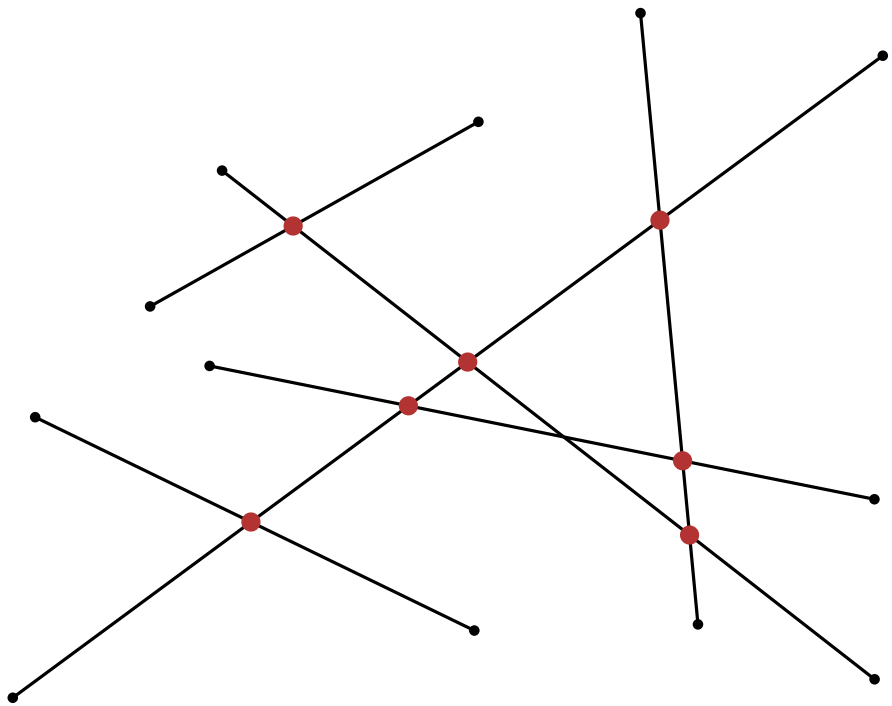
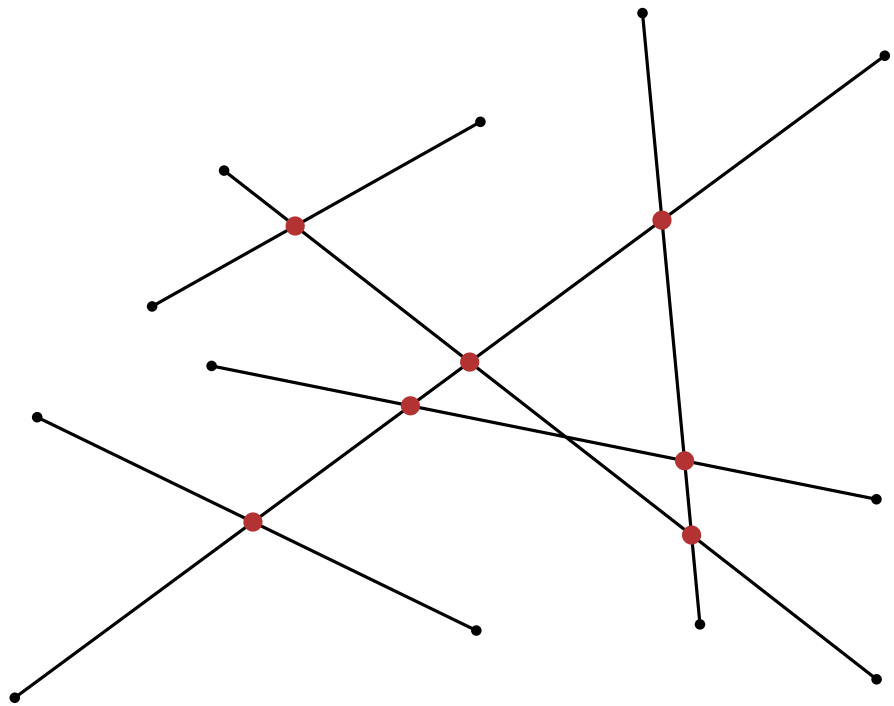
Linesegment Intersection

Problem: Given a set of n line segments in $\mathbb{R}^2$, compute all intersection points.



Linesegment Intersection

Problem: Given a set of n line segments in $\mathbb{R}^2$, compute all intersection points.



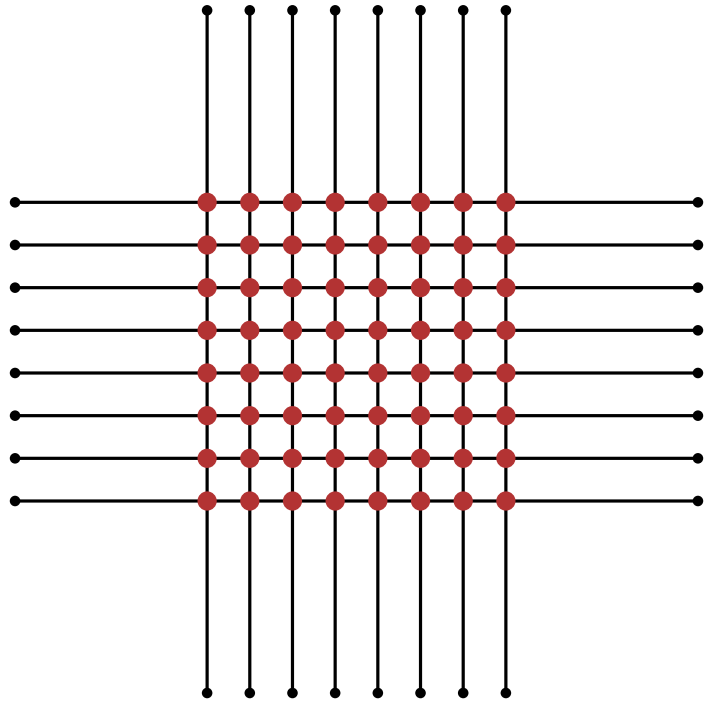
Easy Algorithm:

For each pair of segments check if they intersect

Running time: $O(n^2)$

Linesegment Intersection

Problem: Given a set of n line segments in $\mathbb{R}^2$, compute all intersection points.



Easy Algorithm:

For each pair of segments check if they intersect

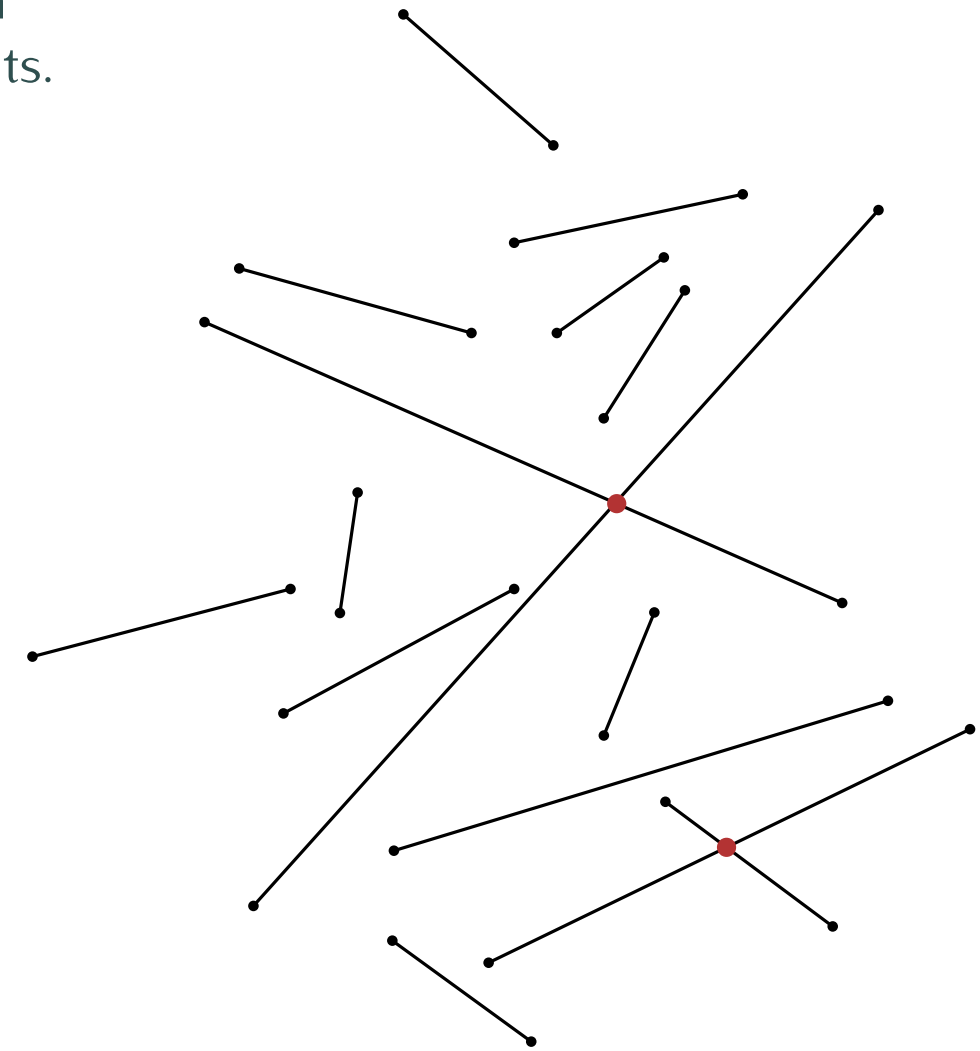
Running time: $O(n^2)$

Optimal in the worst case

Linesegment Intersection

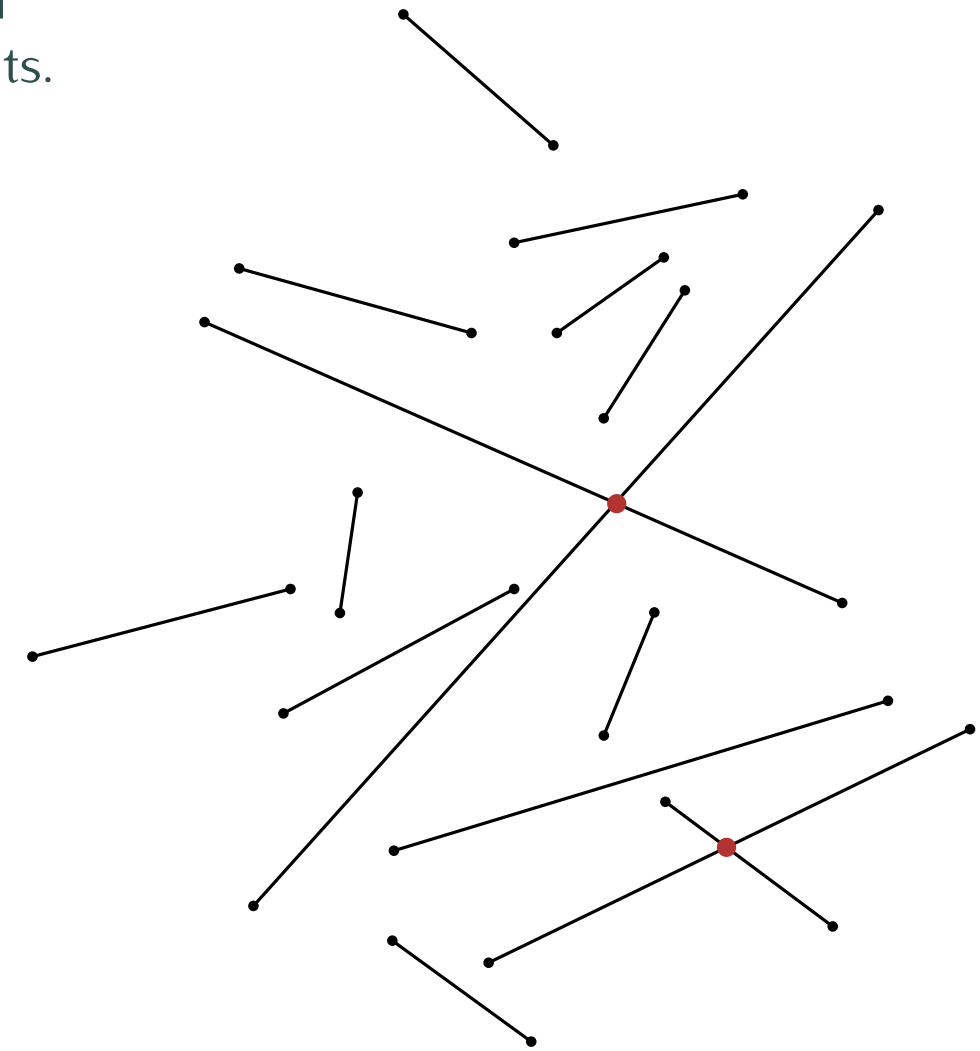
Problem: Given a set of n line segments in $\mathbb{R}^2$, compute all intersection points.

