TRIANGULATING POINT SETS

Vera Sacristán

Computational Geometry
Facultat d’Informàtica de Barcelona
Universitat Politècnica de Catalunya
DEFINITION

A triangulation of a set $P$ of $n$ points in the plane is a graph having $P$ as set of vertices which is rectilinear, planar, and maximal in the number of edges.
DEFINITION

A triangulation of a set $P$ of $n$ points in the plane is a graph having $P$ as set of vertices which is rectilinear, planar, and maximal in the number of edges.
DEFINITION

A triangulation of a set $P$ of $n$ points in the plane is a graph having $P$ as set of vertices which is rectilinear, planar, and maximal in the number of edges.
DEFINITION

A triangulation of a set $P$ of $n$ points in the plane is a graph having $P$ as set of vertices which is rectilinear, planar, and maximal in the number of edges.

Corollary. All the faces of such a graph are triangles, except for the unbounded one, which is the exterior of the convex hull of $P$. 
A triangulation of a set $P$ of $n$ points in the plane is a graph having $P$ as set of vertices which is rectilinear, planar, and maximal in the number of edges.

**Corollary.** All the faces of such a graph are triangles, except for the unbounded one, which is the exterior of the convex hull of $P$.

**COMPLEXITY**

Every triangulation of any set $P$ of $n$ points has:

- $2n - h - 2$ triangles
- $3n - h - 3$ edges

where $h$ is the number of vertices of $ch(P)$. 
DEFINITION

A triangulation of a set $P$ of $n$ points in the plane is a graph having $P$ as set of vertices which is rectilinear, planar, and maximal in the number of edges.

**Corollary.** All the faces of such a graph are triangles, except for the unbounded one, which is the exterior of the convex hull of $P$.

COMPLEXITY

Every triangulation of any set $P$ of $n$ points has:

- $2n - h - 2$ triangles
- $3n - h - 3$ edges

where $h$ is the number of vertices of $ch(P)$.

**Proof.** Each triangle has exactly 3 edges. Each internal edge belongs to exactly 2 triangles. Each external edge belongs to exactly 1 triangle. Therefore, $3t = 2(e - h) + h = 2e - h$.

According to Euler’s formula: $n + (t + 1) = v + f = e + 2$.

Combining both equations:

- $e = n + t - 1 \Rightarrow 3e = 3n + 3t - 3 = 3n + 2e - h - 3 \Rightarrow e = 3n - h - 3$
- $3t = 2e - h = 6n - 2h - 6 - h = 6n - 3h - 6 \Rightarrow t = 2n - h - 2$
DEFINITION

A triangulation of a set $P$ of $n$ points in the plane is a graph having $P$ as set of vertices which is rectilinear, planar, and maximal in the number of edges.

**Corollary.** All the faces of such a graph are triangles, except for the unbounded one, which is the exterior of the convex hull of $P$.

COMPLEXITY

Every triangulation of any set $P$ of $n$ points has:

- $2n - h - 2$ triangles
- $3n - h - 3$ edges

where $h$ is the number of vertices of $ch(P)$.

DEGENERACIES

As you may have noticed, we are assuming that the set $P$ does not contain three or more points on a line. The assumption is hold along the entire chapter.
DATA STRUCTURE

We want to answer the most usual questions for any decomposition of the plane:
- For any given triangle, report its edges/vertices.
- For any given vertex, report the sorted list of edges/triangles incident to it.
- For any given edge, report its endpoints and its adjacent triangles.
DATA STRUCTURE

We want to answer the most usual questions for any decomposition of the plane:
- For any given triangle, report its edges/vertices.
- For any given vertex, report the sorted list of edges/triangles incident to it.
- For any given edge, report its endpoints and its adjacent triangles.

The appropriate structure is, once again, a **DCEL**:
**DATA STRUCTURE**

We want to answer the most usual questions for any decomposition of the plane:
- For any given triangle, report its edges/vertices.
- For any given vertex, report the sorted list of edges/triangles incident to it.
- For any given edge, report its endpoints and its adjacent triangles.

The appropriate structure is, once again, a **DCEL**:

**Table of vertices**

<table>
<thead>
<tr>
<th>v</th>
<th>x</th>
<th>y</th>
<th>e</th>
</tr>
</thead>
</table>
DATA STRUCTURE

We want to answer the most usual questions for any decomposition of the plane:
- For any given triangle, report its edges/vertices.
- For any given vertex, report the sorted list of edges/triangles incident to it.
- For any given edge, report its endpoints and its adjacent triangles.

The appropriate structure is, once again, a **DCEL**: 

Table of vertices 

<table>
<thead>
<tr>
<th>v</th>
<th>x</th>
<th>y</th>
<th>e</th>
</tr>
</thead>
</table>

Table of faces 

<table>
<thead>
<tr>
<th>t</th>
<th>e</th>
</tr>
</thead>
</table>
We want to answer the most usual questions for any decomposition of the plane:
- For any given triangle, report its edges/vertices.
- For any given vertex, report the sorted list of edges/triangles incident to it.
- For any given edge, report its endpoints and its adjacent triangles.

The appropriate structure is, once again, a **DCEL**:

Table of vertices

| v | x | y | e |

Table of faces

| t | e |

**DCEL**

| e | v_B | v_E | f_L | f_R | e_P | e_N |
ALGORITHMS

1. Incremental algorithms
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting
ALGORITHMS

1. Incremental algorithms
   
   1.1. Without sorting
     
     For each $i$, detect whether $p_i$ lies in the interior or the exterior of $ch(p_1, \ldots, p_{i-1})$. If it is external, compute the supporting lines from $p_i$ to $ch(p_1, \ldots, p_{i-1})$ and add all the intermediate diagonals to the triangulation. If it is internal, detect the triangle $T$ containing $p_i$ and partition $T$ into 3 triangles.
ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

For each $i$, detect whether $p_i$ lies in the interior or the exterior of $ch(p_1, \ldots, p_{i-1})$. If it is external, compute the supporting lines from $p_i$ to $ch(p_1, \ldots, p_{i-1})$ and add all the intermediate diagonals to the triangulation. If it is internal, detect the triangle $T$ containing $p_i$ and partition $T$ into 3 triangles.
**ALGORITHMS**

1. Incremental algorithms

   1.1. Without sorting

   For each $i$, detect whether $p_i$ lies in the interior or the exterior of $\text{ch}(p_1, \ldots, p_{i-1})$. If it is external, compute the supporting lines from $p_i$ to $\text{ch}(p_1, \ldots, p_{i-1})$ and add all the intermediate diagonals to the triangulation. If it is internal, detect the triangle $T$ containing $p_i$ and partition $T$ into 3 triangles.
1. Incremental algorithms

1.1. Without sorting

For each $i$, detect whether $p_i$ lies in the interior or the exterior of $ch(p_1, \ldots, p_{i-1})$. If it is external, compute the supporting lines from $p_i$ to $ch(p_1, \ldots, p_{i-1})$ and add all the intermediate diagonals to the triangulation. If it is internal, detect the triangle $T$ containing $p_i$ and partition $T$ into 3 triangles.
ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

For each $i$, detect whether $p_i$ lies in the interior or the exterior of $ch(p_1, \ldots, p_{i-1})$. If it is external, compute the supporting lines from $p_i$ to $ch(p_1, \ldots, p_{i-1})$ and add all the intermediate diagonals to the triangulation. If it is internal, detect the triangle $T$ containing $p_i$ and partition $T$ into 3 triangles.
1. Incremental algorithms

1.1. Without sorting

For each $i$, detect whether $p_i$ lies in the interior or the exterior of $ch(p_1, \ldots, p_{i-1})$. If it is external, compute the supporting lines from $p_i$ to $ch(p_1, \ldots, p_{i-1})$ and add all the intermediate diagonals to the triangulation. If it is internal, detect the triangle $T$ containing $p_i$ and partition $T$ into 3 triangles.
ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

For each $i$, detect whether $p_i$ lies in the interior or the exterior of $ch(p_1, \ldots, p_{i-1})$. If it is external, compute the supporting lines from $p_i$ to $ch(p_1, \ldots, p_{i-1})$ and add all the intermediate diagonals to the triangulation. If it is internal, detect the triangle $T$ containing $p_i$ and partition $T$ into 3 triangles.
ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

For each $i$, detect whether $p_i$ lies in the interior or the exterior of $ch(p_1, \ldots, p_{i-1})$. If it is external, compute the supporting lines from $p_i$ to $ch(p_1, \ldots, p_{i-1})$ and add all the intermediate diagonals to the triangulation. If it is internal, detect the triangle $T$ containing $p_i$ and partition $T$ into 3 triangles.
ALGORITHMS

1. Incremental algorithms

1.1. Without sorting

For each $i$, detect whether $p_i$ lies in the interior or the exterior of $ch(p_1, \ldots, p_{i-1})$. If it is external, compute the supporting lines from $p_i$ to $ch(p_1, \ldots, p_{i-1})$ and add all the intermediate diagonals to the triangulation. If it is internal, detect the triangle $T$ containing $p_i$ and partition $T$ into 3 triangles.
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting

   Running time: $O(n^2)$

   For each $i$, detect whether $p_i$ lies in the interior or the exterior of $ch(p_1, \ldots, p_{i-1})$. If it is external, compute the supporting lines from $p_i$ to $ch(p_1, \ldots, p_{i-1})$ and add all the intermediate diagonals to the triangulation. If it is internal, detect the triangle $T$ containing $p_i$ and partition $T$ into 3 triangles.
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting  \textbf{Running time: } \(O(n^2)\)
   1.2. With sorting
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting
       Running time: $O(n^2)$
   1.2. With sorting
       Start by sorting the points in lexicographical order in $O(n \log n)$ time. The information of the sorted order of the points allows to add the $i$ diagonals in $O(i)$ time, so that the amortized cost of the insertion of all diagonals is done in $O(n)$ time.
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting  \textbf{Running time: } \( O(n^2) \)
   1.2. With sorting  \textbf{Running time: } \( O(n \log n) \)

Start by sorting the points in lexicographical order in \( O(n \log n) \) time. The information of the sorted order of the points allows to add the \( i \) diagonals in \( O(i) \) time, so that the amortized cost of the insertion of all diagonals is done in \( O(n) \) time.
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting
       Running time: $O(n^2)$
   1.2. With sorting
       Running time: $O(n \log n)$
   1.3. With hierarchical structure
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting  
   Running time: $O(n^2)$
   1.2. With sorting      
   Running time: $O(n \log n)$
   1.3. With hierarchical structure

Using an auxiliary enclosing triangle and a hierarchy of triangles: each time a new point is added, a triangle gets subdivided into three children.
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting  \textbf{Running time:} \(O(n^2)\)
   1.2. With sorting \textbf{Running time:} \(O(n \log n)\)
   1.3. With hierarchical structure

Using an auxiliary enclosing triangle and a hierarchy of triangles: each time a new point is added, a triangle gets subdivided into three children.
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting  \[ \text{Running time: } O(n^2) \]
   1.2. With sorting  \[ \text{Running time: } O(n \log n) \]
   1.3. With hierarchical structure

Using an auxiliary enclosing triangle and a hierarchy of triangles: each time a new point is added, a triangle gets subdivided into three children.
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting  \hspace{1cm} \textbf{Running time: } O(n^2)
   1.2. With sorting  \hspace{1cm} \textbf{Running time: } O(n \log n)
   1.3. With hierarchical structure

Using an auxiliary enclosing triangle and a hierarchy of triangles: each time a new point is added, a triangle gets subdivided into three children.
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting
       Running time: $O(n^2)$
   1.2. With sorting
       Running time: $O(n \log n)$
   1.3. With hierarchical structure

Using an auxiliary enclosing triangle and a hierarchy of triangles: each time a new point is added, a triangle gets subdivided into three children.
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting  \textbf{Running time: } $O(n^2)$
   1.2. With sorting  \textbf{Running time: } $O(n \log n)$
   1.3. With hierarchical structure  \textbf{Running time: } $O(n^2)$ worst case, $O(n \log n)$ if balanced

Using an auxiliary enclosing triangle and a hierarchy of triangles: each time a new point is added, a triangle gets subdivided into three children.
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting
   Running time: $O(n^2)$
   1.2. With sorting
   Running time: $O(n \log n)$
   1.3. With hierarchical structure
   Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced
   1.4. Randomized
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting  \(O(n^2)\)
   1.2. With sorting  \(O(n \log n)\)
   1.3. With hierarchical structure  \(O(n^2)\) worst case, \(O(n \log n)\) if balanced
   1.4. Randomized  \(O(n \log n)\) expected
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting
       **Running time:** $O(n^2)$
   1.2. With sorting
       **Running time:** $O(n \log n)$
   1.3. With hierarchical structure
       **Running time:** $O(n^2)$ worst case, $O(n \log n)$ if balanced
   1.4. Randomized
       **Running time:** $O(n \log n)$ expected
   1.5. With auxiliary point(s)
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting  \textbf{Running time:} $O(n^2)$
   1.2. With sorting  \textbf{Running time:} $O(n \log n)$
   1.3. With hierarchical structure  \textbf{Running time:} $O(n^2)$ worst case, $O(n \log n)$ if balanced
   1.4. Randomized  \textbf{Running time:} $O(n \log n)$ expected
   1.5. With auxiliary point(s)

   A fixed point $p$ is used as a reference, and $P \cup \{p\}$ is enclosed in an auxiliary triangle.

