

Computational Geometry

Winter semester 2014/15

Arrangements and Duality

Lecture #12

Rendering und Ray-Tracing

- Determine which object in the scene is visible through a given pixel (given the viewing direction of the observer).

Rendering und Ray-Tracing

- Determine which object in the scene is visible through a given pixel (given the viewing direction of the observer).
- Compute the intensity of the light that the visible object emits in the direction of the pixel.

Rendering und Ray-Tracing

- Determine which object in the scene is visible through a given pixel (given the viewing direction of the observer).
- Compute the intensity of the light that the visible object emits in the direction of the pixel.
- How to color a pixel that is intersected by the boundary of an object?



Rendering und Ray-Tracing

- Determine which object in the scene is visible through a given pixel (given the viewing direction of the observer).
- Compute the intensity of the light that the visible object emits in the direction of the pixel.
- How to color a pixel that is intersected by the boundary of an object?
 - As the object or the background?



Rendering und Ray-Tracing

- Determine which object in the scene is visible through a given pixel (given the viewing direction of the observer).
- Compute the intensity of the light that the visible object emits in the direction of the pixel.
- How to color a pixel that is intersected by the boundary of an object?
 - As the object or the background?
Problem: object boundaries look “pixelated” .



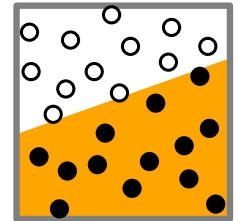
Rendering und Ray-Tracing

- Determine which object in the scene is visible through a given pixel (given the viewing direction of the observer).
- Compute the intensity of the light that the visible object emits in the direction of the pixel.
- How to color a pixel that is intersected by the boundary of an object?
 - As the object or the background?
Problem: object boundaries look “pixelated” .
 - Better: if the object occupies $x\%$ of the pixel area (and the background $(100 - x)\%$, color the pixel with ratio
$$\text{object color} / \text{background color} = x / (100 - x).$$



Rendering und Ray-Tracing

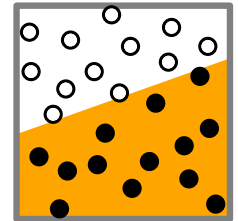
- Determine which object in the scene is visible through a given pixel (given the viewing direction of the observer).
 - Compute the intensity of the light that the visible object emits in the direction of the pixel.
 - How to color a pixel that is intersected by the boundary of an object?
 - As the object or the background?
Problem: object boundaries look “pixelated” .
 - Better: if the object occupies $x\%$ of the pixel area (and the background $(100 - x)\%$, color the pixel with ratio
$$\text{object color} / \text{background color} = x / (100 - x).$$
- Supersampling:* – send many rays through the pixel



Rendering und Ray-Tracing

- Determine which object in the scene is visible through a given pixel (given the viewing direction of the observer).
- Compute the intensity of the light that the visible object emits in the direction of the pixel.

- How to color a pixel that is intersected by the boundary of an object?



- As the object or the background?

Problem: object boundaries look “pixelated” .

- Better: if the object occupies $x\%$ of the pixel area (and the background $(100 - x)\%$, color the pixel with ratio

$$\text{object color} / \text{background color} = x / (100 - x).$$

Supersampling: – send many rays through the pixel

– use ratio
$$\frac{\# \text{ rays that hit object}}{\# \text{ rays that hit background}} = \frac{|\bullet|}{|\circ|}$$

Supersampling

- distribute rays regularly:

Supersampling

- distribute rays regularly:
leads to artefacts that can be visible

Supersampling

- distribute rays regularly:
leads to artefacts that can be visible
- distribute rays randomly:

Supersampling

- distribute rays regularly:
leads to artefacts that can be visible
- distribute rays randomly:
may lead to better results, but –

Supersampling

- distribute rays regularly:
leads to artefacts that can be visible
- distribute rays randomly:
may lead to better results, but –
how to measure the quality of a distribution?

Supersampling

- distribute rays regularly:
leads to artefacts that can be visible
- distribute rays randomly:
may lead to better results, but –
how to measure the quality of a distribution?
(if the distribution *is* bad, simply produce a new one!)

Supersampling

- distribute rays regularly:
leads to artefacts that can be visible
- distribute rays randomly:
may lead to better results, but –
how to measure the quality of a distribution?
(if the distribution *is* bad, simply produce a new one!)

Def. *discrepancy* $\Delta(\text{distribution } S, \text{object } o) =$

Supersampling

- distribute rays regularly:
leads to artefacts that can be visible
- distribute rays randomly:
may lead to better results, but –
how to measure the quality of a distribution?
(if the distribution *is* bad, simply produce a new one!)

Def. *discrepancy* $\Delta(\text{distribution } S, \text{object } o) = |\mu(o) - \mu_S(o)|,$

Supersampling

- distribute rays regularly:
leads to artefacts that can be visible
- distribute rays randomly:
may lead to better results, but –
how to measure the quality of a distribution?
(if the distribution *is* bad, simply produce a new one!)

Def. *discrepancy* $\Delta(\text{distribution } S, \text{object } o) = |\mu(o) - \mu_S(o)|$,
where

$$\mu(o) = \text{area}(Q \cap o)$$

$$Q = [0, 1] \times [0, 1] \hat{=} \text{pixel}$$

$$\mu_S(o) = |S \cap o| / |S|$$

Supersampling

- distribute rays regularly:
leads to artefacts that can be visible
- distribute rays randomly:
may lead to better results, but –
how to measure the quality of a distribution?
(if the distribution *is* bad, simply produce a new one!)

Def. *discrepancy* $\Delta(\text{distribution } S, \text{object } o) = |\mu(o) - \mu_S(o)|$,
and

$$\Delta(S) = \sup_{o \text{ object}} \Delta(S, o).$$

where

$$\mu(o) = \text{area}(Q \cap o)$$

$$Q = [0, 1] \times [0, 1] \hat{=} \text{pixel}$$

$$\mu_S(o) = |S \cap o| / |S|$$

Supersampling

- distribute rays regularly:
leads to artefacts that can be visible
- distribute rays randomly:
may lead to better results, but –
how to measure the quality of a distribution?
(if the distribution *is* bad, simply produce a new one!)

Def. *discrepancy* $\Delta(\text{distribution } S, \text{object } o) = |\mu(o) - \mu_S(o)|$,

and

$$\Delta(S) = \sup_{o \text{ object}} \Delta(S, o).$$

simpler:

$$\Delta(S) = \sup_{h \in \mathcal{H}} \Delta(S, h).$$

where

$$\mu(o) = \text{area}(Q \cap o)$$

$$Q = [0, 1] \times [0, 1] \hat{=} \text{pixel}$$

$$\mu_S(o) = |S \cap o| / |S|$$

Supersampling

- distribute rays regularly:
leads to artefacts that can be visible
- distribute rays randomly:
may lead to better results, but –
how to measure the quality of a distribution?
(if the distribution *is* bad, simply produce a new one!)

Def. *discrepancy* $\Delta(\text{distribution } S, \text{object } o) = |\mu(o) - \mu_S(o)|$,

and

$$\Delta(S) = \sup_{o \text{ object}} \Delta(S, o).$$

simpler:

$$\Delta(S) = \sup_{h \in \mathcal{H}} \Delta(S, h).$$

where

$$\mu(o) = \text{area}(Q \cap o)$$

$$Q = [0, 1] \times [0, 1] \hat{=} \text{pixel}$$

$$\mu_S(o) = |S \cap o| / |S|$$

\mathcal{H} = set of all closed half-planes
whose boundary intersects Q

Halfplane discrepancy

$$\Delta(S) = \sup_{h \in \mathcal{H}} \Delta(S, h).$$

Problem:

Halfplane discrepancy

$$\Delta(S) = \sup_{h \in \mathcal{H}} \Delta(S, h).$$

Problem: is uncountable :-)

Halfplane discrepancy

$$\Delta(S) = \sup_{h \in \mathcal{H}} \Delta(S, h).$$

Problem: is uncountable :-)

Strategy:

Halfplane discrepancy

$$\Delta(S) = \sup_{h \in \mathcal{H}} \Delta(S, h).$$

Problem: is uncountable :-)

Strategy: – choose candidate set $\mathcal{H}' \subseteq \mathcal{H}$ which contains a halfplane h that maximizes $\Delta(S, h)$

Halfplane discrepancy

$$\Delta(S) = \sup_{h \in \mathcal{H}} \Delta(S, h).$$

Problem: is uncountable :-)

- Strategy:**
- choose candidate set $\mathcal{H}' \subseteq \mathcal{H}$ which contains a halfplane h that maximizes $\Delta(S, h)$
 - find $h \in \mathcal{H}'$ with $\Delta(s, h)$ maximum.

