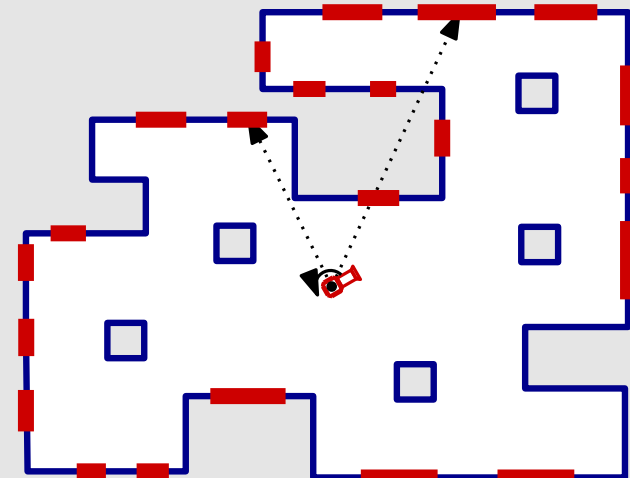
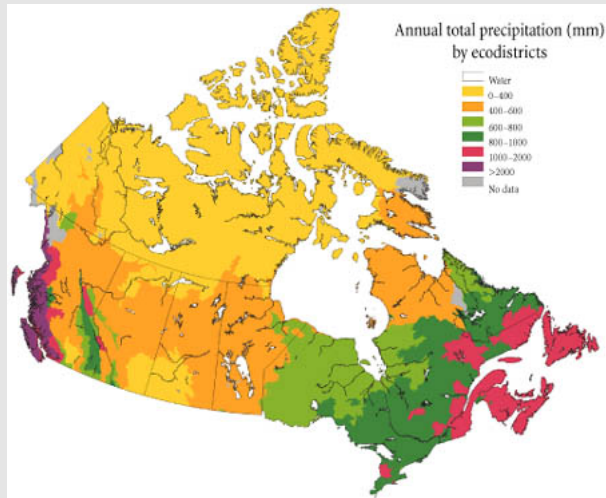


# Map Overlay

en andere toepassingen van plane sweep

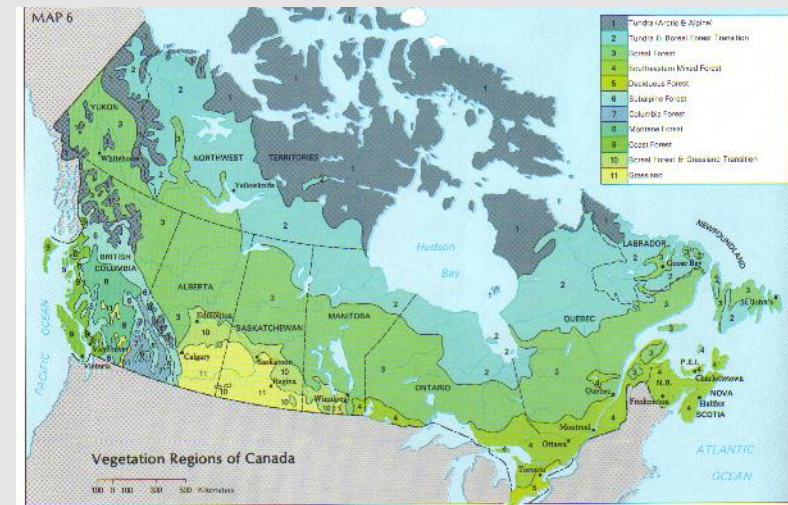
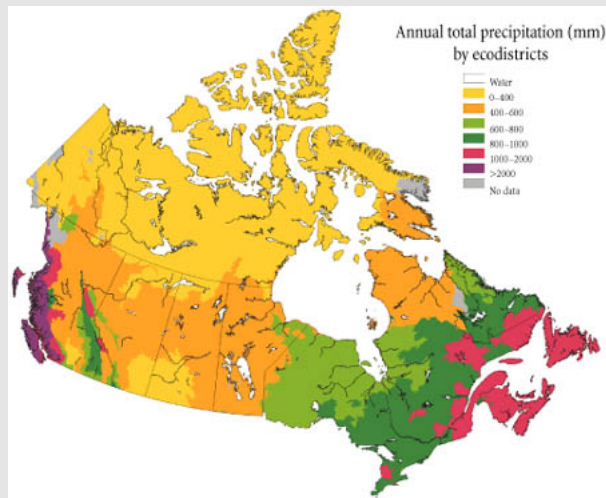
Mark de Berg

TU Eindhoven



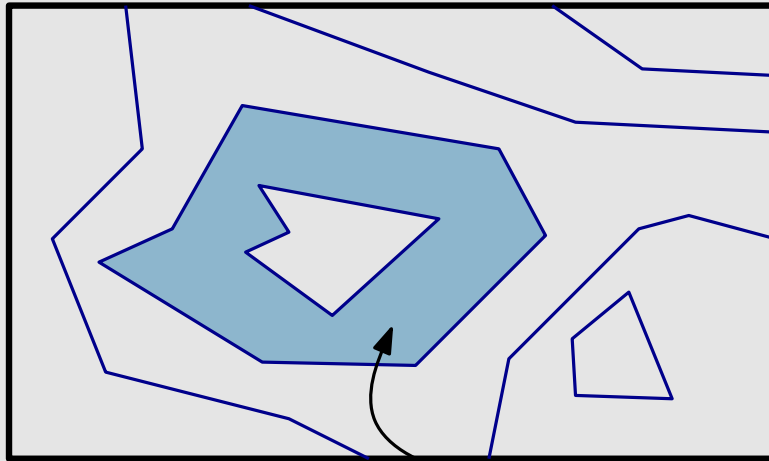
- per *thema* een apart kaartje
- gebruiker kan kaartjes “over elkaar leggen”

## neerslag Canada



## vegetatie Canada

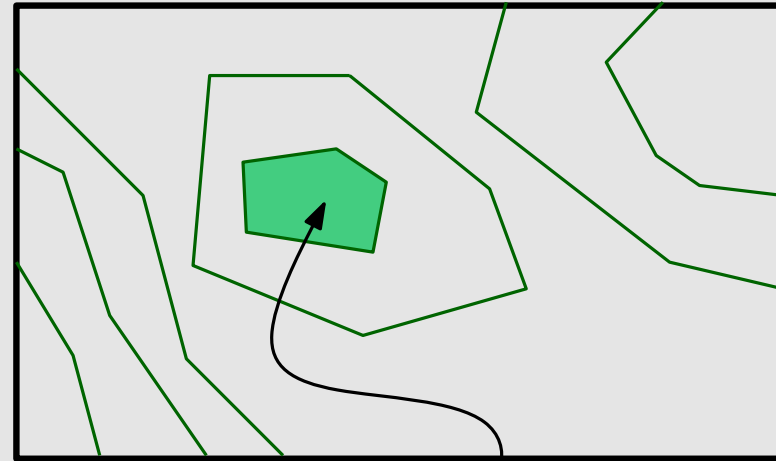
neerslagkaart



neerslag 650-700 mm

+

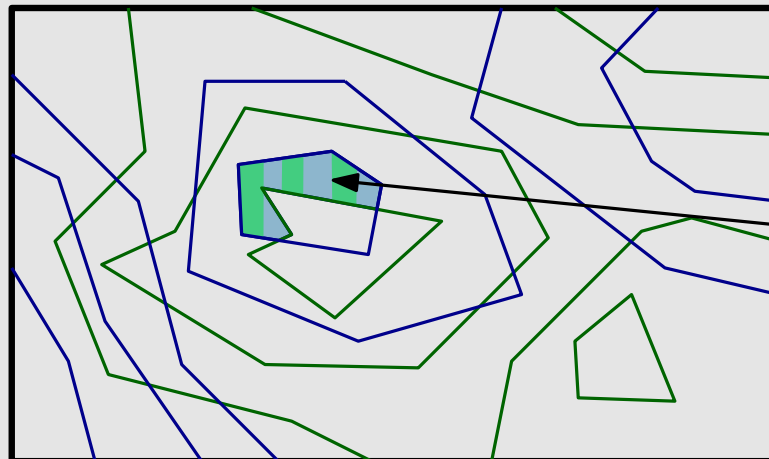
vegetatiekaart



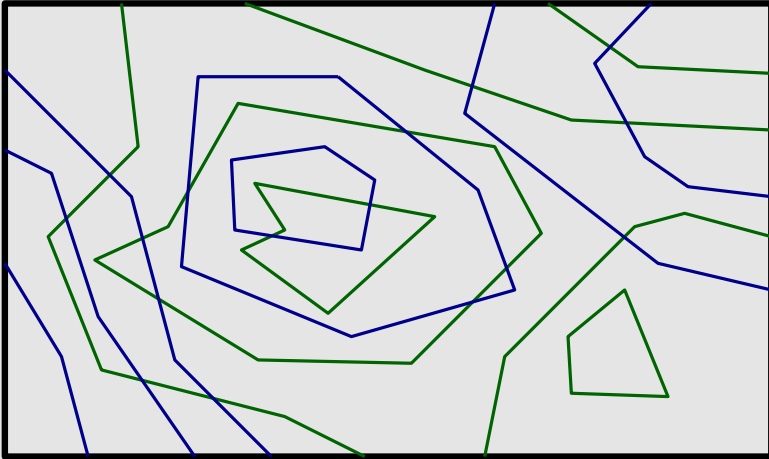
naaldbos

neerslag+vegetatiekaart

=

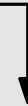


neerslag 650-700 mm  
en naaldbos

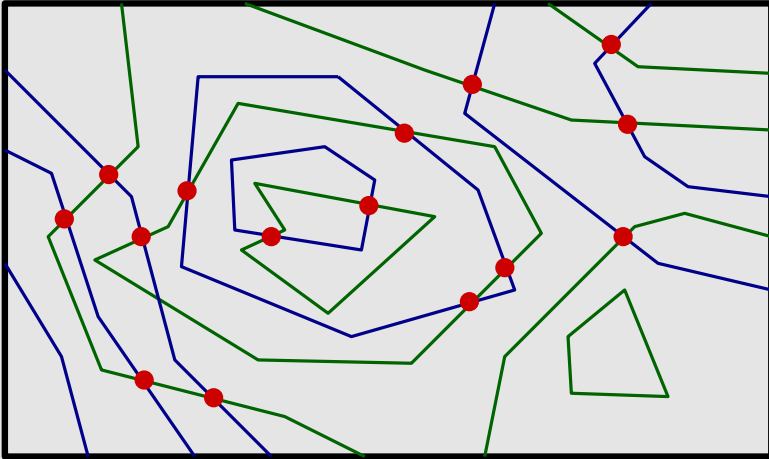


Een eenvoudiger probleem:

bereken alle snijpunten  
tussen de gebiedsgrenzen

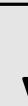


bestaan uit lijnstukken

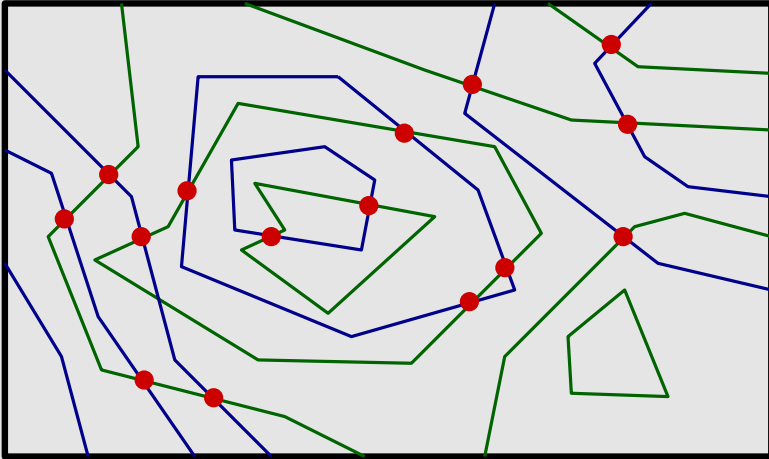


Een eenvoudiger probleem:

bereken alle snijpunten  
tussen de gebiedsgrenzen

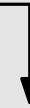


bestaan uit lijnstukken



Een eenvoudiger probleem:

bereken alle snijpunten  
tussen de gebiedsgrenzen



bestaan uit lijnstukken

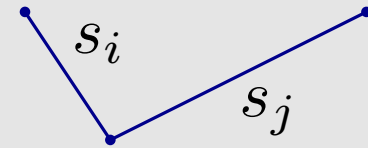
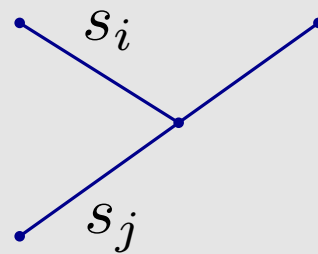
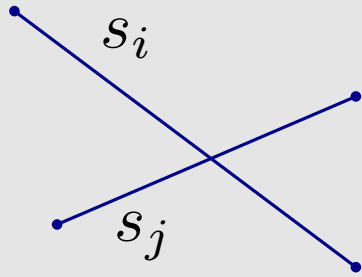
**Invoer:**

lijnstukken  $s_1, \dots, s_n$ , gegeven door coördinaten van eindpunten

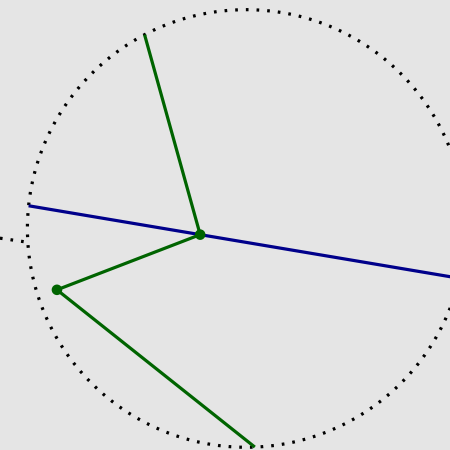
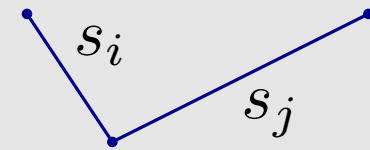
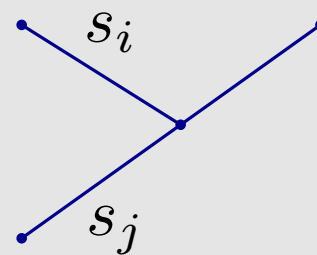
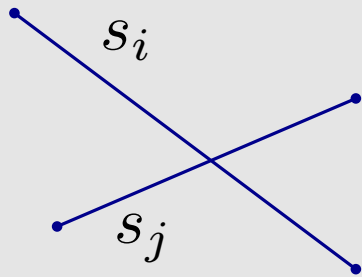
**Uitvoer:**

alle paren  $(s_i, s_j)$  zodat  $s_i$  en  $s_j$  elkaar snijden

Wanneer snijden twee lijnstukken?



