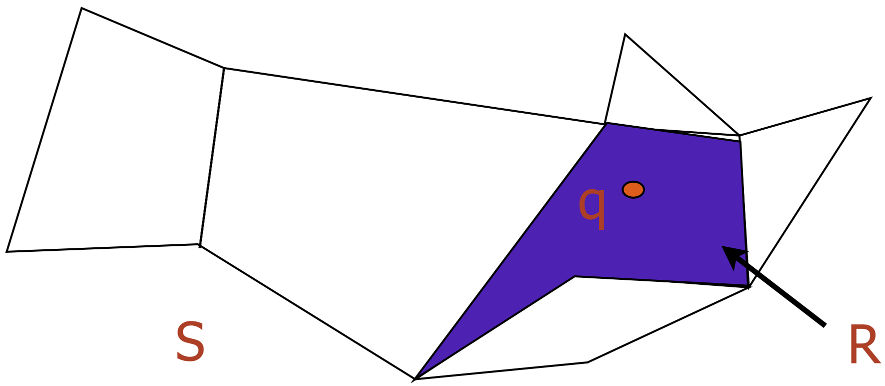


Preprocess S such that:

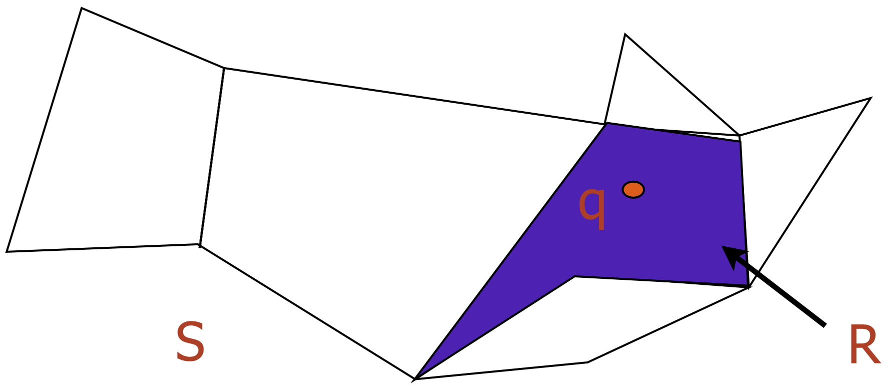
For any query point q

The region/face R containing q can be reported efficiently.

Formally Planar Point Location

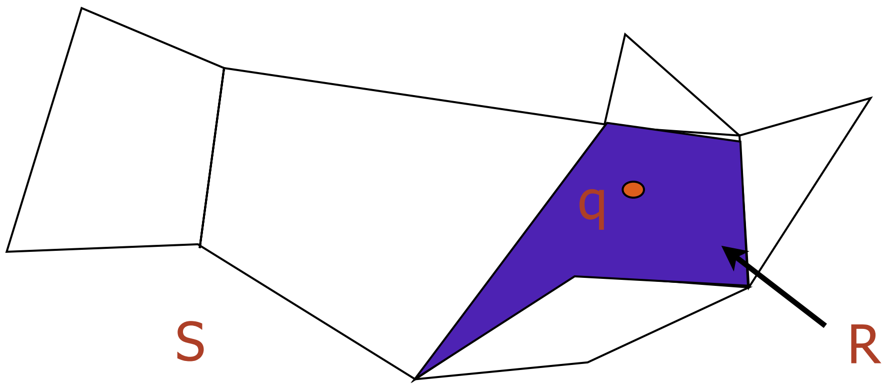


Formally Planar Point Location



Preprocessing Time:

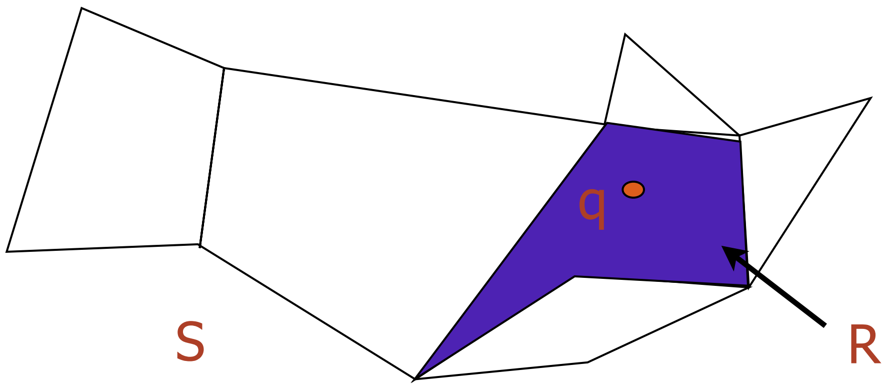
Questions?



Preprocessing Time:

Preprocessing space requirement:

Questions?

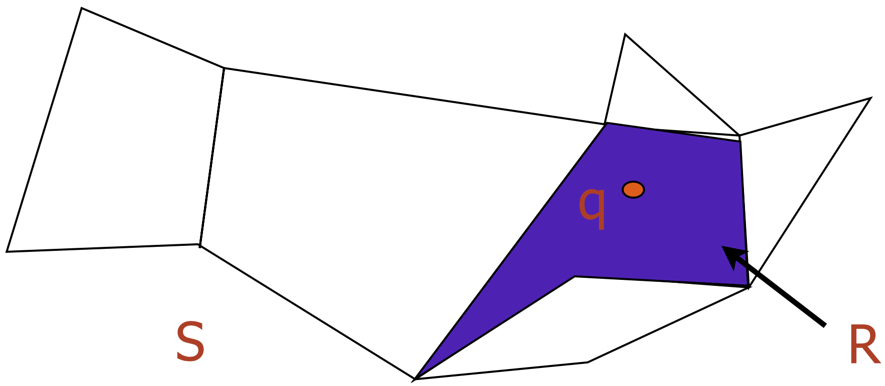


Preprocessing Time:

Preprocessing space requirement:

Query Time:

Questions?

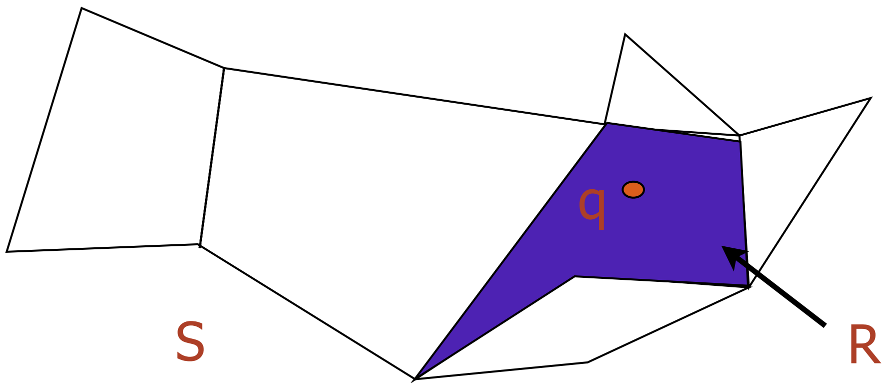


Preprocessing Time: O(n log n)

Preprocessing space requirement:

Query Time:

Questions?

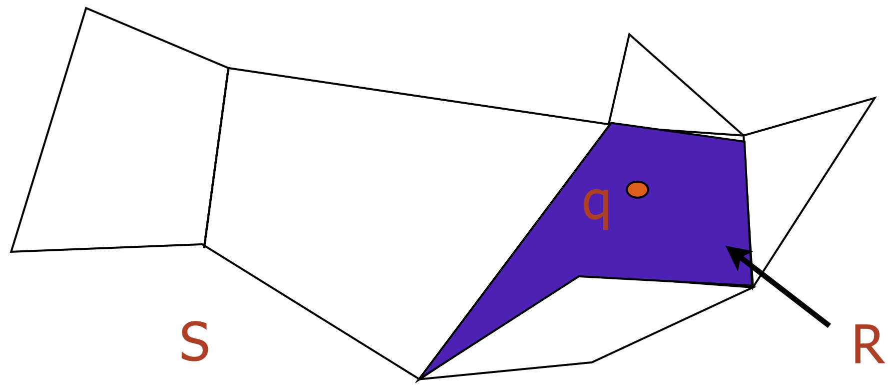


Preprocessing Time: O(n log n)

Preprocessing space requirement: O(n)

Query Time:

Questions?

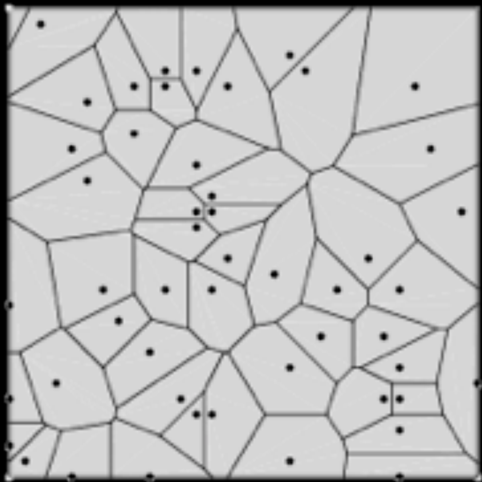


Preprocessing Time: O(n log n)

Preprocessing space requirement: O(n)

Query Time: O(log n)

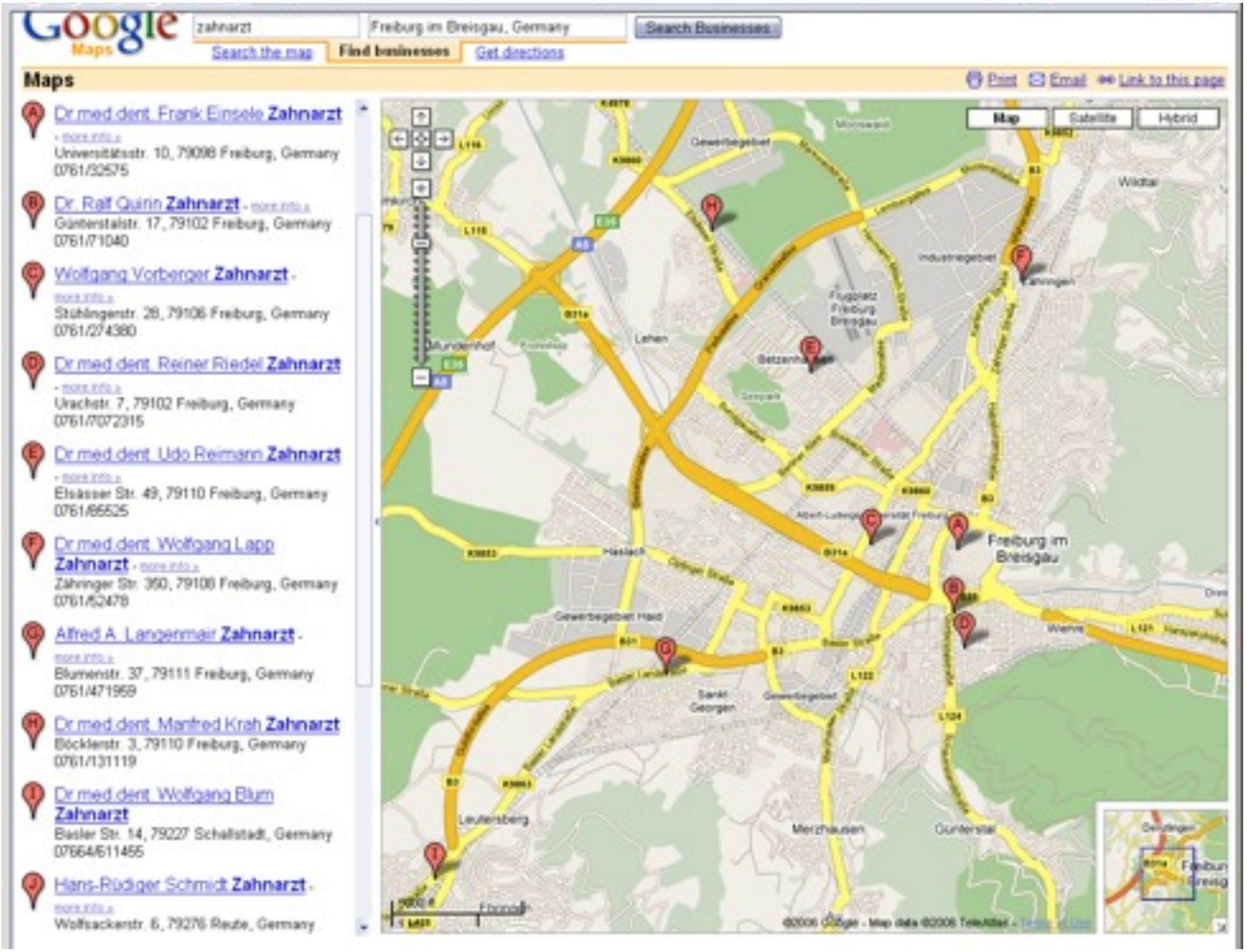
Back to Voronoi Diagram



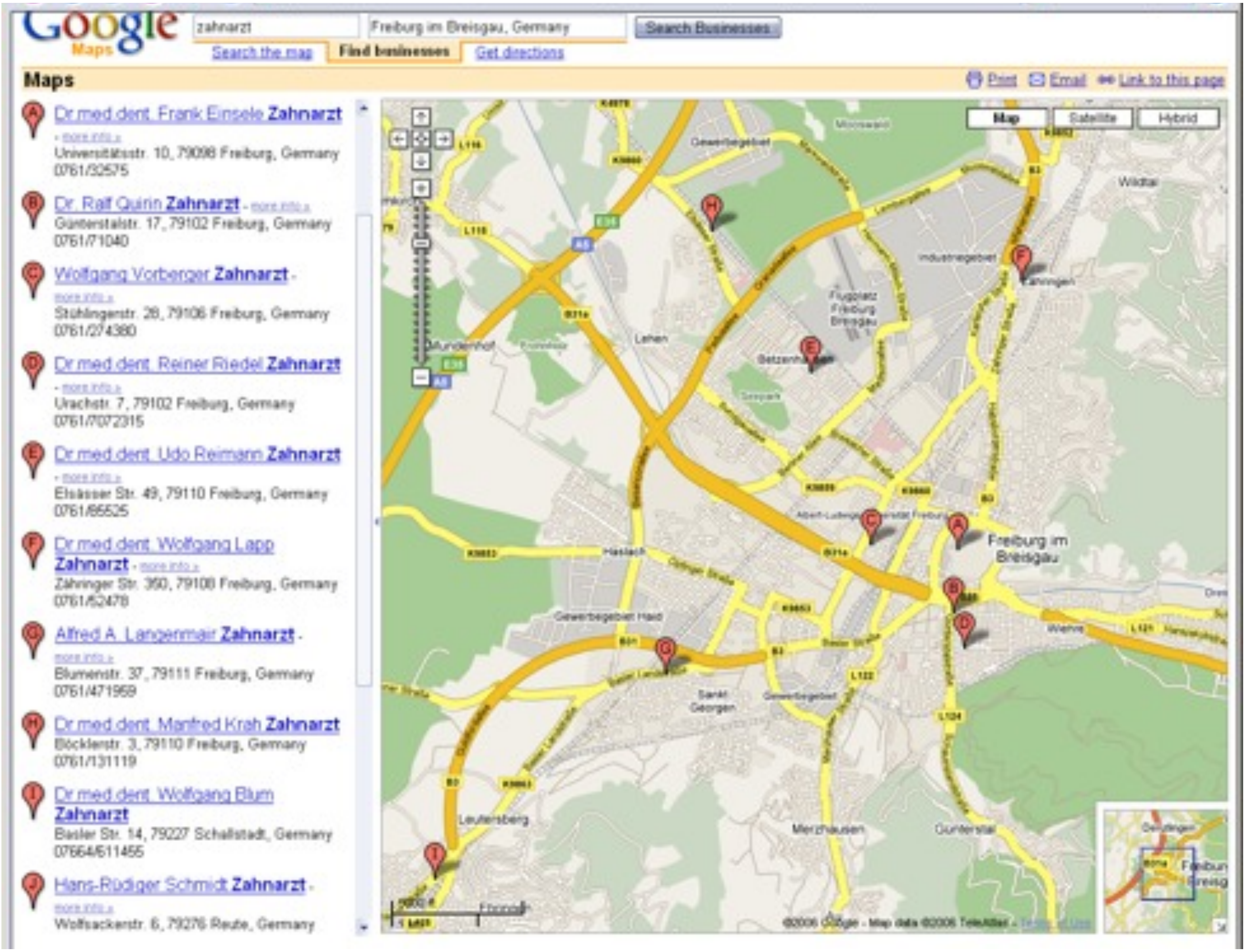
Organization of the Talk

1. Preliminaries
2. Generic Definition
3. Some Technical Details
4. Conclusion

Thank you Google

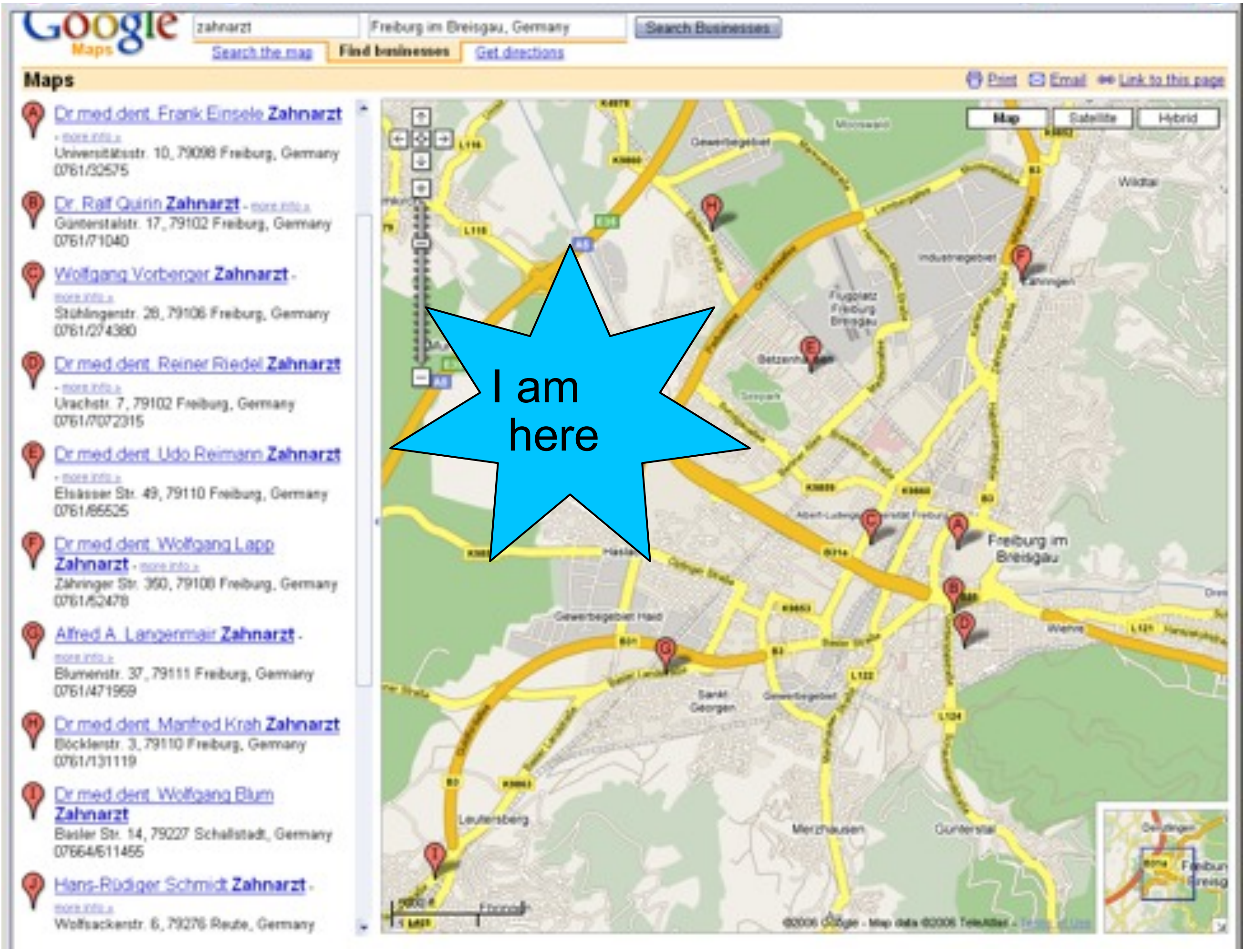


Thank you Google



Viewpoint 1: Locate the nearest dentistry.
Viewpoint 2: Find the 'service area' of potential customers for each dentist.

Thank you Google



Viewpoint 1: Locate the nearest dentistry.

Viewpoint 2: Find the ‘service area’ of potential customers for each dentist.

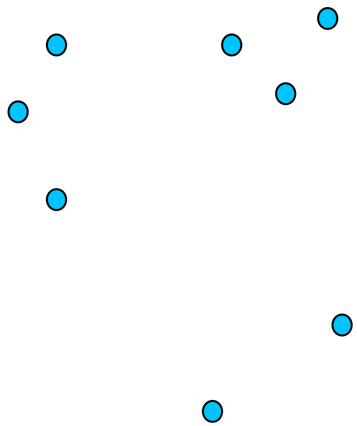
Formal Definition

Formal Definition

P $\rightarrow$ A set of n distinct points (Geometric Objects) in the plane.

Formal Definition

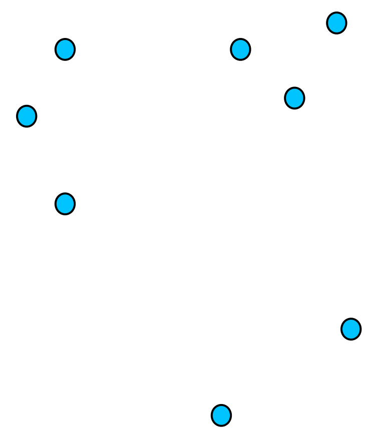
P → A set of *n* distinct points (Geometric Objects) in the plane.



Formal Definition

P $\rightarrow$ A set of *n* distinct points (Geometric Objects) in the plane.

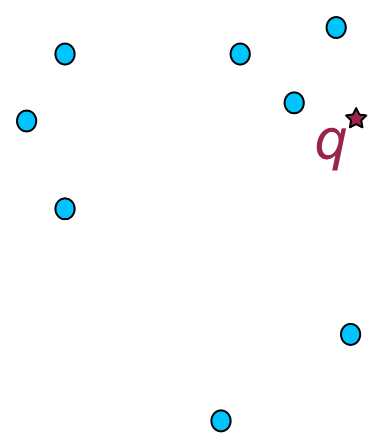
Preprocess P such that closest point $x \in P$ of any query point q can be found efficiently



Formal Definition

P $\rightarrow$ A set of *n* distinct points (Geometric Objects) in the plane.

Preprocess P such that closest point x ∈ P of any query point q can be found efficiently

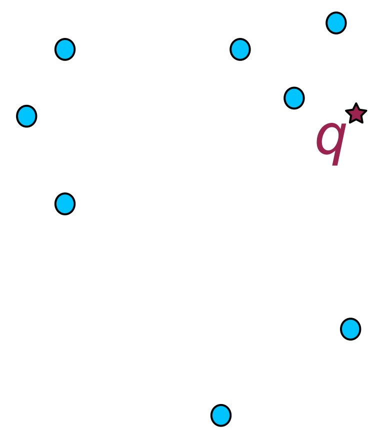


Formal Definition

P → A set of n distinct points (Geometric Objects) in the plane.

Preprocess P such that closest point $x \in P$ of any query point q can be found efficiently

How to solve this efficiently?



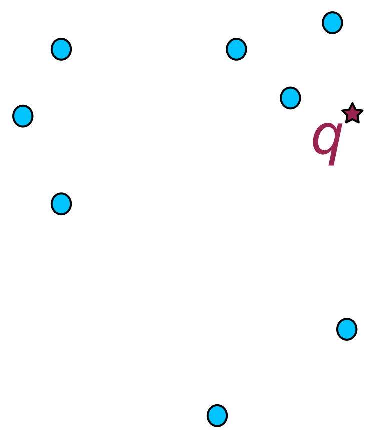
Formal Definition

P $\rightarrow$ A set of n distinct points (Geometric Objects) in the plane.

Preprocess P such that closest point $x \in P$ of any query point q can be found efficiently

How to solve this efficiently?

Subdivision of the plane into n cells such that



Formal Definition

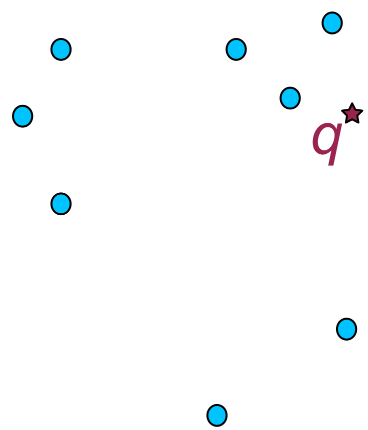
P $\rightarrow$ A set of n distinct points (Geometric Objects) in the plane.

Preprocess P such that closest point $x \in P$ of any query point q can be found efficiently

How to solve this efficiently?

Subdivision of the plane into n cells such that

- each cell contains exactly one site,
- if a point q lies in a cell containing p_i then
  $d(q, p_i) < d(q, p_j)$ for all $p_j \in P$, $j \neq i$.



Formal Definition

$P \rightarrow$ A set of n distinct points (Geometric Objects) in the plane.

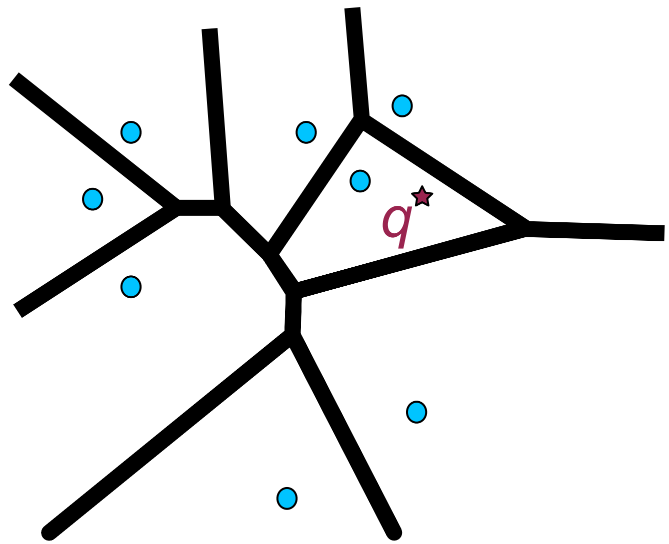
Preprocess P such that closest point $x \in P$ of any query point q can be found efficiently

How to solve this efficiently?

Subdivision of the plane into n cells such that

- each cell contains exactly one site,
- if a point q lies in a cell containing p_i then

  $d(q, p_i) < d(q, p_j)$ for all $p_j \in P, j \neq i$.



Formal Definition

P → A set of *n* distinct points (Geometric Objects) in the plane.

Preprocess P such that closest point $x \in P$ of any query point q can be found efficiently

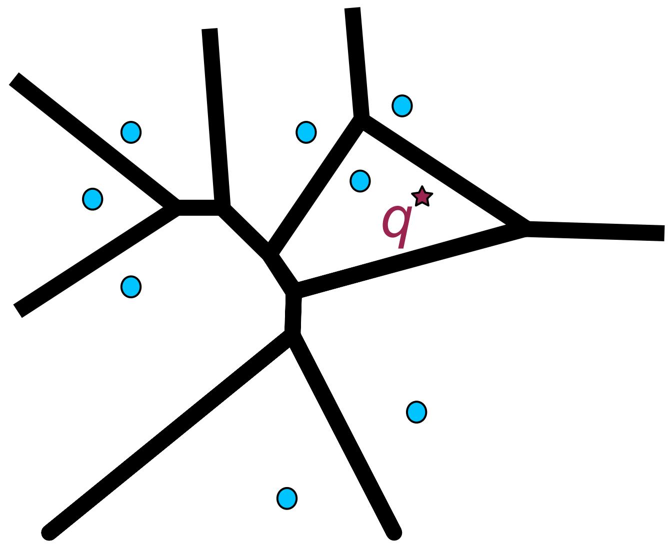
How to solve this efficiently?

Voronoi diagram of *P:*

V(P): Subdivision of the plane into *n* cells such that

- each cell contains exactly one site,
- if a point *q* lies in a cell containing p_i then

$d(q, p_i) < d(q, p_j)$ for all $p_j \in P, j \neq i$.



Formal Definition

$P \rightarrow$ A set of n distinct points (Geometric Objects) in the plane.

Preprocess P such that closest point $x \in P$ of any query point q can be found efficiently

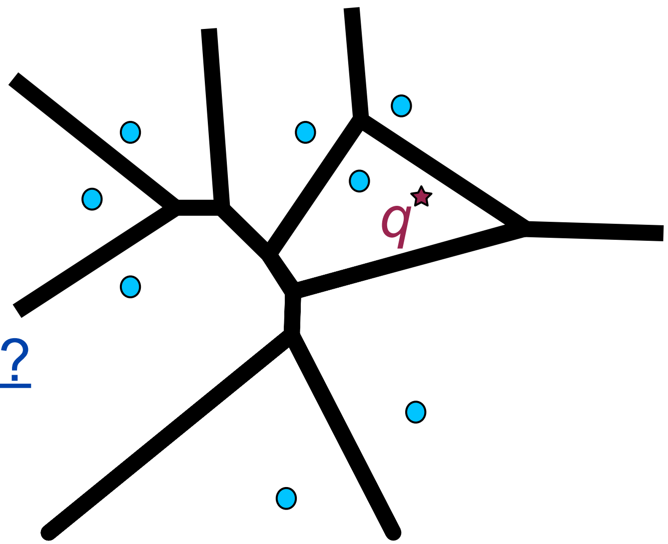
How to solve this efficiently?

