Beating the Spread: Time-Optimal Point Meshing

Don Sheehy
Carnegie Mellon
(soon: INRIA)

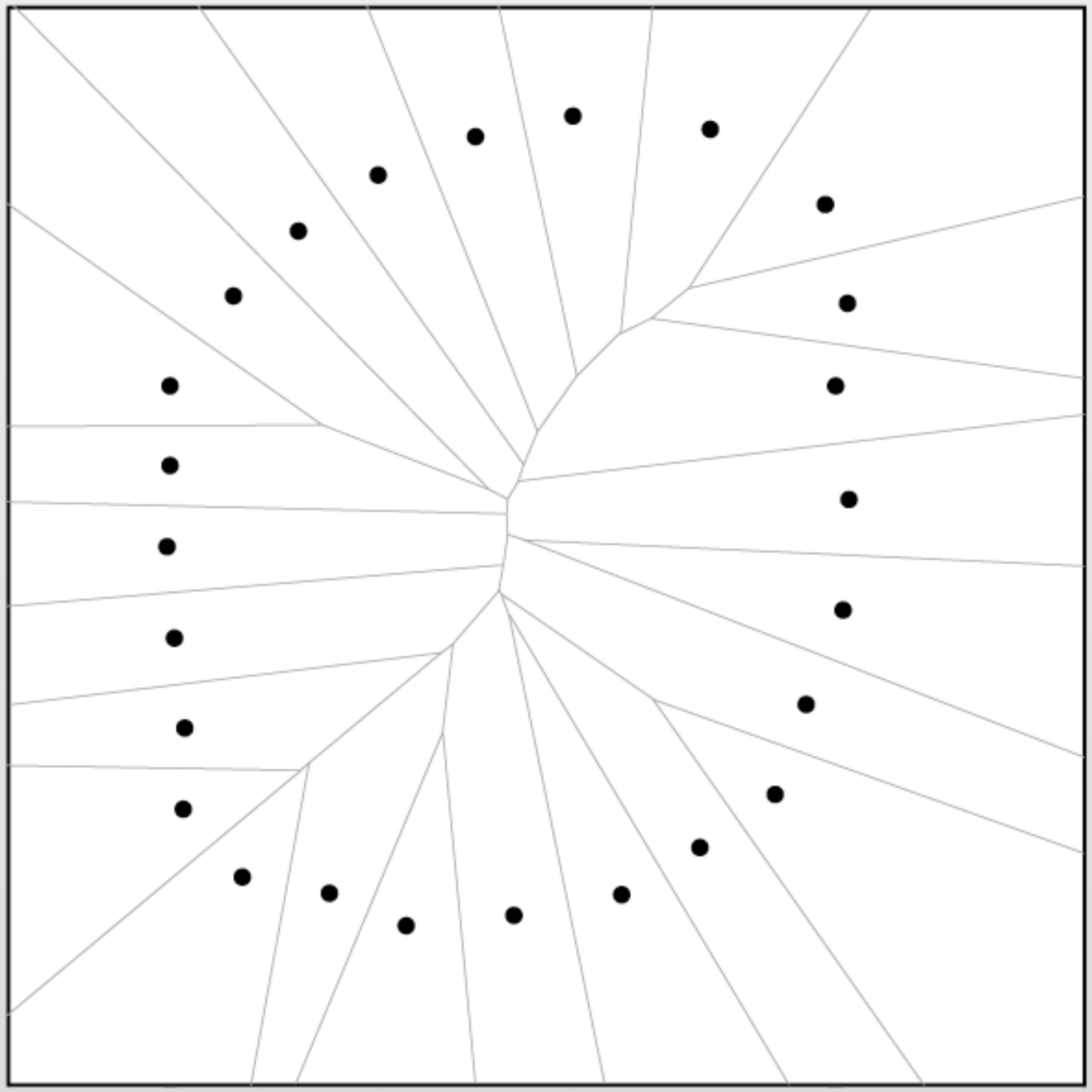
with Gary Miller and Todd Phillips at CMU

Meshing Points

Input: $P \subset \mathbb{R}^d$

Output: $M \supset P$ with a “nice” Voronoi diagram

$n = |P|, m = |M|$

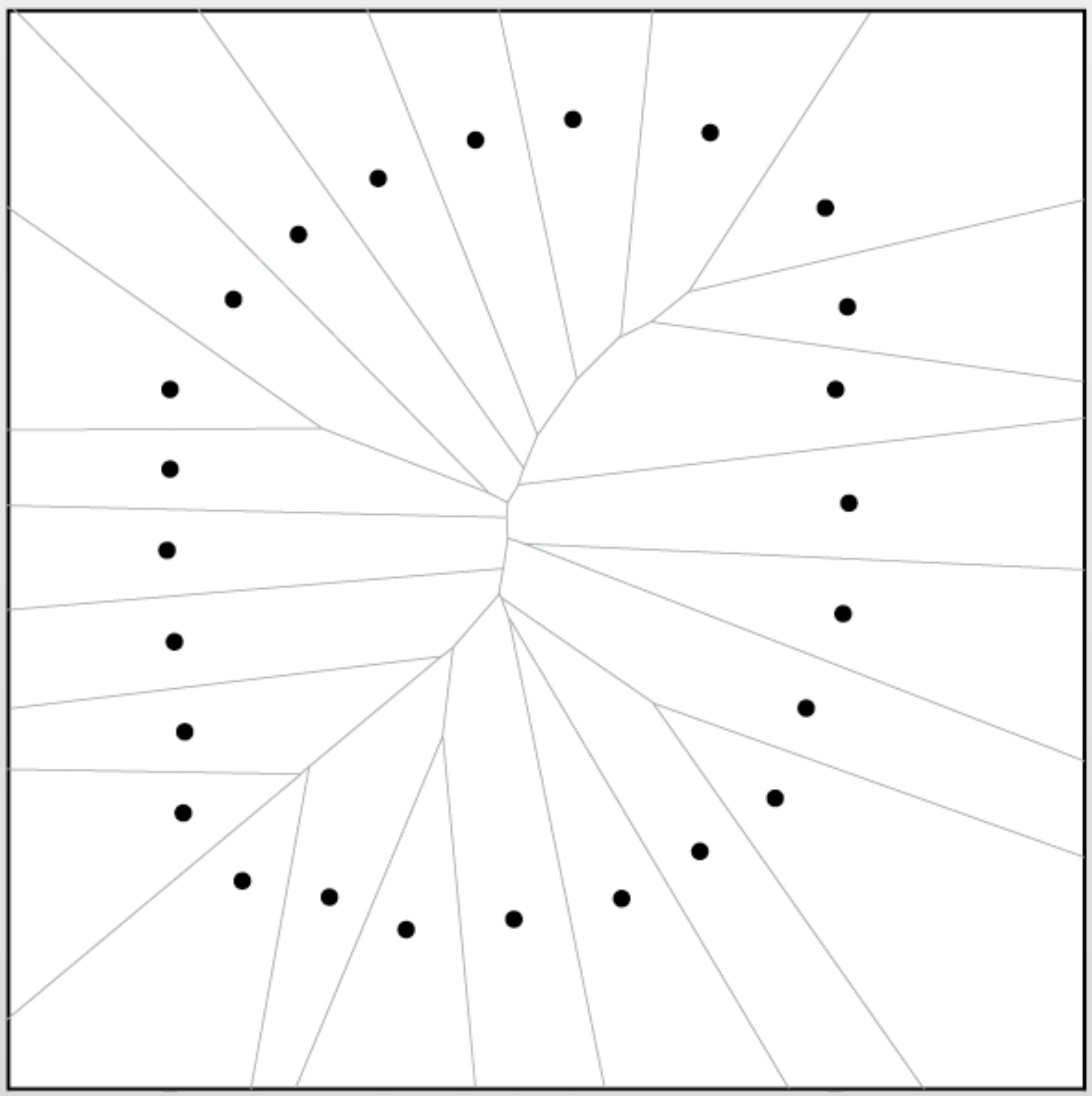


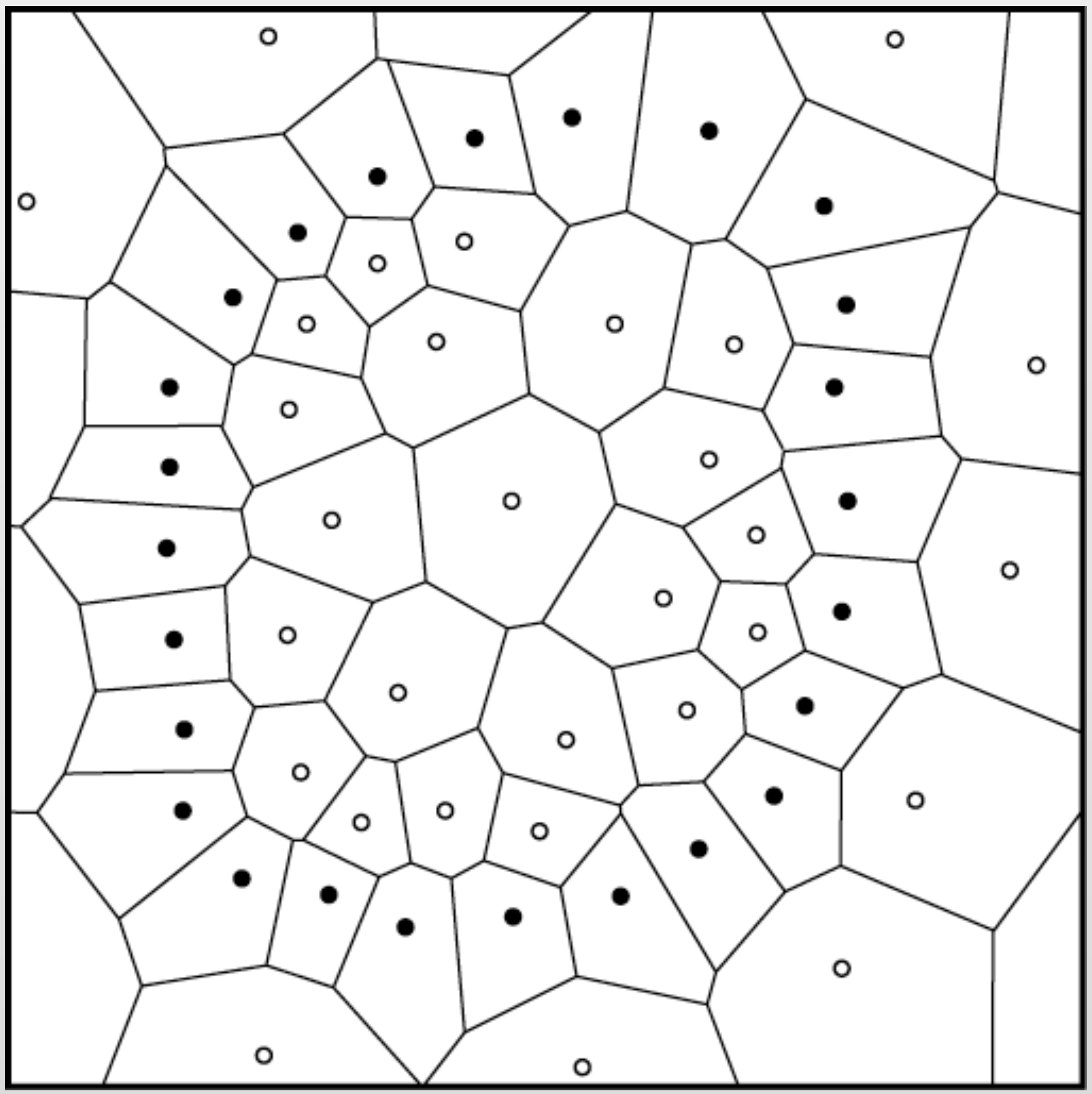
Meshing Points

Input: $P \subset \mathbb{R}^d$

Output: $M \supset P$ with a "nice" Voronoi diagram

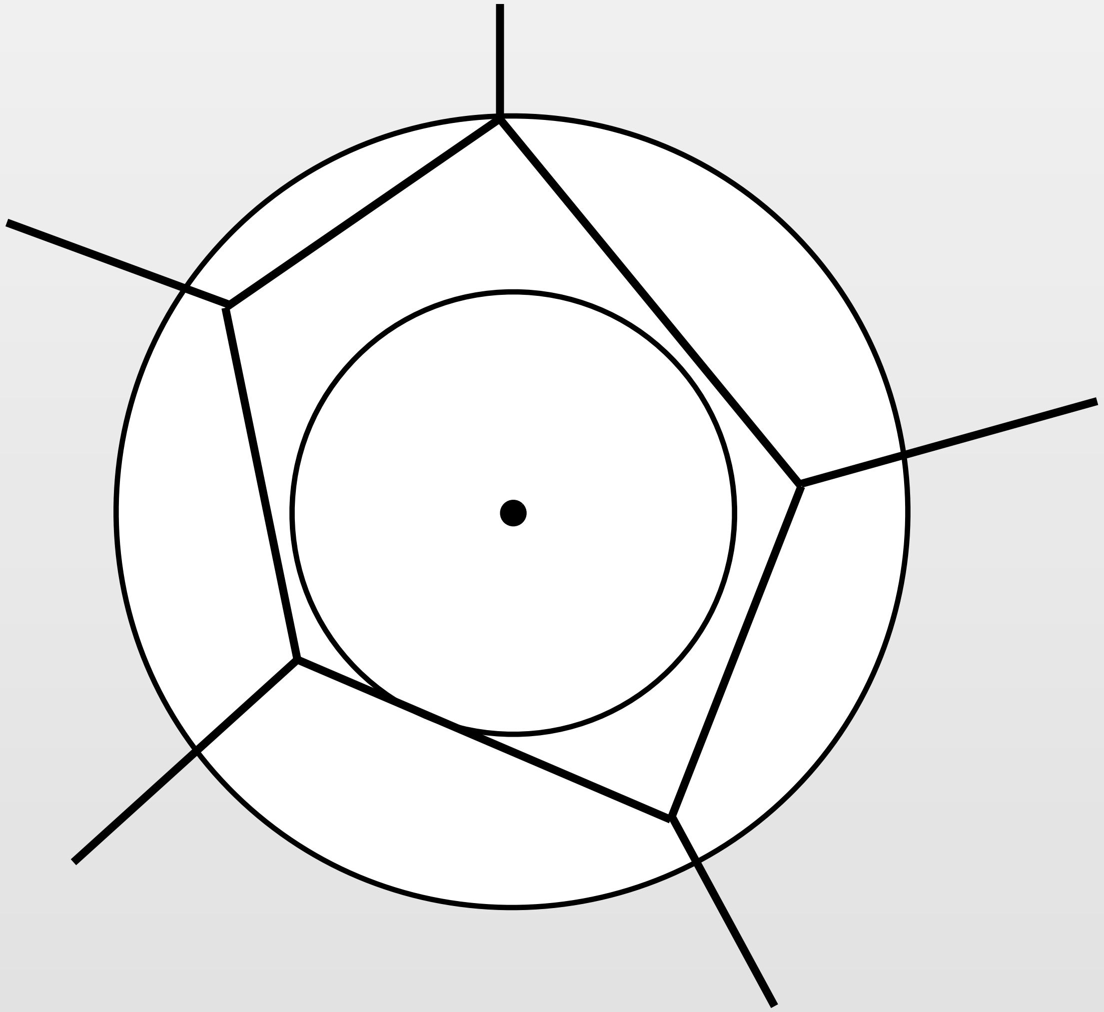
$n = |P|, m = |M|$





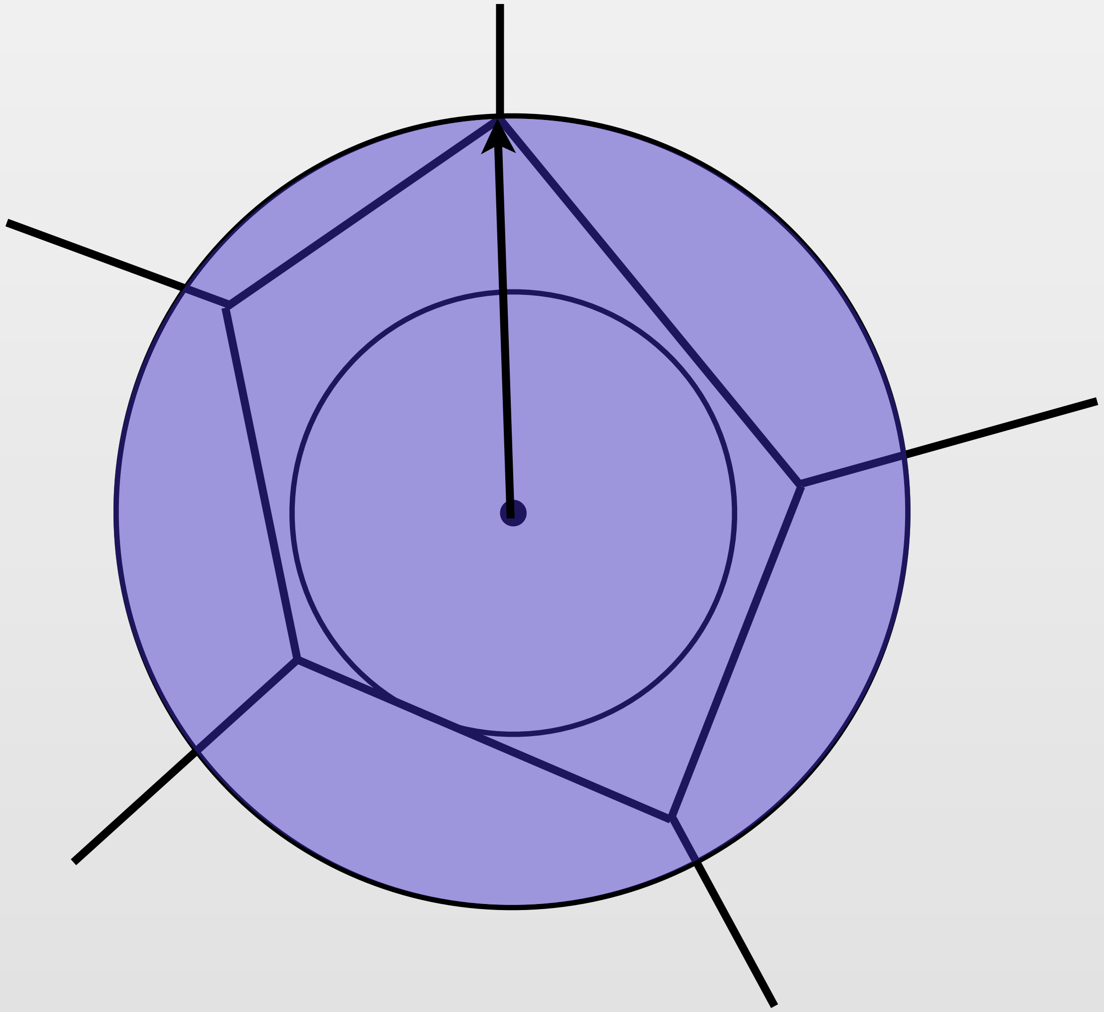
Quality Meshes have cells with bounded *aspect ratio*.