Wanneer snijden twee lijnstukken?

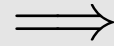




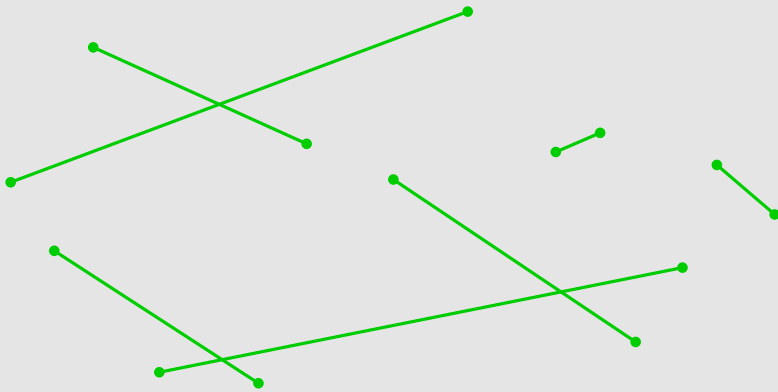




als  $s_i$  helemaal links van  $s_j$  ligt, dan kunnen  $s_i$  en  $s_j$  niet snijden



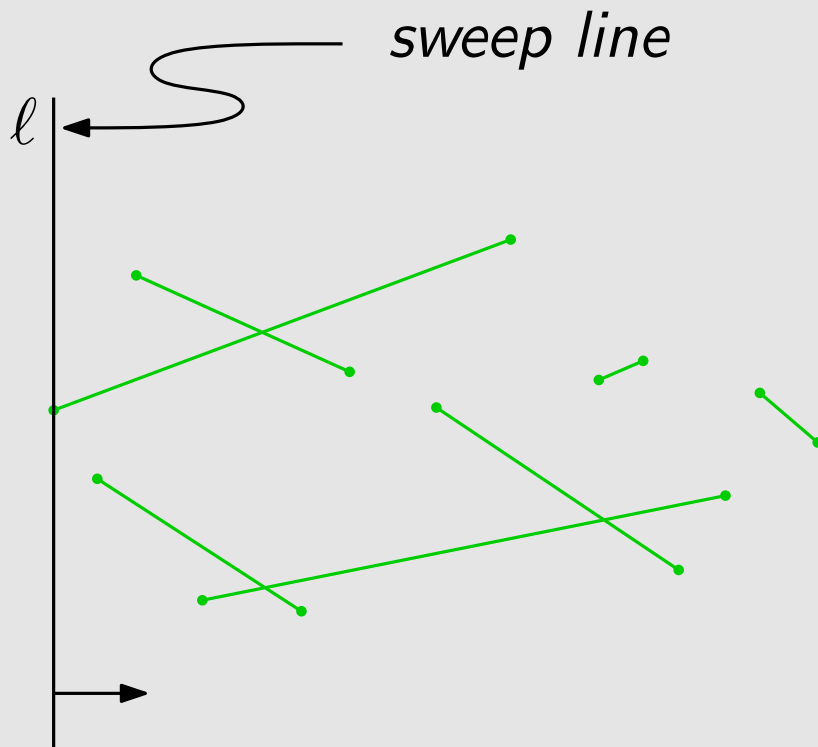
test paren  $(s_i, s_j)$  zdd er een verticale lijn is die  $s_i$  en  $s_j$  snijdt



als  $s_i$  helemaal links van  $s_j$  ligt, dan kunnen  $s_i$  en  $s_j$  niet snijden

$\implies$

test paren  $(s_i, s_j)$  zdd er een verticale lijn is die  $s_i$  en  $s_j$  snijdt

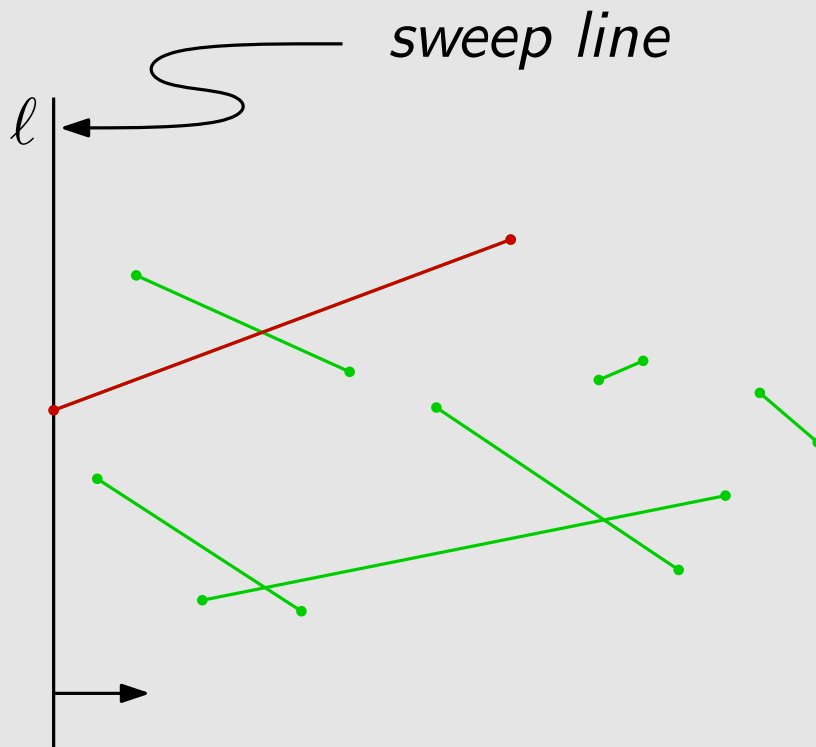


- beweeg sweep line  $\ell$  van links naar rechts
- houd lijnstukken die  $\ell$  snijden bij in lijst  $\mathcal{L}$
- test nieuw lijnstuk in  $\mathcal{L}$  tegen andere lijnstukken in  $\mathcal{L}$

als  $s_i$  helemaal links van  $s_j$  ligt, dan kunnen  $s_i$  en  $s_j$  niet snijden

$\implies$

test paren  $(s_i, s_j)$  zdd er een verticale lijn is die  $s_i$  en  $s_j$  snijdt

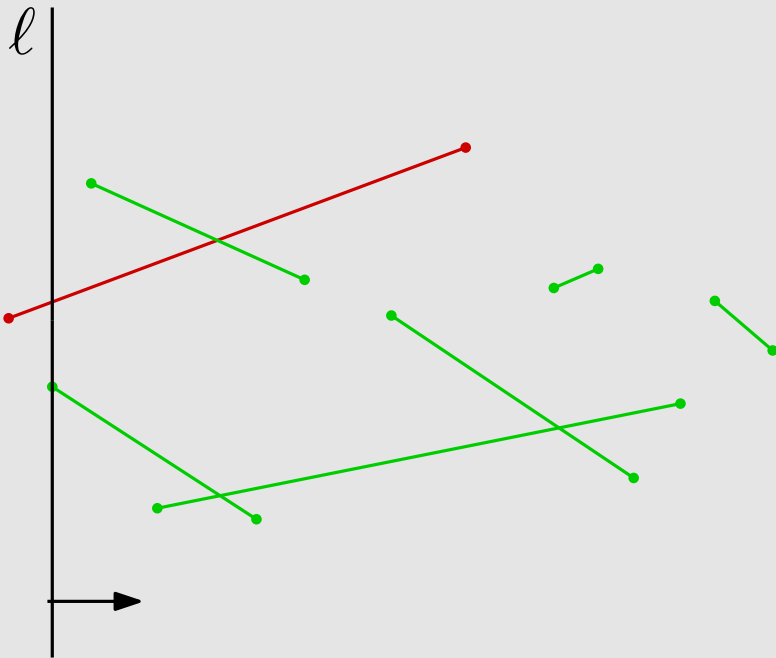


- beweeg sweep line  $\ell$  van links naar rechts
- houd lijnstukken die  $\ell$  snijden bij in lijst  $\mathcal{L}$
- test nieuw lijnstuk in  $\mathcal{L}$  tegen andere lijnstukken in  $\mathcal{L}$

als  $s_i$  helemaal links van  $s_j$  ligt, dan kunnen  $s_i$  en  $s_j$  niet snijden

$\implies$

test paren  $(s_i, s_j)$  zdd er een verticale lijn is die  $s_i$  en  $s_j$  snijdt

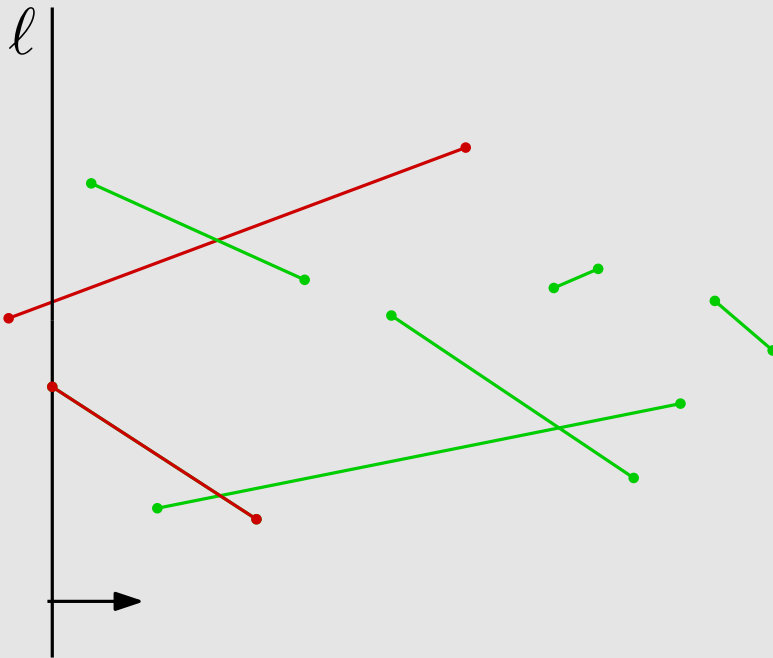


- beweeg sweep line  $\ell$  van links naar rechts
- houd lijnstukken die  $\ell$  snijden bij in lijst  $\mathcal{L}$
- test nieuw lijnstuk in  $\mathcal{L}$  tegen andere lijnstukken in  $\mathcal{L}$

als  $s_i$  helemaal links van  $s_j$  ligt, dan kunnen  $s_i$  en  $s_j$  niet snijden

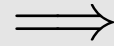
$\implies$

test paren  $(s_i, s_j)$  zdd er een verticale lijn is die  $s_i$  en  $s_j$  snijdt

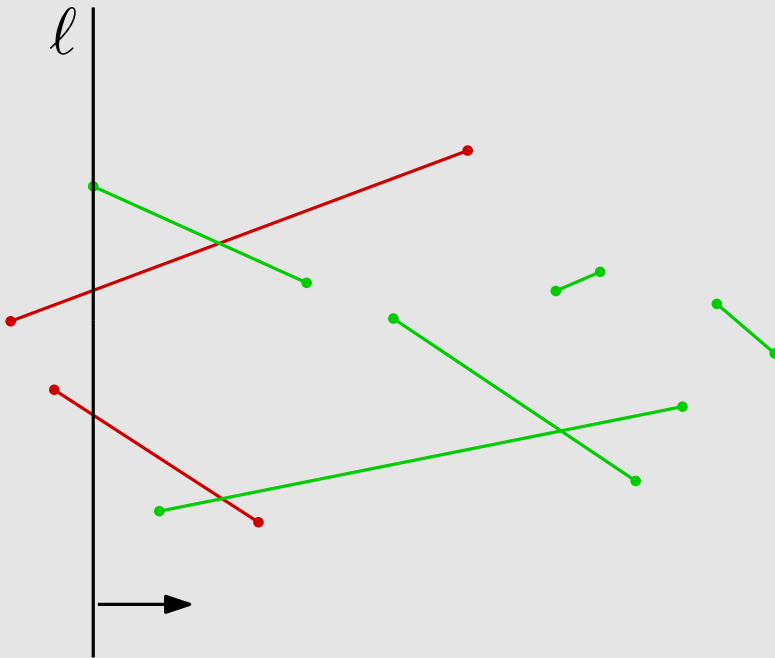


- beweeg sweep line  $\ell$  van links naar rechts
- houd lijnstukken die  $\ell$  snijden bij in lijst  $\mathcal{L}$
- test nieuw lijnstuk in  $\mathcal{L}$  tegen andere lijnstukken in  $\mathcal{L}$

als  $s_i$  helemaal links van  $s_j$  ligt, dan kunnen  $s_i$  en  $s_j$  niet snijden



test paren  $(s_i, s_j)$  zdd er een verticale lijn is die  $s_i$  en  $s_j$  snijdt

