

Algorithmische Geometrie

Voronoi-Diagramme – Fortunes Algorithmus

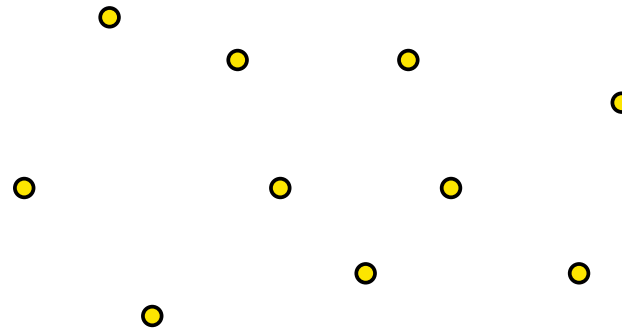


Wer benutzt welchen Briefkasten?

Situation

- gegeben: Menge S von Standorten
- Welche Punkte liegen am nächsten zu welchem Standort?

(bei Euklidischer Metrik)

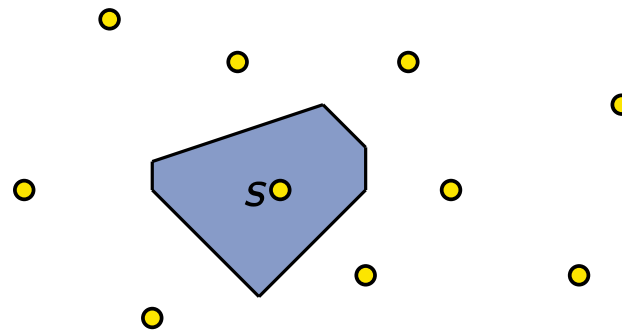


Wer benutzt welchen Briefkasten?

Situation

- gegeben: Menge S von Standorten
- Welche Punkte liegen am nächsten zu welchem Standort?

(bei Euklidischer Metrik)



Voronoi-Diagramm

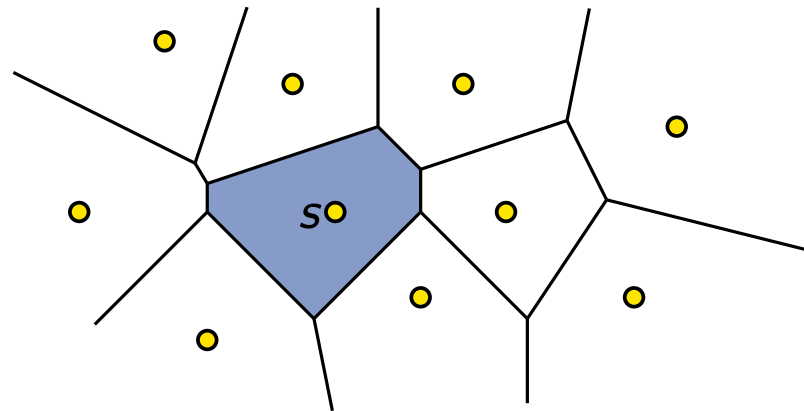
- **Voronoi-Zelle $\mathcal{V}(s)$ von $s \in S$:** Menge der Punkte mit min. Abstand zu s

Wer benutzt welchen Briefkasten?

Situation

- gegeben: Menge S von Standorten
- Welche Punkte liegen am nächsten zu welchem Standort?

(bei Euklidischer Metrik)



Voronoi-Diagramm

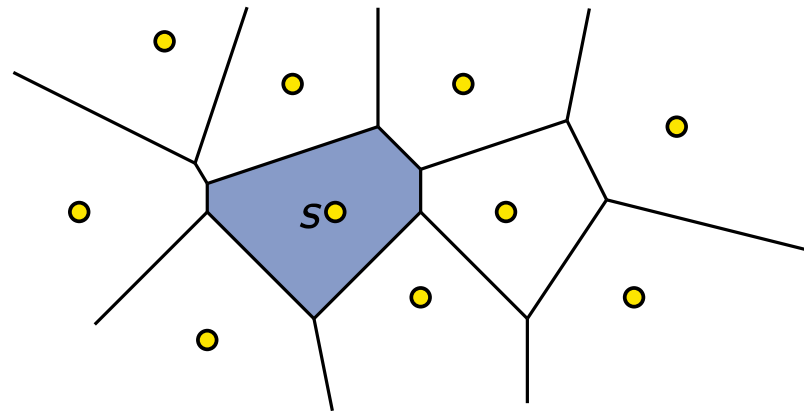
- **Voronoi-Zelle $\mathcal{V}(s)$ von $s \in S$:** Menge der Punkte mit min. Abstand zu s
- Vereinigung aller Zellen ist das **Voronoi-Diagramm $\text{Vor}(S)$**

Wer benutzt welchen Briefkasten?

Situation

- gegeben: Menge S von Standorten
- Welche Punkte liegen am nächsten zu welchem Standort?

(bei Euklidischer Metrik)



Voronoi-Diagramm

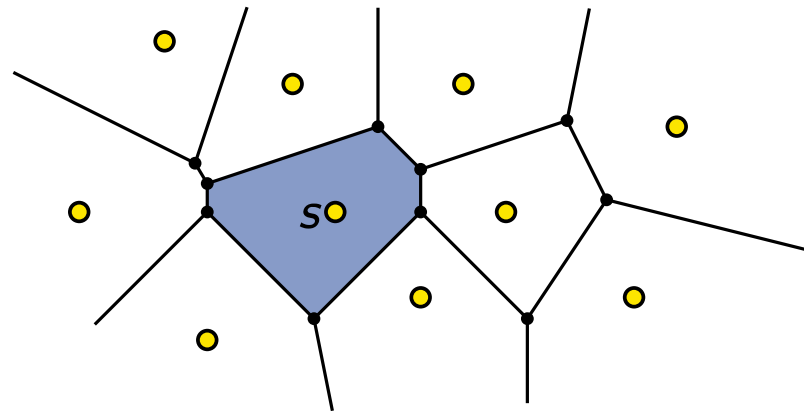
- **Voronoi-Zelle $\mathcal{V}(s)$ von $s \in S$:** Menge der Punkte mit min. Abstand zu s
- Vereinigung aller Zellen ist das **Voronoi-Diagramm $\text{Vor}(S)$**
- planare Unterteilung der Ebene

Wer benutzt welchen Briefkasten?

Situation

- gegeben: Menge S von Standorten
- Welche Punkte liegen am nächsten zu welchem Standort?

(bei Euklidischer Metrik)



Voronoi-Diagramm

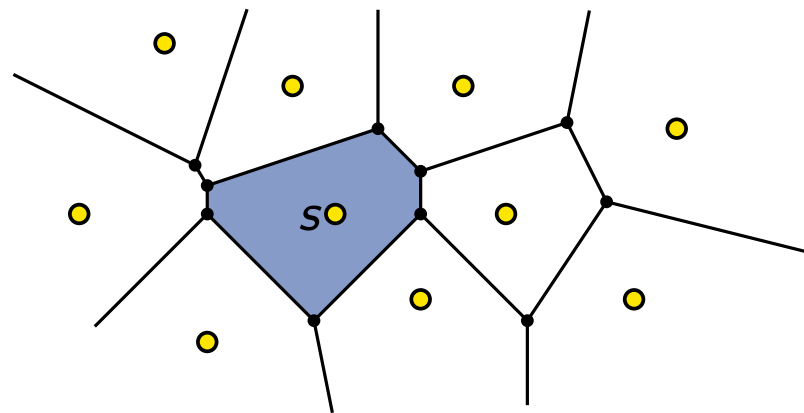
- **Voronoi-Zelle $\mathcal{V}(s)$ von $s \in S$:** Menge der Punkte mit min. Abstand zu s
- Vereinigung aller Zellen ist das **Voronoi-Diagramm $\text{Vor}(S)$**
- planare Unterteilung der Ebene
- Begrenzungslinien zwischen Zellen nennen wir **Kanten**
- Endpunkte der Kanten sind **Knoten**

Wer benutzt welchen Briefkasten?

Situation

- gegeben: Menge S von Standorten
- Welche Punkte liegen am nächsten zu welchem Standort?

(bei Euklidischer Metrik)

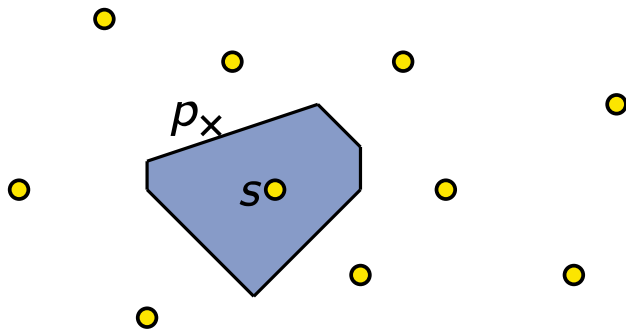


Voronoi-Diagramm

- **Voronoi-Zelle $\mathcal{V}(s)$ von $s \in S$:** Menge der Punkte mit min. Abstand zu s
- Vereinigung aller Zellen ist das **Voronoi-Diagramm $\text{Vor}(S)$**
- planare Unterteilung der Ebene
- Begrenzungslinien zwischen Zellen nennen wir **Kanten**
- Endpunkte der Kanten sind **Knoten**

Ziel für heute: Berechne $\text{Vor}(S)$

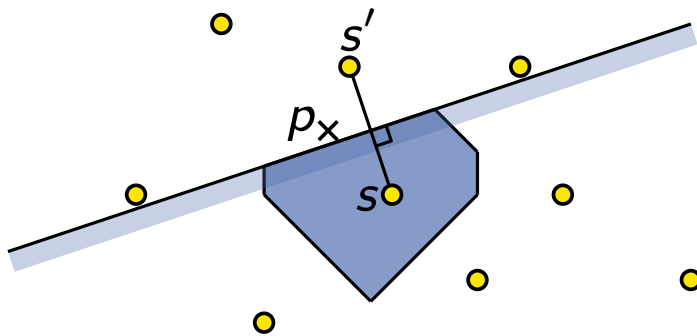
Grundlegende Eigenschaften



Punkte in der Zelle von s

- Warum liegt p nicht in der Zelle von s ?

Grundlegende Eigenschaften

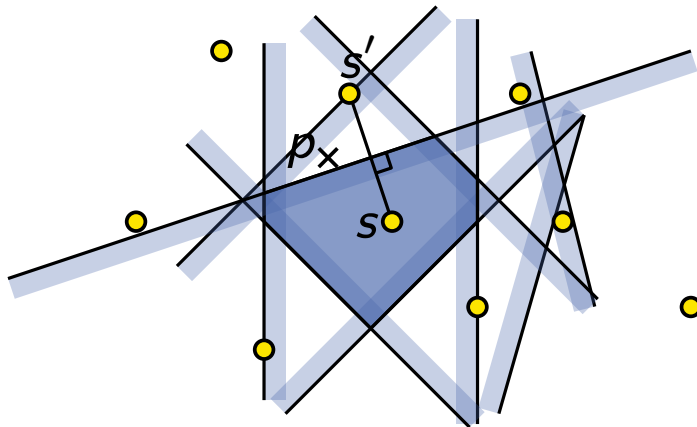


Punkte in der Zelle von s

- Warum liegt p nicht in der Zelle von s ?
- s' ist näher an $p \Leftrightarrow s \perp s'$ trennt s von p

Mittelsenkrechte von s und s'

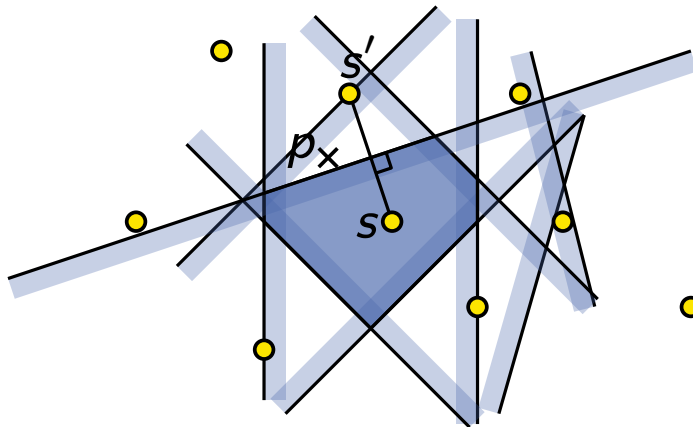
Grundlegende Eigenschaften



Punkte in der Zelle von s

- Warum liegt p nicht in der Zelle von s ?
- s' ist näher an $p \Leftrightarrow s \perp s'$ trennt s von p
- die Zelle von s ist der Schnitt von Halbebenen

Grundlegende Eigenschaften

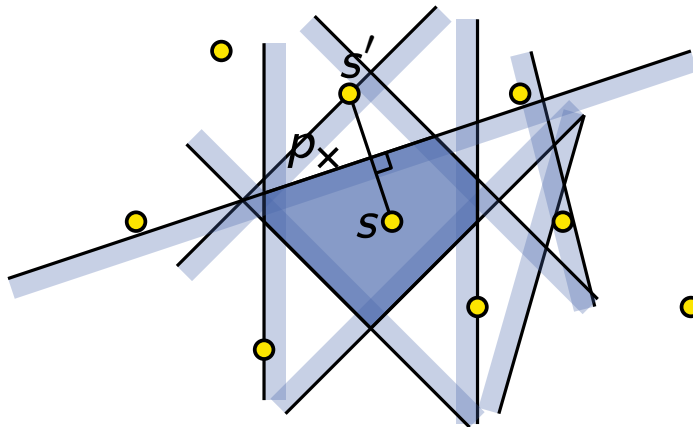


Punkte in der Zelle von s

- Warum liegt p nicht in der Zelle von s ?
- s' ist näher an $p \Leftrightarrow s \perp s'$ trennt s von p
- die Zelle von s ist der Schnitt von Halbebenen
- damit haben wir einen (langsamen) Algorithmus

Wie langsam genau?

