ALGORITHMS FOR CONSTRUCTING VORONOI DIAGRAMS

Vera Sacristán

Computational Geometry
Facultat d’Informàtica de Barcelona
Universitat Politècnica de Catalunya
Naive algorithm
Constructing Voronoi diagrams

NAIVE ALGORITHM
Constructing Voronoi diagrams

NAIVE ALGORITHM

For each $p_i$, construct its Voronoi region $Vor(p_i) = \bigcap_{j \neq i} H_{ij}$. 
Constructing Voronoi diagrams

NAIVE ALGORITHM

For each $p_i$, construct its Voronoi region $Vor(p_i) = \bigcap_{j \neq i} H_{ij}$.

Inconvenients:

- It can cause inconsistency due to precision problems
Constructing Voronoi diagrams

NAIVE ALGORITHM

For each $p_i$, construct its Voronoi region $Vor(p_i) = \bigcap_{j \neq i} H_{ij}$.

Inconvenients:

- It can cause inconsistency due to precision problems
- It does not produce immediate neighborhood information
Constructing Voronoi diagrams

NAIVE ALGORITHM

For each $p_i$, construct its Voronoi region $Vor(p_i) = \bigcap_{j \neq i} H_{ij}$.

Inconvenients:

- It can cause inconsistency due to precision problems
- It does not produce immediate neighborhood information
- It runs in $O(n^2 \log n)$ time
NAIVE ALGORITHM

For each $p_i$, construct its Voronoi region $Vor(p_i) = \bigcap_{j \neq i} H_{ij}$.

Inconvenients:

- It can cause inconsistency due to precision problems
- It does not produce immediate neighborhood information
- It runs in $O(n^2 \log n)$ time

The fact that each Voronoi region, $Vor(p_i)$, is built in optimal $\Theta(n \log n)$ time does not imply that the construction of the entire diagram, $Vor(P)$, requires $\Omega(n^2 \log n)$ time, as we will see.
incremental algorithm
INCREMENTAL ALGORITHM
INCREMENTAL ALGORITHM

Starting with the Voronoi diagram of \( \{p_1, \ldots, p_i\} \)...
INCREMENTAL ALGORITHM

Starting with the Voronoi diagram of \{p_1, \ldots, p_i\}...

... add point \(p_{i+1}\)
**INCREMENTAL ALGORITHM**

Starting with the Voronoi diagram of \( \{p_1, \ldots, p_i\} \)...  

... add point \( p_{i+1} \)

... compute its region
INCREMENTAL ALGORITHM

Starting with the Voronoi diagram of \( \{p_1, \ldots, p_i\} \)...

... add point \( p_{i+1} \)

... compute its region

... and prune the initial diagram.
INCREMENTAL ALGORITHM

Starting with the Voronoi diagram of \( \{p_1, \ldots, p_i\} \)...

... add point \( p_{i+1} \)

Explore all candidates to find the site \( p_j \) \((1 \leq j \leq i)\) closest to \( p_{i+1} \).

... compute its region

... and prune the initial diagram.
INCREMENTSAL ALGORITHM

Starting with the Voronoi diagram of \( \{p_1, \ldots, p_i\} \)...

... add point \( p_{i+1} \)

Explore all candidates to find the site \( p_j \) (\( 1 \leq j \leq i \)) closest to \( p_{i+1} \).

... compute its region

Build its boundary starting from bisector \( b_{i+1,j} \).

... and prune the initial diagram.
INCREMENTAL ALGORITHM

Starting with the Voronoi diagram of \( \{p_1, \ldots, p_i\} \)...

... add point \( p_{i+1} \)

   Explore all candidates to find the site \( p_j \) (\( 1 \leq j \leq i \)) closest to \( p_{i+1} \).

... compute its region

   Build its boundary starting from bisector \( b_{i+1,j} \).

... and prune the initial diagram.
INCREMENTAL ALGORITHM

Starting with the Voronoi diagram of \( \{p_1, \ldots, p_i\} \)...

... add point \( p_{i+1} \)

   Explore all candidates to find the site \( p_j \) \((1 \leq j \leq i)\) closest to \( p_{i+1} \).

... compute its region

   Build its boundary starting from bisector \( b_{i+1,j} \).

... and prune the initial diagram.
INCREMENTAL ALGORITHM

Starting with the Voronoi diagram of \( \{p_1, \ldots, p_i\} \)...

... add point \( p_{i+1} \)

Explore all candidates to find the site \( p_j \) (1 ≤ j ≤ i) closest to \( p_{i+1} \).

... compute its region

Build its boundary starting from bisector \( b_{i+1,j} \).

... and prune the initial diagram.
INCREMENTAL ALGORITHM

Starting with the Voronoi diagram of \( \{p_1, \ldots, p_i\} \)...

... add point \( p_{i+1} \)

   Explore all candidates to find the site \( p_j \) \((1 \leq j \leq i)\) closest to \( p_{i+1} \).

... compute its region

   Build its boundary starting from bisector \( b_{i+1,j} \).