What if there are only few, say k, intersections?



Linesegment Intersection

Problem: Given a set of n line segments in $\mathbb{R}^2$, compute all intersection points.

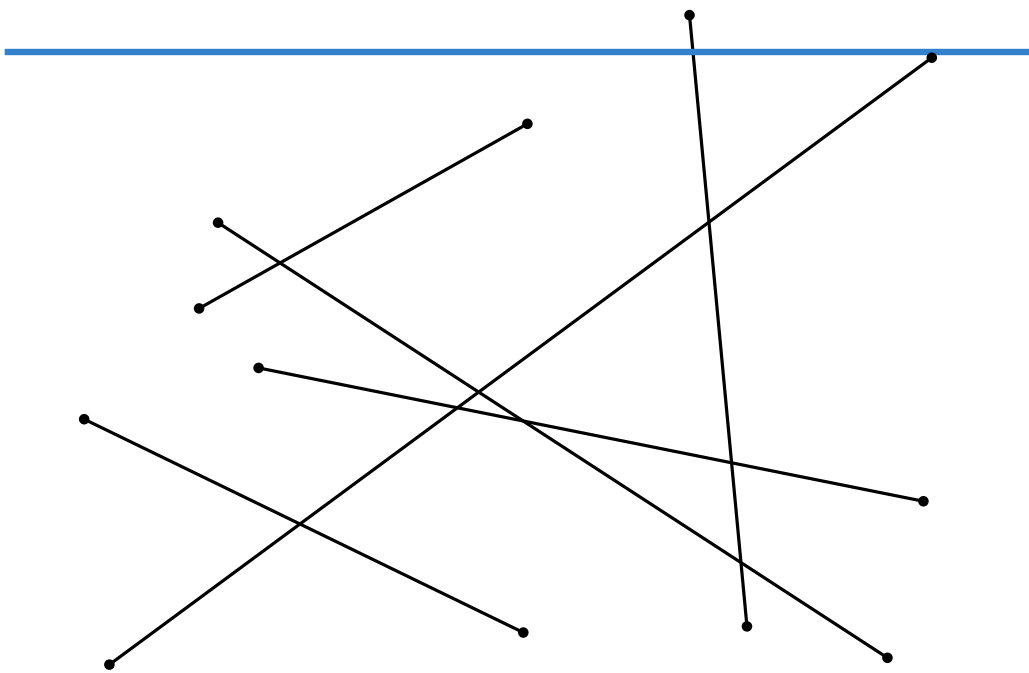


What if there are only few, say k, intersections?

We can compute all k intersections in $O((n+k)\log n)$ time, using a sweep-line algorithm.

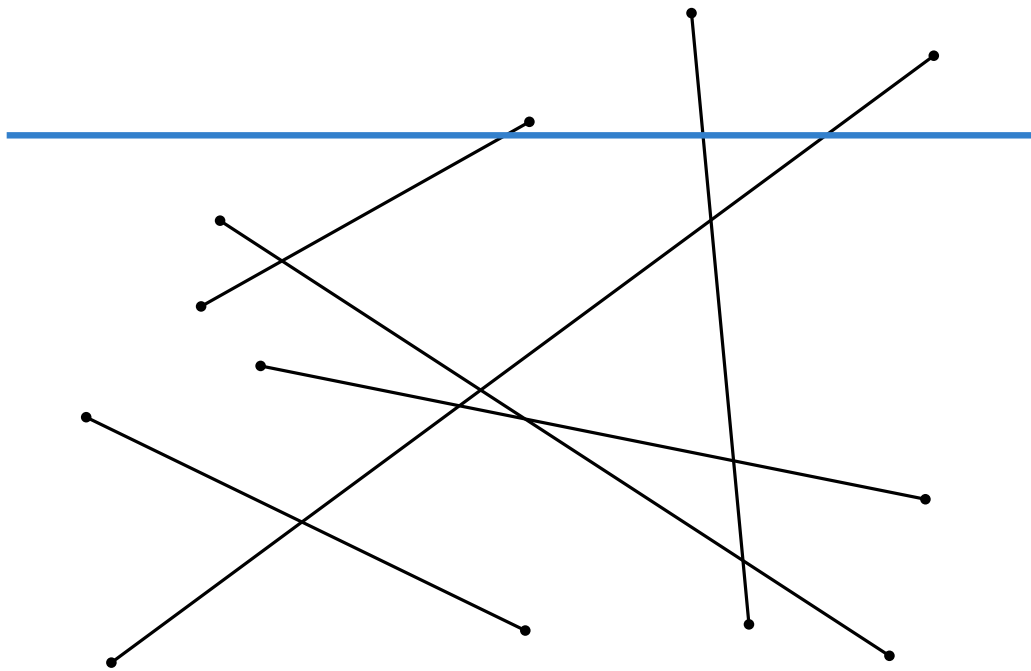
Sweep line paradigm

Idea: Sweep a (horizontal) line ℓ over the plane, make sure that all intersection points above the sweep line have been reported



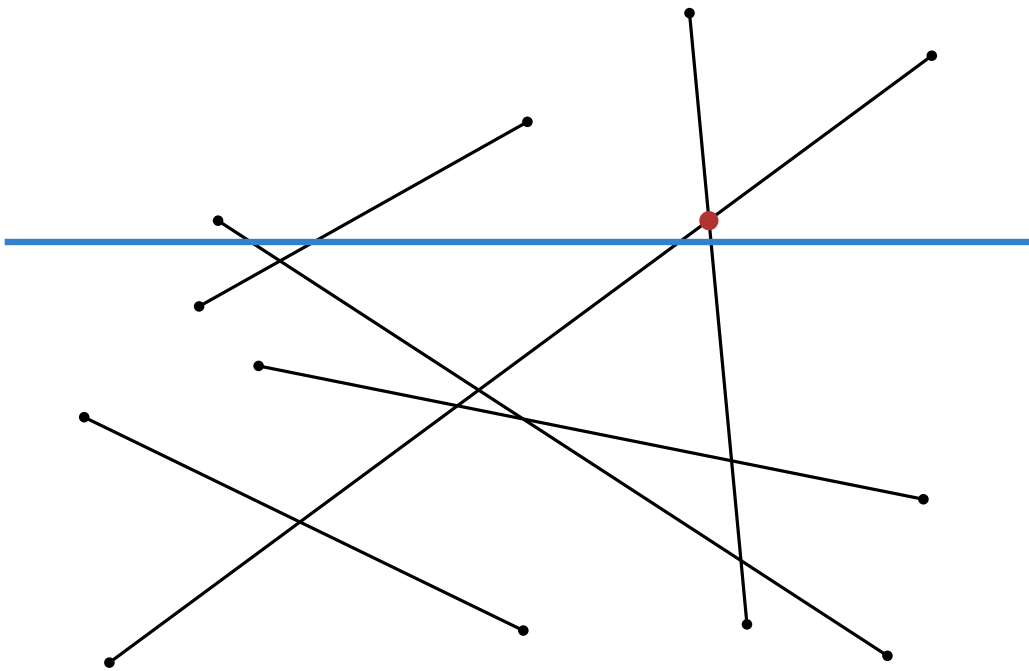
Sweep line paradigm

Idea: Sweep a (horizontal) line ℓ over the plane, make sure that all intersection points above the sweep line have been reported



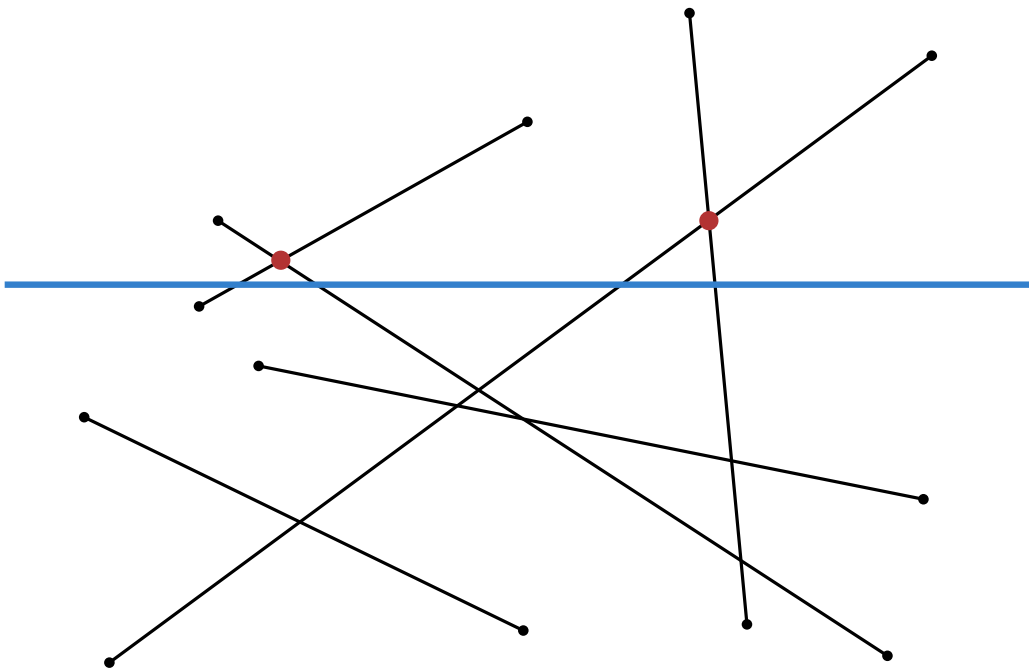
Sweep line paradigm

Idea: Sweep a (horizontal) line ℓ over the plane, make sure that all intersection points above the sweep line have been reported



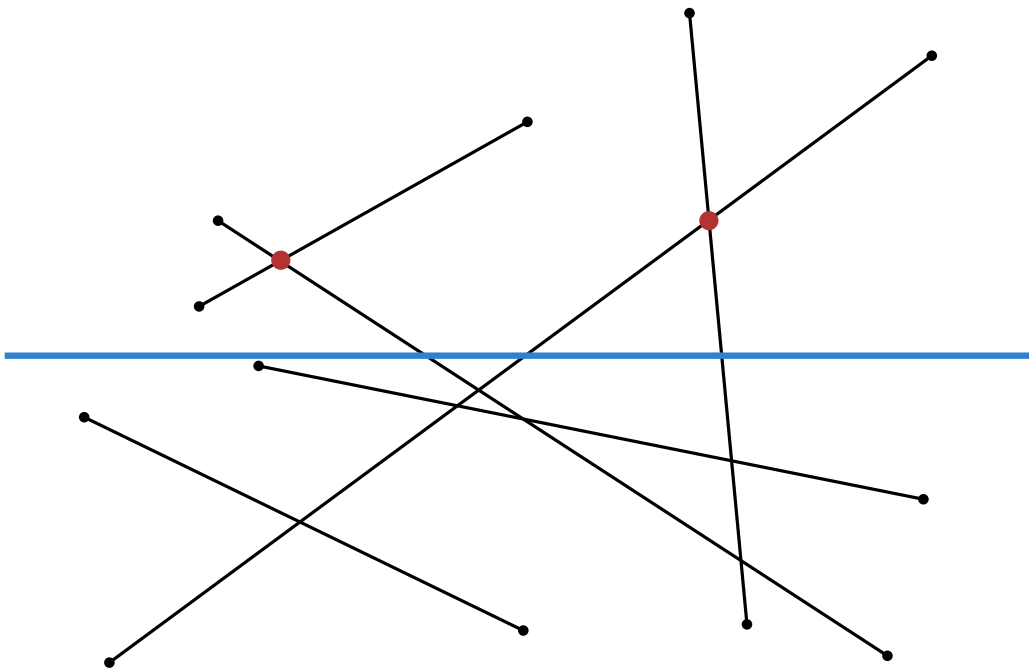
Sweep line paradigm

Idea: Sweep a (horizontal) line ℓ over the plane, make sure that all intersection points above the sweep line have been reported