Grundlegende Eigenschaften



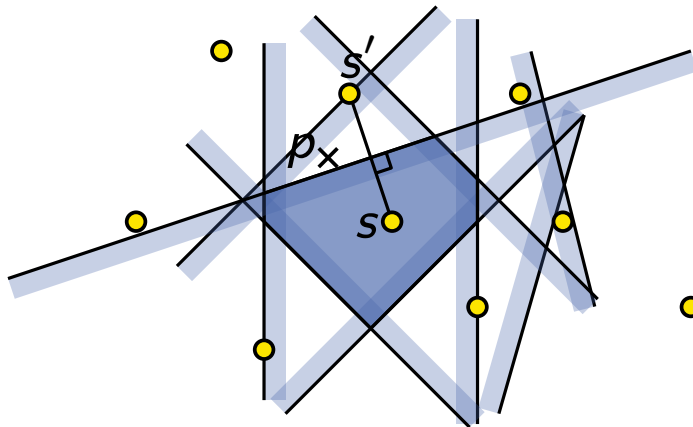
Punkte in der Zelle von s

- Warum liegt p nicht in der Zelle von s ?
- s' ist näher an $p \Leftrightarrow s \perp s'$ trennt s von p
- die Zelle von s ist der Schnitt von Halbebenen
- damit haben wir einen (langsamen) Algorithmus

Wie langsam genau?

Zusammenhang

Grundlegende Eigenschaften



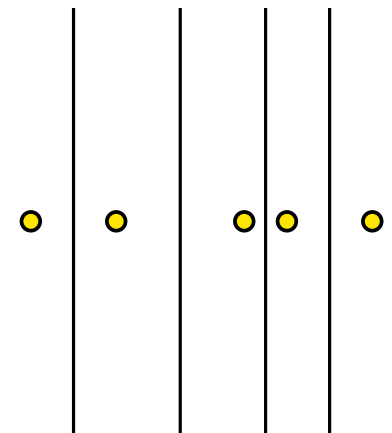
Punkte in der Zelle von s

- Warum liegt p nicht in der Zelle von s ?
- s' ist näher an $p \Leftrightarrow s \perp s'$ trennt s von p
- die Zelle von s ist der Schnitt von Halbebenen
- damit haben wir einen (langsamen) Algorithmus

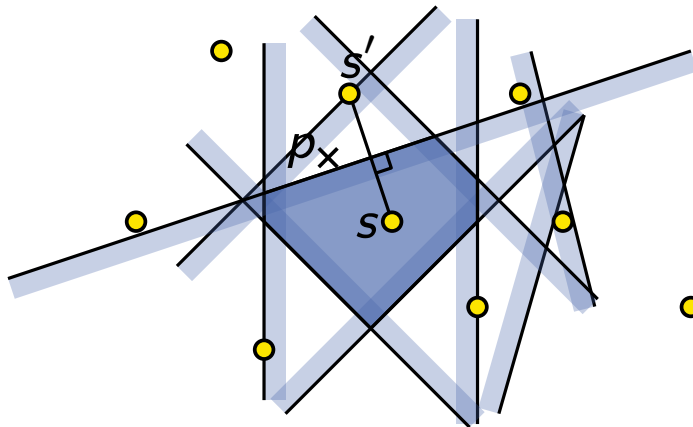
Wie langsam genau?

Zusammenhang

- wenn nicht alle Punkte kollinear sind, dann ist $\text{Vor}(S)$ zusammenhängend
($\text{Vor}(S)$ bezieht sich hier auf die Vereinigung aller Kanten, also quasi den „Graphen“)



Grundlegende Eigenschaften



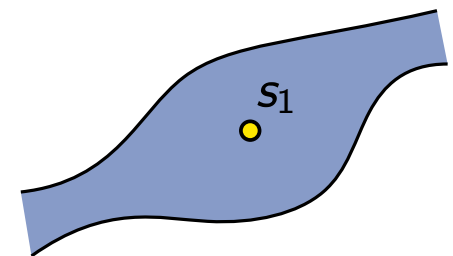
Punkte in der Zelle von s

- Warum liegt p nicht in der Zelle von s ?
- s' ist näher an $p \Leftrightarrow s \perp s'$ trennt s von p
- die Zelle von s ist der Schnitt von Halbebenen
- damit haben wir einen (langsamen) Algorithmus

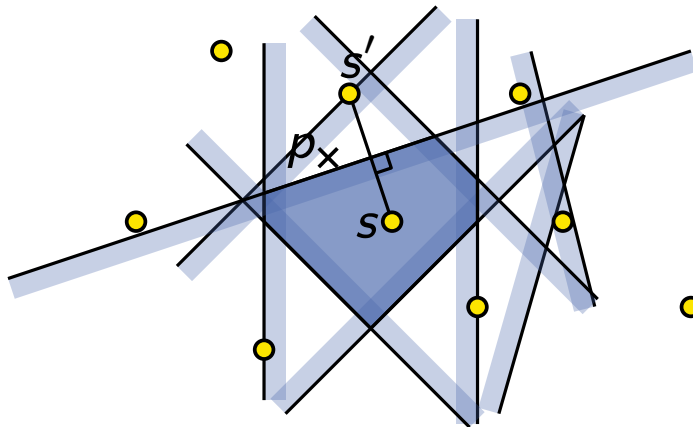
Wie langsam genau?

Zusammenhang

- wenn nicht alle Punkte kollinear sind, dann ist $\text{Vor}(S)$ zusammenhängend
($\text{Vor}(S)$ bezieht sich hier auf die Vereinigung aller Kanten, also quasi den „Graphen“)
- Beweisidee: nimm an, es wäre unzusammenhängend
 - \Rightarrow es gibt $s_1 \in S$, sodass $\mathcal{V}(s_1)$ die Ebene zerteilt



Grundlegende Eigenschaften



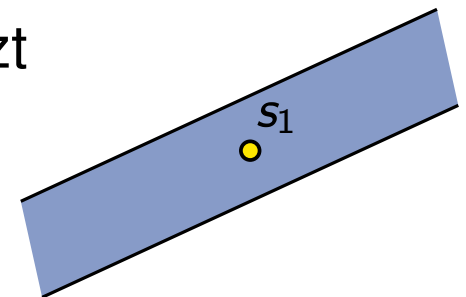
Punkte in der Zelle von s

- Warum liegt p nicht in der Zelle von s ?
- s' ist näher an $p \Leftrightarrow s \perp s'$ trennt s von p
- die Zelle von s ist der Schnitt von Halbebenen
- damit haben wir einen (langsamen) Algorithmus

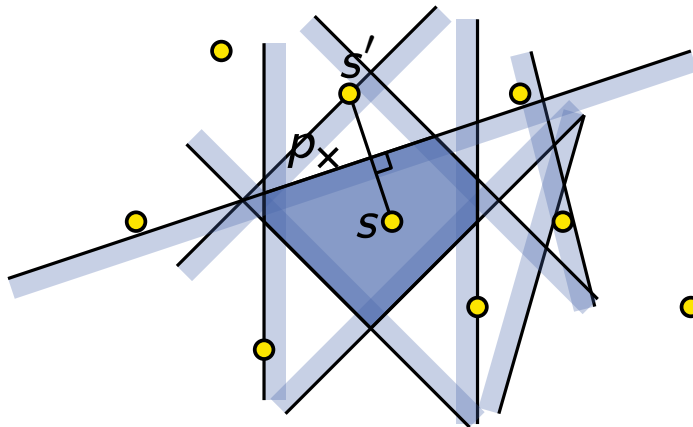
Wie langsam genau?

Zusammenhang

- wenn nicht alle Punkte kollinear sind, dann ist $\text{Vor}(S)$ zusammenhängend
($\text{Vor}(S)$ bezieht sich hier auf die Vereinigung aller Kanten, also quasi den „Graphen“)
- Beweisidee: nimm an, es wäre unzusammenhängend
 - \Rightarrow es gibt $s_1 \in S$, sodass $\mathcal{V}(s_1)$ die Ebene zerteilt
 - $\mathcal{V}(s_1)$ konvex \Rightarrow von zwei parallele Geraden begrenzt



Grundlegende Eigenschaften



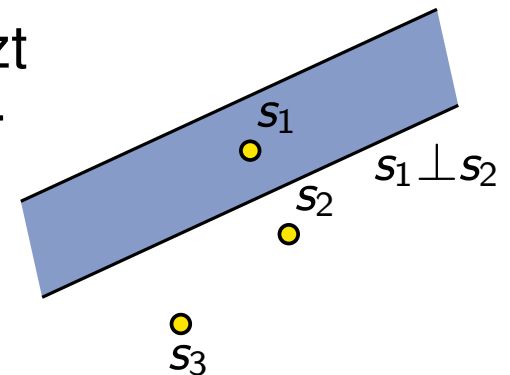
Punkte in der Zelle von s

- Warum liegt p nicht in der Zelle von s ?
- s' ist näher an $p \Leftrightarrow s \perp s'$ trennt s von p
- die Zelle von s ist der Schnitt von Halbebenen
- damit haben wir einen (langsamen) Algorithmus

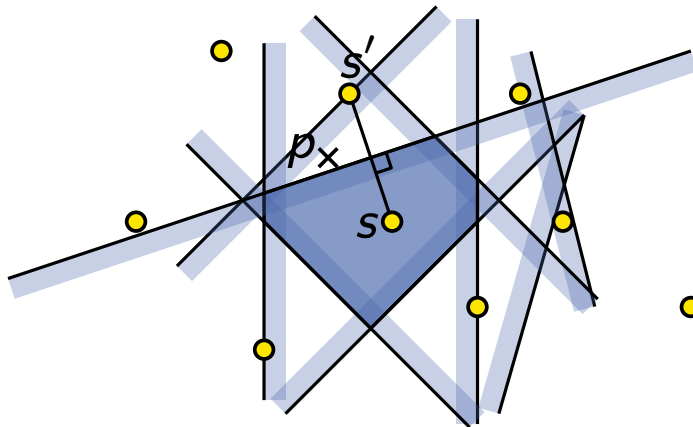
Wie langsam genau?

Zusammenhang

- wenn nicht alle Punkte kollinear sind, dann ist $\text{Vor}(S)$ zusammenhängend
($\text{Vor}(S)$ bezieht sich hier auf die Vereinigung aller Kanten, also quasi den „Graphen“)
- Beweisidee: nimm an, es wäre unzusammenhängend
 - \Rightarrow es gibt $s_1 \in S$, sodass $\mathcal{V}(s_1)$ die Ebene zerteilt
 - $\mathcal{V}(s_1)$ konvex \Rightarrow von zwei parallele Geraden begrenzt
 - sei $s_1 \perp s_2$ eine der Geraden und sei s_3 nicht-kollinear



Grundlegende Eigenschaften



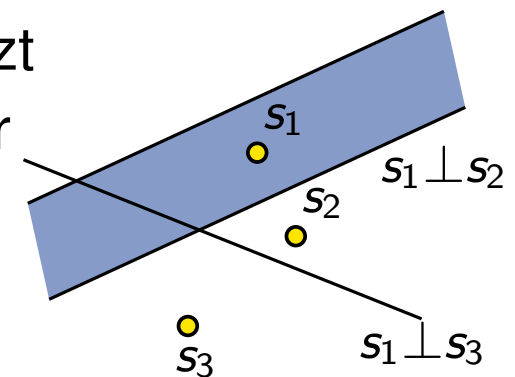
Punkte in der Zelle von s

- Warum liegt p nicht in der Zelle von s ?
- s' ist näher an $p \Leftrightarrow s \perp s'$ trennt s von p
- die Zelle von s ist der Schnitt von Halbebenen
- damit haben wir einen (langsamen) Algorithmus

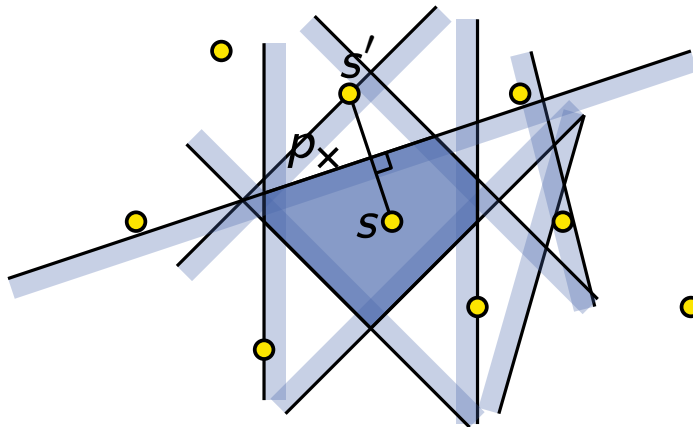
Wie langsam genau?