... and prune the initial diagram.
**INCREMENTAL ALGORITHM**

Starting with the Voronoi diagram of \( \{p_1, \ldots, p_i\} \)...

... add point \( p_{i+1} \)

   Explore all candidates to find the site \( p_j \) (1 ≤ j ≤ i) closest to \( p_{i+1} \).

... compute its region

   Build its boundary starting from bisector \( b_{i+1,j} \).

... and prune the initial diagram.
INCREMENTAL ALGORITHM

Starting with the Voronoi diagram of \( \{p_1, \ldots, p_i\} \)...

... add point \( p_{i+1} \)

   Explore all candidates to find the site \( p_j \) (\( 1 \leq j \leq i \)) closest to \( p_{i+1} \).

... compute its region

   Build its boundary starting from bisector \( b_{i+1,j} \).

... and prune the initial diagram.

   While building the Voronoi region of \( p_{i+1} \), update the DCEL.
INCREMENTAL ALGORITHM

Starting with the Voronoi diagram of \( \{p_1, \ldots, p_i\} \)...

... add point \( p_{i+1} \)

Explore all candidates to find the site \( p_j \) \((1 \leq j \leq i)\) closest to \( p_{i+1} \).

... compute its region

Build its boundary starting from bisector \( b_{i+1,j} \).

... and prune the initial diagram.

While building the Voronoi region of \( p_{i+1} \), update the DCEL.
Constructing Voronoi diagrams

How to update the DCEL
How to update the DCEL

Each time an edge $e$, generated by $p_{i+1}$ and $p_j$, intersects a preexistent edge, $e'$, a new vertex $v$ is created and a new edge starts, $e+1$. Then, these are the tasks to perform:

- Create $v$ with $e(v) = e$
- Assign $v_E(e) = v$, $e_N(e) = e'$, $f_L(e) = i + 1$, $f_R(e) = j$
- Create $e + 1$ and assign $v_B(e + 1) = v$, $e_P(e + 1) = e$
- Delete all edges of the region of $p_j$, that lie between $v_B(e)$ and $v_E(e)$ in clockwise order
- Update $v_*(e') = v$ and $e_*(e') = e+1$
- Update $e(p_j) = e$
How to update the DCEL

Each time an edge $e$, generated by $p_{i+1}$ and $p_j$, intersects a preexistent edge, $e'$, a new vertex $v$ is created and a new edge starts, $e + 1$. Then, these are the tasks to perform:

- Create $v$ with $e(v) = e$
- Assign $v_E(e) = v$, $e_N(e) = e'$, $f_L(e) = i + 1$, $f_R(e) = j$
- Create $e + 1$ and assign $v_B(e + 1) = v$, $e_P(e + 1) = e$
- Delete all edges of the region of $p_j$, that lie between $v_B(e)$ and $v_E(e)$ in clockwise order
- Update $v_*(e') = v$ and $e_*(e') = e + 1$
- Update $e(p_j) = e$
How to update the DCEL

Each time an edge \( e \), generated by \( p_{i+1} \) and \( p_j \), intersects a preexistent edge, \( e' \), a new vertex \( v \) is created and a new edge starts, \( e + 1 \). Then, these are the tasks to perform:

- Create \( v \) with \( e(v) = e \)
- Assign \( v_E(e) = v \), \( e_N(e) = e' \), \( f_L(e) = i + 1 \), \( f_R(e) = j \)
- Create \( e + 1 \) and assign \( v_B(e+1) = v \), \( e_P(e+1) = e \)
- Delete all edges of the region of \( p_j \), that lie between \( v_B(e) \) and \( v_E(e) \) in clockwise order
- Update \( v_*(e') = v \) and \( e_*(e') = e+1 \)
- Update \( e(p_j) = e \)
INCREMENTAL ALGORITHM

Starting with the Voronoi diagram of \{p_1, \ldots, p_i\}...

... add point \(p_{i+1}\)

   Explore all candidates to find the site \(p_j\) \((1 \leq j \leq i)\) closest to \(p_{i+1}\).

... compute its region

   Build its boundary starting from bisector \(b_{i+1,j}\).

... and prune the initial diagram.

   While building the Voronoi region of \(p_{i+1}\), update the DCEL.
INCREMENTAL ALGORITHM

Starting with the Voronoi diagram of \{p_1, \ldots, p_i\}...

... add point \(p_{i+1}\)
  
  Explore all candidates to find the site \(p_j\) \((1 \leq j \leq i)\) closest to \(p_{i+1}\).

... compute its region
  
  Build its boundary starting from bisector \(b_{i+1,j}\).

... and prune the initial diagram.
  
  While building the Voronoi region of \(p_{i+1}\), update the DCEL.

**Running time:** Each step runs in \(O(i)\) time, therefore the total running time of the algorithm is \(O(n^2)\).
divide and conquer algorithm
DIVIDE AND CONQUER ALGORITHM
Let $P$ be a set of $n$ points in the plane.
DIVIDE AND CONQUER ALGORITHM

Let $P$ be a set of $n$ points in the plane.