$$\text{aspect ratio} = \frac{\mathrm{R}}{\mathrm{r}} \leq \tau$$



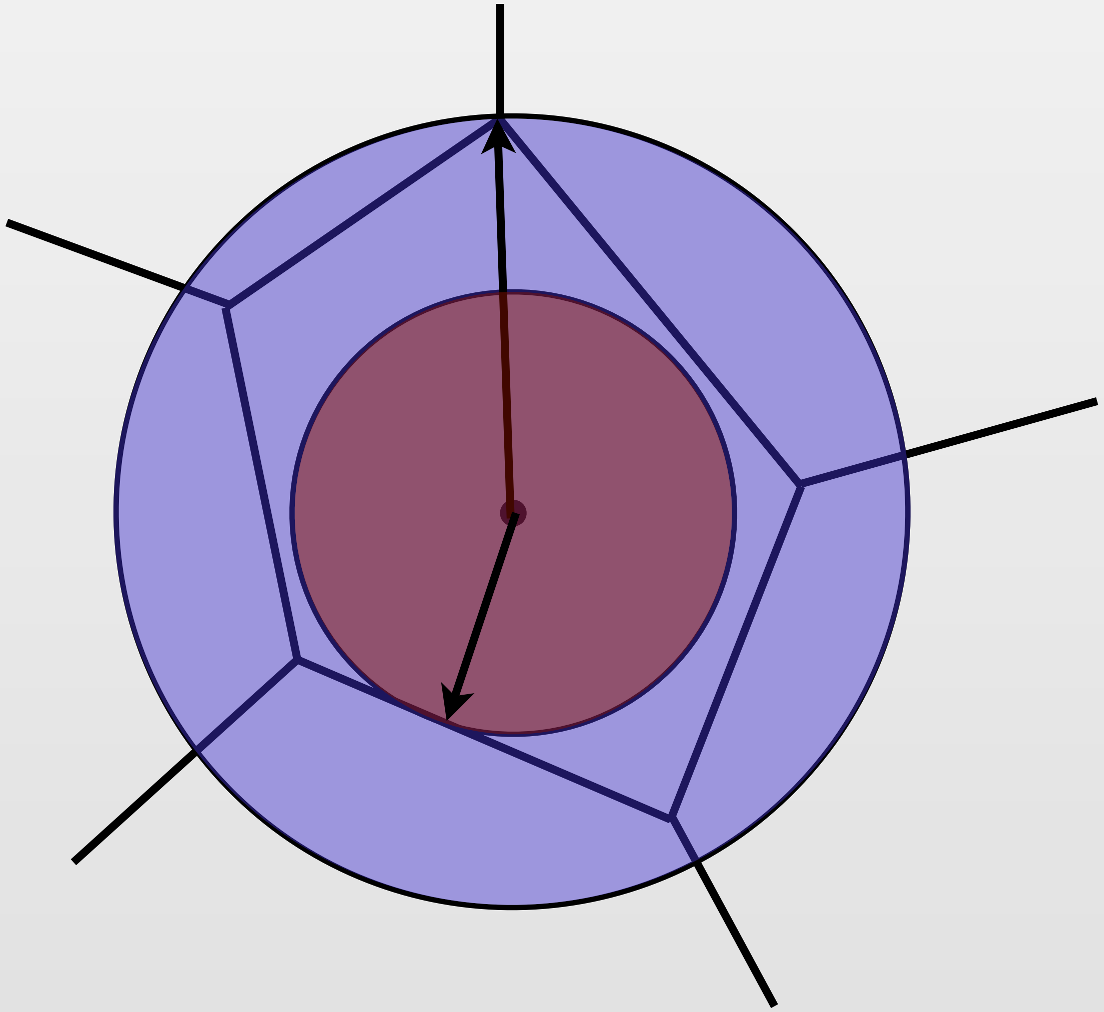
Quality Meshes have cells with bounded *aspect ratio*.

$$\text{aspect ratio} = \frac{R}{r} \leq \tau$$



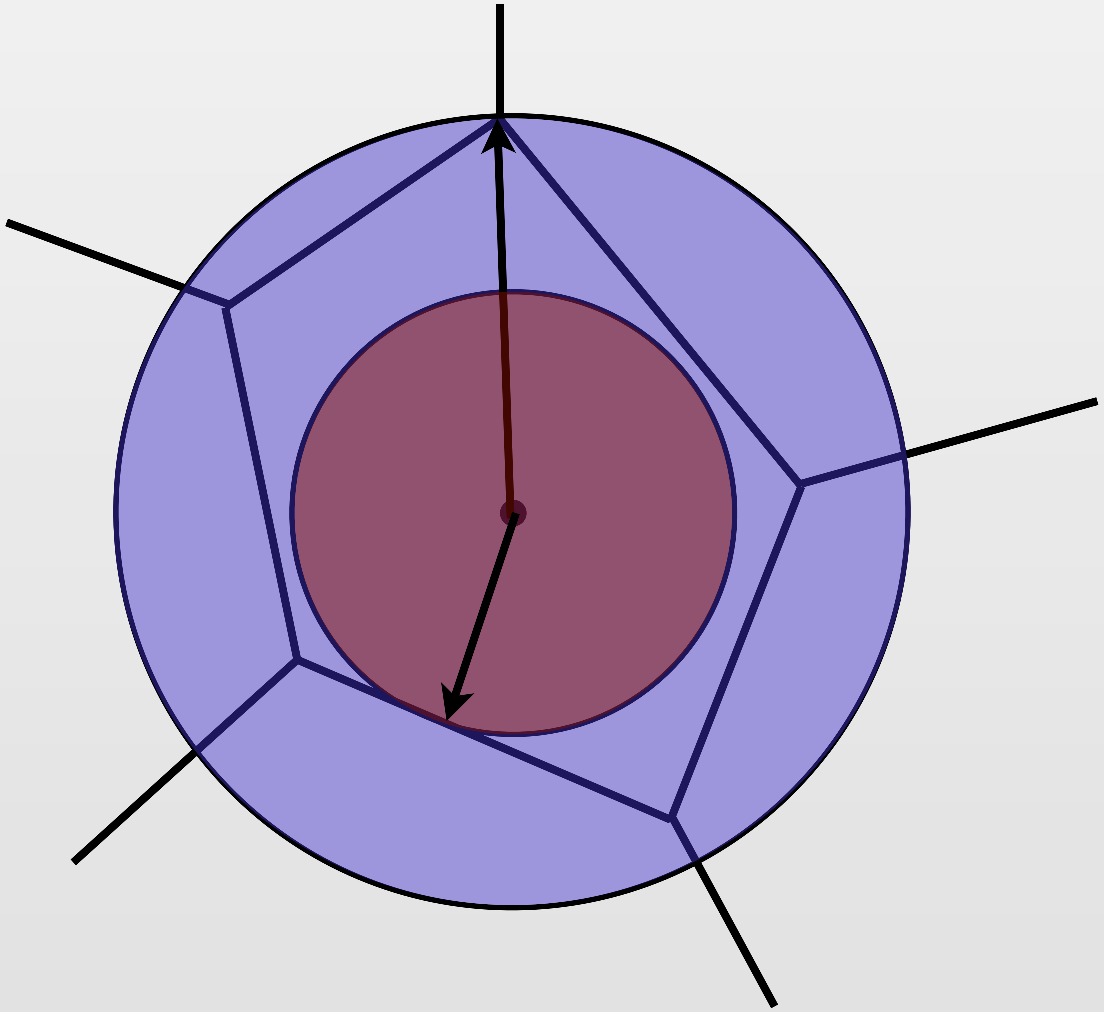
Quality Meshes have cells with bounded *aspect ratio*.

$$\text{aspect ratio} = \frac{\mathrm{R}}{\mathrm{r}} \leq \tau$$



Quality Meshes have cells with bounded *aspect ratio*.

$$\text{aspect ratio} = \frac{\mathbf{R}}{\mathbf{r}} \leq \tau$$



$$\boxed{\tau \geq 2 + \varepsilon}$$

Prior Work

Prior Work

Delaunay Refinement:

Prior Work

Delaunay Refinement:

Chew '89 .... 2D

Prior Work

Delaunay Refinement:

Chew ‘89 .... 2D
Ruppert ’95 .... Optimality in 2D

Prior Work

Delaunay Refinement:

Chew ‘89 .... 2D
Ruppert ’95 .... Optimality in 2D
Shewchuck ‘98 .... Ruppert in 3D

Prior Work

Delaunay Refinement:

- Chew '89 .... 2D
- Ruppert '95 .... Optimality in 2D
- Shewchuck '98 .... Ruppert in 3D
- Hudson - Miller - Phillips '06 .... SVR $O(n \log \Delta + m)$

Prior Work

Delaunay Refinement:

Chew ‘89	2D
Ruppert ’95	Optimality in 2D
Shewchuck ‘98	Ruppert in 3D
Hudson - Miller - Phillips ‘06	SVR $O(n \log \Delta + m)$

Quadtree Methods:

Prior Work

Delaunay Refinement:

Chew ‘89 .... 2D
Ruppert ’95 .... Optimality in 2D
Shewchuck ‘98 .... Ruppert in 3D
Hudson - Miller - Phillips ‘06 .... SVR $O(n \log \Delta + m)$

Quadtree Methods:

Bern - Eppstein - Gilbert ’94 .... QT meshing

Prior Work

Delaunay Refinement:

Chew '89 .... 2D
Ruppert '95 .... Optimality in 2D
Shewchuck '98 .... Ruppert in 3D
Hudson - Miller - Phillips '06 .... SVR $O(n \log \Delta + m)$

Quadtree Methods:

Bern - Eppstein - Gilbert '94 .... QT meshing
Bern - Eppstein - Teng '99 .... QTs in Parallel

Prior Work

Delaunay Refinement:

Chew ‘89 .... 2D
Ruppert ’95 .... Optimality in 2D
Shewchuck ‘98 .... Ruppert in 3D
Hudson - Miller - Phillips ‘06 .... SVR $O(n \log \Delta + m)$

Quadtree Methods:

Bern - Eppstein - Gilbert ’94 .... QT meshing
Bern - Eppstein - Teng ‘99 .... QTs in Parallel

Hybrid Methods:

Prior Work

Delaunay Refinement:

Chew '89 .... 2D
Ruppert '95 .... Optimality in 2D
Shewchuck '98 .... Ruppert in 3D
Hudson - Miller - Phillips '06 .... SVR $O(n \log \Delta + m)$

Quadtree Methods:

Bern - Eppstein - Gilbert '94 .... QT meshing
Bern - Eppstein - Teng '99 .... QTs in Parallel

Hybrid Methods:

Har-Peled - Ungor '05 .... $O(n \log n + m)$ in 2D

Prior Work

Delaunay Refinement:

Chew '89 .... 2D
Ruppert '95 .... Optimality in 2D
Shewchuck '98 .... Ruppert in 3D
Hudson - Miller - Phillips '06 .... SVR $O(n \log \Delta + m)$

Quadtree Methods:

Bern - Eppstein - Gilbert '94 .... QT meshing
Bern - Eppstein - Teng '99 .... QTs in Parallel

Hybrid Methods:

Har-Peled - Ungor '05 .... $O(n \log n + m)$ in 2D

Prior Work

Delaunay Refinement:

Chew ‘89 .... 2D
Ruppert ’95 .... Optimality in 2D
Shewchuck ‘98 .... Ruppert in 3D
Hudson - Miller - Phillips ‘06 .... SVR $O(n \log \Delta + m)$

Quadtree Methods:

Bern - Eppstein - Gilbert ’94 .... QT meshing
Bern - Eppstein - Teng ‘99 .... QTs in Parallel

Hybrid Methods:

Har-Peled - Ungor ‘05 .... $O(n \log n + m)$ in 2D

Many Others

Prior Work

Delaunay Refinement:

Chew '89 .... 2D
Ruppert '95 .... Optimality in 2D
Shewchuck '98 .... Ruppert in 3D
Hudson - Miller - Phillips '06 .... SVR $O(n \log \Delta + m)$

Quadtree Methods:

Bern - Eppstein - Gilbert '94 .... QT meshing
Bern - Eppstein - Teng '99 .... QTs in Parallel

Hybrid Methods:

Har-Peled - Ungor '05 .... $O(n \log n + m)$ in 2D

Many Others

Our Result

$O(n \log n + m)$

for point sets in d dimensions

Prior Work

Delaunay Refinement:

Chew ‘89 .... 2D
Ruppert ’95 .... Optimality in 2D
Shewchuck ‘98 .... Ruppert in 3D
Hudson - Miller - Phillips ‘06 .... SVR $O(n \log \Delta + m)$

Quadtree Methods:

Bern - Eppstein - Gilbert ’94 .... QT meshing
Bern - Eppstein - Teng ‘99 .... QTs in Parallel

Hybrid Methods:

Har-Peled - Ungor ‘05 .... $O(n \log n + m)$ in 2D

Many Others

Our Result

$O(n \log n + m)$

for point sets in d dimensions

Hides dimension terms

Beating the spread.

Beating the spread.

Let $s = |x - y|$ where $(x, y) \in \binom{P}{2}$ is the closest pair.

Beating the spread.

Let $s = |x - y|$ where $(x, y) \in \binom{P}{2}$ is the closest pair.

The **spread** of P is the ratio $\Delta = \frac{\text{diameter}(P)}{s}$.

Beating the spread.

Let $s = |x - y|$ where $(x, y) \in \binom{P}{2}$ is the closest pair.

The **spread** of P is the ratio $\Delta = \frac{\text{diameter}(P)}{s}$.

$$O(n \log \Delta + m)$$

Beating the spread.

Let $s = |x - y|$ where $(x, y) \in \binom{P}{2}$ is the closest pair.

The **spread** of P is the ratio $\Delta = \frac{\text{diameter}(P)}{s}$.

$$O(n \log \Delta + m)$$

Point location

Beating the spread.

Let $s = |x - y|$ where $(x, y) \in \binom{P}{2}$ is the closest pair.

The **spread** of P is the ratio $\Delta = \frac{\text{diameter}(P)}{s}$.

$$O(n \log \Delta + m)$$

Point location

Output sensitive

Beating the spread.

Let $s = |x - y|$ where $(x, y) \in \binom{P}{2}$ is the closest pair.

The **spread** of P is the ratio $\Delta = \frac{\text{diameter}(P)}{s}$.

$$O(n \log \Delta + m)$$

Point location

Output sensitive

For some point sets, $m = \Omega(n \log \Delta)$

Complexity: How big is the mesh?

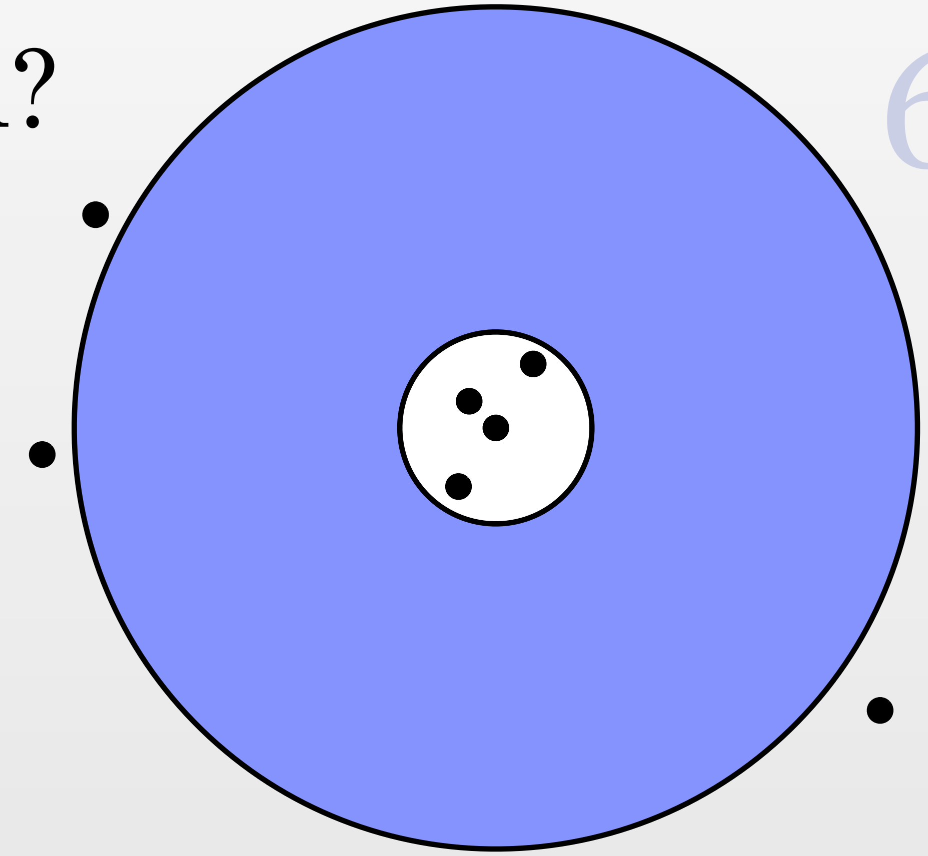
$$m = \int_\Omega \frac{1}{\mathrm{lfs}_P(x)^d}$$

Complexity: How big is the mesh?

How many Steiner Points?

$$m = \int_{\Omega} \frac{1}{\mathrm{lfs}_P(x)^d}$$

Complexity: How big is the mesh?

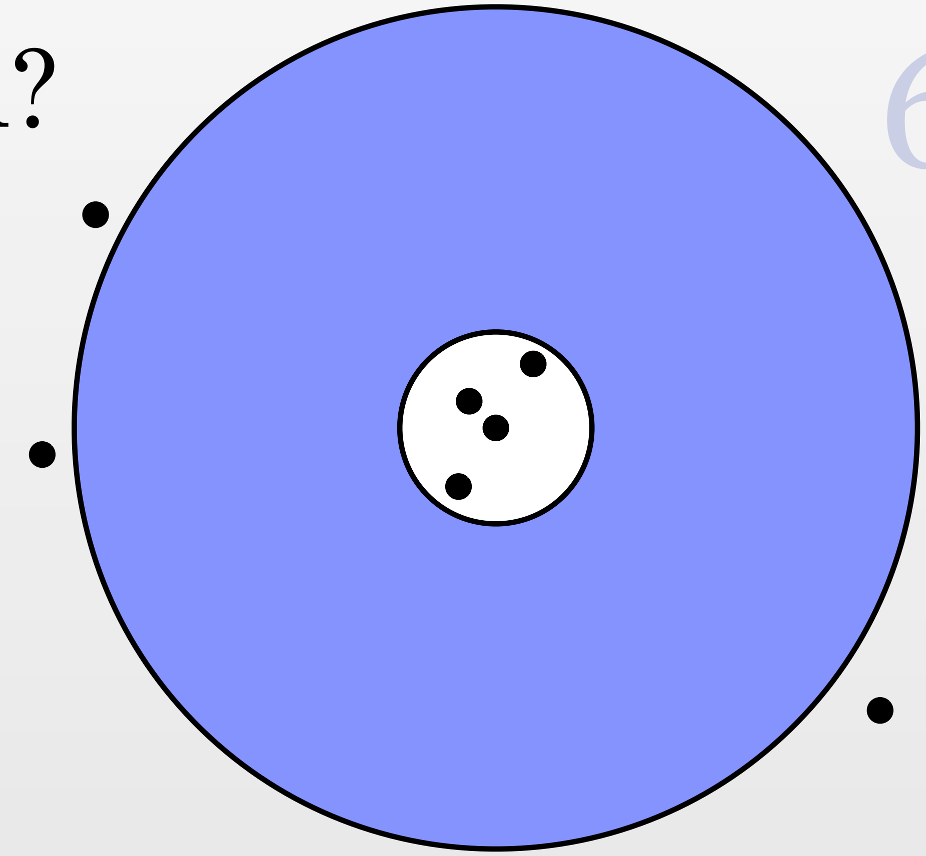


How many Steiner Points?

$$m = \int_\Omega \frac{1}{\mathrm{lfs}_P(x)^d}$$

m = *O(n)* as long as there are no big empty annuli with 2 or more points inside [MPS08].

Complexity: How big is the mesh?