Halfplane discrepancy

$$\Delta(S) = \sup_{h \in \mathcal{H}} \Delta(S, h).$$

Problem: is uncountable :-)

Strategy:

- choose candidate set $\mathcal{H}' \subseteq \mathcal{H}$ which contains a halfplane h that maximizes $\Delta(S, h)$
- find $h \in \mathcal{H}'$ with $\Delta(s, h)$ maximum.

Lemma. The set $\mathcal{H}' = \mathcal{H}_1 \cup \mathcal{H}_2$ contains a halfplane h that maximizes $|S \cap h|$, where

$$\mathcal{H}_1 =$$

$$\mathcal{H}_2 =$$

Halfplane discrepancy

$$\Delta(S) = \sup_{h \in \mathcal{H}} \Delta(S, h).$$

Problem: is uncountable :-)

Strategy:

- choose candidate set $\mathcal{H}' \subseteq \mathcal{H}$ which contains a halfplane h that maximizes $\Delta(S, h)$
- find $h \in \mathcal{H}'$ with $\Delta(s, h)$ maximum.

Lemma. The set $\mathcal{H}' = \mathcal{H}_1 \cup \mathcal{H}_2$ contains a halfplane h that maximizes $|S \cap h|$, where

$\mathcal{H}_1 = \{h \in \mathcal{H} : \{p\} := |\partial h \cap S| = 1 \text{ and } \mu(h) \text{ is locally extreme among all halfplanes that touch } p\}$,

$\mathcal{H}_2 =$

Halfplane discrepancy

$$\Delta(S) = \sup_{h \in \mathcal{H}} \Delta(S, h).$$

Problem: is uncountable :-)

Strategy:

- choose candidate set $\mathcal{H}' \subseteq \mathcal{H}$ which contains a halfplane h that maximizes $\Delta(S, h)$
- find $h \in \mathcal{H}'$ with $\Delta(s, h)$ maximum.

Lemma. The set $\mathcal{H}' = \mathcal{H}_1 \cup \mathcal{H}_2$ contains a halfplane h that maximizes $|S \cap h|$, where

$\mathcal{H}_1 = \{h \in \mathcal{H} : \{p\} := |\partial h \cap S| = 1 \text{ and } \mu(h) \text{ is locally extreme among all halfplanes that touch } p\}$,

$\mathcal{H}_2 = \{h \in \mathcal{H} : |\partial h \cap S| = 2\}$.

Halfplane discrepancy

$$\Delta(S) = \sup_{h \in \mathcal{H}} \Delta(S, h).$$

Problem: is uncountable :-)

Strategy:

- choose candidate set $\mathcal{H}' \subseteq \mathcal{H}$ which contains a halfplane h that maximizes $\Delta(S, h)$
- find $h \in \mathcal{H}'$ with $\Delta(s, h)$ maximum.

Lemma. The set $\mathcal{H}' = \mathcal{H}_1 \cup \mathcal{H}_2$ contains a halfplane h that maximizes $|S \cap h|$, where

$\mathcal{H}_1 = \{h \in \mathcal{H} : \{p\} := |\partial h \cap S| = 1 \text{ and } \mu(h) \text{ is locally extreme among all halfplanes that touch } p\}$,

$\mathcal{H}_2 = \{h \in \mathcal{H} : |\partial h \cap S| = 2\}$.

With $n = |S|$, we have $|\mathcal{H}_1| \in \Theta(n)$, $|\mathcal{H}_2| \in \Theta(n^2)$.

Halfplane discrepancy

$$\Delta(S) = \sup_{h \in \mathcal{H}} \Delta(S, h).$$

Problem: is uncountable :-)

Strategy:

- choose candidate set $\mathcal{H}' \subseteq \mathcal{H}$ which contains a halfplane h that maximizes $\Delta(S, h)$
- find $h \in \mathcal{H}'$ with $\Delta(s, h)$ maximum.

Lemma. The set $\mathcal{H}' = \mathcal{H}_1 \cup \mathcal{H}_2$ contains a halfplane h that maximizes $|S \cap h|$, where

$\mathcal{H}_1 = \{h \in \mathcal{H} : \{p\} := |\partial h \cap S| = 1 \text{ and } \mu(h) \text{ is locally extreme among all halfplanes that touch } p\}$,

$\mathcal{H}_2 = \{h \in \mathcal{H} : |\partial h \cap S| = 2\}$.

With $n = |S|$, we have $|\mathcal{H}_1| \in \Theta(n)$, $|\mathcal{H}_2| \in \Theta(n^2)$.

Thm. The halfplane discrepancy of a set of n pts in the unit square can be computed in $O(n^2)$ time.

TODO!

Idea

For each $h \in \mathcal{H}_1$, we can compute $\Delta(S, h)$ in $O(n)$ time (by brute force).

Idea

For each $h \in \mathcal{H}_1$, we can compute $\Delta(S, h)$ in $O(n)$ time (by brute force). $\Rightarrow O(n^2)$ time

Idea

For each $h \in \mathcal{H}_1$, we can compute $\Delta(S, h)$ in $O(n)$ time (by brute force). $\Rightarrow O(n^2)$ time

But how to compute the discrepancy of the $\Theta(n^2)$ many half planes in \mathcal{H}_2 in $O(n^2)$ total time?

Idea

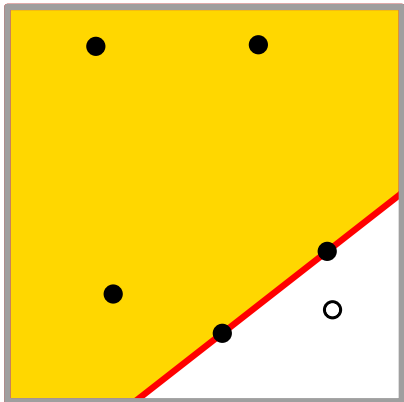
For each $h \in \mathcal{H}_1$, we can compute $\Delta(S, h)$ in $O(n)$ time (by brute force). $\Rightarrow O(n^2)$ time

But how to compute the discrepancy of the $\Theta(n^2)$ many half planes in \mathcal{H}_2 in $O(n^2)$ total time? In particular, we need $|h \cap S|$!

Idea

For each $h \in \mathcal{H}_1$, we can compute $\Delta(S, h)$ in $O(n)$ time (by brute force). $\Rightarrow O(n^2)$ time

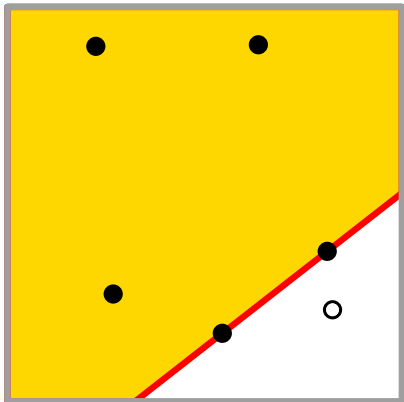
But how to compute the discrepancy of the $\Theta(n^2)$ many half planes in \mathcal{H}_2 in $O(n^2)$ total time? In particular, we need $|h \cap S|$!



Idea

For each $h \in \mathcal{H}_1$, we can compute $\Delta(S, h)$ in $O(n)$ time (by brute force). $\Rightarrow O(n^2)$ time

But how to compute the discrepancy of the $\Theta(n^2)$ many half planes in \mathcal{H}_2 in $O(n^2)$ total time? In particular, we need $|h \cap S|$!

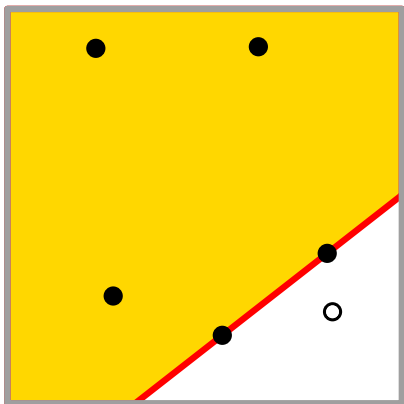


\longrightarrow
 duality \star

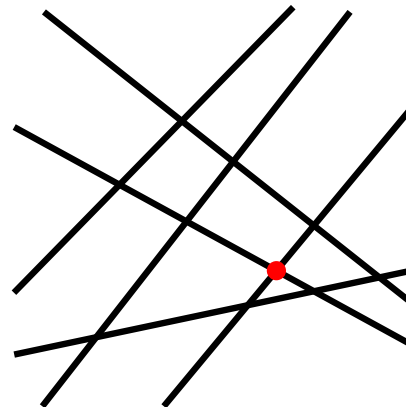
Idea

For each $h \in \mathcal{H}_1$, we can compute $\Delta(S, h)$ in $O(n)$ time (by brute force). $\Rightarrow O(n^2)$ time

But how to compute the discrepancy of the $\Theta(n^2)$ many half planes in \mathcal{H}_2 in $O(n^2)$ total time? In particular, we need $|h \cap S|$!



