Delaunay triangulations: properties, algorithms, and complexity

Olivier Devillers
Delaunay triangulations: properties, algorithms, and complexity

Extra question: what is the difference between an algorithm and a program?
Algorithm

Recipe to go from the input to the output

Formalized description in some language

May use data structure

Proof of correctness

Complexity analysis
Delaunay Triangulation: definition, empty circle property
Delaunay Triangulation: definition, empty circle property

Point set
Delaunay Triangulation: definition, empty circle property

Point set

Query
Delaunay Triangulation: definition, empty circle property

Point set

Query
Delaunay Triangulation: definition, empty circle property

Point set

Query
Delaunay Triangulation: definition, empty circle property

Point set

Query

Nearest neighbor
Delaunay Triangulation: definition, empty circle property

Point set

Voronoi diagram
Delaunay Triangulation: definition, empty circle property

Point set

Voronoi diagram
Delaunay Triangulation: definition, empty circle property

Point set

Voronoi diagram
Delaunay Triangulation: definition, empty circle property

Point set

Voronoi diagram
Delaunay Triangulation: definition, empty circle property

Point set

Voronoi diagram

Delaunay triangulation
Delaunay Triangulation: definition, empty circle property

Point set

Delaunay triangulation

Empty circle property
**Delaunay Triangulation:** definition, empty circle property

- **Point set**
- **Delaunay triangulation**
- **Empty circle property**
Delaunay Triangulation: size
Delaunay Triangulation: size

<table>
<thead>
<tr>
<th>Vertices</th>
<th>Edges</th>
<th>Faces</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>$e$</td>
<td>$t+1$</td>
</tr>
</tbody>
</table>

Euler relation

$$n - e + t + 1 = 2$$
Delaunay Triangulation:  

<table>
<thead>
<tr>
<th>Vertices</th>
<th>Edges</th>
<th>Faces</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>$e$</td>
<td>$t + 1$</td>
</tr>
</tbody>
</table>

Euler relation:  
$n - e + t + 1 = 2$

Triangular faces:  
$3t + k = 2e$
Delaunay Triangulation: size

<table>
<thead>
<tr>
<th>Vertices</th>
<th>Edges</th>
<th>Faces</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>$e$</td>
<td>$t+1$</td>
</tr>
</tbody>
</table>

Euler relation:  

$$n - e + t + 1 = 2$$

Triangular faces:  

$$3t + k = 2e$$

$$t = 2n - k - 2 < 2n$$

$$e = 3n - k - 3 < 3n$$
Delaunay Triangulation: size

<table>
<thead>
<tr>
<th>Vertices</th>
<th>Edges</th>
<th>Faces</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>$e$</td>
<td>$t + 1$</td>
</tr>
</tbody>
</table>

Euler relation

$$n - e + t + 1 = 2$$

Triangular faces

$$3t + k = 2e$$

$$\sum_{p \in S} d^\circ(p) = 2e = 6n - 2k - 6$$

$$\mathbb{E}(d^\circ(p)) = \frac{1}{n} \sum_{p \in S} d^\circ(p) < 6$$

average on the choice of point $p$ in set of points $S$
Delaunay Triangulation: max-min angle
Delaunay Triangulation: max-min angle

Triangulation

Delaunay
Delaunay Triangulation: max-min angle

Triangulation

Delaunay

smallest angle
Delaunay Triangulation: max-min angle

Triangulation

Delaunay

smallest angle
Delaunay Triangulation: max-min angle

Triangulation

Delaunay

smallest angle
Delaunay Triangulation: max-min angle

Triangulation

smallest angle

second smallest angle

Delaunay
Delaunay Triangulation: max-min angle

Proof
Delaunay Triangulation: max-min angle

Definition

Delaunay edge
Delaunay Triangulation: max-min angle

Definition

Delaunay edge

∃ empty circle
Delaunay Triangulation: max-min angle

Definition

locally Delaunay edge w.r.t. a triangulation
Delaunay Triangulation: max-min angle

Definition

locally Delaunay edge w.r.t. a triangulation

∃ circle
not enclosing the two neighbors

neighbor = visible from the edge
Delaunay Triangulation: max-min angle

Lemma \((\forall \text{ edge: locally Delaunay)} \iff \text{Delaunay}\)
Delaunay Triangulation: max-min angle