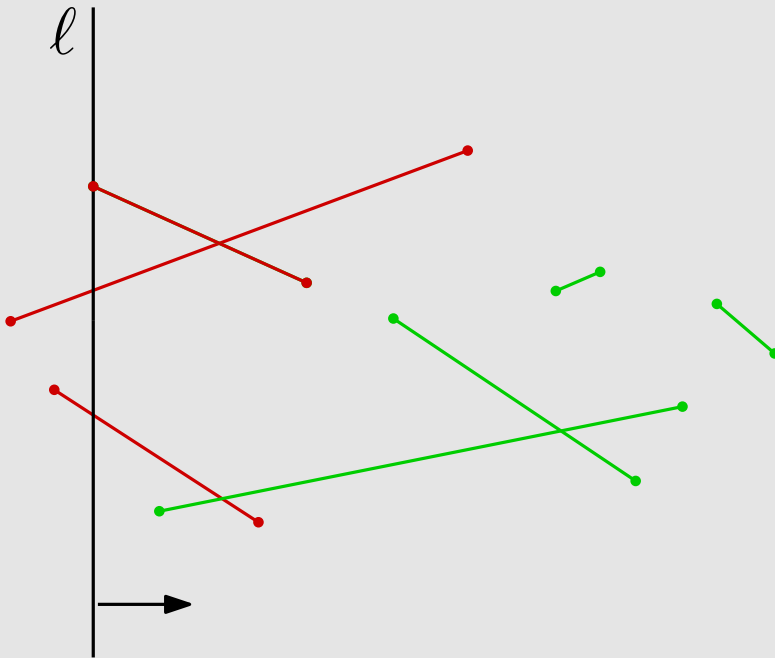


- beweeg sweep line  $\ell$  van links naar rechts
- houd lijnstukken die  $\ell$  snijden bij in lijst  $\mathcal{L}$
- test nieuw lijnstuk in  $\mathcal{L}$  tegen andere lijnstukken in  $\mathcal{L}$

als  $s_i$  helemaal links van  $s_j$  ligt, dan kunnen  $s_i$  en  $s_j$  niet snijden

$\implies$

test paren  $(s_i, s_j)$  zdd er een verticale lijn is die  $s_i$  en  $s_j$  snijdt

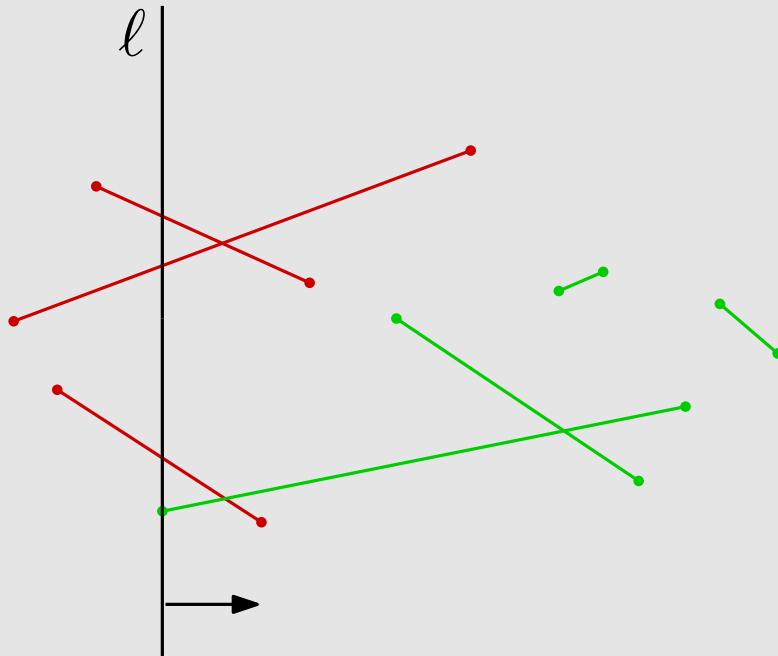


- beweeg sweep line  $\ell$  van links naar rechts
- houd lijnstukken die  $\ell$  snijden bij in lijst  $\mathcal{L}$
- test nieuw lijnstuk in  $\mathcal{L}$  tegen andere lijnstukken in  $\mathcal{L}$

als  $s_i$  helemaal links van  $s_j$  ligt, dan kunnen  $s_i$  en  $s_j$  niet snijden

$\implies$

test paren  $(s_i, s_j)$  zdd er een verticale lijn is die  $s_i$  en  $s_j$  snijdt

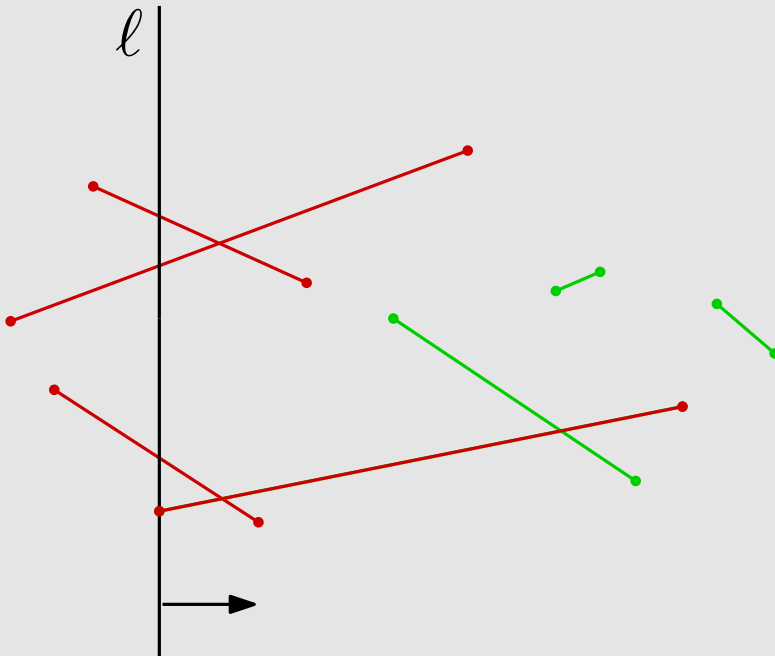


- beweeg sweep line  $\ell$  van links naar rechts
- houd lijnstukken die  $\ell$  snijden bij in lijst  $\mathcal{L}$
- test nieuw lijnstuk in  $\mathcal{L}$  tegen andere lijnstukken in  $\mathcal{L}$

als  $s_i$  helemaal links van  $s_j$  ligt, dan kunnen  $s_i$  en  $s_j$  niet snijden

$\implies$

test paren  $(s_i, s_j)$  zdd er een verticale lijn is die  $s_i$  en  $s_j$  snijdt

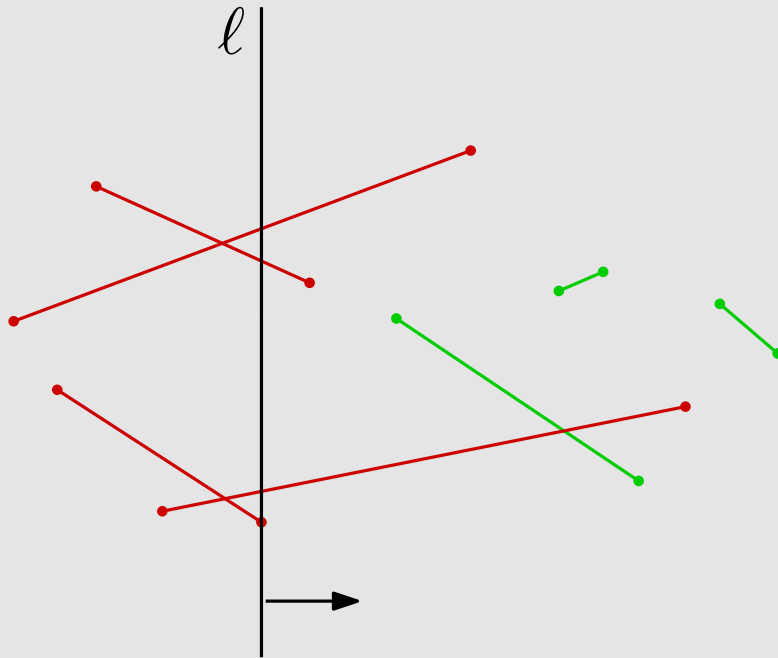


- beweeg sweep line  $\ell$  van links naar rechts
- houd lijnstukken die  $\ell$  snijden bij in lijst  $\mathcal{L}$
- test nieuw lijnstuk in  $\mathcal{L}$  tegen andere lijnstukken in  $\mathcal{L}$

als  $s_i$  helemaal links van  $s_j$  ligt, dan kunnen  $s_i$  en  $s_j$  niet snijden

$\implies$

test paren  $(s_i, s_j)$  zdd er een verticale lijn is die  $s_i$  en  $s_j$  snijdt

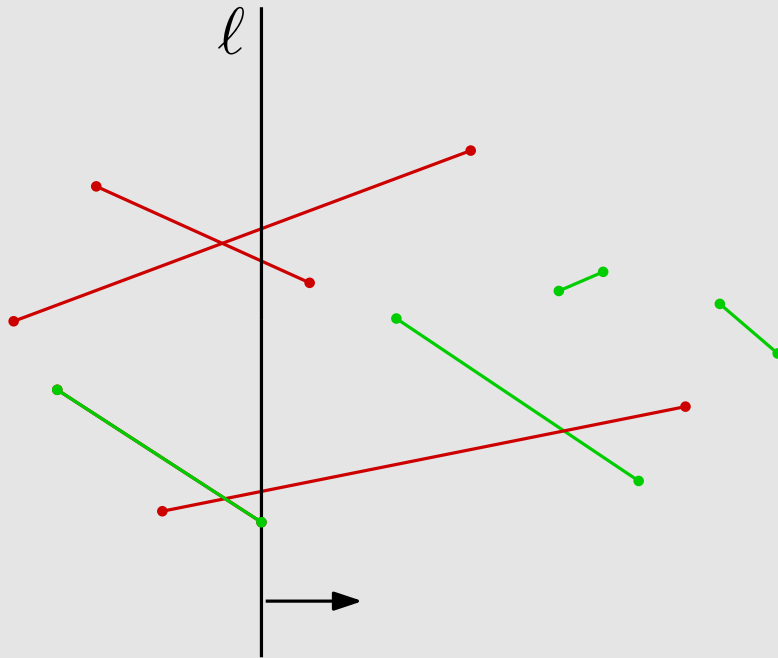


- beweeg sweep line  $\ell$  van links naar rechts
- houd lijnstukken die  $\ell$  snijden bij in lijst  $\mathcal{L}$
- test nieuw lijnstuk in  $\mathcal{L}$  tegen andere lijnstukken in  $\mathcal{L}$

als  $s_i$  helemaal links van  $s_j$  ligt, dan kunnen  $s_i$  en  $s_j$  niet snijden

$\implies$

test paren  $(s_i, s_j)$  zdd er een verticale lijn is die  $s_i$  en  $s_j$  snijdt

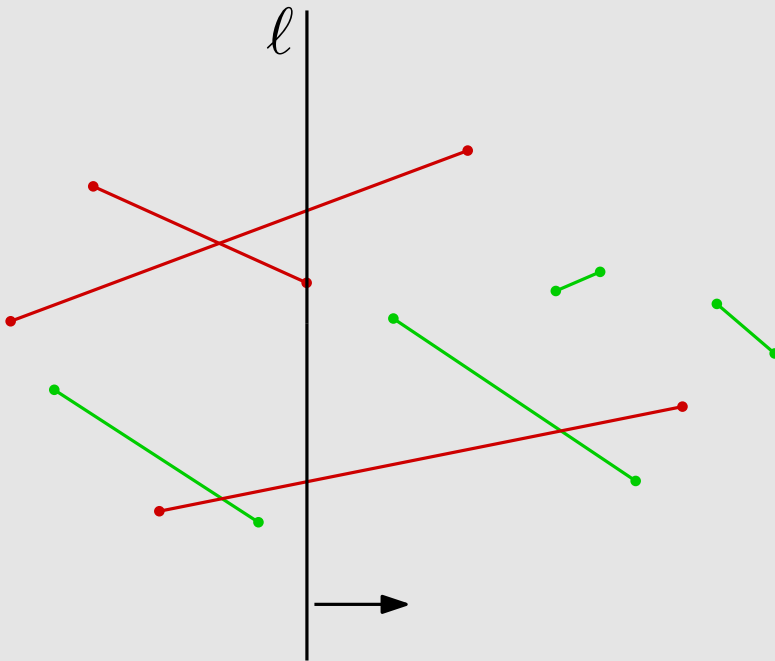


- beweeg sweep line  $\ell$  van links naar rechts
- houd lijnstukken die  $\ell$  snijden bij in lijst  $\mathcal{L}$
- test nieuw lijnstuk in  $\mathcal{L}$  tegen andere lijnstukken in  $\mathcal{L}$

als  $s_i$  helemaal links van  $s_j$  ligt, dan kunnen  $s_i$  en  $s_j$  niet snijden

$\implies$

test paren  $(s_i, s_j)$  zdd er een verticale lijn is die  $s_i$  en  $s_j$  snijdt

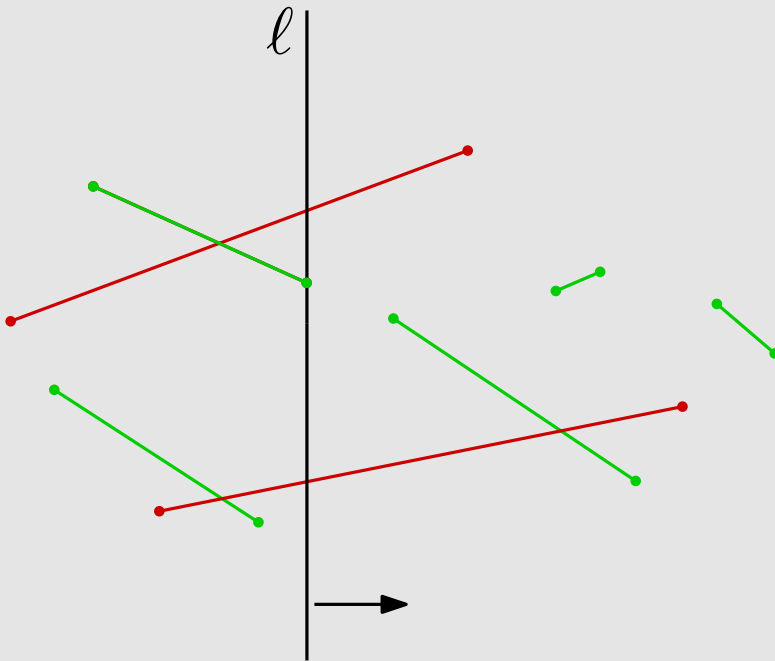


- beweeg sweep line  $\ell$  van links naar rechts
- houd lijnstukken die  $\ell$  snijden bij in lijst  $\mathcal{L}$
- test nieuw lijnstuk in  $\mathcal{L}$  tegen andere lijnstukken in  $\mathcal{L}$

als  $s_i$  helemaal links van  $s_j$  ligt, dan kunnen  $s_i$  en  $s_j$  niet snijden