Zusammenhang

- wenn nicht alle Punkte kollinear sind, dann ist $\text{Vor}(S)$ zusammenhängend
($\text{Vor}(S)$ bezieht sich hier auf die Vereinigung aller Kanten, also quasi den „Graphen“)
- Beweisidee: nimm an, es wäre unzusammenhängend
 - \Rightarrow es gibt $s_1 \in S$, sodass $\mathcal{V}(s_1)$ die Ebene zerteilt
 - $\mathcal{V}(s_1)$ konvex \Rightarrow von zwei parallele Geraden begrenzt
 - sei $s_1 \perp s_2$ eine der Geraden und sei s_3 nicht-kollinear
 - $\Rightarrow s_1 \perp s_2$ schneidet $s_1 \perp s_3$



Grundlegende Eigenschaften



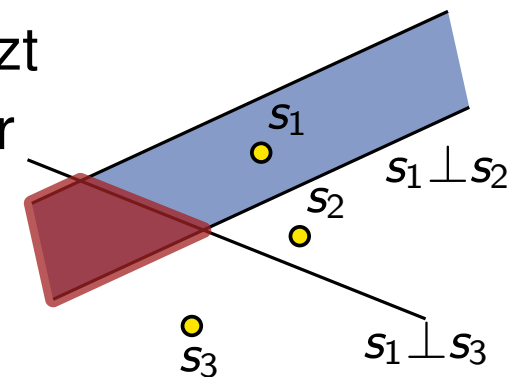
Punkte in der Zelle von s

- Warum liegt p nicht in der Zelle von s ?
- s' ist näher an $p \Leftrightarrow s \perp s'$ trennt s von p
- die Zelle von s ist der Schnitt von Halbebenen
- damit haben wir einen (langsamen) Algorithmus

Wie langsam genau?

Zusammenhang

- wenn nicht alle Punkte kollinear sind, dann ist $\text{Vor}(S)$ zusammenhängend
($\text{Vor}(S)$ bezieht sich hier auf die Vereinigung aller Kanten, also quasi den „Graphen“)
- Beweisidee: nimm an, es wäre unzusammenhängend
 - \Rightarrow es gibt $s_1 \in S$, sodass $\mathcal{V}(s_1)$ die Ebene zerteilt
 - $\mathcal{V}(s_1)$ konvex \Rightarrow von zwei parallele Geraden begrenzt
 - sei $s_1 \perp s_2$ eine der Geraden und sei s_3 nicht-kollinear
 - $\Rightarrow s_1 \perp s_2$ schneidet $s_1 \perp s_3$
 - \Rightarrow Widerspruch



Grundlegende Eigenschaften (2)

Theorem

(Beweis: Übung)

Für $n \geq 3$ Standorte hat das Voronoi-Diagramm maximal $2n - 5$ Knoten und maximal $3n - 6$ Kanten.

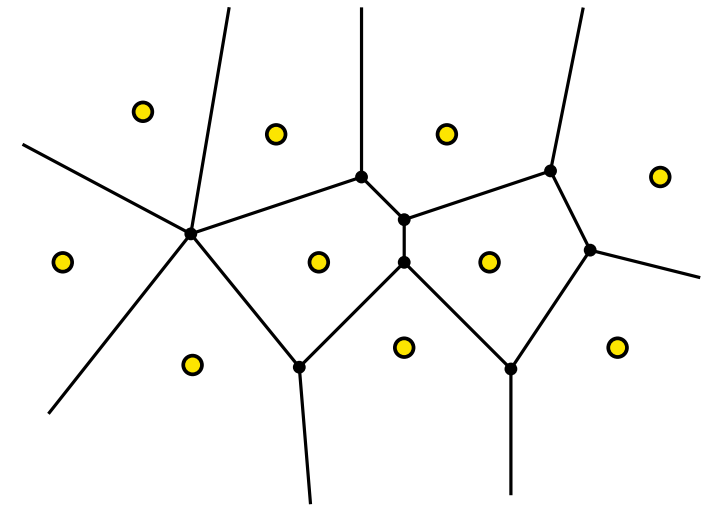
Grundlegende Eigenschaften (2)

Theorem

Für $n \geq 3$ Standorte hat das Voronoi-Diagramm maximal $2n - 5$ Knoten und maximal $3n - 6$ Kanten.

(Beweis: Übung)

Welche Punkte sind Knoten von $\text{Vor}(S)$?



Grundlegende Eigenschaften (2)

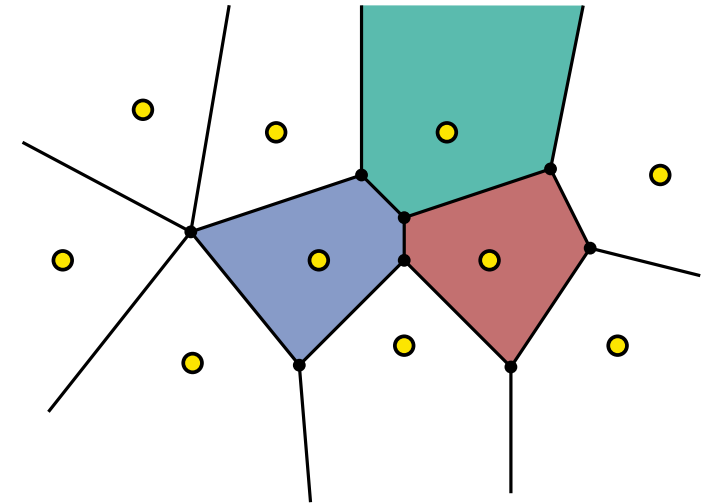
Theorem

Für $n \geq 3$ Standorte hat das Voronoi-Diagramm maximal $2n - 5$ Knoten und maximal $3n - 6$ Kanten.

(Beweis: Übung)

Welche Punkte sind Knoten von $\text{Vor}(S)$?

- an einem Knoten v treffen sich \geq drei Zellen



Grundlegende Eigenschaften (2)

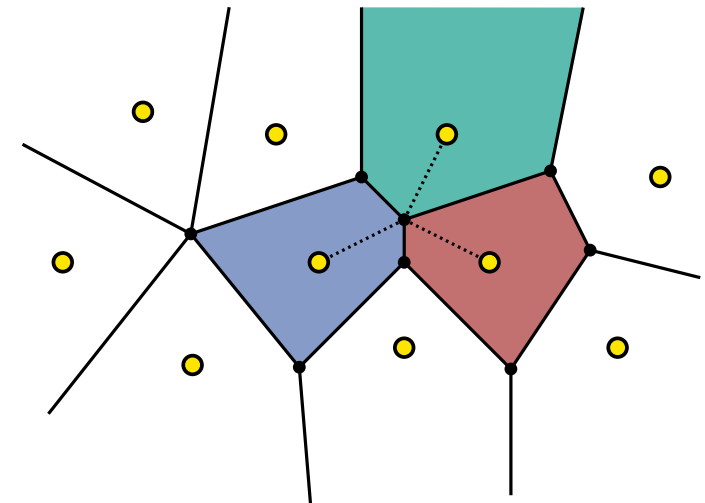
Theorem

Für $n \geq 3$ Standorte hat das Voronoi-Diagramm maximal $2n - 5$ Knoten und maximal $3n - 6$ Kanten.

(Beweis: Übung)

Welche Punkte sind Knoten von $\text{Vor}(S)$?

- an einem Knoten v treffen sich \geq drei Zellen
- v hat gleichen Abstand von den Standorten



Grundlegende Eigenschaften (2)

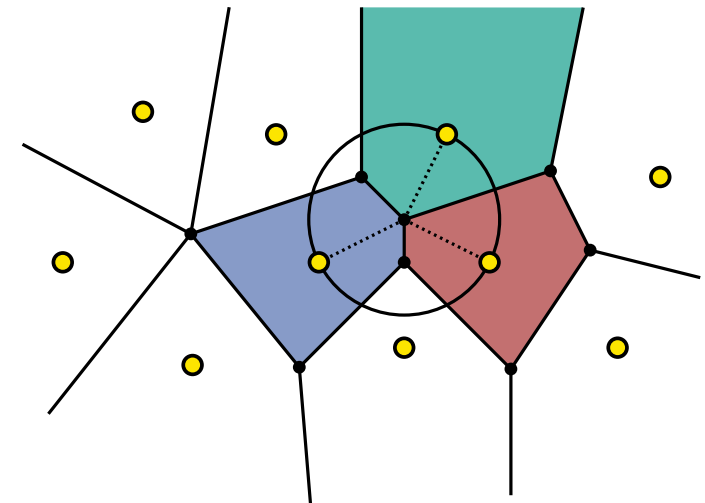
Theorem

Für $n \geq 3$ Standorte hat das Voronoi-Diagramm maximal $2n - 5$ Knoten und maximal $3n - 6$ Kanten.

(Beweis: Übung)

Welche Punkte sind Knoten von $\text{Vor}(S)$?

- an einem Knoten v treffen sich \geq drei Zellen
- v hat gleichen Abstand von den Standorten
 \Rightarrow Kreis mit Zentrum v



Grundlegende Eigenschaften (2)

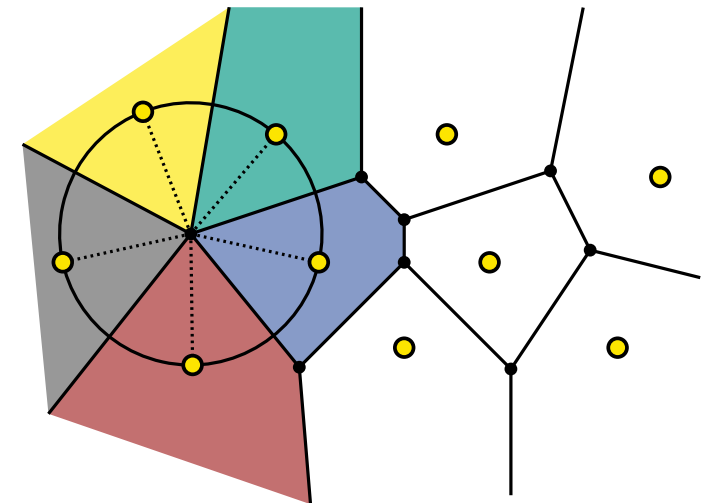
Theorem

Für $n \geq 3$ Standorte hat das Voronoi-Diagramm maximal $2n - 5$ Knoten und maximal $3n - 6$ Kanten.

(Beweis: Übung)

Welche Punkte sind Knoten von $\text{Vor}(S)$?

- an einem Knoten v treffen sich \geq drei Zellen
- v hat gleichen Abstand von den Standorten
 \Rightarrow Kreis mit Zentrum v



Grundlegende Eigenschaften (2)

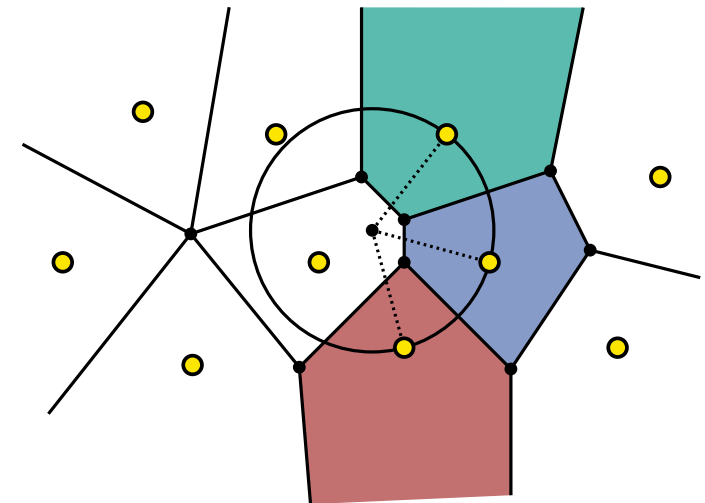
Theorem

Für $n \geq 3$ Standorte hat das Voronoi-Diagramm maximal $2n - 5$ Knoten und maximal $3n - 6$ Kanten.

(Beweis: Übung)

Welche Punkte sind Knoten von $\text{Vor}(S)$?

- an einem Knoten v treffen sich \geq drei Zellen
- v hat gleichen Abstand von den Standorten
 \Rightarrow Kreis mit Zentrum v
- nicht jeder Kreis durch ≥ 3 Standorte hat einen Knoten als Mittelpunkt



Grundlegende Eigenschaften (2)

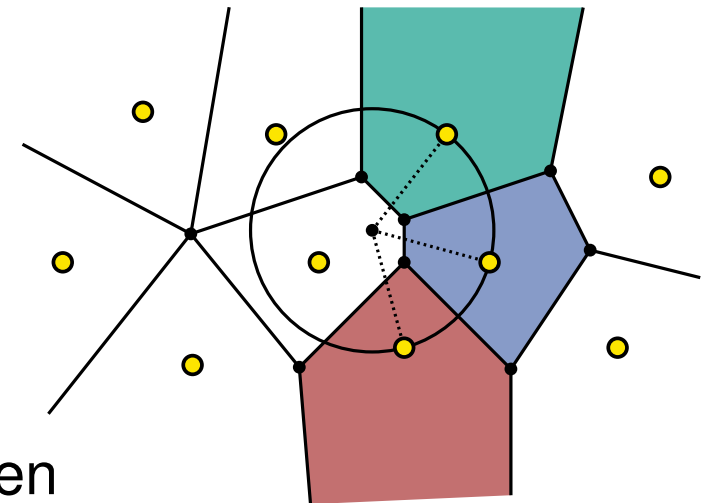
Theorem

Für $n \geq 3$ Standorte hat das Voronoi-Diagramm maximal $2n - 5$ Knoten und maximal $3n - 6$ Kanten.

(Beweis: Übung)

Welche Punkte sind Knoten von $\text{Vor}(S)$?