Voronoi diagram of P:

$V(P)$: Subdivision of the plane into n cells such that

- each cell contains exactly one site,
- if a point q lies in a cell containing p_i then

  $d(q, p_i) < d(q, p_j)$ for all $p_j \in P, j \neq i$.



This is Planar Subdivision so what can we do?

Formal Definition

P → A set of n distinct points (Geometric Objects) in the plane.

Preprocess P such that closest point $x \in P$ of any query point q can be found efficiently

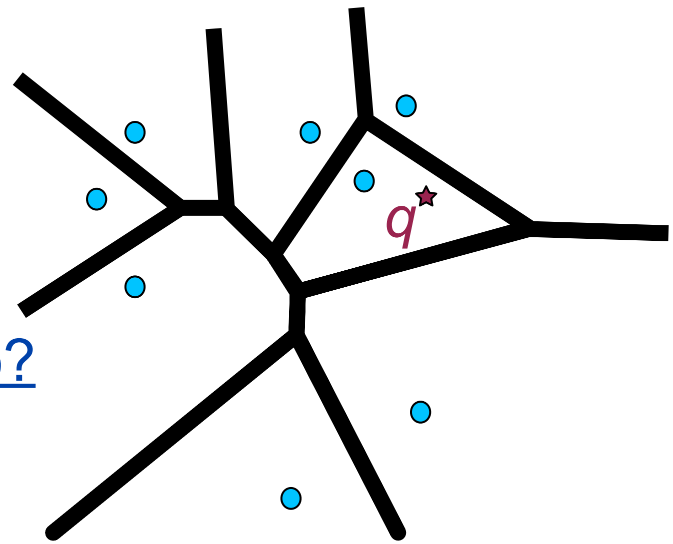
How to solve this efficiently?

Voronoi diagram of P:

$V(P)$: Subdivision of the plane into n cells such that

- each cell contains exactly one site,
- if a point q lies in a cell containing p_i then

  $d(q, p_i) < d(q, p_j)$ for all $p_j \in P, j \neq i$.

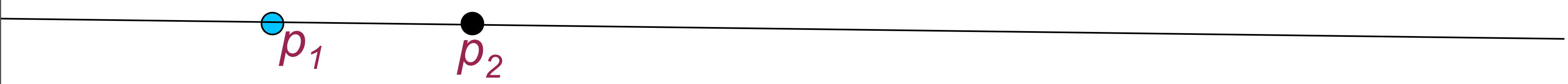


This is Planar Subdivision so what can we do?

Planar point location

Computing the Voronoi Diagram

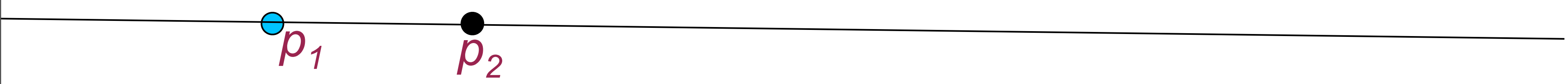
Input: A set of points on a line (special case)



Computing the Voronoi Diagram

Input: A set of points on a line (special case)

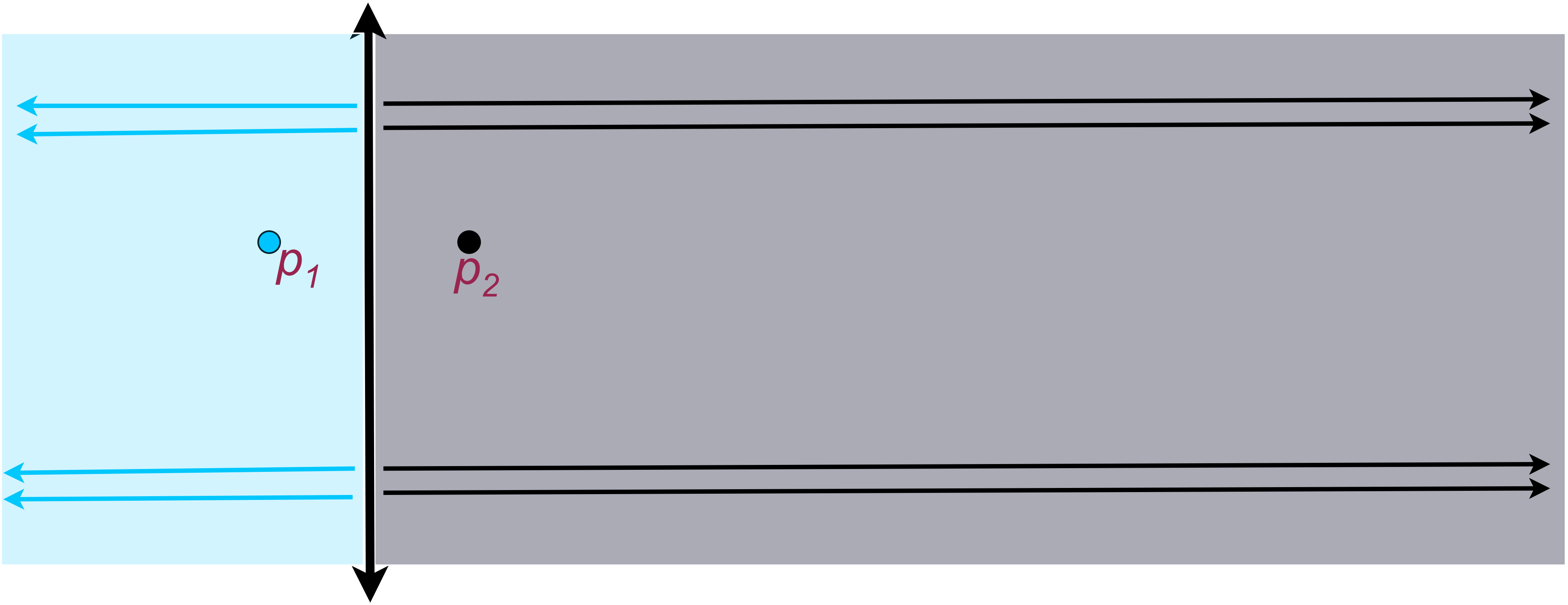
Output: A partitioning of the plane into regions of nearest neighbors



Computing the Voronoi Diagram

Input: A set of points on a line (special case)

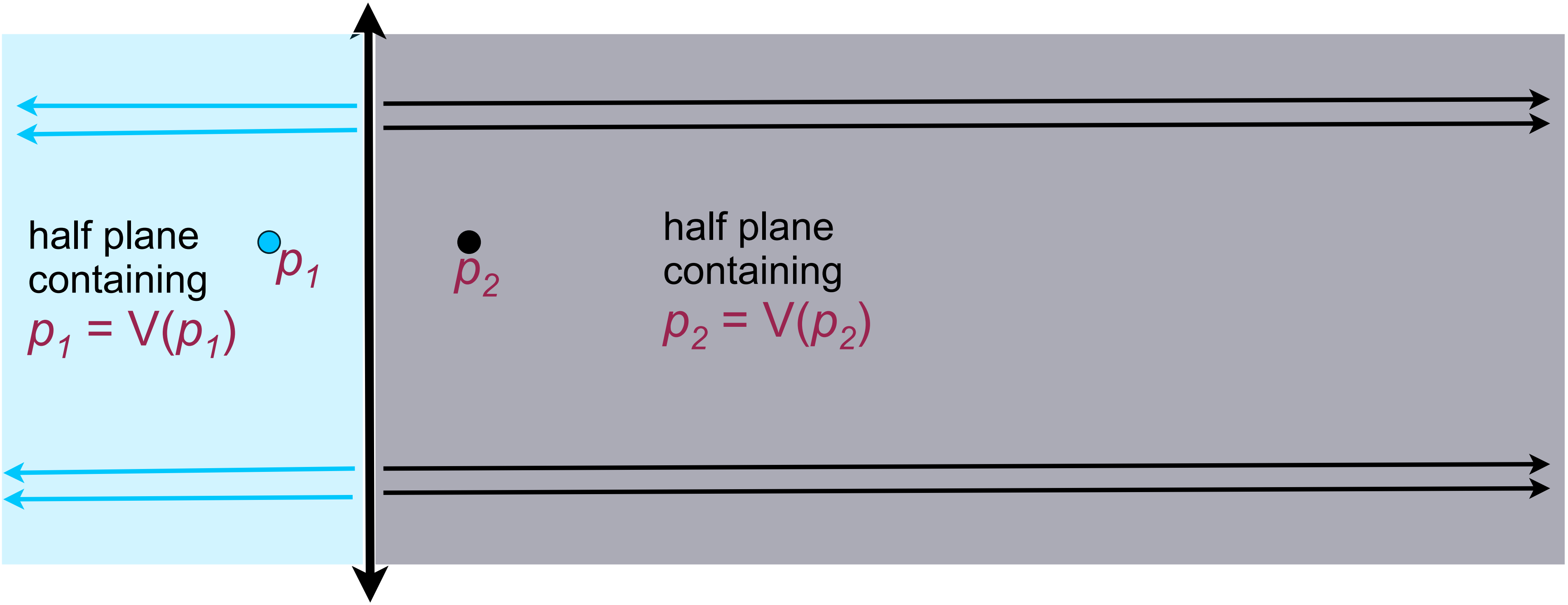
Output: A partitioning of the plane into regions of nearest neighbors



Computing the Voronoi Diagram

Input: A set of points on a line (special case)

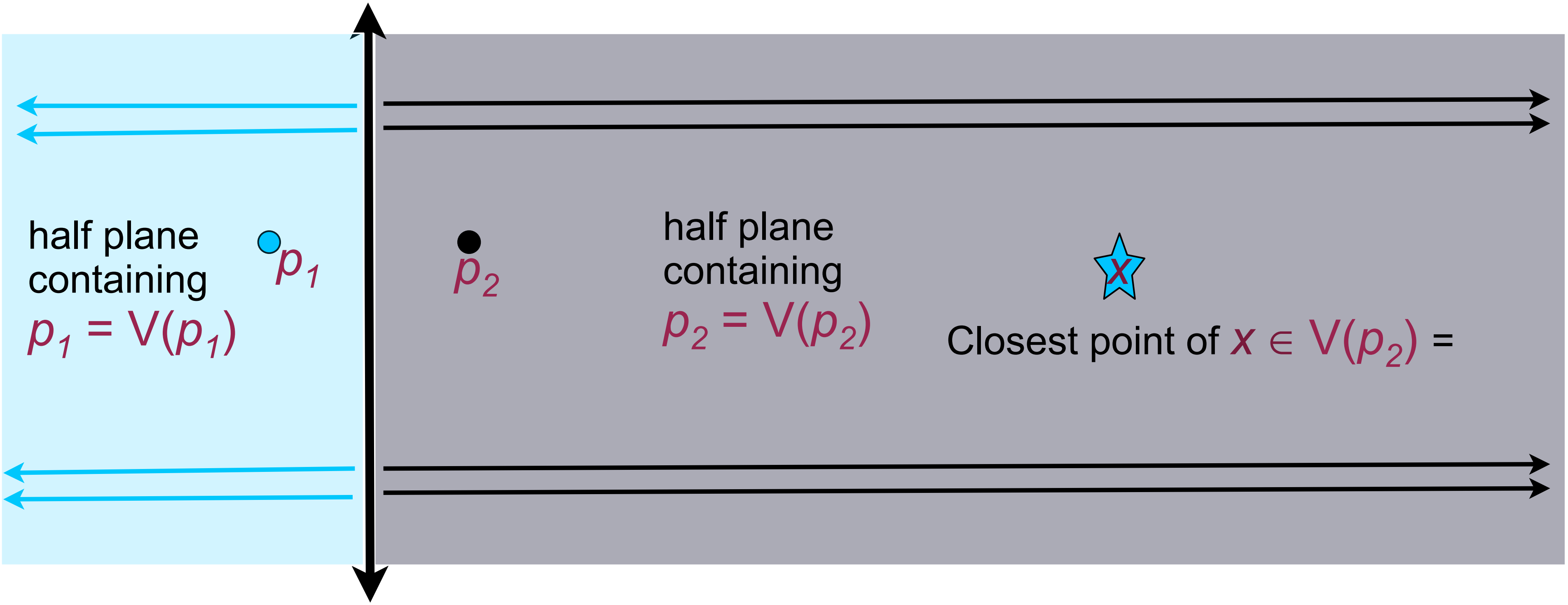
Output: A partitioning of the plane into regions of nearest neighbors



Computing the Voronoi Diagram

Input: A set of points on a line (special case)

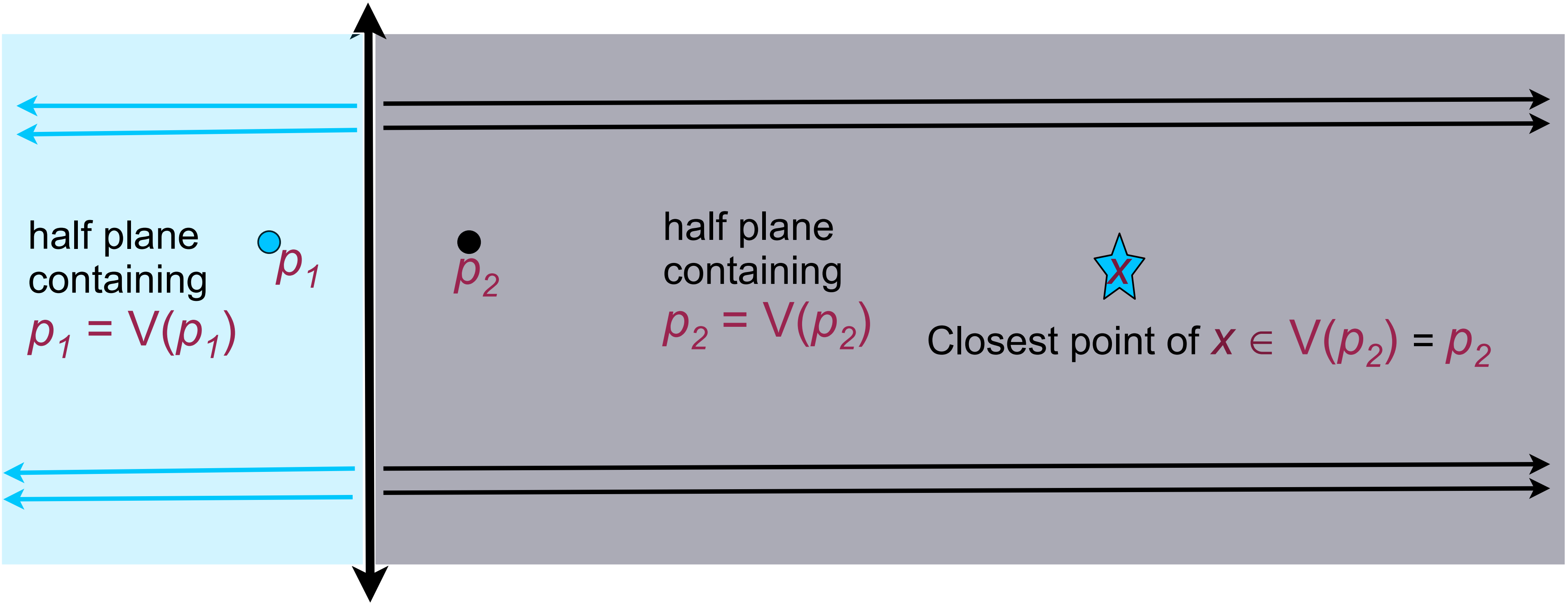
Output: A partitioning of the plane into regions of nearest neighbors



Computing the Voronoi Diagram

Input: A set of points on a line (special case)

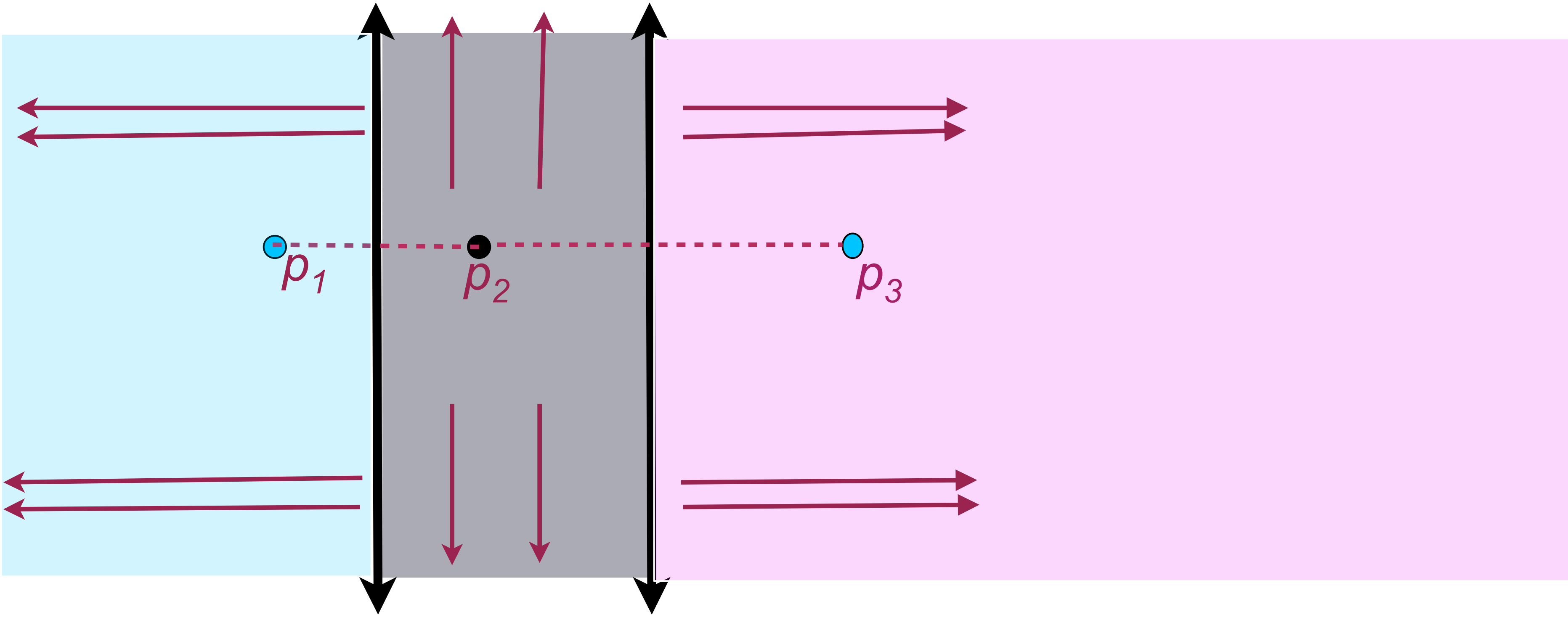
Output: A partitioning of the plane into regions of nearest neighbors



Computing the Voronoi Diagram

Input: A set of points on a line (special case)

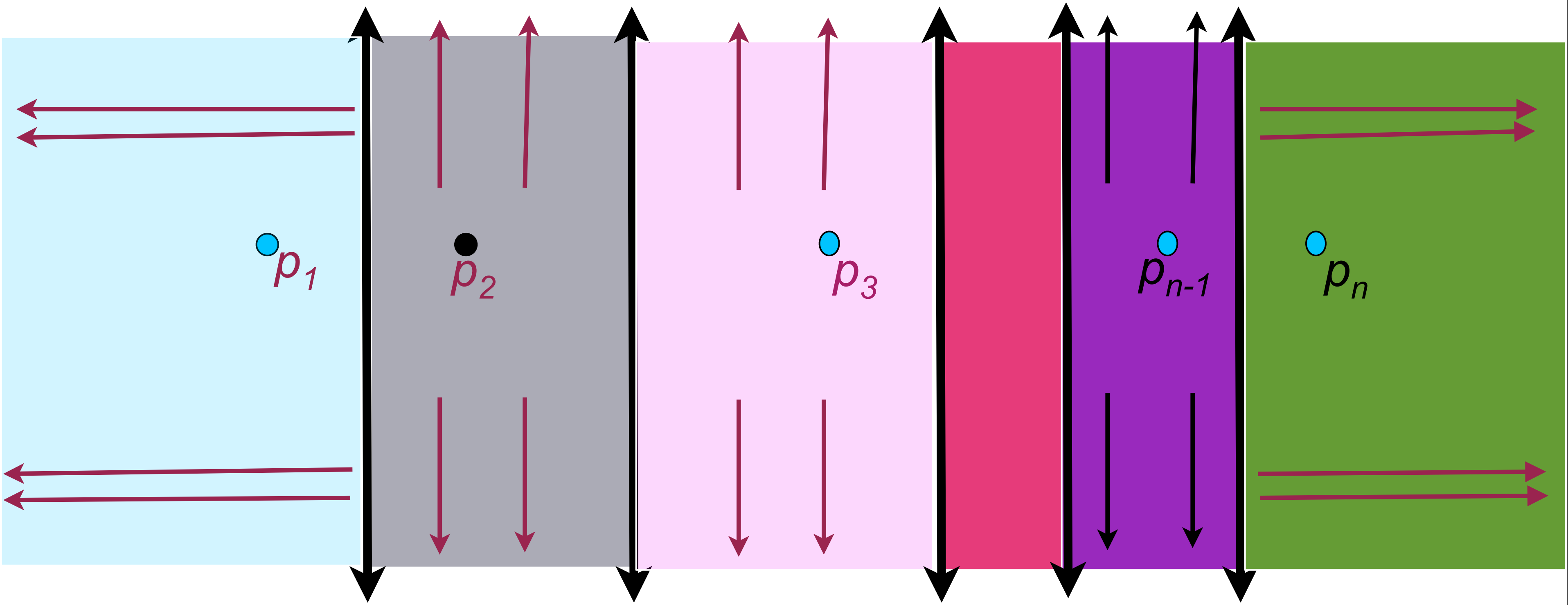
Output: A partitioning of the plane into regions of nearest neighbors



Computing the Voronoi Diagram

Input: A set of points on a line (special case)

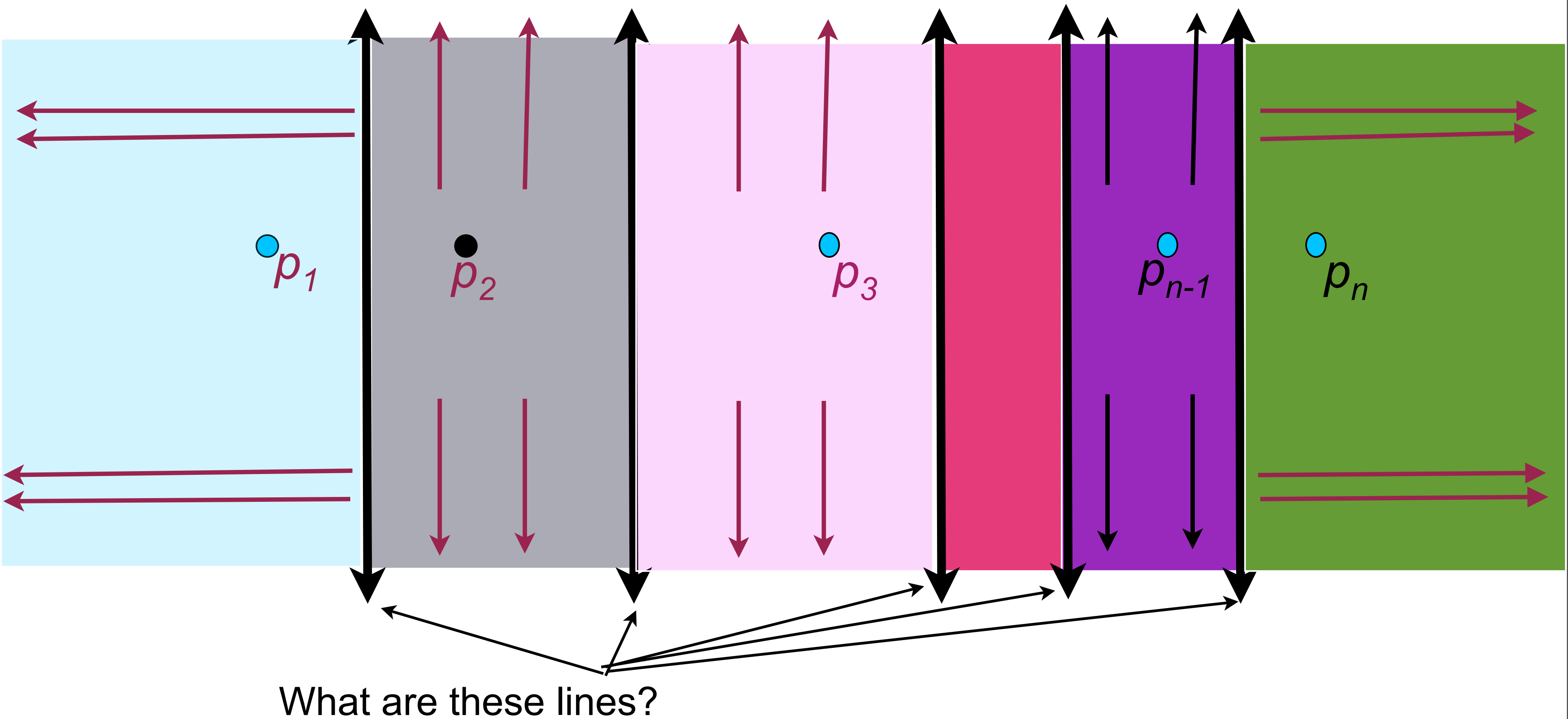
Output: A partitioning of the plane into regions of nearest neighbors



Computing the Voronoi Diagram

Input: A set of points on a line (special case)

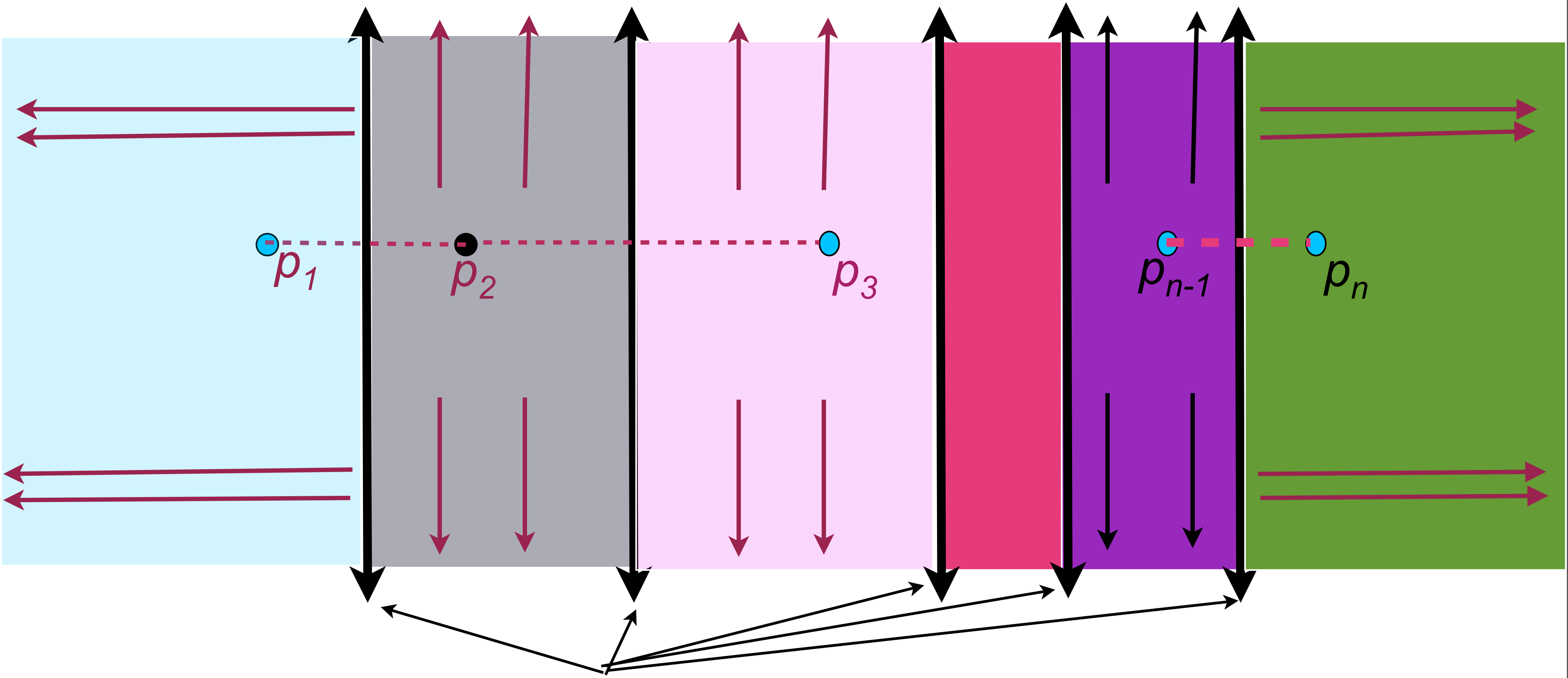
Output: A partitioning of the plane into regions of nearest neighbors



Computing the Voronoi Diagram

Input: A set of points on a line (special case)

Output: A partitioning of the plane into regions of nearest neighbors



What are these lines? Perpendicular bisector of line segment [p_i p_{i+1}]