   When inserting each point $p_i$:
   - Scan the triangles stabbed by the segment $pp_i$.
   - Update, if necessary, the information of the triangle containing $p$. 
1. Incremental algorithms

1.1. Without sorting  \[ \text{Running time: } O(n^2) \]

1.2. With sorting  \[ \text{Running time: } O(n \log n) \]

1.3. With hierarchical structure  \[ \text{Running time: } O(n^2) \text{ worst case, } O(n \log n) \text{ if balanced} \]

1.4. Randomized  \[ \text{Running time: } O(n \log n) \text{ expected} \]

1.5. With auxiliary point(s)

A fixed point \( p \) is used as a reference, and \( P \cup \{p\} \) is enclosed in an auxiliary triangle. When inserting each point \( p_i \):
- Scan the triangles stabbed by the segment \( pp_i \).
- Update, if necessary, the information of the triangle containing \( p \).
ALGORITHMS

1. Incremental algorithms
   
1.1. Without sorting  \( \text{Running time: } O(n^2) \)
1.2. With sorting  \( \text{Running time: } O(n \log n) \)
1.3. With hierarchical structure  \( \text{Running time: } O(n^2) \) worst case, \( O(n \log n) \) if balanced
1.4. Randomized  \( \text{Running time: } O(n \log n) \) expected
1.5. With auxiliary point(s)
   
A fixed point \( p \) is used as a reference, and \( P \cup \{p\} \) is enclosed in an auxiliary triangle. When inserting each point \( p_i \):
   - Scan the triangles stabbed by the segment \( \overline{pp_i} \).
   - Update, if necessary, the information of the triangle containing \( p \).
ALGORITHMS

1. Incremental algorithms

   1.1. Without sorting  \hspace{1cm} \textbf{Running time: } O(n^2)

   1.2. With sorting \hspace{1cm} \textbf{Running time: } O(n \log n)

   1.3. With hierarchical structure \hspace{1cm} \textbf{Running time: } O(n^2) \text{ worst case, } O(n \log n) \text{ if balanced}

   1.4. Randomized \hspace{1cm} \textbf{Running time: } O(n \log n) \text{ expected}

   1.5. With auxiliary point(s)

   A fixed point $p$ is used as a reference, and $P \cup \{p\}$ is enclosed in an auxiliary triangle. When inserting each point $p_i$:
   - Scan the triangles stabbed by the segment $pp_i$.
   - Update, if necessary, the information of the triangle containing $p$.  
   
   ![Diagram showing the process of inserting points into a triangulated set with auxiliary points.]

Computational Geometry, Facultat d'Informàtica de Barcelona, UPC
ALGORITHMS

1. Incremental algorithms
   
   1.1. Without sorting  
      **Running time:** $O(n^2)$
   
   1.2. With sorting  
      **Running time:** $O(n \log n)$
   
   1.3. With hierarchical structure  
      **Running time:** $O(n^2)$ worst case, $O(n \log n)$ if balanced
   
   1.4. Randomized  
      **Running time:** $O(n \log n)$ expected
   
   1.5. With auxiliary point(s)
      A fixed point $p$ is used as a reference, and $P \cup \{p\}$ is enclosed in an auxiliary triangle. When inserting each point $p_i$:
      - Scan the triangles stabbed by the segment $\overline{pp_i}$.
      - Update, if necessary, the information of the triangle containing $p$. 

![Diagram showing a point $p$ used as a reference, and a point $p_i$ being inserted into the triangulation.](image)
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting  Running time: $O(n^2)$
   1.2. With sorting      Running time: $O(n \log n)$
   1.3. With hierarchical structure  Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced
   1.4. Randomized        Running time: $O(n \log n)$ expected
   1.5. With auxiliary point(s)  Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

A fixed point $p$ is used as a reference, and $P \cup \{p\}$ is enclosed in an auxiliary triangle. When inserting each point $p_i$:
- Scan the triangles stabbed by the segment $pp_i$.
- Update, if necessary, the information of the triangle containing $p$.
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting  \( \text{Running time: } O(n^2) \)
   1.2. With sorting  \( \text{Running time: } O(n \log n) \)
   1.3. With hierarchical structure  \( \text{Running time: } O(n^2) \) worst case, \( O(n \log n) \) if balanced
   1.4. Randomized  \( \text{Running time: } O(n \log n) \) expected
   1.5. With auxiliary point(s)  \( \text{Running time: } O(n^2) \) worst case, \( O(n^{3/2}) \) expected

2. Graham's algorithm
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting
   Running time: $O(n^2)$
   1.2. With sorting
   Running time: $O(n \log n)$
   1.3. With hierarchical structure
   Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced
   1.4. Randomized
   Running time: $O(n \log n)$ expected
   1.5. With auxiliary point(s)
   Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham’s algorithm
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting
       Running time: \( O(n^2) \)
   1.2. With sorting
       Running time: \( O(n \log n) \)
   1.3. With hierarchical structure
       Running time: \( O(n^2) \) worst case, \( O(n \log n) \) if balanced
   1.4. Randomized
       Running time: \( O(n \log n) \) expected
   1.5. With auxiliary point(s)
       Running time: \( O(n^2) \) worst case, \( O(n^{3/2}) \) expected

2. Graham’s algorithm
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting
   1.2. With sorting
   1.3. With hierarchical structure
   1.4. Randomized
   1.5. With auxiliary point(s)

2. Graham’s algorithm

Running time: \( O(n^2) \)

Running time: \( O(n \log n) \)

Running time: \( O(n^2) \) worst case, \( O(n \log n) \) if balanced

Running time: \( O(n \log n) \) expected

Running time: \( O(n^2) \) worst case, \( O(n^{3/2}) \) expected
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting
   Running time: $O(n^2)$
   1.2. With sorting
   Running time: $O(n \log n)$
   1.3. With hierarchical structure
   Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced
   1.4. Randomized
   Running time: $O(n \log n)$ expected
   1.5. With auxiliary point(s)
   Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham’s algorithm
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting  \( O(n^2) \)
   1.2. With sorting   \( O(n \log n) \)
   1.3. With hierarchical structure \( O(n^2) \) worst case, \( O(n \log n) \) if balanced
   1.4. Randomized      \( O(n \log n) \) expected
   1.5. With auxiliary point(s) \( O(n^2) \) worst case, \( O(n^{3/2}) \) expected

2. Graham’s algorithm
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting
   Running time: $O(n^2)$
   1.2. With sorting
   Running time: $O(n \log n)$
   1.3. With hierarchical structure
   Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced
   1.4. Randomized
   Running time: $O(n \log n)$ expected
   1.5. With auxiliary point(s)
   Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham’s algorithm
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting
   1.2. With sorting
   1.3. With hierarchical structure
   1.4. Randomized
   1.5. With auxiliary point(s)

2. Graham’s algorithm

Running time:
- **Running time**: $O(n^2)$
- **Running time**: $O(n \log n)$
- **Running time**: $O(n^2)$ worst case, $O(n \log n)$ if balanced
- **Running time**: $O(n \log n)$ expected
- **Running time**: $O(n^2)$ worst case, $O(n^{3/2})$ expected
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting
   Running time: $O(n^2)$
   1.2. With sorting
   Running time: $O(n \log n)$
   1.3. With hierarchical structure
   Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced
   1.4. Randomized
   Running time: $O(n \log n)$ expected
   1.5. With auxiliary point(s)
   Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham’s algorithm
TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting
   Running time: $O(n^2)$
   1.2. With sorting
   Running time: $O(n \log n)$
   1.3. With hierarchical structure
   Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced
   1.4. Randomized
   Running time: $O(n \log n)$ expected
   1.5. With auxiliary point(s)
   Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham’s algorithm
TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting  
       Running time: \(O(n^2)\)
   1.2. With sorting  
       Running time: \(O(n \log n)\)
   1.3. With hierarchical structure  
       Running time: \(O(n^2)\) worst case, \(O(n \log n)\) if balanced
   1.4. Randomized  
       Running time: \(O(n \log n)\) expected
   1.5. With auxiliary point(s)  
       Running time: \(O(n^2)\) worst case, \(O(n^{3/2})\) expected

2. Graham’s algorithm
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting
       Running time: $O(n^2)$
   1.2. With sorting
       Running time: $O(n \log n)$
   1.3. With hierarchical structure
       Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced
   1.4. Randomized
       Running time: $O(n \log n)$ expected
   1.5. With auxiliary point(s)
       Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected
2. Graham’s algorithm
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting
       Running time: $O(n^2)$
   1.2. With sorting
       Running time: $O(n \log n)$
   1.3. With hierarchical structure
       Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced
   1.4. Randomized
       Running time: $O(n \log n)$ expected
   1.5. With auxiliary point(s)
       Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham’s algorithm
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting
   Running time: $O(n^2)$
   1.2. With sorting
   Running time: $O(n \log n)$
   1.3. With hierarchical structure
   Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced
   1.4. Randomized
   Running time: $O(n \log n)$ expected
   1.5. With auxiliary point(s)
   Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham’s algorithm
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting
   Running time: $O(n^2)$
   1.2. With sorting
   Running time: $O(n \log n)$
   1.3. With hierarchical structure
   Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced
   1.4. Randomized
   Running time: $O(n \log n)$ expected
   1.5. With auxiliary point(s)
   Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham’s algorithm
TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting
   Running time: $O(n^2)$
   1.2. With sorting
   Running time: $O(n \log n)$
   1.3. With hierarchical structure
   Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced
   1.4. Randomized
   Running time: $O(n \log n)$ expected
   1.5. With auxiliary point(s)
   Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected
2. Graham’s algorithm
   Running time: $O(n \log n)$
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting  \textbf{Running time: } \(O(n^2)\)
   1.2. With sorting  \textbf{Running time: } \(O(n \log n)\)
   1.3. With hierarchical structure  \textbf{Running time: } \(O(n^2)\) worst case, \(O(n \log n)\) if balanced
   1.4. Randomized  \textbf{Running time: } \(O(n \log n)\) expected
   1.5. With auxiliary point(s)  \textbf{Running time: } \(O(n^2)\) worst case, \(O(n^{3/2})\) expected

2. Graham's algorithm
   \textbf{Running time: } \(O(n \log n)\)

3. Divide and conquer
   \textbf{Running time: } \(O(n \log n)\)
ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting
       Running time: $O(n^2)$
   1.2. With sorting
       Running time: $O(n \log n)$
   1.3. With hierarchical structure
       Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced
   1.4. Randomized
       Running time: $O(n \log n)$ expected
   1.5. With auxiliary point(s)
       Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham’s algorithm
   Running time: $O(n \log n)$