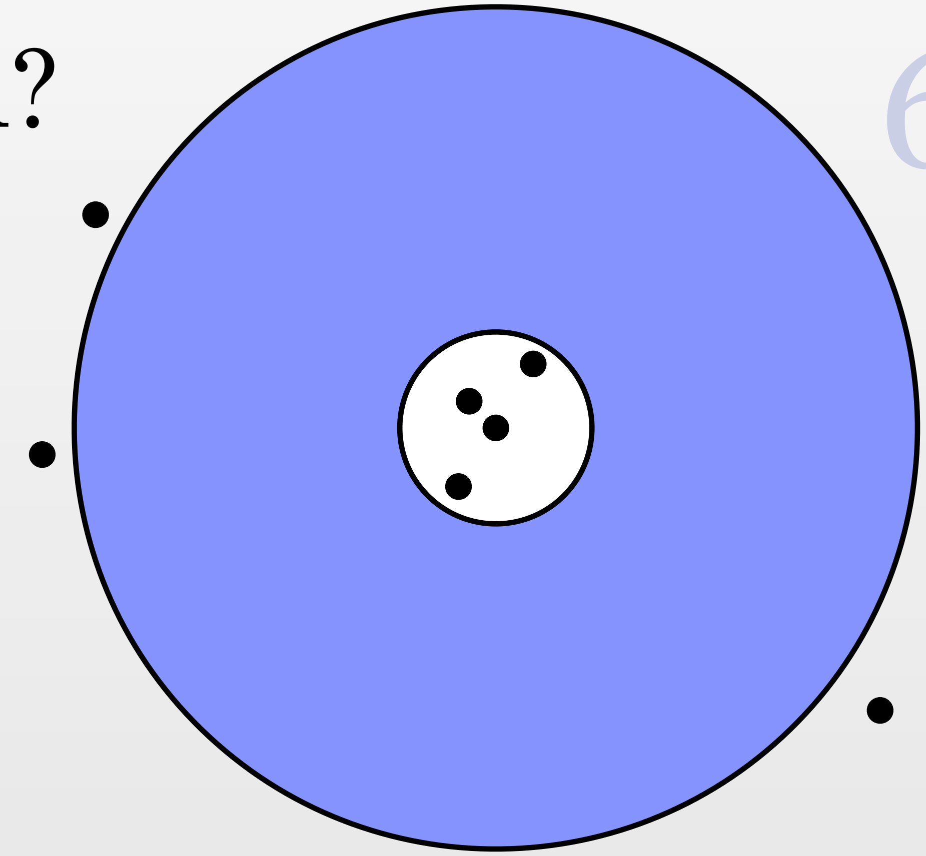
How many Steiner Points?

$$m = \int_\Omega \frac{1}{\mathrm{lfs}_P(x)^d}$$

m = *O(n)* as long as there are no big empty annuli with 2 or more points inside [MPS08].

How many simplices?

Complexity: How big is the mesh?



How many Steiner Points?

$$m = \int_{\Omega} \frac{1}{\mathrm{lfs}_P(x)^d}$$

$m = O(n)$ as long as there are no big empty annuli with 2 or more points inside [MPS08].

How many simplices?

Only $O(m)$ simplices.
Compare to $m^{\lceil d/2 \rceil}$ for general Delaunay triangulations.
Constants depend on aspect ratio.

Complexity: How hard is it to compute a mesh?

Complexity: How hard is it to compute a mesh?

$$O(n \log n + m)$$

Complexity: How hard is it to compute a mesh?

$$O(n \log n + m)$$

Point Location

Complexity: How hard is it to compute a mesh?

$$O(n \log n + m)$$

Point Location

Output Sensitive

Complexity: How hard is it to compute a mesh?

$$O(n \log n + m)$$

Point Location

Output Sensitive

This is optimal in the comparison model.

Complexity: How hard is it to compute a mesh?

$$O(n \log n + m)$$

Point Location

Output Sensitive

This is optimal in the comparison model.

$$O(n \log n)$$

Complexity: How hard is it to compute a mesh?

$$O(n \log n + m)$$

Point Location

Output Sensitive

This is optimal in the comparison model.

$$O(n \log n)$$

Compute a *hierarchical quality* mesh.

Complexity: How hard is it to compute a mesh?

$$O(n \log n + m)$$

Point Location

Output Sensitive

This is optimal in the comparison model.

$$O(n \log n)$$

Compute a *hierarchical quality* mesh.

Key fact: Size is O(n)

Complexity: How hard is it to compute a mesh?

$$O(n \log n + m)$$

Point Location

Output Sensitive

This is optimal in the comparison model.

$$O(n \log n) \; + \; O(m)$$

Compute a *hierarchical quality* mesh.

Key fact: Size is O(n)

Complexity: How hard is it to compute a mesh?

$$O(n \log n + m)$$

Point Location

Output Sensitive

This is optimal in the comparison model.

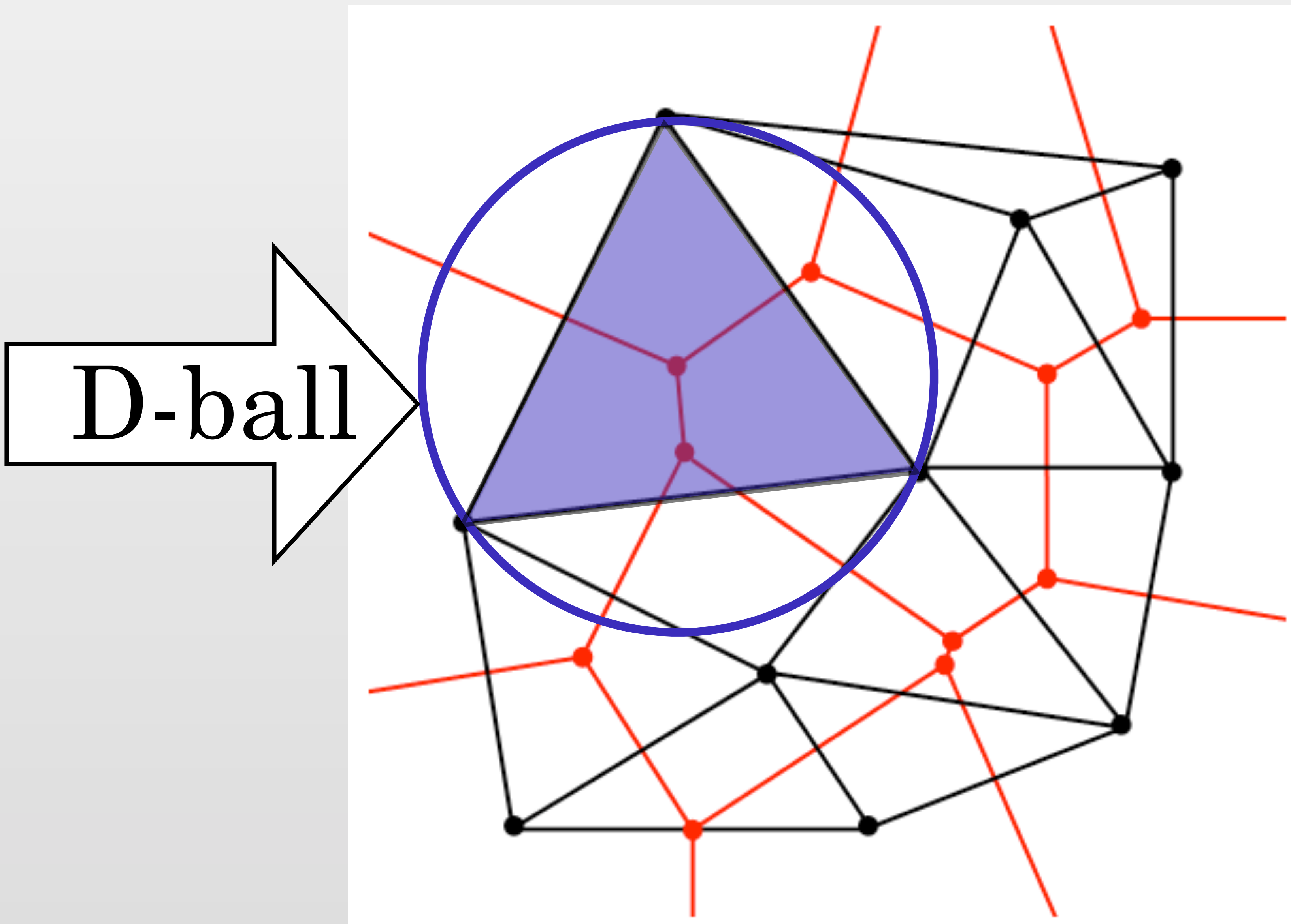
$$O(n \log n) \ + \ O(m)$$

Compute a *hierarchical quality* mesh.

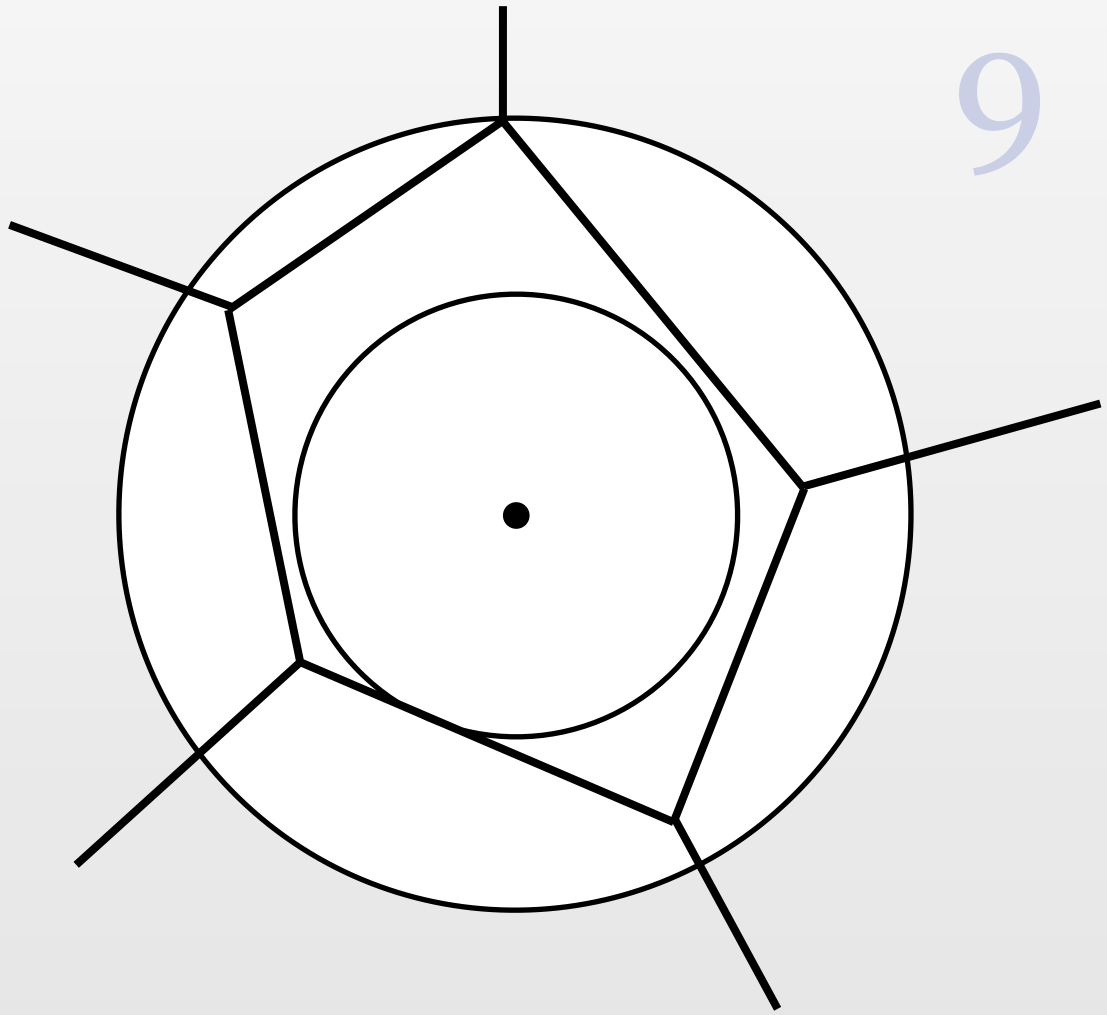
Finishing post-process.
(easy)

Key fact: Size is O(n)

The Delaunay Triangulation is the dual of the Voronoi Diagram.

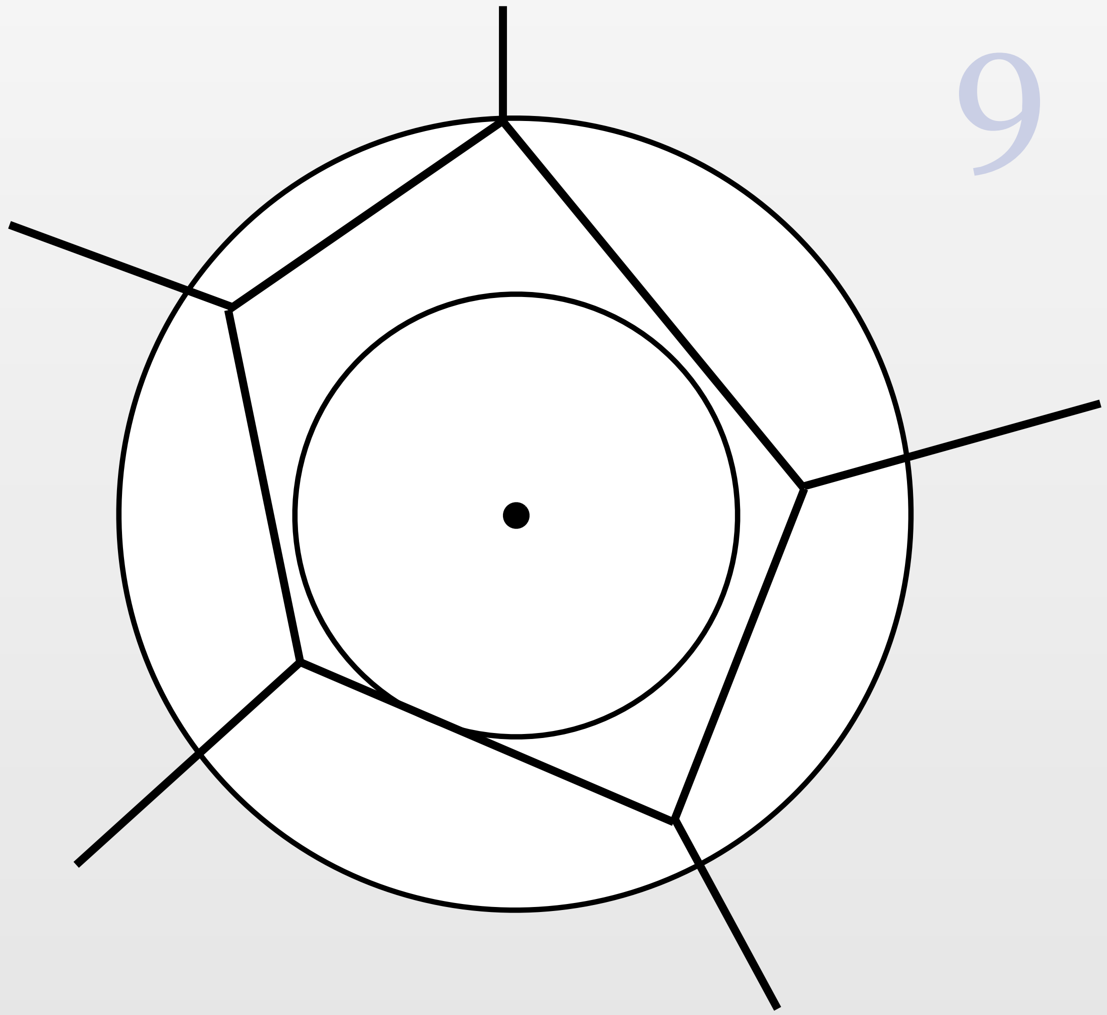


Quality Meshes have several nice properties.

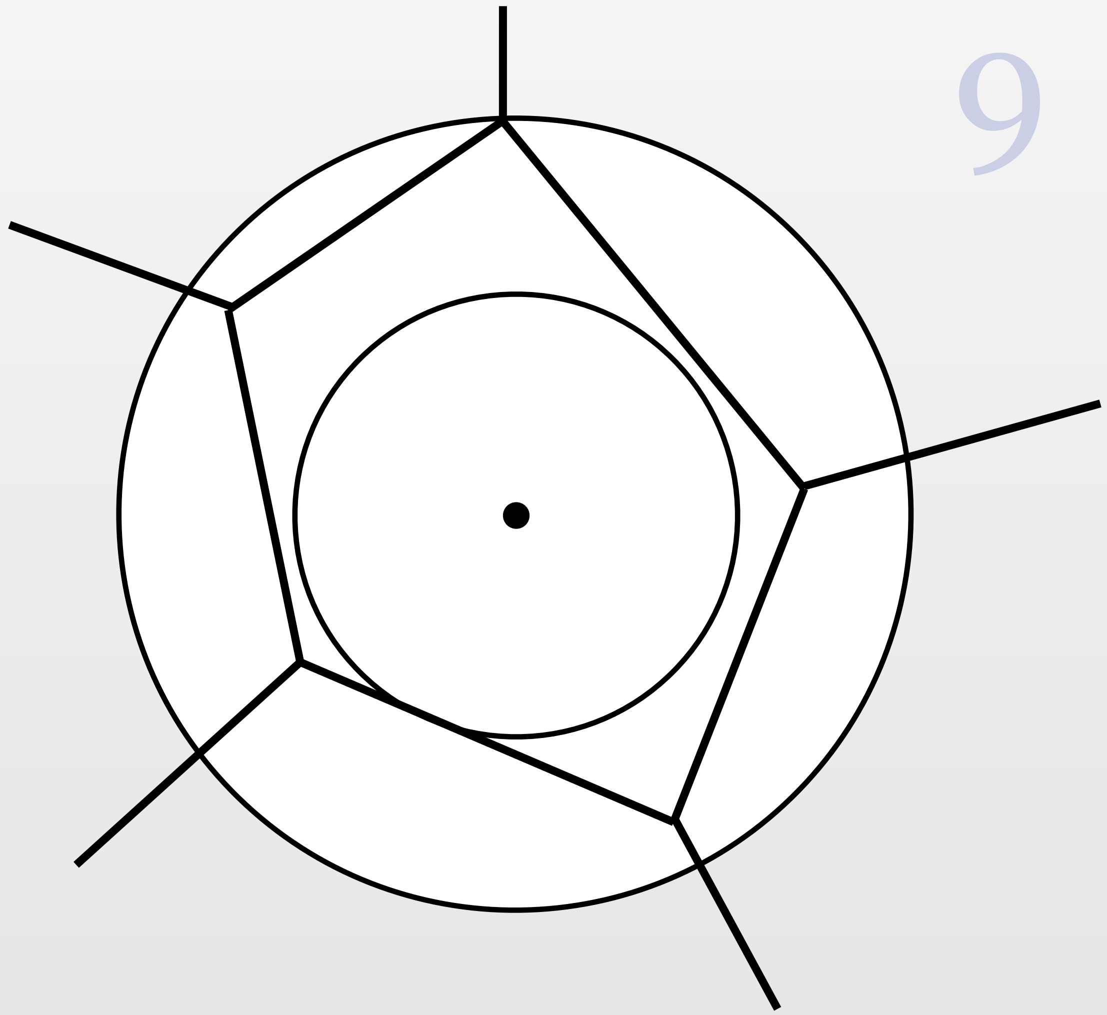


Quality Meshes have several nice properties.