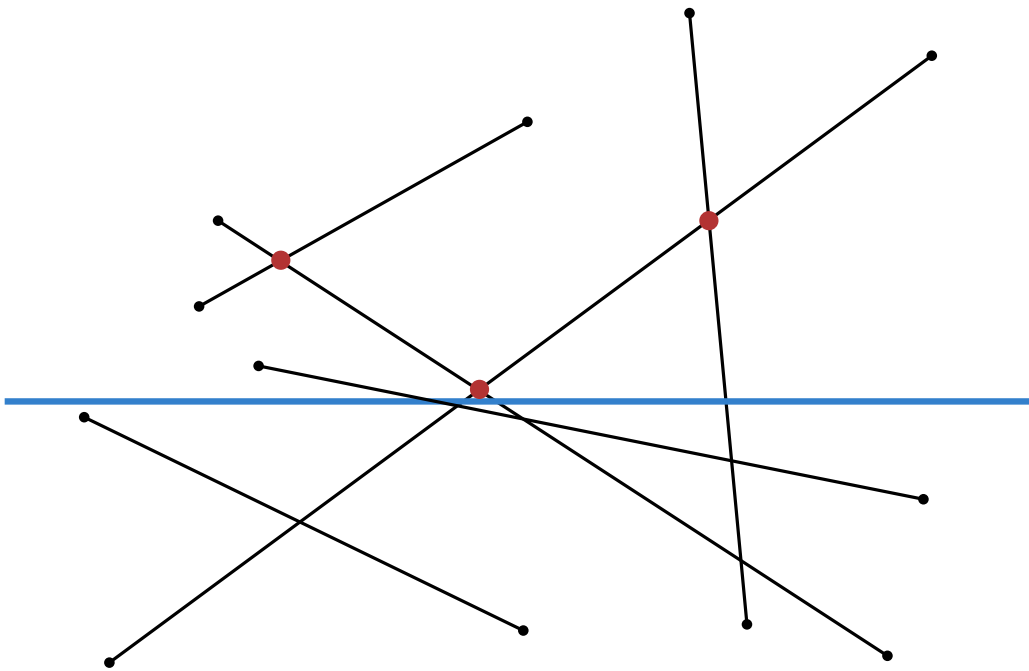
Sweep line paradigm

Idea: Sweep a (horizontal) line ℓ over the plane, make sure that all intersection points above the sweep line have been reported



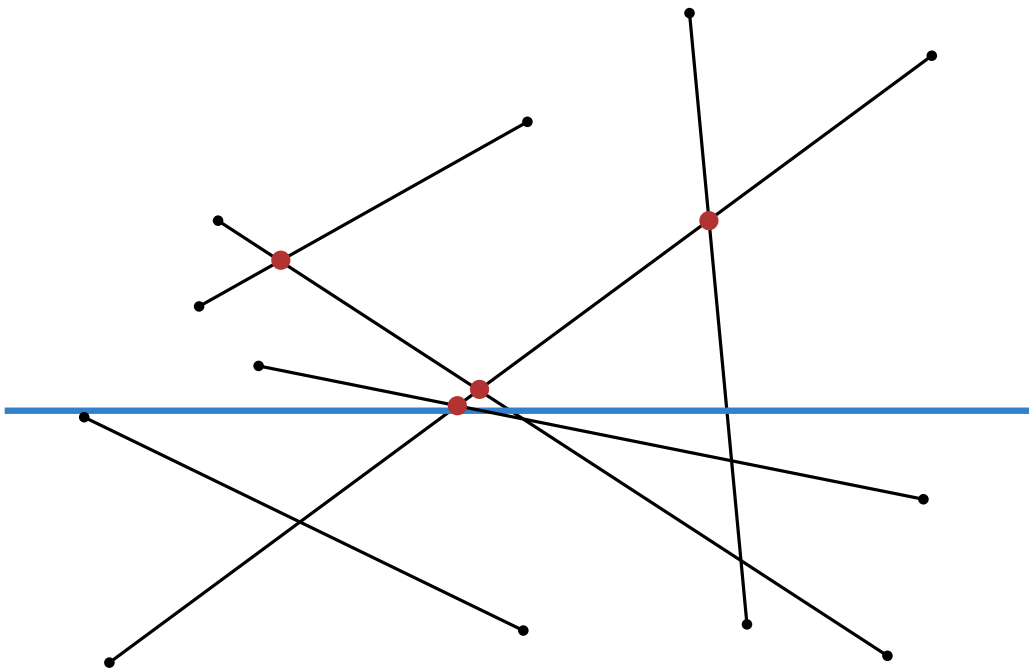
Sweep line paradigm

Idea: Sweep a (horizontal) line ℓ over the plane, make sure that all intersection points above the sweep line have been reported



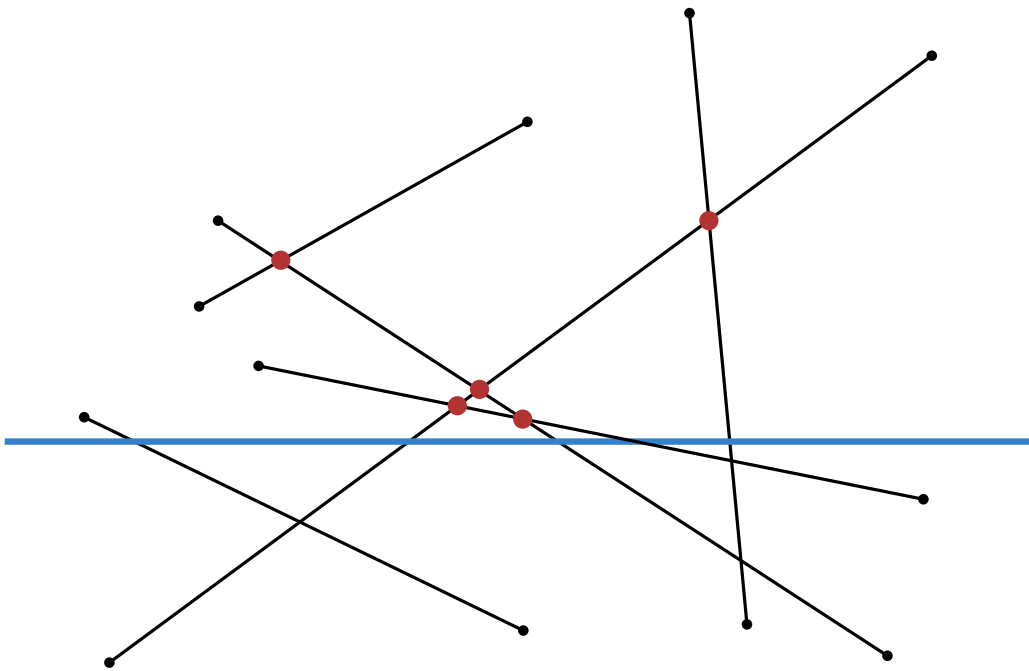
Sweep line paradigm

Idea: Sweep a (horizontal) line ℓ over the plane, make sure that all intersection points above the sweep line have been reported



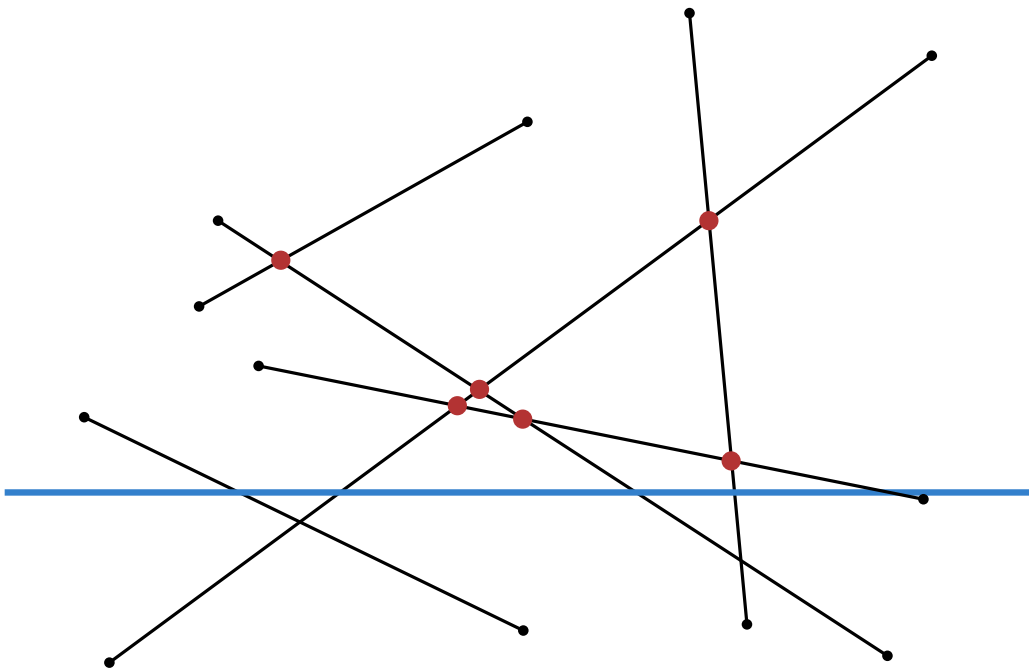
Sweep line paradigm

Idea: Sweep a (horizontal) line ℓ over the plane, make sure that all intersection points above the sweep line have been reported



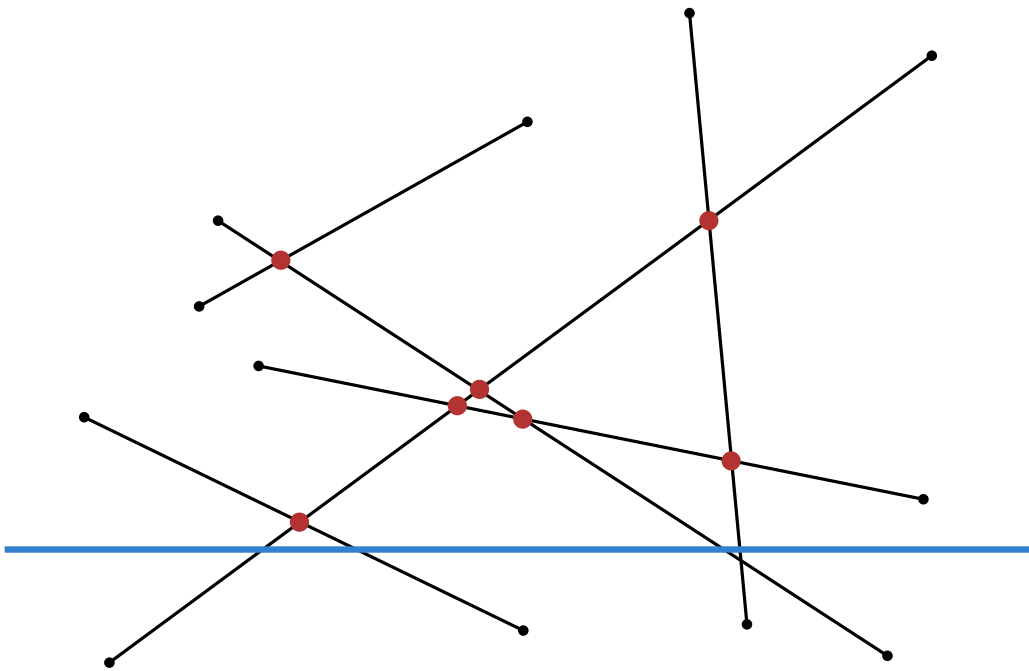
Sweep line paradigm

Idea: Sweep a (horizontal) line ℓ over the plane, make sure that all intersection points above the sweep line have been reported



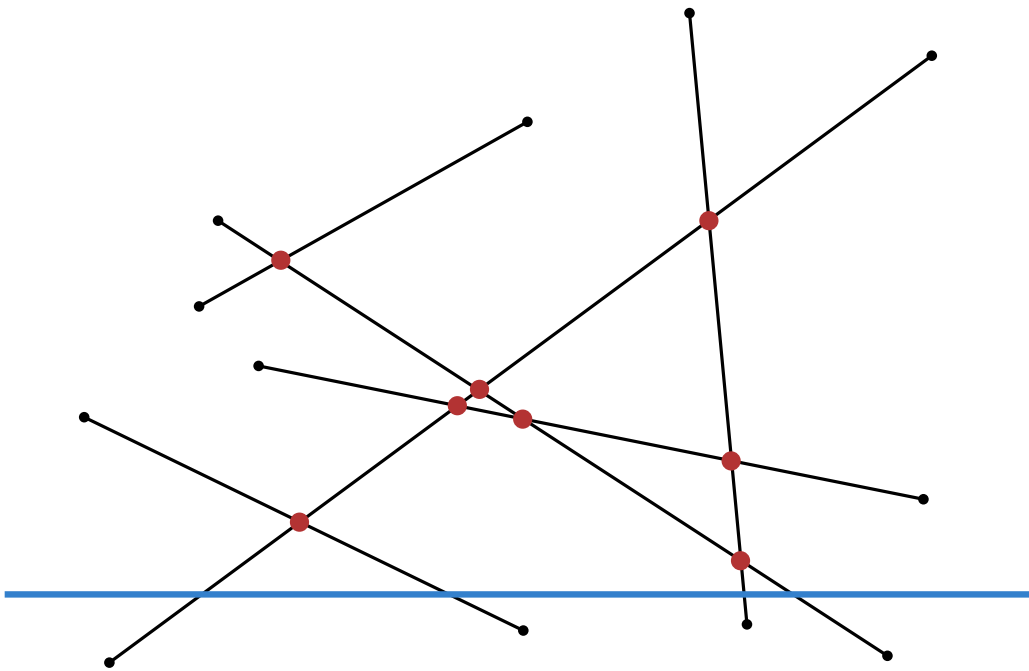
Sweep line paradigm

Idea: Sweep a (horizontal) line ℓ over the plane, make sure that all intersection points above the sweep line have been reported