- an einem Knoten v treffen sich \geq drei Zellen
- v hat gleichen Abstand von den Standorten
 \Rightarrow Kreis mit Zentrum v
- nicht jeder Kreis durch ≥ 3 Standorte hat einen Knoten als Mittelpunkt
- der Kreis darf keinen Standort im Inneren haben



Grundlegende Eigenschaften (2)

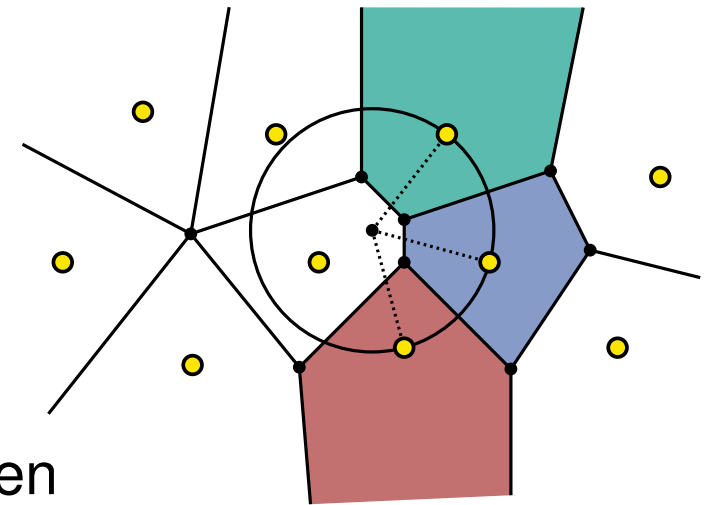
Theorem

Für $n \geq 3$ Standorte hat das Voronoi-Diagramm maximal $2n - 5$ Knoten und maximal $3n - 6$ Kanten.

(Beweis: Übung)

Welche Punkte sind Knoten von $\text{Vor}(S)$?

- an einem Knoten v treffen sich \geq drei Zellen
- v hat gleichen Abstand von den Standorten
 \Rightarrow Kreis mit Zentrum v
- nicht jeder Kreis durch ≥ 3 Standorte hat einen Knoten als Mittelpunkt
- der Kreis darf keinen Standort im Inneren haben
- $C_S(p) =$ größter Kreis um p , der keinen Standort aus S im Inneren enthält



Grundlegende Eigenschaften (2)

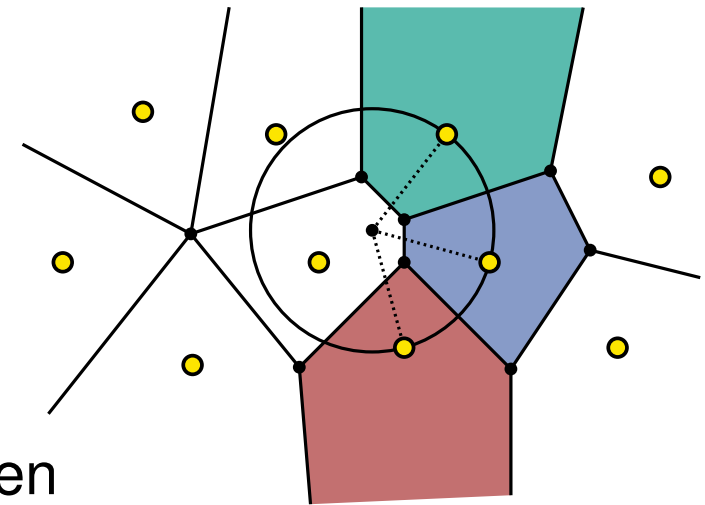
Theorem

Für $n \geq 3$ Standorte hat das Voronoi-Diagramm maximal $2n - 5$ Knoten und maximal $3n - 6$ Kanten.

(Beweis: Übung)

Welche Punkte sind Knoten von $\text{Vor}(S)$?

- an einem Knoten v treffen sich \geq drei Zellen
- v hat gleichen Abstand von den Standorten
 \Rightarrow Kreis mit Zentrum v
- nicht jeder Kreis durch ≥ 3 Standorte hat einen Knoten als Mittelpunkt
- der Kreis darf keinen Standort im Inneren haben
- $C_S(p) =$ größter Kreis um p , der keinen Standort aus S im Inneren enthält



Theorem

Der Punkt $p \in \mathbb{R}^2$ ist ein Knoten von $\text{Vor}(S)$ genau dann wenn auf dem Rand von $C_S(p)$ mindestens drei Standorte liegen.

(Knoten des Voronoi-Diagramms)

Grundlegende Eigenschaften (2)

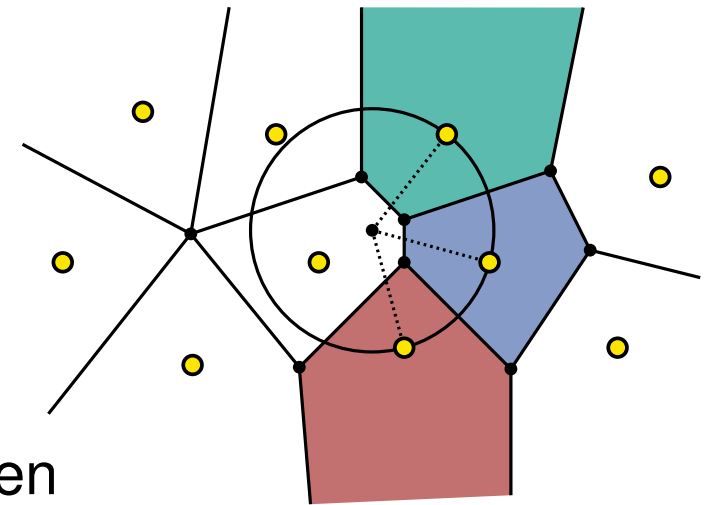
Theorem

Für $n \geq 3$ Standorte hat das Voronoi-Diagramm maximal $2n - 5$ Knoten und maximal $3n - 6$ Kanten.

(Beweis: Übung)

Welche Punkte sind Knoten von $\text{Vor}(S)$?

- an einem Knoten v treffen sich \geq drei Zellen
- v hat gleichen Abstand von den Standorten
 \Rightarrow Kreis mit Zentrum v
- nicht jeder Kreis durch ≥ 3 Standorte hat einen Knoten als Mittelpunkt
- der Kreis darf keinen Standort im Inneren haben
- $C_S(p) =$ größter Kreis um p , der keinen Standort aus S im Inneren enthält



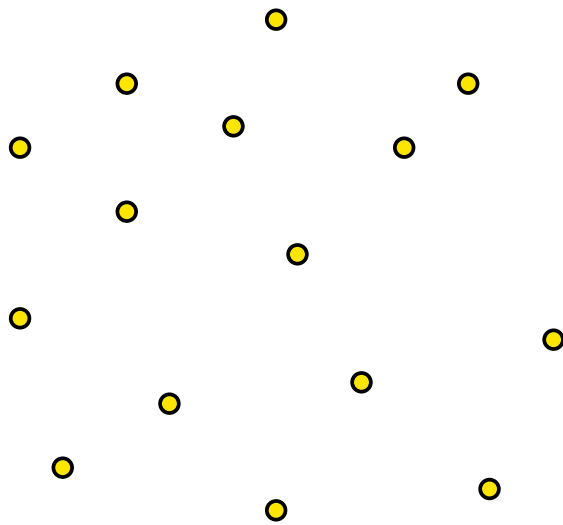
Theorem

(Knoten des Voronoi-Diagramms)

Der Punkt $p \in \mathbb{R}^2$ ist ein Knoten von $\text{Vor}(S)$ genau dann wenn auf dem Rand von $C_S(p)$ mindestens drei Standorte liegen.

Welche analoge Aussage gilt für Kanten?

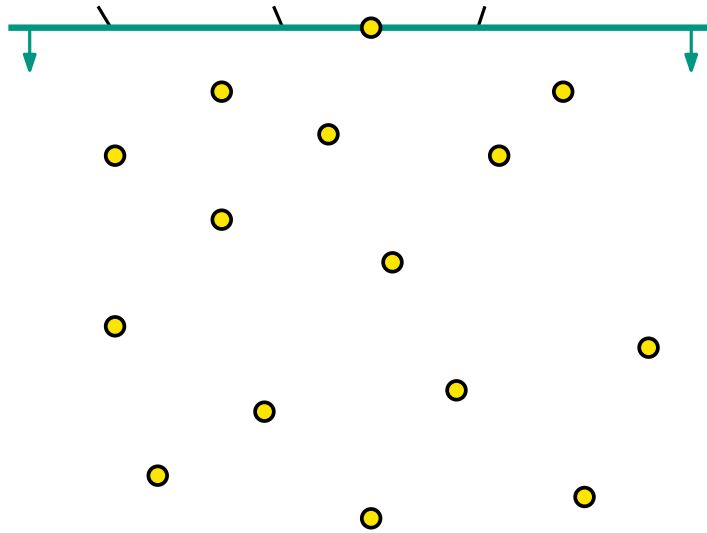
Sweep-Line Algo



Grundsätzlicher Ansatz: Sweep-Line

- schiebe Sweep-Line von oben nach unten
- zu jedem Zeitpunkt: kenne $Vor(S)$ für die Halbebene über der Sweep-Line

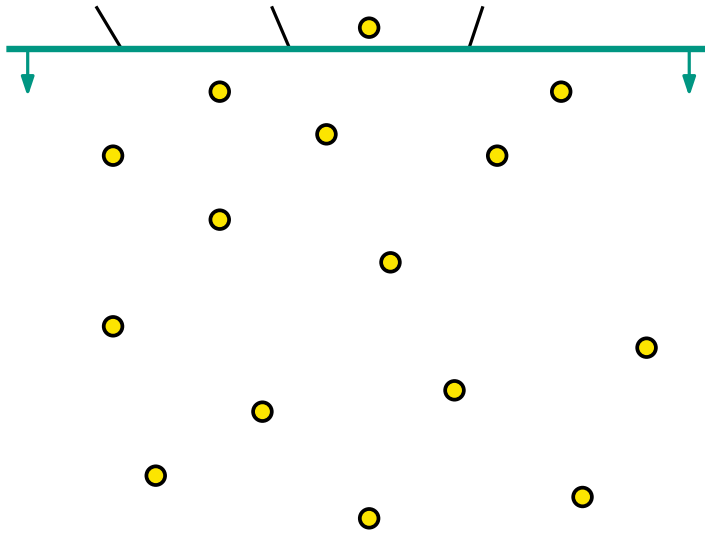
Sweep-Line Algo



Grundsätzlicher Ansatz: Sweep-Line

- schiebe Sweep-Line von oben nach unten
- zu jedem Zeitpunkt: kenne $Vor(S)$ für die Halbebene über der Sweep-Line

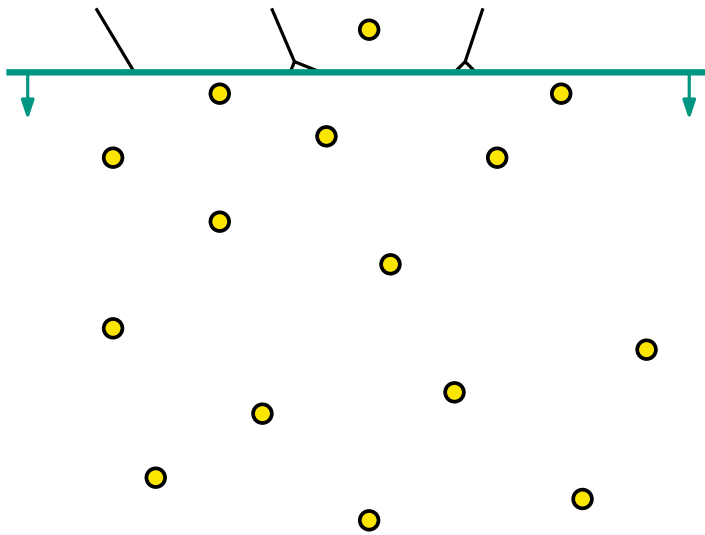
Sweep-Line Algo



Grundsätzlicher Ansatz: Sweep-Line

- schiebe Sweep-Line von oben nach unten
- zu jedem Zeitpunkt: kenne $Vor(S)$ für die Halbebene über der Sweep-Line

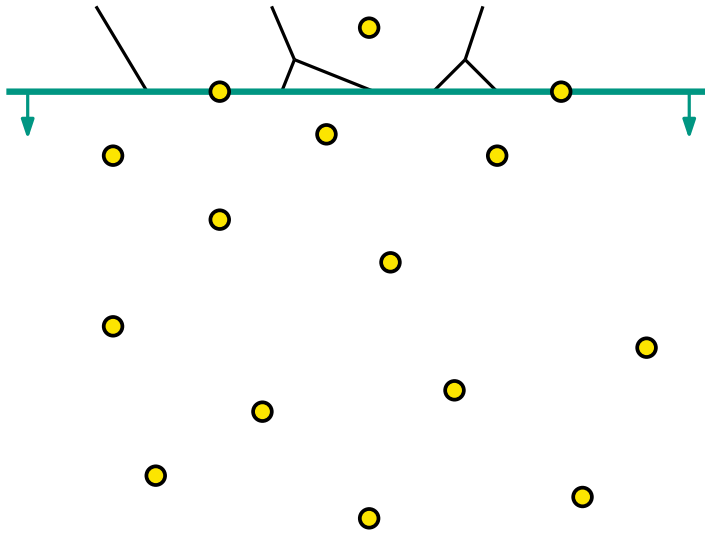
Sweep-Line Algo



Grundsätzlicher Ansatz: Sweep-Line

- schiebe Sweep-Line von oben nach unten
- zu jedem Zeitpunkt: kenne $\text{Vor}(S)$ für die Halbebene über der Sweep-Line