Computing the Voronoi Diagram

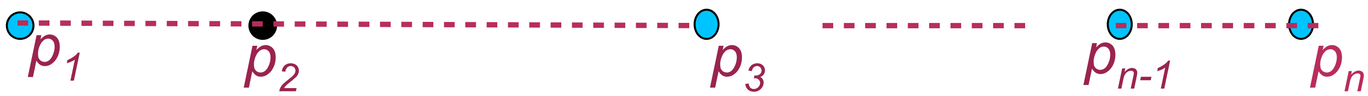
Computing the Voronoi Diagram

Input: A set of points $P=(p_1, p_2, \ldots, p_n)$ on a line (special case)

Computing the Voronoi Diagram

Input: A set of points $P=(p_1, p_2, \ldots, p_n)$ on a line (special case)

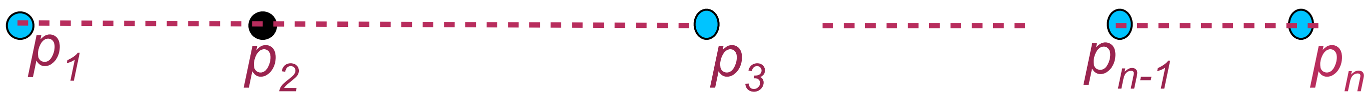
Output: A partitioning of the plane into regions of nearest neighbors



Computing the Voronoi Diagram

Input: A set of points $P=(p_1, p_2, \ldots, p_n)$ on a line (special case)

Output: A partitioning of the plane into regions of nearest neighbors

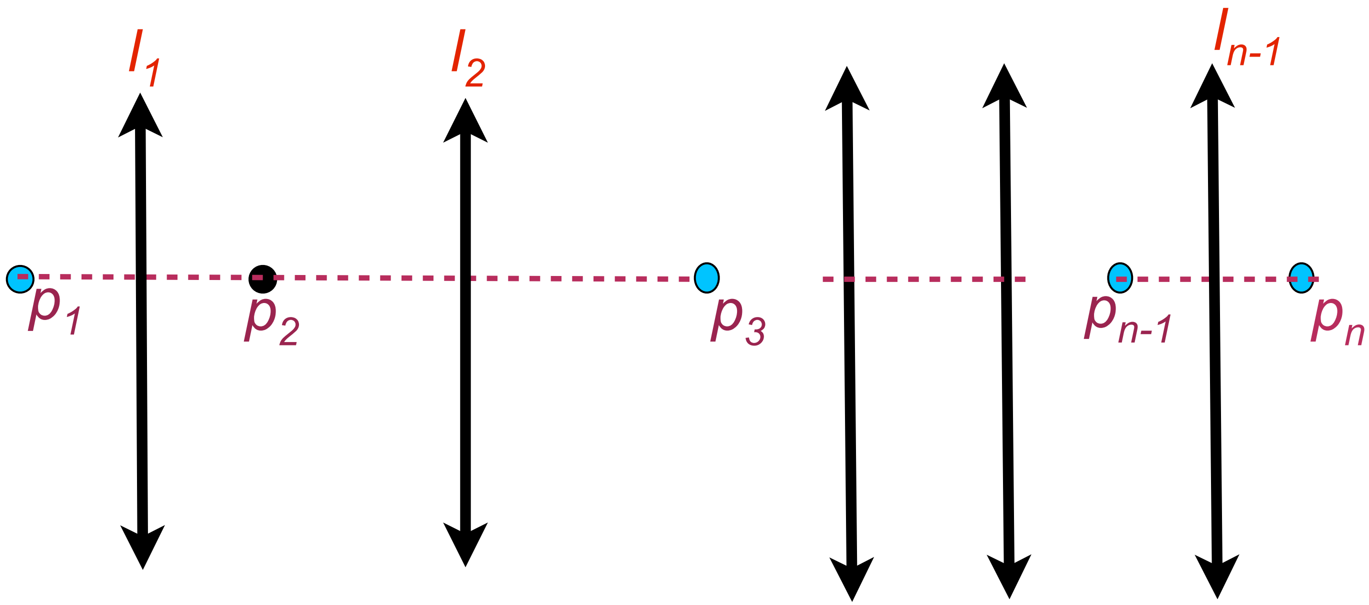


Find Perpendicular bisector l_i of line segment $[\, p_i \, p_{i+1} \,]$

Computing the Voronoi Diagram

Input: A set of points $P=(p_1, p_2, \ldots, p_n)$ on a line (special case)

Output: A partitioning of the plane into regions of nearest neighbors

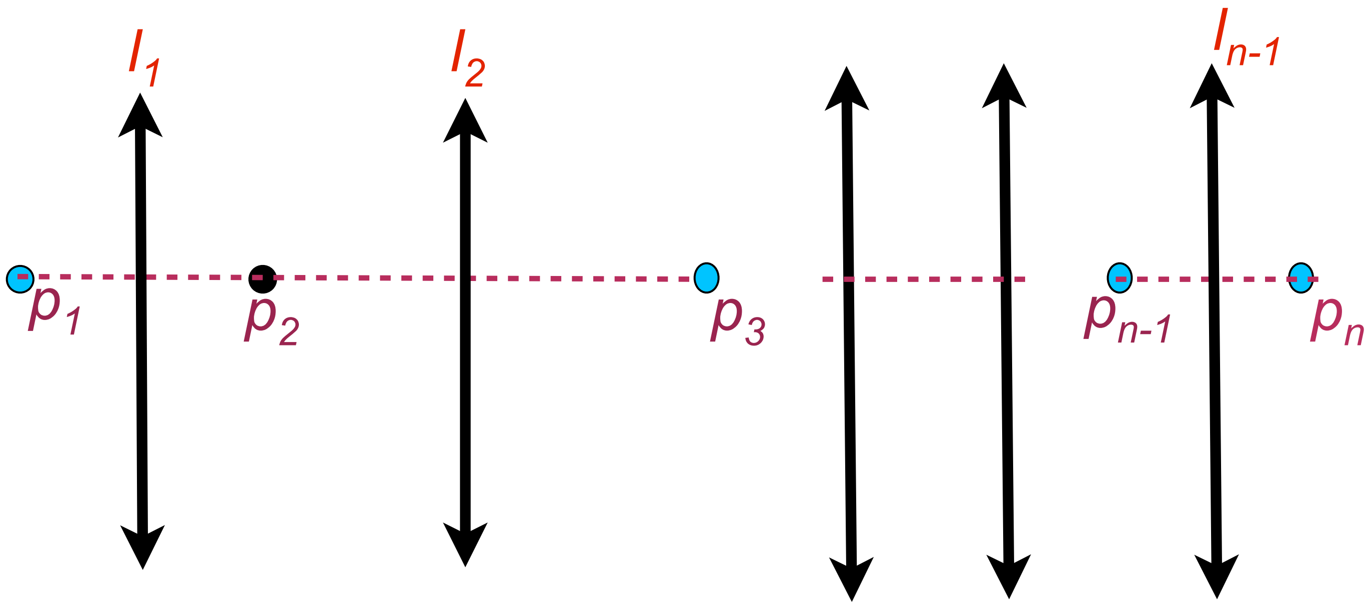


Find Perpendicular bisector l_i of line segment $[\, p_i \, p_{i+1} \,]$

Computing the Voronoi Diagram

Input: A set of points $P=(p_1, p_2, \ldots, p_n)$ on a line (special case)

Output: A partitioning of the plane into regions of nearest neighbors



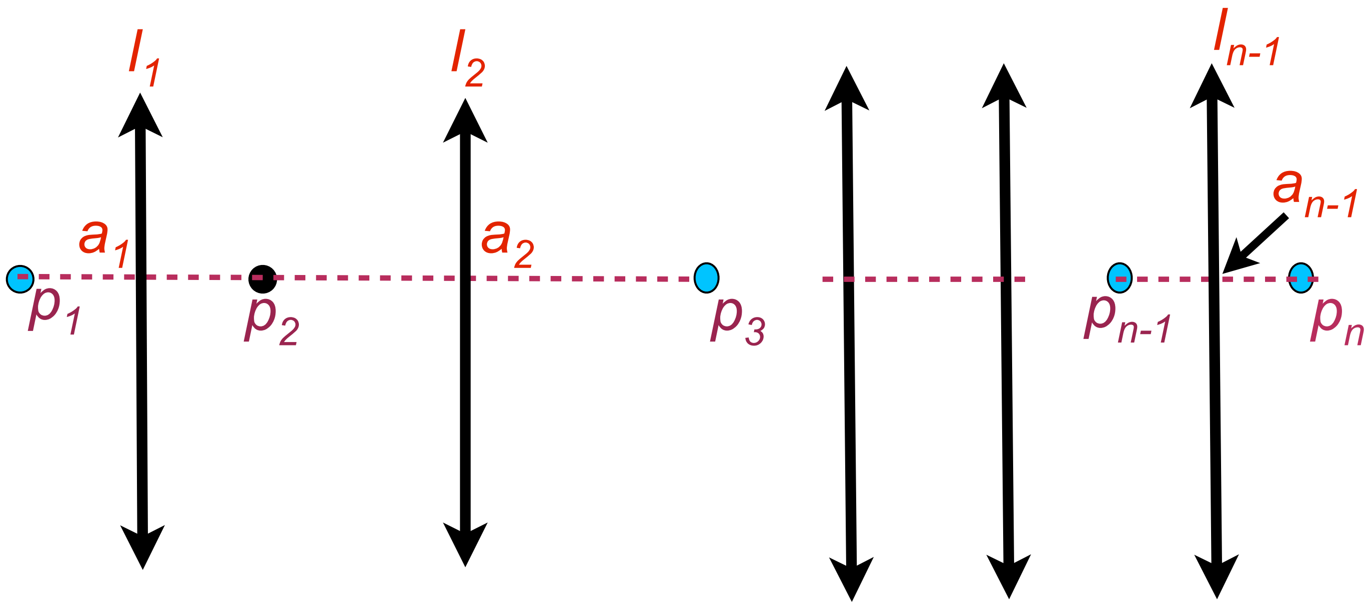
Find Perpendicular bisector l_i of line segment $[p_i\, p_{i+1}]$

Let a_i be the intersection point of l_i and $[p_i\, p_{i+1}]$

Computing the Voronoi Diagram

Input: A set of points $P=(p_1, p_2, \ldots, p_n)$ on a line (special case)

Output: A partitioning of the plane into regions of nearest neighbors



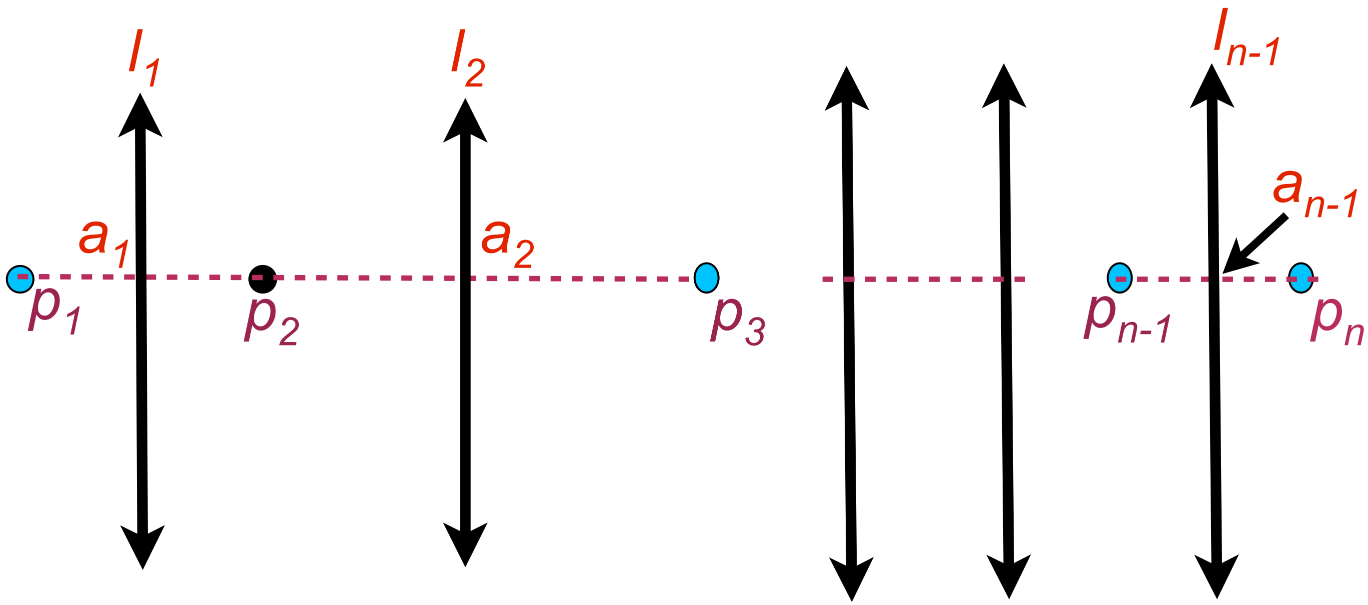
Find Perpendicular bisector l_i of line segment $[\,p_i\, p_{i+1}\,]$

Let a_i be the intersection point of l_i and $[\,p_i\, p_{i+1}\,]$

Computing the Voronoi Diagram

Input: A set of points $P=(p_1, p_2, \dots, p_n)$ on a line (special case)

Output: A partitioning of the plane into regions of nearest neighbors



Find Perpendicular bisector l_i of line segment $[\,p_i\, p_{i+1}\,]$

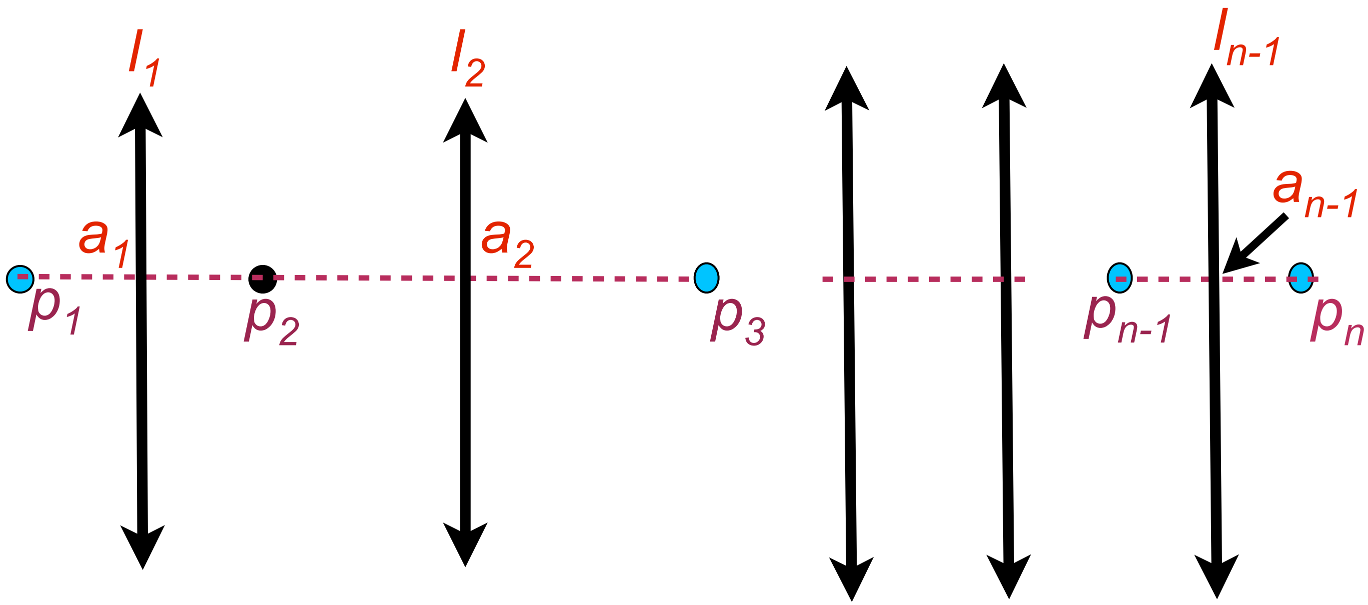
Let a_i be the intersection point of l_i and $[\,p_i\, p_{i+1}\,]$

Sort a_i in increasing x-coordinate

Computing the Voronoi Diagram

Input: A set of points $P=(p_1, p_2, \ldots, p_n)$ on a line (special case)

Output: A partitioning of the plane into regions of nearest neighbors



Find Perpendicular bisector l_i of line segment $[\, p_i\, p_{i+1}\,]$

Let a_i be the intersection point of l_i and $[\, p_i\, p_{i+1}\,]$

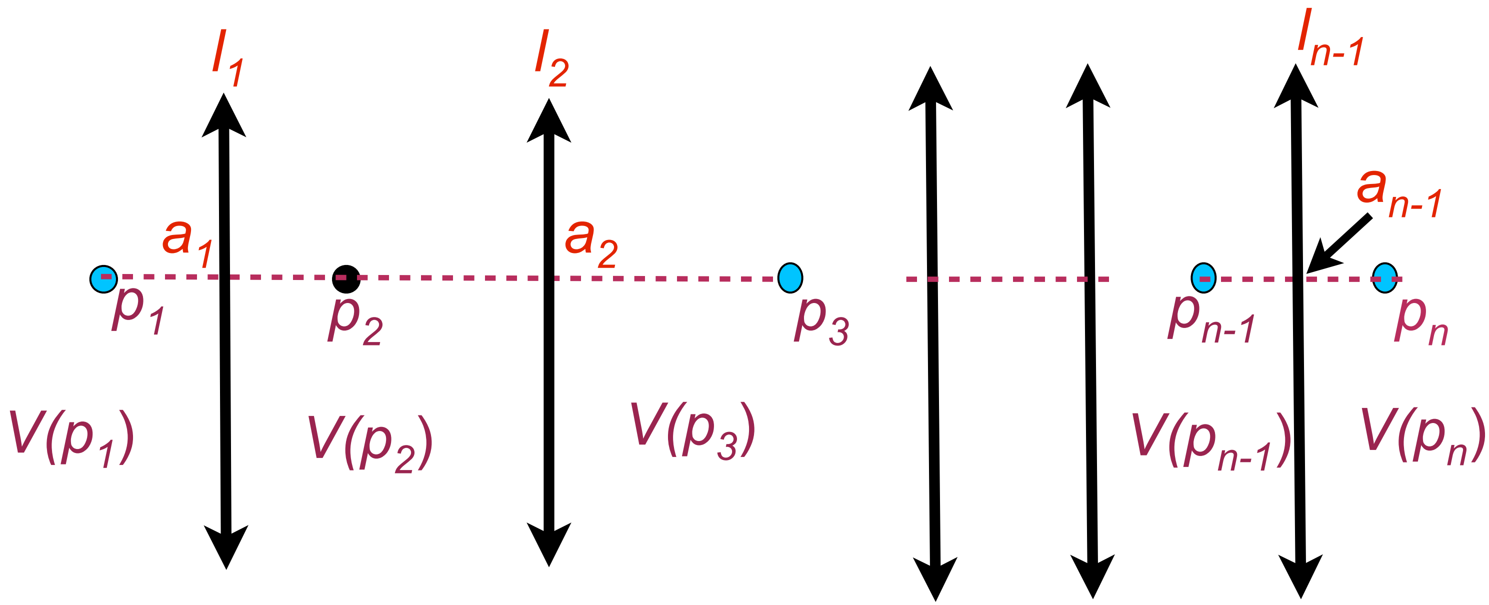
Sort a_i in increasing x-coordinate

This gives us Voronoi Diagram V(P)

Computing the Voronoi Diagram

Input: A set of points $P=(p_1, p_2, \ldots, p_n)$ on a line (special case)

Output: A partitioning of the plane into regions of nearest neighbors



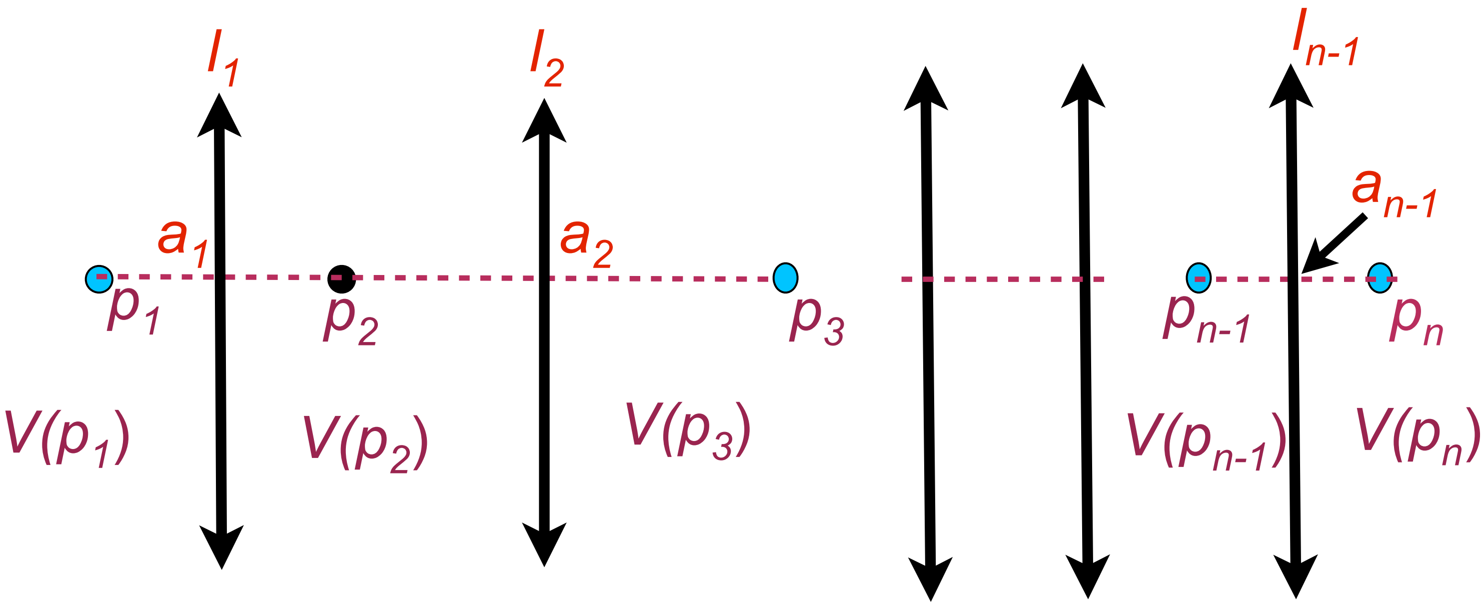
Find Perpendicular bisector l_i of line segment $[\, p_i \, p_{i+1} \,]$

Let a_i be the intersection point of l_i and $[\, p_i \, p_{i+1} \,]$

Sort a_i in increasing x-coordinate

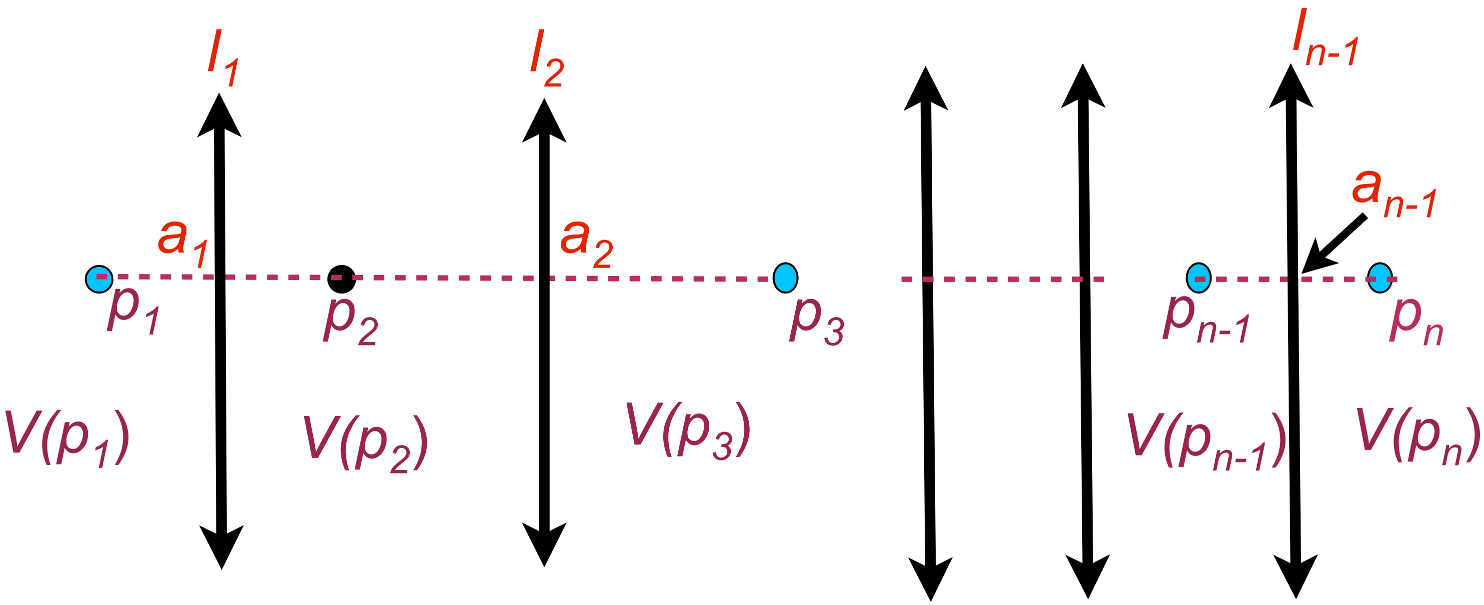
This gives us Voronoi Diagram $V(P)$

Query Answering



Query Answering

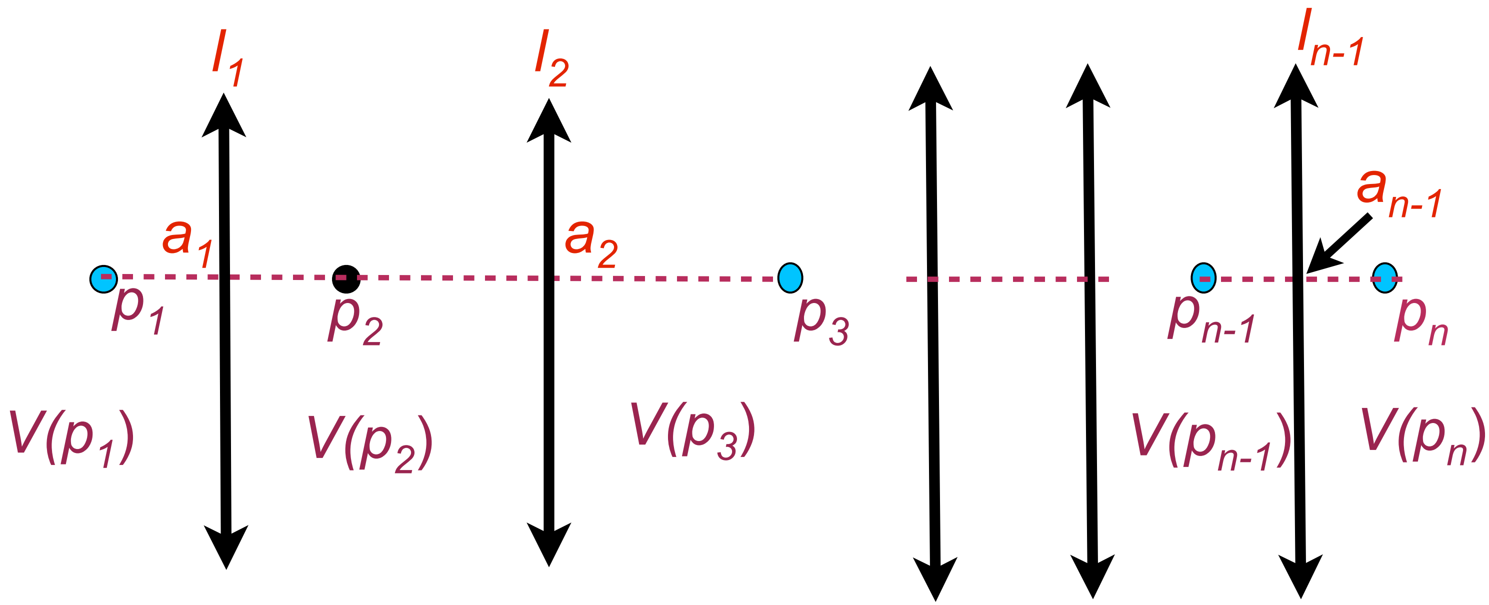
We have a_i's sorted in increasing x-coordinate