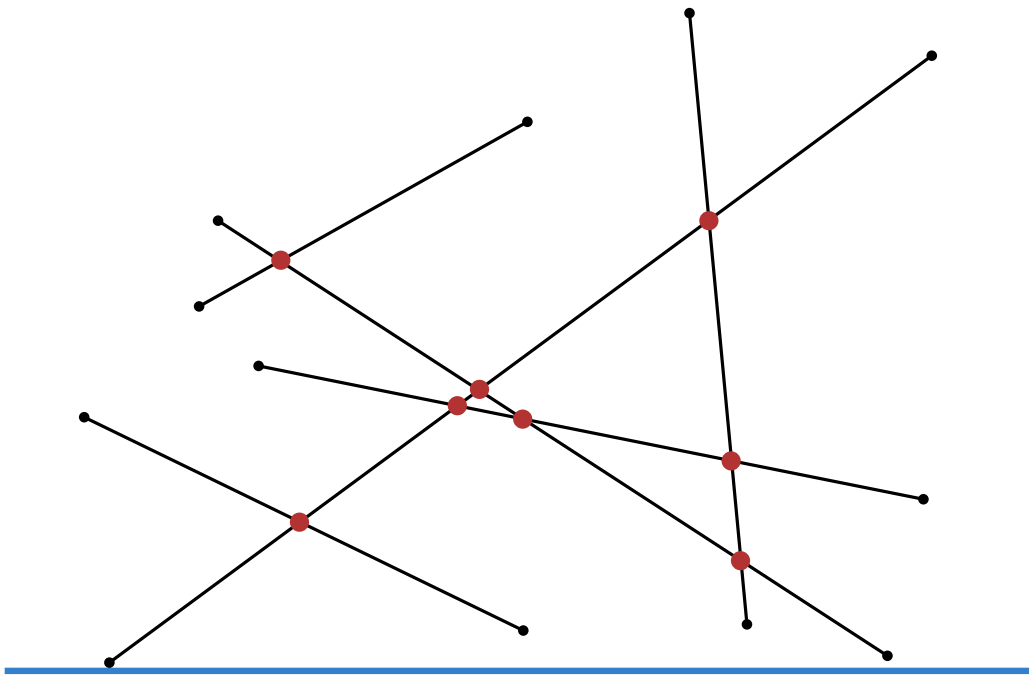
Sweep line paradigm

Idea: Sweep a (horizontal) line ℓ over the plane, make sure that all intersection points above the sweep line have been reported



Sweep line paradigm

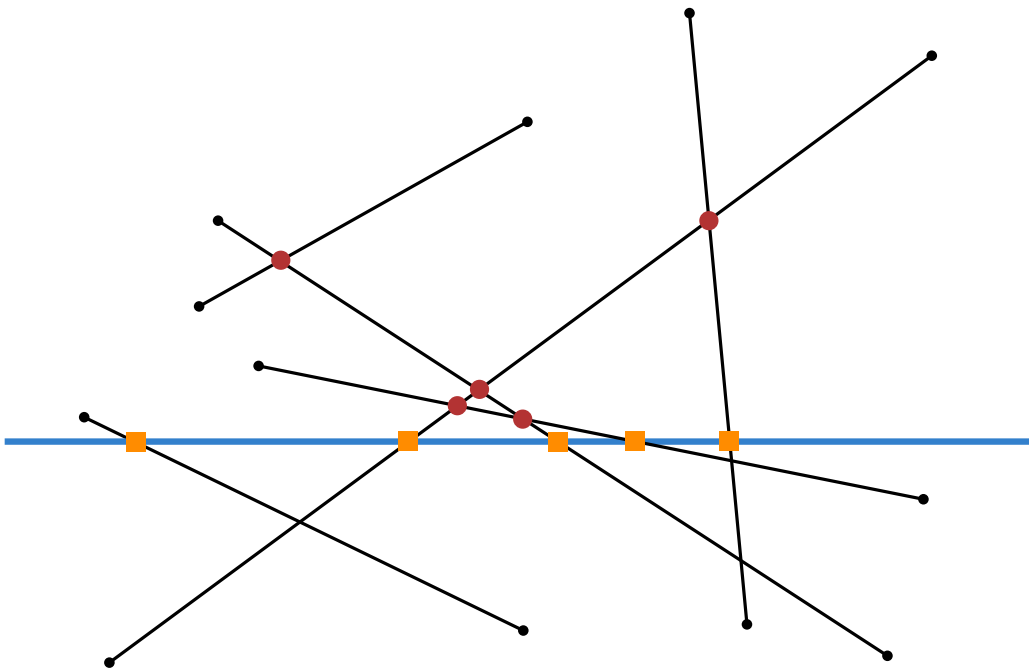
Idea: Sweep a (horizontal) line ℓ over the plane, make sure that all intersection points above the sweep line have been reported



Sweep line paradigm

Idea: Sweep a (horizontal) line ℓ over the plane, make sure that all intersection points above the sweep line have been reported

While sweeping, maintain the order in which the segments intersect ℓ.

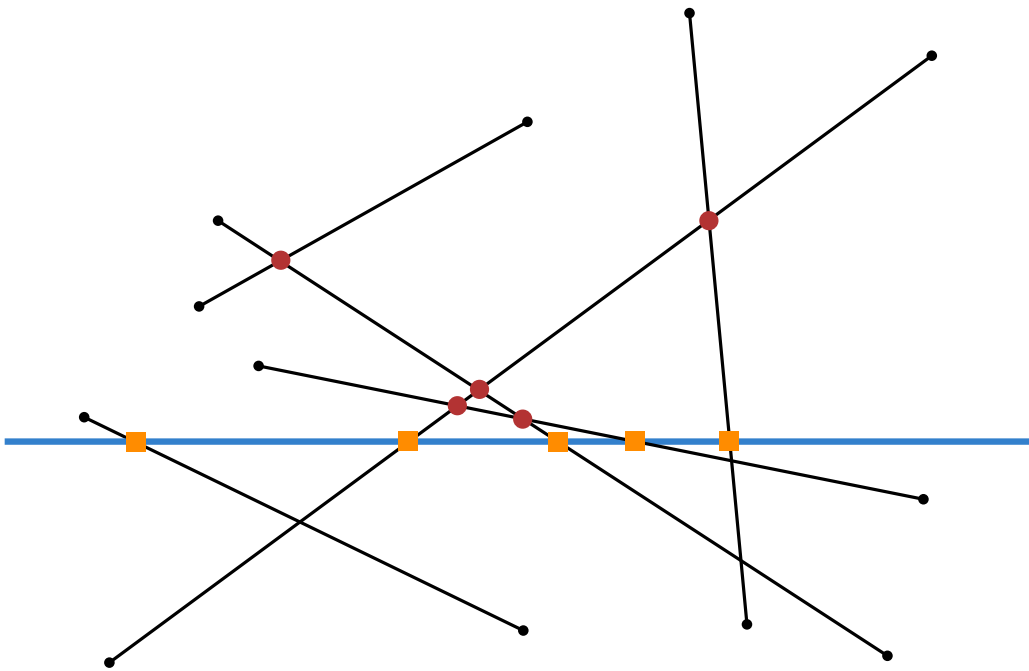


Sweep line paradigm

Idea: Sweep a (horizontal) line ℓ over the plane, make sure that all intersection points above the sweep line have been reported

While sweeping, maintain the order in which the segments intersect ℓ.

Only compare segments when they become neighbors on the sweep line.



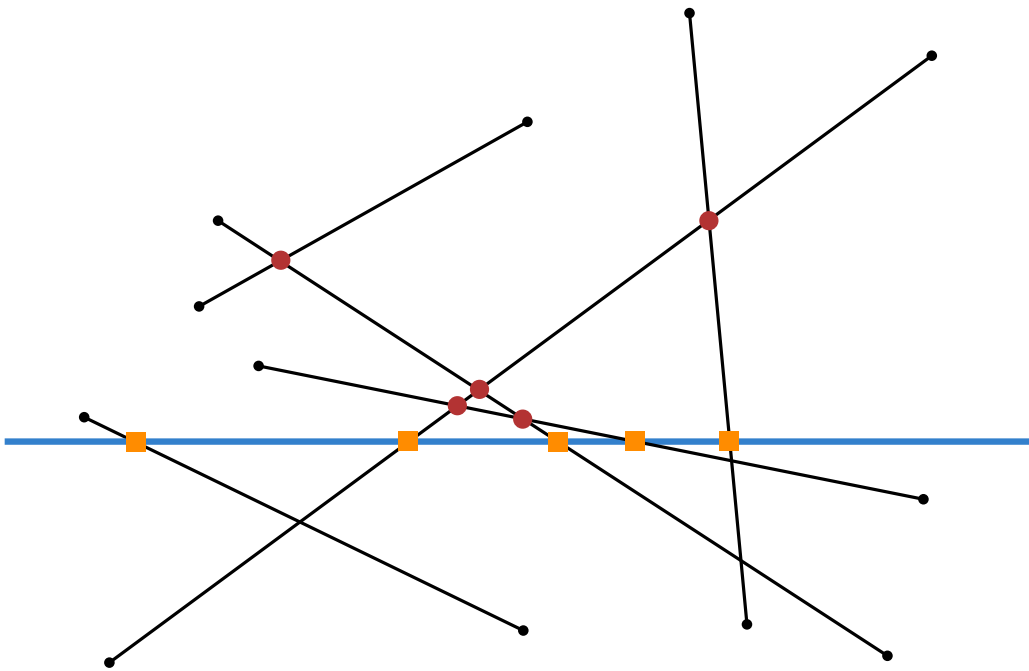
Sweep line paradigm

Idea: Sweep a (horizontal) line ℓ over the plane, make sure that all intersection points above the sweep line have been reported

While sweeping, maintain the order in which the segments intersect ℓ.

Store these segments in the status structure.

Only compare segments when they become neighbors on the sweep line.



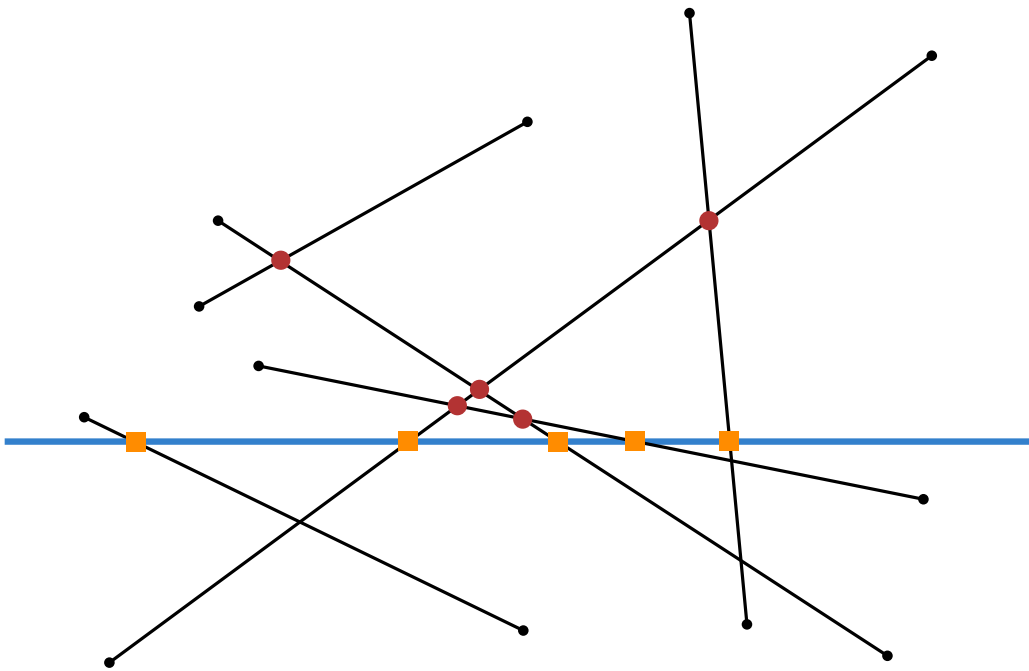
Sweep line paradigm

Idea: Sweep a (horizontal) line ℓ over the plane, make sure that all intersection points above the sweep line have been reported

While sweeping, maintain the order in which the segments intersect ℓ.

Store these segments in the status structure.

Only compare segments when they become neighbors in the status structure.