$\implies$

test paren  $(s_i, s_j)$  zdd er een verticale lijn is die  $s_i$  en  $s_j$  snijdt

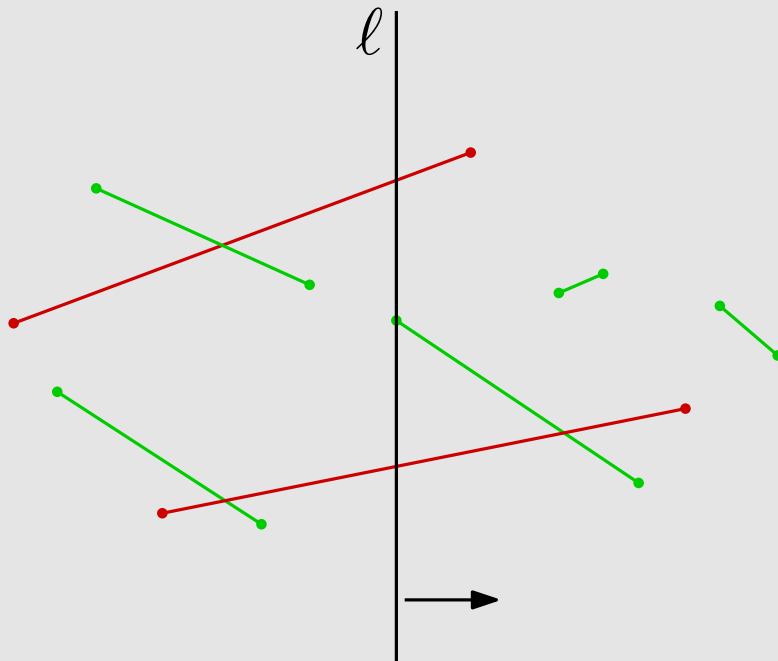


- beweeg sweep line  $\ell$  van links naar rechts
- houd lijnstukken die  $\ell$  snijden bij in lijst  $\mathcal{L}$
- test nieuw lijnstuk in  $\mathcal{L}$  tegen andere lijnstukken in  $\mathcal{L}$

als  $s_i$  helemaal links van  $s_j$  ligt, dan kunnen  $s_i$  en  $s_j$  niet snijden

$\implies$

test paren  $(s_i, s_j)$  zdd er een verticale lijn is die  $s_i$  en  $s_j$  snijdt

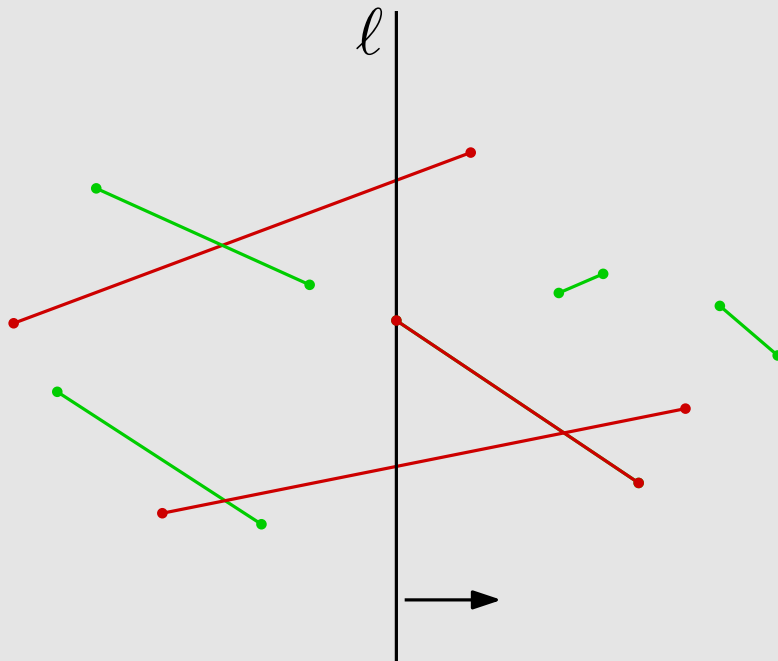


- beweeg sweep line  $\ell$  van links naar rechts
- houd lijnstukken die  $\ell$  snijden bij in lijst  $\mathcal{L}$
- test nieuw lijnstuk in  $\mathcal{L}$  tegen andere lijnstukken in  $\mathcal{L}$

als  $s_i$  helemaal links van  $s_j$  ligt, dan kunnen  $s_i$  en  $s_j$  niet snijden

$\implies$

test paren  $(s_i, s_j)$  zdd er een verticale lijn is die  $s_i$  en  $s_j$  snijdt



- beweeg sweep line  $\ell$  van links naar rechts
- houd lijnstukken die  $\ell$  snijden bij in lijst  $\mathcal{L}$
- test nieuw lijnstuk in  $\mathcal{L}$  tegen andere lijnstukken in  $\mathcal{L}$

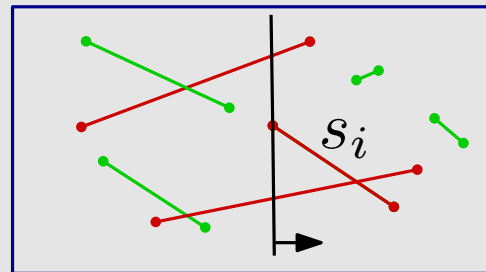
**Plane sweep algoritme:** Beweeg sweep line  $\ell$  van links naar rechts en houd lijnstukken die  $\ell$  snijden bij in lijst  $\mathcal{L}$ .

**Plane sweep algoritme:** Beweeg sweep line  $\ell$  van links naar rechts en houd lijnstukken die  $\ell$  snijden bij in lijst  $\mathcal{L}$ .

**Events:**

**Linker eindpunt:**

- test bijbehorend lijnstuk  $s_i$  met andere lijnstukken in  $\mathcal{L}$
- rapporteer snijdende paren
- voeg  $s_i$  toe aan  $\mathcal{L}$

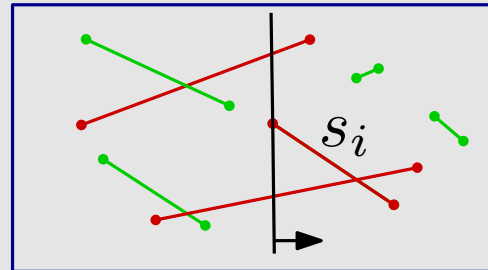


**Plane sweep algoritme:** Beweeg sweep line  $\ell$  van links naar rechts en houd lijnstukken die  $\ell$  snijden bij in lijst  $\mathcal{L}$ .

**Events:**

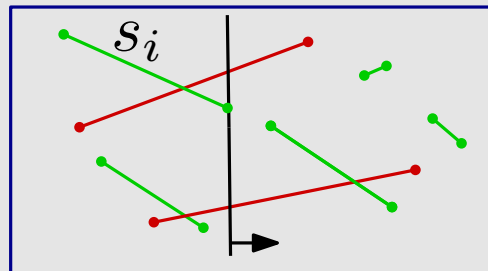
**Linker eindpunt:**

- test bijbehorend lijnstuk  $s_i$  met andere lijnstukken in  $\mathcal{L}$
- rapporteer snijdende paren
- voeg  $s_i$  toe aan  $\mathcal{L}$



**Rechter eindpunt:**

- verwijder  $s_i$  uit  $\mathcal{L}$



**Algoritme** *PlaneSweep1*

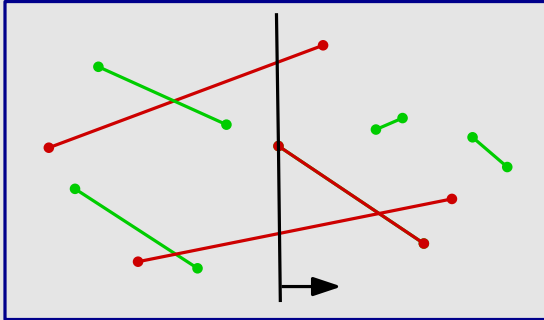
1. Sorteert de eindpunten op  $x$ -coördinaat, van klein naar groot.
2. Laat  $p_1, p_2, \dots, p_{2n}$  de gesorteerde lijst zijn.
3. Initialiseer een lege gelinkte lijst  $\mathcal{L}$ .
4. **for**  $t := 1$  **to**  $2n$
5.     **do** Laat  $s_i$  het lijnstuk zijn waarvan  $p_t$  eindpunt is
6.         **if**  $p_t$  is linkereindpunt van  $s_i$
7.             **then** { **for** alle lijnstukken  $s_j$  in de lijst  $\mathcal{L}$
8.                 **do if**  $s_i$  snijdt  $s_j$  **then** rapporteer paar  $(s_i, s_j)$
9.                 voeg  $s_i$  toe aan  $\mathcal{L}$      }
10.         **else** verwijder  $s_i$  uit  $\mathcal{L}$

**Algoritme** *PlaneSweep1*

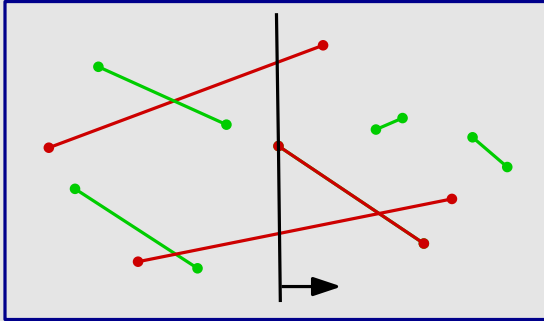
1. Sorteert de eindpunten op  $x$ -coördinaat, van klein naar groot.
2. Laat  $p_1, p_2, \dots, p_{2n}$  de gesorteerde lijst zijn.
3. Initialiseer een lege gelinkte lijst  $\mathcal{L}$ .
4. **for**  $t := 1$  **to**  $2n$
5.     **do** Laat  $s_i$  het lijnstuk zijn waarvan  $p_t$  eindpunt is
6.         **if**  $p_t$  is linkereindpunt van  $s_i$
7.             **then** { **for** alle lijnstukken  $s_j$  in de lijst  $\mathcal{L}$
8.                 **do if**  $s_i$  snijdt  $s_j$  **then** rapporteer paar  $(s_i, s_j)$
9.                 voeg  $s_i$  toe aan  $\mathcal{L}$      }
10.         **else** verwijder  $s_i$  uit  $\mathcal{L}$

wat te doen met

- verticale lijnstukken?
- eindpunten met gelijke  $x$ -coördinaat?



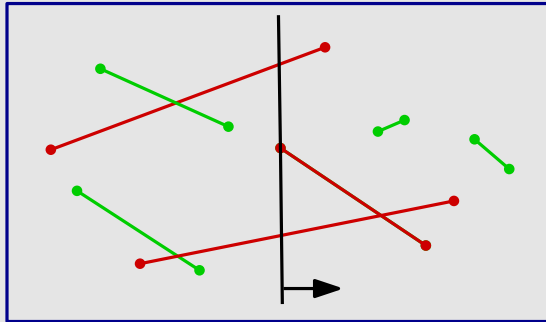
En heeft het geholpen?



En heeft het geholpen?

soms wel ...





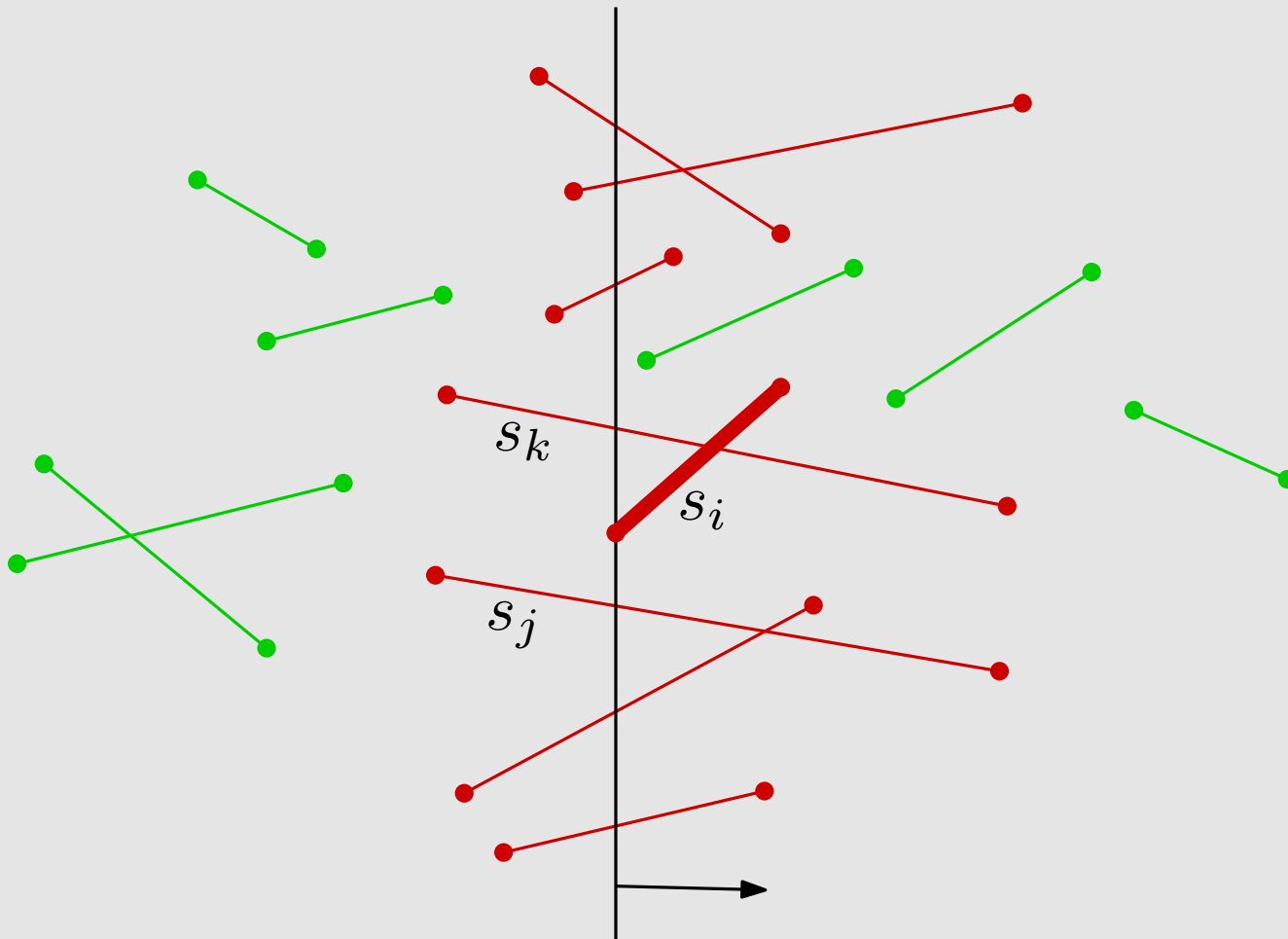
En heeft het geholpen?