Query Answering

We have a_i's sorted in increasing x-coordinate

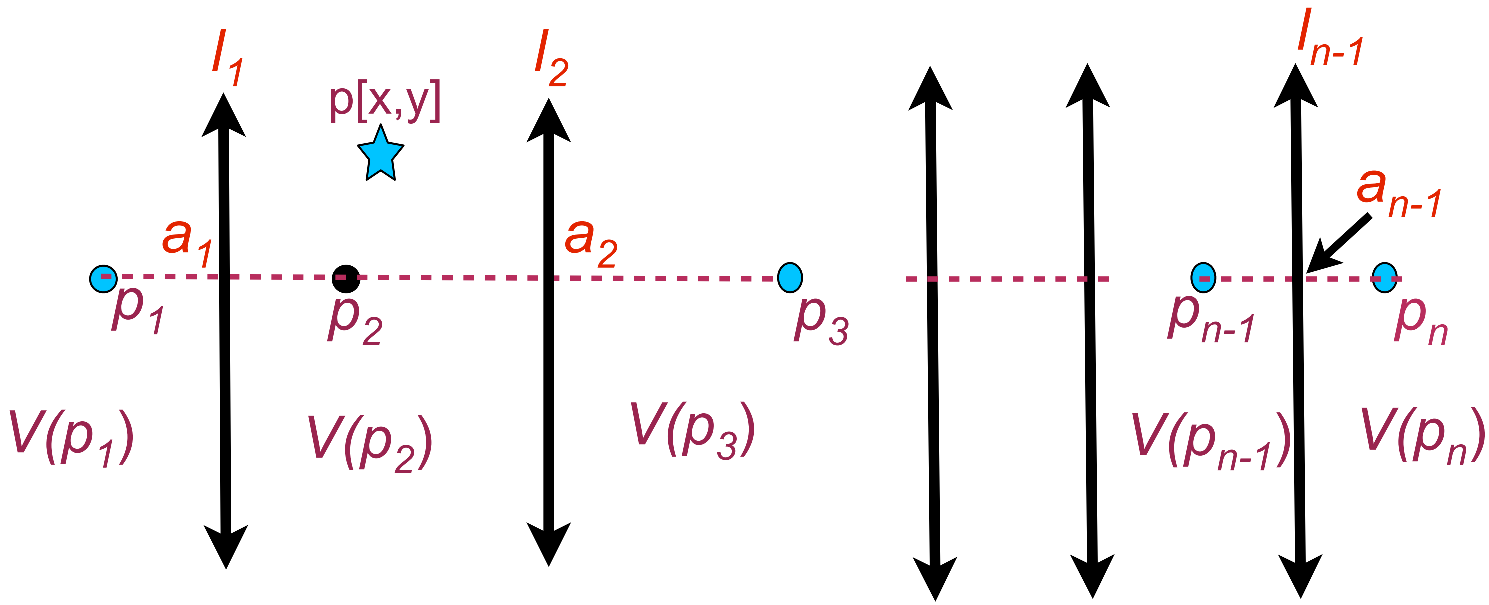
Given a query point p[x,y]



Query Answering

We have a_i's sorted in increasing x-coordinate

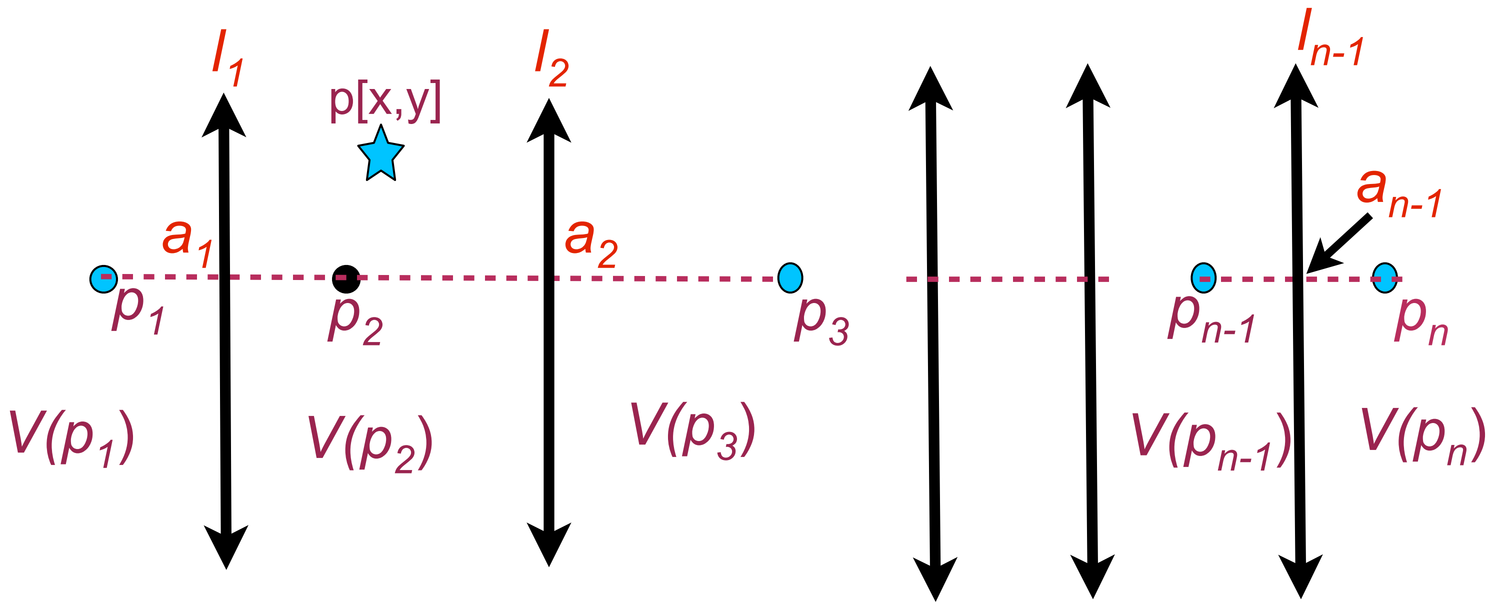
Given a query point p[x,y]



Query Answering

We have a_i's sorted in increasing x-coordinate

Given a query point p[x,y]

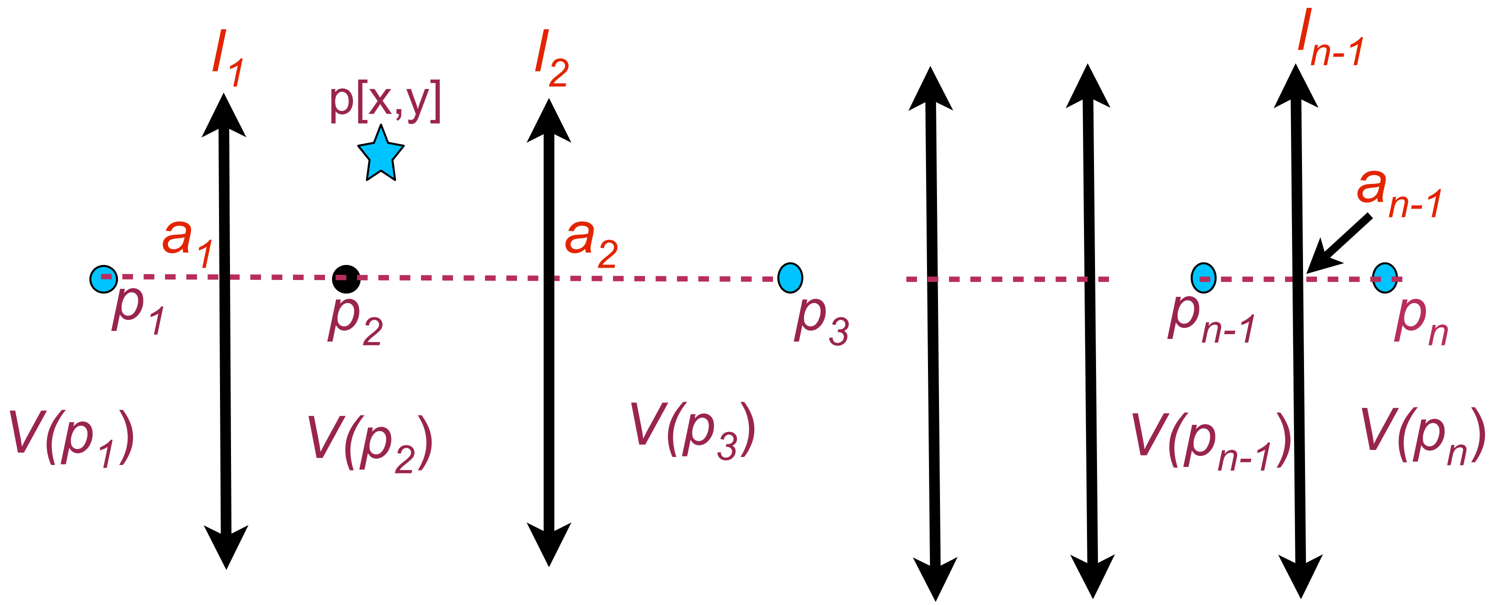


What we have to do?

Query Answering

We have a_i's sorted in increasing x-coordinate

Given a query point p[x,y]

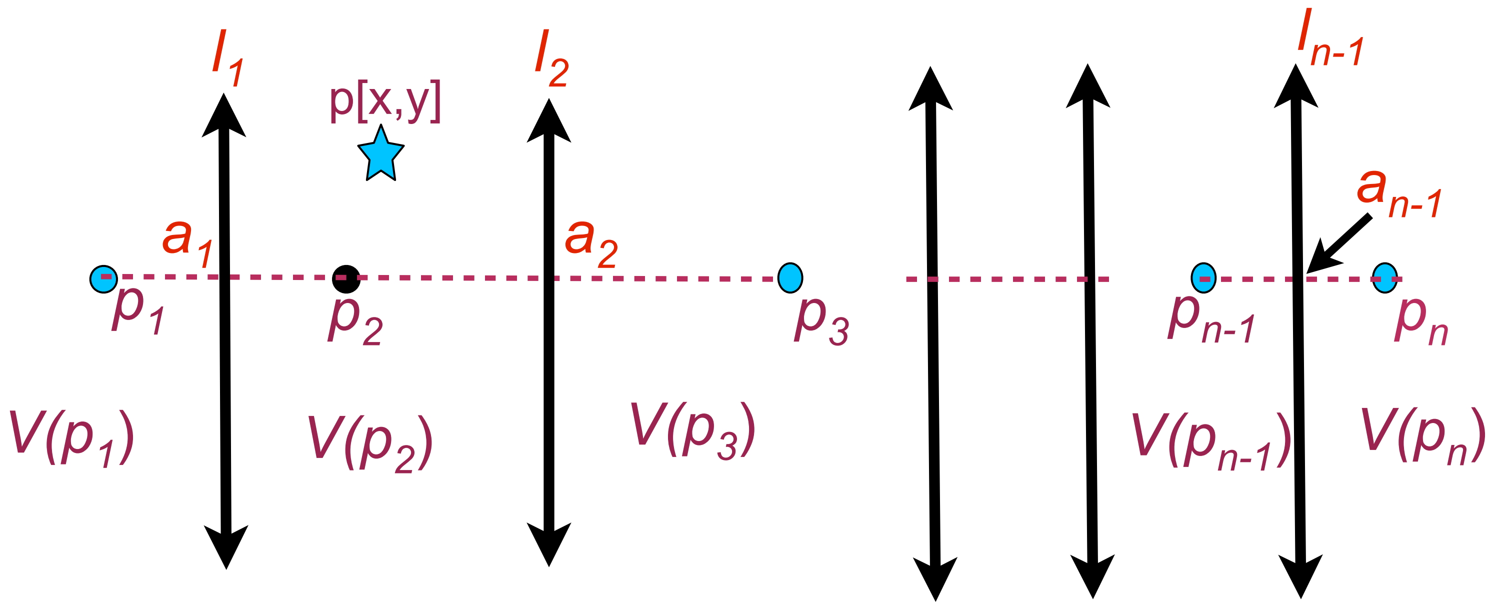


What we have to do? Locate x correctly between a_i and a_{i+1}

Query Answering

We have a_i's sorted in increasing x-coordinate

Given a query point p[x,y]



What we have to do? Locate x correctly between a_i and a_{i+1}

We can forget about y coordinate

Time Complexity analysis

Time Complexity analysis

Preprocessing Time =O(n log n)

Time Complexity analysis

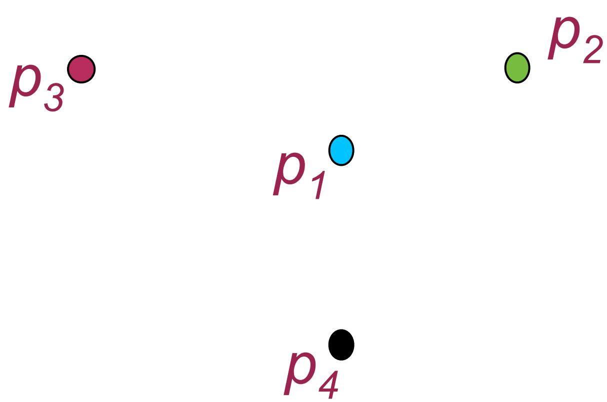
Preprocessing Time =O(n log n)

Query Time =O(log n)

Computing the Voronoi Diagram

Computing the Voronoi Diagram

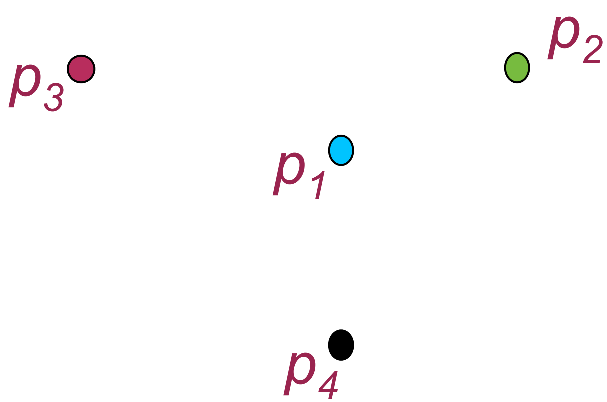
Input: A set of points $P=(p_1, p_2, \ldots, p_n)$ on 2D



Computing the Voronoi Diagram

Input: A set of points $P=(p_1, p_2, \ldots, p_n)$ on 2D

Output: A partitioning of the plane into regions of nearest neighbors

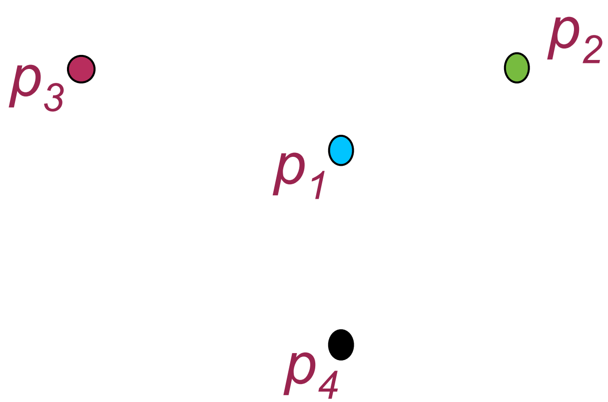


Computing the Voronoi Diagram

Input: A set of points $P=(p_1, p_2, \ldots, p_n)$ on 2D

Output: A partitioning of the plane into regions of nearest neighbors

Find cell for each point one by one?