→
duality ★

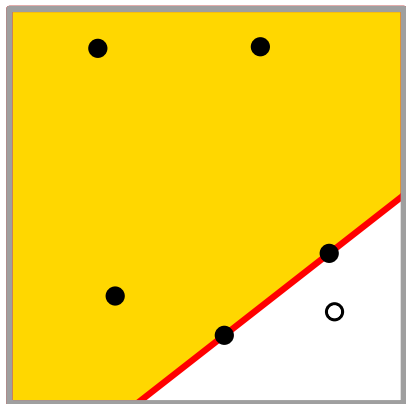


- Compute arrangement \mathcal{A} of the lines in S^*

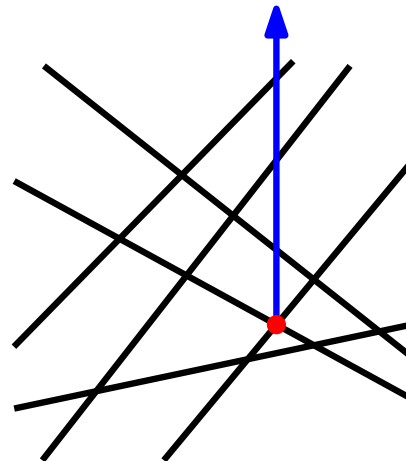
Idea

For each $h \in \mathcal{H}_1$, we can compute $\Delta(S, h)$ in $O(n)$ time (by brute force). $\Rightarrow O(n^2)$ time

But how to compute the discrepancy of the $\Theta(n^2)$ many half planes in \mathcal{H}_2 in $O(n^2)$ total time? In particular, we need $|h \cap S|$!



→
duality \star

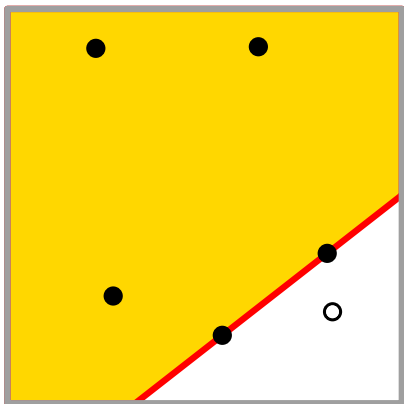


- Compute arrangement \mathcal{A} of the lines in S^*

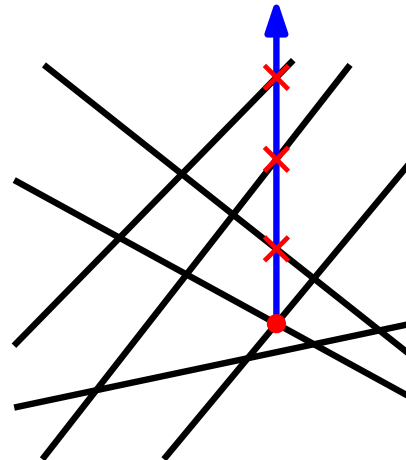
Idea

For each $h \in \mathcal{H}_1$, we can compute $\Delta(S, h)$ in $O(n)$ time (by brute force). $\Rightarrow O(n^2)$ time

But how to compute the discrepancy of the $\Theta(n^2)$ many half planes in \mathcal{H}_2 in $O(n^2)$ total time? In particular, we need $|h \cap S|$!



→
duality \star

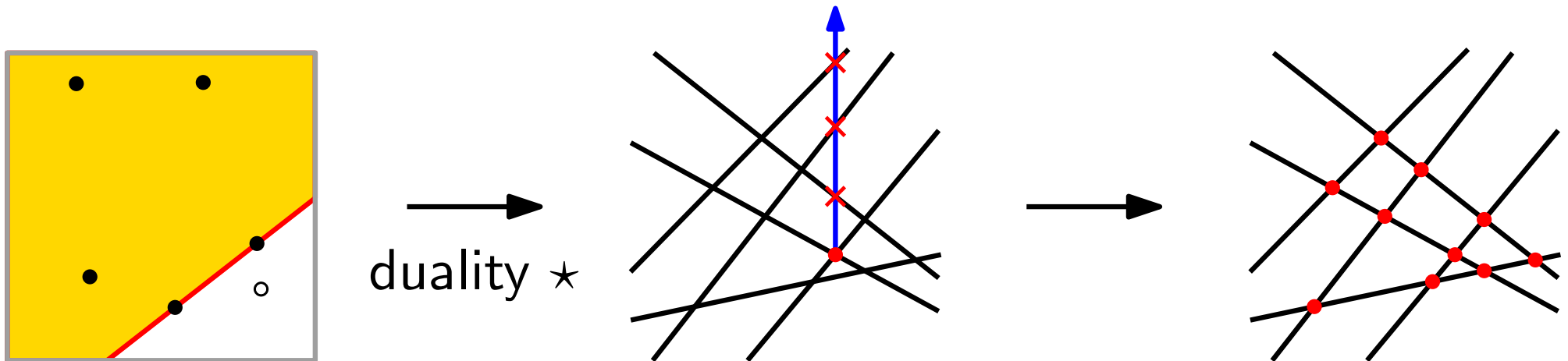


- Compute arrangement \mathcal{A} of the lines in S^*

Idea

For each $h \in \mathcal{H}_1$, we can compute $\Delta(S, h)$ in $O(n)$ time (by brute force). $\Rightarrow O(n^2)$ time

But how to compute the discrepancy of the $\Theta(n^2)$ many half planes in \mathcal{H}_2 in $O(n^2)$ total time? In particular, we need $|h \cap S|$!

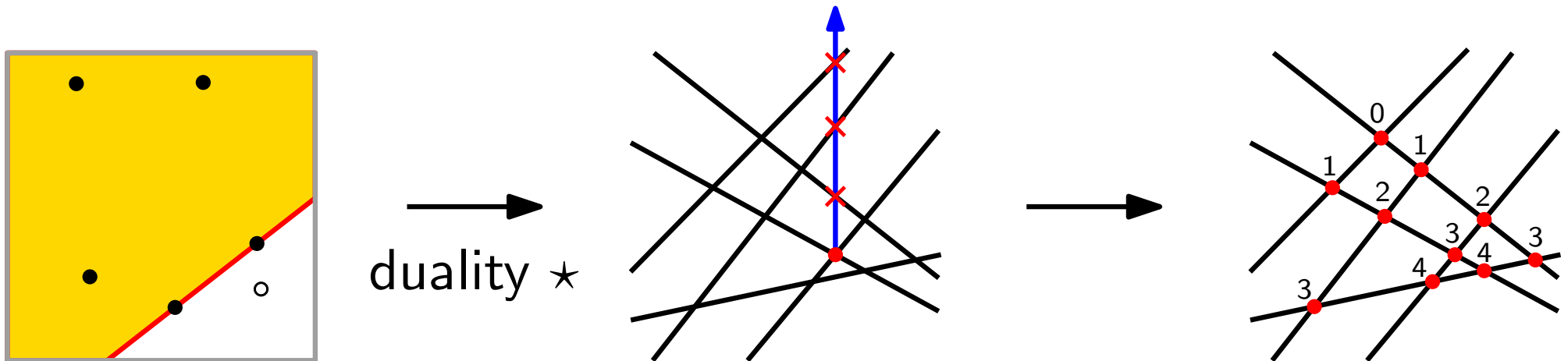


- Compute arrangement \mathcal{A} of the lines in S^*
- For each vertex h^* of \mathcal{A} , compute its *depth* in \mathcal{A} .

Idea

For each $h \in \mathcal{H}_1$, we can compute $\Delta(S, h)$ in $O(n)$ time (by brute force). $\Rightarrow O(n^2)$ time

But how to compute the discrepancy of the $\Theta(n^2)$ many half planes in \mathcal{H}_2 in $O(n^2)$ total time? In particular, we need $|h \cap S|$!

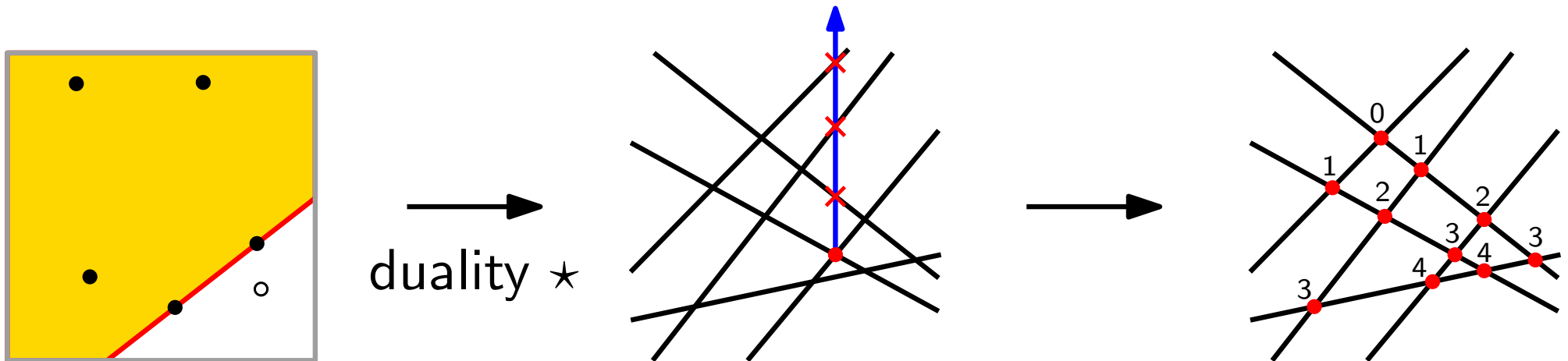


- Compute arrangement \mathcal{A} of the lines in S^*
- For each vertex h^* of \mathcal{A} , compute its *depth* in \mathcal{A} .