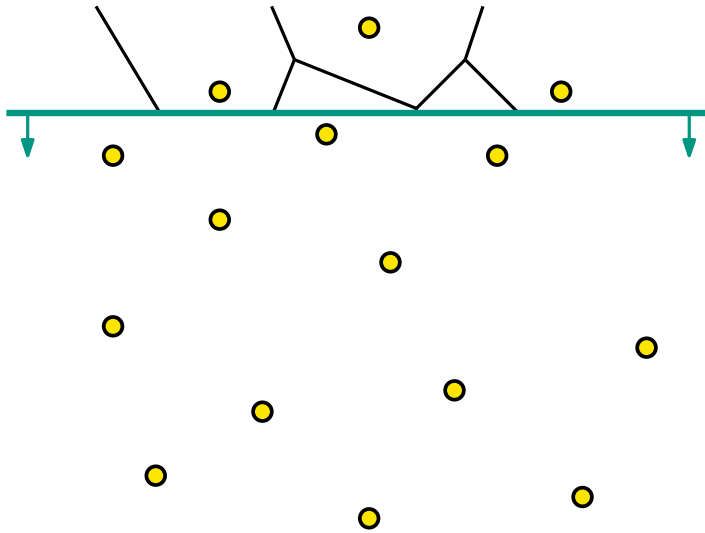
Sweep-Line Algo



Grundsätzlicher Ansatz: Sweep-Line

- schiebe Sweep-Line von oben nach unten
- zu jedem Zeitpunkt: kenne $\text{Vor}(S)$ für die Halbebene über der Sweep-Line

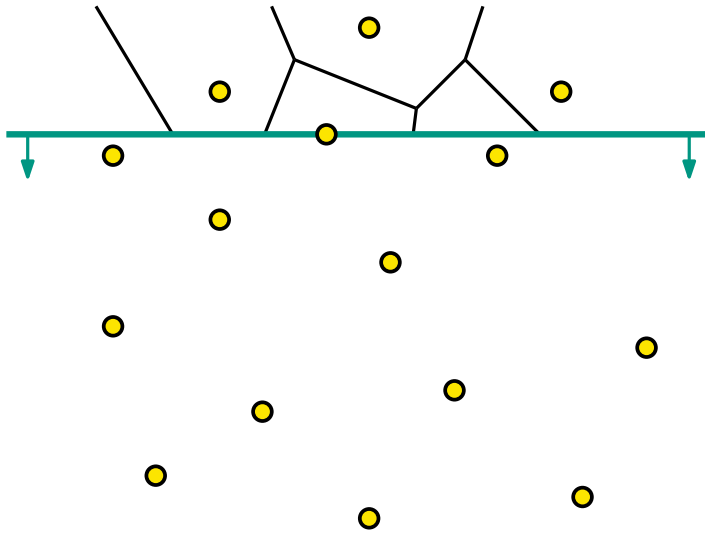
Sweep-Line Algo



Grundsätzlicher Ansatz: Sweep-Line

- schiebe Sweep-Line von oben nach unten
- zu jedem Zeitpunkt: kenne $Vor(S)$ für die Halbebene über der Sweep-Line

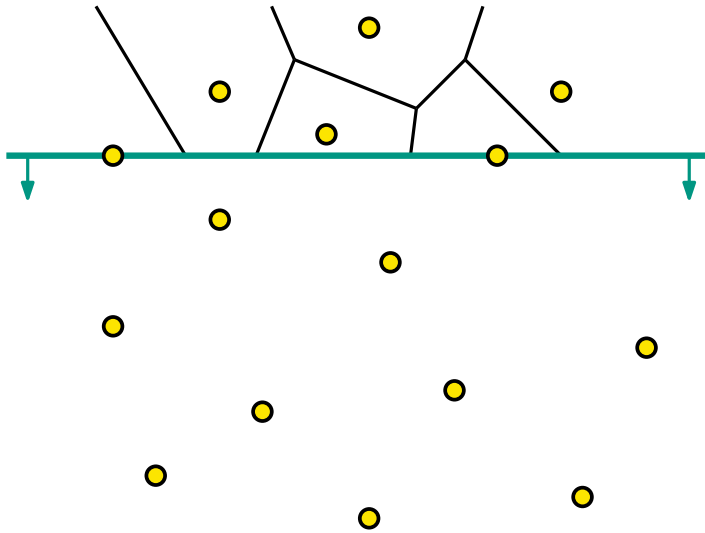
Sweep-Line Algo



Grundsätzlicher Ansatz: Sweep-Line

- schiebe Sweep-Line von oben nach unten
- zu jedem Zeitpunkt: kenne $\text{Vor}(S)$ für die Halbebene über der Sweep-Line

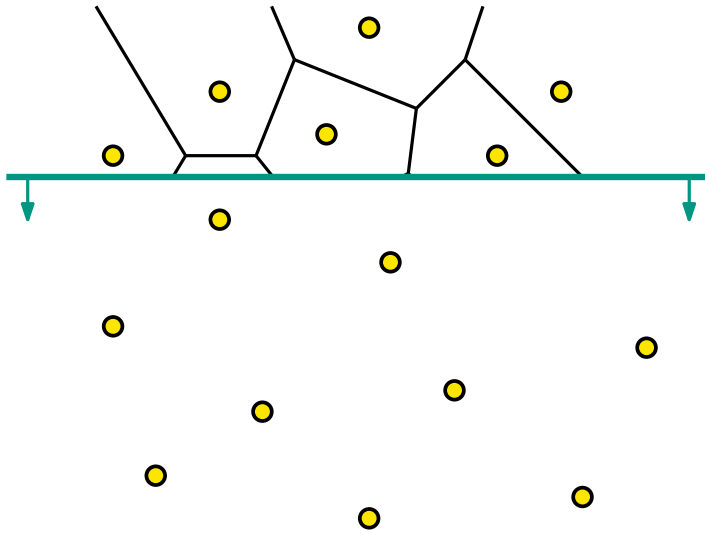
Sweep-Line Algo



Grundsätzlicher Ansatz: Sweep-Line

- schiebe Sweep-Line von oben nach unten
- zu jedem Zeitpunkt: kenne $\text{Vor}(S)$ für die Halbebene über der Sweep-Line

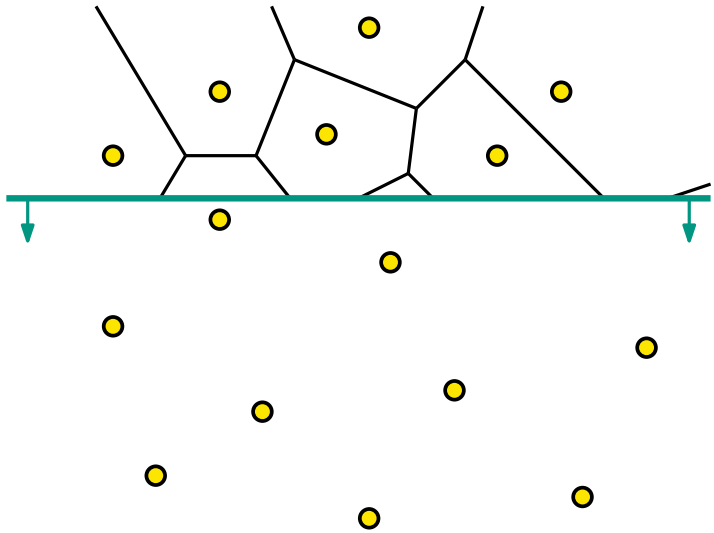
Sweep-Line Algo



Grundsätzlicher Ansatz: Sweep-Line

- schiebe Sweep-Line von oben nach unten
- zu jedem Zeitpunkt: kenne $Vor(S)$ für die Halbebene über der Sweep-Line

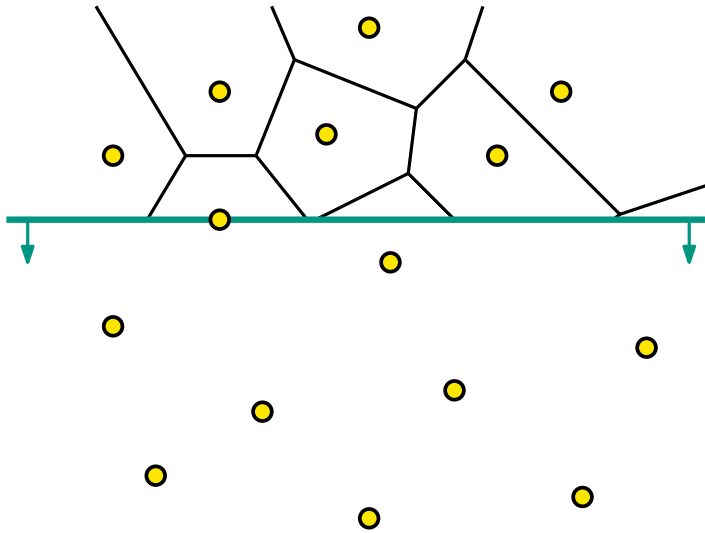
Sweep-Line Algo



Grundsätzlicher Ansatz: Sweep-Line

- schiebe Sweep-Line von oben nach unten
- zu jedem Zeitpunkt: kenne $Vor(S)$ für die Halbebene über der Sweep-Line

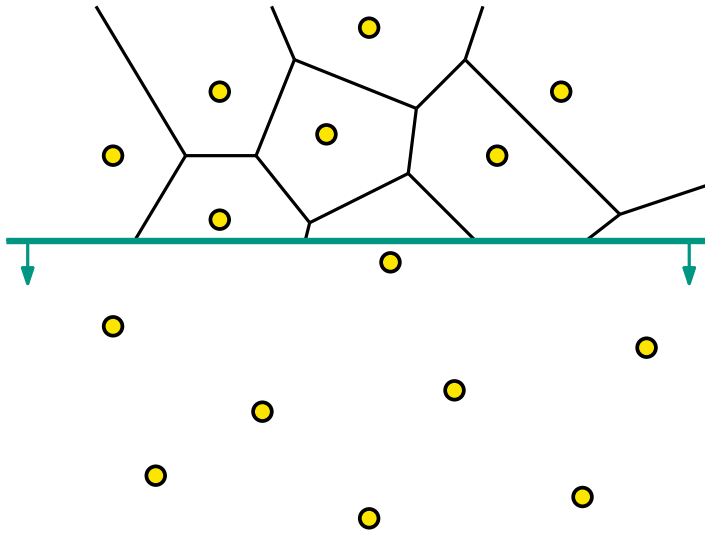
Sweep-Line Algo



Grundsätzlicher Ansatz: Sweep-Line

- schiebe Sweep-Line von oben nach unten
- zu jedem Zeitpunkt: kenne $\text{Vor}(S)$ für die Halbebene über der Sweep-Line

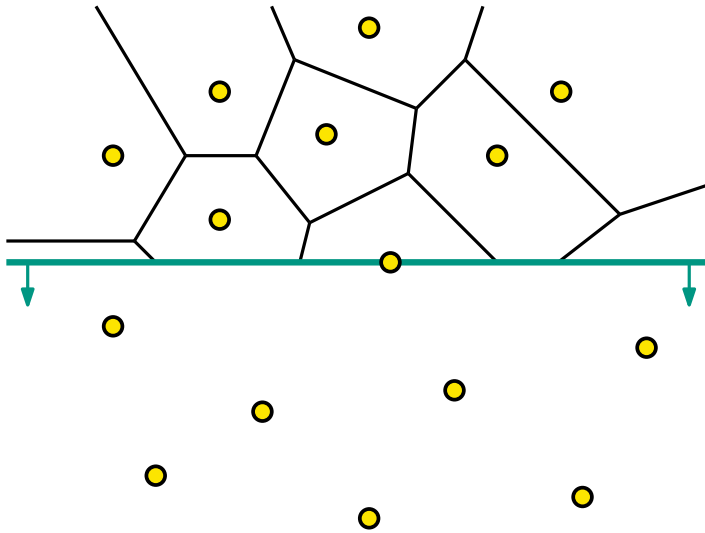
Sweep-Line Algo



Grundsätzlicher Ansatz: Sweep-Line

- schiebe Sweep-Line von oben nach unten
- zu jedem Zeitpunkt: kenne $\text{Vor}(S)$ für die Halbebene über der Sweep-Line

Sweep-Line Algo



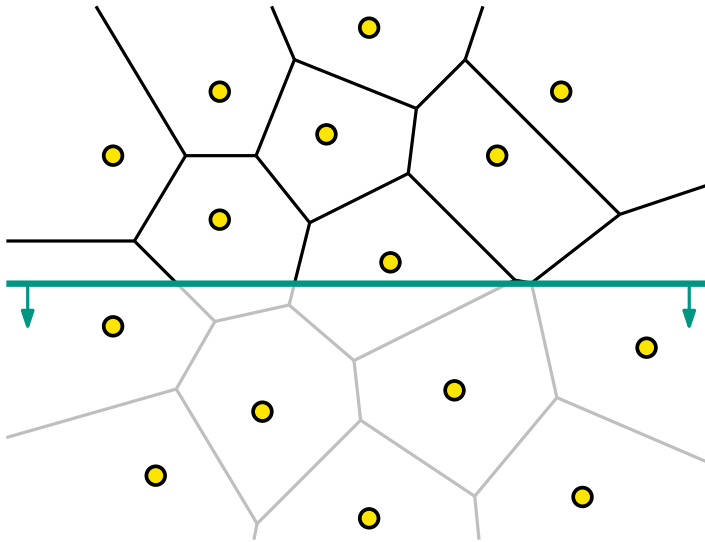
Grundsätzlicher Ansatz: Sweep-Line

- schiebe Sweep-Line von oben nach unten
- zu jedem Zeitpunkt: kenne $\text{Vor}(S)$ für die Halbebene über der Sweep-Line

Sweep-Line Algo

Grundsätzlicher Ansatz: Sweep-Line

- schiebe Sweep-Line von oben nach unten
- zu jedem Zeitpunkt: kenne $Vor(S)$ für die Halbebene über der Sweep-Line