Delaunay balls have constant ply



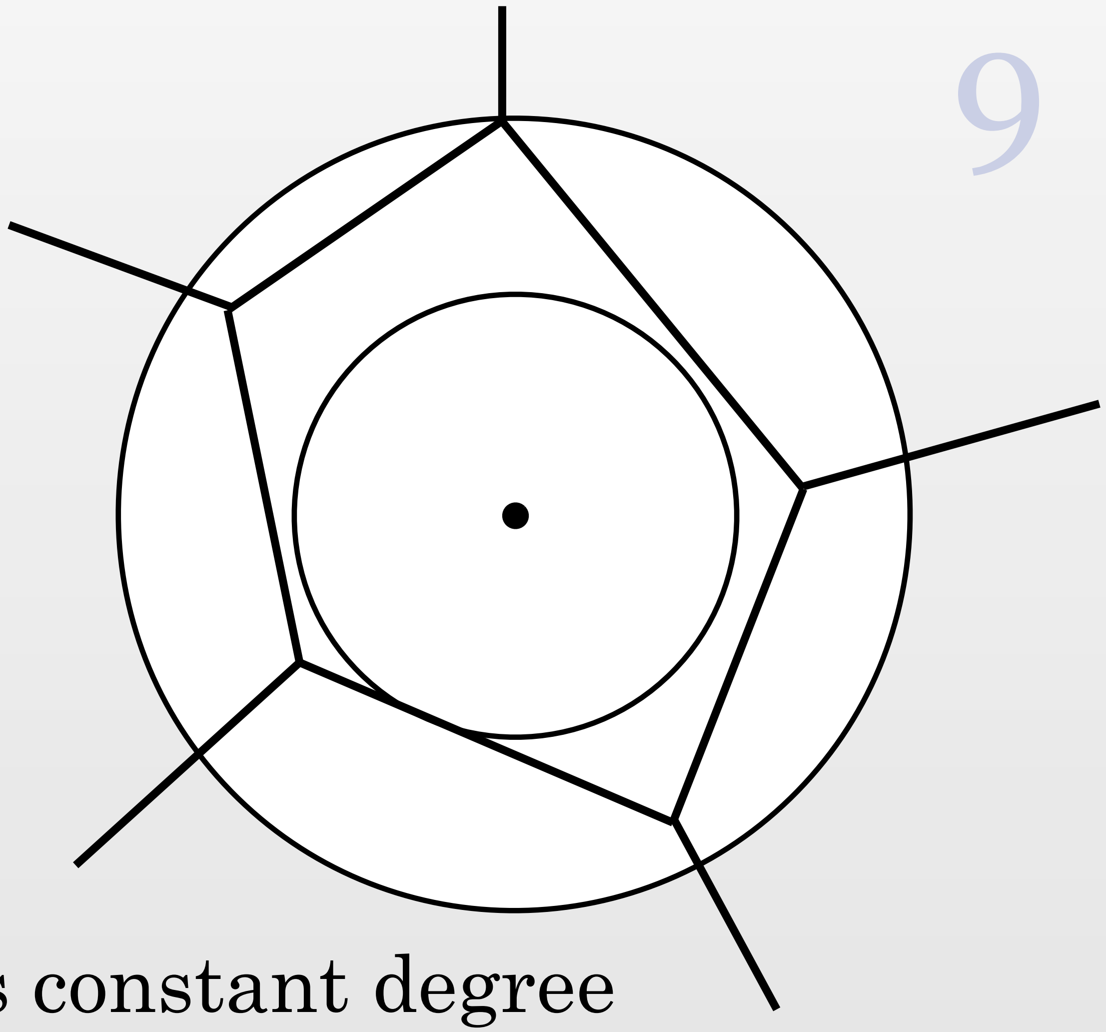
Quality Meshes have several nice properties.



Delaunay balls have constant ply

Total number of faces is O(m)

Quality Meshes have several nice properties.

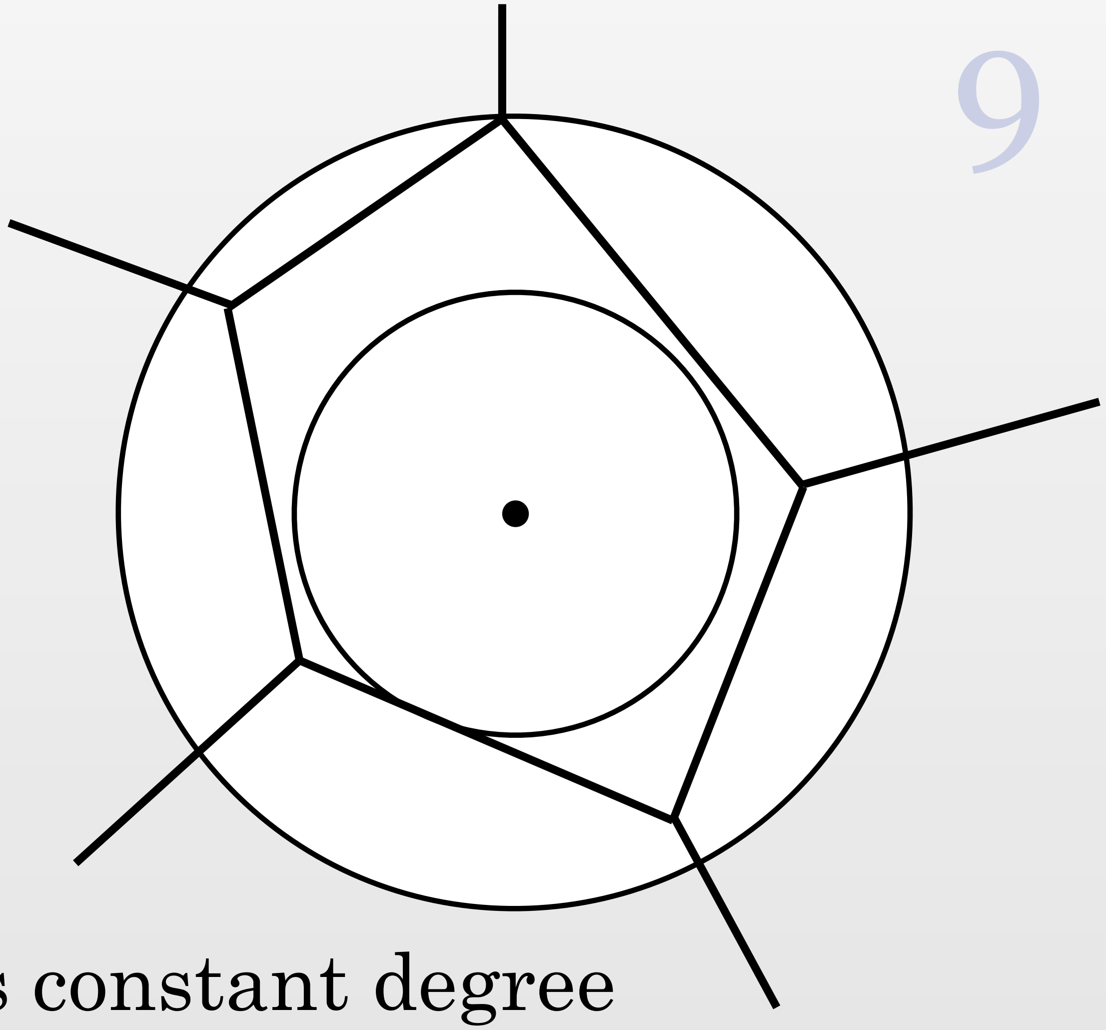


Delaunay balls have constant ply

Total number of faces is O(m)

Intersection graph of Delaunay balls has constant degree

Quality Meshes have several nice properties.



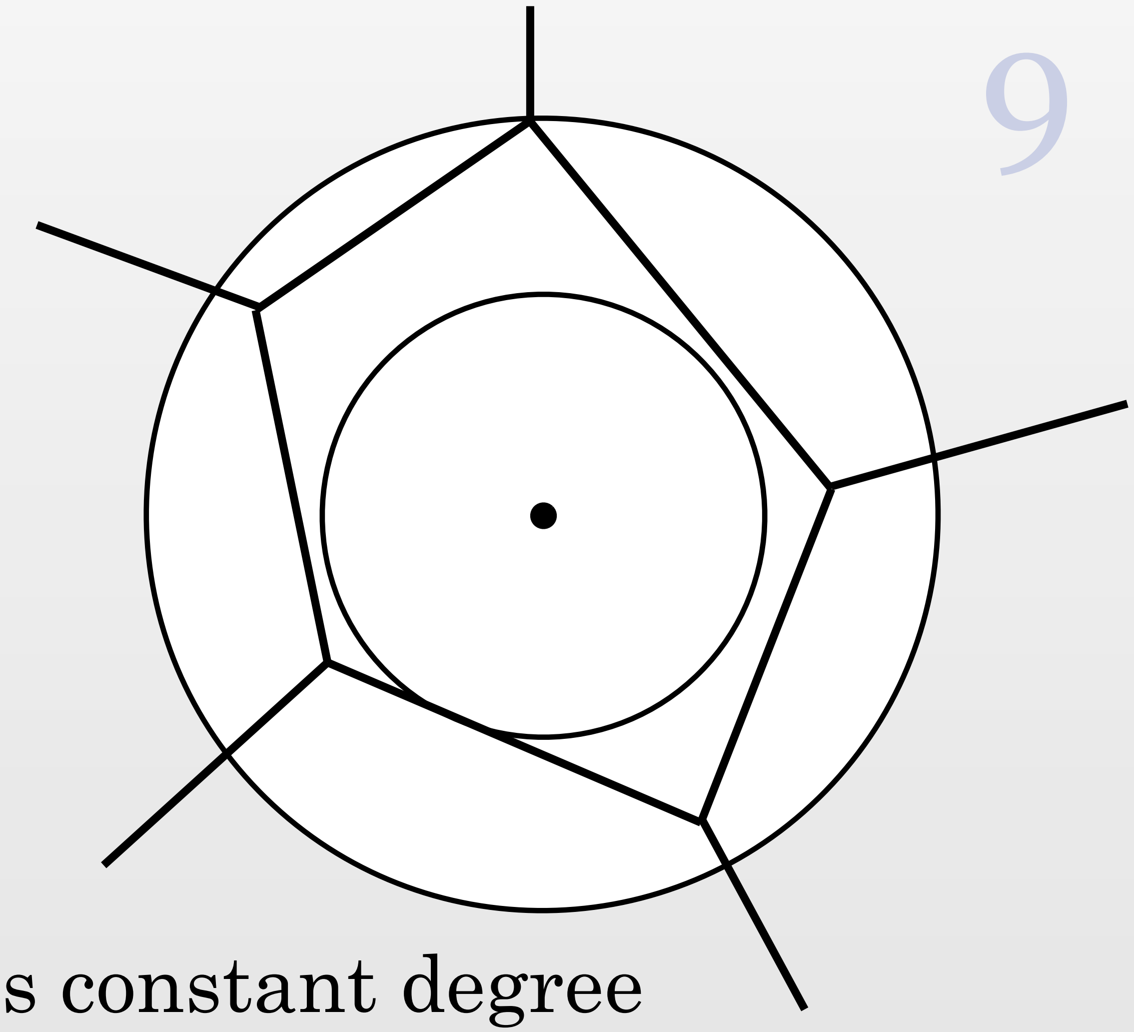
Delaunay balls have constant ply

Total number of faces is O(m)

Intersection graph of Delaunay balls has constant degree

Insertions only take constant time

Quality Meshes have several nice properties.



Delaunay balls have constant ply

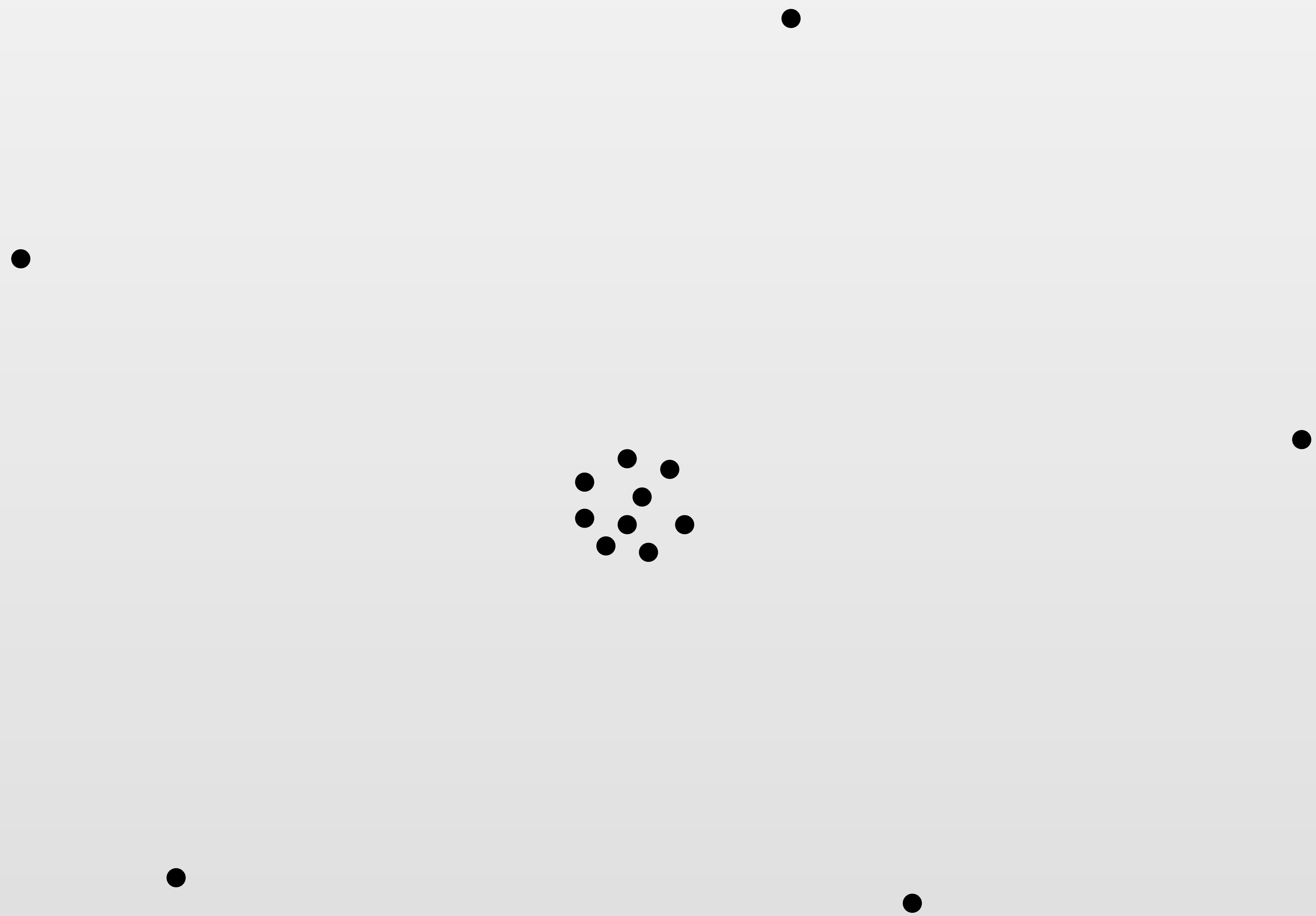
Total number of faces is O(m)

Intersection graph of Delaunay balls has constant degree

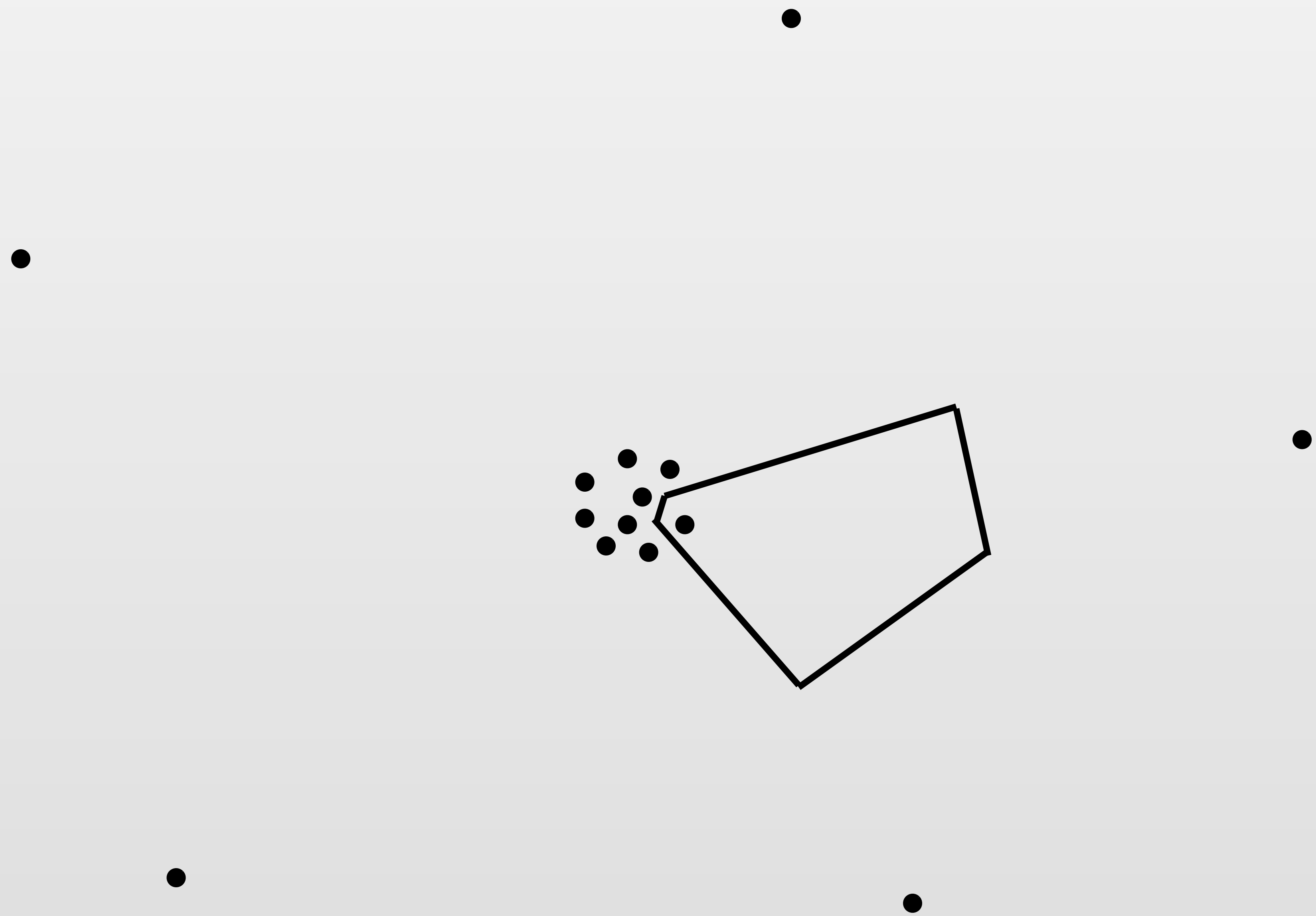
Insertions only take constant time

Voronoi Refinement: If some cell is skinny,
add a Steiner point at its farthest vertex.