Idea

For each $h \in \mathcal{H}_1$, we can compute $\Delta(S, h)$ in $O(n)$ time (by brute force). $\Rightarrow O(n^2)$ time

But how to compute the discrepancy of the $\Theta(n^2)$ many half planes in \mathcal{H}_2 in $O(n^2)$ total time? In particular, we need $|h \cap S|$!

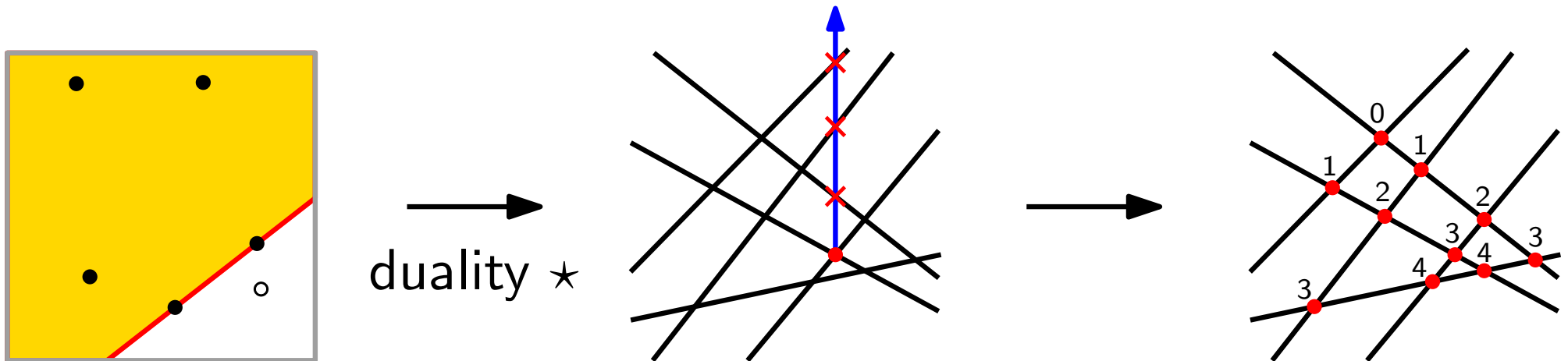


- Compute arrangement \mathcal{A} of the lines in S^*
- For each vertex h^* of \mathcal{A} , compute its *depth* in \mathcal{A} , that is, the number of lines that lie strictly above h^*

Idea

For each $h \in \mathcal{H}_1$, we can compute $\Delta(S, h)$ in $O(n)$ time (by brute force). $\Rightarrow O(n^2)$ time

But how to compute the discrepancy of the $\Theta(n^2)$ many half planes in \mathcal{H}_2 in $O(n^2)$ total time? In particular, we need $|h \cap S|$!

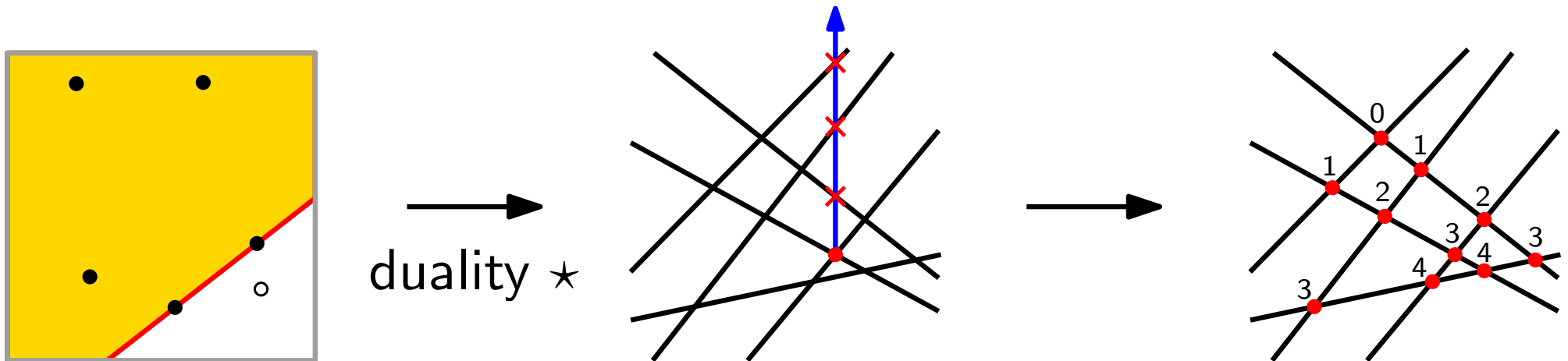


- Compute arrangement \mathcal{A} of the lines in S^*
- For each vertex h^* of \mathcal{A} , compute its *depth* in \mathcal{A} , that is, the number of lines that lie strictly above h^* ($= |S \cap \text{int}(h)|$)

Idea

For each $h \in \mathcal{H}_1$, we can compute $\Delta(S, h)$ in $O(n)$ time (by brute force). $\Rightarrow O(n^2)$ time

But how to compute the discrepancy of the $\Theta(n^2)$ many half planes in \mathcal{H}_2 in $O(n^2)$ total time? In particular, we need $|h \cap S|$!



- Compute arrangement \mathcal{A} of the lines in S^*
- For each vertex h^* of \mathcal{A} , compute its *depth* in \mathcal{A} , that is, the number of lines that lie strictly above h^* ($= |S \cap \text{int}(h)|$) and the degree of h^* in \mathcal{A} ($= 2|S \cap \partial h|$)

Arrangements of Lines

How can we compute the arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of n lines in $O(n^2)$ time?

Arrangements of Lines

How can we compute the arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of n lines in $O(n^2)$ time? – This would be optimal!

Arrangements of Lines

How can we compute the arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of n lines in $O(n^2)$ time? – This would be optimal!

- line-sweep algorithm?

Arrangements of Lines

How can we compute the arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of n lines in $O(n^2)$ time? – This would be optimal!

- line-sweep algorithm? $O(n^2 \log n)$ time...

Arrangements of Lines

How can we compute the arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of n lines in $O(n^2)$ time? – This would be optimal!

- line-sweep algorithm? $O(n^2 \log n)$ time...
- incrementally?

Arrangements of Lines

How can we compute the arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of n lines in $O(n^2)$ time? – This would be optimal!

- line-sweep algorithm? $O(n^2 \log n)$ time...

- incrementally?

If we can insert line i in $O(i)$ time, we get a runtime of $O(n^2)$.

Arrangements of Lines

How can we compute the arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of n lines in $O(n^2)$ time? – This would be optimal!

- line-sweep algorithm? $O(n^2 \log n)$ time...

- incrementally?

If we can insert line i in $O(i)$ time, we get a runtime of $O(n^2)$.

– compute doubly-connected edge list (DCEL) of $\mathcal{A}(\mathcal{L})$

Arrangements of Lines

How can we compute the arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of n lines in $O(n^2)$ time? – This would be optimal!

- line-sweep algorithm? $O(n^2 \log n)$ time...

- incrementally?

If we can insert line i in $O(i)$ time, we get a runtime of $O(n^2)$.

- compute doubly-connected edge list (DCEL) of $\mathcal{A}(\mathcal{L})$

- use sufficiently large rectangle R

Arrangements of Lines

How can we compute the arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of n lines in $O(n^2)$ time? – This would be optimal!

- line-sweep algorithm? $O(n^2 \log n)$ time...

- incrementally?

If we can insert line i in $O(i)$ time, we get a runtime of $O(n^2)$.

- compute doubly-connected edge list (DCEL) of $\mathcal{A}(\mathcal{L})$

- use sufficiently large rectangle R (how?)

Arrangements of Lines

How can we compute the arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of n lines in $O(n^2)$ time? – This would be optimal!

- line-sweep algorithm? $O(n^2 \log n)$ time...

- incrementally?

If we can insert line i in $O(i)$ time, we get a runtime of $O(n^2)$.