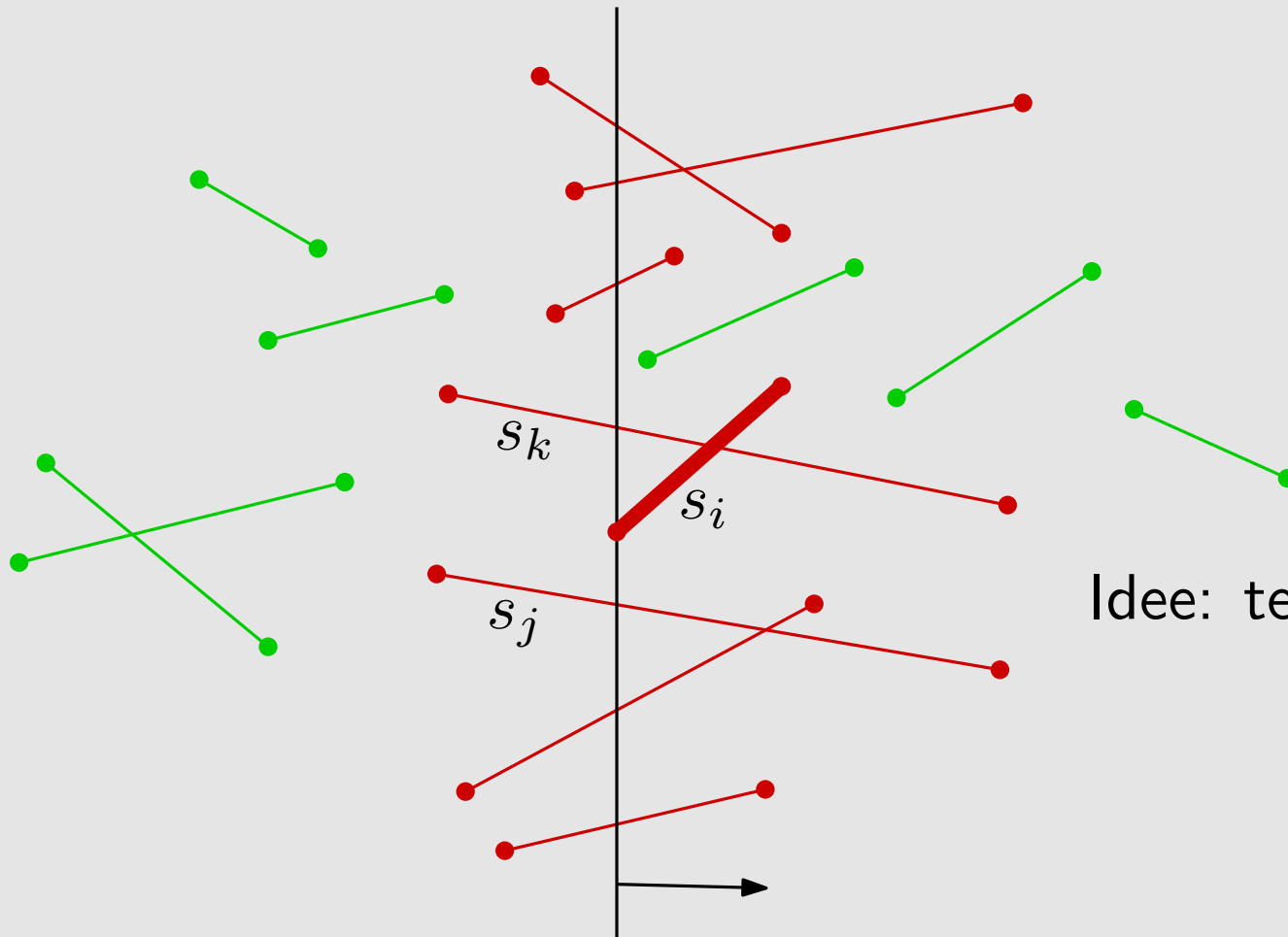
soms wel ...