If the points are vertically partitioned into two subsets $R$ and $B$...
Let $P$ be a set of $n$ points in the plane.

If the points are vertically partitioned into two subsets $R$ and $B$...

...consider the Voronoi diagram of the sets $R$ and $B$...
Let $P$ be a set of $n$ points in the plane.

If the points are vertically partitioned into two subsets $R$ and $B$...

...consider the Voronoi diagram of the sets $R$ and $B$...
Let $P$ be a set of $n$ points in the plane.

If the points are vertically partitioned into two subsets $R$ and $B$...

...consider the Voronoi diagram of the sets $R$ and $B$...
Constructing Voronoi diagrams

DIVIDE AND CONQUER ALGORITHM

Let $P$ be a set of $n$ points in the plane.

If the points are vertically partitioned into two subsets $R$ and $B$...

...consider the Voronoi diagram of the sets $R$ and $B$...
Constructing Voronoi diagrams

DIVIDE AND CONQUER ALGORITHM

Let $P$ be a set of $n$ points in the plane.

If the points are vertically partitioned into two subsets $R$ and $B$...

...consider the Voronoi diagram of the sets $R$ and $B$...
DIVIDE AND CONQUER ALGORITHM

Let $P$ be a set of $n$ points in the plane.

If the points are vertically partitioned into two subsets $R$ and $B$...

...consider the Voronoi diagram of the sets $R$ and $B$...

...then the Voronoi diagram of $P$ substantially coincides with the Voronoi diagrams of $R$ and $B$!
Let \( P \) be a set of \( n \) points in the plane.

If the points are vertically partitioned into two subsets \( R \) and \( B \)...

...consider the Voronoi diagram of the sets \( R \) and \( B \)...

...then the Voronoi diagram of \( P \) substantially coincides with the Voronoi diagrams of \( R \) and \( B \)!

In fact, there exists a monotone chain of edges of \( \text{Vor}(P) \) such that \( \text{Vor}(P) \) coincides with \( \text{Vor}(R) \) to the left of the chain, and it coincides with \( \text{Vor}(B) \) to its right.
Let $P$ be a set of $n$ points in the plane.

If the points are vertically partitioned into two subsets $R$ and $B$...

...consider the Voronoi diagram of the sets $R$ and $B$...

...then the Voronoi diagram of $P$ substantially coincides with the Voronoi diagrams of $R$ and $B$!

In fact, there exists a monotone chain of edges of $\text{Vor}(P)$ such that $\text{Vor}(P)$ coincides with $\text{Vor}(R)$ to the left of the chain, and it coincides with $\text{Vor}(B)$ to its right.
Let $P$ be a set of $n$ points in the plane.

If the points are vertically partitioned into two subsets $R$ and $B$...

...consider the Voronoi diagram of the sets $R$ and $B$...

...then the Voronoi diagram of $P$ substantially coincides with the Voronoi diagrams of $R$ and $B$!

In fact, there exists a monotone chain of edges of $Vor(P)$ such that $Vor(P)$ coincides with $Vor(R)$ to the left of the chain, and it coincides with $Vor(B)$ to its right.
Let $P$ be a set of $n$ points in the plane.

If the points are vertically partitioned into two subsets $R$ and $B$...

...consider the Voronoi diagram of the sets $R$ and $B$...

...then the Voronoi diagram of $P$ substantially coincides with the Voronoi diagrams of $R$ and $B$!

In fact, there exists a monotone chain of edges of $Vor(P)$ such that $Vor(P)$ coincides with $Vor(R)$ to the left of the chain, and it coincides with $Vor(B)$ to its right.
Let $P$ be a set of $n$ points in the plane.

If the points are vertically partitioned into two subsets $R$ and $B$...

...consider the Voronoi diagram of the sets $R$ and $B$...

...then the Voronoi diagram of $P$ substantially coincides with the Voronoi diagrams of $R$ and $B$!

In fact, there exists a monotone chain of edges of $Vor(P)$ such that $Vor(P)$ coincides with $Vor(R)$ to the left of the chain, and it coincides with $Vor(B)$ to its right.
Let $P$ be a set of $n$ points in the plane.

If the points are vertically partitioned into two subsets $R$ and $B$...

...consider the Voronoi diagram of the sets $R$ and $B$...

...then the Voronoi diagram of $P$ substantially coincides with the Voronoi diagrams of $R$ and $B$!

In fact, there exists a monotone chain of edges of $Vor(P)$ such that $Vor(P)$ coincides with $Vor(R)$ to the left of the chain, and it coincides with $Vor(B)$ to its right.
Let $P$ be a set of $n$ points in the plane.

If the points are vertically partitioned into two subsets $R$ and $B$...

...consider the Voronoi diagram of the sets $R$ and $B$...

...then the Voronoi diagram of $P$ substantially coincides with the Voronoi diagrams of $R$ and $B$!