3. Divide and conquer
   Initialization
   - Sort the points by abscissa
   Advance
   - Partition: divide the points into roughly two vertically separated halves
   - Recursion: recursively triangulate each half
   - Fusion: compute the external common tangents and triangulate the intermediate space
TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting
       * Running time: $O(n^2)$
   1.2. With sorting
       * Running time: $O(n \log n)$
   1.3. With hierarchical structure
       * Running time: $O(n^2)$ worst case, $O(n \log n)$ if balanced
   1.4. Randomized
       * Running time: $O(n \log n)$ expected
   1.5. With auxiliary point(s)
       * Running time: $O(n^2)$ worst case, $O(n^{3/2})$ expected

2. Graham’s algorithm
   * Running time: $O(n \log n)$

3. Divide and conquer
   **Initialization**
   - Sort the points by abscissa
   **Advance**
   - Partition: divide the points into roughly two vertically separated halves
   - Recursion: recursively triangulate each half
   - Fusion: compute the external common tangents and triangulate the intermediate space
TRIANGULATING POINT SETS

ALGORITHMS

1. Incremental algorithms
   1.1. Without sorting  \( \text{Running time: } O(n^2) \)
   1.2. With sorting  \( \text{Running time: } O(n \log n) \)
   1.3. With hierarchical structure  \( \text{Running time: } O(n^2) \text{ worst case, } O(n \log n) \text{ if balanced} \)
   1.4. Randomized  \( \text{Running time: } O(n \log n) \text{ expected} \)
   1.5. With auxiliary point(s)  \( \text{Running time: } O(n^2) \text{ worst case, } O(n^{3/2}) \text{ expected} \)

2. Graham’s algorithm  \( \text{Running time: } O(n \log n) \)

3. Divide and conquer  \( \text{Running time: } O(n \log n) \)

LOWER BOUND

This problem has an \( \Omega(n \log n) \) lower bound, since the convex hull of the set of points can be trivially obtained in \( O(n) \) time from the triangulation.
Quality of a triangulation
Quality of a triangulation
Quality of a triangulation

Incremental, without sorting
Quality of a triangulation

Incremental, without sorting
Quality of a triangulation

Incremental, without sorting
Quality of a triangulation

Incremental, without sorting
Quality of a triangulation

Incremental, without sorting
Quality of a triangulation

Incremental, without sorting
Quality of a triangulation

Incremental, without sorting
Quality of a triangulation

Incremental, without sorting
Quality of a triangulation

Incremental, without sorting
Quality of a triangulation

Incremental, without sorting
Quality of a triangulation

Incremental, without sorting
Quality of a triangulation

Incremental, without sorting
Quality of a triangulation

Incremental, without sorting
Quality of a triangulation

Incremental, sorting
Quality of a triangulation

Incremental, sorting
Quality of a triangulation

Incremental, sorting
Quality of a triangulation

Incremental, sorting
Quality of a triangulation

Incremental, sorting
Quality of a triangulation

Incremental, sorting
Quality of a triangulation

Incremental, sorting
Quality of a triangulation

Incremental, sorting
Quality of a triangulation

Incremental, sorting
Quality of a triangulation

Incremental, sorting
Quality of a triangulation

Incremental, sorting
Quality of a triangulation

Incremental, sorting
Quality of a triangulation

Incremental, sorting
Quality of a triangulation

Graham’s
Quality of a triangulation

Graham’s
Quality of a triangulation

Graham’s
Quality of a triangulation

Graham’s
Quality of a triangulation

Graham’s
Quality of a triangulation

Graham’s
Quality of a triangulation

Graham’s
Quality of a triangulation

Graham’s
Quality of a triangulation

Graham’s
Quality of a triangulation

Graham’s
Quality of a triangulation

Graham’s
Quality of a triangulation

Graham’s
Quality of a triangulation

Graham’s
Quality of a triangulation

Divide and conquer
Quality of a triangulation

Divide and conquer
Quality of a triangulation

Divide and conquer
Quality of a triangulation

Divide and conquer
Quality of a triangulation

Divide and conquer
Quality of a triangulation

Divide and conquer
Quality of a triangulation

Incremental, unsorted
Incremental, sorted
Graham’s scan
Divide and conquer
Quality of a triangulation

Incremental, unsorted
Incremental, sorted
Graham’s scan
Divide and conquer

Delaunay triangulation
Quality of a triangulation

Delaunay
Quality of a triangulation

Delaunay
Delaunay Triangulation
DELAUNAY TRIANGULATION

A TOOL FOR INTERPOLATION
DELAUNAY TRIANGULATION

A TOOL FOR INTERPOLATION
DELAUNAY TRIANGULATION

A TOOL FOR INTERPOLATION

Computational Geometry, Facultat d'Informàtica de Barcelona, UPC
DELAUNAY TRIANGULATION

A TOOL FOR INTERPOLATION
DELAUNAY TRIANGULATION

A TOOL FOR INTERPOLATION
DELAUNAY TRIANGULATION

A TOOL FOR INTERPOLATION
DELAUNAY TRIANGULATION

A TOOL FOR INTERPOLATION
DELAUNAY TRIANGULATION

A TOOL FOR INTERPOLATION
DELAUNAY TRIANGULATION

A TOOL FOR INTERPOLATION

Computational Geometry, Facultat d'Informàtica de Barcelona, UPC
DELAUNAY TRIANGULATION

DEFINITION AND PROPERTIES
**Definition**

Given a set $P$ with $n$ points in the plane...
DEFINITION AND PROPERTIES

Definition

Given a set $P$ of $n$ points in the plane, the Delaunay triangulation of $P$, $\text{Del}(P)$, is the rectilinear dual graph of the Voronoi diagram $\text{Vor}(P)$.
DEFINITION AND PROPERTIES

Definition

Given a set $P$ of $n$ points in the plane, the Delaunay triangulation of $P$, $\text{Del}(P)$, is the rectilinear dual graph of the Voronoi diagram $\text{Vor}(P)$. 
**DEFINITION AND PROPERTIES**

**Definition**

Given a set $P$ of $n$ points in the plane, the **Delaunay triangulation** of $P$, $\text{Del}(P)$, is the rectilinear dual graph of the Voronoi diagram $\text{Vor}(P)$.

**Characterization**

- Two points $p_i, p_j \in P$ form a Delaunay edge if and only if there exists a circle through $p_i$ and $p_j$ which does not contain any point of $P$ in its interior.
DEFINITION AND PROPERTIES

Definition

Given a set $P$ of $n$ points in the plane, the Delaunay triangulation of $P$, $Del(P)$, is the rectilinear dual graph of the Voronoi diagram $Vor(P)$.

Characterization

- Two points $p_i, p_j \in P$ form a Delaunay edge if and only if there exists a circle through $p_i$ and $p_j$ which does not contain any point of $P$ in its interior.

- Three points $p_i, p_j, p_k$ form a Delaunay triangle (in general, are vertices of a face) if and only if the circle through them does not contain any point of $P$ in its interior.
DEFINITION AND PROPERTIES

Definition

Given a set $P$ of $n$ points in the plane, the Delaunay triangulation of $P$, $\text{Del}(P)$, is the rectilinear dual graph of the Voronoi diagram $\text{Vor}(P)$.

Characterization

- Two points $p_i, p_j \in P$ form a Delaunay edge if and only if there exists a circle through $p_i$ and $p_j$ which does not contain any point of $P$ in its interior.

- Three points $p_i, p_j, p_k$ form a Delaunay triangle (in general, are vertices of a face) if and only if the circle through them does not contain any point of $P$ in its interior.
DEFINITION AND PROPERTIES

Definition
Given a set $P$ of $n$ points in the plane, the Delaunay triangulation of $P$, $\text{Del}(P)$, is the rectilinear dual graph of the Voronoi diagram $\text{Vor}(P)$.

Property
$\text{Del}(P)$ is a plane graph

Computational Geometry, Facultat d'Informàtica de Barcelona, UPC
DEFINITION AND PROPERTIES

Definition

Given a set \( P \) of \( n \) points in the plane, the Delaunay triangulation of \( P \), \( Del(P) \), is the rectilinear dual graph of the Voronoi diagram \( Vor(P) \).

Property

\( Del(P) \) is a plane graph

If \( pq \) is a Delaunay edge, there exists an empty circle through \( p \) and \( q \). If a segment \( rs \) intersects \( pq \), then every circle through \( r \) and \( s \) contains at least one of \( p \) or \( q \).
DEFINITION AND PROPERTIES

Definition

Given a set $P$ of $n$ points in the plane, the Delaunay triangulation of $P$, $\text{Del}(P)$, is the rectilinear dual graph of the Voronoi diagram $\text{Vor}(P)$.

Property

$\text{Del}(P)$ is a plane graph

Property

$\text{Del}(P)$ is a triangulation of $P$, except when $P$ has three or more concyclic points. In this case, it is a pre-triangulation which can be trivally completed (although this can be done in several different ways).
DEFINITION AND PROPERTIES

Definition
Given a set $P$ of $n$ points in the plane, the Delaunay triangulation of $P$, $\text{Del}(P)$, is the rectilinear dual graph of the Voronoi diagram $\text{Vor}(P)$.

Property
$\text{Del}(P)$ is a plane graph

Property
$\text{Del}(P)$ is a triangulation of $P$, except when $P$ has three or more concyclic points. In this case, it is a pre-triangulation which can be trivally completed (although this can be done in several different ways).
DEFINITION AND PROPERTIES

Definition

Given a set $P$ of $n$ points in the plane, the Delaunay triangulation of $P$, $Del(P)$, is the rectilinear dual graph of the Voronoi diagram $Vor(P)$.

Property

$Del(P)$ is a plane graph

Property

$Del(P)$ is a triangulation of $P$, except when $P$ has three or more concyclic points. In this case, it is a pre-triangulation which can be trivally completed (although this can be done in several different ways).
DEFINITION AND PROPERTIES

Definition
Given a set $P$ of $n$ points in the plane, the Delaunay triangulation of $P$, $\text{Del}(P)$, is the rectilinear dual graph of the Voronoi diagram $\text{Vor}(P)$.

Property
$\text{Del}(P)$ is a plane graph

Property
$\text{Del}(P)$ is a triangulation of $P$, except when $P$ has three or more concyclic points. In this case, it is a pre-triangulation which can be trivally completed (although this can be done in several different ways).
DEFINITION AND PROPERTIES

**Definition**

Given a set $P$ of $n$ points in the plane, the Delaunay triangulation of $P$, $\text{Del}(P)$, is the rectilinear dual graph of the Voronoi diagram $\text{Vor}(P)$.

**Property**

$\text{Del}(P)$ is a plane graph

**Property**

$\text{Del}(P)$ is a triangulation of $P$, except when $P$ has three or more concyclic points. In this case, it is a pre-triangulation which can be trivally completed (although this can be done in several different ways).
DEFINITION AND PROPERTIES

Definition
Given a set $P$ of $n$ points in the plane, the Delaunay triangulation of $P$, $\text{Del}(P)$, is the rectilinear dual graph of the Voronoi diagram $\text{Vor}(P)$.

Property
$\text{Del}(P)$ is a plane graph

Property
$\text{Del}(P)$ is a triangulation of $P$, except when $P$ has three or more concyclic points. In this case, it is a pre-triangulation which can be trivially completed (although this can be done in several different ways).
DEFINITION AND PROPERTIES

Definition
Given a set $P$ of $n$ points in the plane, the Delaunay triangulation of $P$, $\text{Del}(P)$, is the rectilinear dual graph of the Voronoi diagram $\text{Vor}(P)$.