Sweep line paradigm

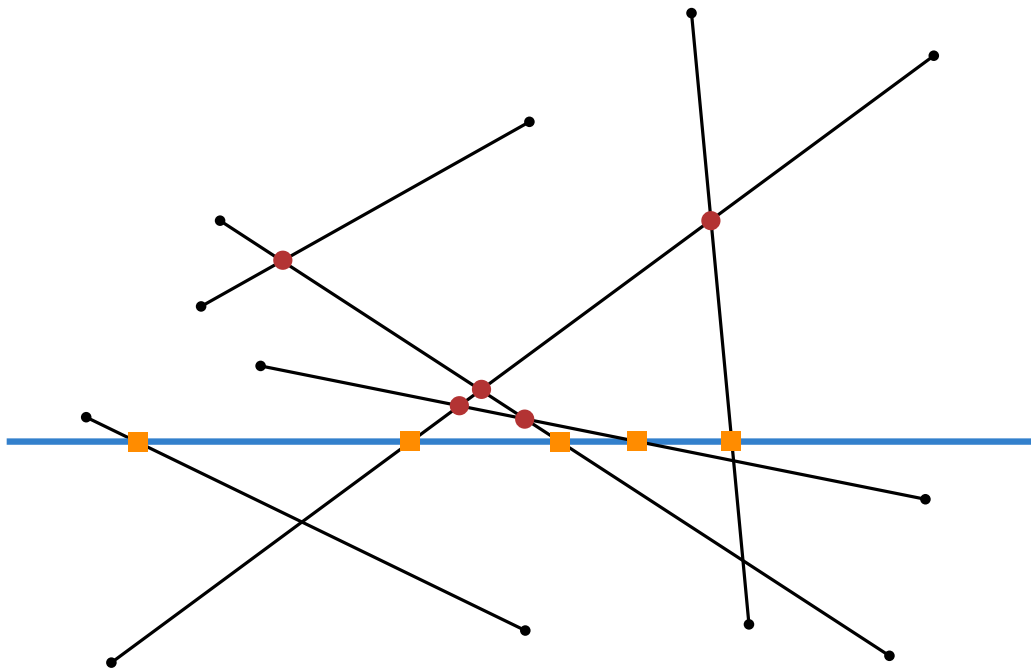
Idea: Sweep a (horizontal) line ℓ over the plane, make sure that all intersection points above the sweep line have been reported

While sweeping, maintain the order in which the segments intersect ℓ.

Store these segments in the status structure.

We use a balanced BST as status structure.

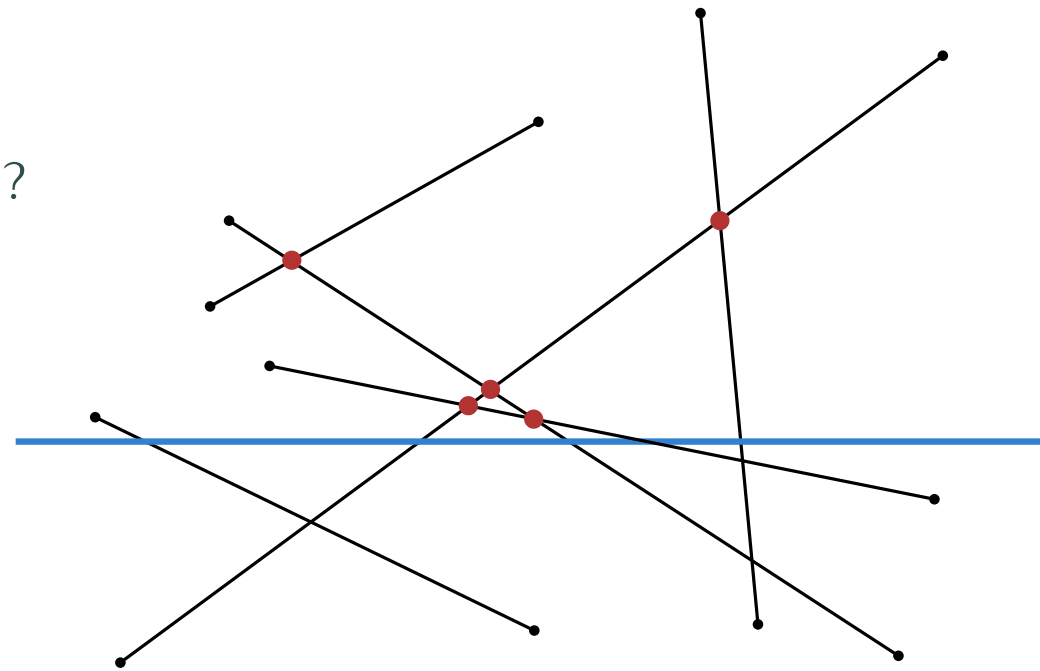
Only compare segments when they become neighbors in the status structure.



Sweep line paradigm

Idea: Sweep a (horizontal) line ℓ over the plane, make sure that all intersection points above the sweep line have been reported

When does the status structure change?



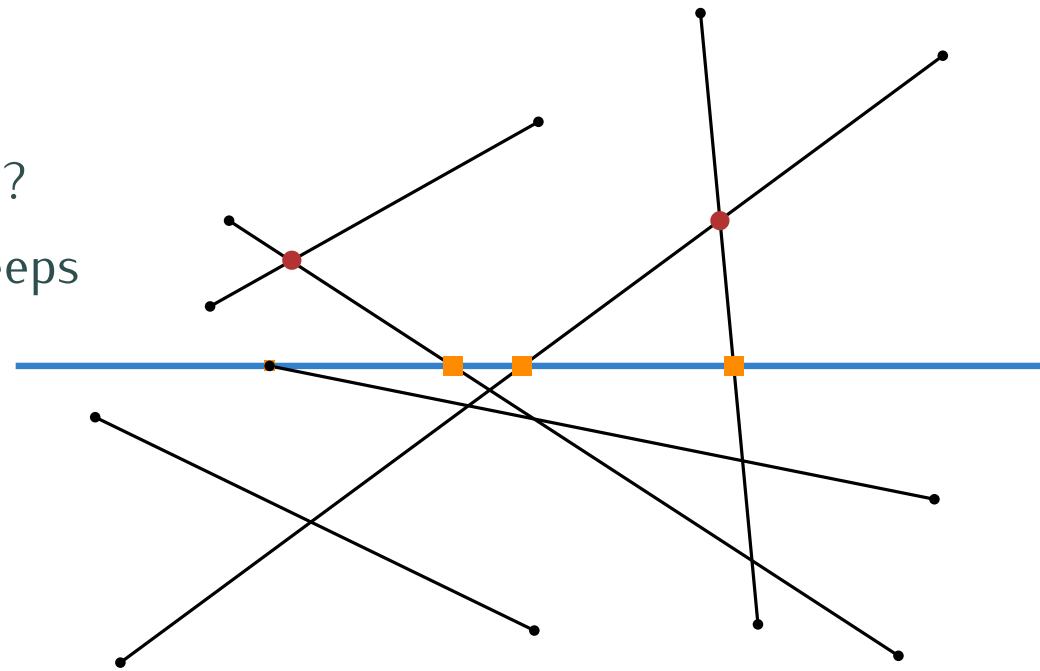
Sweep line paradigm

Idea: Sweep a (horizontal) line ℓ over the plane, make sure that all intersection points above the sweep line have been reported

When does the status structure change?

At discerete events, i.e. when ℓ sweeps over

- the upper endpoint of a segment



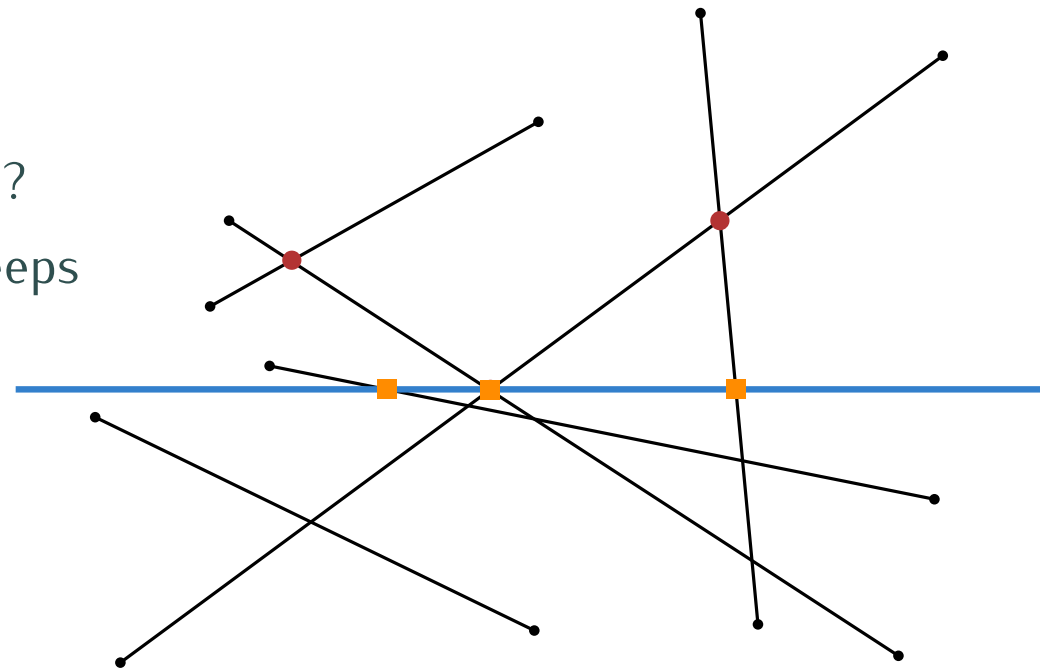
Sweep line paradigm

Idea: Sweep a (horizontal) line ℓ over the plane, make sure that all intersection points above the sweep line have been reported

When does the status structure change?

At discerete events, i.e. when ℓ sweeps over

- the upper endpoint of a segment
- an intersection point



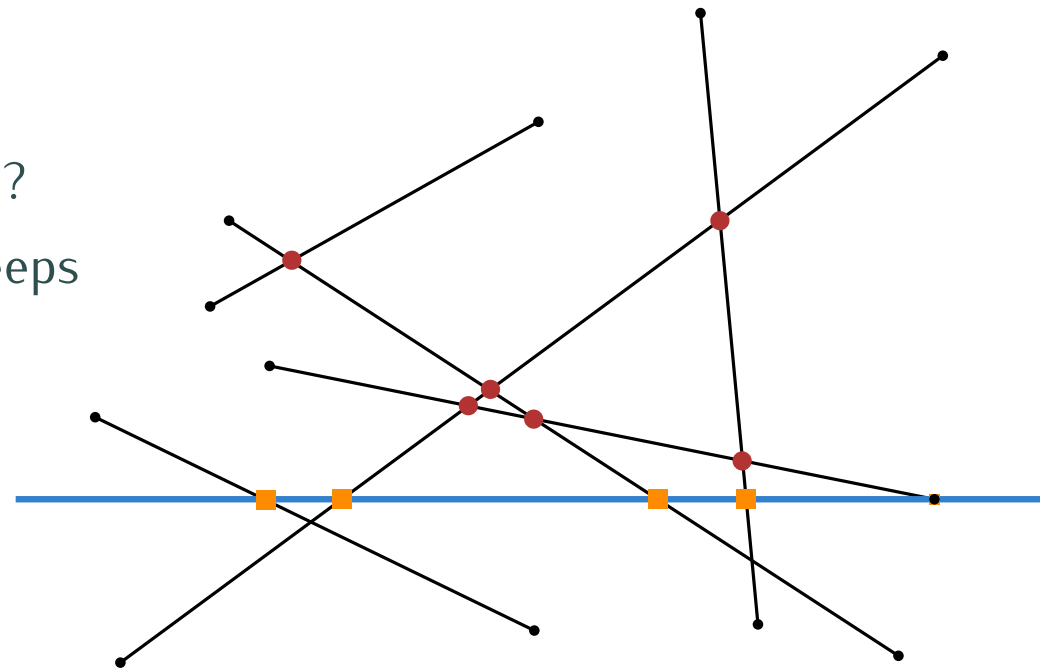
Sweep line paradigm

Idea: Sweep a (horizontal) line ℓ over the plane, make sure that all intersection points above the sweep line have been reported

When does the status structure change?