- compute doubly-connected edge list (DCEL) of $\mathcal{A}(\mathcal{L})$

- use sufficiently large rectangle R (how?) \Rightarrow edges bounded

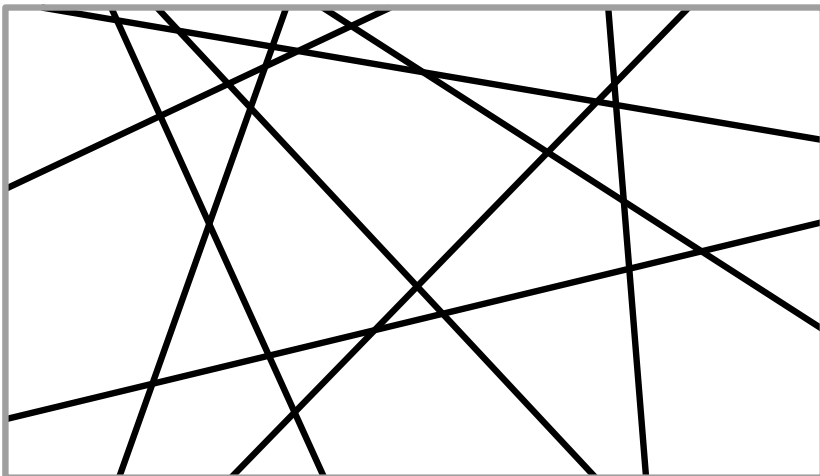
Arrangements of Lines

How can we compute the arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of n lines in $O(n^2)$ time? – This would be optimal!

- line-sweep algorithm? $O(n^2 \log n)$ time...
- incrementally?

If we can insert line i in $O(i)$ time, we get a runtime of $O(n^2)$.

- compute doubly-connected edge list (DCEL) of $\mathcal{A}(\mathcal{L})$
- use sufficiently large rectangle R (how?) \Rightarrow edges bounded



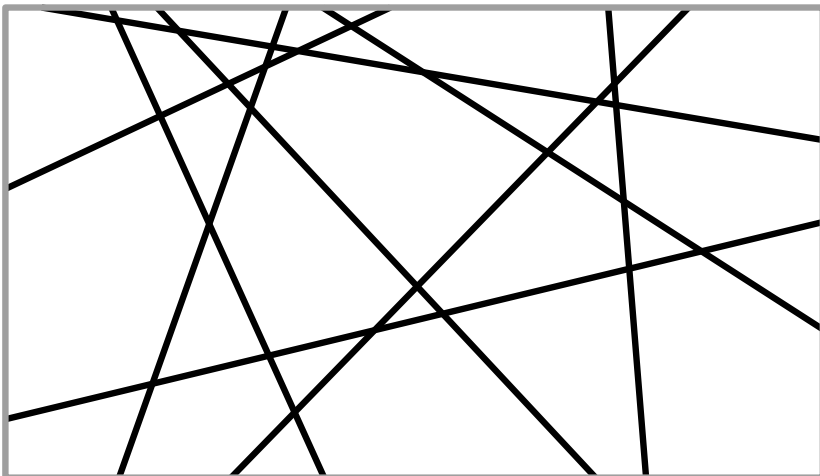
Arrangements of Lines

How can we compute the arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of n lines in $O(n^2)$ time? – This would be optimal!

- line-sweep algorithm? $O(n^2 \log n)$ time...
- incrementally?

If we can insert line i in $O(i)$ time, we get a runtime of $O(n^2)$.

- compute doubly-connected edge list (DCEL) of $\mathcal{A}(\mathcal{L})$
- use sufficiently large rectangle R (how?) \Rightarrow edges bounded



Effort for inserting ℓ_i is bounded by the complexity of the *zone* of ℓ_i .

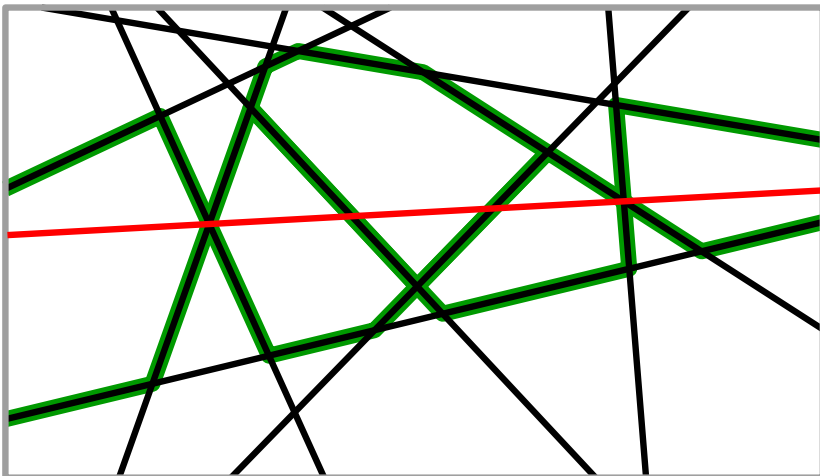
Arrangements of Lines

How can we compute the arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of n lines in $O(n^2)$ time? – This would be optimal!

- line-sweep algorithm? $O(n^2 \log n)$ time...
- incrementally?

If we can insert line i in $O(i)$ time, we get a runtime of $O(n^2)$.

- compute doubly-connected edge list (DCEL) of $\mathcal{A}(\mathcal{L})$
- use sufficiently large rectangle R (how?) \Rightarrow edges bounded



Effort for inserting ℓ_i is bounded by the complexity of the *zone* of ℓ_i

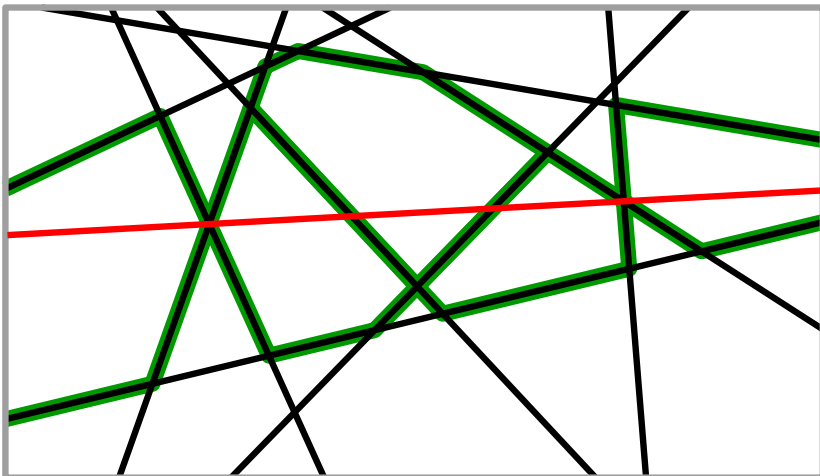
Arrangements of Lines

How can we compute the arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of n lines in $O(n^2)$ time? – This would be optimal!

- line-sweep algorithm? $O(n^2 \log n)$ time...
- incrementally?

If we can insert line i in $O(i)$ time, we get a runtime of $O(n^2)$.

- compute doubly-connected edge list (DCEL) of $\mathcal{A}(\mathcal{L})$
- use sufficiently large rectangle R (how?) \Rightarrow edges bounded



Effort for inserting ℓ_i is bounded by the complexity of the *zone* of ℓ_i , that is, the set of faces whose *closure* intersects ℓ_i

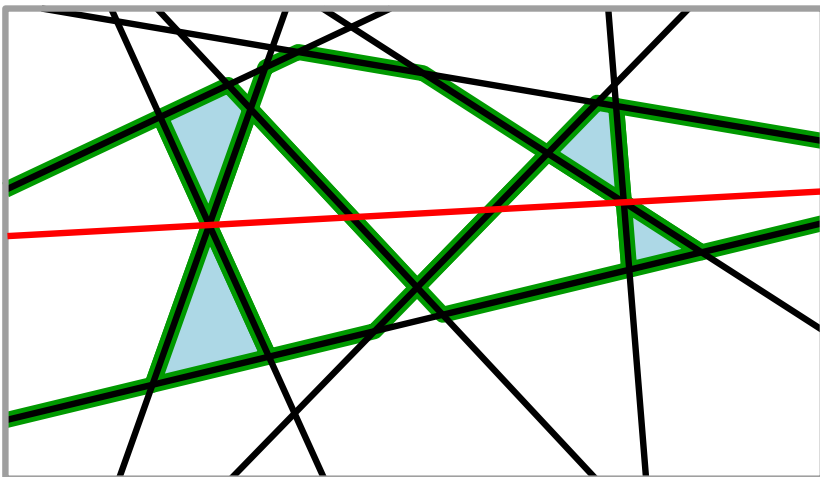
Arrangements of Lines

How can we compute the arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of n lines in $O(n^2)$ time? – This would be optimal!

- line-sweep algorithm? $O(n^2 \log n)$ time...
- incrementally?

If we can insert line i in $O(i)$ time, we get a runtime of $O(n^2)$.