Property
$\text{Del}(P)$ is a plane graph

Property
$\text{Del}(P)$ is a triangulation of $P$, except when $P$ has three or more concyclic points. In this case, it is a pre-triangulation which can be trivially completed (although this can be done in several different ways).
DEFINITION AND PROPERTIES

Definition

Given a set $P$ of $n$ points in the plane, the Delaunay triangulation of $P$, $\text{Del}(P)$, is the rectilinear dual graph of the Voronoi diagram $\text{Vor}(P)$.

Property

$\text{Del}(P)$ is a plane graph

Property

$\text{Del}(P)$ is a triangulation of $P$, except when $P$ has three or more concyclic points. In this case, it is a pre-triangulation which can be trivally completed (although this can be done in several different ways).
DEFINITION AND PROPERTIES

Definition
Given a set $P$ of $n$ points in the plane, the Delaunay triangulation of $P$, $\text{Del}(P)$, is the rectilinear dual graph of the Voronoi diagram $\text{Vor}(P)$.

Property
$\text{Del}(P)$ is a plane graph

Property
$\text{Del}(P)$ is a triangulation of $P$, except when $P$ has three or more concyclic points. In this case, it is a pre-triangulation which can be trivially completed (although this can be done in several different ways).
DEFINITION AND PROPERTIES

Definition
Given a set \( P \) of \( n \) points in the plane, the Delaunay triangulation of \( P \), \( \text{Del}(P) \), is the rectilinear dual graph of the Voronoi diagram \( \text{Vor}(P) \).

Property
\( \text{Del}(P) \) is a plane graph

Property
\( \text{Del}(P) \) is a triangulation of \( P \), except when \( P \) has three or more concyclic points. In this case, it is a pre-triangulation which can be trivially completed (although this can be done in several different ways).
Theorem

\[ T(P) = \text{Del}(P) \text{ iff the circumcircles of the triangles of } T(P) \text{ are empty of points of } P. \]
GLOBAL CHARACTERIZATION

Theorem

\[ T(P) = \text{Del}(P) \text{ iff the circumcircles of the triangles of } T(P) \text{ are empty of points of } P. \]

Let \( p_i \in P \). Let \( p_1, \ldots, p_k \) be the vertices of the triangles of \( T(P) \) incident to \( p_i \), sorted in counterclockwise order, \( C_1, \ldots, C_k \) be their circumcircles, and \( q_1, \ldots, q_k \) their centers (\( q_j \) denotes the center of \( C_j \), the circumcircle of \( p_i, p_j, p_{j+1} \)). We will prove that the polygon \( Q = \{q_1, \ldots, q_k\} \) coincides with \( Vor(p_i) \).
Theorem

\[ T(P) = Del(P) \iff \text{the circumcircles of the triangles of } T(P) \text{ are empty of points of } P. \]

Let \( p_i \in P \). Let \( p_1, \ldots, p_k \) be the vertices of the triangles of \( T(P) \) incident to \( p_i \), sorted in counterclockwise order, \( C_1, \ldots, C_k \) be their circumcircles, and \( q_1, \ldots, q_k \) their centers (\( q_j \) denotes the center of \( C_j \), the circumcircle of \( p_i, p_j, p_{j+1} \)). We will prove that the polygon \( Q = \{q_1, \ldots, q_k\} \) coincides with \( \text{Vor}(p_i) \).

\[ \overline{q_{j-1}q_j} \perp \overline{p_ip_j} \implies Q = \bigcap_{j=1}^{k} H_{ij} \]
Theorem

\( T(P) = Del(P) \) iff the circumcircles of the triangles of \( T(P) \) are empty of points of \( P \).

Let \( p_i \in P \). Let \( p_1, \ldots, p_k \) be the vertices of the triangles of \( T(P) \) incident to \( p_i \), sorted in counterclockwise order, \( C_1, \ldots, C_k \) be their circumcircles, and \( q_1, \ldots, q_k \) their centers (\( q_j \) denotes the center of \( C_j \), the circumcircle of \( p_i, p_j, p_{j+1} \)). We will prove that the polygon \( Q = \{q_1, \ldots, q_k\} \) coincides with \( Vor(p_i) \).

\[
q_{j-1}q_j \perp p_ip_j \implies Q = \bigcap_{j=1}^{k} H_{ij}
\]

If \( h \neq 1, \ldots, k \) then \( q_j \in b(p_i, r_h) \) and, therefore,

\[
\bigcap_{j=1}^{k} H_{ij} \subset H(p_i, r_h) \subset H_{ih}
\]
Theorem

\[ T(P) = Del(P) \iff \text{the circumcircles of the triangles of } T(P) \text{ are empty of points of } P. \]

Let \( p_i \in P \). Let \( p_1, \ldots, p_k \) be the vertices of the triangles of \( T(P) \) incident to \( p_i \), sorted in counterclockwise order, \( C_1, \ldots, C_k \) be their circumcircles, and \( q_1, \ldots, q_k \) their centers (\( q_j \) denotes the center of \( C_j \), the circumcircle of \( p_i, p_j, p_{j+1} \)). We will prove that the polygon \( Q = \{q_1, \ldots, q_k\} \) coincides with \( Vor(p_i) \).

If \( h \neq 1, \ldots, k \) then \( q_j \in b(p_i, r_h) \) and, therefore,

\[ \bigcap_{j=1}^{k} H_{ij} \subset H(p_i, r_h) \subset H_{ih} \]

Hence,

\[ Q = \bigcap_{j=1}^{k} H_{ij} = \bigcap_{j \neq i} H_{ij} = Vor(p_i) \]
DELAUNAY TRIANGULATION

DELAUNAY TRIANGULATION AND 3D CONVEX HULL
Theorem

Let \( P = \{p_1, \ldots, p_n\} \) with \( p_i = (a_i, b_i, 0) \). Let \( p_i^* = (a_i, b_i, a_i^2 + b_i^2) \) be the vertical projection of each point \( p_i \) onto the paraboloid \( z = x^2 + y^2 \). Then \( Del(P) \) is the orthogonal projection onto the plane \( z = 0 \) of the lower convex hull of \( P^* \).
Let $P = \{p_1, \ldots, p_n\}$ with $p_i = (a_i, b_i, 0)$. Let $p^*_i = (a_i, b_i, a^2_i + b^2_i)$ be the vertical projection of each point $p_i$ onto the paraboloid $z = x^2 + y^2$. Then $\text{Del}(P)$ is the orthogonal projection onto the plane $z = 0$ of the lower convex hull of $P^*$. 

Theorem
Theorem

Let $P = \{p_1, \ldots, p_n\}$ with $p_i = (a_i, b_i, 0)$. Let $p_i^* = (a_i, b_i, a_i^2 + b_i^2)$ be the vertical projection of each point $p_i$ onto the paraboloid $z = x^2 + y^2$. Then $Del(P)$ is the orthogonal projection onto the plane $z = 0$ of the lower convex hull of $P^*$. 
Let $P = \{p_1, \ldots, p_n\}$ with $p_i = (a_i, b_i, 0)$. Let $p_i^* = (a_i, b_i, a_i^2 + b_i^2)$ be the vertical projection of each point $p_i$ onto the paraboloid $z = x^2 + y^2$. Then $Del(P)$ is the orthogonal projection onto the plane $z = 0$ of the lower convex hull of $P^*$. 

Theorem

Let $P = \{p_1, \ldots, p_n\}$ with $p_i = (a_i, b_i, 0)$. Let $p_i^* = (a_i, b_i, a_i^2 + b_i^2)$ be the vertical projection of each point $p_i$ onto the paraboloid $z = x^2 + y^2$. Then $\text{Del}(P)$ is the orthogonal projection onto the plane $z = 0$ of the lower convex hull of $P^*$. 
Theorem

Let $P = \{p_1, \ldots, p_n\}$ with $p_i = (a_i, b_i, 0)$. Let $p_i^* = (a_i, b_i, a_i^2 + b_i^2)$ be the vertical projection of each point $p_i$ onto the paraboloid $z = x^2 + y^2$. Then $Del(P)$ is the orthogonal projection onto the plane $z = 0$ of the lower convex hull of $P^*$. 
Theorem

Let $P = \{p_1, \ldots, p_n\}$ with $p_i = (a_i, b_i, 0)$. Let $p_i^* = (a_i, b_i, a_i^2 + b_i^2)$ be the vertical projection of each point $p_i$ onto the paraboloid $z = x^2 + y^2$. Then $Del(P)$ is the orthogonal projection onto the plane $z = 0$ of the lower convex hull of $P^*$. 
Theorem

Let \( P = \{p_1, \ldots, p_n\} \) with \( p_i = (a_i, b_i, 0) \). Let \( p_i^* = (a_i, b_i, a_i^2 + b_i^2) \) be the vertical projection of each point \( p_i \) onto the paraboloid \( z = x^2 + y^2 \). Then \( Del(P) \) is the orthogonal projection onto the plane \( z = 0 \) of the lower convex hull of \( P^* \).
Let $P = \{p_1, \ldots, p_n\}$ with $p_i = (a_i, b_i, 0)$. Let $p_i^* = (a_i, b_i, a_i^2 + b_i^2)$ be the vertical projection of each point $p_i$ onto the paraboloid $z = x^2 + y^2$. Then Del$(P)$ is the orthogonal projection onto the plane $z = 0$ of the lower convex hull of $P^*$.

$p_i^*, p_j^*, p_k^*$ form a (triangular) face of the lower convex hull of $P^*$.
Let $P = \{p_1, \ldots, p_n\}$ with $p_i = (a_i, b_i, 0)$. Let $p^*_i = (a_i, b_i, a_i^2 + b_i^2)$ be the vertical projection of each point $p_i$ onto the paraboloid $z = x^2 + y^2$. Then $\text{Del}(P)$ is the orthogonal projection onto the plane $z = 0$ of the lower convex hull of $P^*$.

$p^*_i, p^*_j, p^*_k$ form a (triangular) face of the lower convex hull of $P^*$

The plane through $p^*_i, p^*_j, p^*_k$ leaves all the remaining points of $P^*$ above it
Theorem

Let $P = \{p_1, \ldots, p_n\}$ with $p_i = (a_i, b_i, 0)$. Let $p_i^* = (a_i, b_i, a_i^2 + b_i^2)$ be the vertical projection of each point $p_i$ onto the paraboloid $z = x^2 + y^2$. Then $\text{Del}(P)$ is the orthogonal projection onto the plane $z = 0$ of the lower convex hull of $P^*$.

$p_i^*, p_j^*, p_k^*$ form a (triangular) face of the lower convex hull of $P^*$

The plane through $p_i^*, p_j^*, p_k^*$ leaves all the remaining points of $P^*$ above it

The circle through $p_i, p_j, p_k$ leaves all the remaining points of $P$ in its exterior
Theorem

Let $P = \{p_1, \ldots, p_n\}$ with $p_i = (a_i, b_i, 0)$. Let $p_i^* = (a_i, b_i, a_i^2 + b_i^2)$ be the vertical projection of each point $p_i$ onto the paraboloid $z = x^2 + y^2$. Then $\text{Del}(P)$ is the orthogonal projection onto the plane $z = 0$ of the lower convex hull of $P^*$. $p_i^*, p_j^*, p_k^*$ form a (triangular) face of the lower convex hull of $P^*$.

The plane through $p_i^*, p_j^*, p_k^*$ leaves all the remaining points of $P^*$ above it.

The circle through $p_i, p_j, p_k$ leaves all the remaining points of $P$ in its exterior.

$p_i, p_j, p_k$ form a triangle of $\text{Del}(P)$. 

Computational Geometry, Facultat d'Informàtica de Barcelona, UPC
DELAUNAY TRIANGULATION

LOCAL CHARACTERIZATION
LOCAL CHARACTERIZATION

Definition

A triangulation $T(P)$ is locally Delaunay if each pair of triangles $p_ip_jp_k$ and $p_ip_jp_l$ sharing an edge $p_ip_j$ satisfies $p_l \notin C_{ijk}$ and $p_k \notin C_{ijl}$. 
A triangulation \( T(P) \) is locally Delaunay if each pair of triangles \( p_i p_j p_k \) and \( p_i p_j p_l \) sharing an edge \( p_i p_j \) satisfies \( p_l \notin C_{ijk} \) and \( p_k \notin C_{ijl} \).