In fact, there exists a monotone chain of edges of $Vor(P)$ such that $Vor(P)$ coincides with $Vor(R)$ to the left of the chain, and it coincides with $Vor(B)$ to its right.
**Definition.** Let $b(R, B)$ be the set of all edges and vertices of $\text{Vor}(P)$ belonging to the common boundary of the regions of some $p_i \in R$ and $p_j \in B$. 

**Constructing Voronoi diagrams**
**Definition.** Let $b(R, B)$ be the set of all edges and vertices of $Vor(P)$ belonging to the common boundary of the regions of some $p_i \in R$ and $p_j \in B$. 

**Constructing Voronoi diagrams**
Definition. Let $b(R, B)$ be the set of all edges and vertices of $Vor(P)$ belonging to the common boundary of the regions of some $p_i \in R$ and $p_j \in B$.

Observation 1. The bisector $b(R, B)$ contains two half-lines, belonging to the bisectors $b_{ij}$ of the two “bridges” connecting the convex hulls of $R$ and $B$. 
Definition. Let \( b(R, B) \) be the set of all edges and vertices of \( \text{Vor}(P) \) belonging to the common boundary of the regions of some \( p_i \in R \) and \( p_j \in B \).

Observation 1. The bisector \( b(R, B) \) contains two half-lines, belonging to the bisectors \( b_{ij} \) of the two “bridges” connecting the convex hulls of \( R \) and \( B \).

Proof. The vertical separation of \( R \) and \( B \) guarantees the existence of the “bridges”, which are the edges of \( \text{ch}(P) \) connecting a \( p_i \in R \) to a \( p_j \in B \).
Definition. Let $b(R, B)$ be the set of all edges and vertices of $Vor(P)$ belonging to the common boundary of the regions of some $p_i \in R$ and $p_j \in B$.

Observation 1. The bisector $b(R, B)$ contains two half-lines, belonging to the bisectors $b_{ij}$ of the two “bridges” connecting the convex hulls of $R$ and $B$.

Observation 2. The bisector $b(R, B)$ is a $y$-monotone chain leaving the regions of the points $p_i \in R$ to its left and those of $p_j \in B$ to its right.
**Definition.** Let $b(R, B)$ be the set of all edges and vertices of $Vor(P)$ belonging to the common boundary of the regions of some $p_i \in R$ and $p_j \in B$.

**Observation 1.** The bisector $b(R, B)$ contains two half-lines, belonging to the bisectors $b_{ij}$ of the two “bridges” connecting the convex hulls of $R$ and $B$.

**Observation 2.** The bisector $b(R, B)$ is a $y$-monotone chain leaving the regions of the points $p_i \in R$ to its left and those of $p_j \in B$ to its right.

**Proof.** Every edge $e_{ij}$ of $b(R, B)$ must be non-horizontal, and leave $p_i \in R$ to its left and $p_j \in B$ to its right.
Definition. Let $b(R, B)$ be the set of all edges and vertices of $Vor(P)$ belonging to the common boundary of the regions of some $p_i \in R$ and $p_j \in B$.

Observation 1. The bisector $b(R, B)$ contains two half-lines, belonging to the bisectors $b_{ij}$ of the two “bridges” connecting the convex hulls of $R$ and $B$.

Observation 2. The bisector $b(R, B)$ is a $y$-monotone chain leaving the regions of the points $p_i \in R$ to its left and those of $p_j \in B$ to its right.

Observation 3. Let $\pi_R$ and $\pi_B$ respectively be the regions of the plane located to the left and to the right of $b(R, B)$. Then $Vor(P)$ consists of $Vor(R) \cap \pi_R$, $Vor(B) \cap \pi_B$ and $b(R, B)$.
**Definition.** Let \( b(R, B) \) be the set of all edges and vertices of \( \text{Vor}(P) \) belonging to the common boundary of the regions of some \( p_i \in R \) and \( p_j \in B \).

**Observation 1.** The bisector \( b(R, B) \) contains two half-lines, belonging to the bisectors \( b_{ij} \) of the two “bridges” connecting the convex hulls of \( R \) and \( B \).

**Observation 2.** The bisector \( b(R, B) \) is a \( y \)-monotone chain leaving the regions of the points \( p_i \in R \) to its left and those of \( p_j \in B \) to its right.

**Observation 3.** Let \( \pi_R \) and \( \pi_B \) respectively be the regions of the plane located to the left and to the right of \( b(R, B) \). Then \( \text{Vor}(P) \) consists of \( \text{Vor}(R) \cap \pi_R \), \( \text{Vor}(B) \cap \pi_B \) and \( b(R, B) \).

**Proof.** Let \( e \) be an edge of \( \text{Vor}(P) \):
- If \( e \) separates two points of \( R \) in \( \text{Vor}(P) \), then it is (a portion of) the edge separating them in \( \text{Vor}(R) \).
  Due to Obs. 2, \( e \) cannot belong to \( \pi_B \).
- If \( e \) separates two points of \( B \), the case is analogous.
- If \( e \) separates one point of \( R \) from one of \( B \), then \( e \in b(R, B) \).
1. Sort the points of $P$ by abscissa (only once) and vertically partition $P$ into two subsets $R$ and $B$, of approximately the same size.
DIVIDE AND CONQUER ALGORITHM

1. Sort the points of \( P \) by abscissa (only once) and vertically partition \( P \) into two subsets \( R \) and \( B \), of approximately the same size.