- compute doubly-connected edge list (DCEL) of $\mathcal{A}(\mathcal{L})$
- use sufficiently large rectangle R (how?) \Rightarrow edges bounded



Effort for inserting ℓ_i is bounded by the complexity of the *zone* of ℓ_i , that is, the set of faces whose *closure* intersects ℓ_i

The Zone Theorem

Theorem. The complexity of the zone of a line ℓ in an arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of m lines is $O(m)$.

The Zone Theorem

Theorem. The complexity of the zone of a line ℓ in an arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of m lines is $O(m)$.

Proof. W.l.o.g. $\ell = x$ -axis.

The Zone Theorem

Theorem. The complexity of the zone of a line ℓ in an arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of m lines is $O(m)$.

Proof. W.l.o.g. $\ell = x$ -axis.
We (initially) assume that no line in \mathcal{L} is horizontal.

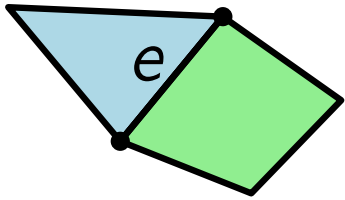
The Zone Theorem

Theorem. The complexity of the zone of a line ℓ in an arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of m lines is $O(m)$.

Proof. W.l.o.g. $\ell = x$ -axis.

We (initially) assume that no line in \mathcal{L} is horizontal.

Every edge e of \mathcal{A} bounds two faces.



The Zone Theorem

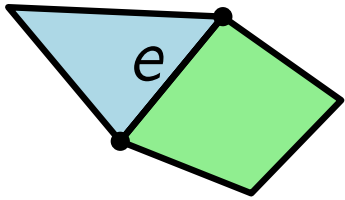
Theorem. The complexity of the zone of a line ℓ in an arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of m lines is $O(m)$.

Proof. W.l.o.g. $\ell = x$ -axis.

We (initially) assume that no line in \mathcal{L} is horizontal.

Every edge e of \mathcal{A} bounds two faces.

We say that e *left-bounds* the face to its **right**.



The Zone Theorem

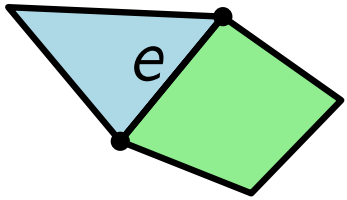
Theorem. The complexity of the zone of a line ℓ in an arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of m lines is $O(m)$.

Proof. W.l.o.g. $\ell = x$ -axis.

We (initially) assume that no line in \mathcal{L} is horizontal.

Every edge e of \mathcal{A} bounds two faces.

We say that e *left-bounds* the face to its **right**.



Show: The number of left-bounding edges in $\text{zone}(\ell)$ is $\leq 5m$.

The Zone Theorem

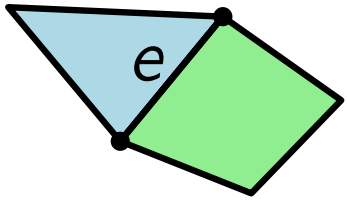
Theorem. The complexity of the zone of a line ℓ in an arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of m lines is $O(m)$.

Proof. W.l.o.g. $\ell = x$ -axis.

We (initially) assume that no line in \mathcal{L} is horizontal.

Every edge e of \mathcal{A} bounds two faces.

We say that e *left-bounds* the face to its **right**.



Show: The number of left-bounding edges in $\text{zone}(\ell)$ is $\leq 5m$.

We use induction over m .

The Zone Theorem

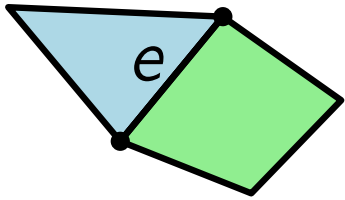
Theorem. The complexity of the zone of a line ℓ in an arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of m lines is $O(m)$.

Proof. W.l.o.g. $\ell = x$ -axis.

We (initially) assume that no line in \mathcal{L} is horizontal.

Every edge e of \mathcal{A} bounds two faces.

We say that e *left-bounds* the face to its **right**.



Show: The number of left-bounding edges in $\text{zone}(\ell)$ is $\leq 5m$.

We use induction over m . The base case ($m = 1$) is trivial.

The Zone Theorem

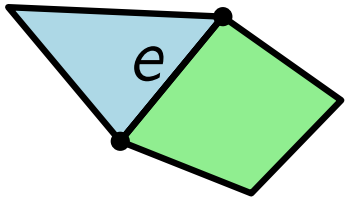
Theorem. The complexity of the zone of a line ℓ in an arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of m lines is $O(m)$.

Proof. W.l.o.g. $\ell = x$ -axis.

We (initially) assume that no line in \mathcal{L} is horizontal.

Every edge e of \mathcal{A} bounds two faces.

We say that e *left-bounds* the face to its **right**.



Show: The number of left-bounding edges in $\text{zone}(\ell)$ is $\leq 5m$.

We use induction over m . The base case ($m = 1$) is trivial.

Let $m > 1$.

The Zone Theorem

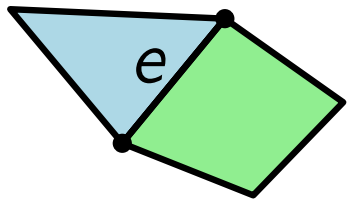
Theorem. The complexity of the zone of a line ℓ in an arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of m lines is $O(m)$.

Proof. W.l.o.g. $\ell = x$ -axis.

We (initially) assume that no line in \mathcal{L} is horizontal.

Every edge e of \mathcal{A} bounds two faces.

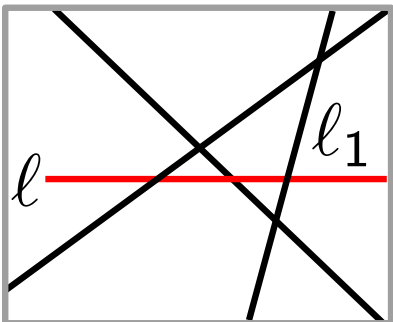
We say that e *left-bounds* the face to its **right**.



Show: The number of left-bounding edges in $\text{zone}(\ell)$ is $\leq 5m$.

We use induction over m . The base case ($m = 1$) is trivial.

Let $m > 1$. Let ℓ_1 be the rightmost line that is intersected by ℓ .



The Zone Theorem

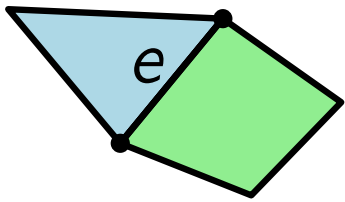
Theorem. The complexity of the zone of a line ℓ in an arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of m lines is $O(m)$.

Proof. W.l.o.g. $\ell = x$ -axis.

We (initially) assume that no line in \mathcal{L} is horizontal.

Every edge e of \mathcal{A} bounds two faces.

We say that e *left-bounds* the face to its **right**.

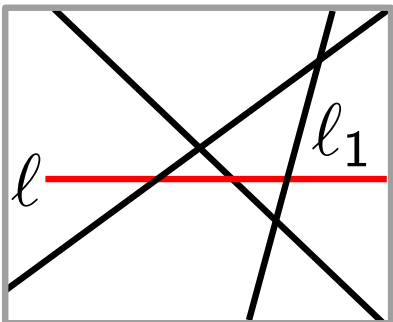


Show: The number of left-bounding edges in $\text{zone}(\ell)$ is $\leq 5m$.

We use induction over m . The base case ($m = 1$) is trivial.

Let $m > 1$. Let ℓ_1 be the rightmost line that is intersected by ℓ .

Induction hypothesis \Rightarrow



The Zone Theorem

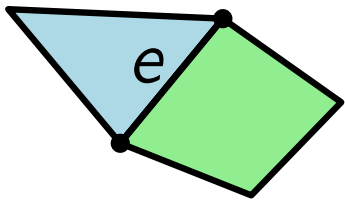
Theorem. The complexity of the zone of a line ℓ in an arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of m lines is $O(m)$.

Proof. W.l.o.g. $\ell = x$ -axis.

We (initially) assume that no line in \mathcal{L} is horizontal.

Every edge e of \mathcal{A} bounds two faces.

We say that e *left-bounds* the face to its **right**.

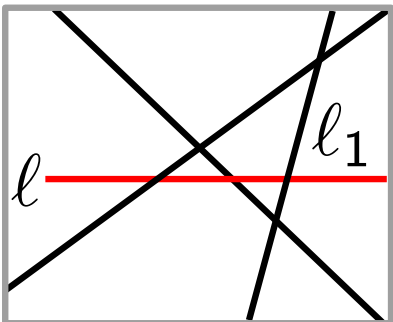


Show: The number of left-bounding edges in $\text{zone}(\ell)$ is $\leq 5m$.

We use induction over m . The base case ($m = 1$) is trivial.

Let $m > 1$. Let ℓ_1 be the rightmost line that is intersected by ℓ .