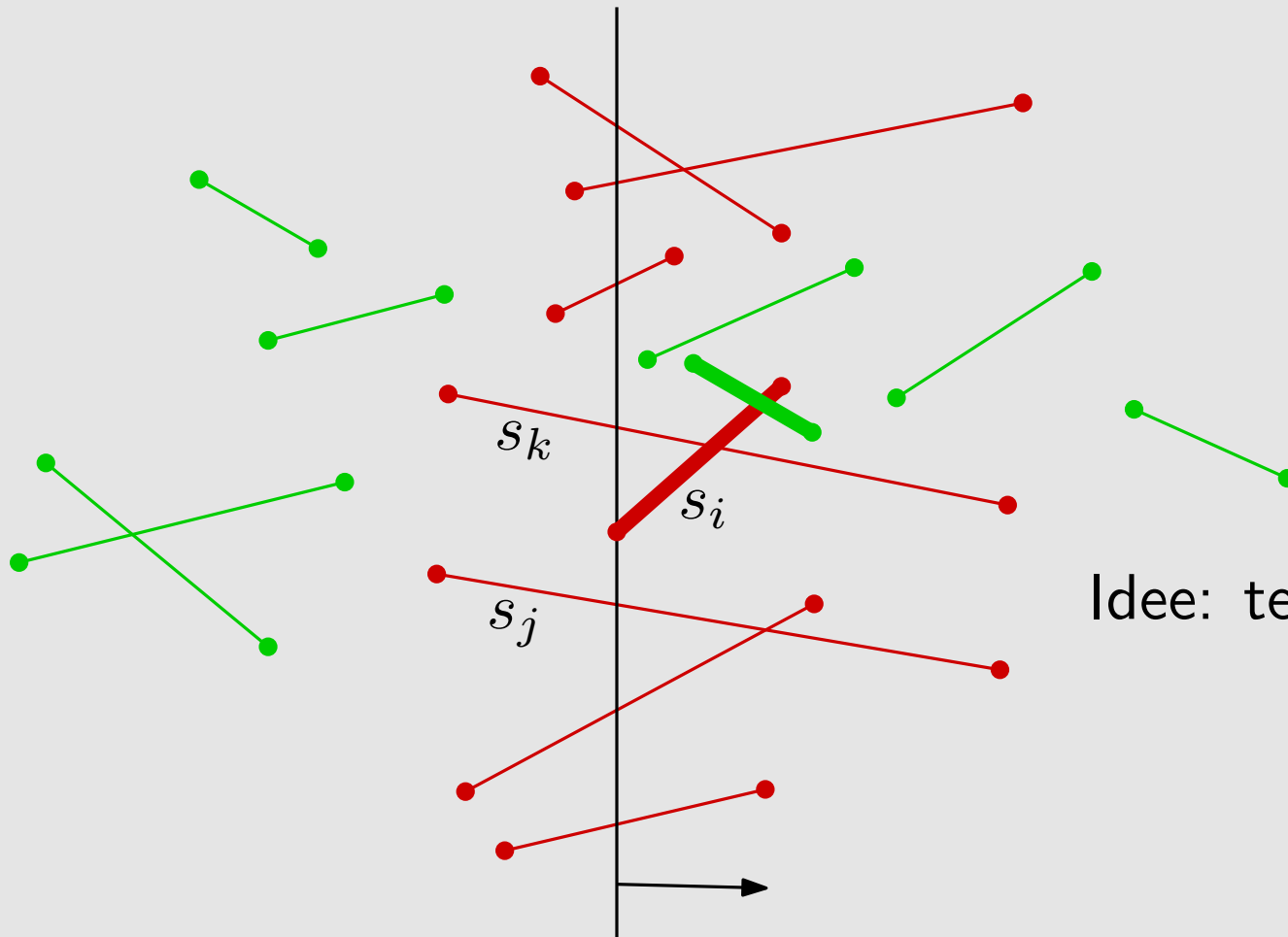
... en soms niet



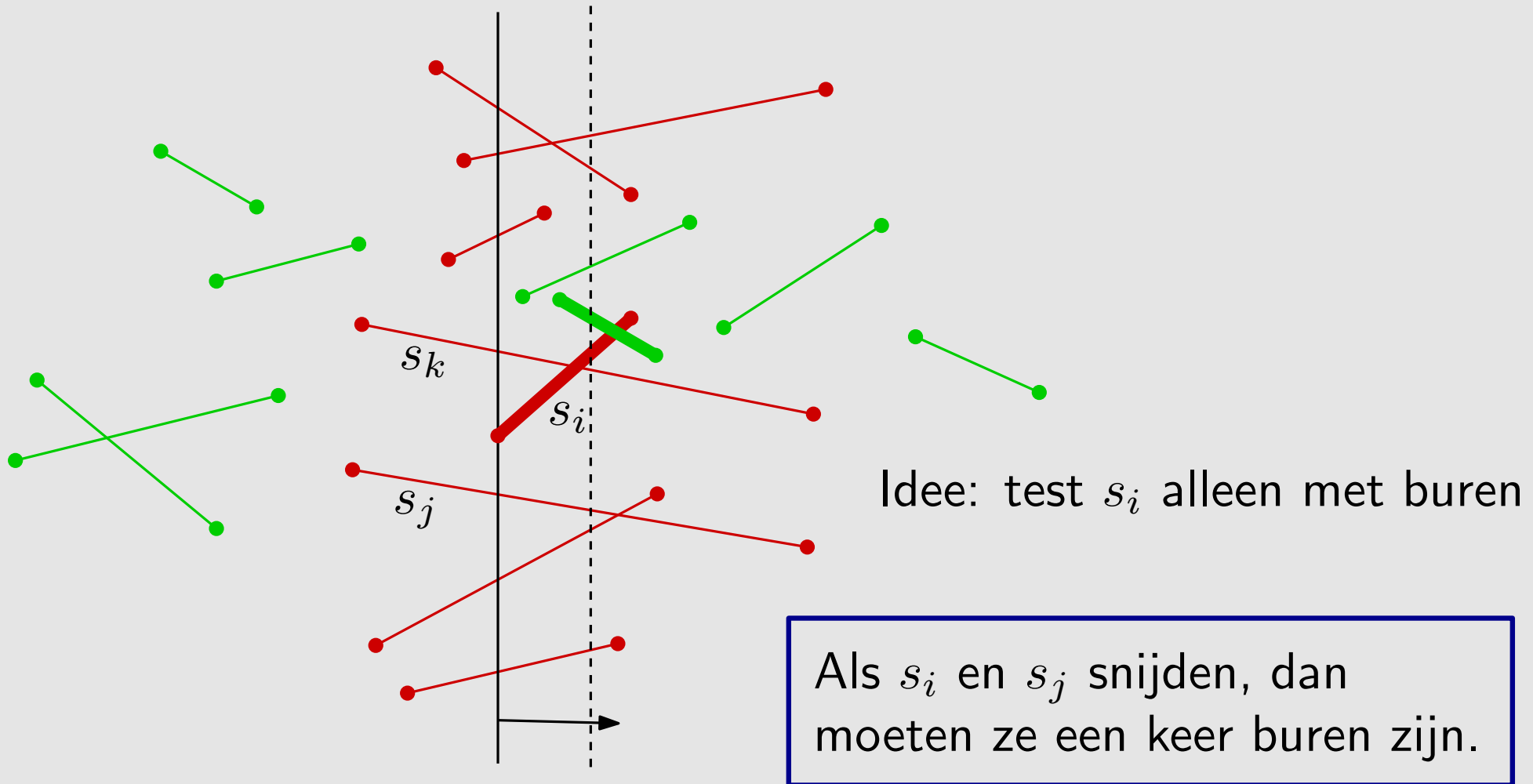


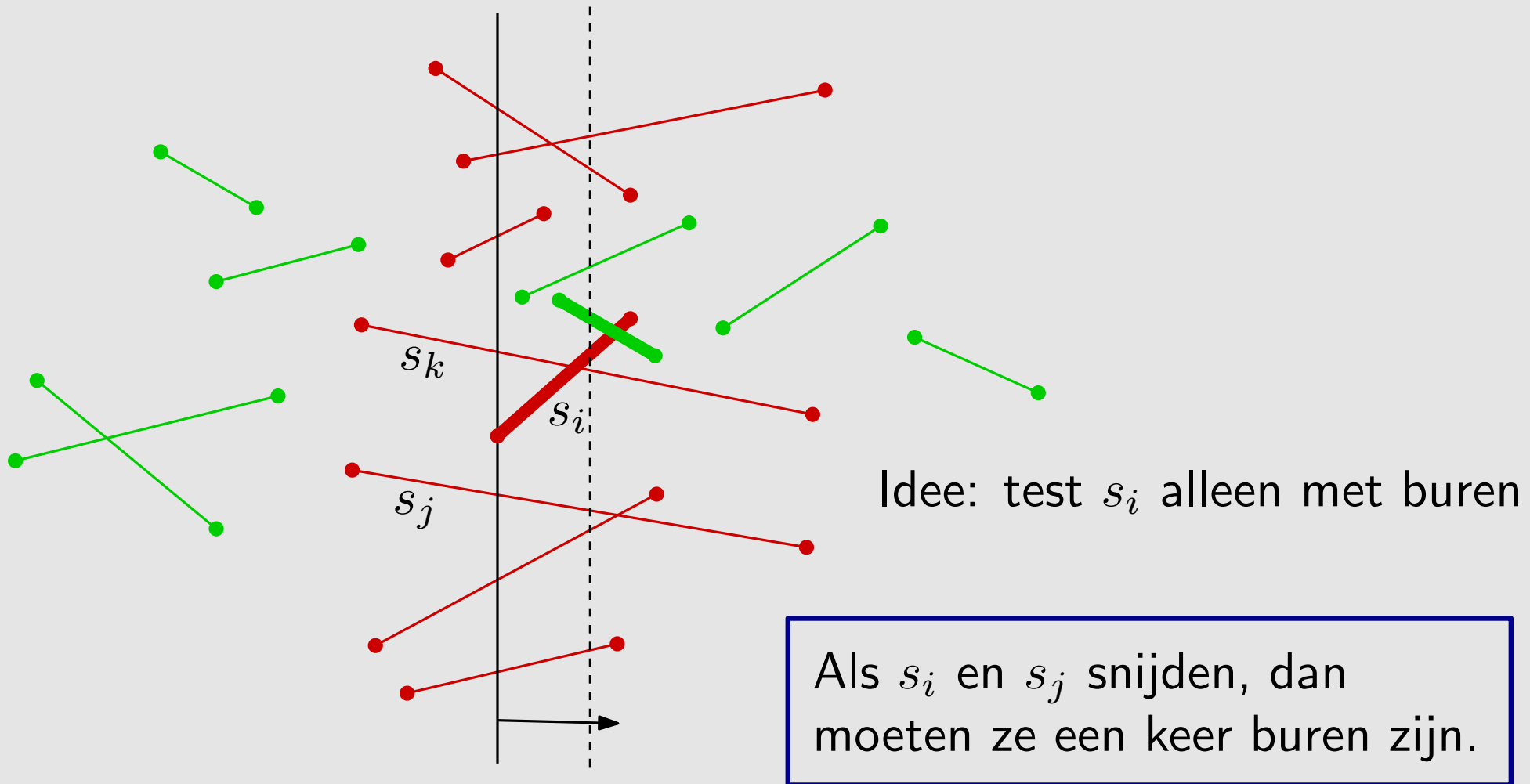


Idee: test  $s_i$  alleen met buren



Idee: test  $s_i$  alleen met buren

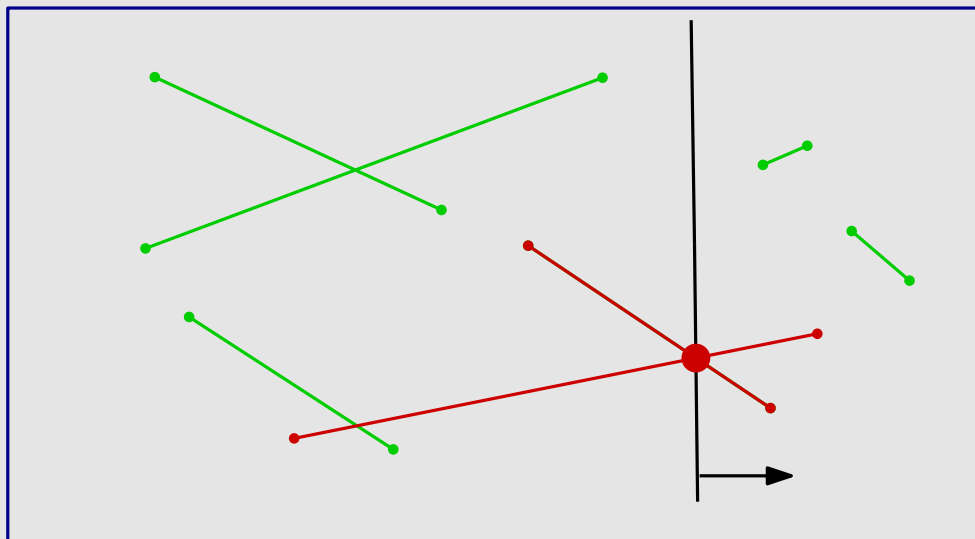




**Nieuw plane sweep algoritme:** Beweeg sweep line  $\ell$  van links naar rechts en houd lijnstukken die  $\ell$  snijden bij in **gesorteerde** lijst  $\mathcal{L}$ .

**Nieuw plane sweep algoritme:** Beweeg sweep line  $\ell$  van links naar rechts en houd lijnstukken die  $\ell$  snijden bij in **gesorteerde** lijst  $\mathcal{L}$ .

volgorde verandert bij snijpunten



drie typen events:

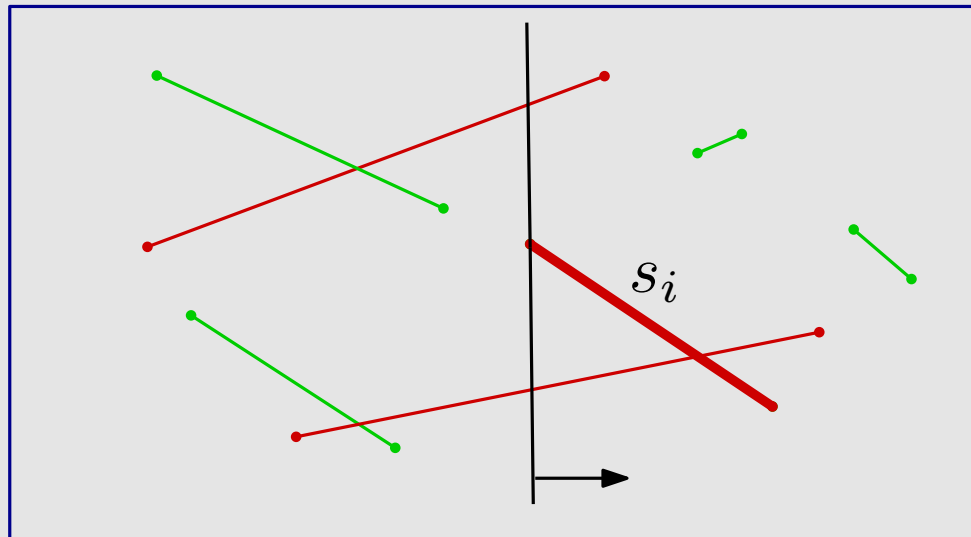
- linker eindpunten
- rechter eindpunten
- snijpunten

**Algoritme** *PlaneSweep2*

1. Zet de  $2n$  eindpunten in een event queue  $Q$ , gesorteerd op  $x$ -coördinaat.
3. Initialiseer een lege lijst  $\mathcal{L}$ .
4. **while**  $Q$  niet leeg
5.     **do** { Haal volgende event punt  $p$  uit  $Q$ .
6.         **if**  $p$  is linker eindpunt **then** *BehandelLinkerEindpunt*( $p$ )
7.         **if**  $p$  is rechter eindpunt **then** *BehandelRechterEindpunt*( $p$ )
8.         **if**  $p$  is snijpunt **then** *BehandelSnijpunt*( $p$ ) }  
}

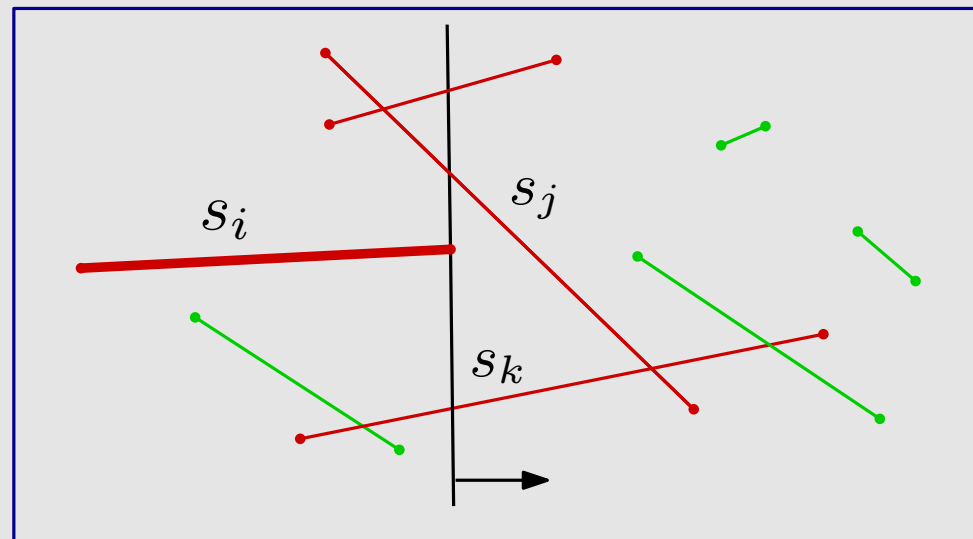
Linker eindpunt:

- voeg  $s_i$  toe in  $\mathcal{L}$  op de juiste positie
- test bijbehorend lijnstuk  $s_i$  met buren in  $\mathcal{L}$
- rapporteer snijdende paren
- voeg snijpunten toe aan event queue  $Q$



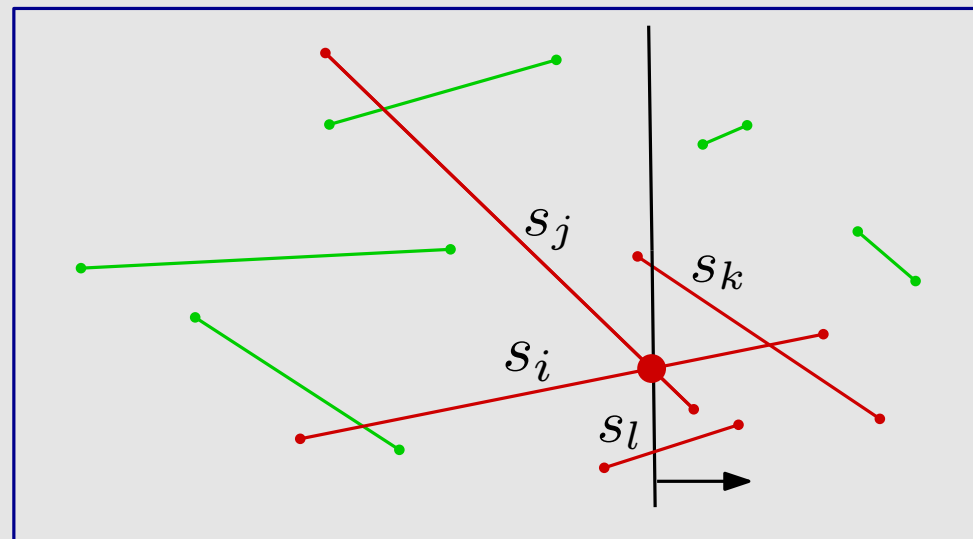
## Rechter eindpunt:

- verwijder  $s_i$  uit  $\mathcal{L}$
- test of oude buren van  $s_i$  elkaar snijden;
- rapporteer snijdende paren
- voeg snijpunten toe aan event queue  $Q$

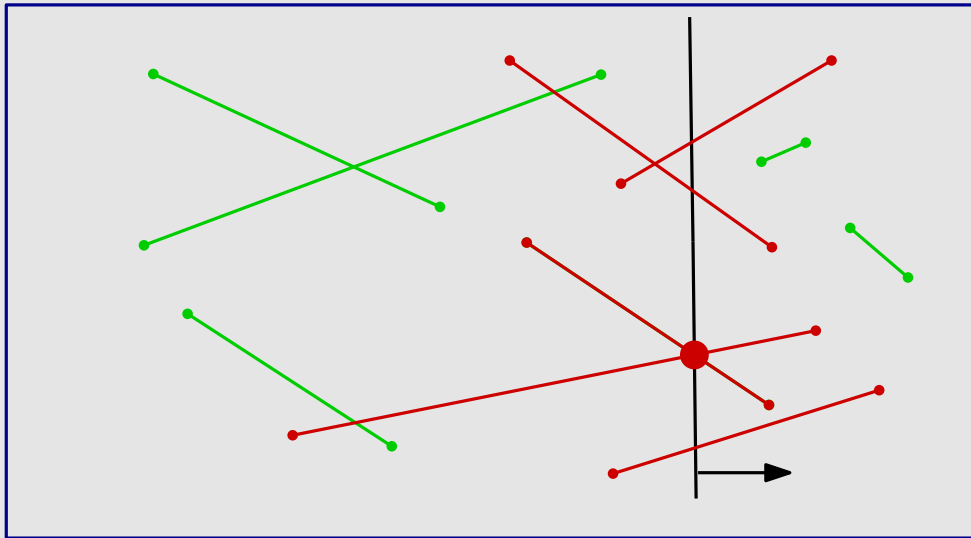


## Snijpunt:

- verwissel snijdende lijnstukken in  $\mathcal{L}$
- test nieuwe buurparen of ze snijden
- rapporteer snijdende paren
- voeg snijpunten toe aan event queue  $Q$



**Plane sweep algoritme:** Beweeg sweep line  $\ell$  van links naar rechts en houd lijnstukken die  $\ell$  snijden bij in **gesorteerde** lijst  $\mathcal{L}$ .



operaties op  $\mathcal{L}$ :