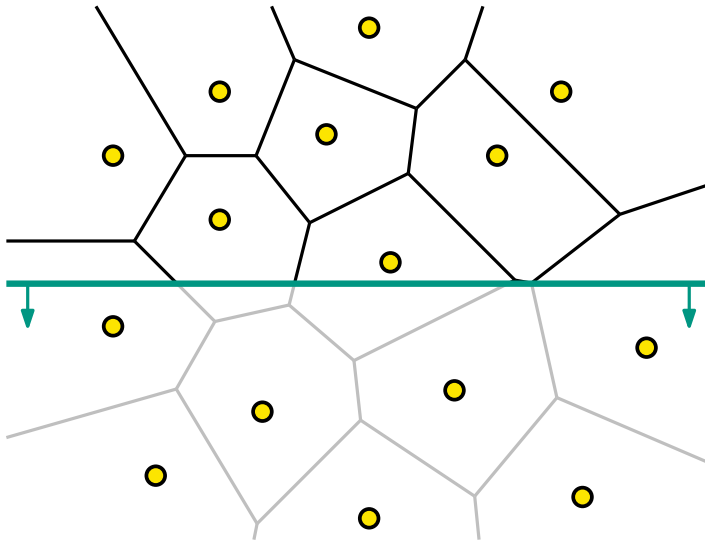


Sweep-Line Algo

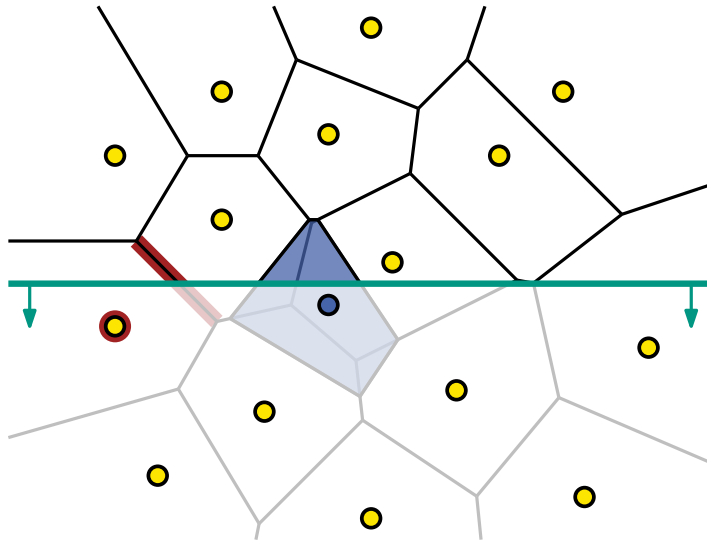
Grundsätzlicher Ansatz: Sweep-Line

- schiebe Sweep-Line von oben nach unten
- zu jedem Zeitpunkt: kenne $Vor(S)$ für die Halbebene über der Sweep-Line

Problem



Sweep-Line Algo



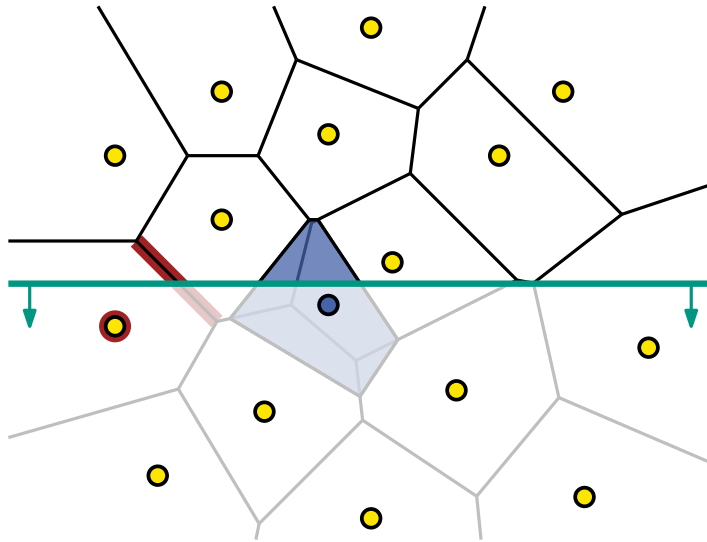
Grundsätzlicher Ansatz: Sweep-Line

- schiebe Sweep-Line von oben nach unten
- zu jedem Zeitpunkt: kenne $Vor(S)$ für die Halbebene über der Sweep-Line

Problem

- der Teil oberhalb der Sweep-Line hängen von Standorten unterhalb der Sweep-Line ab

Sweep-Line Algo



Grundsätzlicher Ansatz: Sweep-Line

- schiebe Sweep-Line von oben nach unten
- zu jedem Zeitpunkt: kenne $Vor(S)$ für die Halbebene über der Sweep-Line

Problem

- der Teil oberhalb der Sweep-Line hängen von Standorten unterhalb der Sweep-Line ab
- aber das kann ja (hoffentlich) nur knapp über der Sweep-Line passieren!?!

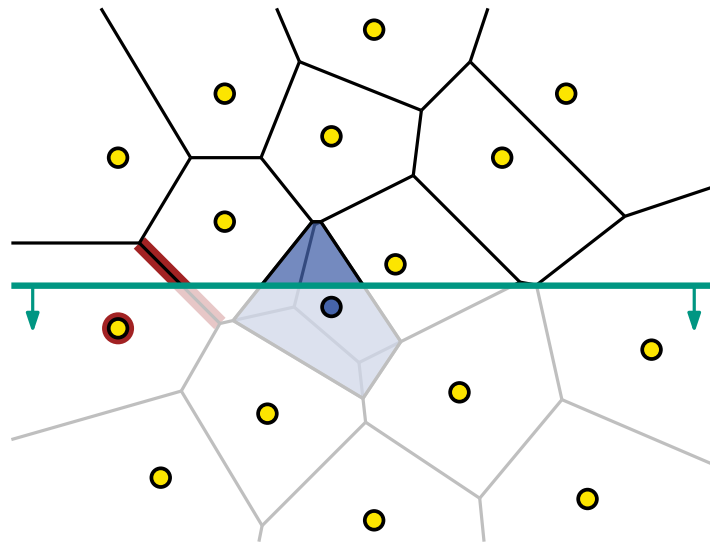
Sweep-Line Algo

Grundsätzlicher Ansatz: Sweep-Line

- schiebe Sweep-Line von oben nach unten
- zu jedem Zeitpunkt: kenne $Vor(S)$ für die Halbebene über der Sweep-Line

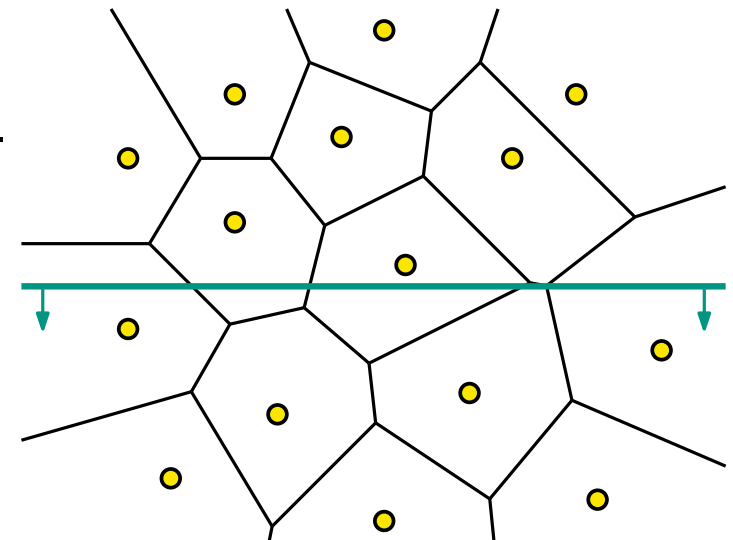
Problem

- der Teil oberhalb der Sweep-Line hängen von Standorten unterhalb der Sweep-Line ab
- aber das kann ja (hoffentlich) nur knapp über der Sweep-Line passieren!?!



Abgewandelter Sweep-Line Ansatz

- Bereich gesehener Punkte muss nicht mit fertig berechnetem Bereich übereinstimmen



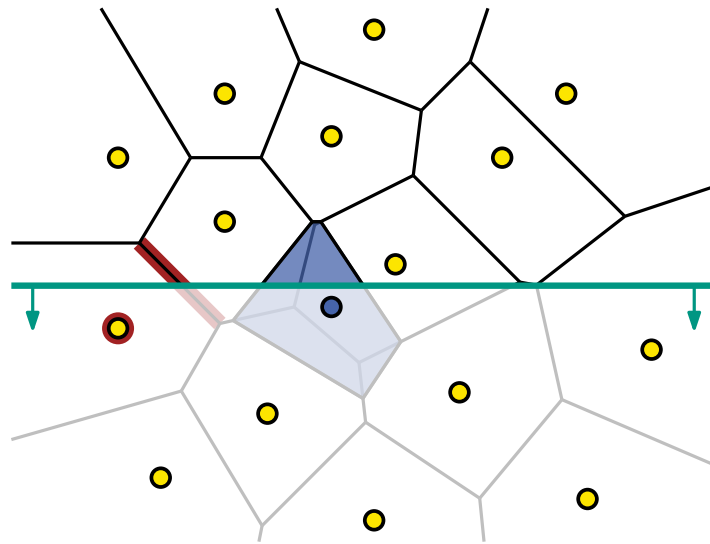
Sweep-Line Algo

Grundsätzlicher Ansatz: Sweep-Line

- schiebe Sweep-Line von oben nach unten
- zu jedem Zeitpunkt: kenne $Vor(S)$ für die Halbebene über der Sweep-Line

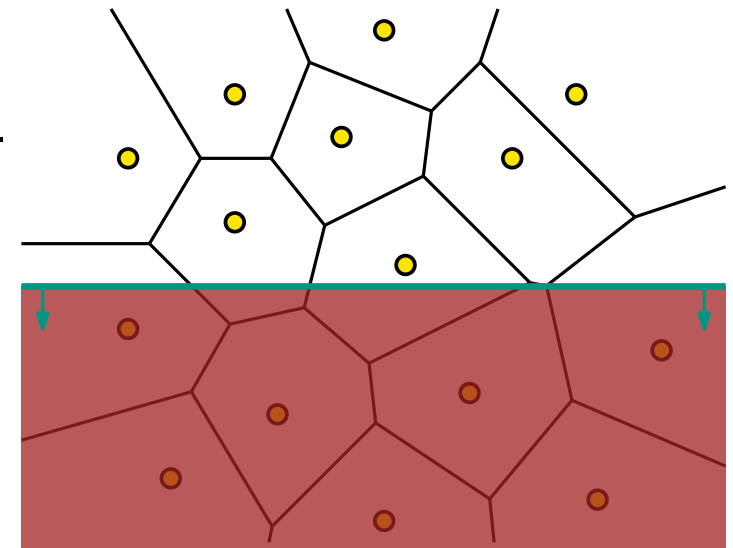
Problem

- der Teil oberhalb der Sweep-Line hängen von Standorten unterhalb der Sweep-Line ab
- aber das kann ja (hoffentlich) nur knapp über der Sweep-Line passieren!?!



Abgewandelter Sweep-Line Ansatz

- Bereich gesehener Punkte muss nicht mit fertig berechnetem Bereich übereinstimmen
- unter der Sweep-Line: unbekannter Bereich



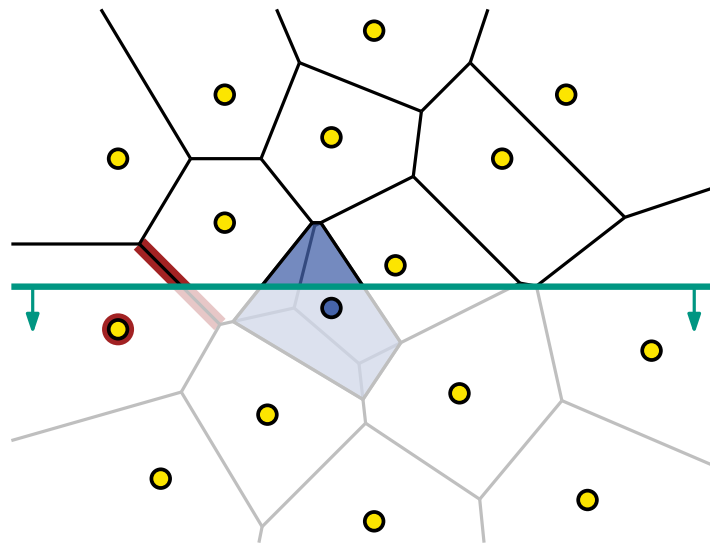
Sweep-Line Algo

Grundsätzlicher Ansatz: Sweep-Line

- schiebe Sweep-Line von oben nach unten
- zu jedem Zeitpunkt: kenne $Vor(S)$ für die Halbebene über der Sweep-Line

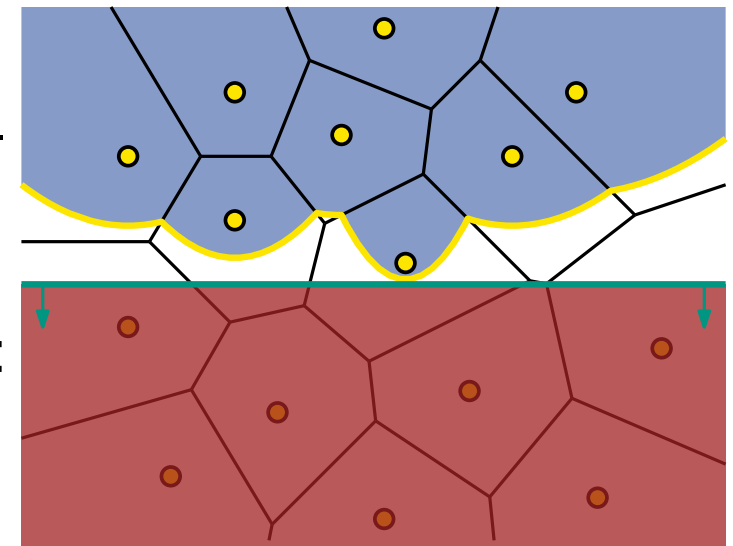
Problem

- der Teil oberhalb der Sweep-Line hängen von Standorten unterhalb der Sweep-Line ab
- aber das kann ja (hoffentlich) nur knapp über der Sweep-Line passieren!?!