Lemma \((\forall \text{ edge: locally Delaunay}) \iff \text{Delaunay}\)

Proof:

choose an edge
Delaunay Triangulation: max-min angle

Lemma \((\forall \text{ edge}: \text{ locally Delaunay}) \iff \text{ Delaunay}\)

Proof:

- choose an edge
- edges of the quadrilateral are locally Delaunay
Delaunay Triangulation: max-min angle

Lemma \((\forall \text{ edge: locally Delaunay}) \iff \text{ Delaunay}\)

Proof:

- choose an edge
- edges of the quadrilateral are locally Delaunay
Delaunay Triangulation: max-min angle

Lemma \((\forall \text{ edge: locally Delaunay}) \iff \text{Delaunay}\)

Proof:

- choose an edge
- edges of the quadrilateral are locally Delaunay
- Vertices visible through one edge are outside circle
Delaunay Triangulation: max-min angle

Lemma \( (\forall \text{ edge: locally Delaunay}) \iff \text{Delaunay} \)

Proof:

- choose an edge
- edges of the quadrilateral are locally Delaunay
- Vertices visible through one edge are outside circle
- Induction \( \rightarrow \) all vertices outside circle
Delaunay Triangulation: max-min angle

Lemma
For four points in convex position

Delaunay $\iff$ maximize the smallest angle

Two possible triangulation
Delaunay Triangulation: \text{max-min angle}

Lemma  For four points in convex position

Delaunay $\iff$ maximize the smallest angle

Case 1: smallest angle in corner
Delaunay Triangulation: max-min angle

Lemma For four points in convex position

Delaunay $\iff$ maximize the smallest angle

Case 1: smallest angle in corner

$\exists$ a smaller angle $\in$ other triangulation
Delaunay Triangulation: max-min angle

Lemma  For four points in convex position

Delaunay $\iff$ maximize the smallest angle

Case 2: smallest angle along diagonal

\[ \delta \]
Delaunay Triangulation: max-min angle

Lemma  For four points in convex position

Delaunay $\iff$ maximize the smallest angle

Case 2: smallest angle along diagonal
Delaunay Triangulation: max-min angle

Lemma
For four points in convex position

Delaunay \iff maximize the smallest angle

Case 2: smallest angle along diagonal

\exists a smaller angle \in other triangulation
Delaunay Triangulation: max-min angle

Map: Triangulations $\rightarrow \mathbb{R}^{6n-3k-4}$ smallest angle $\alpha_1$
Delaunay Triangulation: max-min angle

Map: Triangulations $\rightarrow \mathbb{R}^{6n-3k-4}$

smallest angle $\alpha_1$
second smallest angle $\alpha_2$
Delaunay Triangulation: \text{max-min angle}

Map: Triangulations $\mapsto \mathbb{R}^{6n-3k-4}$

- Smallest angle $\alpha_1$
- Second smallest angle $\alpha_2$
- Third smallest angle $\alpha_3$
Delaunay Triangulation: \( \text{max-min angle} \)

Map: Triangulations \( \rightarrow \mathbb{R}^{6n-3k-4} \)

\[ (\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{6n-3k-4}) \]

- Smallest angle \( \alpha_1 \)
- Second smallest angle \( \alpha_2 \)
- Third smallest angle \( \alpha_3 \)
Delaunay Triangulation: max-min angle

Map: Triangulations $\rightarrow \mathbb{R}^{6n-3k-4}$

$(\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{6n-3k-4})$

smallest angle $\alpha_1$
second smallest angle $\alpha_2$
third smallest angle $\alpha_3$

sort triangulations in lexicographic order
Delaunay Triangulation: \( \text{max-min angle} \)

Theorem:

Delaunay maximizes minimum angles (in lexicographic order)
Delaunay Triangulation: max-min angle

Theorem:

Delaunay maximizes minimum angles (in lexicographic order)

Proof:

Let $T$ be the triangulation maximizing angles
Delaunay Triangulation: max-min angle

Theorem:

Delaunay maximizes minimum angles (in lexicographic order)

Proof:

Let $T$ be the triangulation maximizing angles

$\implies \forall$ convex quadrilateral (from 2 triangles $\in T$)

the diagonal maximizes smallest angle (in quad)
Delaunay Triangulation: max-min angle

Theorem:
Delaunay maximizes minimum angles (in lexicographic order)

Proof:
Let $T$ be the triangulation maximizing angles

$\Rightarrow \forall$ convex quadrilateral (from 2 triangles $\in T$)

the diagonal maximizes smallest angle (in quad)

$\Rightarrow \forall$ edge, it is locally Delaunay
Delaunay Triangulation: max-min angle

Theorem:
Delaunay maximizes minimum angles (in lexicographic order)

Proof:
Let $T$ be the triangulation maximizing angles

$\implies \forall$ convex quadrilateral (from 2 triangles $\in T$)
the diagonal maximizes smallest angle (in quad)

$\implies \forall$ edge, it is locally Delaunay

$\implies T = $ Delaunay
Delaunay triangulation

A stupid algorithm for sorting numbers
Delaunay triangulation

A stupid algorithm for sorting numbers

project on parabola
Delaunay triangulation

A stupid algorithm for sorting numbers

project on parabola
compute Delaunay triang.
Delaunay triangulation

A stupid algorithm for sorting numbers

project on parabola
compute Delaunay triang.
find lowest point
Delaunay triangulation

A stupid algorithm for sorting numbers

- Project on parabola
- Compute Delaunay triangulation
- Find lowest point
- Enumerate $x$ coordinates in ccw CH order

11 - 5
A stupid algorithm for sorting numbers

\[ O(n) \] project on parabola
\[ f(n) \] compute Delaunay triang.
\[ O(n) \] find lowest point
\[ O(n) \] enumerate \( x \) coordinates in ccw CH order

Lower bound on sorting

\[ \implies f(n) + O(n) \geq \Omega(n \log n) \]
Delaunay triangulation

A stupid algorithm for sorting numbers

\[ O(n) \text{ project on parabola} \]
\[ f(n) \text{ compute Delaunay triang.} \]
\[ O(n) \text{ find lowest point} \]
\[ O(n) \text{ enumerate } x \text{ coordinates in ccw CH order} \]

Lower bound on sorting

\[ f(n) + O(n) \geq \Omega(n \log n) \]
Delaunay Triangulation: predicates
Delaunay Triangulation: predicates

Orientation predicate

$pqr + ?$

\[
\begin{vmatrix}
  x_q - x_p & x_r - x_p \\
  y_q - y_p & y_r - y_p
\end{vmatrix} = \begin{vmatrix}
  1 & 1 & 1 \\
  x_p & x_q & x_r \\
  y_p & y_q & y_r
\end{vmatrix} > 0
\]

$pqr - ?$

\[
\begin{vmatrix}
  1 & 1 & 1 \\
  x_p & x_q & x_r \\
  y_p & y_q & y_r
\end{vmatrix} < 0
\]

$pqr 0 ?$

\[
\begin{vmatrix}
  1 & 1 & 1 \\
  x_p & x_q & x_r \\
  y_p & y_q & y_r
\end{vmatrix} = 0
\]

12 - 2
Delaunay Triangulation: predicates

- Incircle predicate

$\text{pqr ccw triangle}$

- Query $s$ inside circumcircle
Delaunay Triangulation: predicates

$pqr$ ccw triangle

query $s$ cocircular
Delaunay Triangulation: predicates

$pqr$ ccw triangle

query $s$ outside circumcircle
Delaunay Triangulation: predicates

$pqr$ ccw triangle

query $s$
Delaunay Triangulation: predicates

Space of circles

\[ p = (x, y) \sim p^* = (x, y, x^2 + y^2) \]
Delaunay Triangulation: predicates

Space of circles

- $s$ inside/outside of circle through $pqr$
- Equivalent plane through $p^*q^*r^*$
  - above/below $s^*$

incircle predicate

- Equivalent 3D orientation predicate
Delaunay Triangulation: predicates

Space of circles

$s$ inside/outside of circle through $pqr$
$
\leadsto$ plane through $p^*q^*r^*$
above/below $s^*$

incircle predicate

$\leadsto$ 3D orientation predicate

\[
\begin{array}{cccccc}
\text{sign} & 1 & 1 & 1 & 1 & 1 \\
\text{sign} & x_p & x_q & x_r & x_s \\
y_p & y_q & y_r & y_s \\
x_p^2 + y_p^2 & x_q^2 + y_q^2 & x_r^2 + y_r^2 & x_s^2 + y_s^2 \\
\end{array}
\]
Delaunay Triangulation: data structure