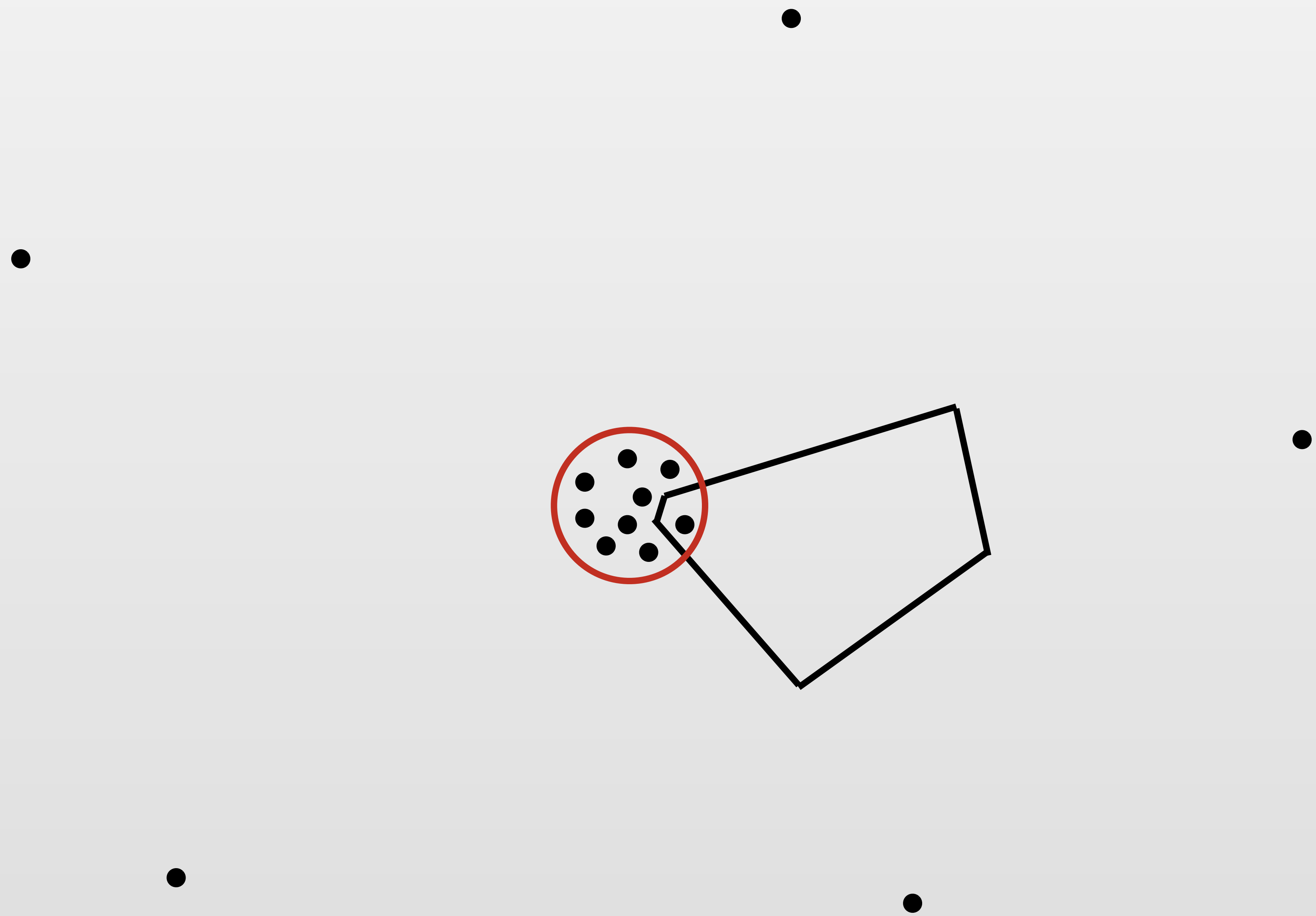
We replace *quality* with *hierarchical quality*.



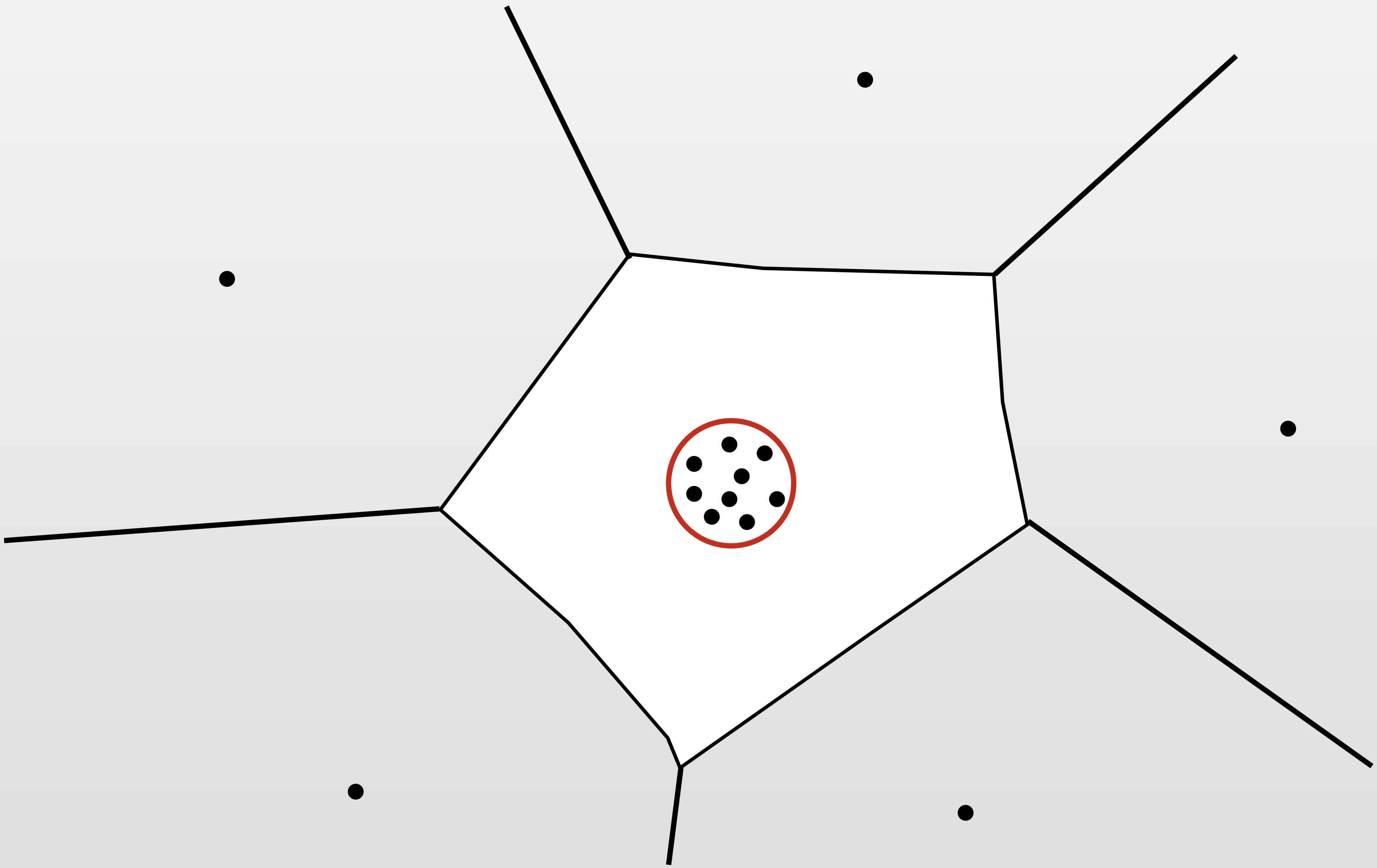
We replace *quality* with *hierarchical quality*.



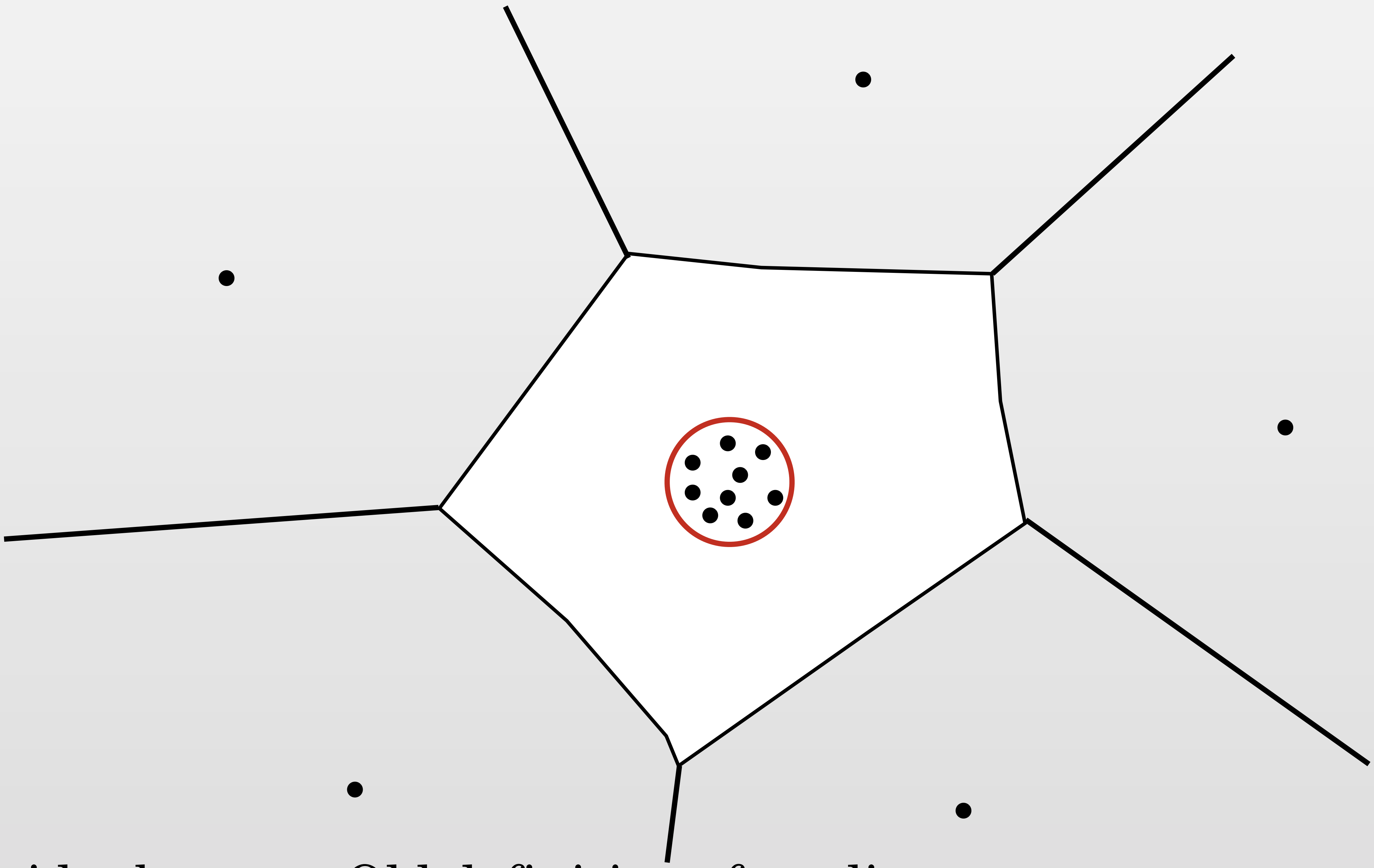
We replace *quality* with *hierarchical quality*.



We replace *quality* with *hierarchical quality*.

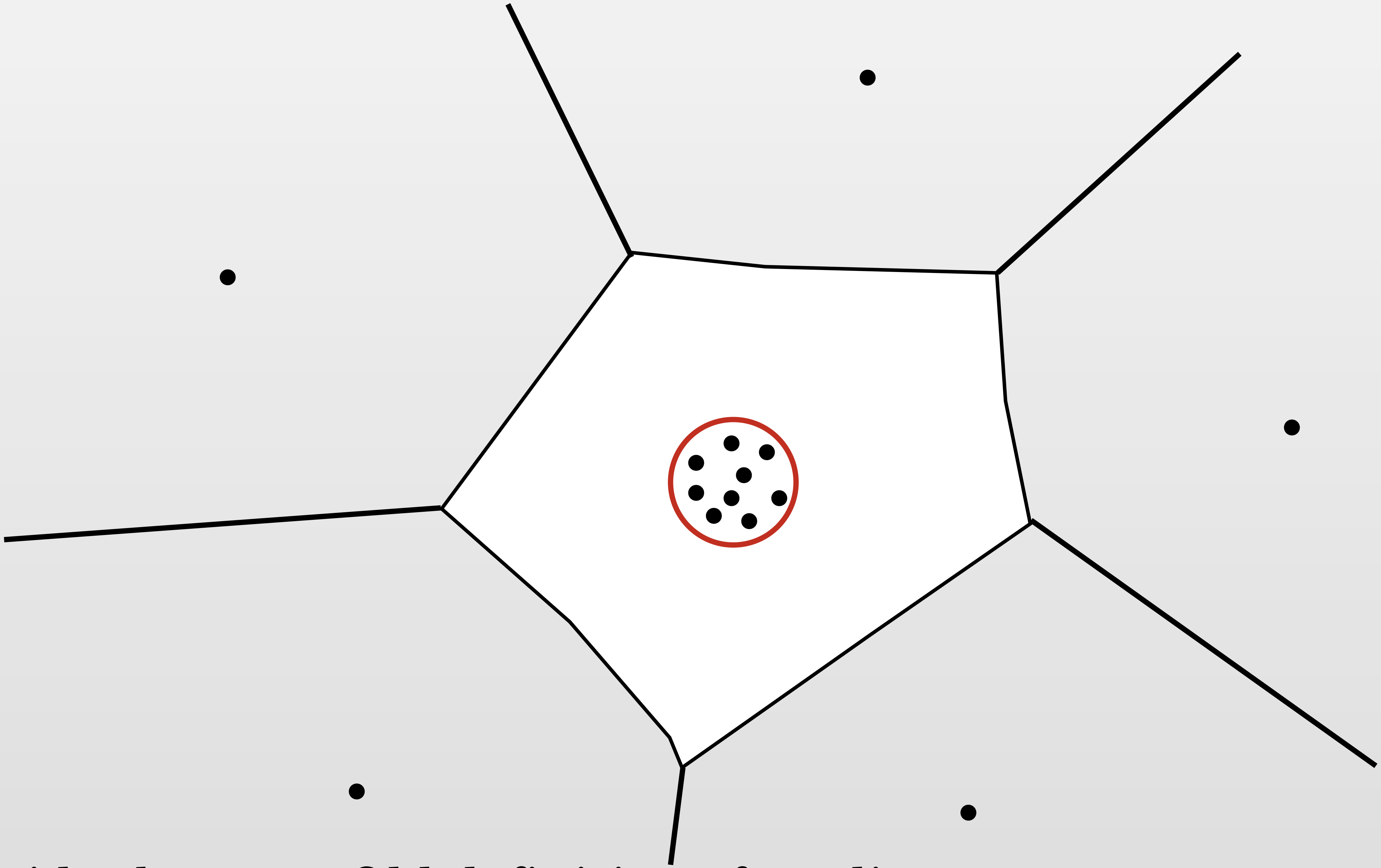


We replace *quality* with *hierarchical quality*.



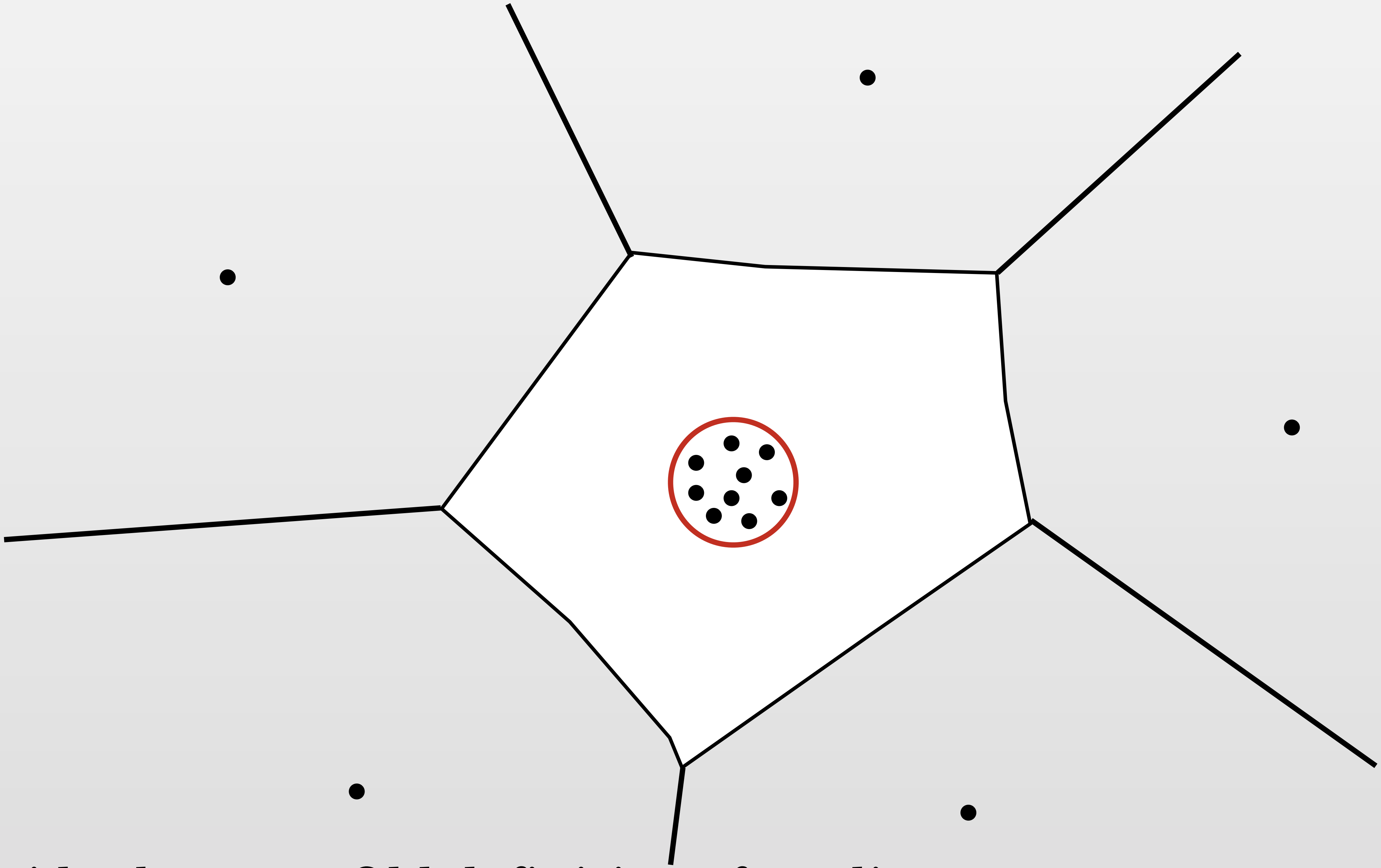
Inside the cage: Old definition of quality.

We replace *quality* with *hierarchical quality*.



Inside the cage: Old definition of quality.
Outside: Treat the whole cage as a single object.

We replace *quality* with *hierarchical quality*.



Inside the cage: Old definition of quality.
Outside: Treat the whole cage as a single object.

Has the same important properties as quality meshes: ply, degree,...

The sparse meshing model:

The sparse meshing model:

1. Build a Voronoi diagram incrementally.

The sparse meshing model:

1. Build a Voronoi diagram incrementally.
2. Interleave input and Steiner point insertions.

The sparse meshing model:

1. Build a Voronoi diagram incrementally.
2. Interleave input and Steiner point insertions.
3. Recover quality after each input point.

The sparse meshing model:

This is how we avoid the worst-case Voronoi bounds.

1. Build a Voronoi diagram incrementally.
2. Interleave input and Steiner point insertions.
3. Recover quality after each input point.

The New Algorithm

1. Build a Voronoi diagram incrementally.
2. Interleave input and Steiner point insertions.
3. Recover quality after each input point.