2. Recursively compute \( \text{Vor}(R) \) and \( \text{Vor}(B) \).
Constructing Voronoi diagrams

DIVIDE AND CONQUER ALGORITHM

1. Sort the points of $P$ by abscissa (only once) and vertically partition $P$ into two subsets $R$ and $B$, of approximately the same size.

2. Recursively compute $Vor(R)$ and $Vor(B)$. 
1. Sort the points of $P$ by abscissa (only once) and vertically partition $P$ into two subsets $R$ and $B$, of approximately the same size.

2. Recursively compute $Vor(R)$ and $Vor(B)$.

3. Compute the separating chain.
DIVIDE AND CONQUER ALGORITHM

1. Sort the points of $P$ by abscissa (only once) and vertically partition $P$ into two subsets $R$ and $B$, of approximately the same size.

2. Recursively compute $Vor(R)$ and $Vor(B)$.

3. Compute the separating chain.

4. Prune the portion of $Vor(R)$ lying to the right of the chain and the portion of $Vor(B)$ lying to its left.
DIVIDE AND CONQUER ALGORITHM

1. Sort the points of $P$ by abscissa (only once) and vertically partition $P$ into two subsets $R$ and $B$, of approximately the same size.

2. Recursively compute $Vor(R)$ and $Vor(B)$.

3. Compute the separating chain.

4. Prune the portion of $Vor(R)$ lying to the right of the chain and the portion of $Vor(B)$ lying to its left.
1. Sort the points of $P$ by abscissa (only once) and vertically partition $P$ into two subsets $R$ and $B$, of approximately the same size.

2. Recursively compute $Vor(R)$ and $Vor(B)$.

3. Compute the separating chain.

4. Prune the portion of $Vor(R)$ lying to the right of the chain and the portion of $Vor(B)$ lying to its left.
Constructing Voronoi diagrams

How to compute the chain?
How to compute the chain?

Initialization
Find the two halflines
Constructing Voronoi diagrams

How to compute the chain?

Initialization
Find the two halflines
How to compute the chain?

Initialization
Find the two halflines
How to compute the chain?

Initialization

Find the two halflines
Constructing Voronoi diagrams

How to compute the chain?

Initialization
Find the two halflines
How to compute the chain?

Initialization
Find the two halflines
How to compute the chain?

Initialization
Find the two halflines
How to compute the chain?

Initialization
Find the two halflines
How to compute the chain?

Initialization
Find the two halflines
How to compute the chain?

Initialization
Find the two halflines

Advance
Starting with one of the halflines, and until getting to the other one, do:

Each time an edge $e \in b(R, B)$ begins, such that $e \subset b_{ij}$, $p_i \in R$ and $p_j \in B$, do:
- Detect its intersection with $Vor_R(p_i)$
- Detect its intersection with $Vor_B(p_j)$
- Choose the first of the two intersection points
- Detect the site $p_k$ corresponding to the new starting region
- Replace $p_i$ or $p_j$ (as required) by $p_k$
- Restart with the new edge
Constructing Voronoi diagrams

How to compute the chain?

Initialization
Find the two halflines

Advance
Starting with one of the halflines, and until getting to the other one, do:

Each time an edge $e \in b(R, B)$ begins, such that $e \subset b_{ij}$, $p_i \in R$ and $p_j \in B$, do:

- Detect its intersection with $Vor_R(p_i)$
- Detect its intersection with $Vor_B(p_j)$
- Choose the first of the two intersection points
- Detect the site $p_k$ corresponding to the new starting region
- Replace $p_i$ or $p_j$ (as required) by $p_k$
- Restart with the new edge
How to compute the chain?

Initialization
Find the two halflines

Advance
Starting with one of the halflines, and until getting to the other one, do:

Each time an edge \( e \in b(R, B) \) begins, such that \( e \subset b_{ij}, p_i \in R \) and \( p_j \in B \), do:

- Detect its intersection with \( \text{Vor}_R(p_i) \)
- Detect its intersection with \( \text{Vor}_B(p_j) \)
- Choose the first of the two intersection points
- Detect the site \( p_k \) corresponding to the new starting region
- Replace \( p_i \) or \( p_j \) (as required) by \( p_k \)
- Restart with the new edge
Constructing Voronoi diagrams

How to compute the chain?

Initialization
Find the two halflines

Advance
Starting with one of the halflines, and until getting to the other one, do:

Each time an edge $e \in b(R, B)$ begins, such that $e \subset b_{ij}$, $p_i \in R$ and $p_j \in B$, do:
- Detect its intersection with $Vor_R(p_i)$
- Detect its intersection with $Vor_B(p_j)$
- Choose the first of the two intersection points
- Detect the site $p_k$ corresponding to the new starting region
- Replace $p_i$ or $p_j$ (as required) by $p_k$
- Restart with the new edge
How to compute the chain?