- voeg lijnstuk in
- verwijder lijnstuk
- vind buren
- verwissel buren

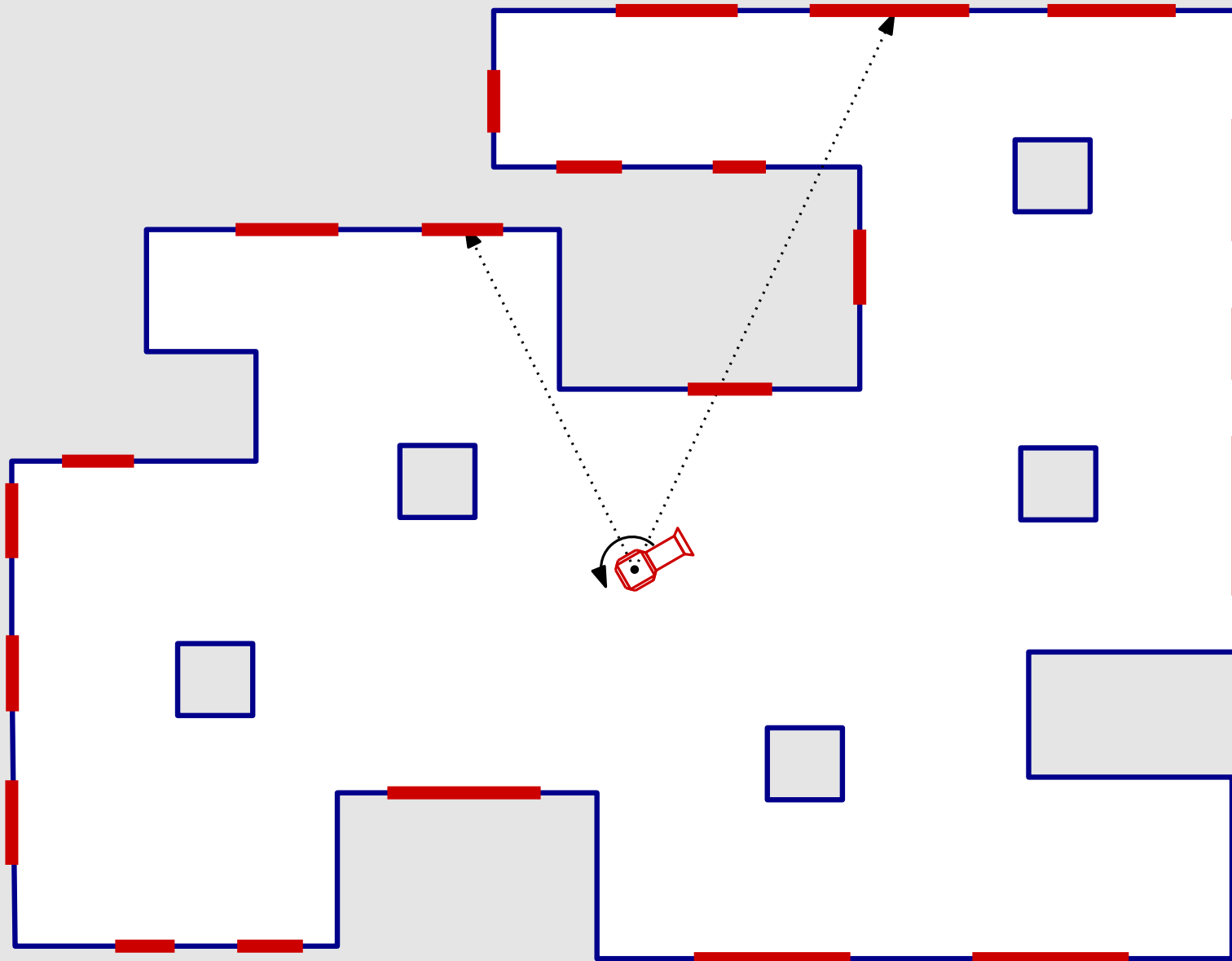
gebruik gebalanceerde boom in plaats van gesorteerde lijst

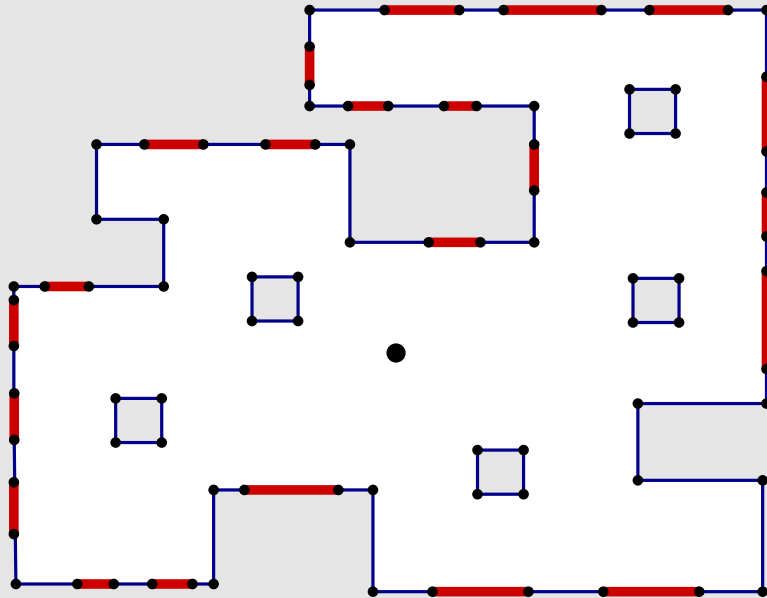
$\implies$  operaties in  $O(\log n)$  tijd

$\implies$  totale tijd  $O((n + k) \log n)$  met  $k = \#$  snijpunten

Lijnstuk intersectie:

- naïef algoritme:  $O(n^2)$  tijd, ook bij weinig snijpunten
- plane sweep:  $O(n \log n)$  bij weinig snijpunten  
(algemeen:  $O((n + k) \log n)$ )



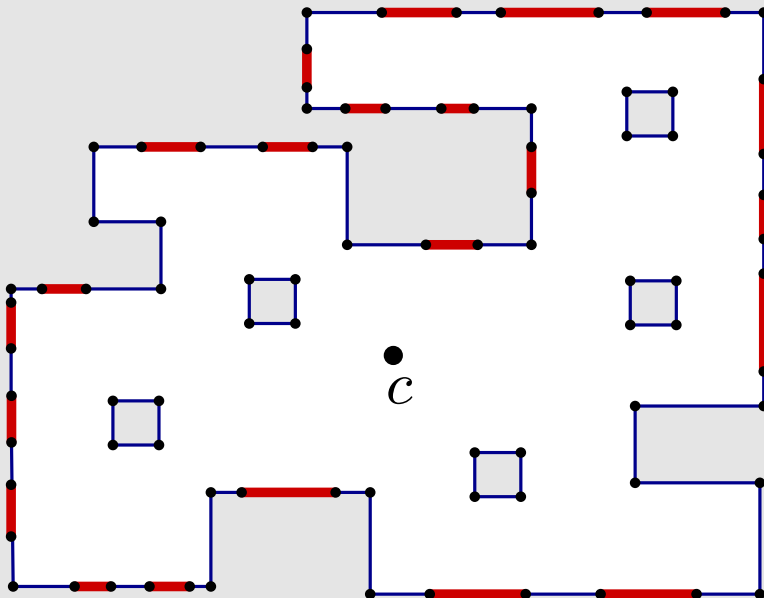


$S$ :  $n$  lijnstukken die elkaar niet snijden  
(wel: gemeenschappelijke eindpunten)

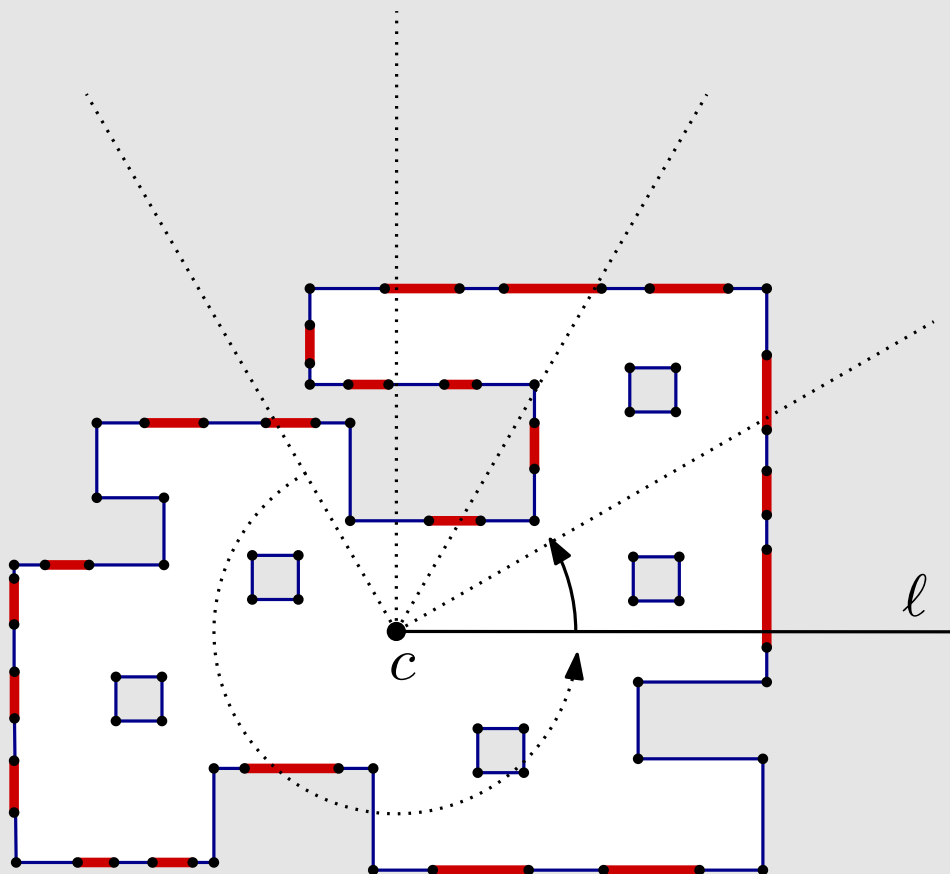
$c$ : een punt (de bewakingscamera)

Rapporteer alle lijnstukken in  $S$  die (geheel of gedeeltelijk) zichtbaar zijn vanuit  $c$ .

Rapporteer alle lijnstukken in  $S$  die (geheel of gedeeltelijk) zichtbaar zijn vanuit  $c$ .



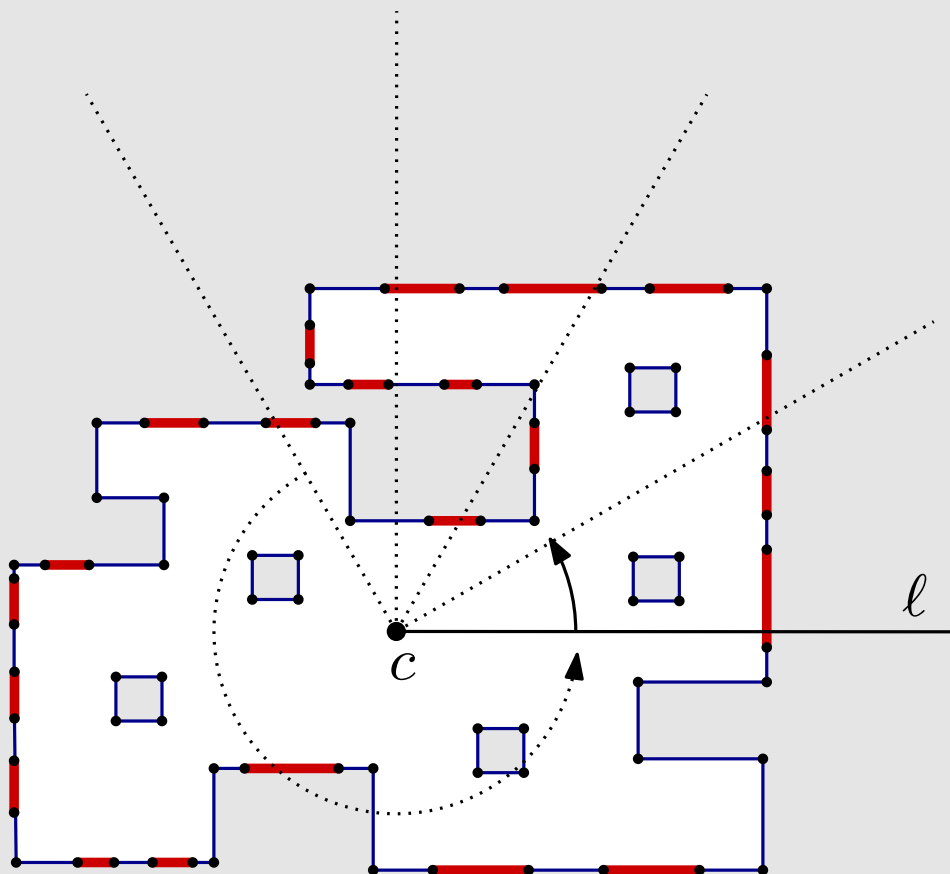
Rapporteer alle lijnstukken in  $S$  die (geheel of gedeeltelijk) zichtbaar zijn vanuit  $c$ .



“Draaiend” sweep line algoritme.

Houd lijnstukken die  $\ell$  snijden bij

Rapporteer alle lijnstukken in  $S$  die (geheel of gedeeltelijk) zichtbaar zijn vanuit  $c$ .

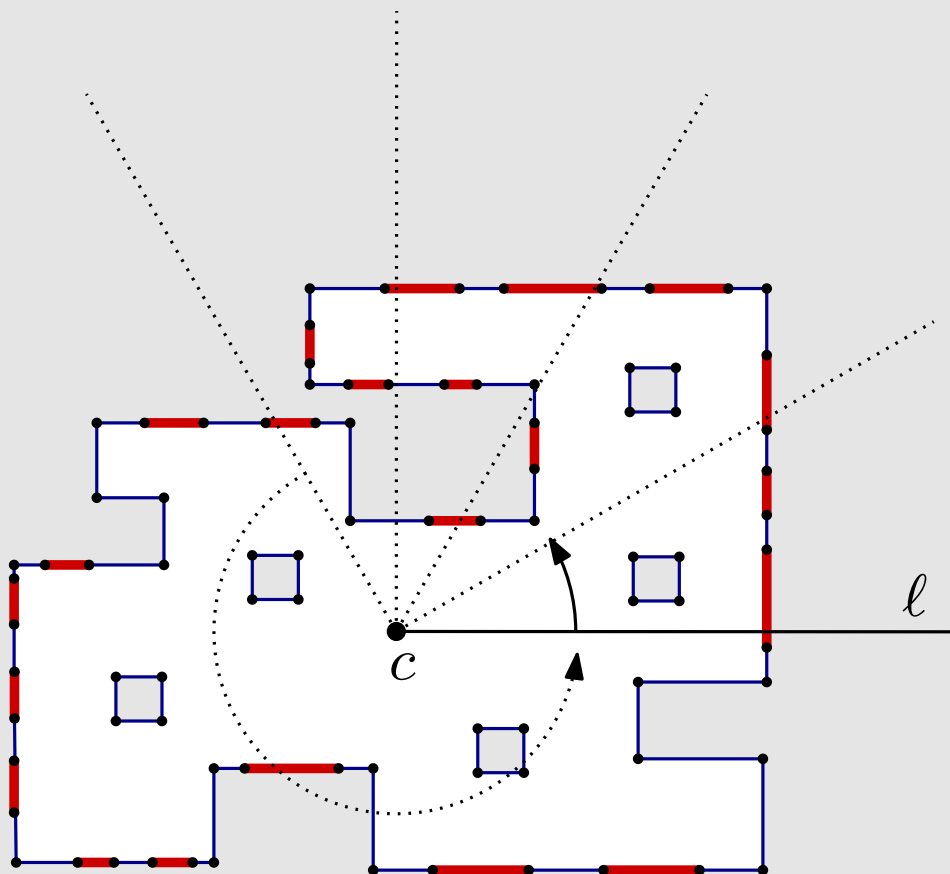


“Draaiend” sweep line algoritme.