Induction hypothesis \Rightarrow $\text{zone}(\ell)$ in $\mathcal{A}(\mathcal{L} \setminus \{\ell_1\})$ has $\leq 5(m - 1)$ left-bounding edges.



The Zone Theorem

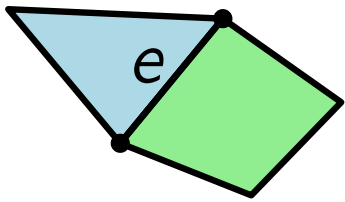
Theorem. The complexity of the zone of a line ℓ in an arrangement $\mathcal{A}(\mathcal{L})$ of a set \mathcal{L} of m lines is $O(m)$.

Proof. W.l.o.g. $\ell = x$ -axis.

We (initially) assume that no line in \mathcal{L} is horizontal.

Every edge e of \mathcal{A} bounds two faces.

We say that e *left-bounds* the face to its **right**.



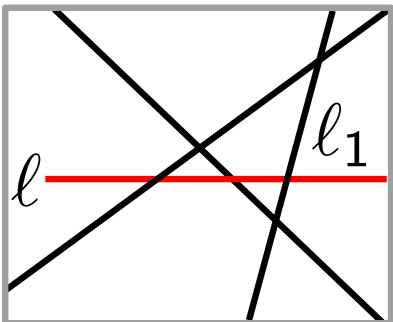
Show: The number of left-bounding edges in $\text{zone}(\ell)$ is $\leq 5m$.

We use induction over m . The base case ($m = 1$) is trivial.

Let $m > 1$. Let ℓ_1 be the rightmost line that is intersected by ℓ .

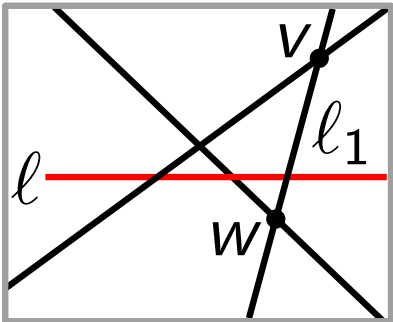
Induction hypothesis \Rightarrow $\text{zone}(\ell)$ in $\mathcal{A}(\mathcal{L} \setminus \{\ell_1\})$ has $\leq 5(m - 1)$ left-bounding edges.

Now we re-insert ℓ_1 into $\mathcal{A}(\mathcal{L} \setminus \{\ell_1\})$.



Proof cont'd

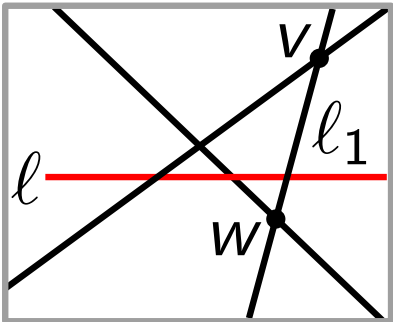
Let v be the first intersection pt on l_1 above l .
 w below (if such pts exist)



Proof cont'd

Let v be the first intersection pt on l_1 above l .
 w below (if such pts exist)

$\Rightarrow \overline{vw}$ is a new left-bounding edge.

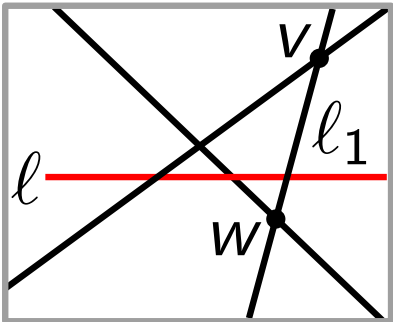


Proof cont'd

Let v be the first intersection pt on l_1 above l .
 w below (if such pts exist)

$\Rightarrow \overline{vw}$ is a new left-bounding edge.

Additionally, l_1 splits existing left-bounding edges in v and w .



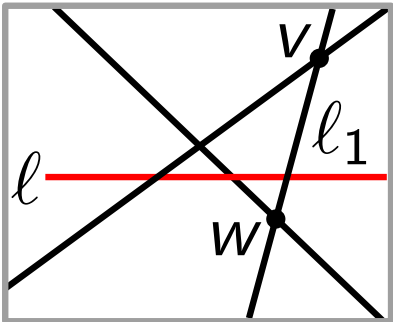
Proof cont'd

Let v be the first intersection pt on l_1 above w . (if such pts exist)
 below

$\Rightarrow \overline{vw}$ is a new left-bounding edge.

Additionally, l_1 splits existing left-bounding edges in v and w .

$\Rightarrow \#$ of left-bounding edges increases by ≤ 3 (less if v or w don't exist).



Proof cont'd

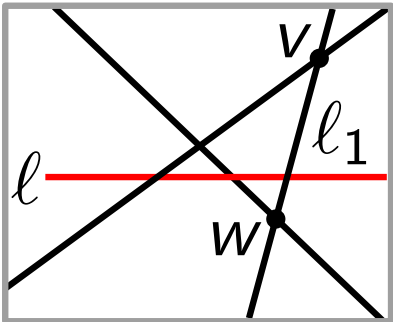
Let v be the first intersection pt on l_1 above l .
 w below (if such pts exist)

$\Rightarrow \overline{vw}$ is a new left-bounding edge.

Additionally, l_1 splits existing left-bounding edges in v and w .

$\Rightarrow \#$ of left-bounding edges increases by ≤ 3 (less if v or w don't exist).

If l_1 is not unique, let l_1 be an arbitrary line through the rightmost intersection pt on l .



Proof cont'd

Let v be the first intersection pt on l_1 above l .
 w below (if such pts exist)

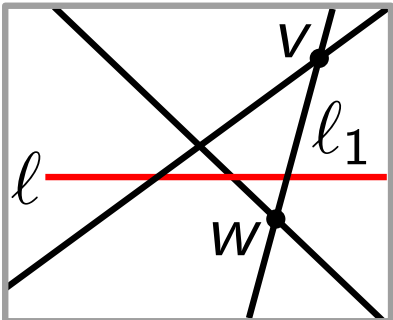
$\Rightarrow \overline{vw}$ is a new left-bounding edge.

Additionally, l_1 splits existing left-bounding edges in v and w .

$\Rightarrow \#$ of left-bounding edges increases by ≤ 3 (less if v or w don't exist).

If l_1 is not unique, let l_1 be an arbitrary line through the rightmost intersection pt on l .

$\Rightarrow l_1$ causes ≤ 5 new left-bounding edges (argue similarly as above)



Proof cont'd

Let v be the first intersection pt on l_1 above l .
 w below (if such pts exist)

$\Rightarrow \overline{vw}$ is a new left-bounding edge.

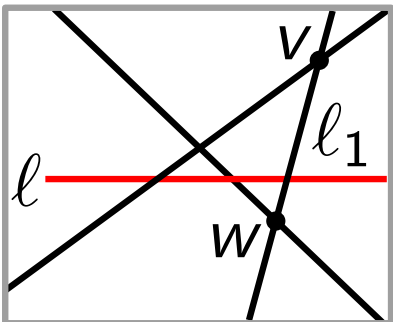
Additionally, l_1 splits existing left-bounding edges in v and w .

$\Rightarrow \#$ of left-bounding edges increases by ≤ 3 (less if v or w don't exist).

If l_1 is not unique, let l_1 be an arbitrary line through the rightmost intersection pt on l .

$\Rightarrow l_1$ causes ≤ 5 new left-bounding edges (argue similarly as above)

\Rightarrow total $\#$ left-bounding edges $\leq 5(m - 1) + 5 = 5m$.



Proof cont'd

Let v be the first intersection pt on l_1 above l .
 w below (if such pts exist)

$\Rightarrow \overline{vw}$ is a new left-bounding edge.

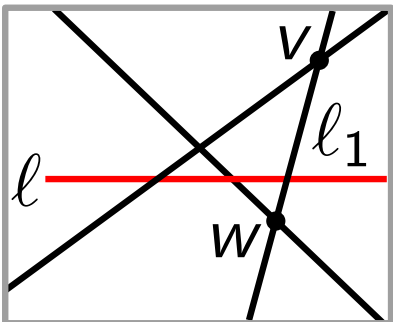
Additionally, l_1 splits existing left-bounding edges in v and w .

$\Rightarrow \#$ of left-bounding edges increases by ≤ 3 (less if v or w don't exist).

If l_1 is not unique, let l_1 be an arbitrary line through the rightmost intersection pt on l .

$\Rightarrow l_1$ causes ≤ 5 new left-bounding edges (argue similarly as above)

\Rightarrow total $\#$ left-bounding edges $\leq 5(m - 1) + 5 = 5m$.



If lines in \mathcal{L} are horizontal, then slightly rotating them only increases the complexity of the zone.

Proof cont'd

Let v be the first intersection pt on l_1 above l .
 w below (if such pts exist)

$\Rightarrow \overline{vw}$ is a new left-bounding edge.