Initialization
Find the two halflines

Advance
Starting with one of the halflines, and until getting to the other one, do:
Each time an edge $e \in b(R, B)$ begins, such that $e \subset b_{ij}$, $p_i \in R$ and $p_j \in B$, do:
- Detect its intersection with $Vor_R(p_i)$
- Detect its intersection with $Vor_B(p_j)$
- Choose the first of the two intersection points
- Detect the site $p_k$ corresponding to the new starting region
- Replace $p_i$ or $p_j$ (as required) by $p_k$
- Restart with the new edge
How to compute the chain?

**Initialization**
Find the two halflines

**Advance**
Starting with one of the halflines, and until getting to the other one, do:

Each time an edge \( e \in b(R, B) \) begins, such that \( e \subset b_{ij}, p_i \in R \) and \( p_j \in B \), do:

- Detect its intersection with \( Vor_R(p_i) \)
- Detect its intersection with \( Vor_B(p_j) \)
- Choose the first of the two intersection points
- Detect the site \( p_k \) corresponding to the new starting region
- Replace \( p_i \) or \( p_j \) (as required) by \( p_k \)
- Restart with the new edge

Constructing Voronoi diagrams
Constructing Voronoi diagrams

How to compute the chain?

Initialization
Find the two halflines

Advance
Starting with one of the halflines, and until getting to the other one, do:

Each time an edge $e \in b(R, B)$ begins, such that $e \subset b_{ij}$, $p_i \in R$ and $p_j \in B$, do:
- Detect its intersection with $Vor_R(p_i)$
- Detect its intersection with $Vor_B(p_j)$
- Choose the first of the two intersection points
- Detect the site $p_k$ corresponding to the new starting region
- Replace $p_i$ or $p_j$ (as required) by $p_k$
- Restart with the new edge
How to compute the chain?

Initialization
Find the two halflines

Advance
Starting with one of the halflines, and until getting to the other one, do:

Each time an edge \( e \in b(R, B) \) begins, such that \( e \subset b_{ij}, p_i \in R \) and \( p_j \in B \), do:
- Detect its intersection with \( \text{Vor}_R(p_i) \)
- Detect its intersection with \( \text{Vor}_B(p_j) \)
- Choose the first of the two intersection points
- Detect the site \( p_k \) corresponding to the new starting region
- Replace \( p_i \) or \( p_j \) (as required) by \( p_k \)
- Restart with the new edge
How to compute the chain?

Initialization
Find the two halflines

Advance
Starting with one of the halflines, and until getting to the other one, do:

Each time an edge \( e \in b(R, B) \) begins, such that \( e \subset b_{ij}, p_i \in R \) and \( p_j \in B \), do:

- Detect its intersection with \( Vor_R(p_i) \)
- Detect its intersection with \( Vor_B(p_j) \)
- Choose the first of the two intersection points
- Detect the site \( p_k \) corresponding to the new starting region
- Replace \( p_i \) or \( p_j \) (as required) by \( p_k \)
- Restart with the new edge
**Constructing Voronoi diagrams**

**How to compute the chain?**

**Initialization**

Find the two halflines

**Advance**

Starting with one of the halflines, and until getting to the other one, do:

Each time an edge $e \in b(R, B)$ begins, such that $e \subset b_{ij}$, $p_i \in R$ and $p_j \in B$, do:

- Detect its intersection with $Vor_R(p_i)$
- Detect its intersection with $Vor_B(p_j)$
- Choose the first of the two intersection points
- Detect the site $p_k$ corresponding to the new starting region
- Replace $p_i$ or $p_j$ (as required) by $p_k$
- Restart with the new edge
Constructing Voronoi diagrams

How to compute the chain?

Initialization
Find the two halflines

Advance
Starting with one of the halflines, and until getting to the other one, do:

Each time an edge $e \in b(R, B)$ begins, such that $e \subset b_{ij}$, $p_i \in R$ and $p_j \in B$, do:

- Detect its intersection with $Vor_R(p_i)$
- Detect its intersection with $Vor_B(p_j)$
- Choose the first of the two intersection points
- Detect the site $p_k$ corresponding to the new starting region
- Replace $p_i$ or $p_j$ (as required) by $p_k$
- Restart with the new edge
How to compute the chain?

Initialization
Find the two halflines

Advance
Starting with one of the halflines, and until getting to the other one, do:
Each time an edge $e \in b(R, B)$ begins, such that $e \subset b_{ij}$, $p_i \in R$ and $p_j \in B$, do:
- Detect its intersection with $Vor_R(p_i)$
- Detect its intersection with $Vor_B(p_j)$
- Choose the first of the two intersection points
- Detect the site $p_k$ corresponding to the new starting region
- Replace $p_i$ or $p_j$ (as required) by $p_k$
- Restart with the new edge
How to compute the chain?