Data structure for (Delaunay) triangulation

Representing incidences

Representing hull boundary

Representing user’s data

...
Delaunay Triangulation: very naive algorithm
Delaunay Triangulation: very naive algorithm

For each triple of points \((p, q, r)\)

If \(pqr\) ccw

Delaunay = true;

For each point \(s\)

If \(s\) in circle \(pqr\)

Delaunay = false;

Output \(pqr\)
Delaunay Triangulation: very naive algorithm

For each triple of points \((p, q, r)\)

If \(pqr\) ccw

Delaunay = true;  \(\text{Correctness: easy}\)

For each point \(s\)

If \(s\) in circle \(pqr\)

Delaunay = false;

Output \(pqr\)
Delaunay Triangulation: very naive algorithm

For each triple of points \((p, q, r)\)

If \(pqr\) ccw

Delaunay = true;

For each point \(s\)

If \(s\) in circle \(pqr\)

Delaunay = false;

Output \(pqr\)

Correctness: easy

Complexity: \(O(n^4)\)
Delaunay Triangulation: very naive algorithm

For each triple of points \((p, q, r)\)

If \(pqr\) ccw

Delaunay = true;

Correctness: easy

For each point \(s\)

If \(s\) in circle \(pqr\)

Delaunay = false;

Complexity: \(O(n^4)\)

Output \(pqr\) does not compute incidences
Delaunay Triangulation: incremental algorithm
Delaunay Triangulation: incremental algorithm

New point
Delaunay Triangulation: incremental algorithm

New point

Locate
Delaunay Triangulation: incremental algorithm

New point
Locate

e.g.: visibility walk
Delaunay Triangulation: incremental algorithm

New point
Locate

e.g.: visibility walk
Delaunay Triangulation: incremental algorithm

New point

Locate

e.g.: visibility walk
Delaunay Triangulation: incremental algorithm

New point

Locate

e.g.: visibility walk
Delaunay Triangulation: incremental algorithm

New point
Locate

e.g.: visibility walk
Delaunay Triangulation: incremental algorithm

New point
Locate

e.g.: visibility walk
Delaunay Triangulation: incremental algorithm

New point

Locate

e.g.: visibility walk
Delaunay Triangulation: incremental algorithm

New point

Locate

e.g.: visibility walk
Delaunay Triangulation: incremental algorithm

New point
Locate

e.g.: visibility walk

not unique
Delaunay Triangulation: incremental algorithm

New point
Locate

not unique

e.g.: visibility walk
Delaunay Triangulation: incremental algorithm

Visibility walk terminates
Delaunay Triangulation: incremental algorithm

Visibility walk terminates
Delaunay Triangulation: incremental algorithm

Visibility walk terminates

May loop
Delaunay Triangulation: incremental algorithm

Visibility walk terminates

May loop

Not Delaunay
Delaunay Triangulation: incremental algorithm

Visibility walk terminates

Delaunay Triangulation: pencils of circles

Power of a point w.r.t a circle

$$\lambda \ (x^2 + y^2 - 2a'x - 2b'y + c')$$

$$+(1 - \lambda) \ (x^2 + y^2 - 2ax - 2by + c) = 0$$

blue yields smaller power
black yields smaller power
equal power
Delaunay Triangulation: incremental algorithm