At discerete events, i.e. when ℓ sweeps over

- the upper endpoint of a segment
- an intersection point
- the lower endpoint of a segment



Sweep line paradigm

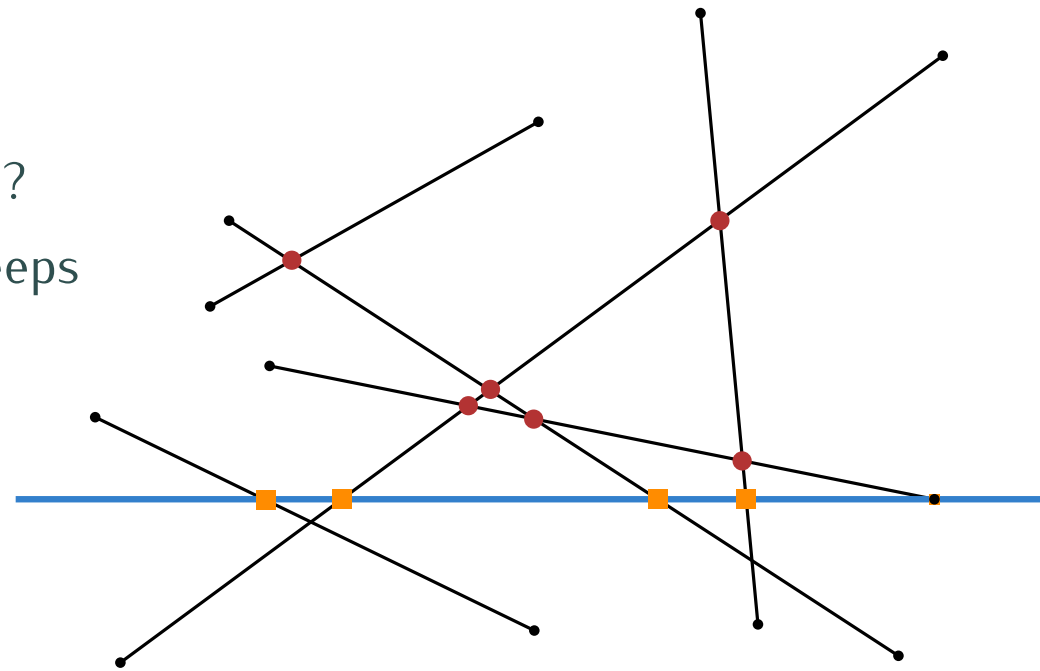
Idea: Sweep a (horizontal) line ℓ over the plane, make sure that all intersection points above the sweep line have been reported

When does the status structure change?

At discerete events, i.e. when ℓ sweeps over

- the upper endpoint of a segment
- an intersection point
- the lower endpoint of a segment

Maintain the events in an event queue.



Sweep line paradigm

Idea: Sweep a (horizontal) line ℓ over the plane, make sure that all intersection points above the sweep line have been reported

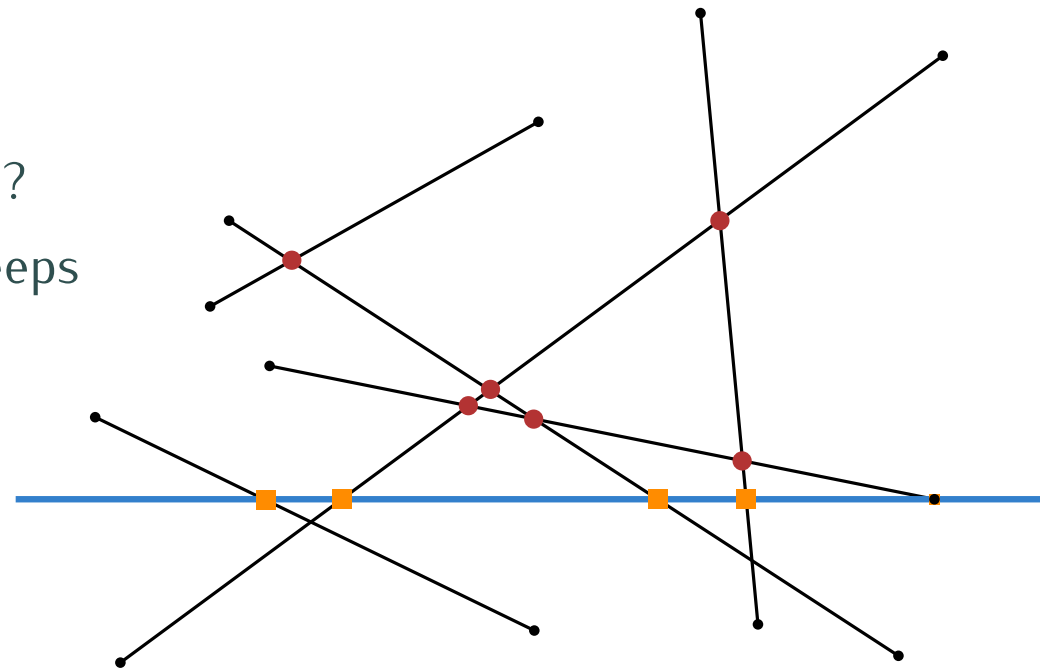
When does the status structure change?

At discerete events, i.e. when ℓ sweeps over

- the upper endpoint of a segment
- an intersection point
- the lower endpoint of a segment

Maintain the events in an event queue.

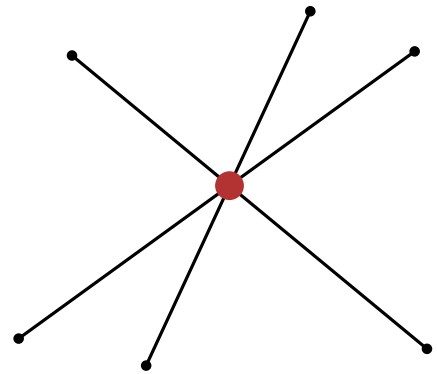
We use a BST as event queue.



General position

Our algorithm works for segments in general position, i.e. if there are

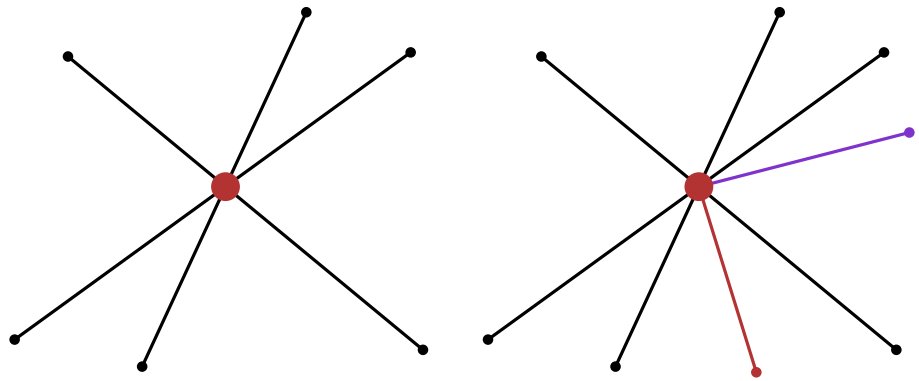
- no three segments that intersect in a single point



General position

Our algorithm works for segments in general position, i.e. if there are

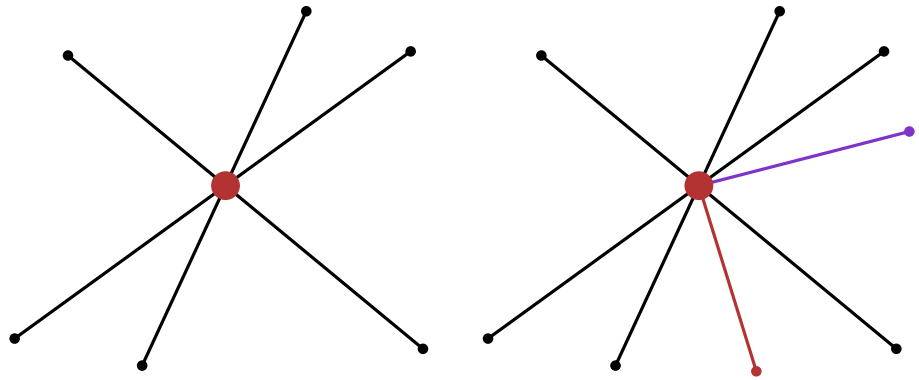
- no three segments that intersect in a single point



General position

Our algorithm works for segments in general position, i.e. if there are

- no three segments that intersect in a single point

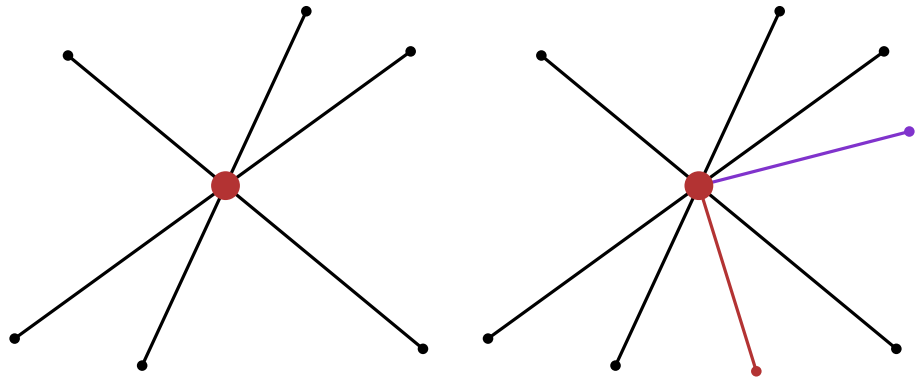


- no horizontal segments

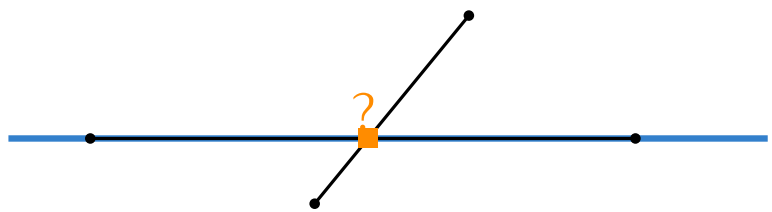
General position

Our algorithm works for segments in general position, i.e. if there are

- no three segments that intersect in a single point



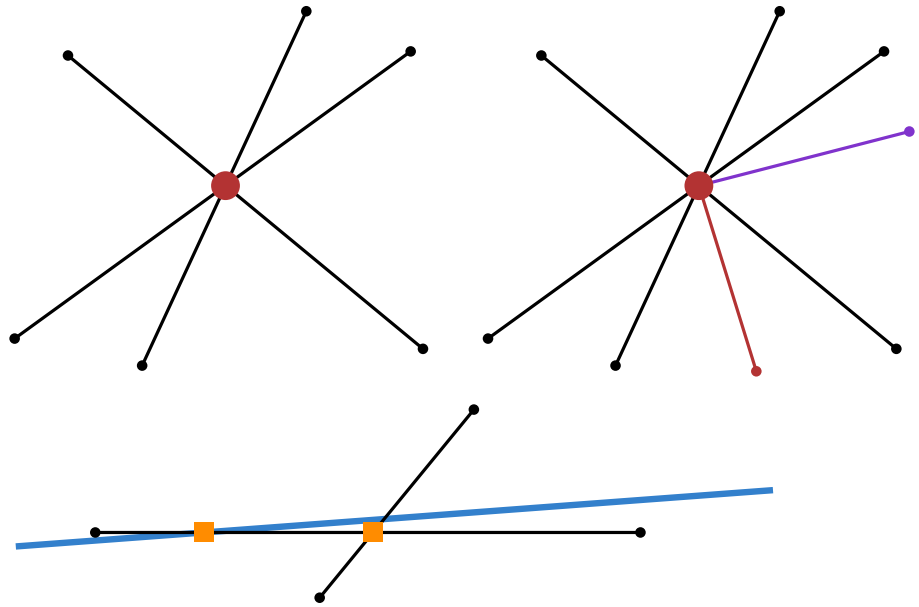
- no horizontal segments



General position

Our algorithm works for segments in general position, i.e. if there are

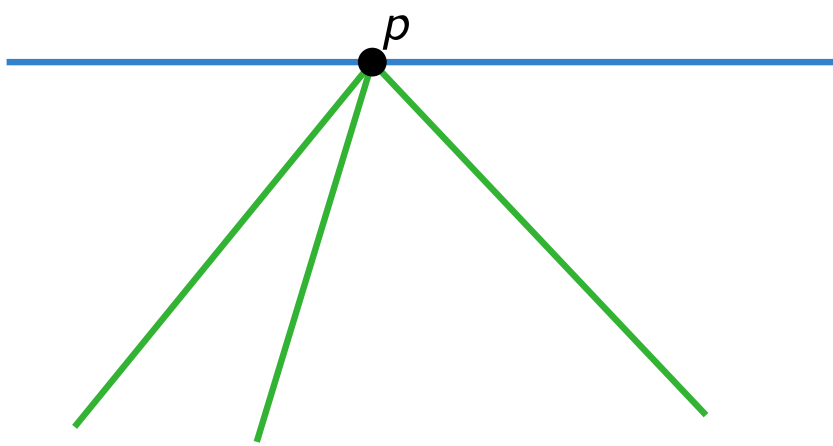
- no three segments that intersect in a single point