Initialization
Find the two halflines

Advance
Starting with one of the halflines, and until getting to the other one, do:

Each time an edge $e \in b(R, B)$ begins, such that $e \subset b_{ij}$, $p_i \in R$ and $p_j \in B$, do:

- Detect its intersection with $Vor_R(p_i)$
- Detect its intersection with $Vor_B(p_j)$
- Choose the first of the two intersection points
- Detect the site $p_k$ corresponding to the new starting region
- Replace $p_i$ or $p_j$ (as required) by $p_k$
- Restart with the new edge
Constructing Voronoi diagrams

How to compute the chain?

Initialization
Find the two halflines

Advance
Starting with one of the halflines, and until getting to the other one, do:

Each time an edge \( e \in b(R, B) \) begins, such that \( e \subset b_{ij}, \ p_i \in R \) and \( p_j \in B \), do:

- Detect its intersection with \( \text{Vor}_R(p_i) \)
- Detect its intersection with \( \text{Vor}_B(p_j) \)
- Choose the first of the two intersection points
- Detect the site \( p_k \) corresponding to the new starting region
- Replace \( p_i \) or \( p_j \) (as required) by \( p_k \)
- Restart with the new edge
How to compute the chain?

Initialization
Find the two halflines

Advance
Starting with one of the halflines, and until getting to the other one, do:

Each time an edge \( e \in b(R, B) \) begins, such that \( e \subset b_{ij}, p_i \in R \) and \( p_j \in B \), do:
- Detect its intersection with \( \text{Vor}_R(p_i) \)
- Detect its intersection with \( \text{Vor}_B(p_j) \)
- Choose the first of the two intersection points
- Detect the site \( p_k \) corresponding to the new starting region
- Replace \( p_i \) or \( p_j \) (as required) by \( p_k \)
- Restart with the new edge
Constructing Voronoi diagrams

How to compute the chain?

Initialization
Find the two halflines

Advance
Starting with one of the halflines, and until getting to the other one, do:

Each time an edge $e \in b(R, B)$ begins, such that $e \subset b_{ij}$, $p_i \in R$ and $p_j \in B$, do:

- Detect its intersection with $Vor_R(p_i)$
- Detect its intersection with $Vor_B(p_j)$
- Choose the first of the two intersection points
- Detect the site $p_k$ corresponding to the new starting region
- Replace $p_i$ or $p_j$ (as required) by $p_k$
- Restart with the new edge
How to compute the chain?

**Initialization**
Find the two halflines

**Advance**
Starting with one of the halflines, and until getting to the other one, do:

Each time an edge \( e \in b(R, B) \) begins, such that \( e \subset b_{ij}, p_i \in R \) and \( p_j \in B \), do:

- Detect its intersection with \( \text{Vor}_R(p_i) \)
- Detect its intersection with \( \text{Vor}_B(p_j) \)
- Choose the first of the two intersection points
- Detect the site \( p_k \) corresponding to the new starting region
- Replace \( p_i \) or \( p_j \) (as required) by \( p_k \)
- Restart with the new edge

Constructing Voronoi diagrams
Computational Geometry, Facultat d'Informàtica de Barcelona, UPC

Constructing Voronoi diagrams

How to compute the chain?

Initialization
Find the two halflines

Advance
Starting with one of the halflines, and until getting to the other one, do:

Each time an edge $e \in b(R, B)$ begins, such that $e \subset b_{ij}$, $p_i \in R$ and $p_j \in B$, do:

- Detect its intersection with $Vor_R(p_i)$
- Detect its intersection with $Vor_B(p_j)$
- Choose the first of the two intersection points
- Detect the site $p_k$ corresponding to the new starting region
- Replace $p_i$ or $p_j$ (as required) by $p_k$
- Restart with the new edge
How to compute the chain?

Initialization
Find the two halflines

Advance
Starting with one of the halflines, and until getting to the other one, do:

Each time an edge \( e \in b(R, B) \) begins, such that \( e \subset b_{ij}, p_i \in R \) and \( p_j \in B \), do:
- Detect its intersection with \( \text{Vor}_R(p_i) \)
- Detect its intersection with \( \text{Vor}_B(p_j) \)
- Choose the first of the two intersection points
- Detect the site \( p_k \) corresponding to the new starting region
- Replace \( p_i \) or \( p_j \) (as required) by \( p_k \)
- Restart with the new edge
How to compute the chain?

Initialization
Find the two halflines

Advance
Starting with one of the halflines, and until getting to the other one, do:

Each time an edge $e \in b(R, B)$ begins, such that $e \subset b_{ij}$, $p_i \in R$ and $p_j \in B$, do:

- Detect its intersection with $Vor_R(p_i)$
- Detect its intersection with $Vor_B(p_j)$
- Choose the first of the two intersection points
- Detect the site $p_k$ corresponding to the new starting region
- Replace $p_i$ or $p_j$ (as required) by $p_k$
- Restart with the new edge
How to compute the chain?