Visibility walk terminates
Delaunay Triangulation: incremental algorithm

Visibility walk terminates

Green power < Red power
Delaunay Triangulation: incremental algorithm

Visibility walk terminates

Green power < Red power

Power decreases
Delaunay Triangulation: incremental algorithm

Visibility walk terminates

Green power < Red power

Power decreases

Visibility walk terminates
Delaunay Triangulation: incremental algorithm

New point
Locate
Search conflicts
Delaunay Triangulation: incremental algorithm

New point

Locate

Search conflicts
Delaunay Triangulation: incremental algorithm

New point
Locate
Search conflicts
Delaunay Triangulation: incremental algorithm

New point

Locate

Search conflicts
Delaunay Triangulation: incremental algorithm

New point
Locate
Search conflicts
Delaunay Triangulation: incremental algorithm

New point

Locate

Search conflicts
Delaunay Triangulation: incremental algorithm

New point
Locate
Search conflicts
Delaunay Triangulation: incremental algorithm

New point
Locate
Search conflicts
Delaunay Triangulation: incremental algorithm

New point

Locate

Search conflicts
Delaunay Triangulation: incremental algorithm

New point
Locate
Search conflicts
Delaunay Triangulation: incremental algorithm

New point
Locate
Search conflicts
Delaunay Triangulation: incremental algorithm

New point
Locate
Search conflicts
Delaunay Triangulation: incremental algorithm

New point
Locate
Search conflicts
Delaunay Triangulation: incremental algorithm

New point
Delaunay Triangulation: incremental algorithm

New point
Delaunay Triangulation: incremental algorithm

Proof of correctness

circle of new triangle
Delaunay Triangulation: incremental algorithm

Proof of correctness

circle of new triangle $\subset \bigcup$
Delaunay Triangulation: incremental algorithm

Complexity

Locate

Search conflicts
Delaunay Triangulation: incremental algorithm

Complexity

Locate

Search conflicts

\# triangles in conflict

\# triangles neighboring triangles in conflict
Delaunay Triangulation: incremental algorithm

Complexity

Locate

Search conflicts

\# triangles in conflict

\# triangles neighboring triangles in conflict

degree of new point in new triangulation

\( < n \)
Delaunay Triangulation: incremental algorithm

Complexity

Locate

Walk may visit all triangles

< 2n

Search conflicts

degree of new point in new triangulation

< n
Delaunay Triangulation: incremental algorithm

Complexity

Locate \( O(n) \) per insertion

Search conflicts
Delaunay Triangulation: incremental algorithm

Complexity

Locate $O(n)$ per insertion

Search conflicts $O(n^2)$ for the whole construction
Delaunay Triangulation: incremental algorithm

Complexity

Locate

Search conflicts
Delaunay Triangulation: incremental algorithm

Complexity

Locate

Search conflicts

Insertion: $\Omega(n)$

Whole construction: $\Omega(n^2)$
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

Certified Delaunay triangles
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

Certified Delaunay triangles
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

Certified Delaunay triangles
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

Certified Delaunay triangles
Certified Delaunay edges
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

New point
Locate vertically
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

New point
Locate vertically
Create edge
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

Closing a triangle?
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

Next circle event
Close triangle
Delaunay Triangulation: sweep-line algorithm

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Circle events processed</th>
<th>Point events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Triangulation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>List of events ($x$ sorted)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>List of boundary edges (ccw sorted)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

20 - 1
# Delaunay Triangulation: sweep-line algorithm

## Complexity

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Circle events processed</th>
<th>Point events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>$2n$</td>
<td>$n$</td>
</tr>
<tr>
<td>Triangulation</td>
<td>create 2 triangles per event</td>
<td>create one edge per event</td>
</tr>
<tr>
<td>List of events ($x$ sorted)</td>
<td>$\leq 3$ deletions per event, $\leq 2$ insertions per event</td>
<td>$\leq 2$ deletions, $\leq 2$ insertions per event</td>
</tr>
<tr>
<td>List of boundary edges (ccw sorted)</td>
<td>replace 2 edges by 1 per event</td>
<td>locate, then insert 2 edges per event</td>
</tr>
</tbody>
</table>

20 - 2
## Delaunay Triangulation: sweep-line algorithm

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Circle events processed</th>
<th>Point events</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2(n)</td>
<td>(n)</td>
</tr>
<tr>
<td><strong>O(1)</strong> per operation</td>
<td>create 2 triangles per event</td>
<td>create one edge per event</td>
</tr>
<tr>
<td><strong>O(log n)</strong> per operation</td>
<td>(\leq 3 ) deletions per event</td>
<td>(\leq 2 ) deletions per event</td>
</tr>
<tr>
<td></td>
<td>(\leq 2 ) insertions per event</td>
<td>(\leq 2 ) insertions per event</td>
</tr>
<tr>
<td><strong>List of boundary edges</strong></td>
<td>replace 2 edges by 1 per event</td>
<td>locate, then insert 2 edges per event</td>
</tr>
</tbody>
</table>