- no horizontal segments

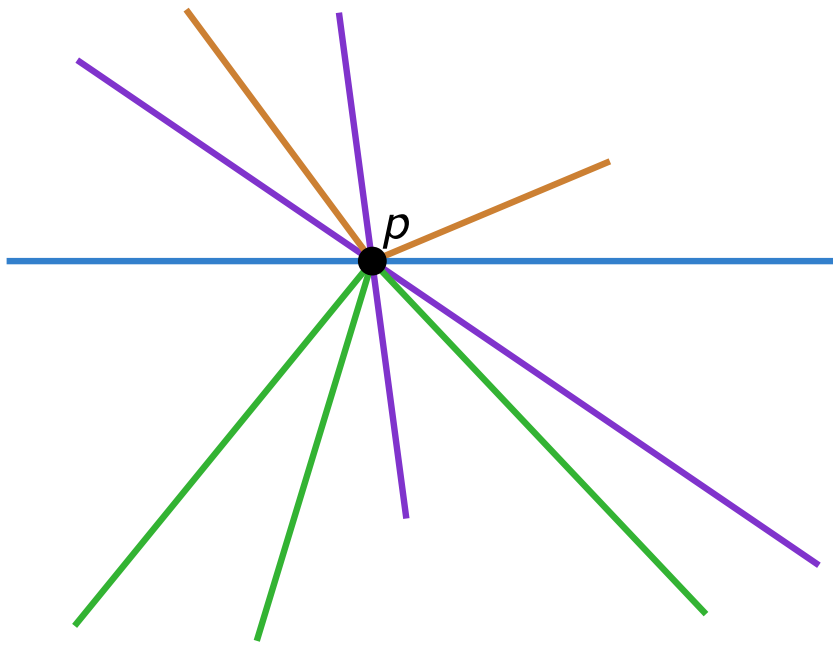
Handling an Eventpoint p

1. Let $U(p)$ be the segments whose upper endpoint is p.



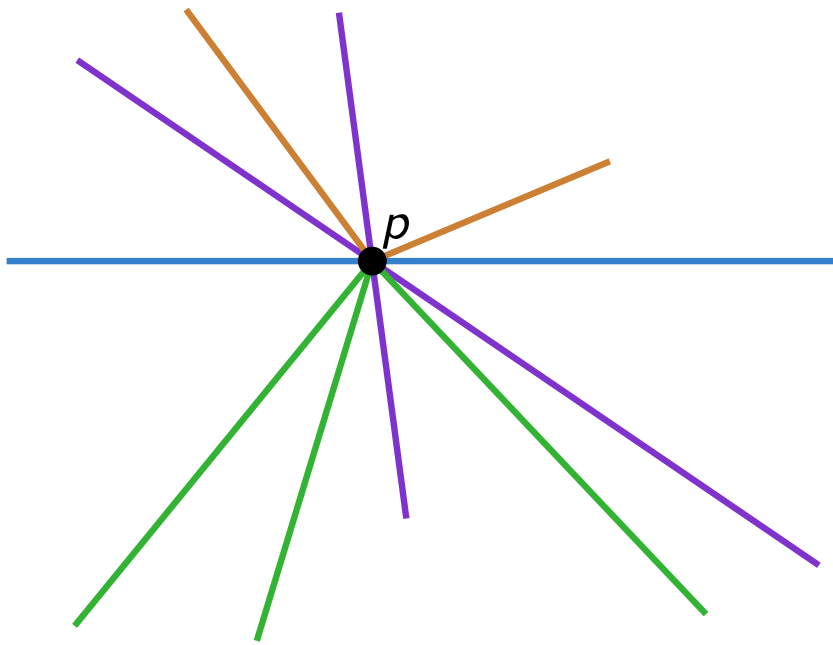
Handling an Eventpoint p

1. Let $U(p)$ be the segments whose upper endpoint is p.
2. Find the segments in the status structure $\mathcal{T}$ that contain p; let $L(p)$ be the segments whose lower endpoint is p, and let $C(p)$ be the segments that contain p in their interior.



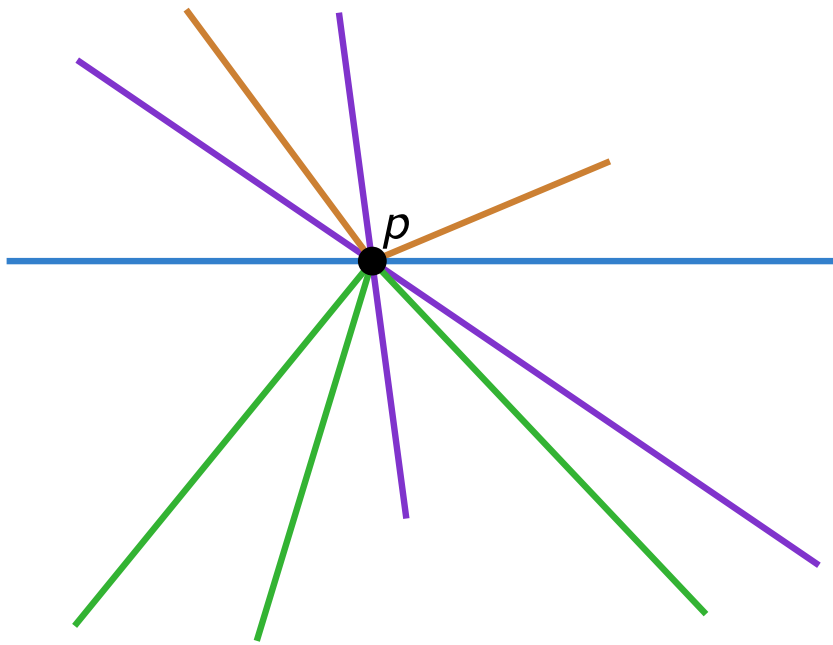
Handling an Eventpoint p

1. Let $U(p)$ be the segments whose upper endpoint is p.
2. Find the segments in the status structure $\mathcal{T}$ that contain p; let $L(p)$ be the segments whose lower endpoint is p, and let $C(p)$ be the segments that contain p in their interior.
3. **if** $|U(p) \cup L(p) \cup C(p)| \geq 2$ **then**
   Report p and all segments in $U(p) \cup L(p) \cup C(p)$



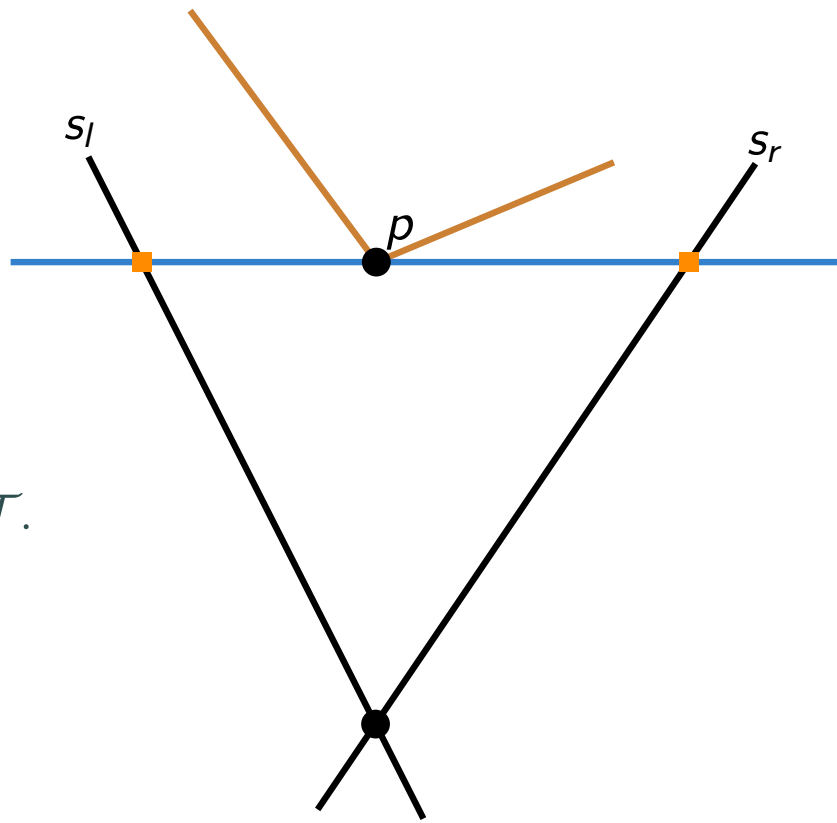
Handling an Eventpoint p

1. Let $U(p)$ be the segments whose upper endpoint is p.
2. Find the segments in the status structure $\mathcal{T}$ that contain p; let $L(p)$ be the segments whose lower endpoint is p, and let $C(p)$ be the segments that contain p in their interior.
3. **if** $|U(p) \cup L(p) \cup C(p)| \geq 2$ **then**
   Report p and all segments in $U(p) \cup L(p) \cup C(p)$
4. Delete the segments in $L(p)$ and $C(p)$ from $\mathcal{T}$.
5. Insert the segments in $U(p)$ and $C(p)$ into $\mathcal{T}$.



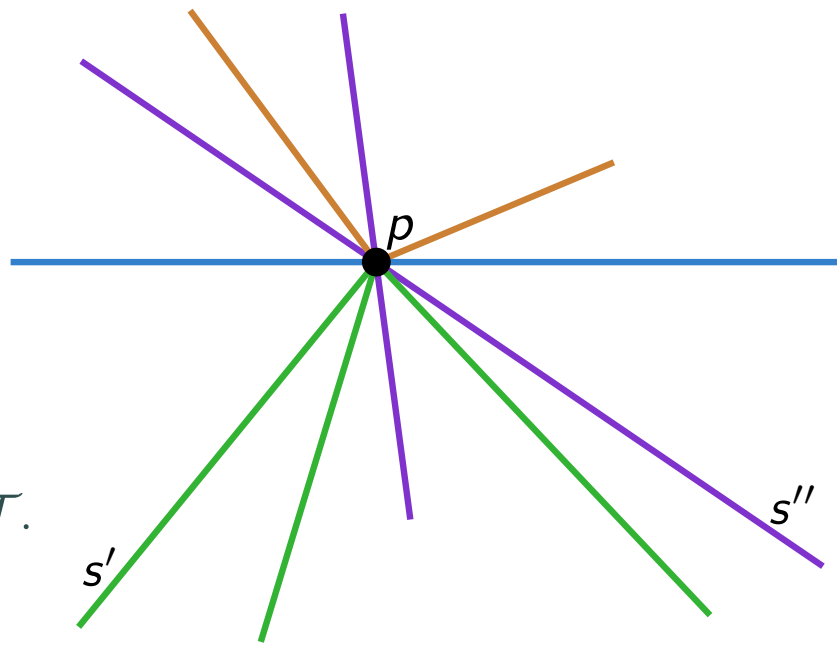
Handling an Eventpoint p

1. Let $U(p)$ be the segments whose upper endpoint is p.
2. Find the segments in the status structure $\mathcal{T}$ that contain p; let $L(p)$ be the segments whose lower endpoint is p, and let $C(p)$ be the segments that contain p in their interior.
3. **if** $|U(p) \cup L(p) \cup C(p)| \geq 2$ **then**
   Report p and all segments in $U(p) \cup L(p) \cup C(p)$
4. Delete the segments in $L(p)$ and $C(p)$ from $\mathcal{T}$.
5. Insert the segments in $U(p)$ and $C(p)$ into $\mathcal{T}$.
6. **if** $U(p) \cup C(p) = \emptyset$ **then**
7. Find the left and right neighbors s_l and s_r of p in $\mathcal{T}$.
8. FindNewEvent(s_l, s_r, p)



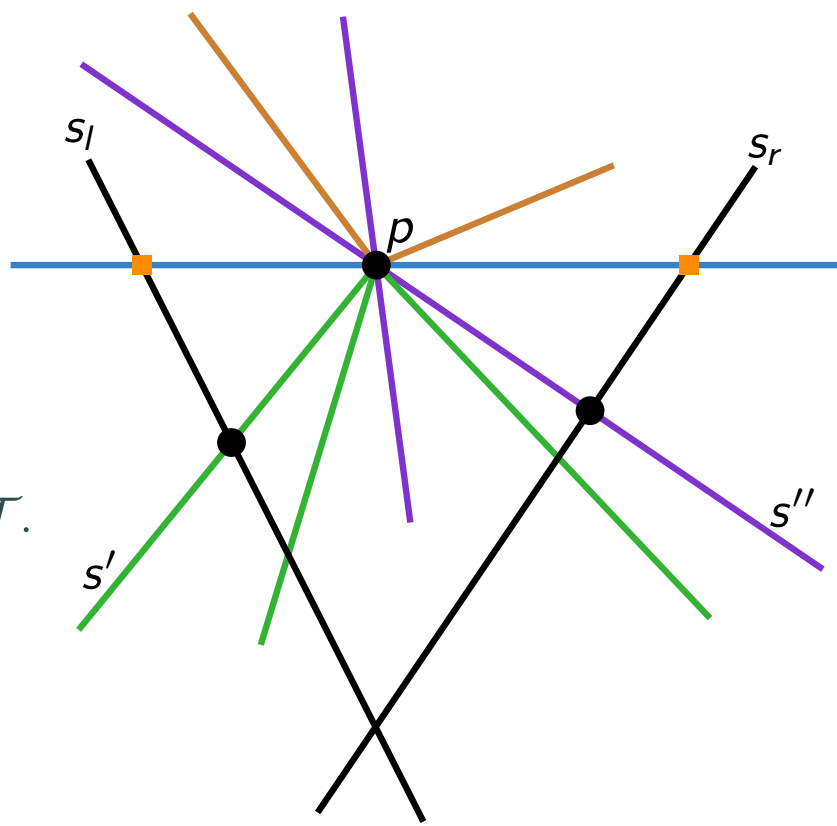
Handling an Eventpoint p

1. Let $U(p)$ be the segments whose upper endpoint is p.
2. Find the segments in the status structure $\mathcal{T}$ that contain p; let $L(p)$ be the segments whose lower endpoint is p, and let $C(p)$ be the segments that contain p in their interior.
3. **if** $|U(p) \cup L(p) \cup C(p)| \geq 2$ **then**
   Report p and all segments in $U(p) \cup L(p) \cup C(p)$
4. Delete the segments in $L(p)$ and $C(p)$ from $\mathcal{T}$.
5. Insert the segments in $U(p)$ and $C(p)$ into $\mathcal{T}$.
6. **if** $U(p) \cup C(p) = \emptyset$ **then**
7. Find the left and right neighbors s_l and s_r of p in $\mathcal{T}$.
8. FindNewEvent(s_l, s_r, p)
9. **else**
   Let s' and s'' be the leftmost and rightmost segment in $U(p) \cup C(p)$