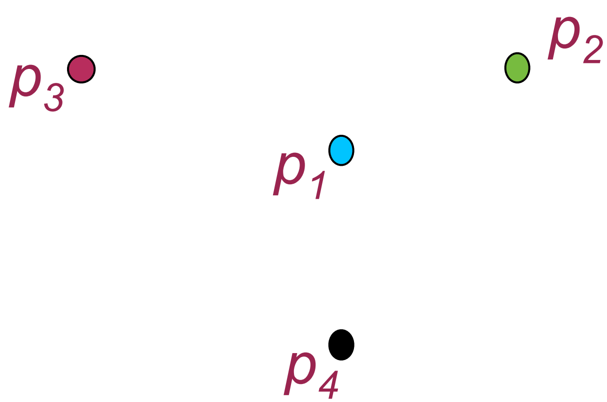


Computing the Voronoi Diagram

Input: A set of points $P=(p_1, p_2, \ldots, p_n)$ on 2D

Output: A partitioning of the plane into regions of nearest neighbors

Find cell for each point one by one? use perpendicular bisector argument



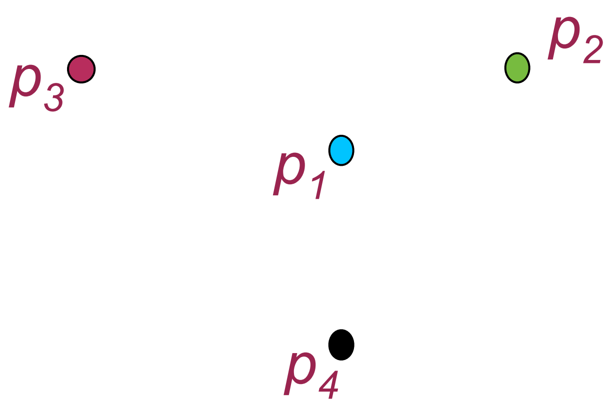
Computing the Voronoi Diagram

Input: A set of points $P=(p_1, p_2, \ldots, p_n)$ on 2D

Output: A partitioning of the plane into regions of nearest neighbors

Find cell for each point one by one? use perpendicular bisector argument

Find region for p_1



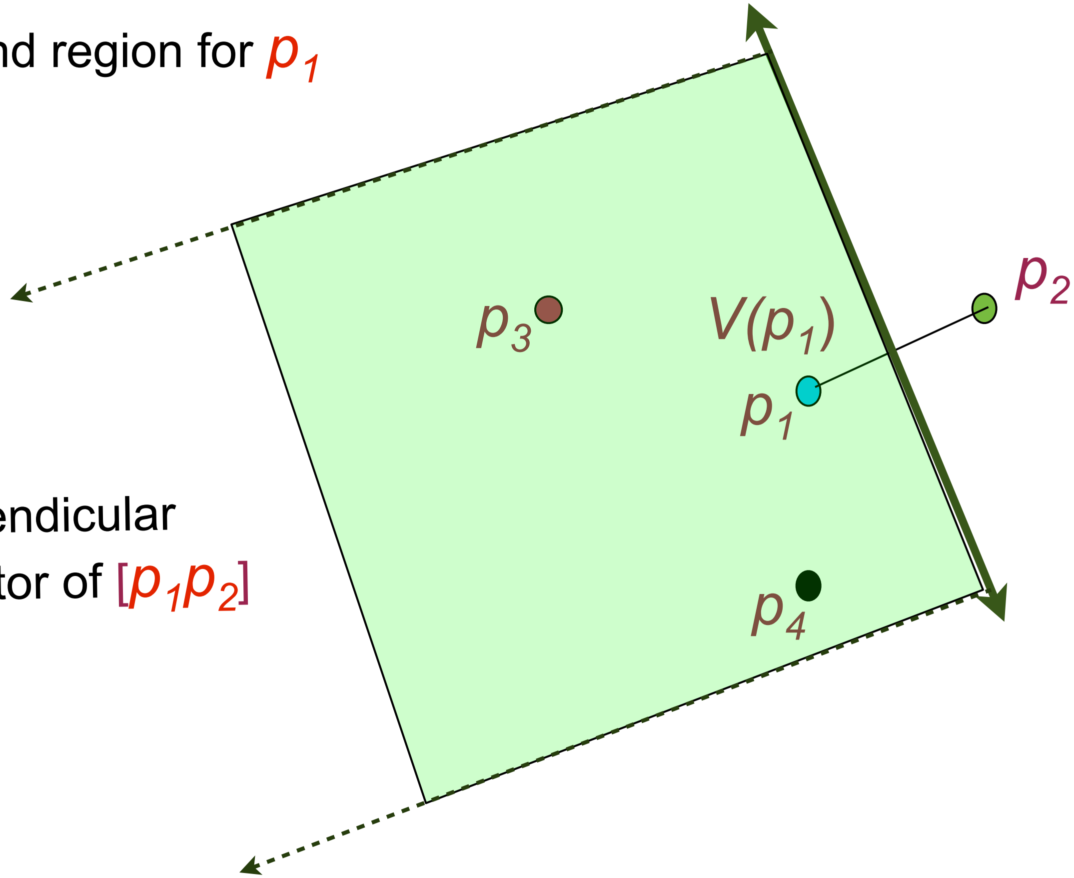
Computing the Voronoi Diagram

Input: A set of points $P=(p_1, p_2, \ldots, p_n)$ on 2D

Output: A partitioning of the plane into regions of nearest neighbors

Find cell for each point one by one? use perpendicular bisector argument

Find region for p_1



Perpendicular bisector of $[p_1 p_2]$

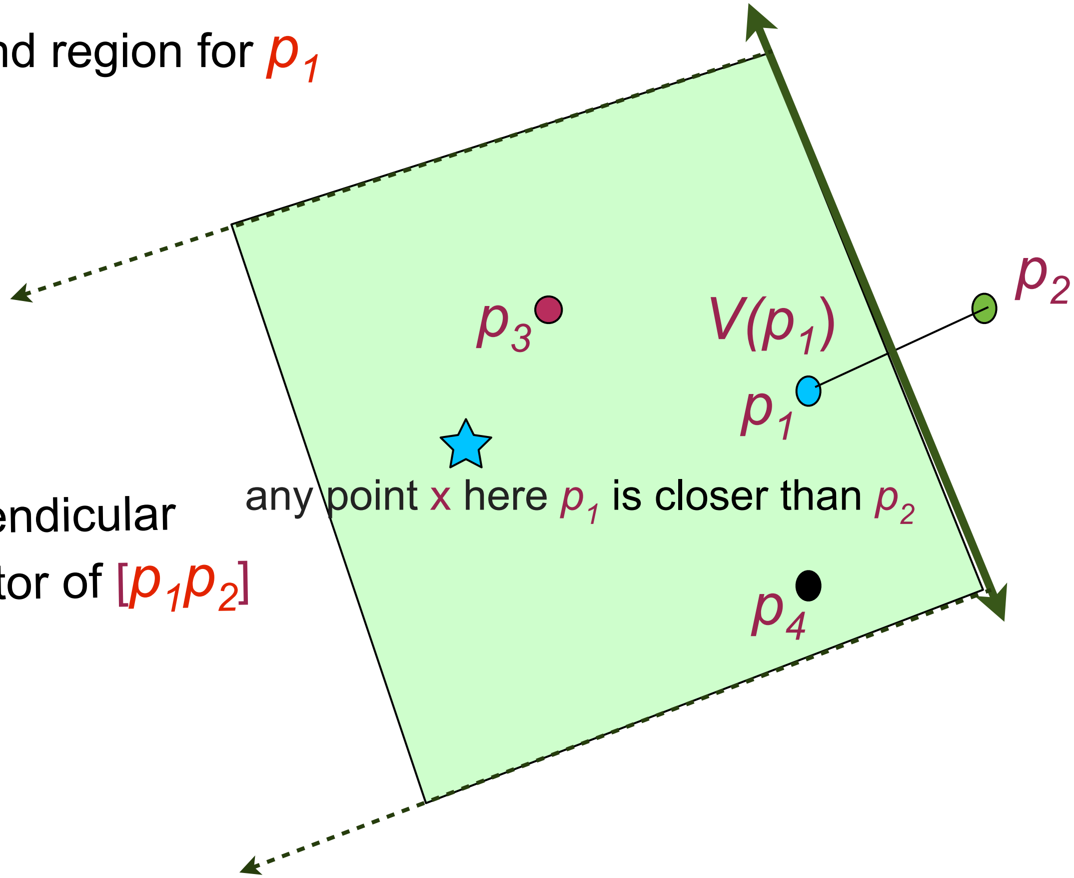
Computing the Voronoi Diagram

Input: A set of points $P=(p_1, p_2, \ldots, p_n)$ on 2D

Output: A partitioning of the plane into regions of nearest neighbors

Find cell for each point one by one? use perpendicular bisector argument

Find region for p_1



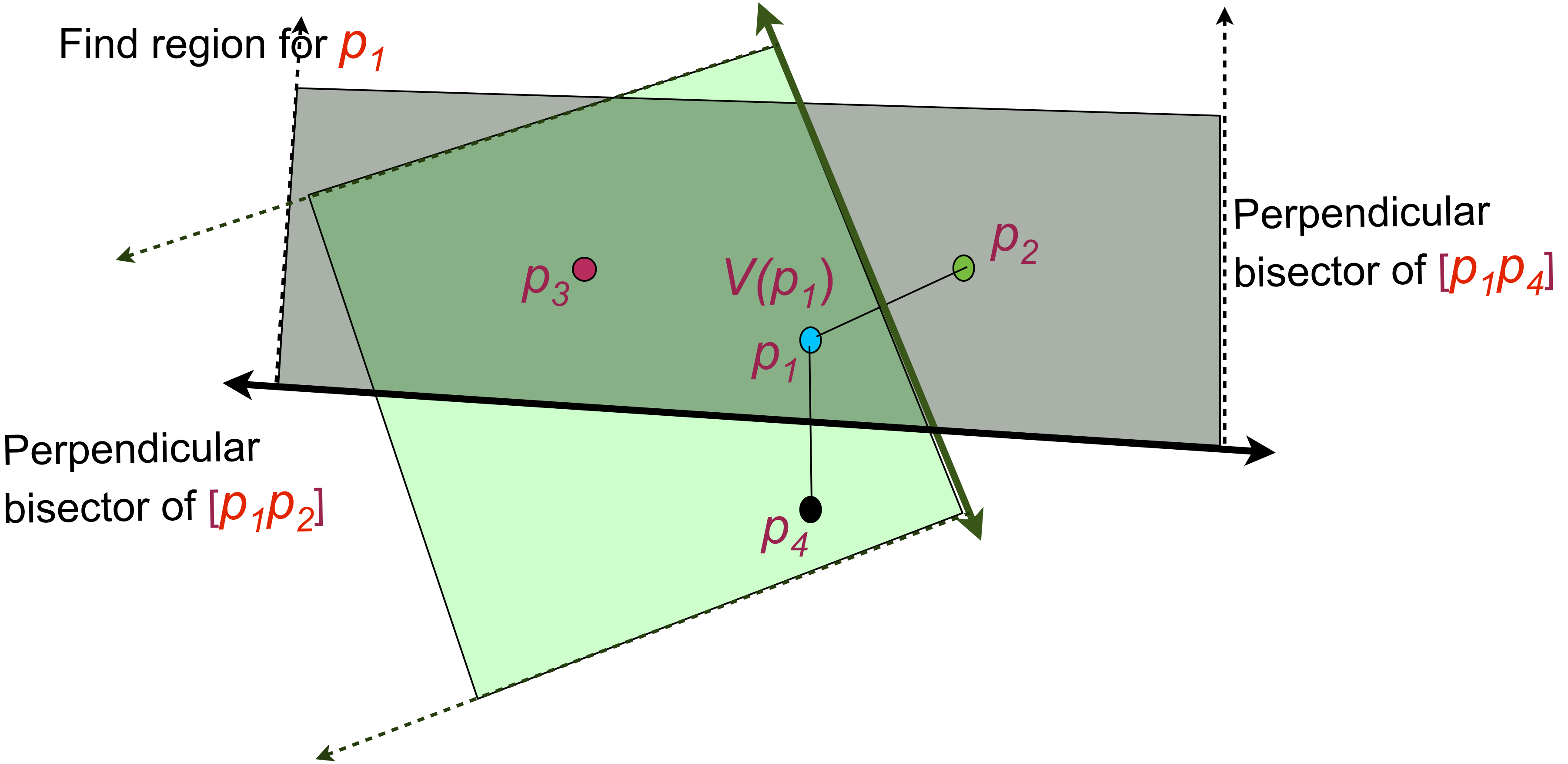
Perpendicular bisector of $[p_1 p_2]$

Computing the Voronoi Diagram

Input: A set of points $P=(p_1, p_2, \ldots, p_n)$ on 2D

Output: A partitioning of the plane into regions of nearest neighbors

Find cell for each point one by one? use perpendicular bisector argument

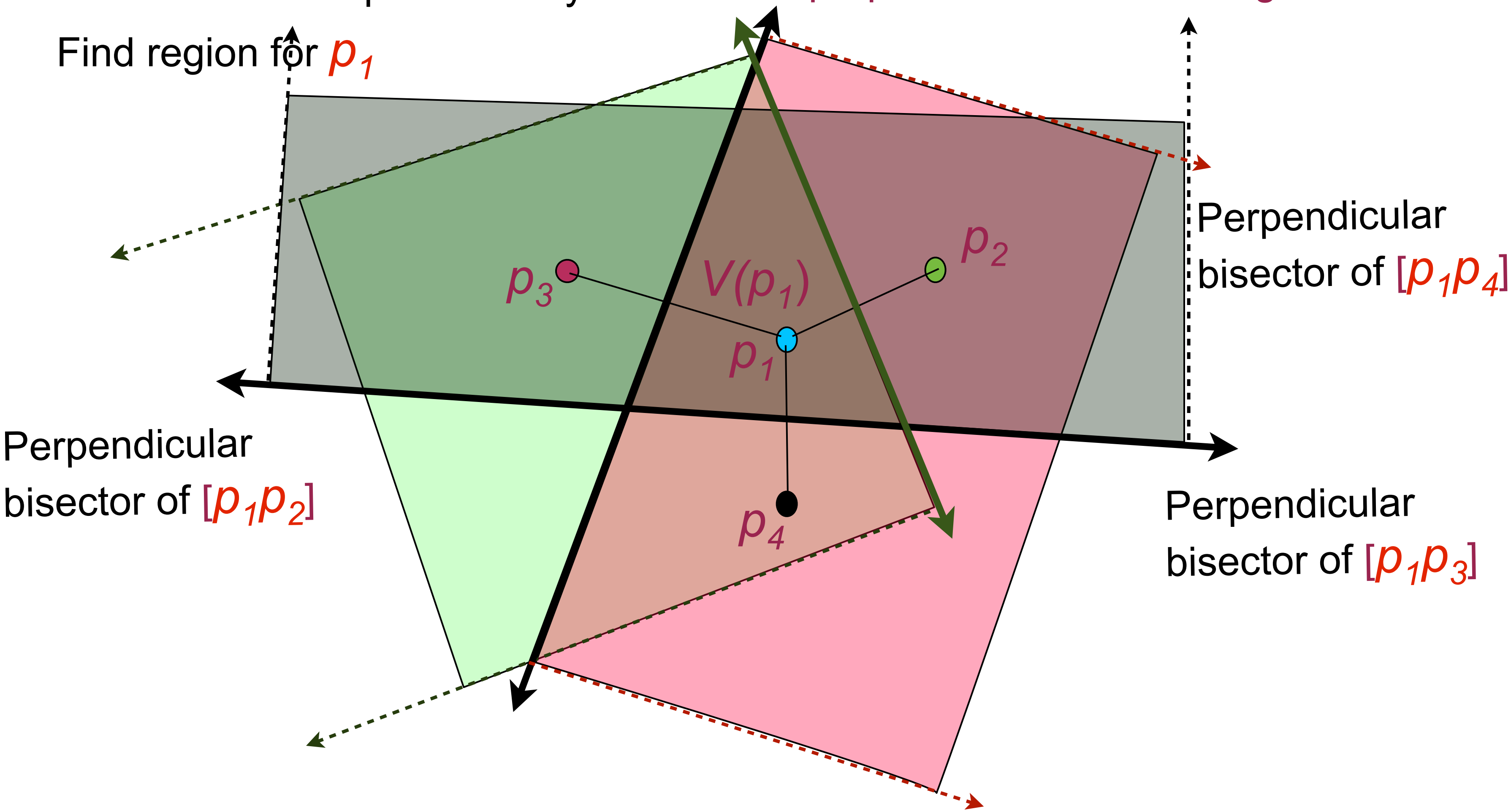


Computing the Voronoi Diagram

Input: A set of points $P=(p_1, p_2, \ldots, p_n)$ on 2D

Output: A partitioning of the plane into regions of nearest neighbors

Find cell for each point one by one? use perpendicular bisector argument



Computing the Voronoi Diagram

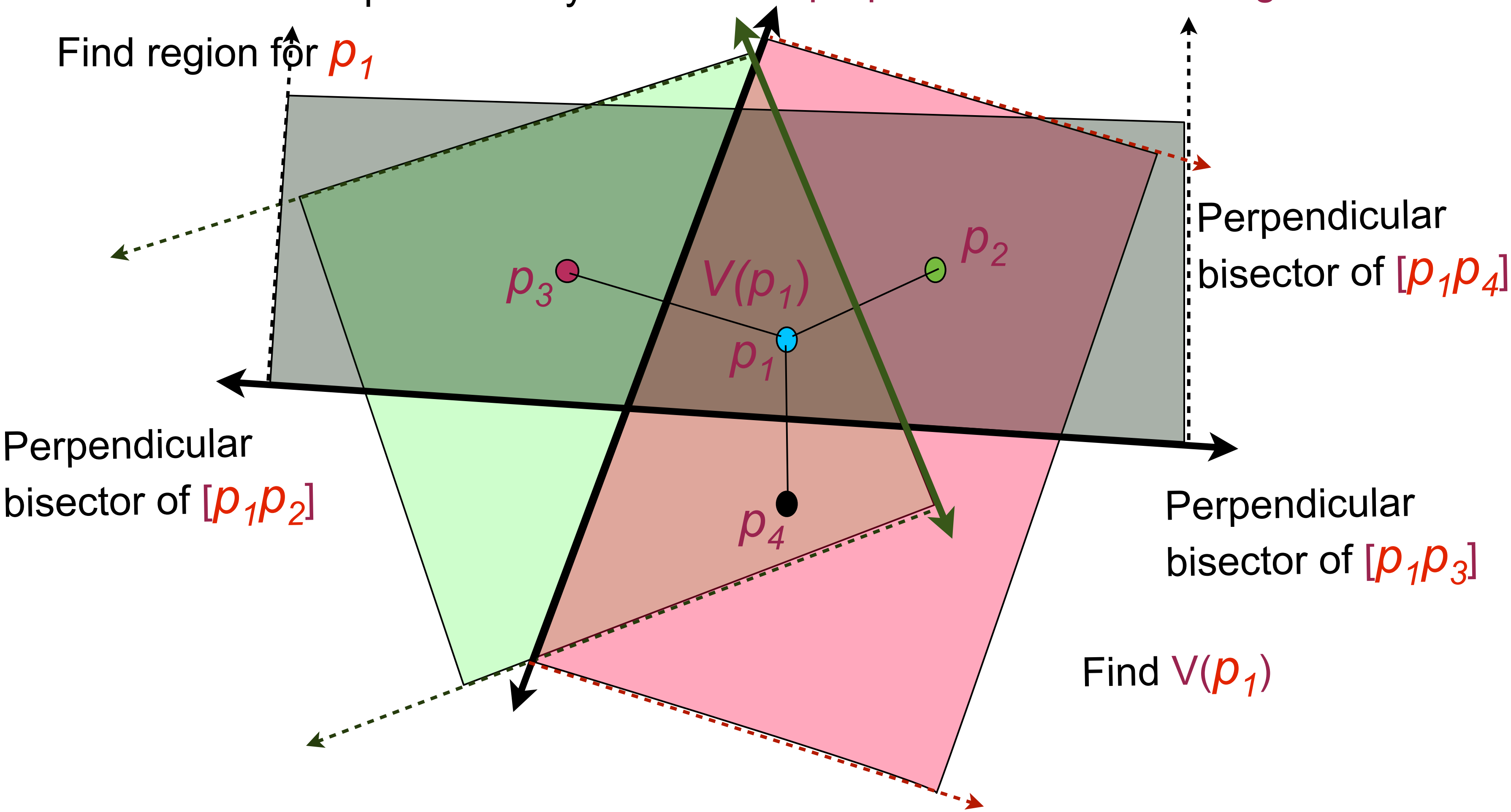
Input: A set of points $P=(p_1, p_2, \ldots, p_n)$ on 2D

Output: A partitioning of the plane into regions of nearest neighbors

Find cell for each point one by one?

use perpendicular bisector argument

Find region for p_1



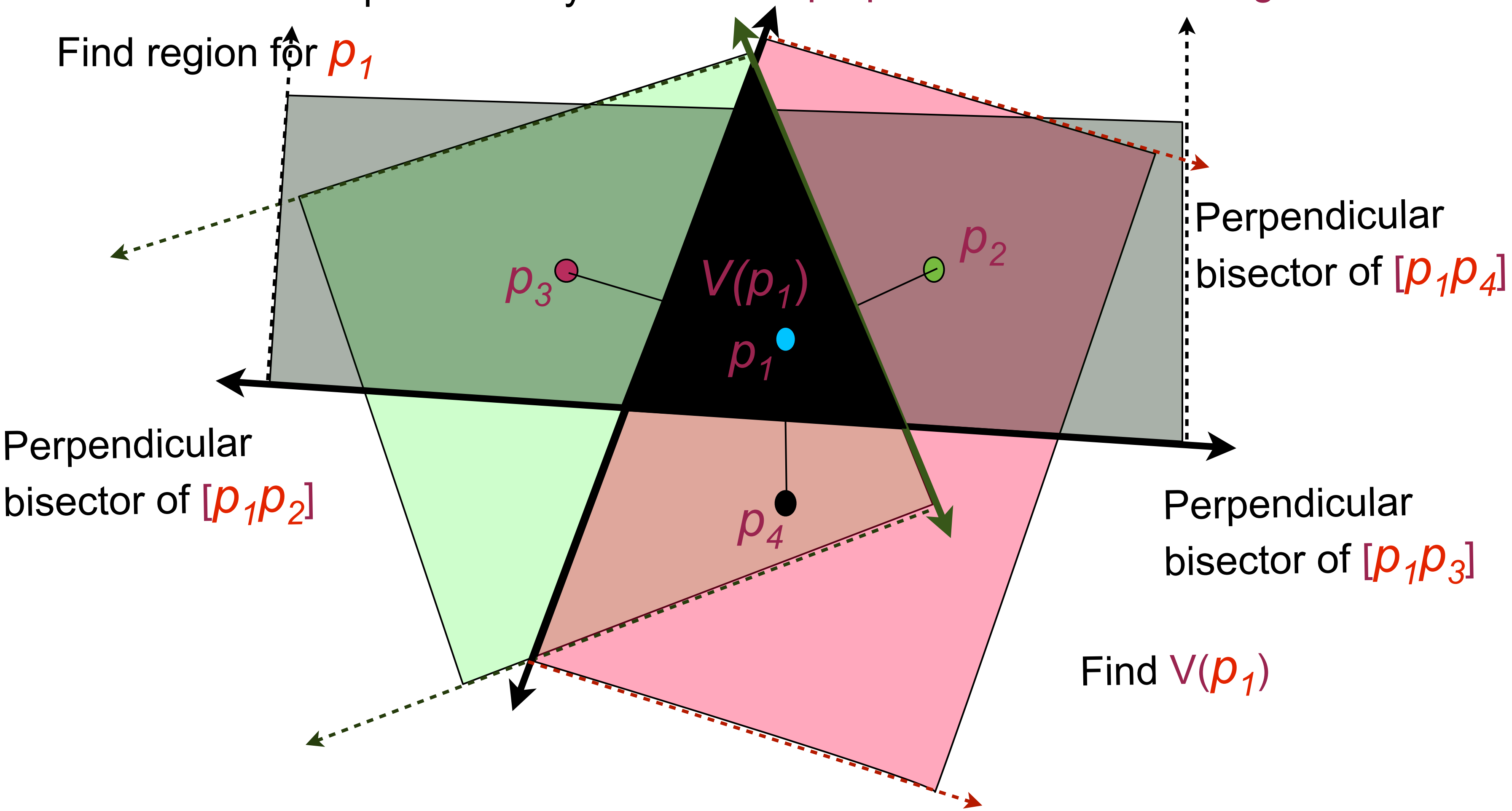
Find $V(p_1)$

Computing the Voronoi Diagram

Input: A set of points $P=(p_1, p_2, \ldots, p_n)$ on 2D

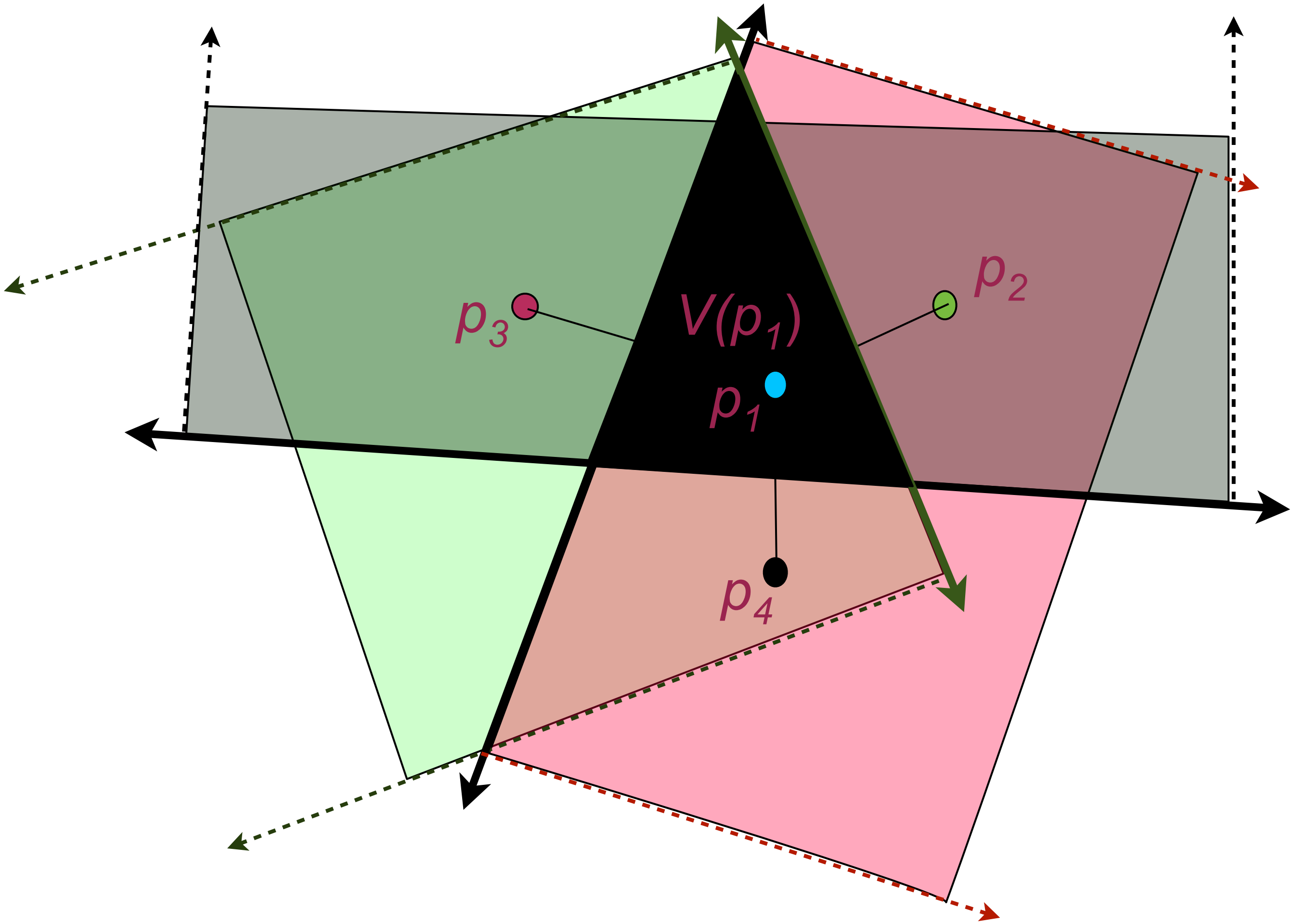
Output: A partitioning of the plane into regions of nearest neighbors

Find cell for each point one by one? use perpendicular bisector argument



Computing the Voronoi Diagram

How do we find $V(p_1)$?



Computing the Voronoi Diagram

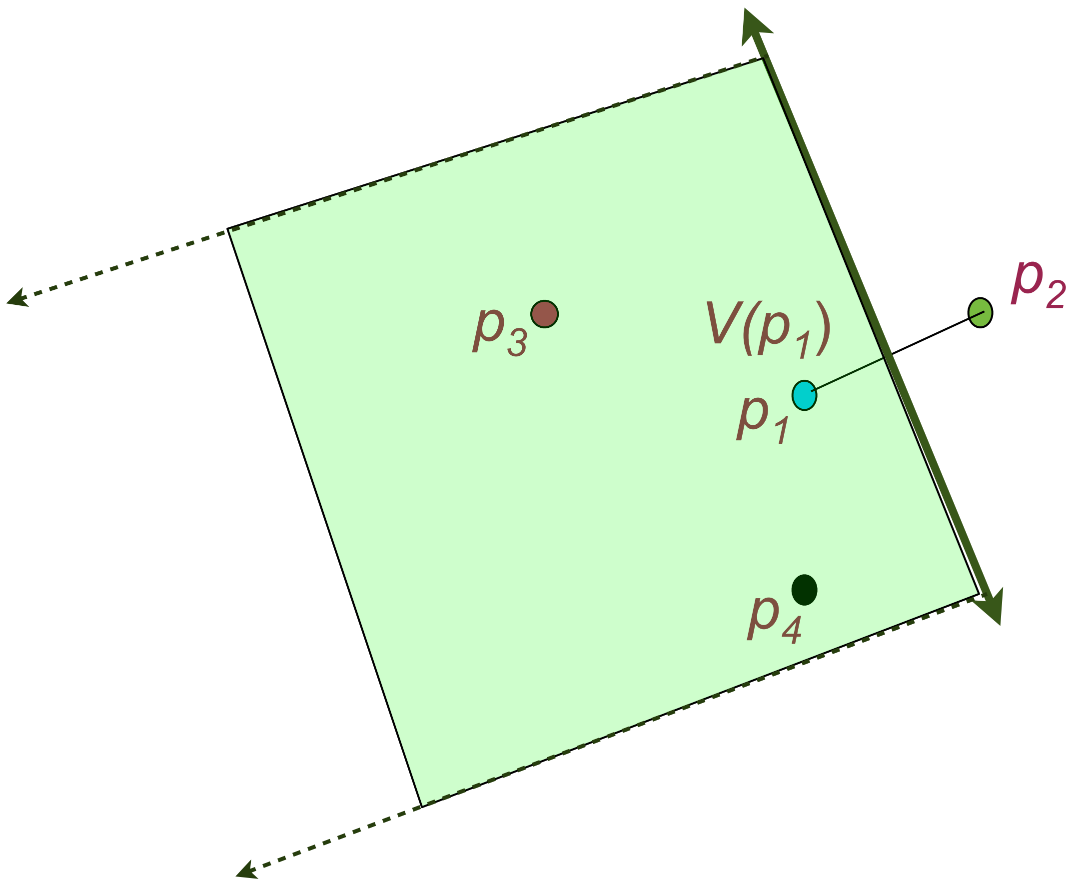
How do we find $V(p_1)$?

Go back

Computing the Voronoi Diagram

How do we find V(p_1)? Go back

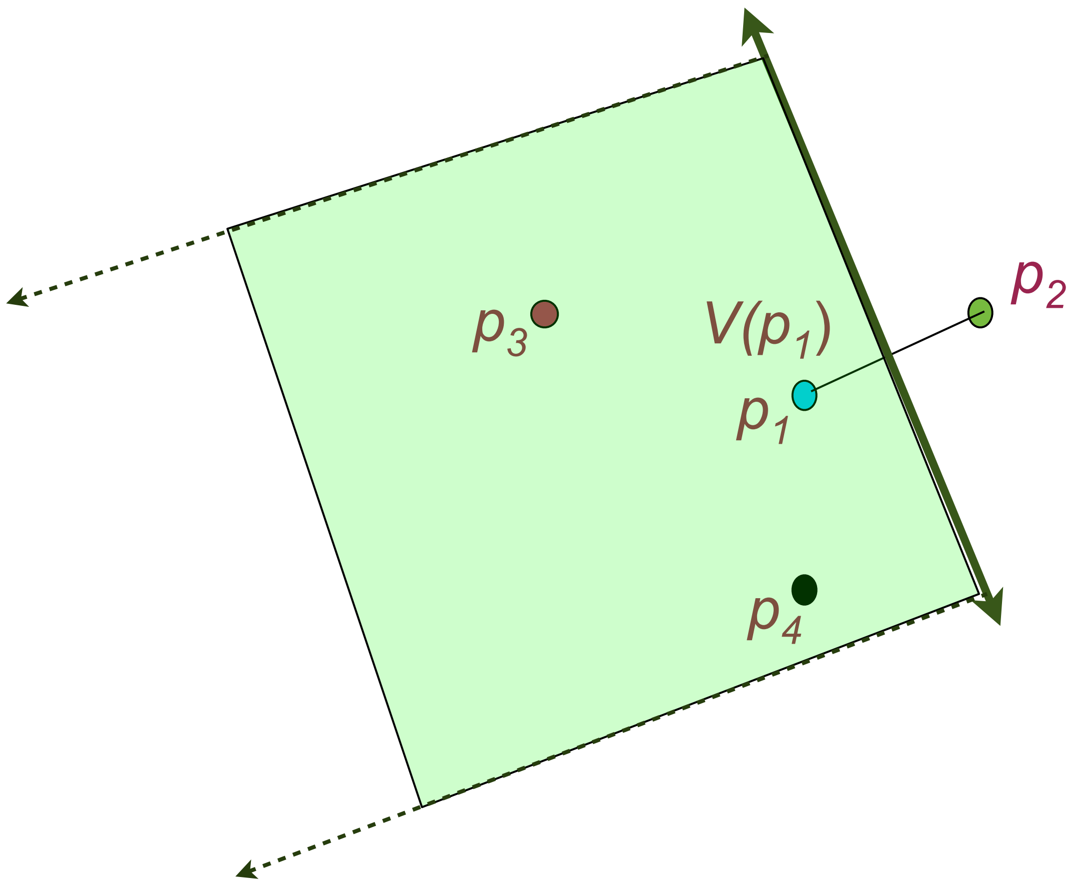
What is this region?



Computing the Voronoi Diagram

How do we find V(p_1)? Go back

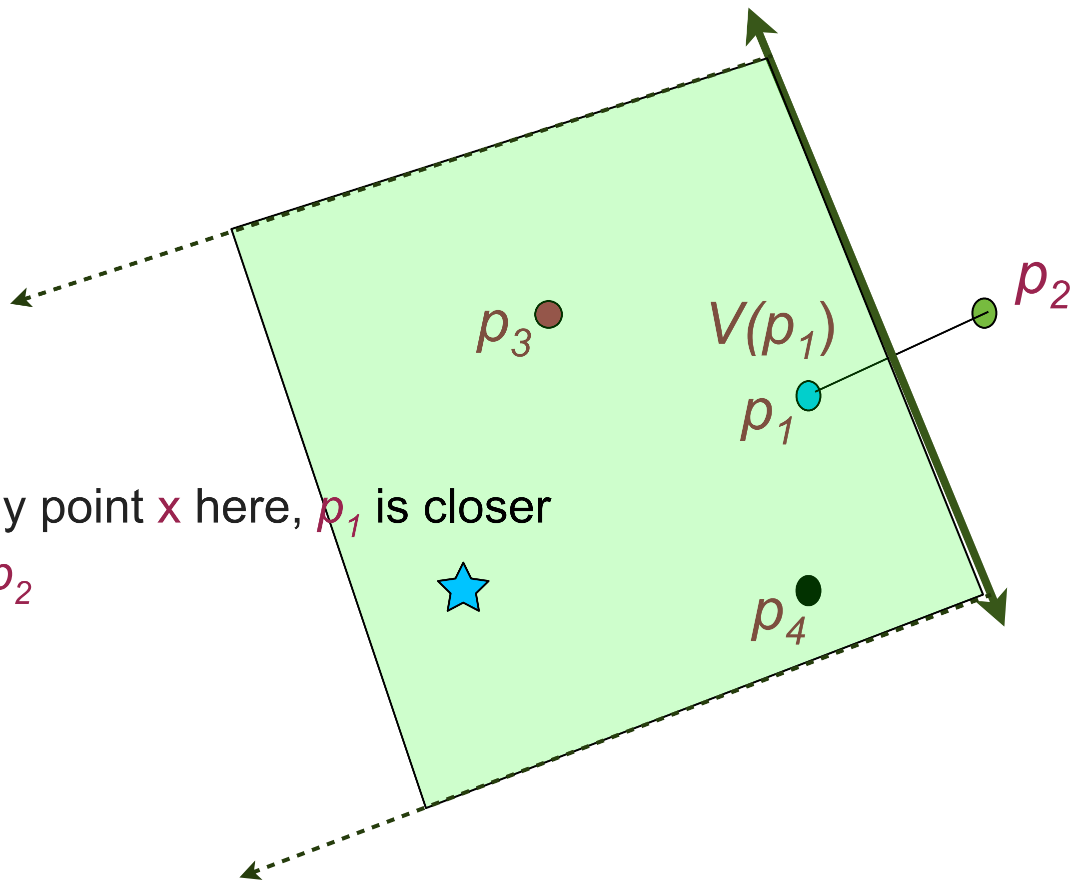
What is this region? Half-plane, say H_1 , containing p_1



Computing the Voronoi Diagram

How do we find V(p_1)? Go back

What is this region? Half-plane, say H_1 , containing p_1

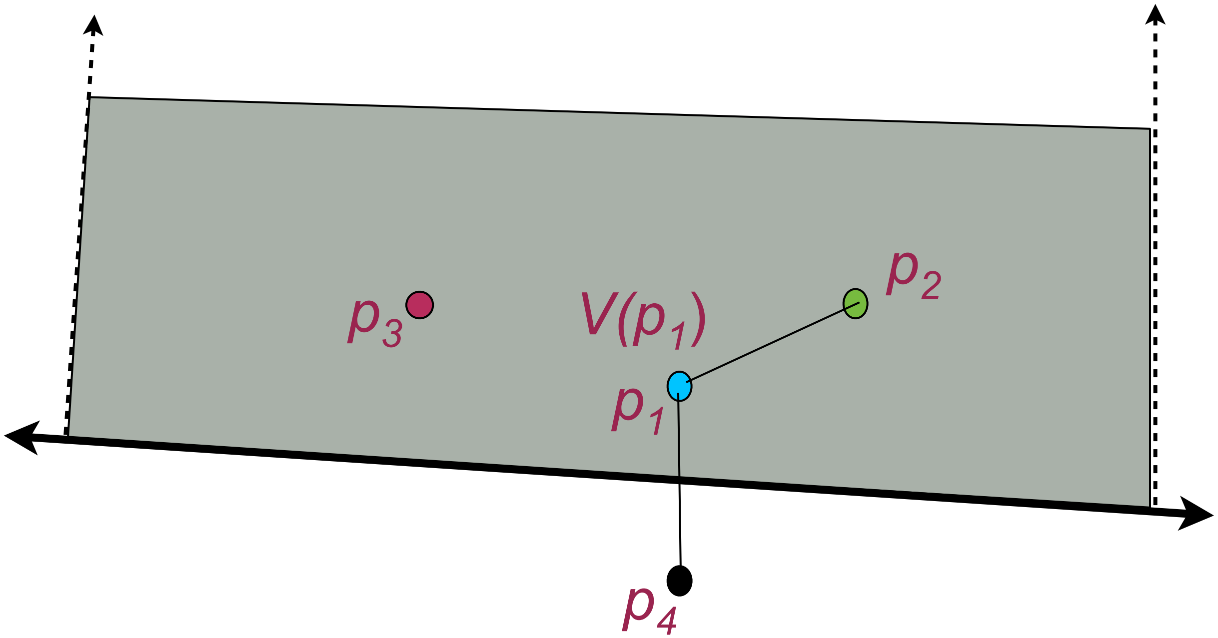


for any point x here, p_1 is closer than p_2

Computing the Voronoi Diagram

How do we find V(p_1)?   Go back

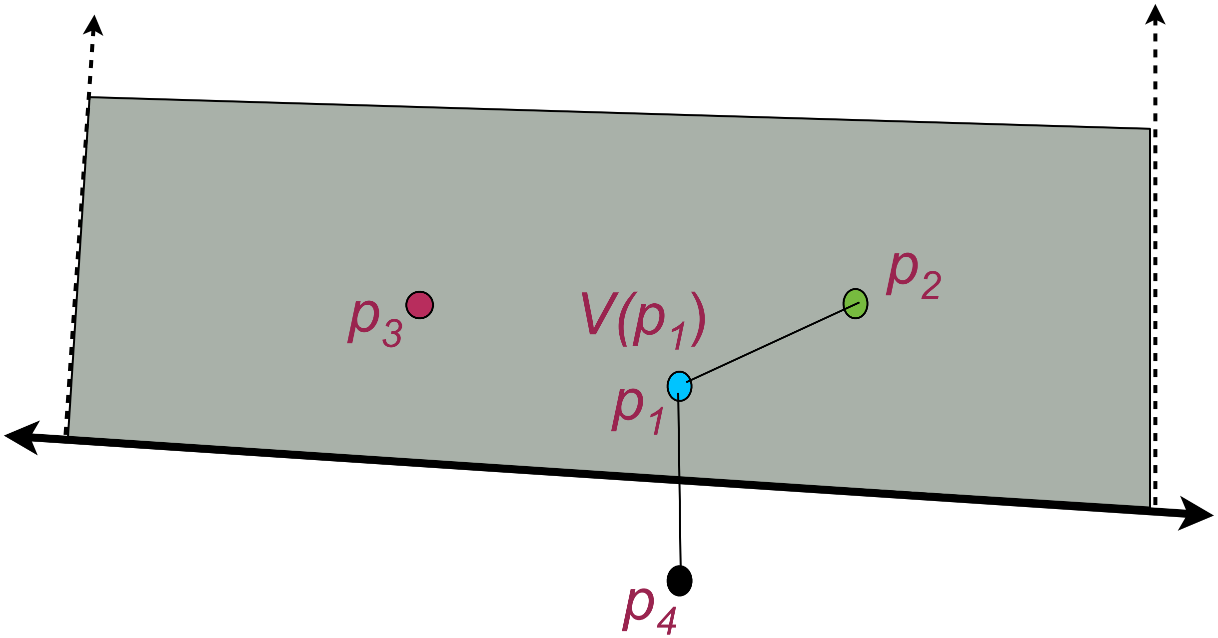
What is this region?