### Notes
- List of events \(x\) sorted
- List of boundary edges \(ccw\) sorted
- Complexity: 20 - 3
Delaunay Triangulation: sweep-line algorithm

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Circle events processed</th>
<th>Point events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>2n</td>
<td>n</td>
</tr>
<tr>
<td>Triangulation</td>
<td>O(1) per operation</td>
<td></td>
</tr>
<tr>
<td>List of events (x sorted)</td>
<td>O(log n) per operation</td>
<td></td>
</tr>
<tr>
<td>List of boundary edges</td>
<td>O(log n) per operation</td>
<td></td>
</tr>
</tbody>
</table>

- Create 2 triangles per event
- Create one edge per event
- Replace 2 edges by 1 per event
- Locate, then insert 2 edges per event

- Complexity: O(n log n) processed

20 - 4
Delaunay Triangulation: predicates
Delaunay Triangulation: predicates

$x$ comparisons

21 - 2
Delaunay Triangulation: predicates

\[ x \text{ comparisons} \]

\[ y \text{ comparisons} \]
Delaunay Triangulation: predicates

$x$ comparisons

$y$ comparisons

more intricate than orientation and incircle
Delaunay Triangulation: divide & conquer (sketch)
Delaunay Triangulation: divide & conquer (sketch)
Delaunay Triangulation: divide & conquer (sketch)

Divide
Recurse
Delaunay Triangulation: divide & conquer (sketch)
Delaunay Triangulation: divide & conquer (sketch)
Delaunay Triangulation: divide & conquer (sketch)

Divide
Recurse
Conquer

$O(n \log n)$

balanced
linear time
easier conquer
Jump and walk

Use randomness hypotheses
Jump and walk

Use randomness hypotheses
Jump and walk

Use randomness hypotheses
Jump and walk

Use randomness hypotheses

Hopefully shorter walk

Designed for random points

\[ O\left(\sqrt[3]{n}\right) \text{ expected location time} \]
Jump and walk (no distribution hypothesis) Randomized
Jump and walk (no distribution hypothesis)

$$\mathbb{E} \left[ \# \text{ of } \bigcirc \text{ in } \bigcirc \right] = \frac{n}{k}$$
Jump and walk (no distribution hypothesis)

\[ \mathbb{E} [\text{# of in-circles}] = \frac{n}{k} \]

Walk length = \( O \left( \frac{n}{k} \right) \)

choose \( k = \frac{2}{\sqrt{n}} \)
Jump and walk (no distribution hypothesis)  

\[ \mathbb{E} \left[ \# \text{ of in}(\bullet) \right] = \frac{n}{k} \]

Walk length = \( O \left( \frac{n}{k} \right) \)

choose \( k = \sqrt{n} \)
Jump and walk (no distribution hypothesis)

\[ \mathbb{E} \left[ \text{# of } \bullet \text{ in } \bigcirc \right] = \frac{n}{k} \]

Walk length = \( O \left( \frac{n}{k} \right) \)

choose \( k = \sqrt{n} \)
Jump and walk (no distribution hypothesis) Randomized Delaunay hierarchy

\[ \mathbb{E} [\# \text{ of } \bullet \text{ in } \bigcirc] = \frac{n}{k} \]

\[ \frac{n}{k_1} + \frac{k_1}{k_2} \]

Walk length = \( O \left( \frac{n}{k} \right) \)

choose \( k = \sqrt{n} \)
Jump and walk (no distribution hypothesis) Randomized

$\mathbb{E} \left[ \text{# of } \cdot \text{ in } \bigcirc \right] = \frac{n}{k}$

Walk length $= O \left( \frac{n}{k} \right)$

$\frac{n}{k_1} + \frac{k_1}{k_2} + \frac{k_2}{k_3} + \ldots$

choose $k = 2\sqrt{n}$
Jump and walk (no distribution hypothesis)  
\[ \mathbb{E} [\text{# of in } \bigcirc] = \frac{n}{k} \]