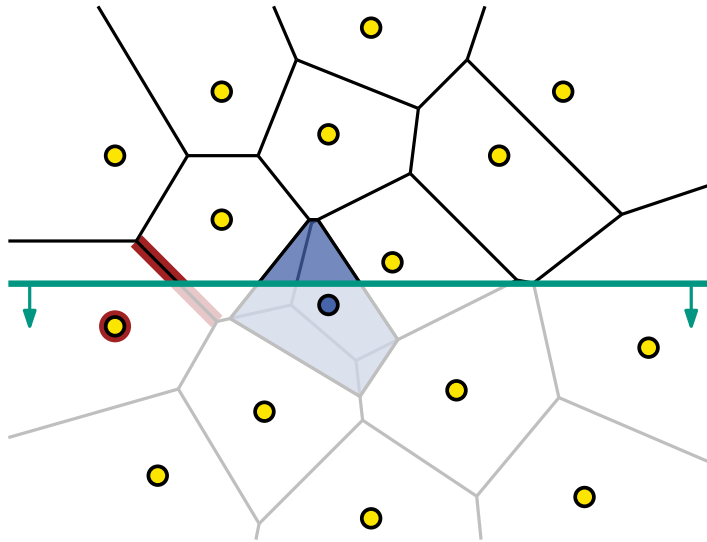


Abgewandelter Sweep-Line Ansatz

- Bereich gesehener Punkte muss nicht mit fertig berechnetem Bereich übereinstimmen
- unter der Sweep-Line: unbekannter Bereich
- gewisse Region oberhalb der Sweep-Line: bekannter Bereich



Sweep-Line Algo



Grundsätzlicher Ansatz: Sweep-Line

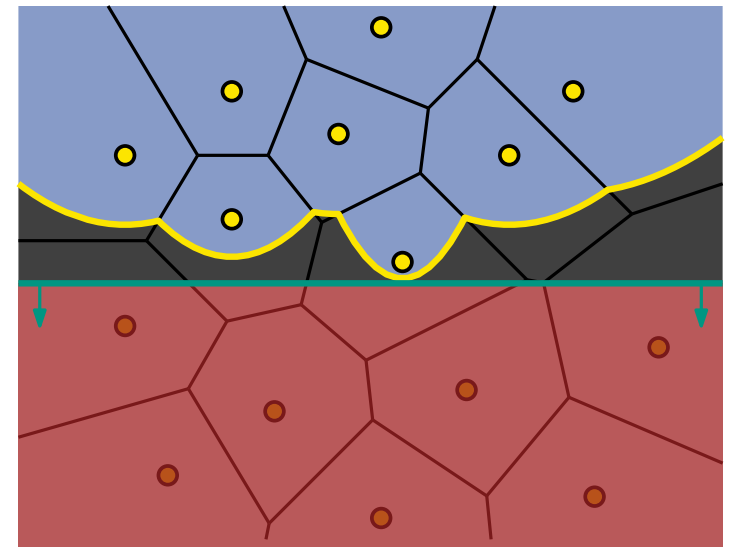
- schiebe Sweep-Line von oben nach unten
- zu jedem Zeitpunkt: kenne $\text{Vor}(S)$ für die Halbebene über der Sweep-Line

Problem

- der Teil oberhalb der Sweep-Line hängen von Standorten unterhalb der Sweep-Line ab
- aber das kann ja (hoffentlich) nur knapp über der Sweep-Line passieren!?!

Abgewandelter Sweep-Line Ansatz

- Bereich gesehener Punkte muss nicht mit fertig berechnetem Bereich übereinstimmen
- unter der Sweep-Line: unbekannter Bereich
- gewisse Region oberhalb der Sweep-Line: bekannter Bereich
- Graubereich dazwischen



Bekannter Bereich

Wo endet der bekannte Bereich?

- betrachte Punkt $p \in \mathbb{R}^2$ oberhalb von ℓ

(gegeben die Sweep-Line ℓ)

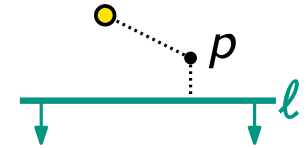


Bekannter Bereich

Wo endet der bekannte Bereich?

- betrachte Punkt $p \in \mathbb{R}^2$ oberhalb von ℓ
- Fall 1: p ist näher an ℓ als an jedem Standort

(gegeben die Sweep-Line ℓ)

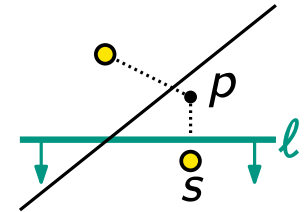


Bekannter Bereich

Wo endet der bekannte Bereich?

(gegeben die Sweep-Line ℓ)

- betrachte Punkt $p \in \mathbb{R}^2$ oberhalb von ℓ
- Fall 1: p ist näher an ℓ als an jedem Standort
 - es könnte Standort s unter ℓ geben, sodass $p \in \mathcal{V}(s)$

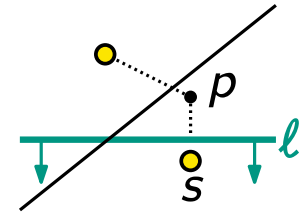


Bekannter Bereich

Wo endet der bekannte Bereich?

(gegeben die Sweep-Line ℓ)

- betrachte Punkt $p \in \mathbb{R}^2$ oberhalb von ℓ
- Fall 1: p ist näher an ℓ als an jedem Standort
 - es könnte Standort s unter ℓ geben, sodass $p \in \mathcal{V}(s)$
 - p ist also Teil des Graubereichs

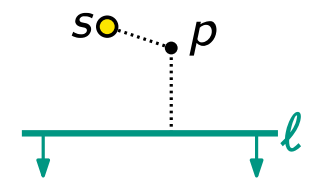
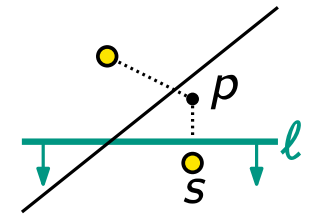


Bekannter Bereich

Wo endet der bekannte Bereich?

(gegeben die Sweep-Line ℓ)

- betrachte Punkt $p \in \mathbb{R}^2$ oberhalb von ℓ
- Fall 1: p ist näher an ℓ als an jedem Standort
 - es könnte Standort s unter ℓ geben, sodass $p \in \mathcal{V}(s)$
 - p ist also Teil des Graubereichs
- Fall 2: p ist näher an einem Standort s über ℓ als an ℓ

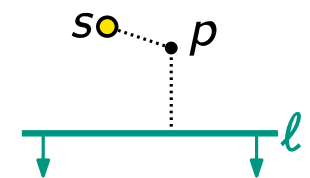
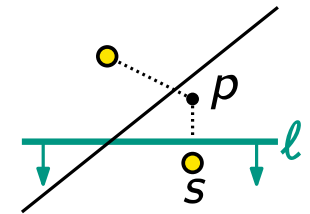


Bekannter Bereich

Wo endet der bekannte Bereich?

(gegeben die Sweep-Line ℓ)

- betrachte Punkt $p \in \mathbb{R}^2$ oberhalb von ℓ
- Fall 1: p ist näher an ℓ als an jedem Standort
 - es könnte Standort s unter ℓ geben, sodass $p \in \mathcal{V}(s)$
 - p ist also Teil des Graubereichs
- Fall 2: p ist näher an einem Standort s über ℓ als an ℓ
 - es gibt keinen Standort unter ℓ , der näher an p liegt
 - die Voronoi-Zelle von p ist also bekannt

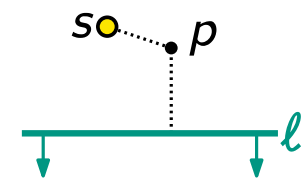
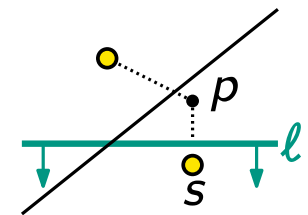


Bekannter Bereich

Wo endet der bekannte Bereich?

(gegeben die Sweep-Line ℓ)

- betrachte Punkt $p \in \mathbb{R}^2$ oberhalb von ℓ
- Fall 1: p ist näher an ℓ als an jedem Standort
 - es könnte Standort s unter ℓ geben, sodass $p \in \mathcal{V}(s)$
 - p ist also Teil des Graubereichs
- Fall 2: p ist näher an einem Standort s über ℓ als an ℓ
 - es gibt keinen Standort unter ℓ , der näher an p liegt
 - die Voronoi-Zelle von p ist also bekannt
- bekannter Bereich: $\{p \in \mathbb{R}^2 \mid \exists s \in S \text{ dist}(p, s) \leq \text{dist}(p, \ell)\}$

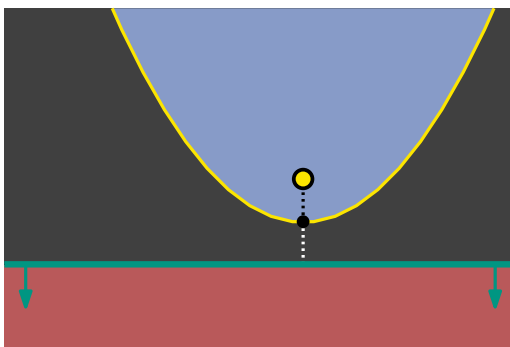
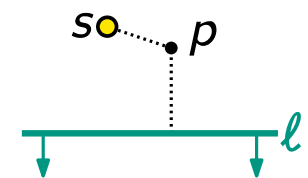
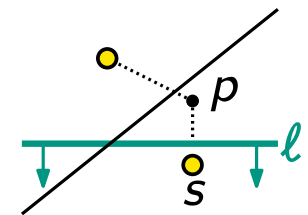


Bekannter Bereich

Wo endet der bekannte Bereich?

(gegeben die Sweep-Line ℓ)

- betrachte Punkt $p \in \mathbb{R}^2$ oberhalb von ℓ
- Fall 1: p ist näher an ℓ als an jedem Standort
 - es könnte Standort s unter ℓ geben, sodass $p \in \mathcal{V}(s)$
 - p ist also Teil des Graubereichs
- Fall 2: p ist näher an einem Standort s über ℓ als an ℓ
 - es gibt keinen Standort unter ℓ , der näher an p liegt
 - die Voronoi-Zelle von p ist also bekannt
- bekannter Bereich: $\{p \in \mathbb{R}^2 \mid \exists s \in S \text{ dist}(p, s) \leq \text{dist}(p, \ell)\}$

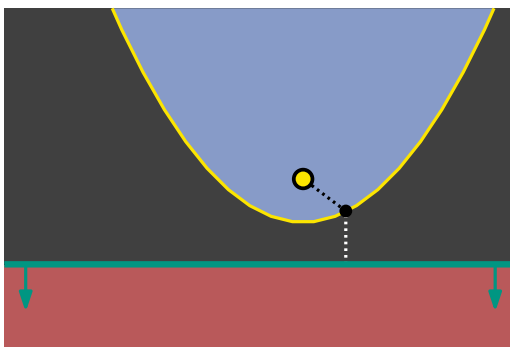
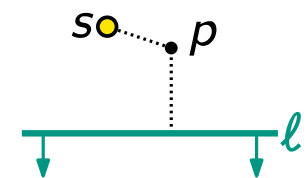
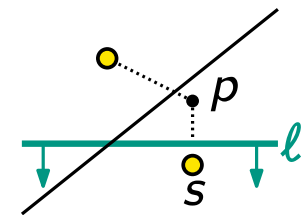


Bekannter Bereich

Wo endet der bekannte Bereich?

(gegeben die Sweep-Line ℓ)

- betrachte Punkt $p \in \mathbb{R}^2$ oberhalb von ℓ
- Fall 1: p ist näher an ℓ als an jedem Standort
 - es könnte Standort s unter ℓ geben, sodass $p \in \mathcal{V}(s)$
 - p ist also Teil des Graubereichs
- Fall 2: p ist näher an einem Standort s über ℓ als an ℓ
 - es gibt keinen Standort unter ℓ , der näher an p liegt
 - die Voronoi-Zelle von p ist also bekannt
- bekannter Bereich: $\{p \in \mathbb{R}^2 \mid \exists s \in S \text{ dist}(p, s) \leq \text{dist}(p, \ell)\}$

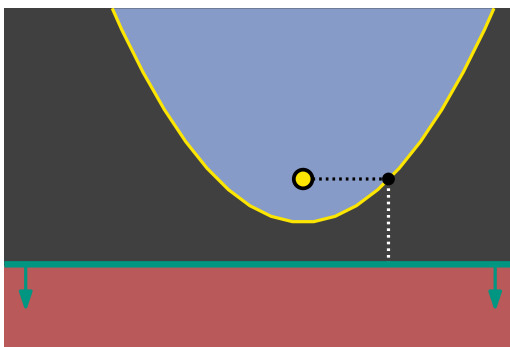
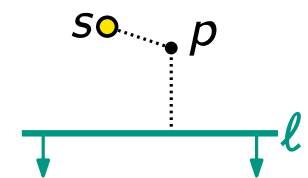
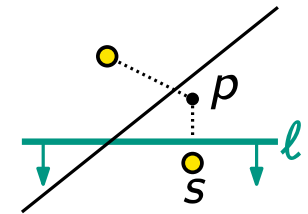


Bekannter Bereich

Wo endet der bekannte Bereich?