The New Algorithm

1 Build a Voronoi diagram incrementally.

2 Interleave input and Steiner point insertions.

3 Recover quality after each input point.

hierarchical quality

The New Algorithm

1 Build a Voronoi diagram incrementally.

2 Interleave input and Steiner point insertions.

3 Recover quality after each input point.

hierarchical quality

4 Store uninserted input points in the D-Balls.

The New Algorithm

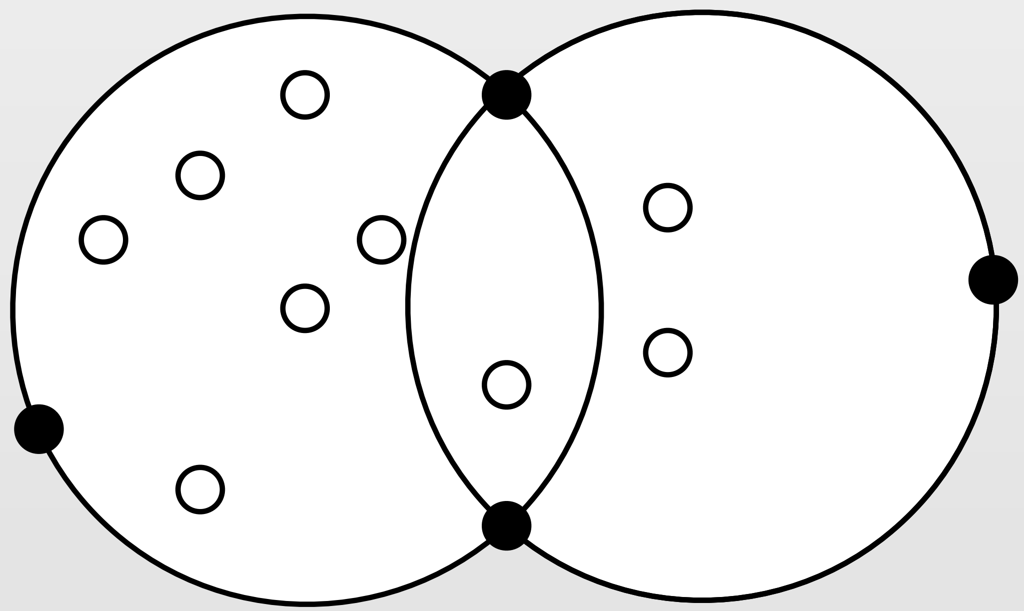
1. Build a Voronoi diagram incrementally.
2. Interleave input and Steiner point insertions.
3. Recover quality after each input point.

   hierarchical quality

4. Store uninserted input points in the D-Balls.
5. Order the input points using range space nets.

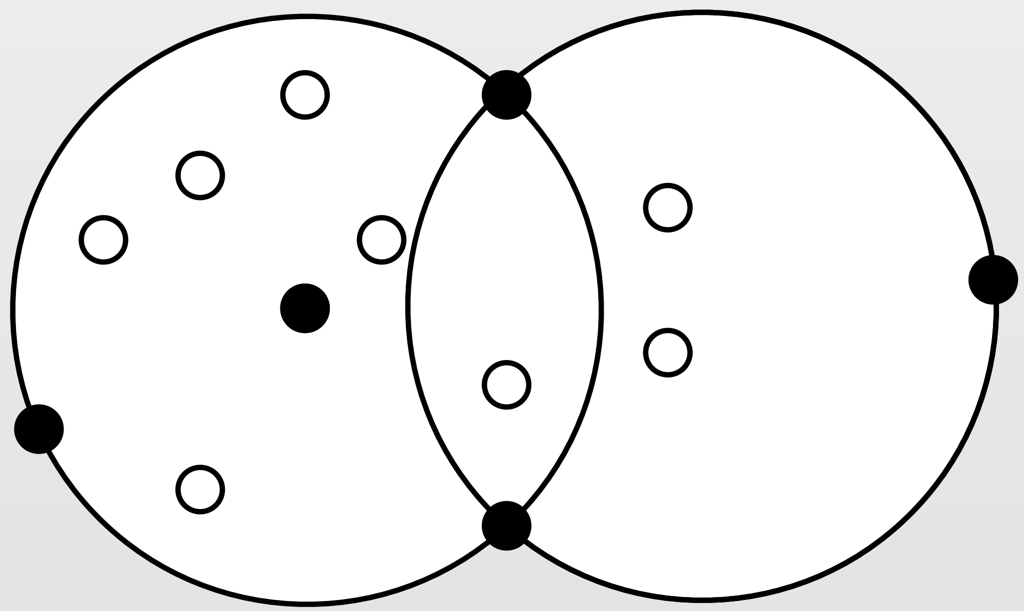
Point Location in the D-Balls.

Idea: Store the uninserted points in the D-balls.
When the balls change, make local updates.



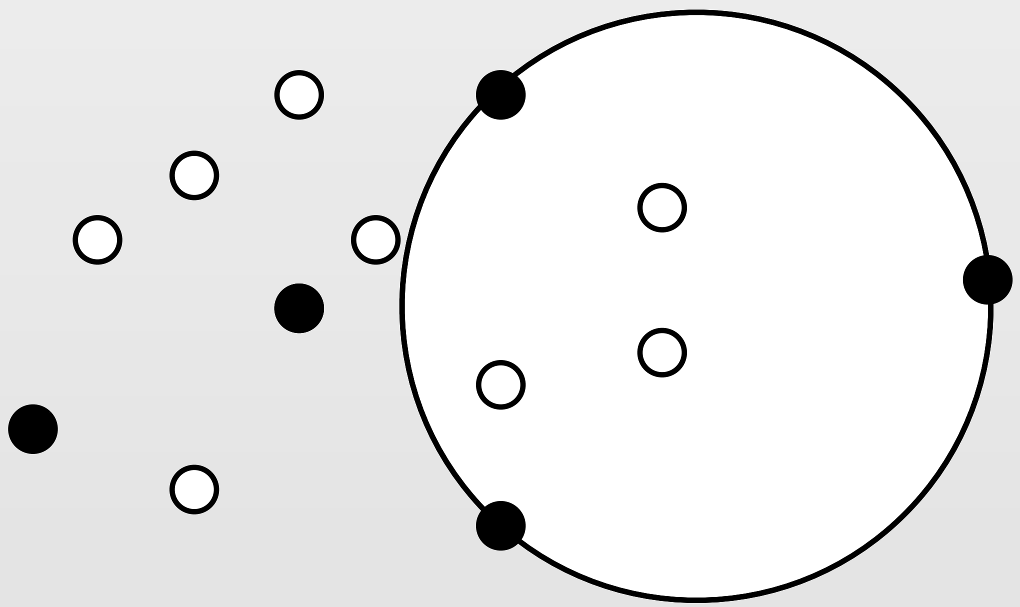
Point Location in the D-Balls.

Idea: Store the uninserted points in the D-balls.
When the balls change, make local updates.



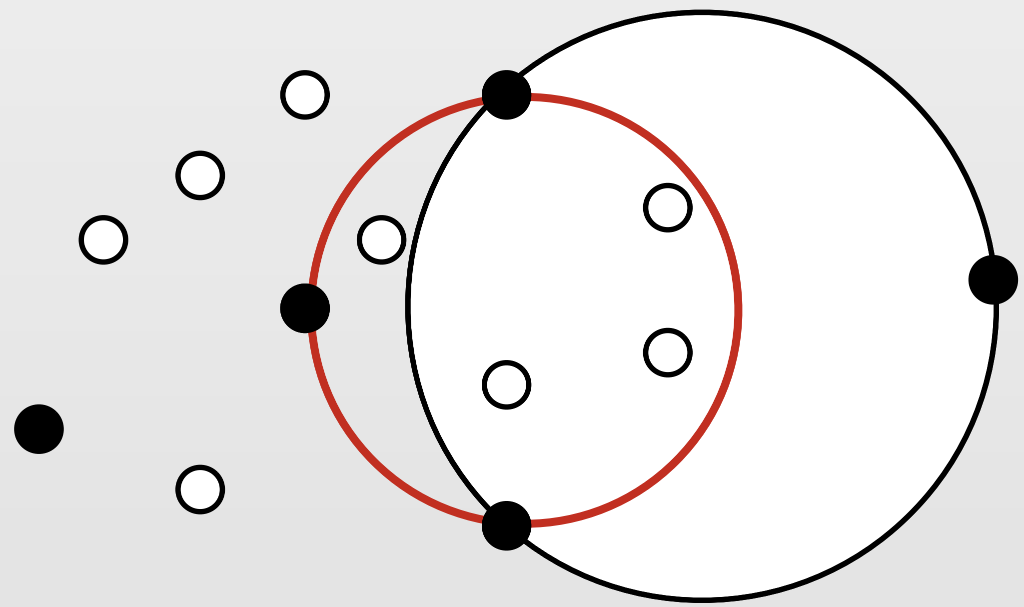
Point Location in the D-Balls.

Idea: Store the uninserted points in the D-balls.
When the balls change, make local updates.



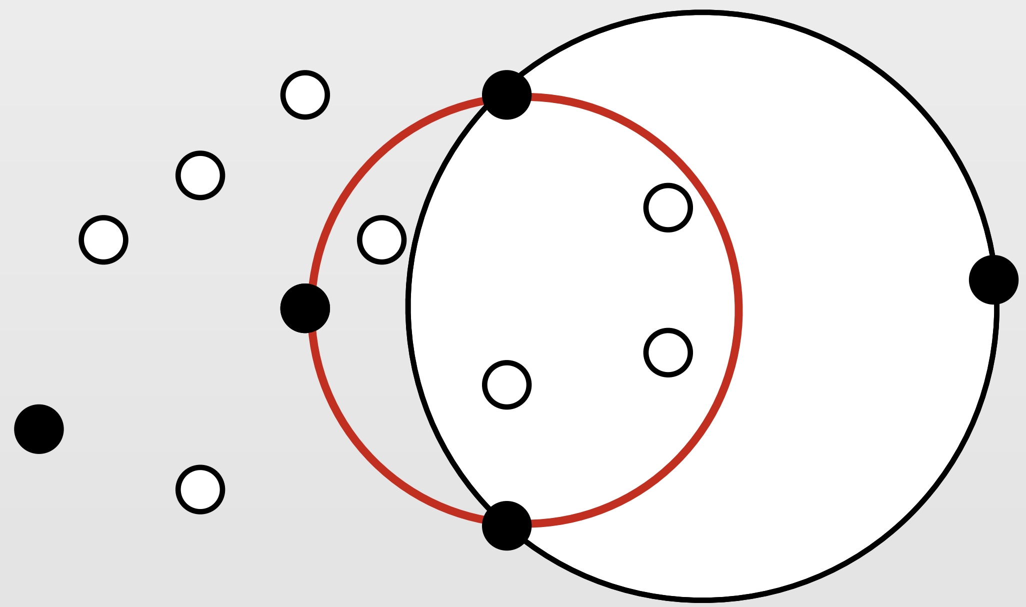
Point Location in the D-Balls.

Idea: Store the uninserted points in the D-balls.
When the balls change, make local updates.



Point Location in the D-Balls.

Idea: Store the uninserted points in the D-balls.
When the balls change, make local updates.



It's a history DAG!

Range Nets

Range Nets

Definition. *A **range space** is a pair* (X, R)*, where* X *is a set (the vertices) and* R *is a collection of subsets (the ranges).*

Range Nets

Definition. *A* ***range space*** *is a pair* (X, R)*, where* X *is a set (the vertices) and* R *is a collection of subsets (the ranges).*

For us $X = P$ and R is the set of open balls.

Range Nets

Definition. *A* ***range space*** *is a pair* (X, R), *where* X *is a set (the vertices) and* R *is a collection of subsets (the ranges).*

For us $X = P$ and R is the set of open balls.

Definition. *Given a range space* (X, R), *a set* $N \subset X$ *is a* ***range space*** ε***-net*** *if for all ranges* $r \in R$ *that contain at least* $\varepsilon|X|$ *vertices,* r *contains a vertex from* N.

Range Nets

Definition. *A* ***range space*** *is a pair* (X, R)*, where* X *is a set (the vertices) and* R *is a collection of subsets (the ranges).*

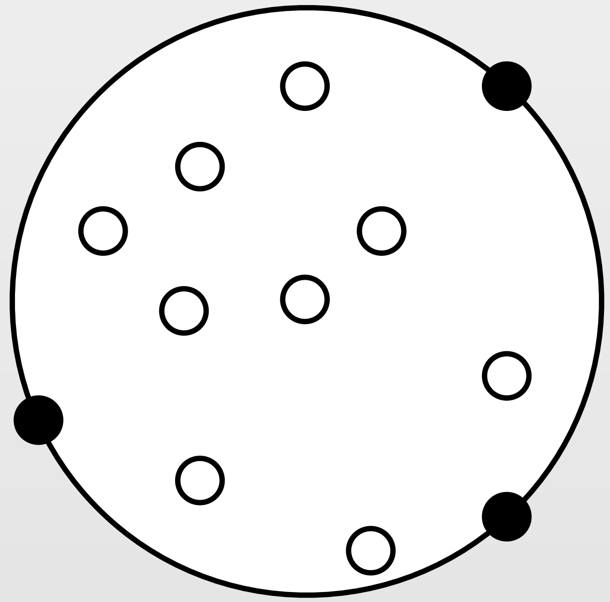
For us $X = P$ and R is the set of open balls.

Definition. *Given a range space* (X, R)*, a set* $N \subset X$ *is a* ***range space*** ε***-net*** *if for all ranges* $r \in R$ *that contain at least* $\varepsilon|X|$ *vertices,* r *contains a vertex from* N*.*

Theorem: [Chazelle & Matousek 96] For ε, d fixed constants, ε-nets of size $O(1)$ can be computed in $O(n)$ deterministic time.

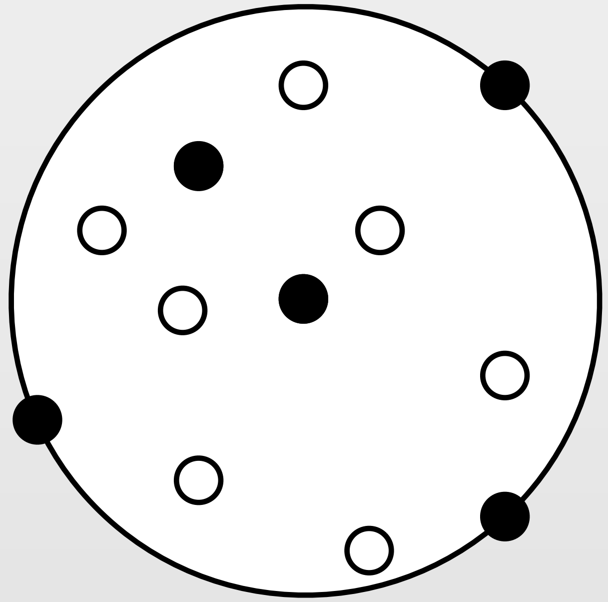
Ordering the inputs

For each D-Ball, select a $\frac{1}{2d}$-net of the points it contains.
Take the union of these nets and call it a round.
Insert these.
Repeat.



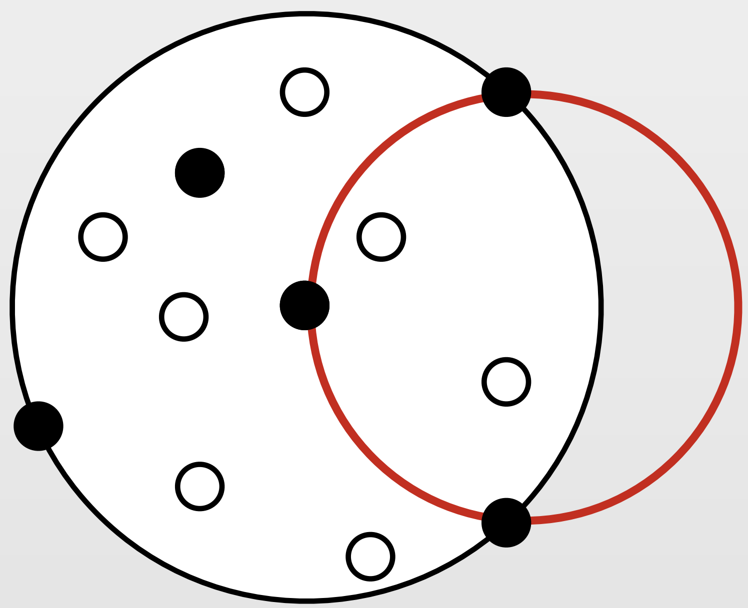
Ordering the inputs

For each D-Ball, select a $\frac{1}{2d}$-net of the points it contains.
Take the union of these nets and call it a round.
Insert these.
Repeat.



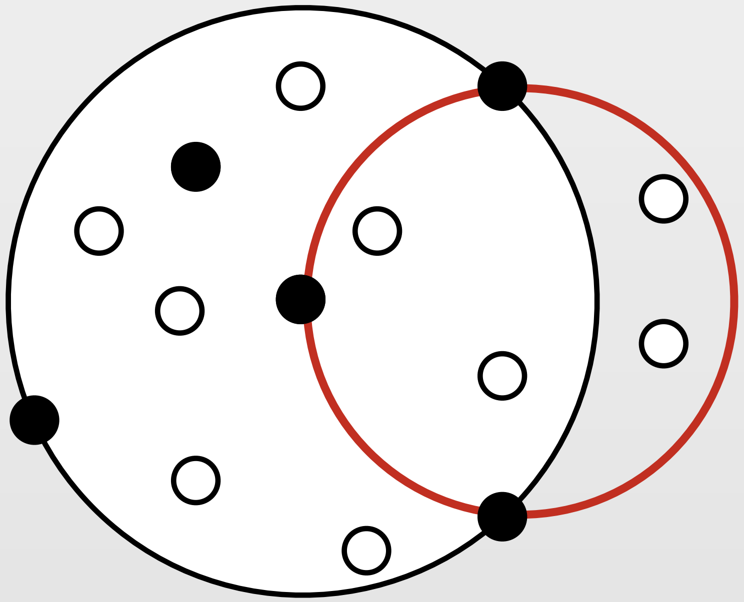
Ordering the inputs

For each D-Ball, select a $\frac{1}{2d}$-net of the points it contains.
Take the union of these nets and call it a round.
Insert these.
Repeat.



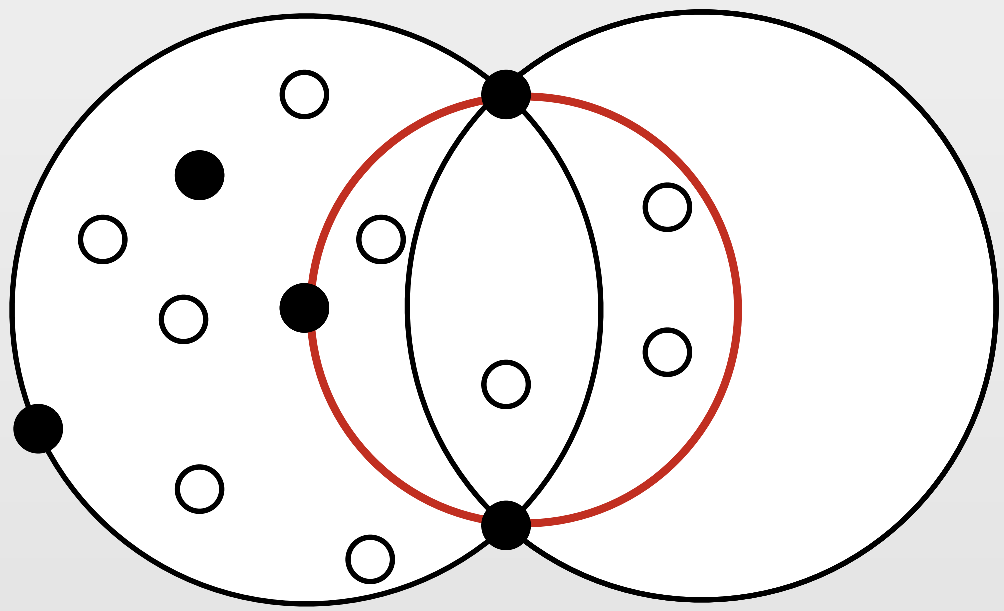
Ordering the inputs

For each D-Ball, select a $\frac{1}{2d}$-net of the points it contains.
Take the union of these nets and call it a round.
Insert these.
Repeat.