Computing the Voronoi Diagram

How do we find V(p_1)? Go back

What is this region? Half-plane, say H_2 , containing p_1

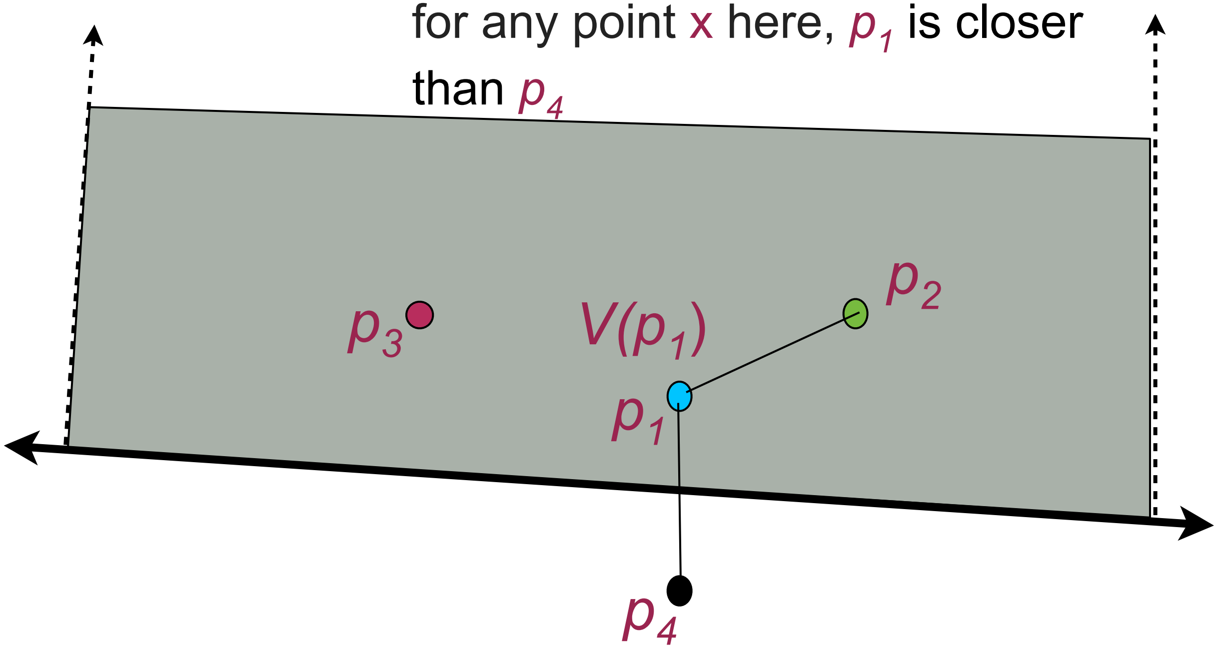


Computing the Voronoi Diagram

How do we find V(p_1)? Go back

What is this region? Half-plane, say H_2 , containing p_1

for any point x here, p_1 is closer than p_4

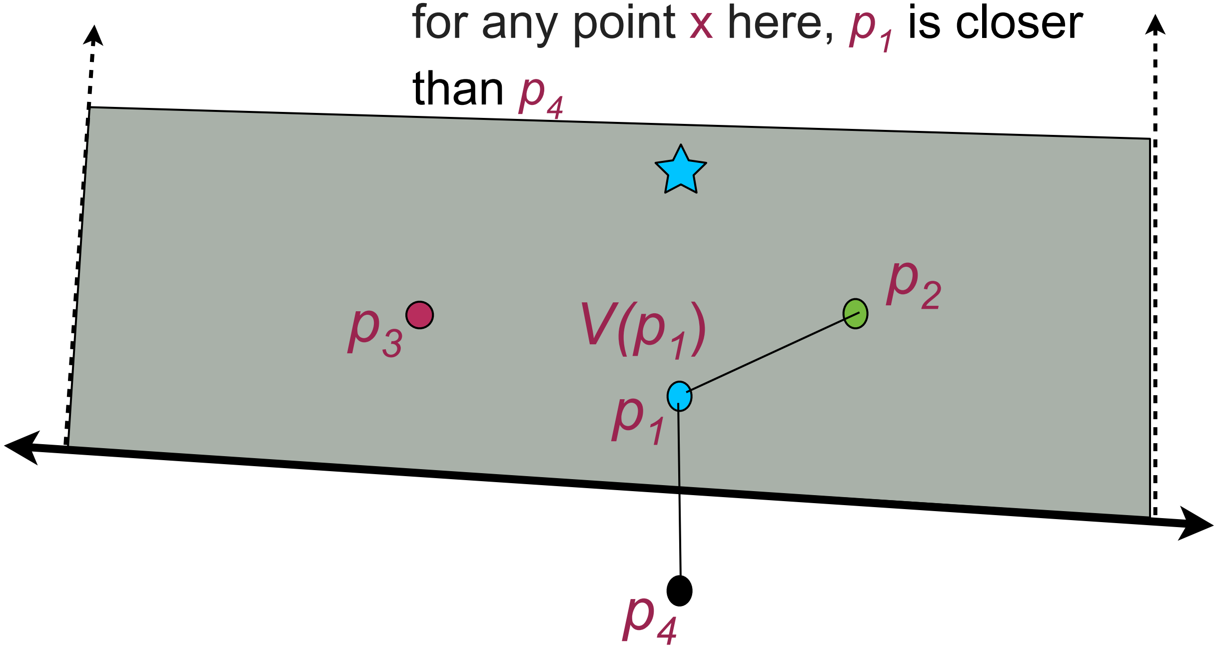


Computing the Voronoi Diagram

How do we find V(p_1)? Go back

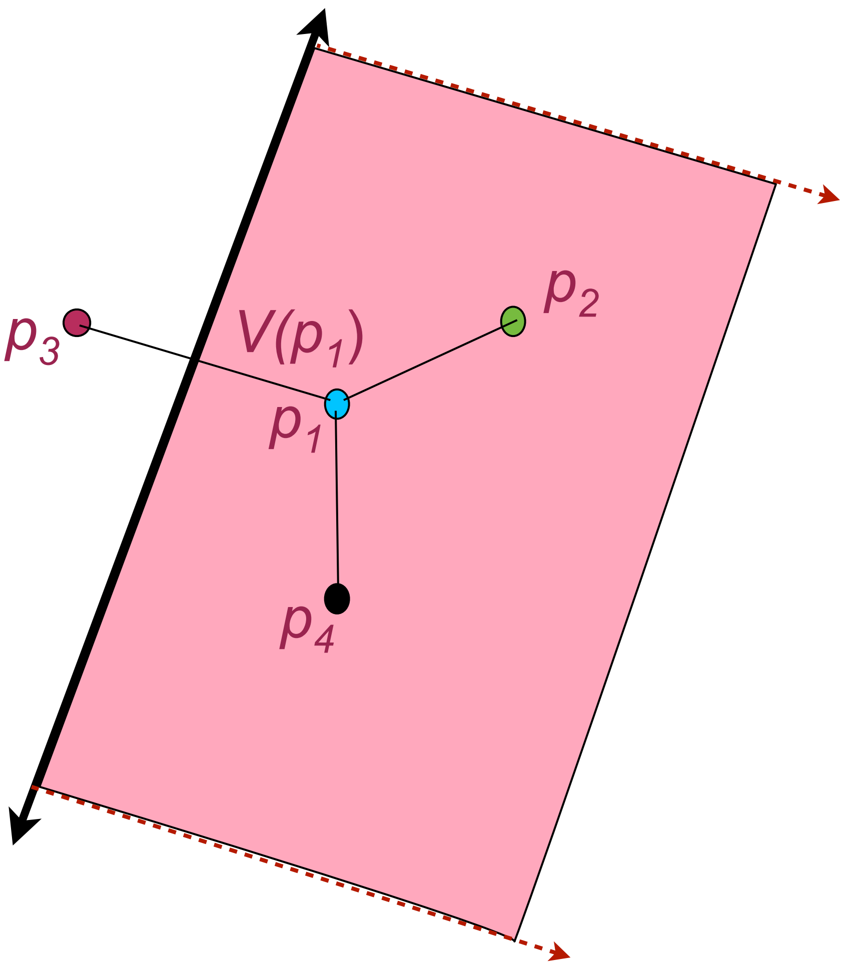
What is this region? Half-plane, say H_2 , containing p_1

for any point x here, p_1 is closer than p_4



Computing the Voronoi Diagram

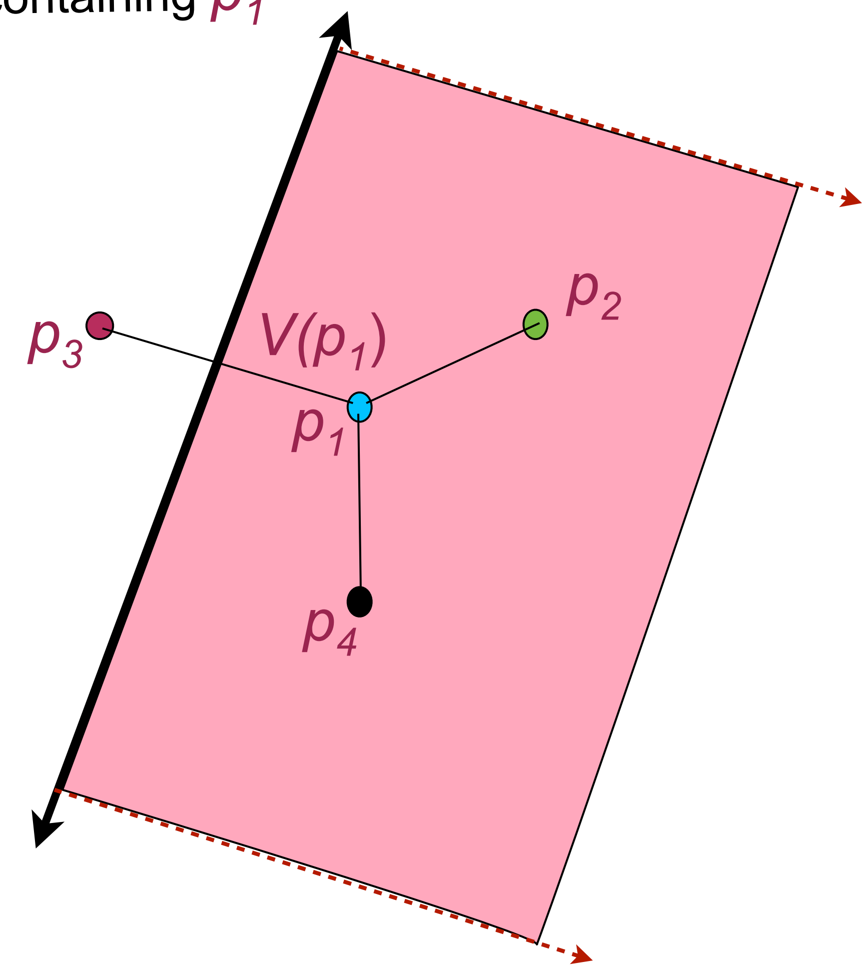
What is this region?



Computing the Voronoi Diagram

What is this region?

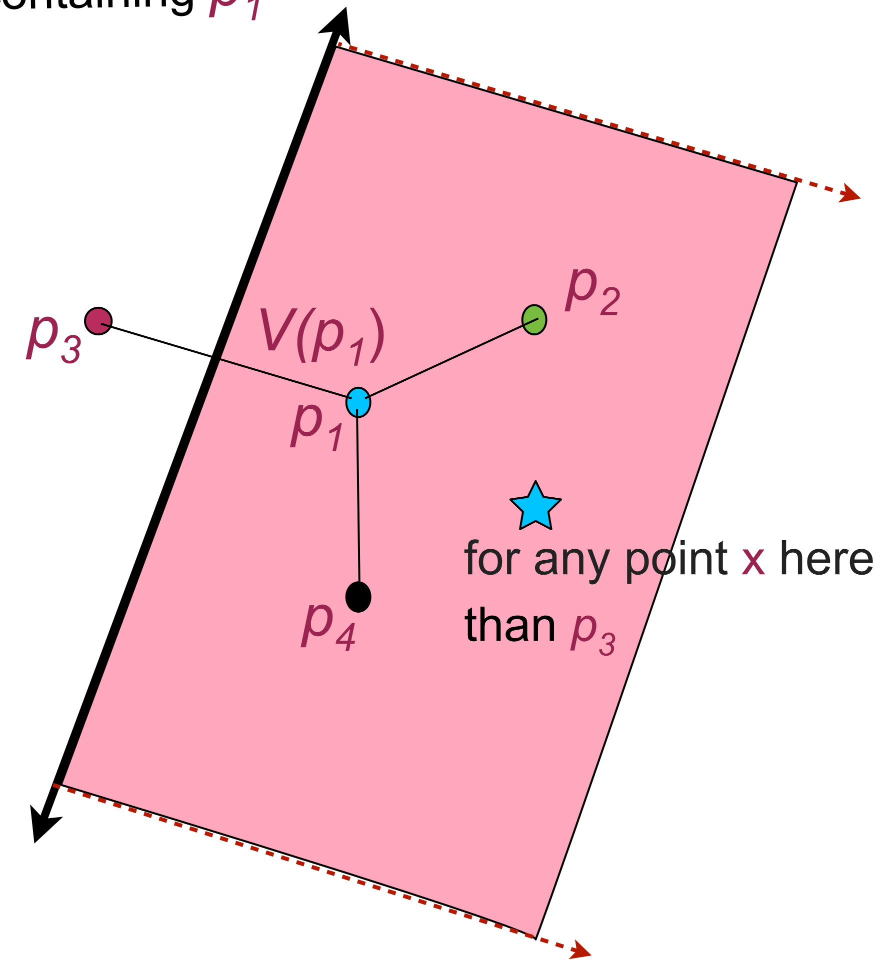
Half-plane, say H_3 , containing p_1



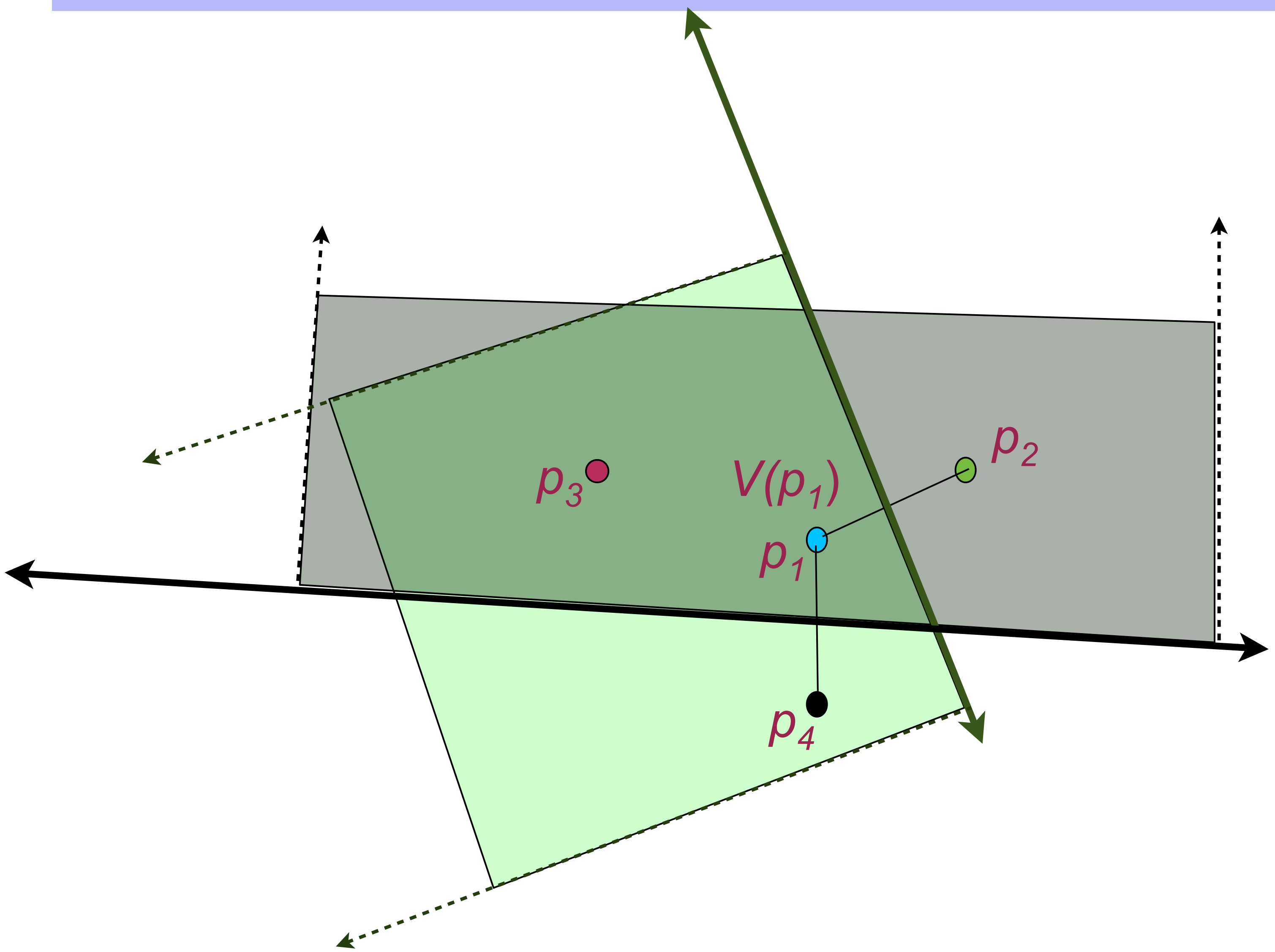
Computing the Voronoi Diagram

What is this region?

Half-plane, say H_3, containing p_1

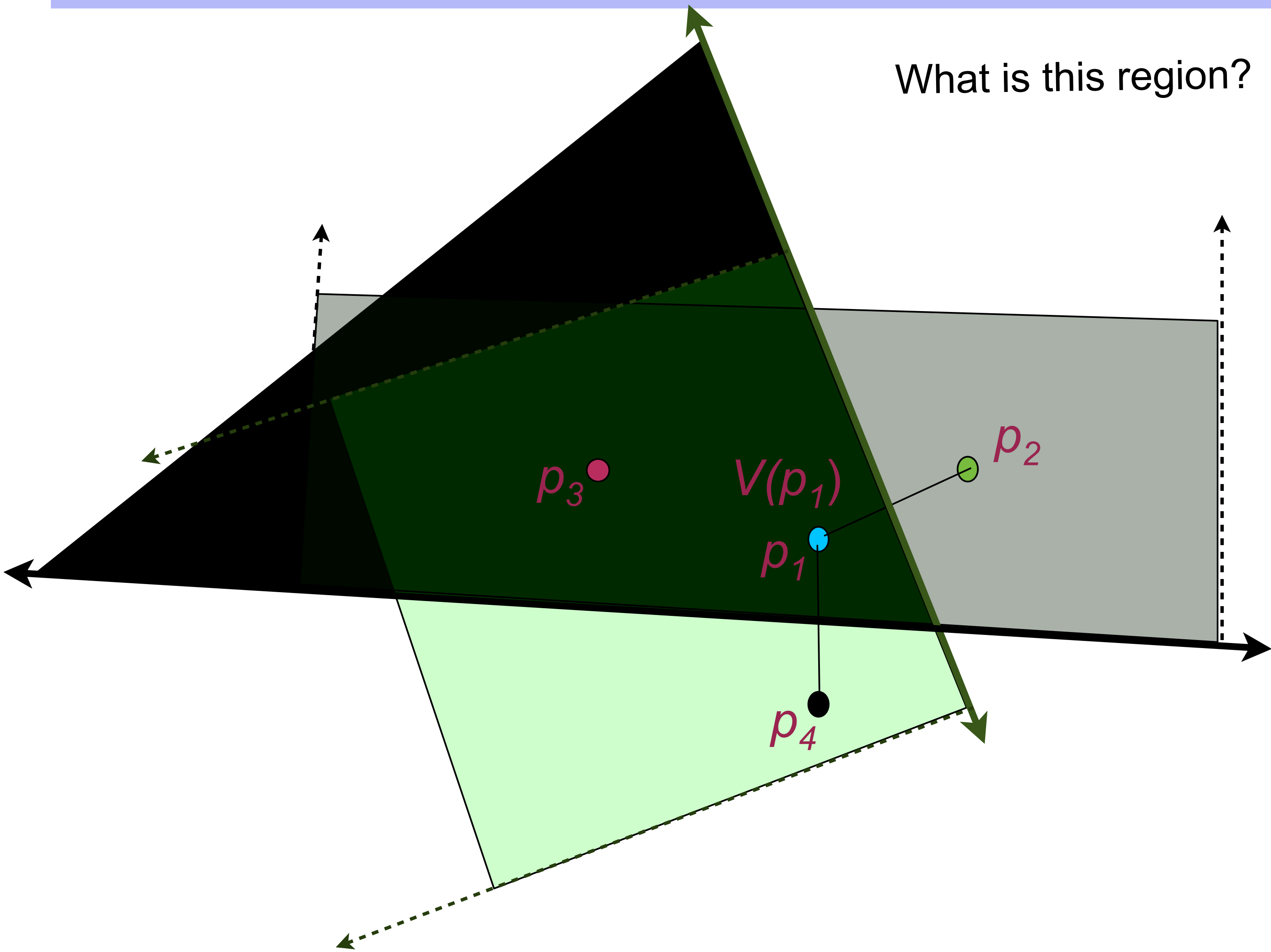


Computing the Voronoi Diagram



Computing the Voronoi Diagram

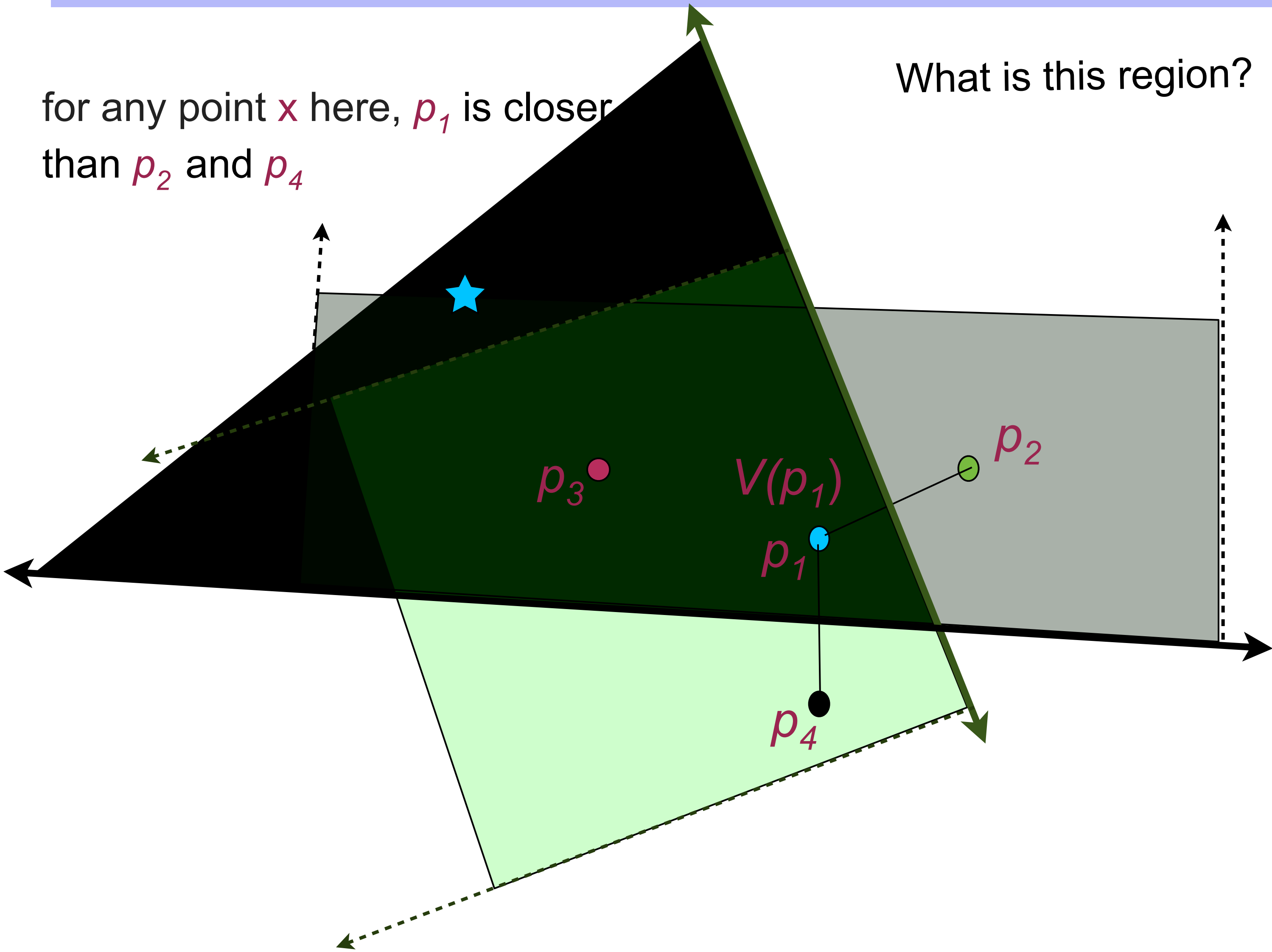
What is this region?



Computing the Voronoi Diagram

for any point x here, p_1 is closer than p_2 and p_4

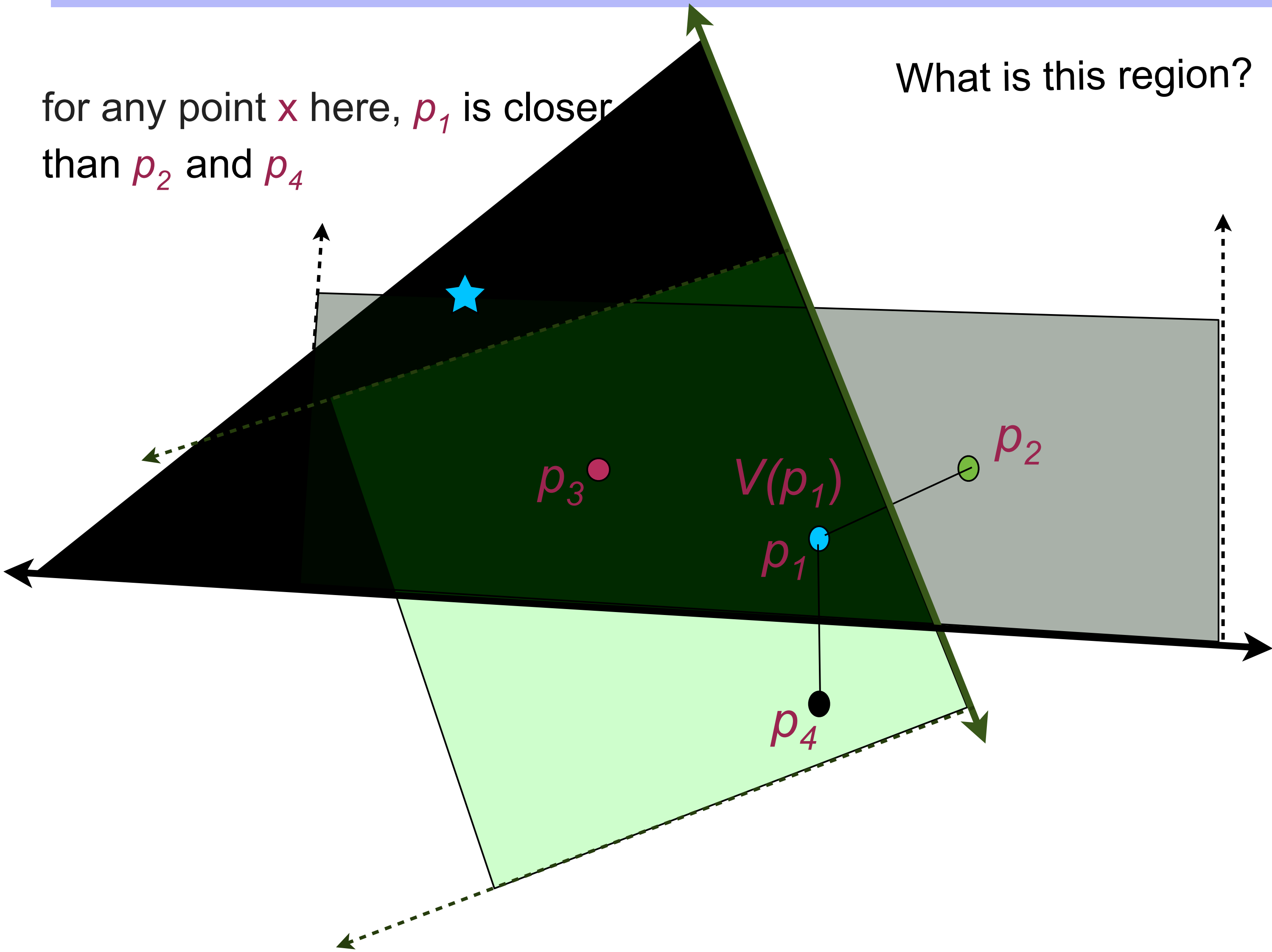
What is this region?