The edge \( p_i p_j \) is locally Delaunay.

The edge \( p_i p_j \) is not locally Delaunay.

The edge \( p_i p_j \) is locally Delaunay.

In fact, the quadrilateral \( p_i p_l p_j p_k \) is not convex.
Theorem

A triangulation $T(P)$ is a Delaunay triangulation if and only if it is locally Delaunay.
Theorem

A triangulation $T(P)$ is a Delaunay triangulation if and only if it is locally Delaunay.

Suppose that $T(P)$ was locally Delaunay without being a Delaunay triangulation.
LOCAL CHARACTERIZATION

Theorem

A triangulation $T(P)$ is a Delaunay triangulation if and only if it is locally Delaunay.

Suppose that $T(P)$ was locally Delaunay without being a Delaunay triangulation.

There would exist a triangle $T_{ijk} = p_ip_jp_k$ and a point $pl$ such that $pl \in int(C_{ijk})$. 

![Diagram](image-url)
Theorem

A triangulation $T(P)$ is a Delaunay triangulation if and only if it is locally Delaunay.

Suppose that $T(P)$ was locally Delaunay without being a Delaunay triangulation. There would exist a triangle $T_{ijk} = pi pj pk$ and a point $pl$ such that $pl \in \text{int}(C_{ijk})$. Let $\overline{pi pj}$ be the edge of $T_{ijk}$ separating $pl$ from $T_{ijk}$. Among all 4-tuples in this situation, let $ijkl$ maximize the angle $pi pl pj$. 

Computational Geometry, Facultat d'Informàtica de Barcelona, UPC
Theorem

A triangulation $T(P)$ is a Delaunay triangulation if and only if it is locally Delaunay.

Suppose that $T(P)$ was locally Delaunay without being a Delaunay triangulation. There would exist a triangle $T_{ijk} = p_ip_jp_k$ and a point $p_l$ such that $p_l \in \text{int}(C_{ijk})$.

Let $\overline{p_ip_j}$ be the edge of $T_{ijk}$ separating $p_l$ from $T_{ijk}$. Among all 4-tuples in this situation, let $ijkl$ maximize the angle $p_ip_lp_j$.

Let $T_{ijm}$ be the triangle adjacent to $T_{ijk}$ though the edge $\overline{p_ip_j}$.
Theorem

A triangulation $T(P)$ is a Delaunay triangulation if and only if it is locally Delaunay.

Suppose that $T(P)$ was locally Delaunay without being a Delaunay triangulation. There would exist a triangle $T_{ijk} = p_ip_jp_k$ and a point $p_l$ such that $p_l \in \text{int}(C_{ijk})$. Let $\overline{p_ip_j}$ be the edge of $T_{ijk}$ separating $p_l$ from $T_{ijk}$. Among all 4-tuples in this situation, let $ijkl$ maximize the angle $\angle p_ip_lp_j$.

Let $T_{ijm}$ be the triangle adjacent to $T_{ijk}$ though the edge $\overline{p_ip_j}$. As $T(P)$ is locally Delaunay, $m \neq l$. 

Computational Geometry, Facultat d'Informàtica de Barcelona, UPC
Theorem

A triangulation $T(P)$ is a Delaunay triangulation if and only if it is locally Delaunay.

Suppose that $T(P)$ was locally Delaunay without being a Delaunay triangulation. There would exist a triangle $T_{ijk} = pi pj pk$ and a point $pl$ such that $pl \in \text{int}(C_{ijk})$.

Let $pi pj$ be the edge of $T_{ijk}$ separating $pl$ from $T_{ijk}$. Among all 4-tuples in this situation, let $ijkl$ maximize the angle $pi pl pj$.

Let $T_{ijm}$ be the triangle adjacent to $T_{ijk}$ though the edge $pi pj$.

As $T(P)$ is locally Delaunay, $m \neq l$. Then $pl \in C_{ijm}$.
DELUNAY TRIANGULATION

LOCAL CHARACTERIZATION

**Theorem**

A triangulation $T(P)$ is a Delaunay triangulation if and only if it is locally Delaunay.

Suppose that $T(P)$ was locally Delaunay without being a Delaunay triangulation.

There would exist a triangle $T_{ijk} = p_ip_jp_k$ and a point $p_l$ such that $p_l \in \text{int}(C_{ijk})$.

Let $\overline{p_ip_j}$ be the edge of $T_{ijk}$ separating $p_l$ from $T_{ijk}$. Among all 4-tuples in this situation, let $ijkl$ maximize the angle $p_ip_lp_j$.

Let $T_{ijm}$ be the triangle adjacent to $T_{ijk}$ though the edge $p_ip_j$.

As $T(P)$ is locally Delaunay, $m \neq l$.

Then $p_l \in C_{ijm}$.

Hence, one of the angles $p_ip_lp_m$ or $p_jp_ip_m$ would be greater than $p_ip_lp_j$.
DELAUNAY TRIANGULATION

DELAUNAY FLIPS
DELAUNAY FLIPS

We intend to prove that $\text{Del}(P)$ can be obtained from any triangulation of $P$ by Delaunay flips, which consist in deleting the diagonal of a convex quadrilateral if it is not locally Delaunay, and replacing it by the other diagonal of the quadrilateral.
DELÀUNAY FLIPS

We intend to prove that $\text{Del}(P)$ can be obtained from any triangulation of $P$ by Delaunay flips, which consist in deleting the diagonal of a convex quadrilateral if it is not locally Delaunay, and replacing it by the other diagonal of the quadrilateral.

**Lemma 1.** Let $C$ be a circle, $\overline{ab}$ a chord of $C$, and $p$, $q$, $r$ and $s$ four points lying to the same side of the line $\overline{ab}$. If $r$ is internal to $C$, $p$ and $q$ lie in $C$, and $s$ is external to $C$, then the following relations hold between the angles formed at $p$, $q$, $r$ and $s$ by the chord $\overline{ab}$: $\hat{s} < \hat{p} = \hat{q} < \hat{r}$.
We intend to prove that $\text{Del}(P)$ can be obtained from any triangulation of $P$ by Delaunay flips, which consist in deleting the diagonal of a convex quadrilateral if it is not locally Delaunay, and replacing it by the other diagonal of the quadrilateral.

**Lemma 1.** Let $C$ be a circle, $\overline{ab}$ a chord of $C$, and $p$, $q$, $r$ and $s$ four points lying to the same side of the line $\overline{ab}$. If $r$ is internal to $C$, $p$ and $q$ lie in $C$, and $s$ is external to $C$, then the following relations hold between the angles formed at $p$, $q$, $r$ and $s$ by the chord $\overline{ab}$: $\hat{s} < \hat{p} = \hat{q} < \hat{r}$.

![Diagram](https://via.placeholder.com/150)
We intend to prove that $\text{Del}(P)$ can be obtained from any triangulation of $P$ by Delaunay flips, which consist in deleting the diagonal of a convex quadrilateral if it is not locally Delaunay, and replacing it by the other diagonal of the quadrilateral.

**Lemma 1.** Let $C$ be a circle, $\overline{ab}$ a chord of $C$, and $p$, $q$, $r$ and $s$ four points lying to the same side of the line $\overline{ab}$. If $r$ is internal to $C$, $p$ and $q$ lie in $C$, and $s$ is external to $C$, then the following relations hold between the angles formed at $p$, $q$, $r$ and $s$ by the chord $\overline{ab}$: $\hat{s} < \hat{p} = \hat{q} < \hat{r}$.

Let us prove that $\hat{p} = \hat{q}$:
We intend to prove that $\text{Del}(P)$ can be obtained from any triangulation of $P$ by Delaunay flips, which consist in deleting the diagonal of a convex quadrilateral if it is not locally Delaunay, and replacing it by the other diagonal of the quadrilateral.

**Lemma 1.** Let $C$ be a circle, $ab$ a chord of $C$, and $p$, $q$, $r$ and $s$ four points lying to the same side of the line $ab$. If $r$ is internal to $C$, $p$ and $q$ lie in $C$, and $s$ is external to $C$, then the following relations hold between the angles formed at $p$, $q$, $r$ and $s$ by the chord $ab$: $\hat{s} < \hat{p} = \hat{q} < \hat{r}$.

Let us prove that $\hat{p} = \hat{q}$:

First case:

\[
\begin{align*}
2\delta + 2\gamma + 2\beta &= \pi \\
2\alpha + 2\beta &= \pi
\end{align*}
\]  
\[\Rightarrow 2\alpha = 2\gamma + 2\delta \Rightarrow \alpha = \gamma + \delta \Rightarrow \hat{p} = \hat{q} = \alpha
\]
DELAUNAY FLIPS

We intend to prove that $\text{Del}(P)$ can be obtained from any triangulation of $P$ by Delaunay flips, which consist in deleting the diagonal of a convex quadrilateral if it is not locally Delaunay, and replacing it by the other diagonal of the quadrilateral.

**Lemma 1.** Let $C$ be a circle, $\overline{ab}$ a chord of $C$, and $p$, $q$, $r$ and $s$ four points lying to the same side of the line $\overline{ab}$. If $r$ is internal to $C$, $p$ and $q$ lie in $C$, and $s$ is external to $C$, then the following relations hold between the angles formed at $p$, $q$, $r$ and $s$ by the chord $\overline{ab}$: $\hat{s} < \hat{p} = \hat{q} < \hat{r}$.

Let us prove that $\hat{p} = \hat{q}$:

**First case:**

$$2\delta + 2\gamma + 2\beta = \pi \quad \Rightarrow \quad 2\alpha + 2\beta = \pi$$

$$2\alpha + 2\beta = \pi$$

$\Rightarrow \quad 2\alpha = 2\gamma + 2\delta \Rightarrow \alpha = \gamma + \delta \Rightarrow \hat{p} = \hat{q} = \alpha$

**Second case:**

$$2\alpha + \epsilon + 2\delta = \pi \quad \Rightarrow \quad 2\alpha + 2\delta - 2\gamma = 0 \Rightarrow \alpha = \gamma - \delta \Rightarrow \hat{p} = \hat{q} = \alpha$$

![Diagram of Lemma 1](https://via.placeholder.com/150)
We intend to prove that $\text{Del}(P)$ can be obtained from any triangulation of $P$ by Delaunay flips, which consist in deleting the diagonal of a convex quadrilateral if it is not locally Delaunay, and replacing it by the other diagonal of the quadrilateral.

**Lemma 1.** Let $C$ be a circle, $\overline{ab}$ a chord of $C$, and $p$, $q$, $r$ and $s$ four points lying to the same side of the line $\overline{ab}$. If $r$ is internal to $C$, $p$ and $q$ lie in $C$, and $s$ is external to $C$, then the following relations hold between the angles formed at $p$, $q$, $r$ and $s$ by the chord $\overline{ab}$: $\widehat{s} < \widehat{p} = \widehat{q} < \widehat{r}$.

Let us prove that $\widehat{p} = \widehat{q}$:

The remaining relations follow immediately:
We intend to prove that $\text{Del}(P)$ can be obtained from any triangulation of $P$ by Delaunay flips, which consist in deleting the diagonal of a convex quadrilateral if it is not locally Delaunay, and replacing it by the other diagonal of the quadrilateral.

**Lemma 2.** When the chord $\overline{ab}$ is a diameter of $C$, the angle $\widehat{p}$ for any $p \in C$ is $\pi/2$. 
We intend to prove that $\text{Del}(P)$ can be obtained from any triangulation of $P$ by Delaunay flips, which consist in deleting the diagonal of a convex quadrilateral if it is not locally Delaunay, and replacing it by the other diagonal of the quadrilateral.