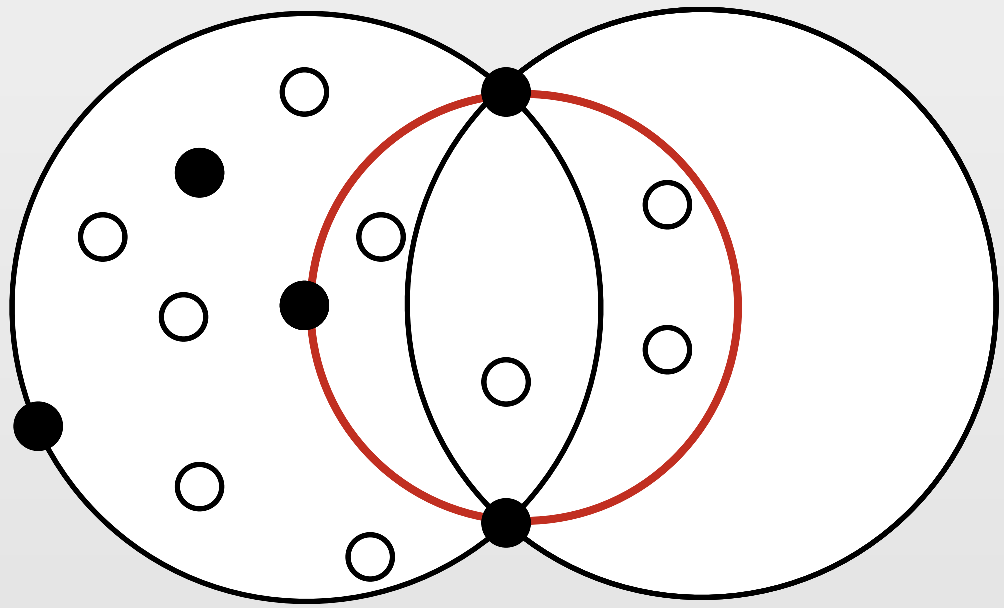
Ordering the inputs

For each D-Ball, select a $\frac{1}{2d}$-net of the points it contains.
Take the union of these nets and call it a round.
Insert these.
Repeat.



Ordering the inputs

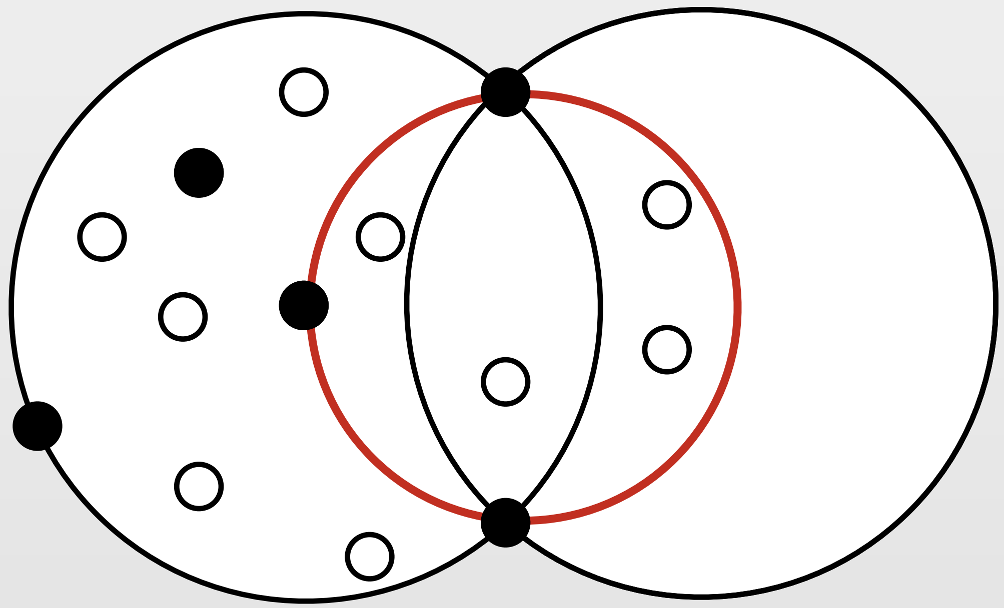
For each D-Ball, select a $\frac{1}{2d}$-net of the points it contains.
Take the union of these nets and call it a round.
Insert these.
Repeat.



Lemma. *Let M be a set of vertices. If an open ball B contains no points of M, then B is contained in the union of d D-balls of M.*

Ordering the inputs

For each D-Ball, select a $\frac{1}{2d}$-net of the points it contains.
Take the union of these nets and call it a round.
Insert these.
Repeat.



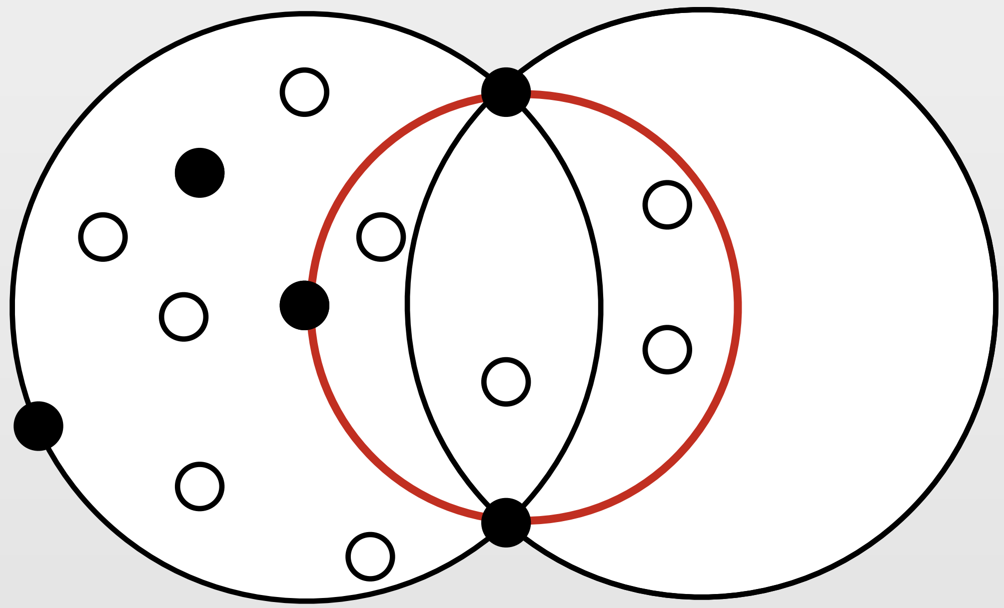
$$\max_{\text{D-balls } B} |B \cap P|$$

Goes down by a factor of 2 each round.

Lemma. *Let M be a set of vertices. If an open ball B contains no points of M, then B is contained in the union of d D-balls of M.*

Ordering the inputs

For each D-Ball, select a $\frac{1}{2d}$-net of the points it contains.
Take the union of these nets and call it a round.
Insert these.
Repeat.



$$\max_{\text{D-balls } B} |B \cap P|$$

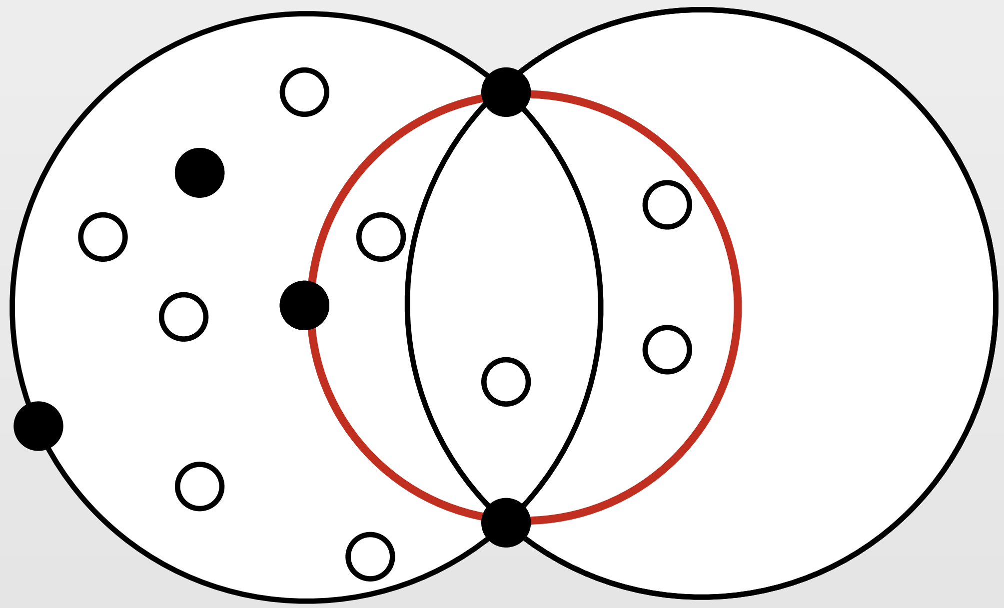
Goes down by a factor of 2 each round.

=> $\log(n)$ Rounds

Lemma. *Let M be a set of vertices. If an open ball B contains no points of M, then B is contained in the union of d D-balls of M.*

Ordering the inputs

For each D-Ball, select a $\frac{1}{2d}$-net of the points it contains.
Take the union of these nets and call it a round.
Insert these.
Repeat.



$$\max_{\text{D-balls } B} |B \cap P|$$

Goes down by a factor of 2 each round.

=> $\log(n)$ Rounds

Lemma. *Let M be a set of vertices. If an open ball B contains no points of M, then B is contained in the union of d D-balls of M.*

Note: After $k = \log \frac{1}{\epsilon}$ rounds, the intermediate mesh is a **weak ϵ-net** for the range space of Euclidean balls.
Size: $O\left(\frac{1}{\epsilon}\right)$, Time: $O(nk) = O(n \log \frac{1}{\epsilon})$.

To complete the analysis, we must show that the cost of a Round is O(n).

log n rounds x $O(n)$ time/round = $O(n \log n)$

Watch an uninserted point x.
Claim: x only gets touched $O(1)$ times per round.

To complete the analysis, we must show that the cost of a Round is O(n).

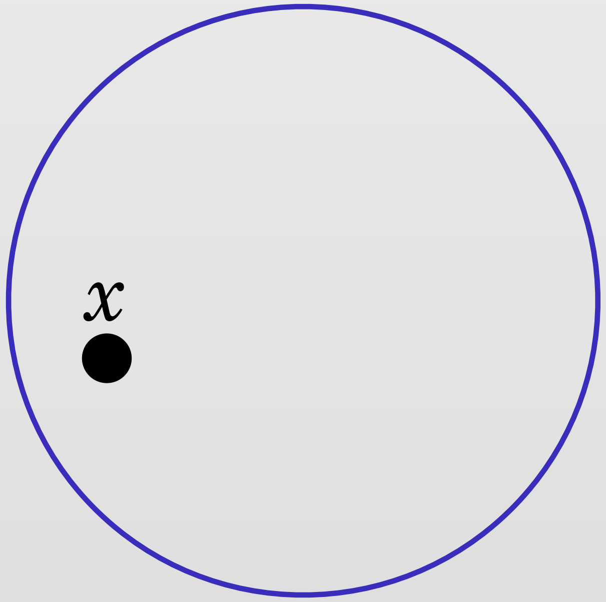
log n rounds x $O(n)$ time/round = $O(n \log n)$

Watch an uninserted point x.
Claim: x only gets touched $O(1)$ times per round.

To complete the analysis, we must show that the cost of a Round is O(n).

$\log n$ rounds x $O(n)$ time/round = $O(n \log n)$

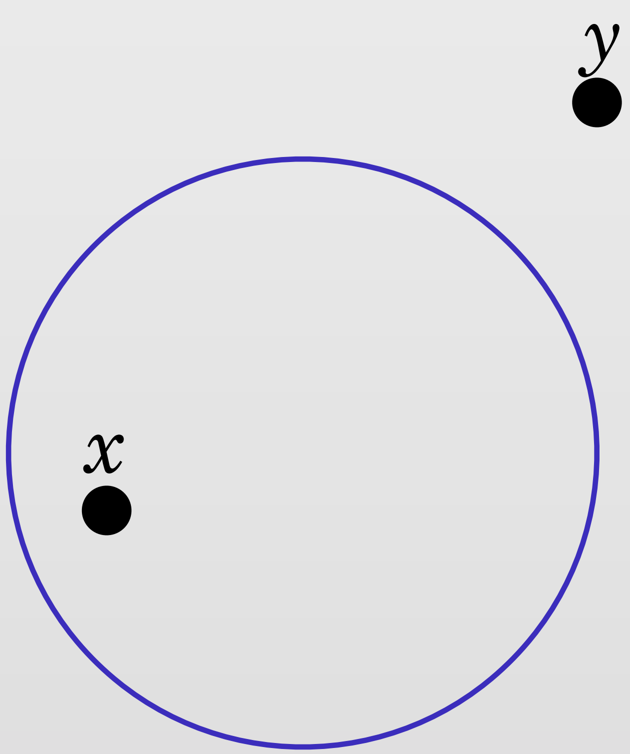
Watch an uninserted point x.
Claim: x only gets touched $O(1)$ times per round.



To complete the analysis, we must show that the cost of a Round is O(n).

$\log n$ rounds x $O(n)$ time/round = $O(n \log n)$

Watch an uninserted point x.
Claim: x only gets touched $O(1)$ times per round.

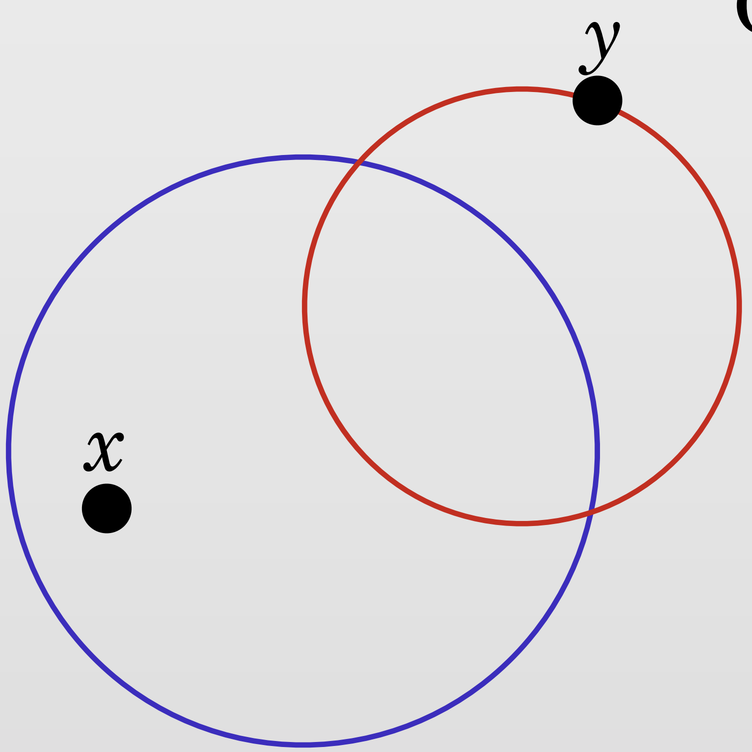


To complete the analysis, we must show that the cost of a Round is O(n).

$\log n$ rounds x $O(n)$ time/round = $O(n \log n)$

Watch an uninserted point x.

Claim: x only gets touched $O(1)$ times per round.

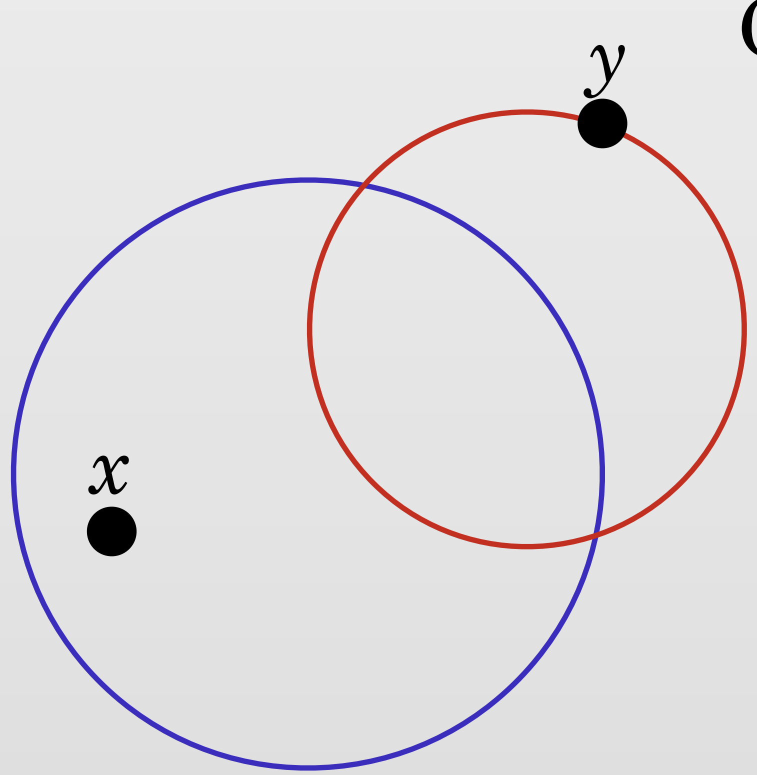


To complete the analysis, we must show that the cost of a Round is O(n).

$\log n$ rounds x $O(n)$ time/round = $O(n \log n)$

Watch an uninserted point x.

Claim: x only gets touched $O(1)$ times per round.

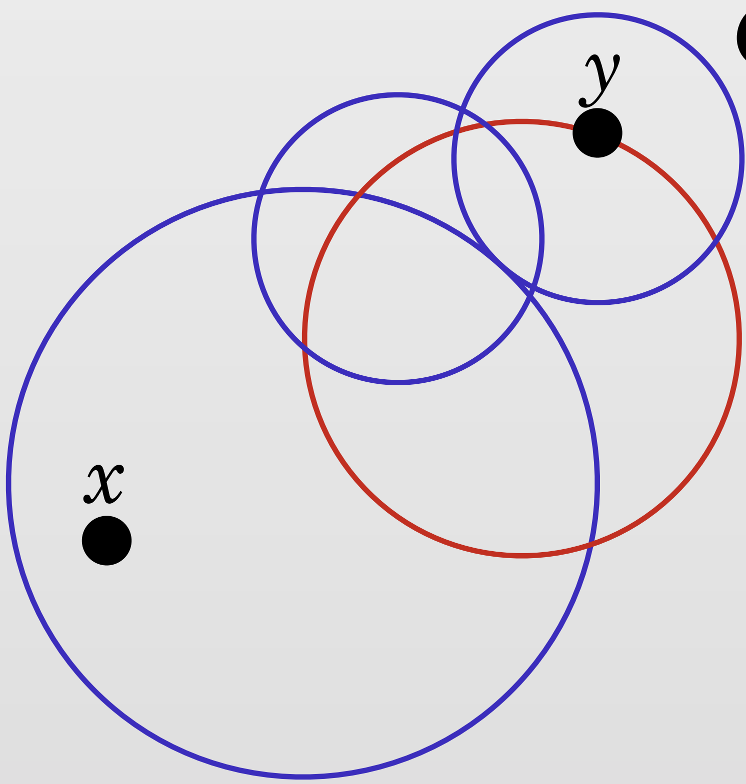


y touches x => y is "close" to x

close = 2 hops among D-balls

To complete the analysis, we must show that the cost of a Round is O(n).

log n rounds x $O(n)$ time/round = $O(n \log n)$



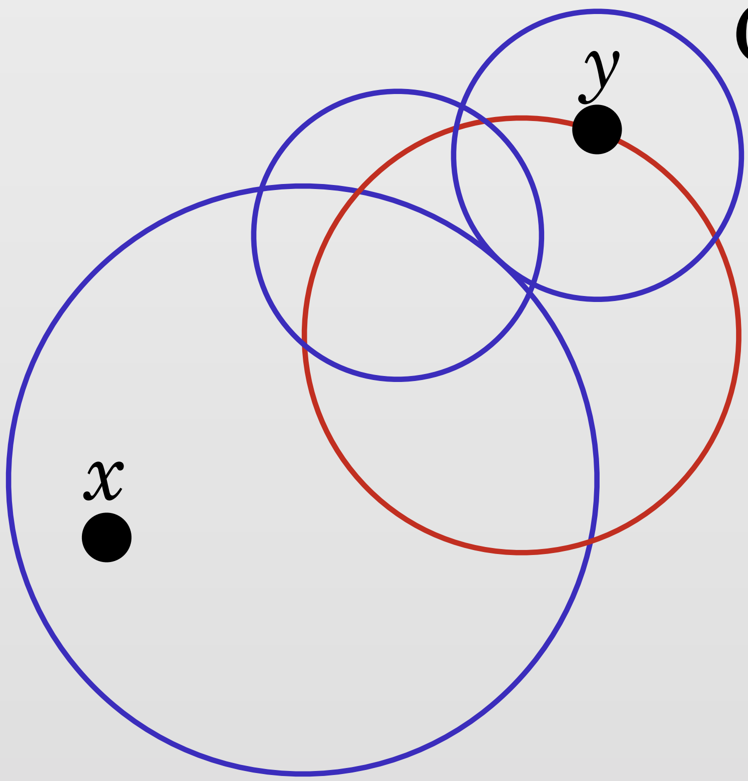
Watch an uninserted point x.
Claim: x only gets touched $O(1)$ times per round.

y touches x => y is "close" to x
close = 2 hops among D-balls

To complete the analysis, we must show that the cost of a Round is O(n).

log n rounds x $O(n)$ time/round = $O(n \log n)$

Watch an uninserted point x.
Claim: x only gets touched $O(1)$ times per round.