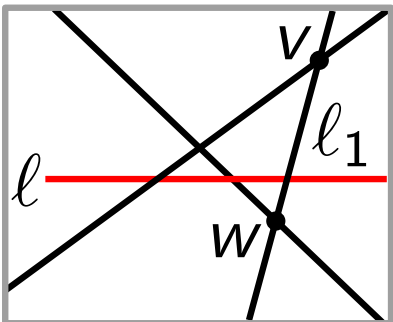
Additionally, l_1 splits existing left-bounding edges in v and w .

$\Rightarrow \#$ of left-bounding edges increases by ≤ 3 (less if v or w don't exist).

If l_1 is not unique, let l_1 be an arbitrary line through the rightmost intersection pt on l .

$\Rightarrow l_1$ causes ≤ 5 new left-bounding edges (argue similarly as above)

\Rightarrow total $\#$ left-bounding edges $\leq 5(m - 1) + 5 = 5m$.



If lines in \mathcal{L} are horizontal, then slightly rotating them only increases the complexity of the zone.

\Rightarrow Our above bound holds in this case, too.



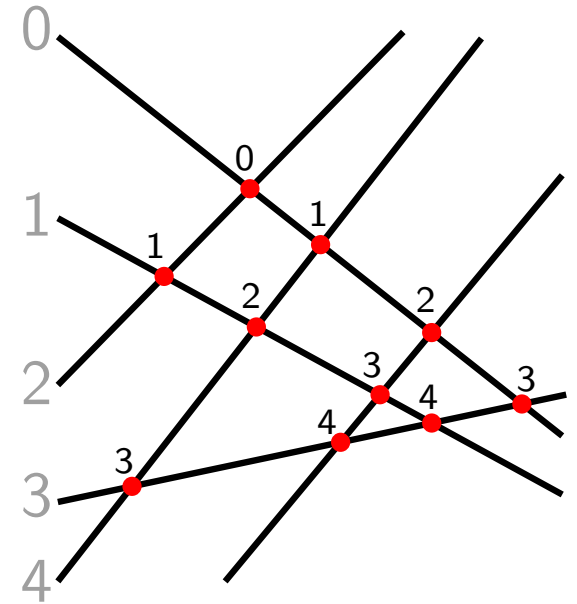
Back to Discrepancy

Theorem. Given a set \mathcal{L} of n lines, we can compute a DCEL of the arrangement induced by \mathcal{L} in $O(n^2)$ time.

Back to Discrepancy

Theorem. Given a set \mathcal{L} of n lines, we can compute a DCEL of the arrangement induced by \mathcal{L} in $O(n^2)$ time.

It remains to compute, for each vertex of the arrangement, its depth (and degree).

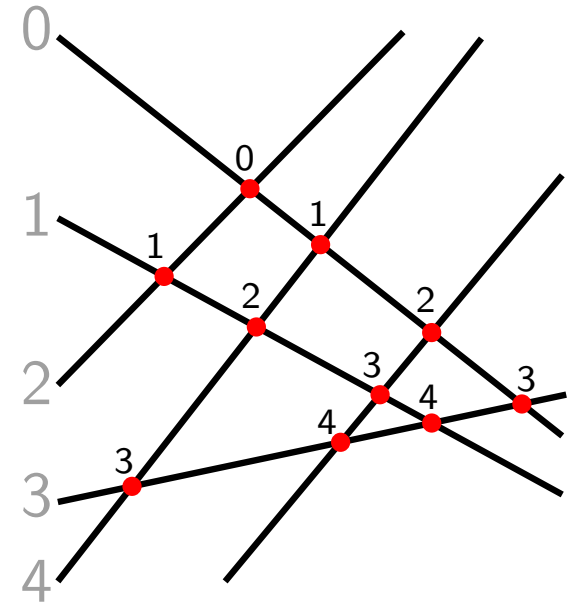


Back to Discrepancy

Theorem. Given a set \mathcal{L} of n lines, we can compute a DCEL of the arrangement induced by \mathcal{L} in $O(n^2)$ time.

It remains to compute, for each vertex of the arrangement, its depth (and degree).

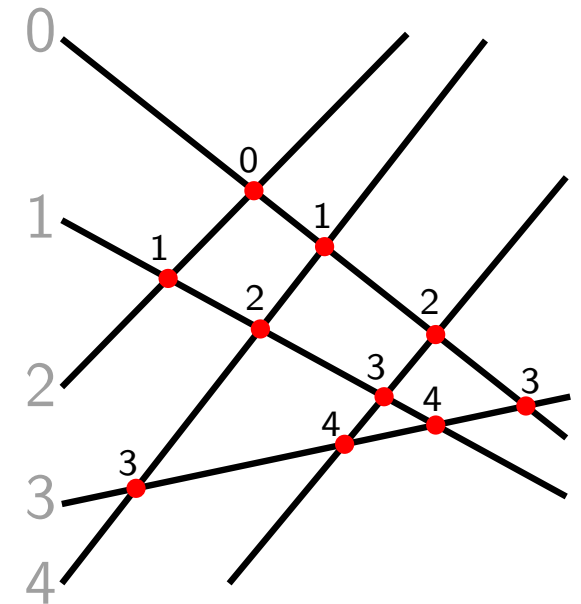
- Sort lines in S^* according to slope



Back to Discrepancy

Theorem. Given a set \mathcal{L} of n lines, we can compute a DCEL of the arrangement induced by \mathcal{L} in $O(n^2)$ time.

It remains to compute, for each vertex of the arrangement, its depth (and degree).

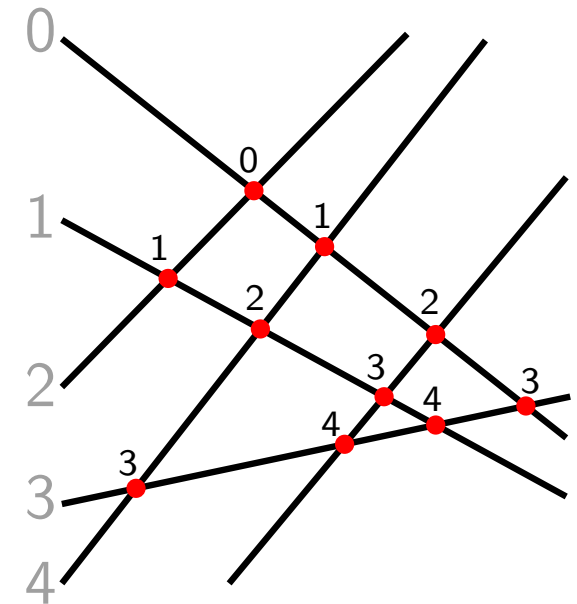


- Sort lines in S^* according to slope
 \Rightarrow depth at $x = -\infty$ corresponds to rank in this order (minus 1)

Back to Discrepancy

Theorem. Given a set \mathcal{L} of n lines, we can compute a DCEL of the arrangement induced by \mathcal{L} in $O(n^2)$ time.

It remains to compute, for each vertex of the arrangement, its depth (and degree).

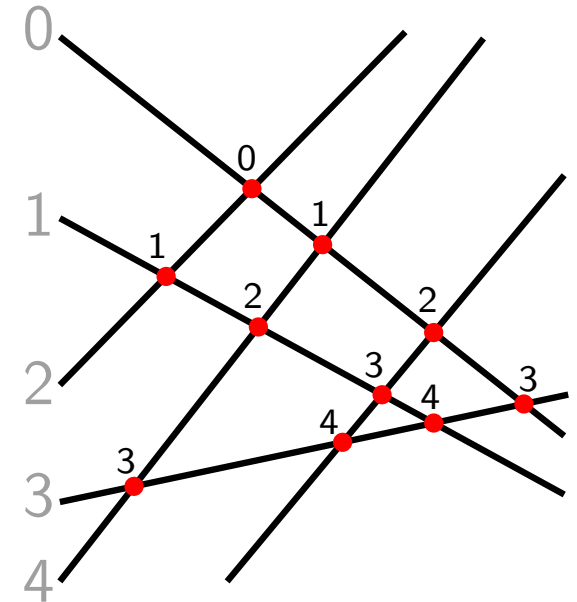


- Sort lines in S^* according to slope
 \Rightarrow depth at $x = -\infty$ corresponds to rank in this order (minus 1)
- Traverse each line from left to right.

Back to Discrepancy

Theorem. Given a set \mathcal{L} of n lines, we can compute a DCEL of the arrangement induced by \mathcal{L} in $O(n^2)$ time.

It remains to compute, for each vertex of the arrangement, its depth (and degree).



- Sort lines in S^* according to slope
 \Rightarrow depth at $x = -\infty$ corresponds to rank in this order (minus 1)
- Traverse each line from left to right.
 Adjust its depth by $+1$ (-1) if the intersecting line comes from below (above).