**Lemma 2.** When the chord $\overline{ab}$ is a diameter of $C$, the angle $\hat{p}$ for any $p \in C$ is $\pi/2$.

Since in this case $2\alpha = \pi$.
DELAUNAY FLIPS

We intend to prove that Del(\(P\)) can be obtained from any triangulation of \(P\) by Delaunay flips, which consist in deleting the diagonal of a convex quadrilateral if it is not locally Delaunay, and replacing it by the other diagonal of the quadrilateral.

**Lemma 3.** Given any chord \(\overline{ab}\) in a circle \(C\), if one of the arcs corresponds to \(\alpha\), then the other one corresponds to \(\pi - \alpha\).
DELAUNAY FLIPS

We intend to prove that $\text{Del}(P)$ can be obtained from any triangulation of $P$ by Delaunay flips, which consist in deleting the diagonal of a convex quadrilateral if it is not locally Delaunay, and replacing it by the other diagonal of the quadrilateral.

**Lemma 3.** Given any chord $\overline{ab}$ in a circle $C$, if one of the arcs corresponds to $\alpha$, then the other one corresponds to $\pi - \alpha$. 
We intend to prove that \( \text{Del}(P) \) can be obtained from any triangulation of \( P \) by Delaunay flips, which consist in deleting the diagonal of a convex quadrilateral if it is not locally Delaunay, and replacing it by the other diagonal of the quadrilateral.

**Lemma 3.** Given any chord \( \overline{ab} \) in a circle \( C \), if one of the arcs corresponds to \( \alpha \), then the other one corresponds to \( \pi - \alpha \).

\[
\begin{align*}
\alpha_1 + \beta + \gamma &= \frac{\pi}{2} \\
\alpha_2 + \beta + \delta &= \frac{\pi}{2}
\end{align*}
\Rightarrow \begin{cases}
\alpha + 2\beta + \gamma + \delta = \pi \\
x + \gamma + \delta = \pi
\end{cases}
\Rightarrow \begin{cases}
x = \alpha + 2\beta \\
2\alpha + 2\beta = \pi
\end{cases}
\Rightarrow x = \pi - \alpha
\]
DELAUNAY FLIPS

We intend to prove that $\text{Del}(P)$ can be obtained from any triangulation of $P$ by Delaunay flips, which consist in deleting the diagonal of a convex quadrilateral if it is not locally Delaunay, and replacing it by the other diagonal of the quadrilateral.

**Lemma 4.** Let $\overline{pq}$ be the common edge of the triangles $pqa$ and $pqb$, forming a convex quadrilateral. Then:

\[ a \in \text{ext}(C_{pqb}) \iff b \in \text{ext}(C_{pqa}) \]
DELAUNAY FLIPS

We intend to prove that $\text{Del}(P)$ can be obtained from any triangulation of $P$ by Delaunay flips, which consist in deleting the diagonal of a convex quadrilateral if it is not locally Delaunay, and replacing it by the other diagonal of the quadrilateral.

**Lemma 4.** Let $pq$ be the common edge of the triangles $pqa$ and $pqb$, forming a convex quadrilateral. Then:

$$a \in \text{ext}(C_{pqb}) \iff b \in \text{ext}(C_{pqa})$$

![Diagram of Lemma 4](image-url)
We intend to prove that $\text{Del}(P)$ can be obtained from any triangulation of $P$ by Delaunay flips, which consist in deleting the diagonal of a convex quadrilateral if it is not locally Delaunay, and replacing it by the other diagonal of the quadrilateral.

**Lemma 4.** Let $pq$ be the common edge of the triangles $pqa$ and $pqb$, forming a convex quadrilateral. Then:

\[ a \in \text{ext}(C_{pqb}) \iff b \in \text{ext}(C_{pqa}) \]

\[ a \in \text{ext}(C_{pqb}) \iff \alpha < \pi - \beta \iff \beta < \pi - \alpha \iff b \in \text{ext}(C_{pqa}) \]
We intend to prove that $\text{Del}(P)$ can be obtained from any triangulation of $P$ by Delaunay flips, which consist in deleting the diagonal of a convex quadrilateral if it is not locally Delaunay, and replacing it by the other diagonal of the quadrilateral.

**Lemma 5.** Consider a convex quadrilateral with diagonals $\overline{ab}$ and $\overline{pq}$. Then:

$$\overline{ab} \text{ is not locally Delaunay} \iff \overline{pq} \text{ is locally Delaunay}$$
We intend to prove that $\text{Del}(P)$ can be obtained from any triangulation of $P$ by Delaunay flips, which consist in deleting the diagonal of a convex quadrilateral if it is not locally Delaunay, and replacing it by the other diagonal of the quadrilateral.

**Lemma 5.** Consider a convex quadrilateral with diagonals $\overline{ab}$ and $\overline{pq}$. Then:

$\overline{ab}$ is not locally Delaunay $\iff$ $\overline{pq}$ is locally Delaunay

\[
\begin{align*}
\overline{ab} & \text{ is not locally Delaunay} \\
\iff q & \in \text{int}(C_{abp}) \\
\iff \hat{aqp} & > \hat{abp} \\
\iff b & \in \text{ext}(C_{apq}) \\
\iff \overline{pq} & \text{ is locally Delaunay}
\end{align*}
\]
DELAUNAY TRIANGULATION

DELAUNAY FLIPS

We intend to prove that $\text{Del}(P)$ can be obtained from any triangulation of $P$ by Delaunay flips, which consist in deleting the diagonal of a convex quadrilateral if it is not locally Delaunay, and replacing it by the other diagonal of the quadrilateral.

Lemma 6. Let $P$ be a set of points $p_i = (x_i, y_i, 0)$ in the plane, and let $P^*$ be the set of their vertical projections $p^* = (x_i, y_i, x_i^2 + y_i^2)$ onto the unit paraboloid. Producing a Delaunay flip in a triangulation of $P$ corresponds to “sticking” a tetrahedron from below to the corresponding polyhedrization of $P^*$. 
DELAUNAY FLIPS

We intend to prove that $\text{Del}(P)$ can be obtained from any triangulation of $P$ by Delaunay flips, which consist in deleting the diagonal of a convex quadrilateral if it is not locally Delaunay, and replacing it by the other diagonal of the quadrilateral.

Lemma 6. Let $P$ be a set of points $p_i = (x_i, y_i, 0)$ in the plane, and let $P^*$ be the set of their vertical projections $p^* = (x_i, y_i, x_i^2 + y_i^2)$ onto the unit paraboloid. Producing a Delaunay flip in a triangulation of $P$ corresponds to “sticking” a tetrahedron from below to the corresponding polyhedrization of $P^*$. 
DELAUNAY FLIPS

We intend to prove that $\text{Del}(P)$ can be obtained from any triangulation of $P$ by Delaunay flips, which consist in deleting the diagonal of a convex quadrilateral if it is not locally Delaunay, and replacing it by the other diagonal of the quadrilateral.

Lemma 6. Let $P$ be a set of points $p_i = (x_i, y_i, 0)$ in the plane, and let $P^*$ be the set of their vertical projections $p^* = (x_i, y_i, x_i^2 + y_i^2)$ onto the unit paraboloid. Producing a Delaunay flip in a triangulation of $P$ corresponds to “sticking” a tetrahedron from below to the corresponding polyhedrization of $P^*$. 

Computational Geometry, Facultat d'Informàtica de Barcelona, UPC
DELAUNAY FLIPS

We intend to prove that \( \text{Del}(P) \) can be obtained from any triangulation of \( P \) by Delaunay flips, which consist in deleting the diagonal of a convex quadrilateral if it is not locally Delaunay, and replacing it by the other diagonal of the quadrilateral.

**Lemma 6.** Let \( P \) be a set of points \( p_i = (x_i, y_i, 0) \) in the plane, and let \( P^* \) be the set of their vertical projections \( p^* = (x_i, y_i, x_i^2 + y_i^2) \) onto the unit paraboloid. Producing a Delaunay flip in a triangulation of \( P \) corresponds to “sticking” a tetrahedron from below to the corresponding polyhedrization of \( P^* \).

Once flipped, the quadrilateral is locally Delaunay: the fourth point lies in the exterior of the circumcircle of the triangle.

In the paraboloid, this means that the fourth point lies above the triangular face of the polyhedrization.
DELAUNAY FLIPS

We intend to prove that $Del(P)$ can be obtained from any triangulation of $P$ by Delaunay flips, which consist in deleting the diagonal of a convex quadrilateral if it is not locally Delaunay, and replacing it by the other diagonal of the quadrilateral.

**Corollary.** Given any triangulation of $P$, performing locally Delaunay flips is a procedure converging to $Del(P)$. 
1. Compute the Voronoi diagram by any of the known methods and dualize it.
1. Compute the Voronoi diagram by any of the known methods and dualize it.

2. Project the points onto the paraboloid, compute the 3D convex hull by any of the known methods, and project it back onto the plane.
ALGORITHMS

1. Compute the Voronoi diagram by any of the known methods and dualize it.

2. Project the points onto the paraboloid, compute the 3D convex hull by any of the known methods, and project it back onto the plane.

3. Compute a triangulation, by any of the known methods, and apply Delaunay flips.
ALGORITHMS

1. Compute the Voronoi diagram by any of the known methods and dualize it.

2. Project the points onto the paraboloid, compute the 3D convex hull by any of the known methods, and project it back onto the plane.

3. Compute a triangulation, by any of the known methods, and apply Delaunay flips.

4. Incremental algorithm
   
   Compute an enclosing triangle for \( \{p_1, \ldots, p_n\} \)
   
   Compute \( \text{Del}(p_1, \ldots, p_{i+1}) \) from \( \text{Del}(p_1, \ldots, p_i) \)
INCREMENTAL ALGORITHM

Let $D_i = Del(p_1, \ldots, p_i)$ and $p = p_{i+1}$. 
Let $D_i = Del(p_1, \ldots, p_i)$ and $p = p_{i+1}$.

Observation 1. If $qrs$ is the triangle of $D_i$ containing $p$, then $pq$, $pr$ and $ps$ are edges of $D_{i+1}$. 
Let $D_i = Del(p_1, \ldots, p_i)$ and $p = p_{i+1}$.

**Observation 1.** If $qrs$ is the triangle of $D_i$ containing $p$, then $\overline{pq}$, $\overline{pr}$ and $\overline{ps}$ are edges of $D_{i+1}$.

As $C_{qrs}$ is empty, there exist empty circles $C_{pq}$, such as the circle through $p$ and $q$ tangent to $C_{qrs}$ in $q$. Similarly for $r$ and $s$. 
DELAUNAY TRIANGULATION

INCREMENTAL ALGORITHM

Let $D_i = Del(p_1, \ldots, p_i)$ and $p = p_{i+1}$.

Observation 1. If $qrs$ is the triangle of $D_i$ containing $p$, then $pq$, $pr$ and $ps$ are edges of $D_{i+1}$.

Observation 2. Let $pqr$ be a triangle incident to $p$. The edge $qr$ may not be a Delaunay edge.
Let $D_i = Del(p_1, \ldots, p_i)$ and $p = p_{i+1}$.

**Observation 1.** If $qrs$ is the triangle of $D_i$ containing $p$, then $pq$, $pr$ and $ps$ are edges of $D_{i+1}$.

**Observation 2.** Let $pqr$ be a triangle incident to $p$. The edge $qr$ may not be a Delaunay edge.

Since $p$ may lie in the interior of $C_{qrt}$. 
Let \( D_i = \text{Del}(p_1, \ldots, p_i) \) and \( p = p_{i+1} \).

**Observation 1.** If \( qrs \) is the triangle of \( D_i \) containing \( p \), then \( pq, pr \) and \( ps \) are edges of \( D_{i+1} \).

**Observation 2.** Let \( pqr \) be a triangle incident to \( p \). The edge \( qr \) may not be a Delaunay edge.

**Observation 3.** The insertion of the point \( p \) can only violate the Delaunay property of the triangles incident to \( p \).
INCREMENTAL ALGORITHM

Let $D_i = Del(p_1, \ldots, p_i)$ and $p = p_{i+1}$.