Handling an Eventpoint p

1. Let $U(p)$ be the segments whose upper endpoint is p.
2. Find the segments in the status structure $\mathcal{T}$ that contain p; let $L(p)$ be the segments whose lower endpoint is p, and let $C(p)$ be the segments that contain p in their interior.
3. **if** $|U(p) \cup L(p) \cup C(p)| \geq 2$ **then**
   Report p and all segments in $U(p) \cup L(p) \cup C(p)$
4. Delete the segments in $L(p)$ and $C(p)$ from $\mathcal{T}$.
5. Insert the segments in $U(p)$ and $C(p)$ into $\mathcal{T}$.
6. **if** $U(p) \cup C(p) = \emptyset$ **then**
7. Find the left and right neighbors s_l and s_r of p in $\mathcal{T}$.
8. FindNewEvent(s_l, s_r, p)
9. **else**
   Let s' and s'' be the leftmost and rightmost segment in $U(p) \cup C(p)$
10. Let s_l be the left neighbor of s' and s_r the right neighbor of s''.
11. FindNewEvent(s_ℓ, s', p)
12. FindNewEvent(s'', s_r, p)



Analysis

$$\text{Running time} = O\left(\sum_{i=1}^{\#\text{events}} \text{time to handle event } p_i\right)$$

Analysis

$$\text{Running time} = O\left(\sum_{i=1}^{\#\text{events}} \text{time to handle event } p_i\right)$$

$$\begin{aligned}\#\text{events} &= \#\text{endpoints} + \#\text{intersection points} \\ &= 2n + k\end{aligned}$$

Analysis

$$\text{Running time} = O\left(\sum_{i=1}^{\#\text{events}} \text{time to handle event } p_i\right)$$

$$\begin{aligned}\#\text{events} &= \#\text{endpoints} + \#\text{intersection points} \\ &= 2n + k\end{aligned}$$

$$\text{time to handle event } p_i = O((\#\text{segments that contain point } p_i) \log n)$$

Analysis

$$\text{Running time} = O\left(\left(\sum_{i=1}^{\#\text{events}} \#\text{segments that contain point } p_i\right) \log n\right)$$

$$\begin{aligned}\#\text{events} &= \#\text{endpoints} + \#\text{intersection points} \\ &= 2n + k\end{aligned}$$

time to handle event $p_i = O((\#\text{segments that contain point } p_i) \log n)$

Analysis

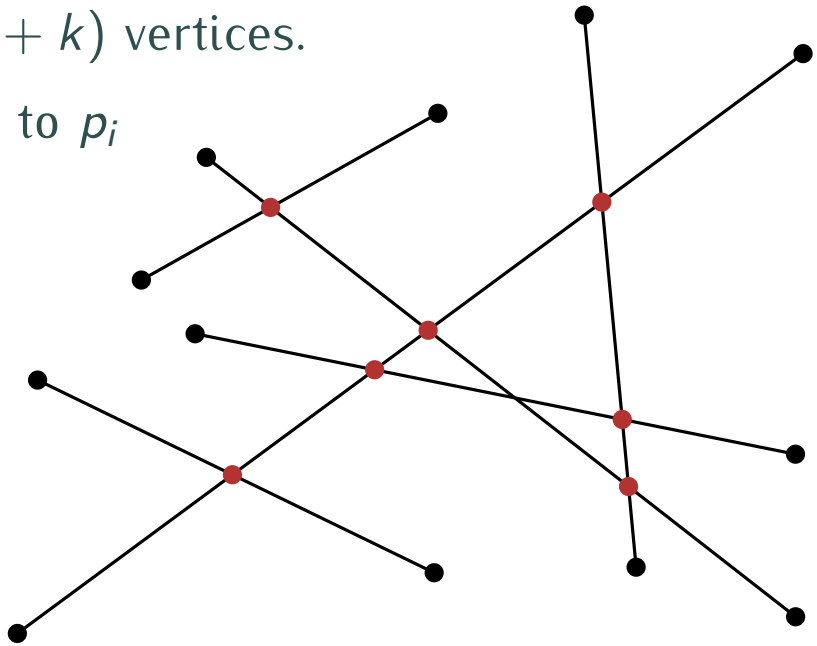
$$\text{Running time} = O\left(\left(\sum_{i=1}^{\#\text{events}} \#\text{segments that contain point } p_i\right) \log n\right)$$

$$\begin{aligned}\#\text{events} &= \#\text{endpoints} + \#\text{intersection points} \\ &= 2n + k\end{aligned}$$

time to handle event $p_i = O((\#\text{segments that contain point } p_i) \log n)$

The output corresponds to a planar graph with $O(n + k)$ vertices.

$\#\text{segments that contain point } p_i = \#\text{edges incident to } p_i$



Analysis

$$\text{Running time} = O\left(\left(\sum_{i=1}^{\#\text{events}} \#\text{segments that contain point } p_i\right) \log n\right)$$

$$\begin{aligned}\#\text{events} &= \#\text{endpoints} + \#\text{intersection points} \\ &= 2n + k\end{aligned}$$

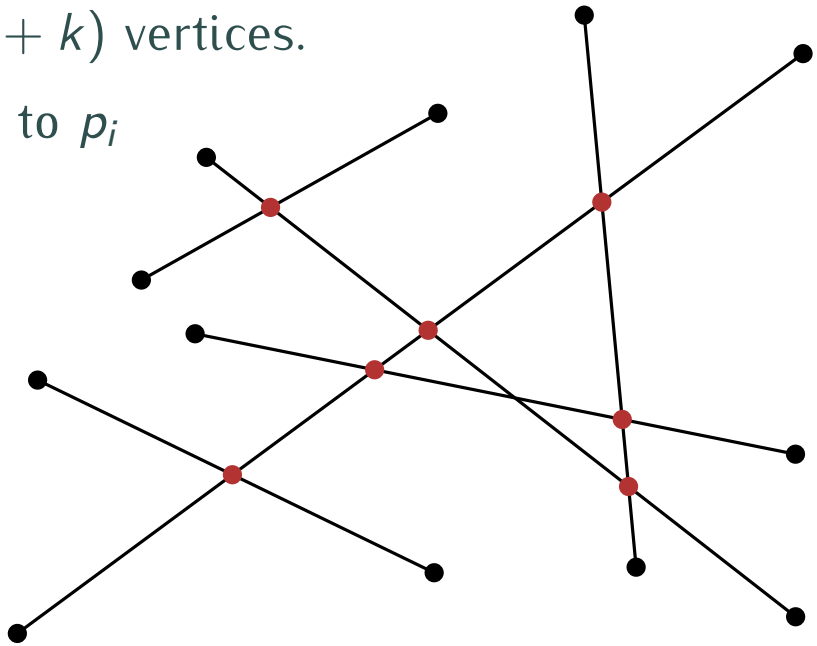
time to handle event $p_i = O((\#\text{segments that contain point } p_i) \log n)$

The output corresponds to a planar graph with $O(n + k)$ vertices.

$\#\text{segments that contain point } p_i = \#\text{edges incident to } p_i$

Such a graph has $O(n + k)$ edges

$$\sum_{i=1}^{\#\text{events}} \#\text{segments that contain point } p_i = O(n + k)$$



Analysis

Running time = $O((n+k)\log n)$

#events = #endpoints + #intersection points
$= 2n + k$

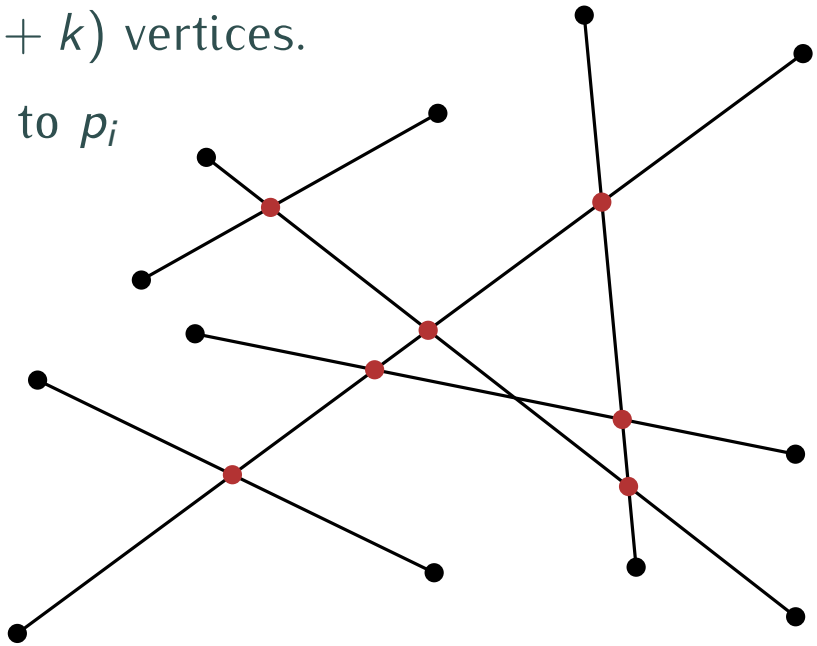
time to handle event p_i = $O((\#\text{segments that contain point } p_i)\log n)$

The output corresponds to a planar graph with $O(n+k)$ vertices.

#segments that contain point p_i = #edges incident to p_i

Such a graph has $O(n+k)$ edges

$$\sum_{i=1}^{\#\text{events}} \#\text{segments that contain point } p_i = O(n+k)$$



Analysis

Running time = $O((n+k)\log n)$

#events = #endpoints + #intersection points
$= 2n + k$

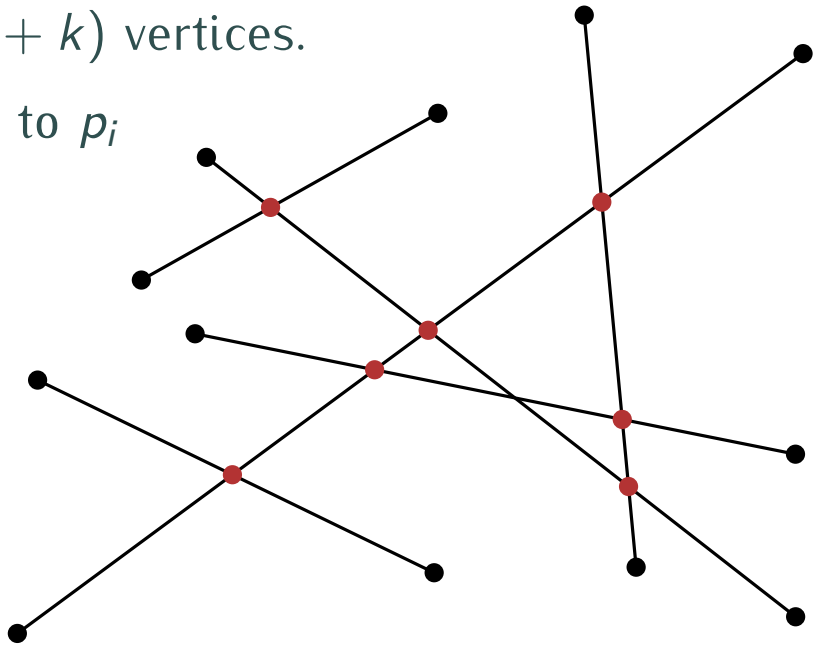
time to handle event $p_i = O((\#\text{segments that contain point } p_i)\log n)$

The output corresponds to a planar graph with $O(n+k)$ vertices.

#segments that contain point p_i = #edges incident to p_i

Such a graph has $O(n+k)$ edges

$$\sum_{i=1}^{\#\text{events}} \#\text{segments that contain point } p_i = O(n+k)$$



Theorem. Let S be a set of n line segments in $\mathbb{R}^2$. We can compute all k intersection points in $O((n+k)\log n)$ time.