Houd lijnstukken die  $\ell$  snijden bij

Events: eindpunten

Rapporteer alle lijnstukken in  $S$  die (geheel of gedeeltelijk) zichtbaar zijn vanuit  $c$ .



“Draaiend” sweep line algoritme.

Houd lijnstukken die  $l$  snijden bij

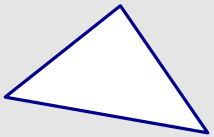
Events: eindpunten

Actie bij event:

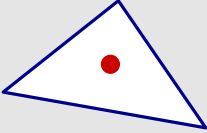
- invoegen / verwijderen van lijnstukken
- rapporteer voorste lijnstuk

Hoeveel camera's zijn maximaal nodig om een veelhoek met  $n$  hoekpunten helemaal te bewaken?

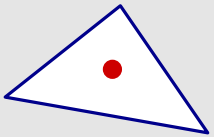
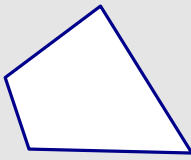
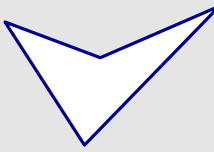
Hoeveel camera's zijn maximaal nodig om een veelhoek met  $n$  hoekpunten helemaal te bewaken?

$n$	voorbeelden	max aantal camera's
3		
4		
6		
$n$		

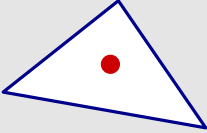
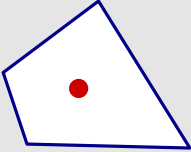
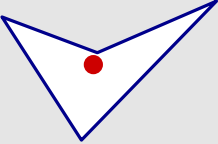
Hoeveel camera's zijn maximaal nodig om een veelhoek met  $n$  hoekpunten helemaal te bewaken?

$n$	voorbeelden	max aantal camera's
3		1
4		
6		
$n$		

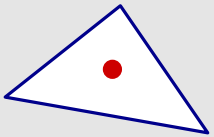
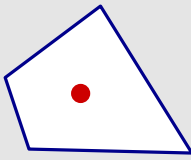
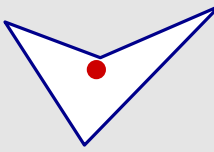
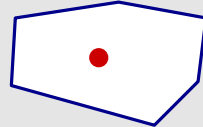
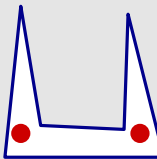
Hoeveel camera's zijn maximaal nodig om een veelhoek met  $n$  hoekpunten helemaal te bewaken?

$n$	voorbeelden	max aantal camera's
3		1
4	 of 	
6		
$n$		

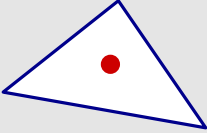
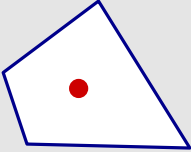
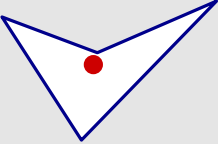
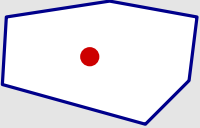
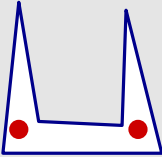
Hoeveel camera's zijn maximaal nodig om een veelhoek met  $n$  hoekpunten helemaal te bewaken?

$n$	voorbeelden	max aantal camera's
3		1
4	 of 	1
6		
$n$		

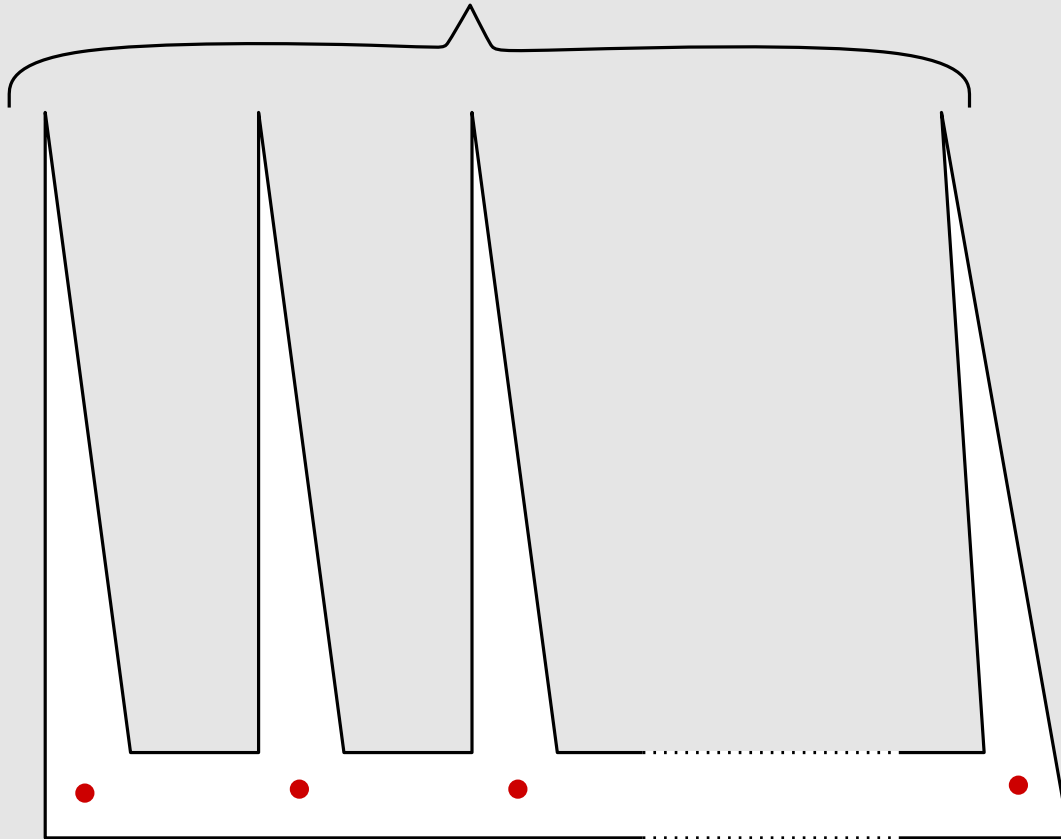
Hoeveel camera's zijn maximaal nodig om een veelhoek met  $n$  hoekpunten helemaal te bewaken?

$n$	voorbeelden	max aantal camera's
3		1
4	 of 	1
6	 of 	2
$n$		

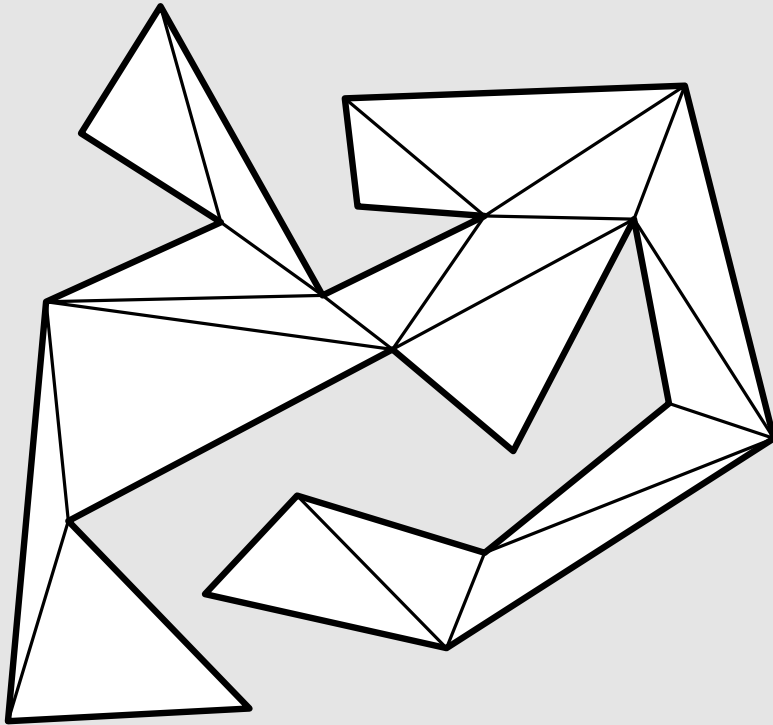
Hoeveel camera's zijn maximaal nodig om een veelhoek met  $n$  hoekpunten helemaal te bewaken?

$n$	voorbeelden	max aantal camera's
3		1
4	 of 	1
6	 of 	2
$n$	? of ? of ...	algemene formule: $n, \sqrt{n}, n/10, \dots?$

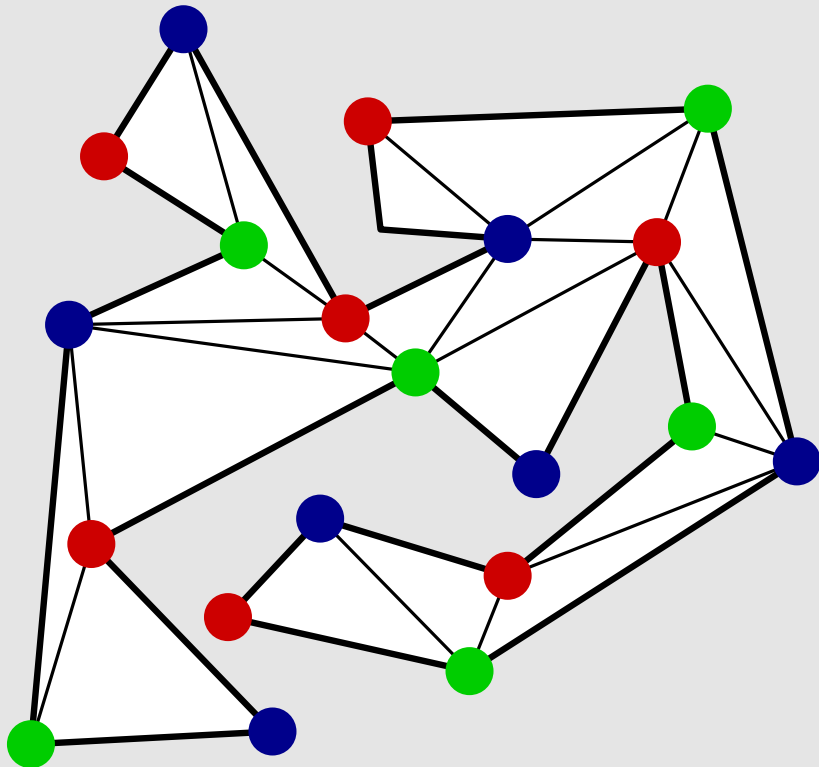
$n/3$  "punten"



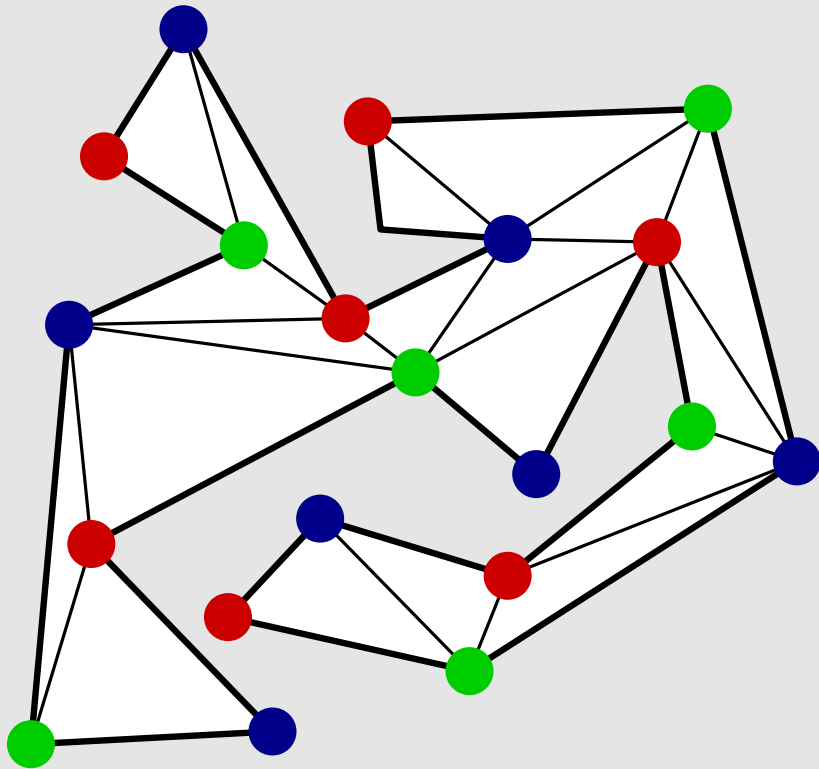
voor sommige  $n$ -hoeken  
heb je  $n/3$  camera's nodig



- elke  $n$ -hoek kan in  $n - 2$  driehoeken opgedeeld worden



- elke  $n$ -hoek kan in  $n - 2$  driehoeken opgedeeld worden
- hoekpunten kunnen altijd rood/groen/blauw gekleurd worden zodat elke driehoek drie kleuren heeft



- elke  $n$ -hoek kan in  $n - 2$  driehoeken opgedeeld worden
- hoekpunten kunnen altijd rood/groen/blauw gekleurd worden zodat elke driehoek drie kleuren heeft
- plaats camera's op minst gebruikte kleur

Soms heb je  $n/3$  camera's nodig, en aan  $n/3$  camera's heb je altijd genoeg.