Walk length = \( O \left( \frac{n}{k} \right) \)

choose \( k = 2\sqrt{n} \)

\[ \frac{n}{k_1} + \frac{k_1}{k_2} + \frac{k_2}{k_3} + \ldots \]

choose \( \frac{k_i}{k_{i+1}} = \alpha \)

Randomized Delaunay hierarchy
Jump and walk (no distribution hypothesis) \[ \mathbb{E} \left[ \text{\# of \textbullet\ in } \mathcal{C} \right] = \frac{n}{k} \]

Walk length = \( O \left( \frac{n}{k} \right) \)

Randomized Delaunay hierarchy
\[ \frac{n}{k_1} + \frac{k_1}{k_2} + \frac{k_2}{k_3} + \ldots \]

choose \( \frac{k_i}{k_{i+1}} = \alpha \)

choose \( k = \sqrt{n} \)

point location in \( O(\alpha \log_\alpha n) \)
Jump and walk (no distribution hypothesis) \[ E[\# \text{ of } \bullet \text{ in } \odot] = \frac{n}{k} \]

Walk length \( = O\left(\frac{n}{k}\right)\)

Choose \( k = \sqrt{n} \)

Walk length \( = O\left(\frac{n}{k}\right) = O\left(\frac{n}{\sqrt{n}}\right) = O\left(\sqrt{n}\right) \)

Delaunay hierarchy

\[ \frac{n}{k_1} + \frac{k_1}{k_2} + \frac{k_2}{k_3} + \ldots \]

Choose \( \frac{k_i}{k_{i+1}} = \alpha \)

Point location in \( O\left(\alpha \log \alpha n\right) \)

Point location in \( O\left(\sqrt{\alpha} \log \alpha n\right) \)
Technical detail

Walk length $= O\left(\# \text{ of } \bullet \text{ in } \bigcirc \right) = O\left(\frac{n}{k}\right)$
Walk length $= \mathcal{O}\left(\text{# of \ red \ points \ in } \bigcirc \right) = \mathcal{O}\left(\frac{n}{k}\right)$

random point

not a random point
Technical detail

Walk length $= O\left(\# \text{ of } \bullet \text{ in } \bigcirc \right) = O\left(\frac{n}{k}\right)$

$$\mathbb{E}[d^\circ \bullet] = \frac{1}{n} \sum_q d^\circ NN(q) = \frac{1}{n} \sum_q \sum_{v=NN(q)} d^\circ v$$

$$= \frac{1}{n} \sum_v \sum_{q;v=NN(q)} d^\circ v \leq \frac{1}{n} \sum_v 6d^\circ v \leq 36$$
Walk length $= O\left(\# \text{ of } \bullet \text{ in } \bigcirc \right) = O\left(\frac{n}{k}\right)$

$$\mathbb{E}[d^\circ \bullet] = \frac{1}{n} \sum_q d^\circ NN(q) = \frac{1}{n} \sum_q \sum_{v=NN(q)} d^\circ v$$

$$= \frac{1}{n} \sum_v \sum_{q;v=NN(q)} d^\circ v \leq \frac{1}{n} \sum_v 6d^\circ v \leq 36$$
Randomization

Drawbacks of random order

- non locality of memory access
- data structure for point location

→ Hilbert sort
27 - 1
Drawbacks of random order

- non locality of memory access
- data structure for point location

→ Hilbert sort

Walk should be fast

Last point is not at all a random point

→ No control of degree of last point
Biased Random Insertion Order (BRIO)
Biased Random Insertion Order (BRIO)
Biased Random Insertion Order (BRIO)
Biased Random Insertion Order (BRIO)
Biased Random Insertion Order (BRIO)
Biased Random Insertion Order (BRIO)
Algorithm

Recipe to go from the input to the output

Formalized description in some language

May use data structure

Proof of correctness

Complexity analysis
Algorithm

Recipe to go from the input to the output

Implementation of an algorithm

Formalized description in some language

Translation in a programming language (C++)

May use data structure

May use software library

Proof of correctness

Debugging

Complexity analysis

Running time

Program
Algorithm may be difficult to transform into Program

Recipe to go from the input to the output

Implementation of an algorithm

Formalized description in some language

Translation in a programming language (C++)

May use data structure

May use software library

Proof of correctness

Debugging

Complexity analysis

Running time

30 - 3
The end