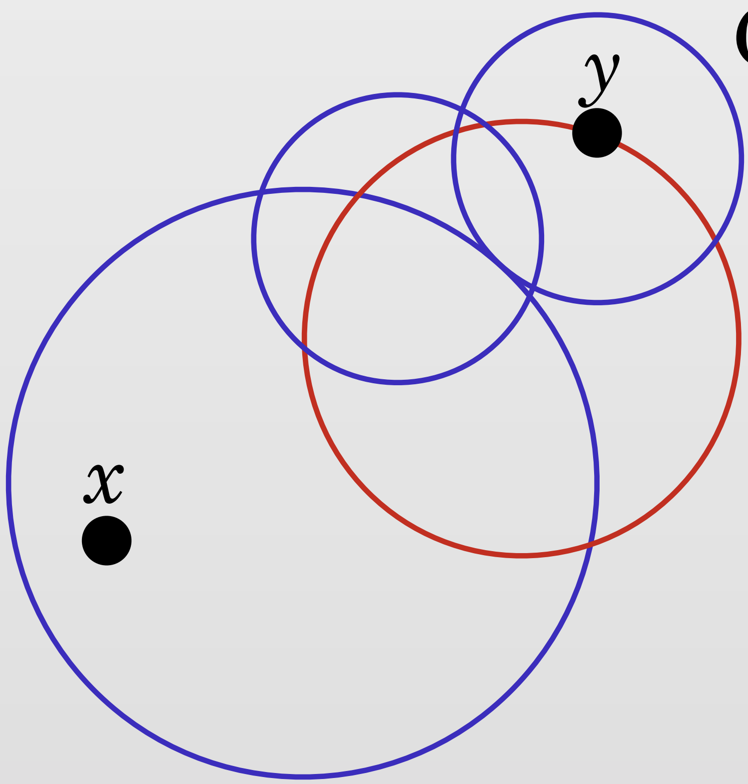
y touches x => y is "close" to x
close = 2 hops among D-balls

Only $O(1)$ D-balls are within 2 hops.

To complete the analysis, we must show that the cost of a Round is O(n).

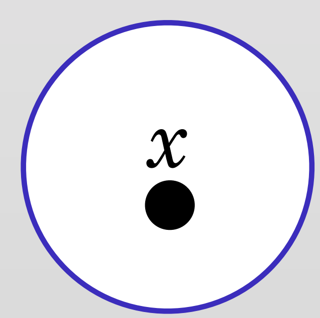
$\log n$ rounds x $O(n)$ time/round = $O(n \log n)$

Watch an uninserted point x.
Claim: x only gets touched $O(1)$ times per round.



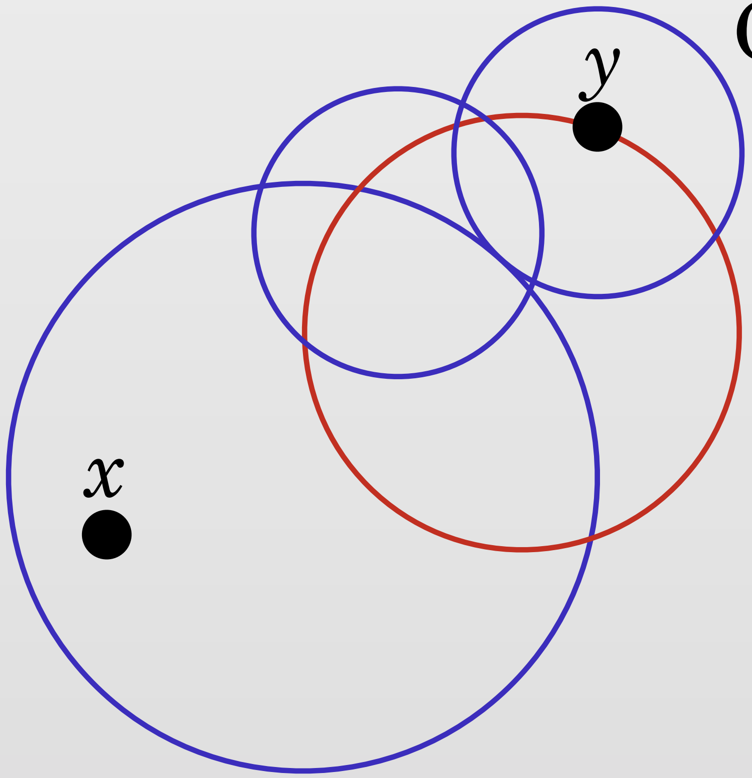
y touches x => y is "close" to x
close = 2 hops among D-balls

Only $O(1)$ D-balls are within 2 hops.



To complete the analysis, we must show that the cost of a Round is O(n).

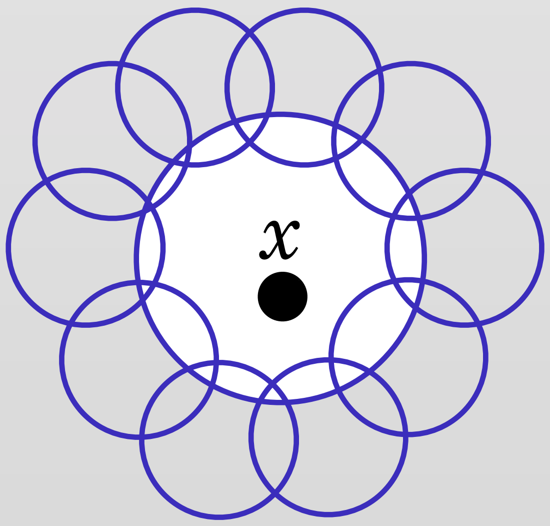
$\log n$ rounds x $O(n)$ time/round = $O(n \log n)$



Watch an uninserted point x.
Claim: x only gets touched $O(1)$ times per round.

y touches x => y is "close" to x
close = 2 hops among D-balls

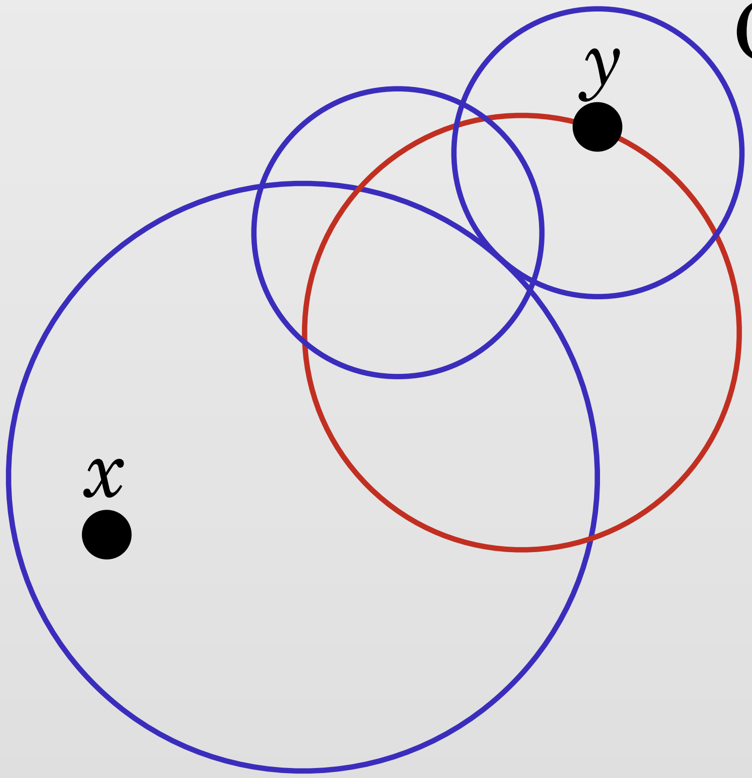
Only $O(1)$ D-balls are within 2 hops.



To complete the analysis, we must show that the cost of a Round is O(n).

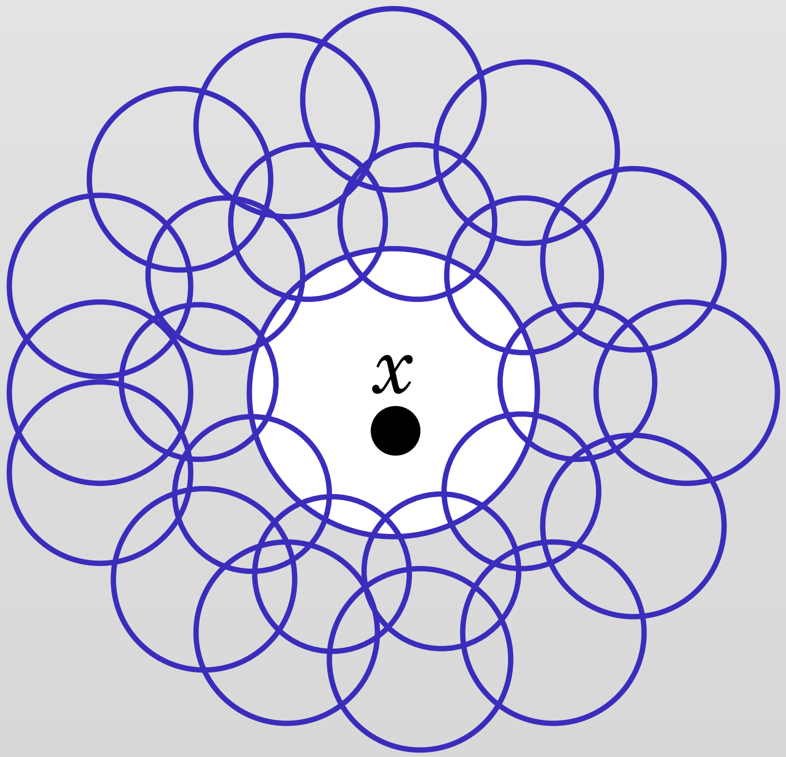
$\log n$ rounds x $O(n)$ time/round = $O(n \log n)$

Watch an uninserted point x.
Claim: x only gets touched $O(1)$ times per round.



y touches x => y is “close” to x
close = 2 hops among D-balls

Only $O(1)$ D-balls are within 2 hops.

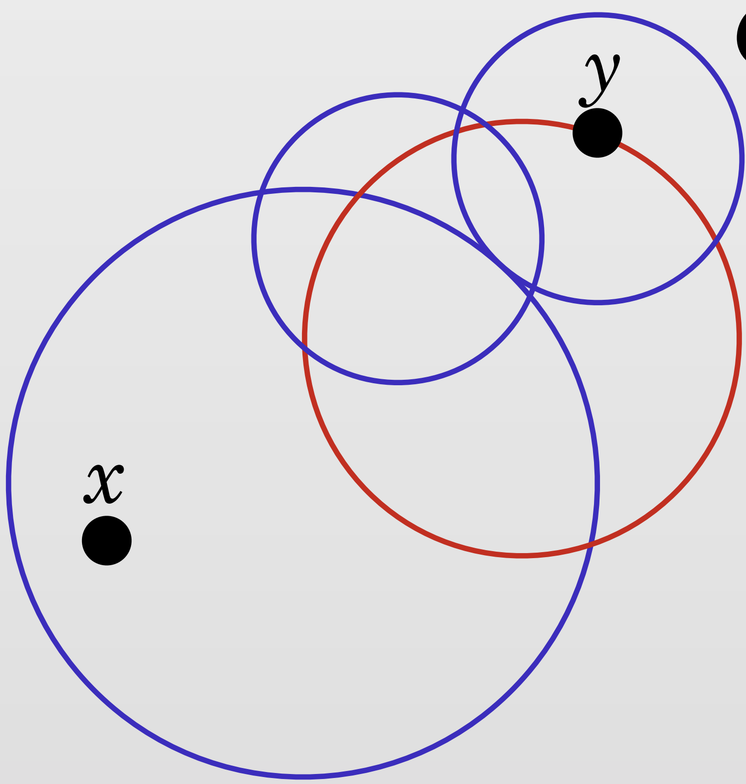


To complete the analysis, we must show that the cost of a Round is O(n).

$\log n$ rounds x $O(n)$ time/round = $O(n \log n)$

Watch an uninserted point x.

Claim: x only gets touched $O(1)$ times per round.

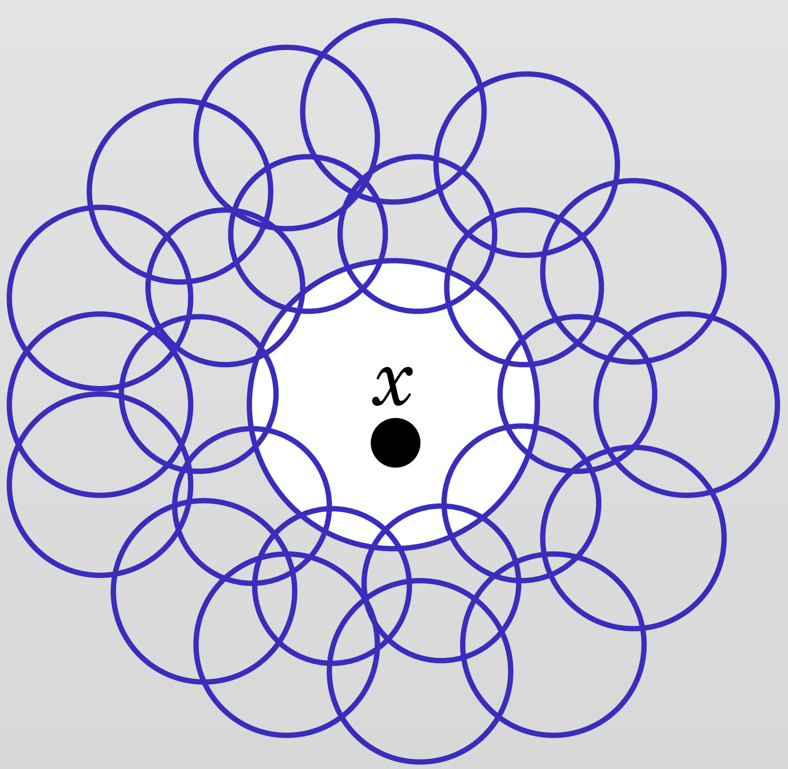


y touches x => y is “close” to x

close = 2 hops among D-balls

Only $O(1)$ D-balls are within 2 hops.

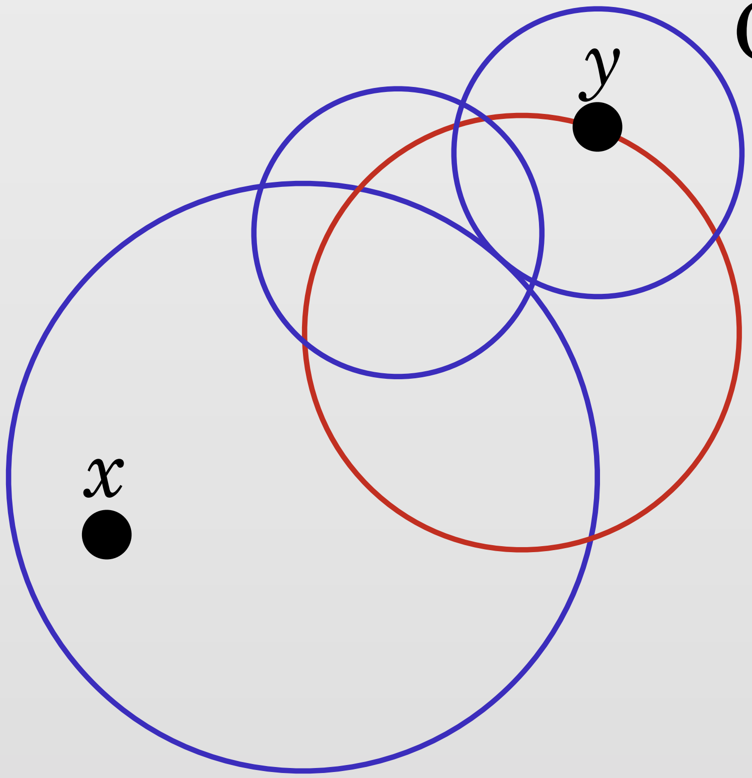
Only $O(1)$ points are added to any D-ball in a round.



To complete the analysis, we must show that the cost of a Round is O(n).

$\log n$ rounds x $O(n)$ time/round = $O(n \log n)$

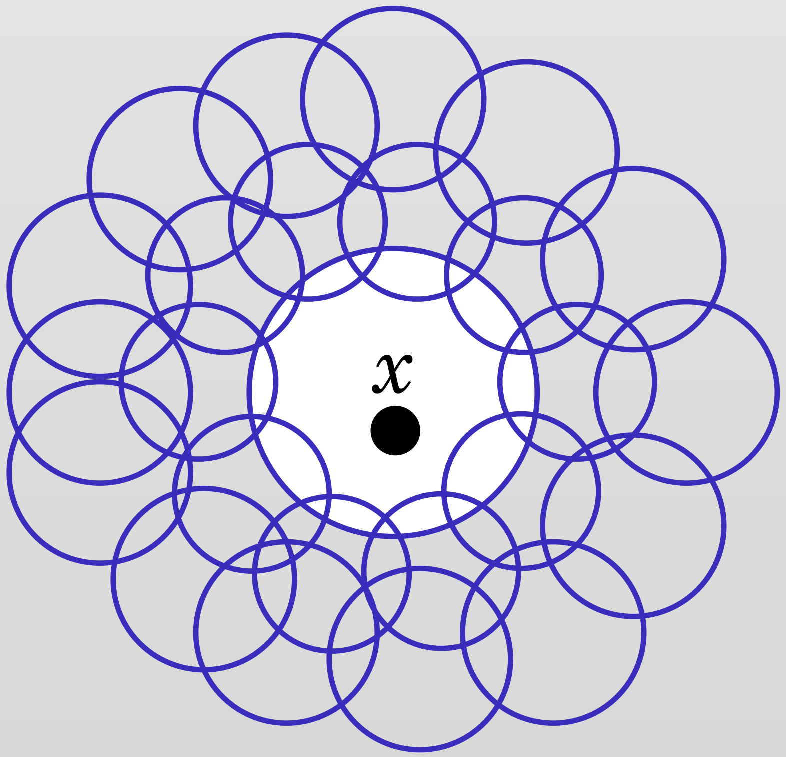
Watch an uninserted point x.
Claim: x only gets touched $O(1)$ times per round.



y touches x => y is "close" to x
close = 2 hops among D-balls

Only $O(1)$ D-balls are within 2 hops.

Only $O(1)$ points are added to any D-ball in a round.



$O(n)$ total work per round.

Meshing Points in (optimal)
$O(n \log n + m)$ time.

Thank you.