Computing the Voronoi Diagram

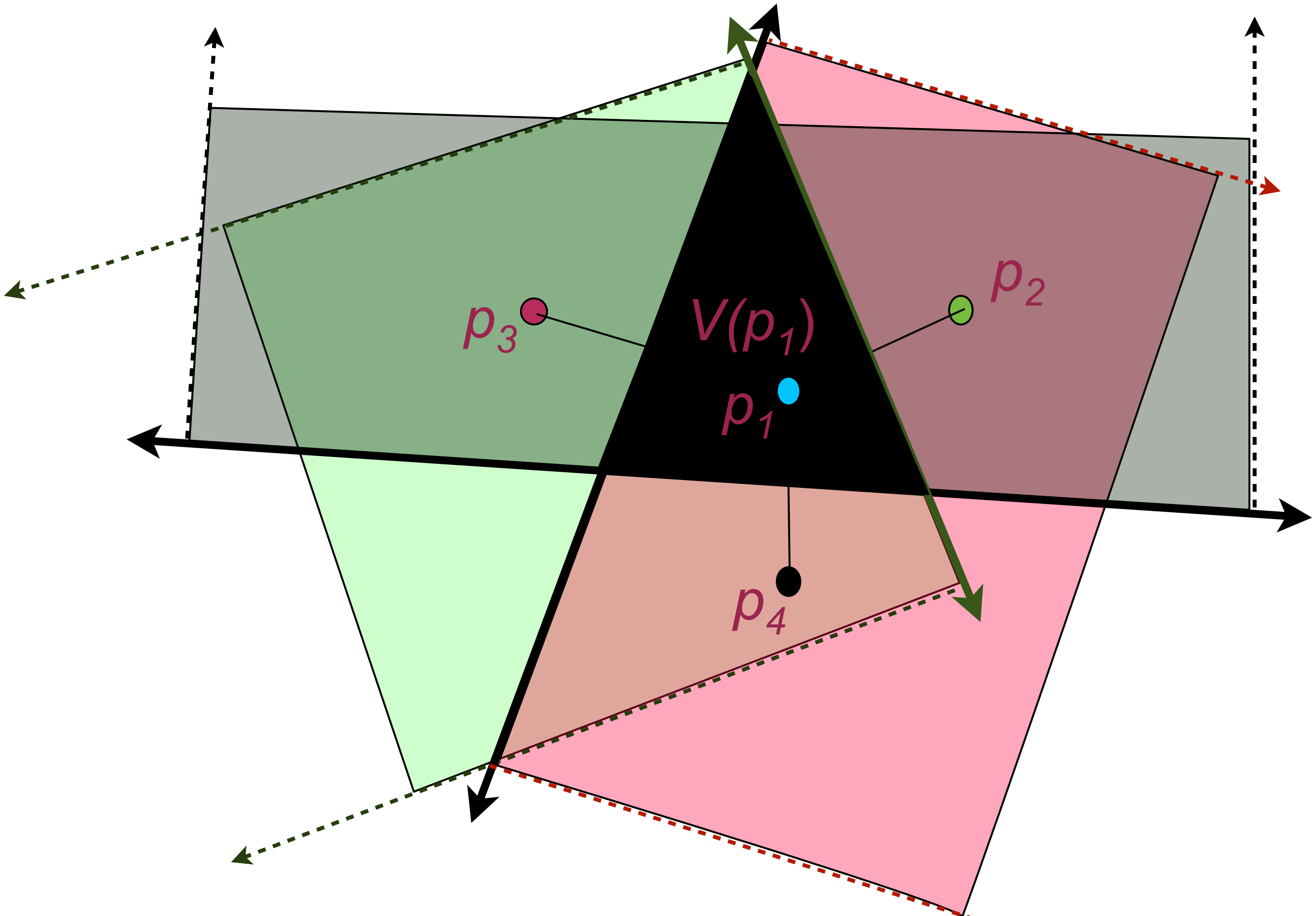
for any point x here, p_1 is closer than p_2 and p_4

What is this region? $H_1 \cap H_2$



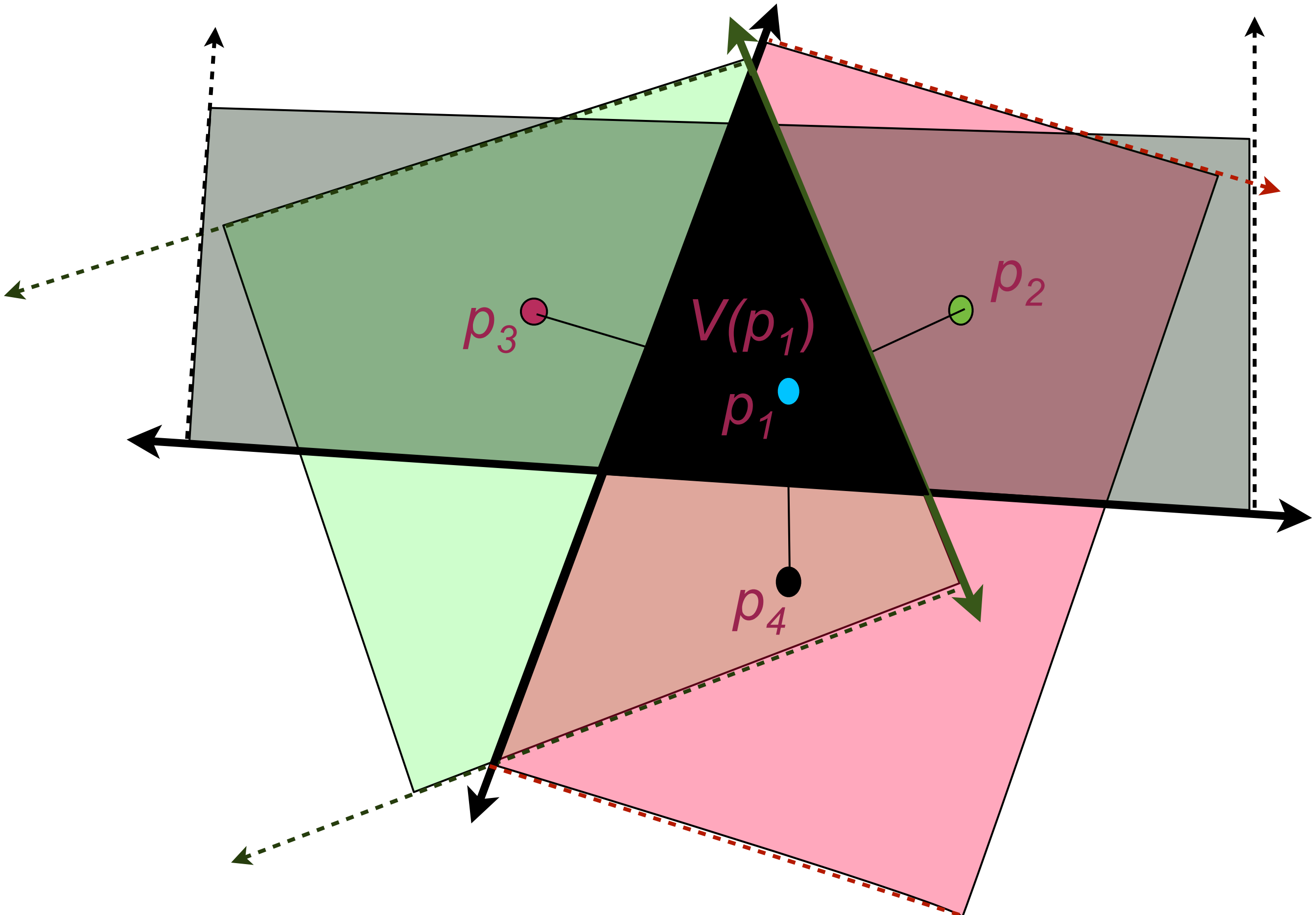
Computing the Voronoi Diagram

What is $V(p_1)$?



Computing the Voronoi Diagram

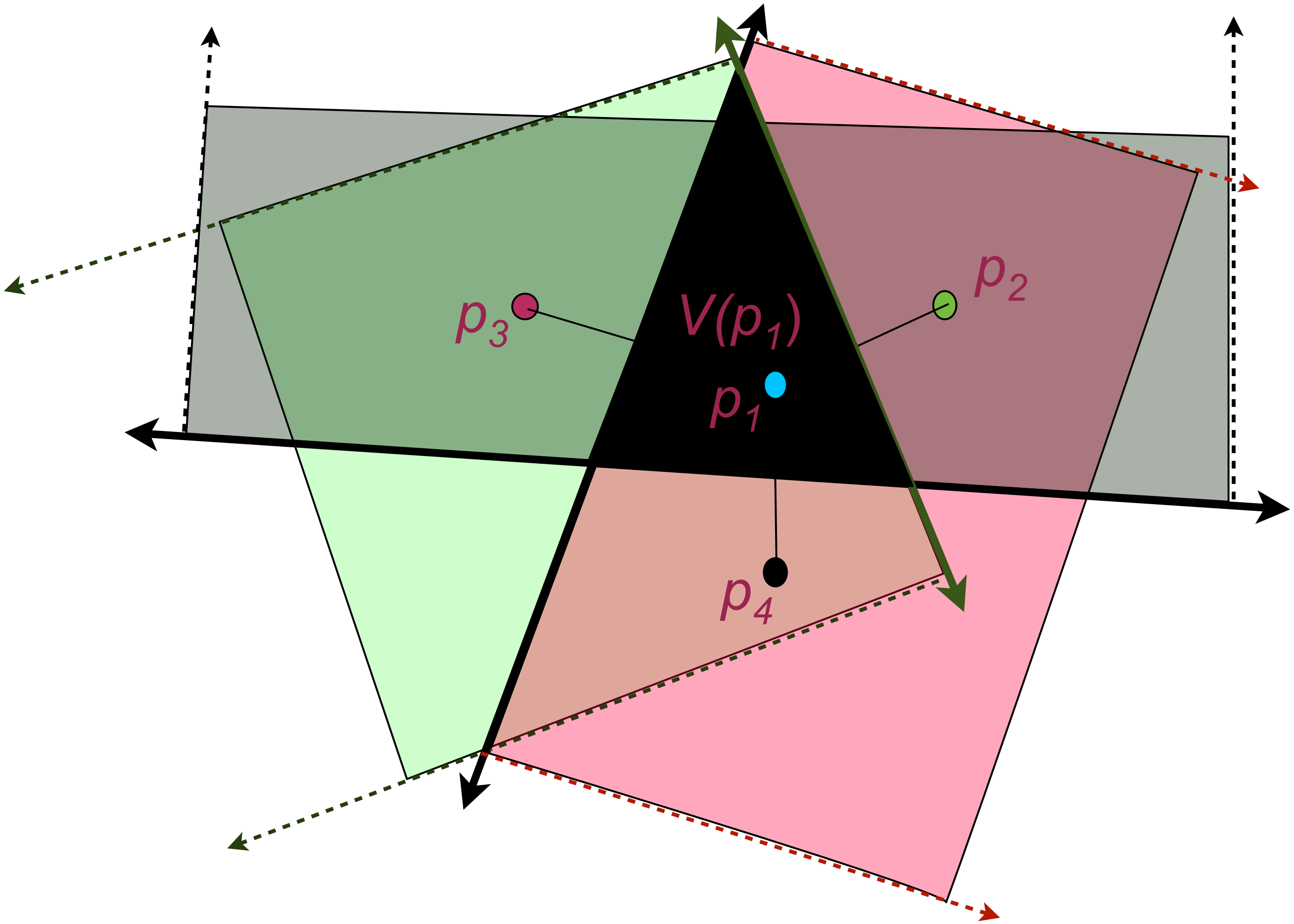
What is V(p_1)? $H_1 \cap H_2 \cap H_3$



Computing the Voronoi Diagram

What is $V(p_1)$? $H_1 \cap H_2 \cap H_3$

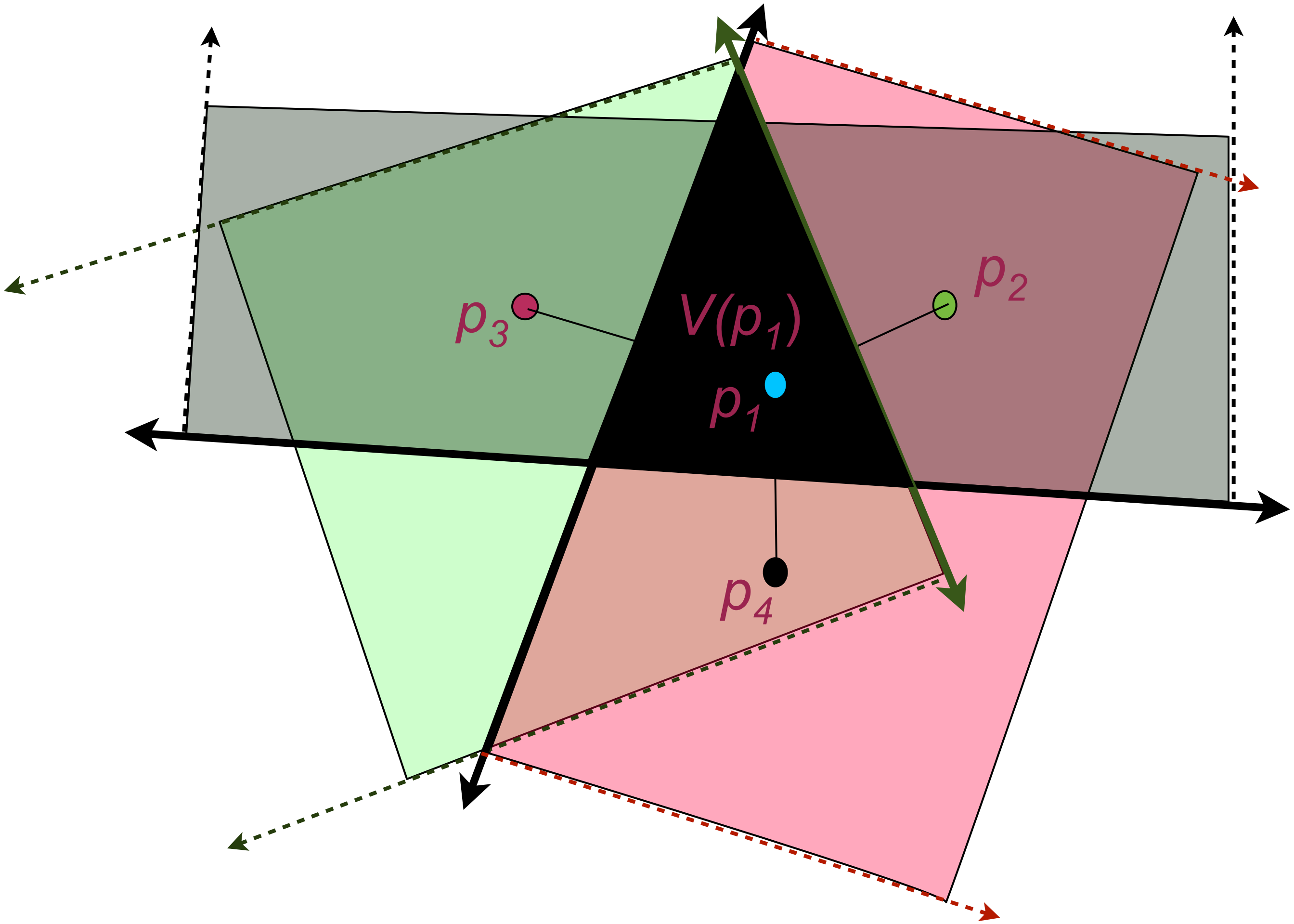
In general, what would be $V(p_1)$?



Computing the Voronoi Diagram

What is V(p_1)? $H_1 \cap H_2 \cap H_3$

In general, what would be V(p_1)? Intersection of (n-1) hyperplanes

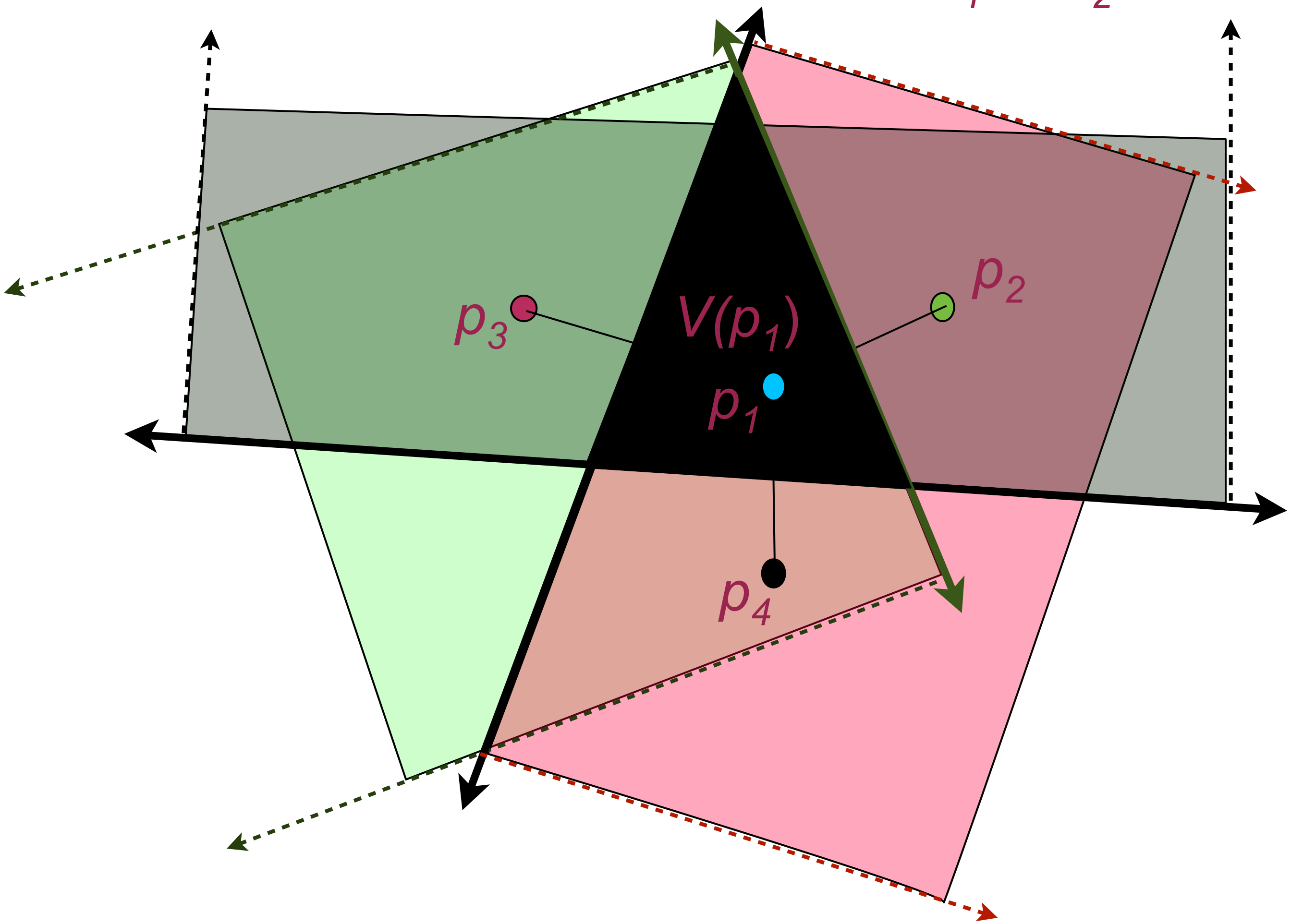


Computing the Voronoi Diagram

What is V(p_1)? $H_1 \cap H_2 \cap H_3$

In general, what would be V(p_1)? Intersection of (n-1) hyperplanes

$H_1 \cap H_2 \cap \ldots \cap H_{n-1}$



Time complexity of this Brute Force Algorithm

Time complexity of this Brute Force Algorithm

Intersection of (n-1) hyperplanes can be found in O(n log n) time

Time complexity of this Brute Force Algorithm

Intersection of (n-1) hyperplanes can be found in O(n log n) time

Total time complexity :

Time complexity of this Brute Force Algorithm

Intersection of (n-1) hyperplanes can be found in $O(n \log n)$ time

Total time complexity : $O(n^2 \log n)$

Time Complexity of Best Algorithms for Voronoi Diagram

Time Complexity of Best Algorithms for Voronoi Diagram

Voronoi Diagram can be constructed in O(n log n) time

\

Time Complexity of Best Algorithms for Voronoi Diagram

Voronoi Diagram can be constructed in O(n log n) time

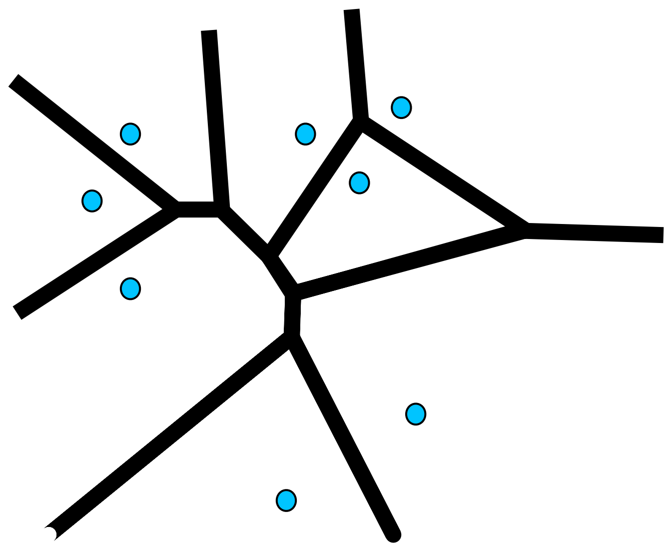
There are well-known algorithms like:

1. Fortune’s Line Sweep
2. Divide and Conquer
3. Lifting points in 3D

Size of the Voronoi Diagram

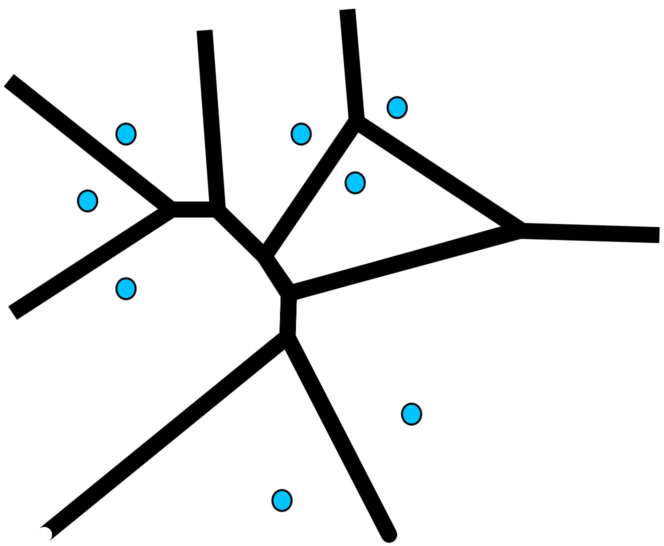
Size of the Voronoi Diagram

Size means: number of vertices, edges and faces



Size of the Voronoi Diagram

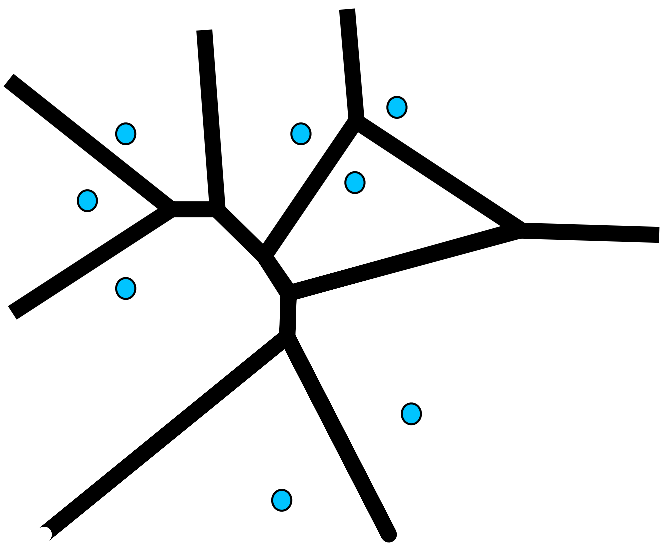
Size means: number of vertices, edges and faces



Lower bound (Smallest Size possible):

Size of the Voronoi Diagram

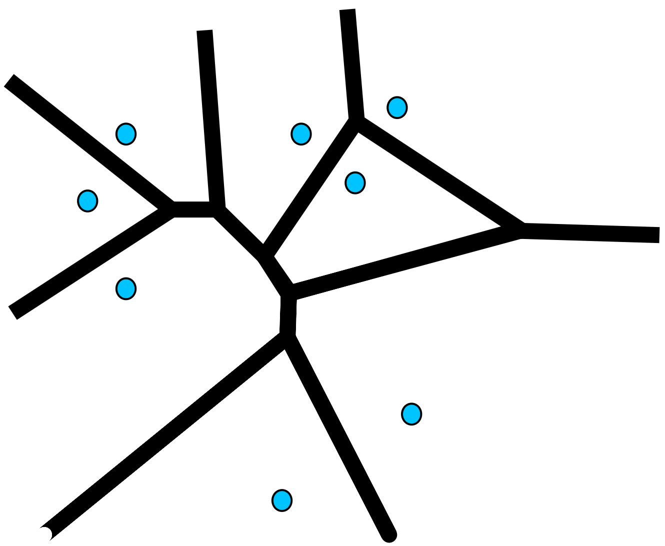
Size means: number of vertices, edges and faces



Lower bound (Smallest Size possible): n, where n is number of sites

Size of the Voronoi Diagram

Size means: number of vertices, edges and faces

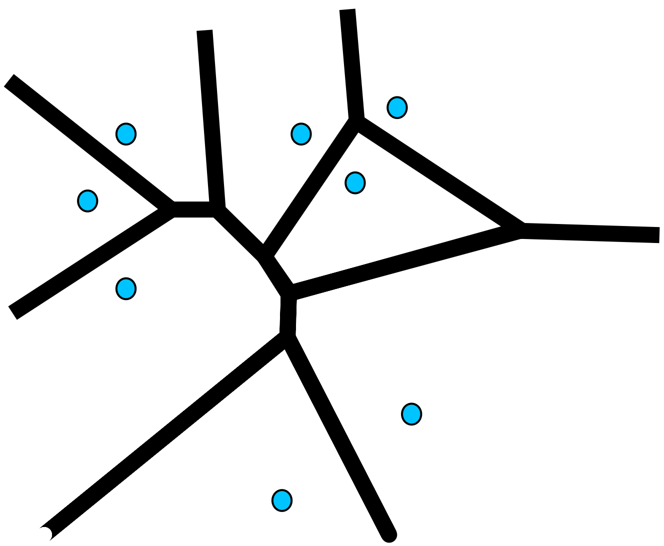


Lower bound (Smallest Size possible): n, where n is number of sites

Trivial Upper bound (Biggest Size possible):

Size of the Voronoi Diagram

Size means: number of vertices, edges and faces

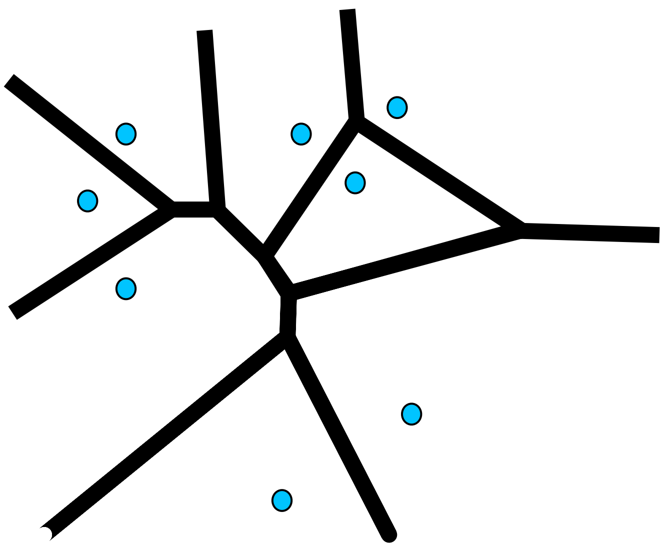


Lower bound (Smallest Size possible): n, where n is number of sites

Trivial Upper bound (Biggest Size possible): O(n log n)

Size of the Voronoi Diagram

Size means: number of vertices, edges and faces



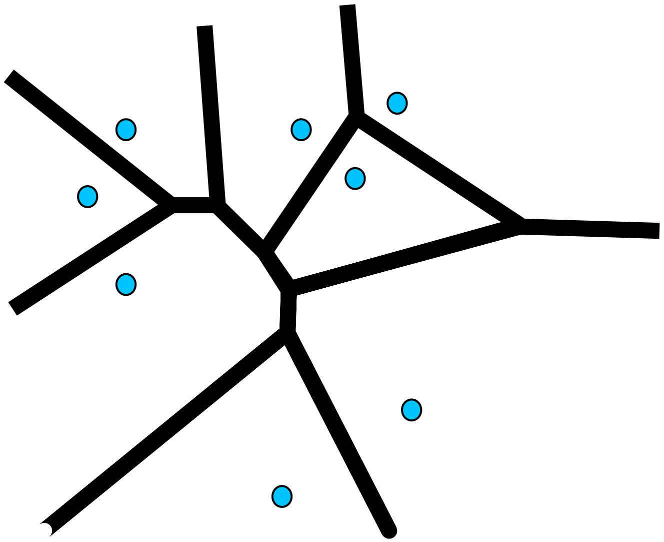
Lower bound (Smallest Size possible): n, where n is number of sites

Trivial Upper bound (Biggest Size possible): O(n log n)

Ultimate Upper Bound (Biggest Size possible):

Size of the Voronoi Diagram

Size means: number of vertices, edges and faces



Lower bound (Smallest Size possible): n, where n is number of sites

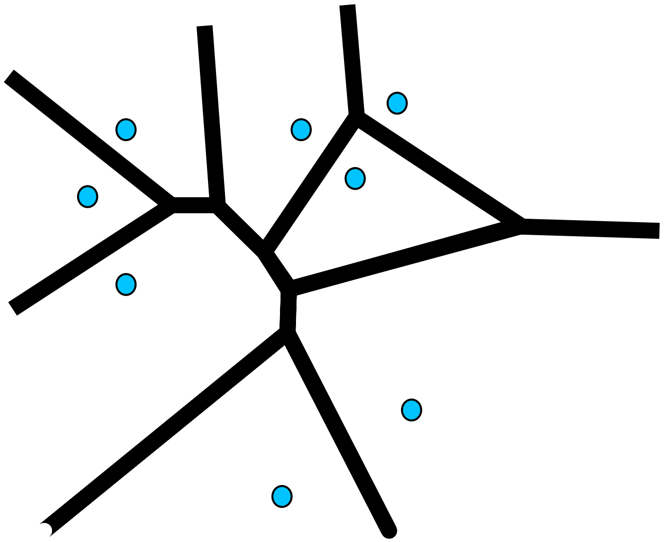
Trivial Upper bound (Biggest Size possible): O(n log n)

Ultimate Upper Bound (Biggest Size possible): O(n)

Why to bother about Size?