**Observation 1.** If $qrs$ is the triangle of $D_i$ containing $p$, then $pq$, $pr$ and $ps$ are edges of $D_{i+1}$.

**Observation 2.** Let $pqr$ be a triangle incident to $p$. The edge $qr$ may not be a Delaunay edge.

**Observation 3.** The insertion of the point $p$ can only violate the Delaunay property of the triangles incident to $p$.

Obvious, because the property is local: it affects only quadrilaterals formed by two triangles sharing an edge.
INCREMENTAL ALGORITHM

Let $D_i = Del(p_1, \ldots, p_i)$ and $p = p_{i+1}$.

**Observation 1.** If $qrs$ is the triangle of $D_i$ containing $p$, then $pq$, $pr$ and $ps$ are edges of $D_{i+1}$.

**Observation 2.** Let $pqr$ be a triangle incident to $p$. The edge $qr$ may not be a Delaunay edge.

**Observation 3.** The insertion of the point $p$ can only violate the Delaunay property of the triangles incident to $p$.

**Algorithm**

Each time a new point is added to the triangulation, and before adding the next point, the following routine is executed:

**Flips**

While there are still triangles incident to $p$ non locally Delaunay, flip them.
INCREMEENTAL ALGORITHM

Let \( D_i = Del(p_1, \ldots, p_i) \) and \( p = p_{i+1} \).

**Observation 1.** If \( qrs \) is the triangle of \( D_i \) containing \( p \), then \( pq, pr \) and \( ps \) are edges of \( D_{i+1} \).

**Observation 2.** Let \( pqr \) be a triangle incident to \( p \). The edge \( qr \) may not be a Delaunay edge.

**Observation 3.** The insertion of the point \( p \) can only violate the Delaunay property of the triangles incident to \( p \).

**Algorithm**

Each time a new point is added to the triangulation, and before adding the next point, the following routine is executed:

**Flips**

While there are still triangles incident to \( p \) non locally Delaunay, flip them.
INCREMEantal ALGORITHM

Let $D_i = Del(p_1, \ldots, p_i)$ and $p = p_{i+1}$.

Observation 1. If $qrs$ is the triangle of $D_i$ containing $p$, then $pq$, $pr$ and $ps$ are edges of $D_{i+1}$.

Observation 2. Let $pqr$ be a triangle incident to $p$. The edge $qr$ may not be a Delaunay edge.

Observation 3. The insertion of the point $p$ can only violate the Delaunay property of the triangles incident to $p$.

Algorithm

Each time a new point is added to the triangulation, and before adding the next point, the following routine is executed:

Flips

While there are still triangles incident to $p$ non locally Delaunay, flip them.
DELAUNAY TRIANGULATION

INCREMENTAL ALGORITHM

Let \( D_i = Del(p_1, \ldots, p_i) \) and \( p = p_{i+1} \).

Observation 1. If \( qrs \) is the triangle of \( D_i \) containing \( p \), then \( \overline{pq}, \overline{pr} \) and \( \overline{ps} \) are edges of \( D_{i+1} \).

Observation 2. Let \( pqr \) be a triangle incident to \( p \). The edge \( \overline{qr} \) may not be a Delaunay edge.

Observation 3. The insertion of the point \( p \) can only violate the Delaunay property of the triangles incident to \( p \).

Algorithm

Each time a new point is added to the triangulation, and before adding the next point, the following routine is executed:

Flips

While there are still triangles incident to \( p \) non locally Delaunay, flip them.
INCREMENTAL ALGORITHM

Let $D_i = Del(p_1, \ldots, p_i)$ and $p = p_{i+1}$.

Observation 1. If $qrs$ is the triangle of $D_i$ containing $p$, then $pq$, $pr$ and $ps$ are edges of $D_{i+1}$.

Observation 2. Let $pqr$ be a triangle incident to $p$. The edge $qr$ may not be a Delaunay edge.

Observation 3. The insertion of the point $p$ can only violate the Delaunay property of the triangles incident to $p$.

Algorithm

Each time a new point is added to the triangulation, and before adding the next point, the following routine is executed:

Flips

While there are still triangles incident to $p$ non locally Delaunay, flip them.
DELAUNAY TRIANGULATION

INCREMENTAL ALGORITHM

Let $D_i = Del(p_1, \ldots, p_i)$ and $p = p_{i+1}$.

Observation 1. If $qrs$ is the triangle of $D_i$ containing $p$, then $pq$, $pr$ and $ps$ are edges of $D_{i+1}$.

Observation 2. Let $pqr$ be a triangle incident to $p$. The edge $qr$ may not be a Delaunay edge.

Observation 3. The insertion of the point $p$ can only violate the Delaunay property of the triangles incident to $p$.

Algorithm

Each time a new point is added to the triangulation, and before adding the next point, the following routine is executed:

Flips

While there are still triangles incident to $p$ non locally Delaunay, flip them.
INCREMENATL ALGORITHM

Let $D_i = Del(p_1, \ldots, p_i)$ and $p = p_{i+1}$.

Observation 1. If $qrs$ is the triangle of $D_i$ containing $p$, then $pq$, $pr$ and $ps$ are edges of $D_{i+1}$.

Observation 2. Let $pqr$ be a triangle incident to $p$. The edge $qr$ may not be a Delaunay edge.

Observation 3. The insertion of the point $p$ can only violate the Delaunay property of the triangles incident to $p$.

Algorithm

Each time a new point is added to the triangulation, and before adding the next point, the following routine is executed:

Flips

While there are still triangles incident to $p$ non locally Delaunay, flip them.
INCREMENTAL ALGORITHM

Let \( D_i = Del(p_1, \ldots, p_i) \) and \( p = p_{i+1} \).

**Observation 1.** If \( qrs \) is the triangle of \( D_i \) containing \( p \), then \( \overline{pq}, \overline{pr} \) and \( \overline{ps} \) are edges of \( D_{i+1} \).

**Observation 2.** Let \( pqr \) be a triangle incident to \( p \). The edge \( \overline{qr} \) may not be a Delaunay edge.

**Observation 3.** The insertion of the point \( p \) can only violate the Delaunay property of the triangles incident to \( p \).

**Algorithm**

Each time a new point is added to the triangulation, and before adding the next point, the following routine is executed:

**Flips**

While there are still triangles incident to \( p \) non locally Delaunay, flip them.
INCREMENTAL ALGORITHM

Let $D_i = Del(p_1, \ldots, p_i)$ and $p = p_{i+1}$.

Observation 1. If $qrs$ is the triangle of $D_i$ containing $p$, then $pq$, $pr$ and $ps$ are edges of $D_{i+1}$.

Observation 2. Let $pqr$ be a triangle incident to $p$. The edge $qr$ may not be a Delaunay edge.

Observation 3. The insertion of the point $p$ can only violate the Delaunay property of the triangles incident to $p$.

Algorithm

Each time a new point is added to the triangulation, and before adding the next point, the following routine is executed:

Flips

While there are still triangles incident to $p$ non locally Delaunay, flip them.
INCREMEental ALGORITHM

Let $D_i = Del(p_1, \ldots, p_i)$ and $p = p_{i+1}$.

Observation 1. If $qrs$ is the triangle of $D_i$ containing $p$, then $pq$, $pr$ and $ps$ are edges of $D_{i+1}$.

Observation 2. Let $pqr$ be a triangle incident to $p$. The edge $qr$ may not be a Delaunay edge.

Observation 3. The insertion of the point $p$ can only violate the Delaunay property of the triangles incident to $p$.

Algorithm

Each time a new point is added to the triangulation, and before adding the next point, the following routine is executed:

Flips

While there are still triangles incident to $p$ non locally Delaunay, flip them.
DELAUNAY TRIANGULATION

INCREMENTAL ALGORITHM

Let $D_i = Del(p_1, \ldots, p_i)$ and $p = p_{i+1}$.

Observation 1. If $qrs$ is the triangle of $D_i$ containing $p$, then $\overline{pq}$, $\overline{pr}$ and $\overline{ps}$ are edges of $D_{i+1}$.

Observation 2. Let $pqr$ be a triangle incident to $p$. The edge $\overline{qr}$ may not be a Delaunay edge.

Observation 3. The insertion of the point $p$ can only violate the Delaunay property of the triangles incident to $p$.

Algorithm

Each time a new point is added to the triangulation, and before adding the next point, the following routine is executed:

Flips

While there are still triangles incident to $p$ non locally Delaunay, flip them.

Computational Geometry, Facultat d'Informàtica de Barcelona, UPC
INCREMEENTAL ALGORITHM

Let $D_i = Del(p_1, \ldots, p_i)$ and $p = p_{i+1}$.

Observation 1. If $qrs$ is the triangle of $D_i$ containing $p$, then $pq$, $pr$ and $ps$ are edges of $D_{i+1}$.

Observation 2. Let $pqr$ be a triangle incident to $p$. The edge $qr$ may not be a Delaunay edge.

Observation 3. The insertion of the point $p$ can only violate the Delaunay property of the triangles incident to $p$.

Algorithm

Each time a new point is added to the triangulation, and before adding the next point, the following routine is executed:

Flips

While there are still triangles incident to $p$ non locally Delaunay, flip them.
INCREMENTAL ALGORITHM

Let $D_i = Del(p_1, \ldots, p_i)$ and $p = p_{i+1}$.

Observation 1. If $qrs$ is the triangle of $D_i$ containing $p$, then $pq$, $pr$ and $ps$ are edges of $D_{i+1}$.

Observation 2. Let $pqr$ be a triangle incident to $p$. The edge $qr$ may not be a Delaunay edge.

Observation 3. The insertion of the point $p$ can only violate the Delaunay property of the triangles incident to $p$.

Algorithm

Each time a new point is added to the triangulation, and before adding the next point, the following routine is executed:

Flips

While there are still triangles incident to $p$ non locally Delaunay, flip them.
INCREMENTAL ALGORITHM

Let \( D_i = Del(p_1, \ldots, p_i) \) and \( p = p_{i+1} \).

Observation 1. If \( qrs \) is the triangle of \( D_i \) containing \( p \), then \( pq, pr \) and \( ps \) are edges of \( D_{i+1} \).

Observation 2. Let \( pqr \) be a triangle incident to \( p \). The edge \( qr \) may not be a Delaunay edge.

Observation 3. The insertion of the point \( p \) can only violate the Delaunay property of the triangles incident to \( p \).

Algorithm

Each time a new point is added to the triangulation, and before adding the next point, the following routine is executed:

Flips

While there are still triangles incident to \( p \) non locally Delaunay, flip them.
DELAUNAY TRIANGULATION

INCREMENTAL ALGORITHM

Let $D_i = Del(p_1, \ldots, p_i)$ and $p = p_{i+1}$.

Observation 1. If $qrs$ is the triangle of $D_i$ containing $p$, then $pq$, $pr$ and $ps$ are edges of $D_{i+1}$.

Observation 2. Let $pqr$ be a triangle incident to $p$. The edge $qr$ may not be a Delaunay edge.

Observation 3. The insertion of the point $p$ can only violate the Delaunay property of the triangles incident to $p$.

Algorithm

Each time a new point is added to the triangulation, and before adding the next point, the following routine is executed:

Flips

While there are still triangles incident to $p$ non locally Delaunay, flip them.

Added running time

The added running time of performing the flips when adding $p_i$ is

$$O(\text{degree of } p_i \text{ in } D_i) = O(n).$$
DELAUNAY TRIANGULATION

INCREMENTAL ALGORITHM

Let \( D_i = \text{Del}(p_1, \ldots, p_i) \) and \( p = p_{i+1} \).

Observation 1. If \( qrs \) is the triangle of \( D_i \) containing \( p \), then \( pq, pr \) and \( ps \) are edges of \( D_{i+1} \).

Observation 2. Let \( pqr \) be a triangle incident to \( p \). The edge \( qr \) may not be a Delaunay edge.

Observation 3. The insertion of the point \( p \) can only violate the Delaunay property of the triangles incident to \( p \).