Initialization
Find the two halflines

Advance
Starting with one of the halflines, and until getting to the other one, do:

Each time an edge \( e \in b(R, B) \) begins, such that \( e \subset b_{ij}, p_i \in R \) and \( p_j \in B \), do:

- Detect its intersection with \( Vor_R(p_i) \)
- Detect its intersection with \( Vor_B(p_j) \)
- Choose the first of the two intersection points
- Detect the site \( p_k \) corresponding to the new starting region
- Replace \( p_i \) or \( p_j \) (as required) by \( p_k \)
- Restart with the new edge
Constructing Voronoi diagrams

How to compute the chain?

Initialization
Find the two halflines

Advance
Starting with one of the halflines, and until getting to the other one, do:

Each time an edge $e \in b(R, B)$ begins, such that $e \subset b_{ij}$, $p_i \in R$ and $p_j \in B$, do:

- Detect its intersection with $Vor_R(p_i)$
- Detect its intersection with $Vor_B(p_j)$
- Choose the first of the two intersection points
- Detect the site $p_k$ corresponding to the new starting region
- Replace $p_i$ or $p_j$ (as required) by $p_k$
- Restart with the new edge
How to compute the chain?

**Initialization**
Find the two halflines

**Advance**
Starting with one of the halflines, and until getting to the other one, do:

Each time an edge $e \in b(R, B)$ begins, such that $e \subset b_{ij}$, $p_i \in R$ and $p_j \in B$, do:

- Detect its intersection with $Vor_R(p_i)$
- Detect its intersection with $Vor_B(p_j)$
- Choose the first of the two intersection points
- Detect the site $p_k$ corresponding to the new starting region
- Replace $p_i$ or $p_j$ (as required) by $p_k$
- Restart with the new edge
How to compute the chain?

Initialization running time: $O(n)$
How to compute the chain?

**Initialization running time:** $O(n)$

From $Vor(R)$ and $Vor(B)$. 
Constructing Voronoi diagrams

How to compute the chain?

Initialization running time: $O(n)$

Advance running time: $O(n)$
How to compute the chain?

Initialization running time: $O(n)$

Advance running time: $O(n)$

If $e$ is an edge of $b(R, B)$ that entered $Vor_R(p_i)$ through some vertex $v \in Vor(P)$, then the exit point of $b(R, B)$ is found clockwise along the boundary of $Vor_R(p_i)$.
How to do the merging?
Constructing Voronoi diagrams

How to do the merging?

It consists in updating the DCEL:
How to do the merging?

It consists in updating the DCEL:
Each time a face $Vor_B(p_i)$ is left through an edge $e' \in b_{ij}$, while staying in the same face $Vor_R(p_k)$, a new vertex $v$ is created, an edge $e$ ends and another edge $e + 1$ begins:

- Create the new vertex $v$ and assign $e(v) = e$
- Create $e + 1$ and assign to it $v_B = v$ and $e_P = e'$
- Assign to $e$: $v_E = v$, $e_N = e + 1$, $f_L = i$ and $f_R = k$
- Delete all edges of $Vor_B(p_i)$ found in counterclockwise order between the entry and exit points
- Update for $e'$: $v_* = v$, $e_* = e$
- Update $e(p_i) = e$

The procedure is analogous when exiting a face $Vor_R(p_i)$. 
DIVIDE AND CONQUER ALGORITHM

1. Sort the points of $P$ by abscissa (only once) and vertically partition $P$ into two subsets $R$ and $B$, of approximately the same size.

2. Recursively compute $Vor(R)$ and $Vor(B)$.

3. Compute the separating chain.

4. Prune the portion of $Vor(R)$ lying to the right of the chain and the portion of $Vor(B)$ lying to its left.

The total running time of the algorithm is $O(n \log n)$.
1. Sort the points of $P$ by abscissa (only once) and vertically partition $P$ into two subsets $R$ and $B$, of approximately the same size.

2. Recursively compute $Vor(R)$ and $Vor(B)$.

3. Compute the separating chain.

4. Prune the portion of $Vor(R)$ lying to the right of the chain and the portion of $Vor(B)$ lying to its left.

The total running time of the algorithm is $O(n \log n)$.

This running time is optimal, because $ch(P)$ can be computed from $Vor(P)$ in $O(n)$ time.
DIVIDE AND CONQUER ALGORITHM

1. Sort the points of $P$ by abscissa (only once) and vertically partition $P$ into two subsets $R$ and $B$, of approximately the same size.

2. Recursively compute $Vor(R)$ and $Vor(B)$.

3. Compute the separating chain.

4. Prune the portion of $Vor(R)$ lying to the right of the chain and the portion of $Vor(B)$ lying to its left.

The total running time of the algorithm is $O(n \log n)$

This running time is optimal, because $ch(P)$ can be computed from $Vor(P)$ in $O(n)$ time.

OTHER ALGORITHMS

There exist other algorithms with the same running time:
- Fortune’s Algorithm (sweep)
- 3D projection algorithm
Constructing Voronoi diagrams

TWO BOOKS WITH MUCH MORE INFORMATION

A. Okabe, B. Boots, K. Sugihara, S. N. Chiu
Spatial Tessellations

F. Aurenhammer, R. Klein, D.-T. Lee
Voronoi Diagrams and Delaunay Triangulations