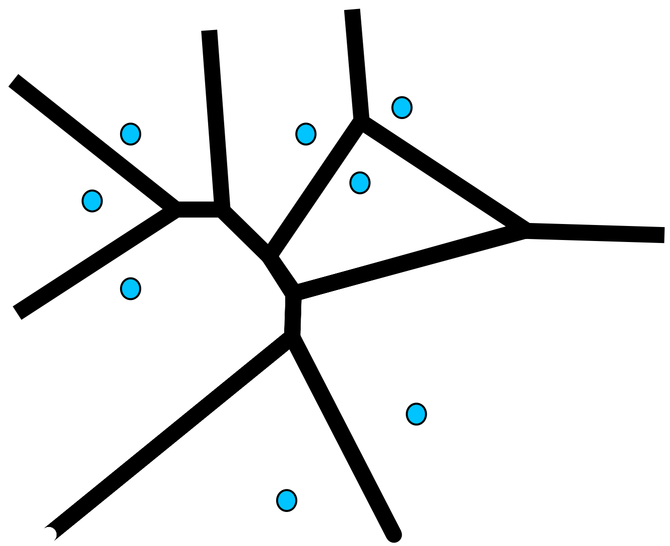
Why to bother about Size?

Voronoi Diagram is



Why to bother about Size?

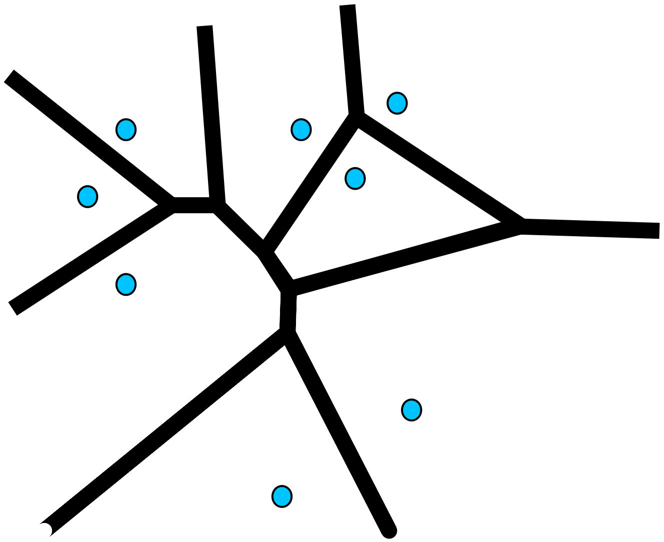
Voronoi Diagram is   Planar Subdivision



Why to bother about Size?

Voronoi Diagram is Planar Subdivision

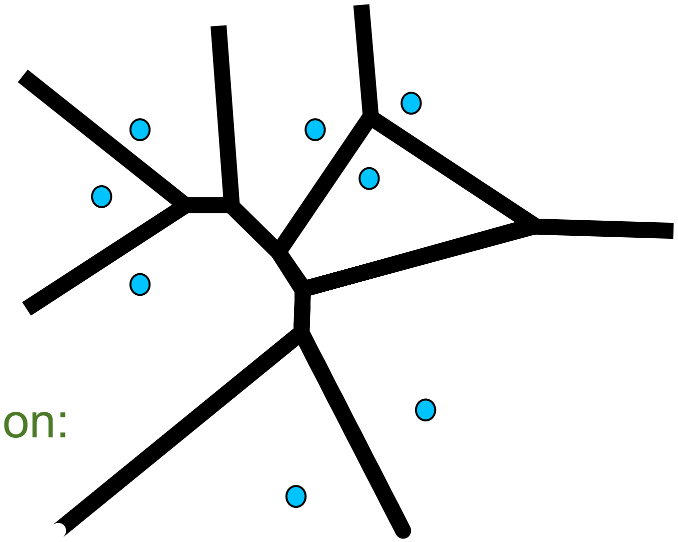
Want to do Planar point Location to get closest point Efficiently



Why to bother about Size?

Voronoi Diagram is Planar Subdivision

Want to do Planar point Location to get closest point Efficiently



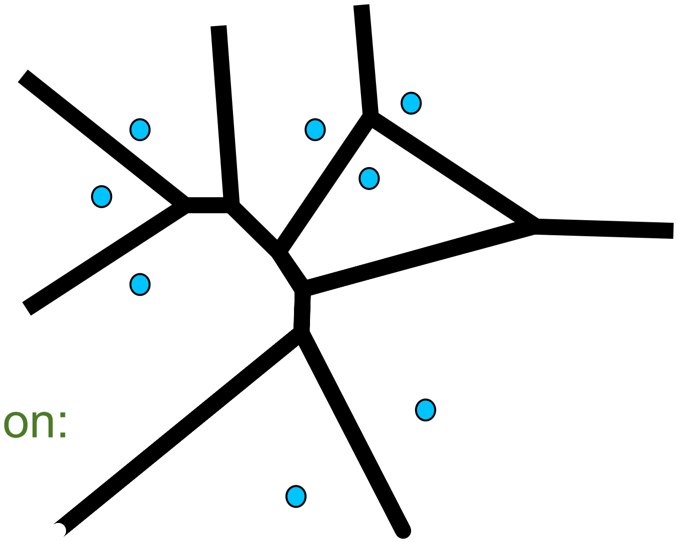
For Planar point Location:

Preprocessing Time:	O(n log n)
Preprocessing space requirement:	O(n)
Query Time:	O(log n)

Why to bother about Size?

Voronoi Diagram is Planar Subdivision

Want to do Planar point Location to get closest point Efficiently



For Planar point Location:

Preprocessing Time: O(n log n)

Preprocessing space requirement: O(n)

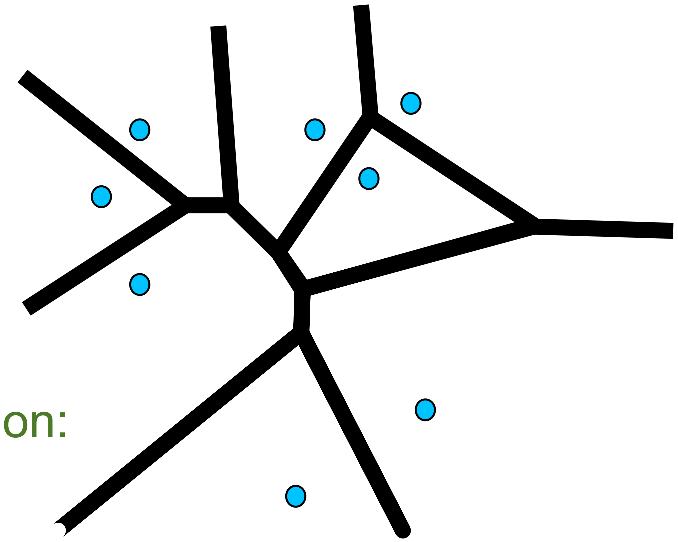
Query Time: O(log n)

But there is a big if, What is that if?

Why to bother about Size?

Voronoi Diagram is   Planar Subdivision

Want to do Planar point Location to get closest point Efficiently



For Planar point Location:

Preprocessing Time: O(n log n)

Preprocessing space requirement: O(n)

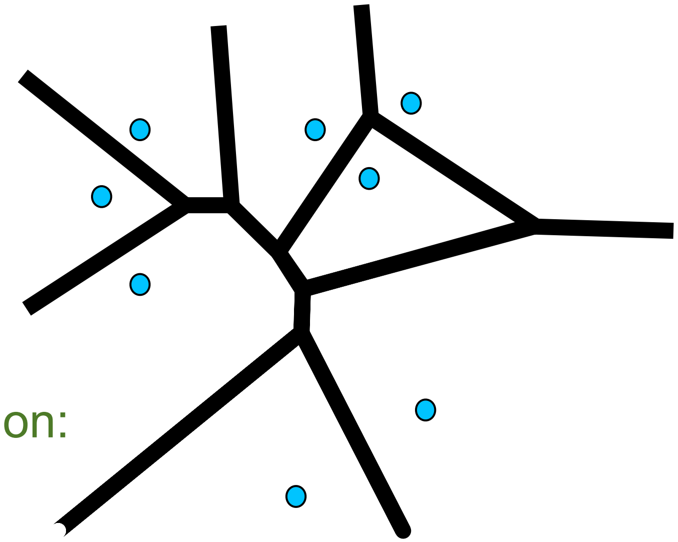
Query Time: O(log n)

But there is a big if, What is that if? The size of planar subdivision=

Why to bother about Size?

Voronoi Diagram is Planar Subdivision

Want to do Planar point Location to get closest point Efficiently



For Planar point Location:

Preprocessing Time: O(n log n)

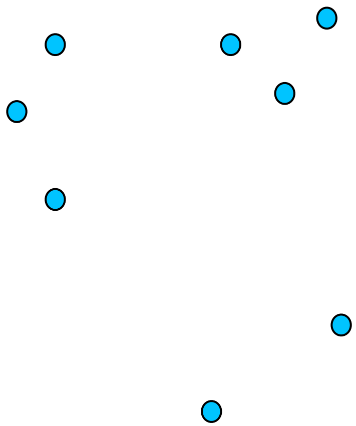
Preprocessing space requirement: O(n)

Query Time: O(log n)

But there is a big if, What is that if? The size of planar subdivision= O(n)

Summary

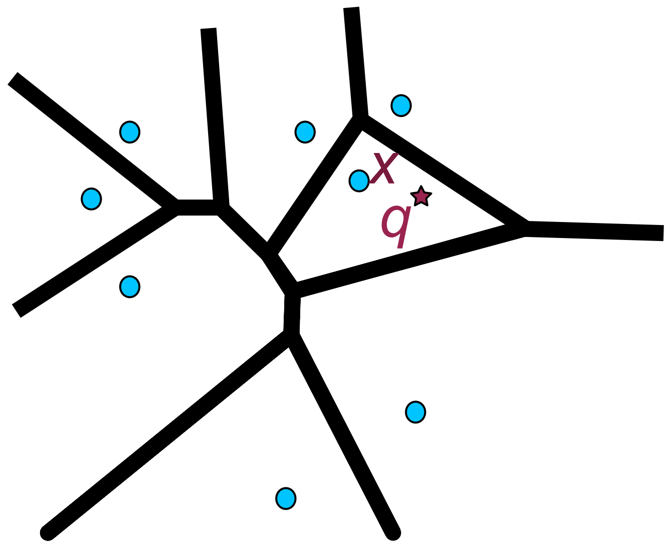
P $\rightarrow$ A set of n distinct points (Geometric Objects) in the plane.



Summary

P $\rightarrow$ A set of n distinct points (Geometric Objects) in the plane.

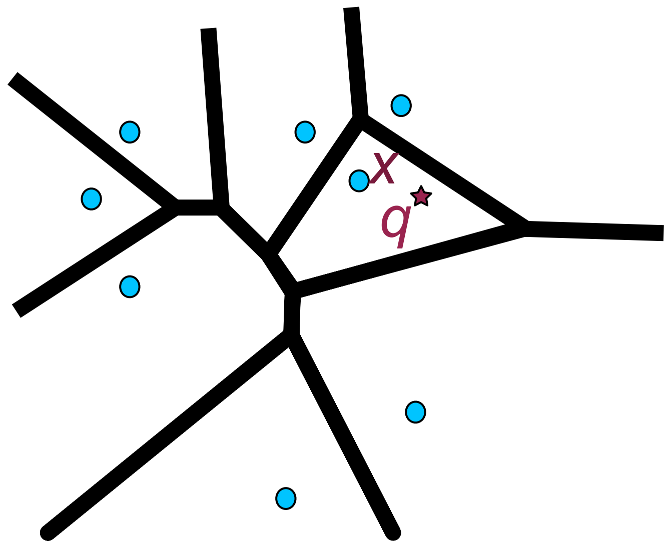
We can Preprocess P such that closest point $x \in P$ of any query point q can be found in $O(\log n)$ time Using Planar point location



Summary

P $\rightarrow$ A set of n distinct points (Geometric Objects) in the plane.

We can Preprocess P such that closest point $x \in P$ of any query point q can be found in $O(\log n)$ time Using Planar point location

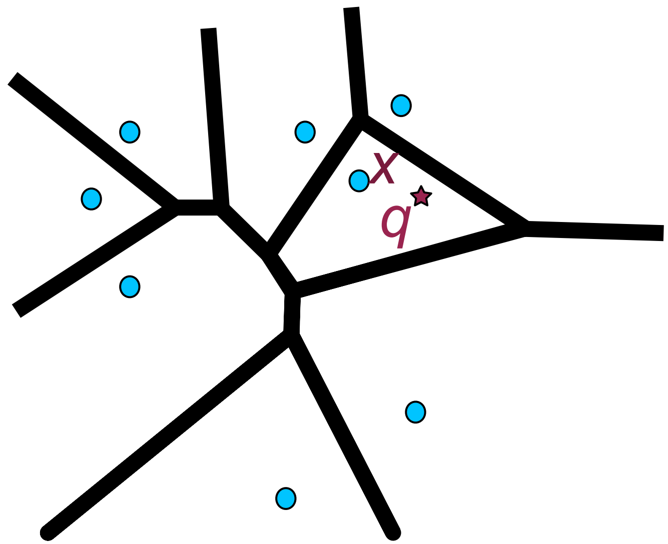


Preprocess structure is called Voronoi Diagram V(P)

Summary

P $\rightarrow$ A set of n distinct points (Geometric Objects) in the plane.

We can Preprocess P such that closest point $x \in P$ of any query point q can be found in O(log n) time Using Planar point location

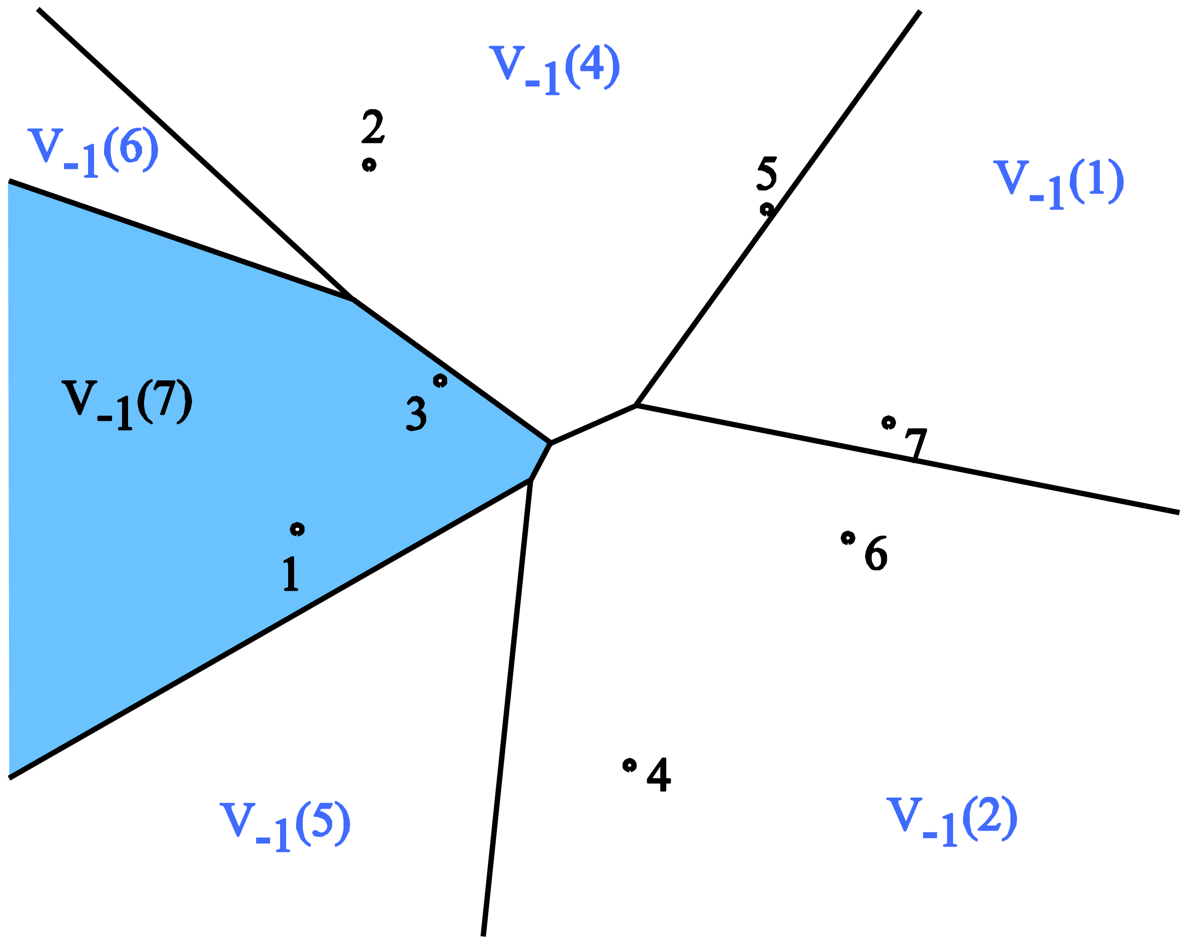


Preprocess structure is called Voronoi Diagram V(P)

V(P) can be constructed in O(n log n) time and can be stored in O(n) space

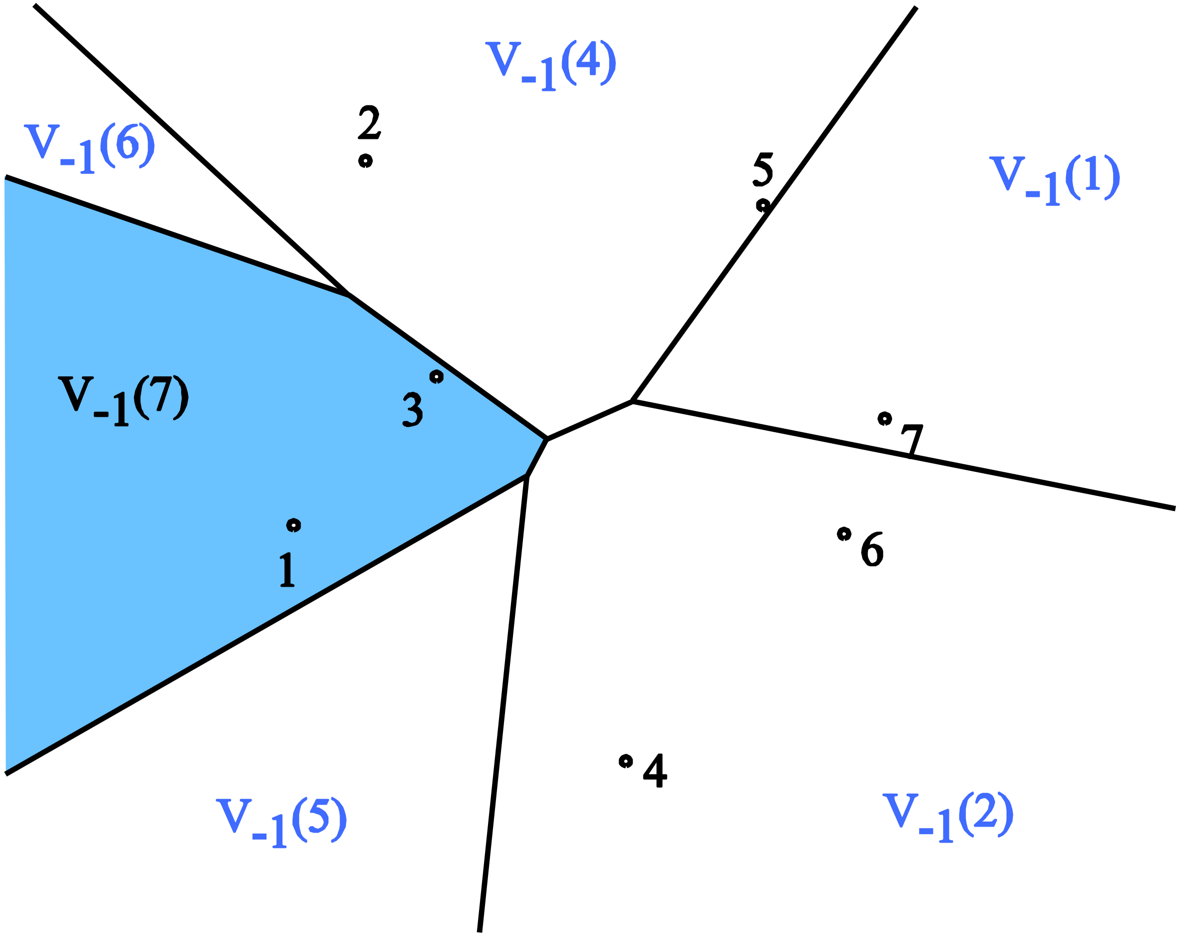
Other Kind of Voronoi Diagrams

Furthest Point Voronoi Diagram



Furthest Point Voronoi Diagram

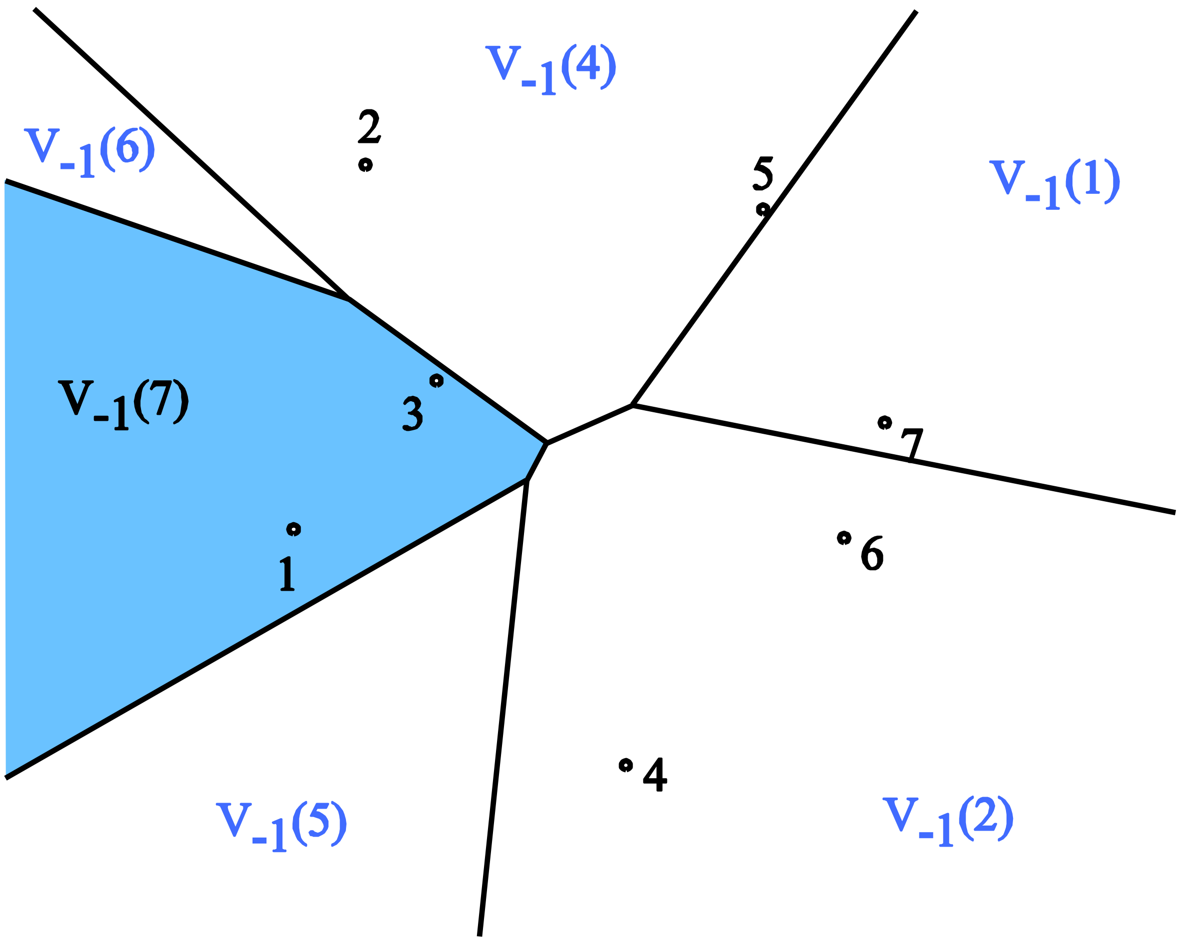
FV(P): the partition of the plane formed by the farthest point Voronoi regions, their edges, and vertices



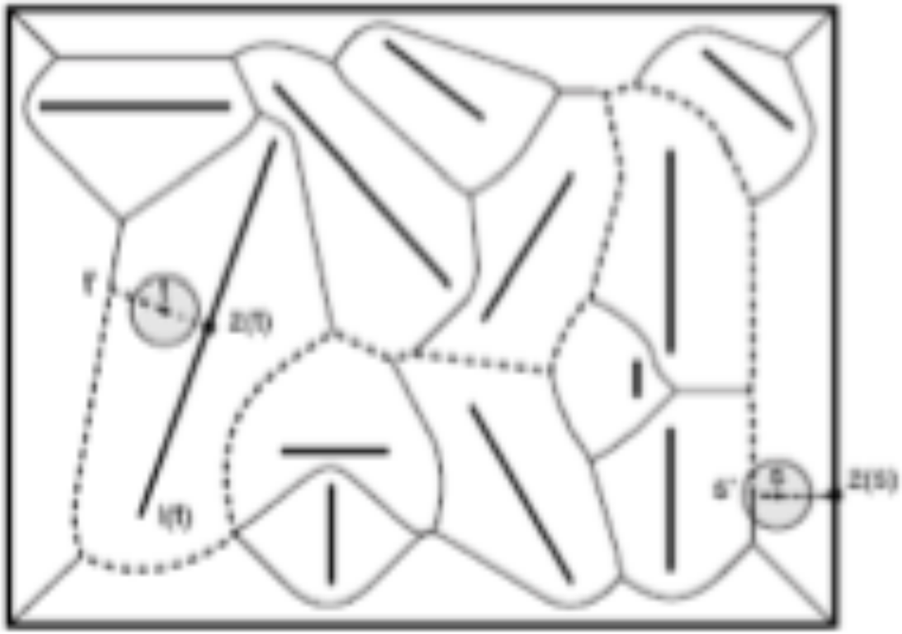
Furthest Point Voronoi Diagram

FV(P): the partition of the plane formed by the farthest point Voronoi regions, their edges, and vertices

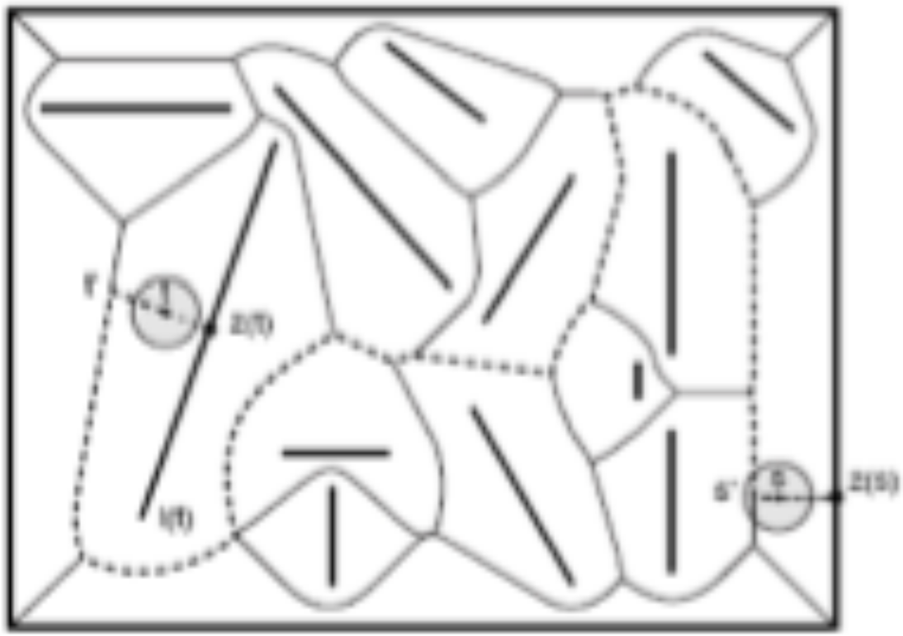
$V_{-1}(p_i)$: the set of point of the plane farther from p_i than from any other site



Voronoi diagram for line segments



Voronoi diagram for line segments



Moving a disk from *s* to *t* in the presence of barriers

Organization of the Talk

1. Preliminaries
2. Generic Definition
3. Some Technical Details
4. Conclusion

Organization of the Talk

1. Preliminaries
2. Generic Definition
3. Some Technical Details
4. Conclusion

Conclusion

Conclusion

Voronoi Diagram is a very Fundamental Interesting Geometric Structure

Conclusion

Voronoi Diagram is a very Fundamental Interesting Geometric Structure

This has wide range of application to solve different problem in Geometry, Facility Location, Engineering Sciences, Biological Sciences, Nano Sciences to name a few

Conclusion

Voronoi Diagram is a very Fundamental Interesting Geometric Structure

This has wide range of application to solve different problem in Geometry, Facility Location, Engineering Sciences, Biological Sciences, Nano Sciences to name a few

The Applications of this structure are so wide that

Conclusion

Voronoi Diagram is a very Fundamental Interesting Geometric Structure

This has wide range of application to solve different problem in Geometry, Facility Location, Engineering Sciences, Biological Sciences, Nano Sciences to name a few

The Applications of this structure are so wide that

There is dedicated Symposiums on Voronoi Diagram:

INTERNATIONAL SYMPOSIUM on VORONOI DIAGRAMS in science and engineering

Thank You