Algorithm

Each time a new point is added to the triangulation, and before adding the next point, the following routine is executed:

Flips

While there are still triangles incident to \( p \) non locally Delaunay, flip them.

Added running time

The added running time of performing the flips when adding \( p_i \) is

\[
O(\text{degree of } p_i \text{ in } D_i) = O(n).
\]

As the average order is smaller than 6, the expected added running time is not \( O(n^2) \) but simply \( O(n) \).
Among all the triangulations of $P$, the Delaunay triangulation maximizes the minimum angle (the angles of $Del(P)$ are less acute).
Among all the triangulations of $P$, the Delaunay triangulation maximizes the minimum angle (the angles of $\text{Del}(P)$ are less acute).

Let us be more precise:

If $\mathcal{T} = \{T_1, \ldots, T_t\}$ is a triangulation of $P$, the “fineness” of $\mathcal{T}$ is the increasingly sorted list of the angles of all the triangles $T_i$ of $\mathcal{T}$: $F(\mathcal{T}) = (\alpha_1, \ldots, \alpha_{3t})$. 
Among all the triangulations of $P$, the Delaunay triangulation maximizes the minimum angle (the angles of $\text{Del}(P)$ are less acute).

Let us be more precise:

If $\mathcal{T} = \{T_1, \ldots, T_t\}$ is a triangulation of $P$, the “fineness” of $\mathcal{T}$ is the increasingly sorted list of the angles of all the triangles $T_i$ of $\mathcal{T}$: $F(\mathcal{T}) = (\alpha_1, \ldots, \alpha_{3t})$.

Since every triangulation of $P$ has $t = 2n - h - 2$ triangles, these $3t$-tuples can be compared and lexicographically sorted.
DELAUNAY TRIANGULATION

DELAUNAY TRIANGULATION AND EQUIANGULARITY

Among all the triangulations of $P$, the Delaunay triangulation maximizes the minimum angle (the angles of $\text{Del}(P)$ are less acute).

Let us be more precise:

If $\mathcal{T} = \{T_1, \ldots, T_t\}$ is a triangulation of $P$, the “fineness” of $\mathcal{T}$ is the increasingly sorted list of the angles of all the triangles $T_i$ of $\mathcal{T}$: $F(\mathcal{T}) = (\alpha_1, \ldots, \alpha_{3t})$.

Since every triangulation of $P$ has $t = 2n - h - 2$ triangles, these $3t$-tuples can be compared and lexicographically sorted.

The Delaunay triangulation maximizes the “fineness”:

$$F(\text{Del}(P)) \geq F(\mathcal{T}), \; \forall \mathcal{T} \text{ triangulation of } P.$$
Among all the triangulations of $P$, the Delaunay triangulation maximizes the minimum angle (the angles of $Del(P)$ are less acute).

Let us be more precise:

If $T = \{T_1, \ldots, T_t\}$ is a triangulation of $P$, the “fineness” of $T$ is the increasingly sorted list of the angles of all the triangles $T_i$ of $T$: $F(T) = (\alpha_1, \ldots, \alpha_{3t})$.

Since every triangulation of $P$ has $t = 2n - h - 2$ triangles, these $3t$-tuples can be compared and lexicographically sorted.

The Delaunay triangulation maximizes the “fineness”:

$$F(Del(P)) \geq F(T), \quad \forall T \text{ triangulation of } P.$$ 

The proof of this statement requires a last lemma.
Lemma 7. Let $a$, $b$, $c$ and $d$ be four points forming a convex quadrilateral, in counterclockwise order. Let $\mathcal{T}$ and $\mathcal{T}'$ be the two possible triangulations of the quadrilateral: $\mathcal{T}$ uses the diagonal $ac$ and $\mathcal{T}'$ uses $bd$. Let $\epsilon$ and $\epsilon'$ respectively be the minimum angles of $\mathcal{T}$ and $\mathcal{T}'$. Then:

\[ \epsilon > \epsilon' \iff d \in \text{ext}(C_{abc}) \]
\[ \epsilon = \epsilon' \iff d \in \partial(C_{abc}) \]
\[ \epsilon < \epsilon' \iff d \in \text{int}(C_{abc}) \]
Due to the symmetry of the problem, we only need to prove that $\epsilon > \epsilon' \iff d \in ext(C_{abc})$. 

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{diagram.png}
\end{figure}
Due to the symmetry of the problem, we only need to prove that $\epsilon > \epsilon' \iff d \in ext(C_{abc})$.

If $\epsilon > \epsilon'$, then $d \in ext(C_{abc})$.
Due to the symmetry of the problem, we only need to prove that $\epsilon > \epsilon' \iff d \in ext(C_{abc})$.

If $\epsilon > \epsilon'$, then $d \in ext(C_{abc})$

If $\epsilon > \epsilon'$, then $\epsilon'$ cannot be $\alpha$, nor $\gamma$. 
Due to the symmetry of the problem, we only need to prove that $\epsilon > \epsilon' \iff d \in \text{ext}(C_{abc})$.

If $\epsilon > \epsilon'$, then $d \in \text{ext}(C_{abc})$.

- If $\epsilon > \epsilon'$, then $\epsilon'$ cannot be $\alpha$, nor $\gamma$.
- If $\epsilon' = \delta_2$, then $\delta_2 = \epsilon' < \epsilon \leq \alpha_1$ and, therefore, $d \in \text{ext}(C_{abc})$. 
Due to the symmetry of the problem, we only need to prove that $\epsilon > \epsilon' \iff d \in ext(C_{abc})$.

If $\epsilon > \epsilon'$, then $d \in ext(C_{abc})$

- If $\epsilon > \epsilon'$, then $\epsilon'$ cannot be $\alpha$, nor $\gamma$.
- If $\epsilon' = \delta_2$, then $\delta_2 = \epsilon' < \epsilon \leq \alpha_1$ and, therefore, $d \in ext(C_{abc})$.
- If $\epsilon' = \delta_1$, then $\delta_1 = \epsilon' < \epsilon \leq \gamma_1$ and, therefore, $d \in ext(C_{abc})$. 
Due to the symmetry of the problem, we only need to prove that $\epsilon > \epsilon' \iff d \in ext(C_{abc})$.

If $\epsilon > \epsilon'$, then $d \in ext(C_{abc})$

If $\epsilon > \epsilon'$, then $\epsilon'$ cannot be $\alpha$, nor $\gamma$.

If $\epsilon' = \delta_2$, then $\delta_2 = \epsilon' < \epsilon \leq \alpha_1$ and, therefore, $d \in ext(C_{abc})$.

If $\epsilon' = \delta_1$, then $\delta_1 = \epsilon' < \epsilon \leq \gamma_1$ and, therefore, $d \in ext(C_{abc})$.

If $\epsilon' = \beta_1$, then $\beta_1 = \epsilon' < \epsilon \leq \gamma_2$ and, therefore, $d \in ext(C_{abc})$. 
Due to the symmetry of the problem, we only need to prove that $\epsilon > \epsilon' \iff d \in \text{ext}(C_{abc})$.

If $\epsilon > \epsilon'$, then $d \in \text{ext}(C_{abc})$

   If $\epsilon > \epsilon'$, then $\epsilon'$ cannot be $\alpha$, nor $\gamma$.

   If $\epsilon' = \delta_2$, then $\delta_2 = \epsilon' < \epsilon \leq \alpha_1$ and, therefore, $d \in \text{ext}(C_{abc})$.

   If $\epsilon' = \delta_1$, then $\delta_1 = \epsilon' < \epsilon \leq \gamma_1$ and, therefore, $d \in \text{ext}(C_{abc})$.

   If $\epsilon' = \beta_1$, then $\beta_1 = \epsilon' < \epsilon \leq \gamma_2$ and, therefore, $d \in \text{ext}(C_{abc})$.

   If $\epsilon' = \beta_2$, then $\beta_2 = \epsilon' < \epsilon \leq \alpha_2$ and, therefore, $d \in \text{ext}(C_{abc})$. 
Due to the symmetry of the problem, we only need to prove that $\epsilon > \epsilon' \iff d \in \text{ext}(C_{abc})$.

If $\epsilon > \epsilon'$, then $d \in \text{ext}(C_{abc})$.
Due to the symmetry of the problem, we only need to prove that $\epsilon > \epsilon' \iff d \in \text{ext}(C_{abc})$. If $\epsilon > \epsilon'$, then $d \in \text{ext}(C_{abc})$. If $d \in \text{ext}(C_{abc})$, then $\epsilon > \epsilon'$. 
Due to the symmetry of the problem, we only need to prove that \( \epsilon > \epsilon' \iff d \in \text{ext}(C_{abc}) \).

If \( \epsilon > \epsilon' \), then \( d \in \text{ext}(C_{abc}) \)

If \( d \in \text{ext}(C_{abc}) \), then \( \epsilon > \epsilon' \)

- If \( \epsilon = \beta \), then \( \epsilon = \beta > \beta_1 \geq \epsilon' \)
- If \( \epsilon = \delta \), then \( \epsilon = \delta > \delta_1 \geq \epsilon' \)
Due to the symmetry of the problem, we only need to prove that $\epsilon > \epsilon' \iff d \in \text{ext}(C_{abc})$.

If $\epsilon > \epsilon'$, then $d \in \text{ext}(C_{abc})$.

If $d \in \text{ext}(C_{abc})$, then $\epsilon > \epsilon'$.

If $\epsilon = \beta$, then $\epsilon = \beta > \beta_1 \geq \epsilon'$

If $\epsilon = \delta$, then $\epsilon = \delta > \delta_1 \geq \epsilon'$

If $\epsilon = \alpha_1$, then $\epsilon = \alpha_1 = \overline{\alpha_1} > \delta_2 \geq \epsilon'$
Due to the symmetry of the problem, we only need to prove that $\epsilon > \epsilon' \iff d \in \text{ext}(C_{abc})$.

If $\epsilon > \epsilon'$, then $d \in \text{ext}(C_{abc})$

If $d \in \text{ext}(C_{abc})$, then $\epsilon > \epsilon'$

  - If $\epsilon = \beta$, then $\epsilon = \beta > \beta_1 \geq \epsilon'$
  - If $\epsilon = \delta$, then $\epsilon = \delta > \delta_1 \geq \epsilon'$
  - If $\epsilon = \alpha_1$, then $\epsilon = \alpha_1 = \alpha_1 > \delta_2 \geq \epsilon'$
  - If $\epsilon = \alpha_2$, then $\epsilon = \alpha_2 = \alpha_2 > \beta_2 \geq \epsilon'$
  - If $\epsilon = \gamma_1$, then $\epsilon = \gamma_1 = \gamma_1 > \delta_1 \geq \epsilon'$
  - If $\epsilon = \gamma_2$, then $\epsilon = \gamma_2 = \gamma_2 > \beta_1 \geq \epsilon'$
Corollary. The Delaunay triangulation is the most equiangular among all triangulations of a given set of points.
Corollary. The Delaunay triangulation is the most equiangular among all triangulations of a given set of points.

If $P$ does not contain four or more concyclic points, it follows from the previous lemma.
Corollary. The Delaunay triangulation is the most equiangular among all triangulations of a given set of points.

If $P$ does not contain four or more concyclic points, it follows from the previous lemma.

If $P$ contains four or more concyclic points, $\text{Del}(P)$ contains a polygon inscribed in a circle which can be triangulated in several ways. Nevertheless, Lemma 1 (on the geometrical locus of all the points from which a segment is seen under a given angle) guarantees that every triangulation of a polygon inscribed in a circle has the same fineness, since each edge of the polygon belongs to a triangle, and every possible triangle gives rise to the same angle.
The Delaunay triangulation is used to interpolate terrains, because it also minimizes the roughness of the terrain, in other words, the integral of the square of the $L_2$-norm of the terrain’s gradient.

It is important to notice that this property is independent from the data, in other words, it is independent from the values of the $z$-coordinates of the input points.
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*