(gegeben die Sweep-Line ℓ)

- betrachte Punkt $p \in \mathbb{R}^2$ oberhalb von ℓ
- Fall 1: p ist näher an ℓ als an jedem Standort
 - es könnte Standort s unter ℓ geben, sodass $p \in \mathcal{V}(s)$
 - p ist also Teil des Graubereichs
- Fall 2: p ist näher an einem Standort s über ℓ als an ℓ
 - es gibt keinen Standort unter ℓ , der näher an p liegt
 - die Voronoi-Zelle von p ist also bekannt
- bekannter Bereich: $\{p \in \mathbb{R}^2 \mid \exists s \in S \text{ dist}(p, s) \leq \text{dist}(p, \ell)\}$

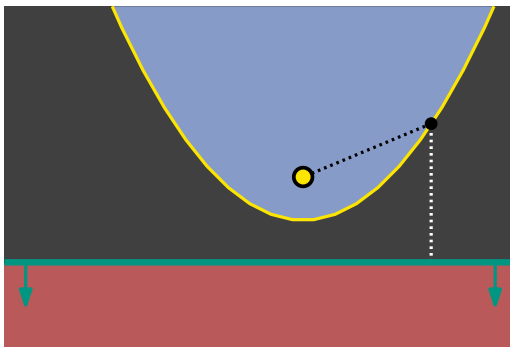
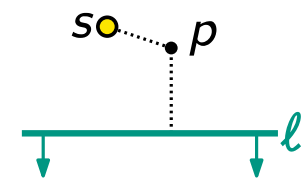
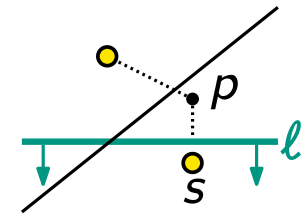


Bekannter Bereich

Wo endet der bekannte Bereich?

(gegeben die Sweep-Line ℓ)

- betrachte Punkt $p \in \mathbb{R}^2$ oberhalb von ℓ
- Fall 1: p ist näher an ℓ als an jedem Standort
 - es könnte Standort s unter ℓ geben, sodass $p \in \mathcal{V}(s)$
 - p ist also Teil des Graubereichs
- Fall 2: p ist näher an einem Standort s über ℓ als an ℓ
 - es gibt keinen Standort unter ℓ , der näher an p liegt
 - die Voronoi-Zelle von p ist also bekannt
- bekannter Bereich: $\{p \in \mathbb{R}^2 \mid \exists s \in S \text{ dist}(p, s) \leq \text{dist}(p, \ell)\}$

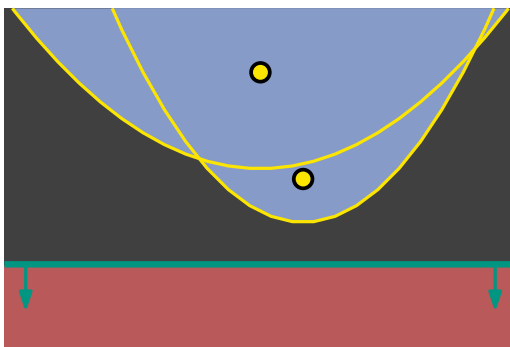
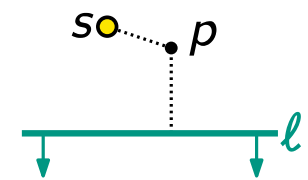
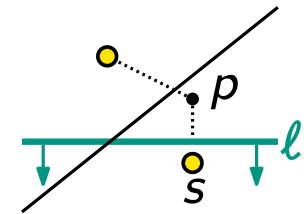


Bekannter Bereich

Wo endet der bekannte Bereich?

(gegeben die Sweep-Line ℓ)

- betrachte Punkt $p \in \mathbb{R}^2$ oberhalb von ℓ
- Fall 1: p ist näher an ℓ als an jedem Standort
 - es könnte Standort s unter ℓ geben, sodass $p \in \mathcal{V}(s)$
 - p ist also Teil des Graubereichs
- Fall 2: p ist näher an einem Standort s über ℓ als an ℓ
 - es gibt keinen Standort unter ℓ , der näher an p liegt
 - die Voronoi-Zelle von p ist also bekannt
- bekannter Bereich: $\{p \in \mathbb{R}^2 \mid \exists s \in S \text{ dist}(p, s) \leq \text{dist}(p, \ell)\}$

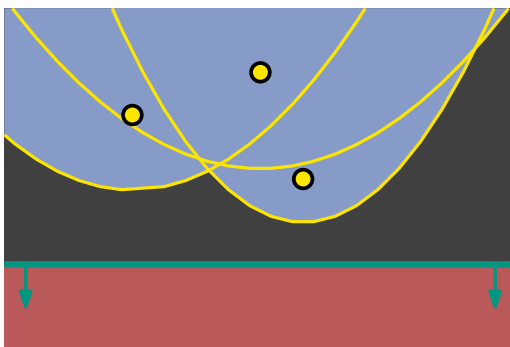
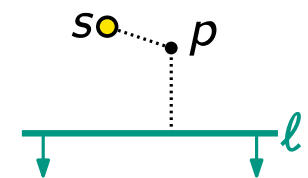
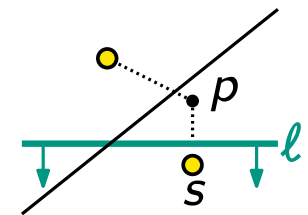


Bekannter Bereich

Wo endet der bekannte Bereich?

(gegeben die Sweep-Line ℓ)

- betrachte Punkt $p \in \mathbb{R}^2$ oberhalb von ℓ
- Fall 1: p ist näher an ℓ als an jedem Standort
 - es könnte Standort s unter ℓ geben, sodass $p \in \mathcal{V}(s)$
 - p ist also Teil des Graubereichs
- Fall 2: p ist näher an einem Standort s über ℓ als an ℓ
 - es gibt keinen Standort unter ℓ , der näher an p liegt
 - die Voronoi-Zelle von p ist also bekannt
- bekannter Bereich: $\{p \in \mathbb{R}^2 \mid \exists s \in S \text{ dist}(p, s) \leq \text{dist}(p, \ell)\}$

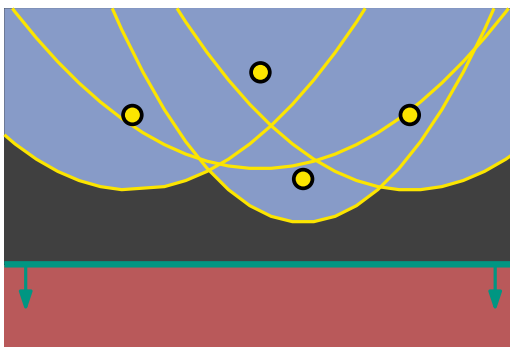
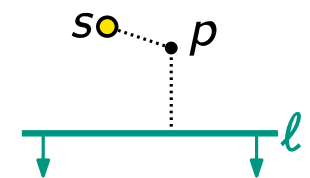
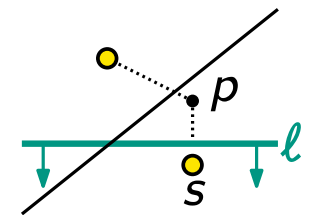


Bekannter Bereich

Wo endet der bekannte Bereich?

(gegeben die Sweep-Line ℓ)

- betrachte Punkt $p \in \mathbb{R}^2$ oberhalb von ℓ
- Fall 1: p ist näher an ℓ als an jedem Standort
 - es könnte Standort s unter ℓ geben, sodass $p \in \mathcal{V}(s)$
 - p ist also Teil des Graubereichs
- Fall 2: p ist näher an einem Standort s über ℓ als an ℓ
 - es gibt keinen Standort unter ℓ , der näher an p liegt
 - die Voronoi-Zelle von p ist also bekannt
- bekannter Bereich: $\{p \in \mathbb{R}^2 \mid \exists s \in S \text{ dist}(p, s) \leq \text{dist}(p, \ell)\}$

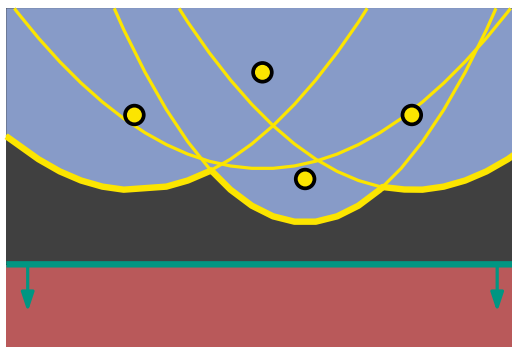
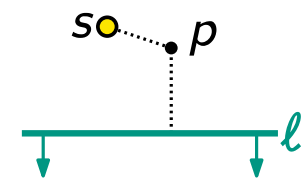
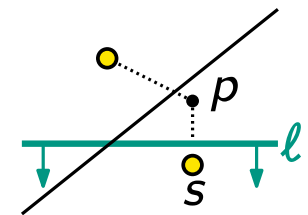


Bekannter Bereich

Wo endet der bekannte Bereich?

(gegeben die Sweep-Line ℓ)

- betrachte Punkt $p \in \mathbb{R}^2$ oberhalb von ℓ
- Fall 1: p ist näher an ℓ als an jedem Standort
 - es könnte Standort s unter ℓ geben, sodass $p \in \mathcal{V}(s)$
 - p ist also Teil des Graubereichs
- Fall 2: p ist näher an einem Standort s über ℓ als an ℓ
 - es gibt keinen Standort unter ℓ , der näher an p liegt
 - die Voronoi-Zelle von p ist also bekannt
- bekannter Bereich: $\{p \in \mathbb{R}^2 \mid \exists s \in S \text{ dist}(p, s) \leq \text{dist}(p, \ell)\}$



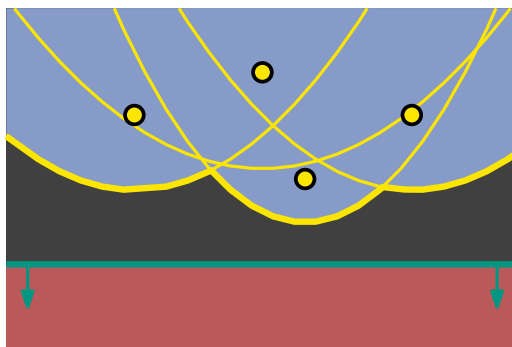
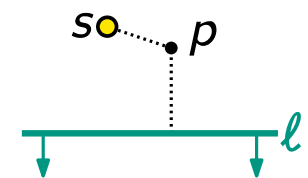
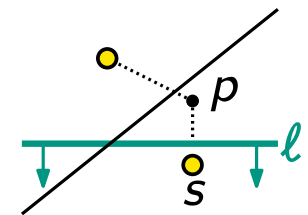
- die Grenze zwischen bekanntem Bereich und Graubereich heißt **Beach-Line**

Bekannter Bereich

Wo endet der bekannte Bereich?

(gegeben die Sweep-Line ℓ)

- betrachte Punkt $p \in \mathbb{R}^2$ oberhalb von ℓ
- Fall 1: p ist näher an ℓ als an jedem Standort
 - es könnte Standort s unter ℓ geben, sodass $p \in \mathcal{V}(s)$
 - p ist also Teil des Graubereichs
- Fall 2: p ist näher an einem Standort s über ℓ als an ℓ
 - es gibt keinen Standort unter ℓ , der näher an p liegt
 - die Voronoi-Zelle von p ist also bekannt
- bekannter Bereich: $\{p \in \mathbb{R}^2 \mid \exists s \in S \text{ dist}(p, s) \leq \text{dist}(p, \ell)\}$



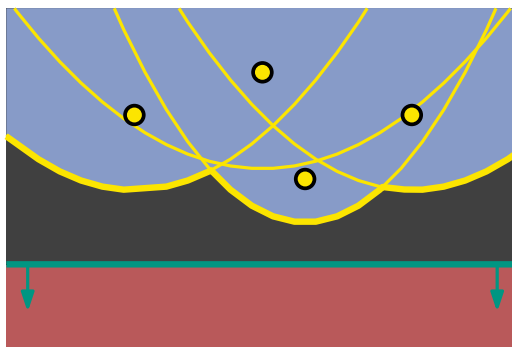
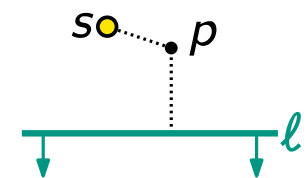
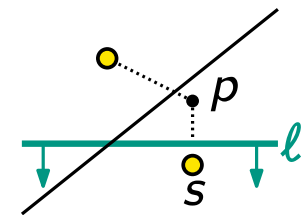
- die Grenze zwischen bekanntem Bereich und Graubereich heißt **Beach-Line**
- jede Teilkurve der Beach-Line gehört zu einem Standort

Bekannter Bereich

Wo endet der bekannte Bereich?

(gegeben die Sweep-Line ℓ)

- betrachte Punkt $p \in \mathbb{R}^2$ oberhalb von ℓ
- Fall 1: p ist näher an ℓ als an jedem Standort
 - es könnte Standort s unter ℓ geben, sodass $p \in \mathcal{V}(s)$
 - p ist also Teil des Graubereichs
- Fall 2: p ist näher an einem Standort s über ℓ als an ℓ
 - es gibt keinen Standort unter ℓ , der näher an p liegt
 - die Voronoi-Zelle von p ist also bekannt
- bekannter Bereich: $\{p \in \mathbb{R}^2 \mid \exists s \in S \text{ dist}(p, s) \leq \text{dist}(p, \ell)\}$

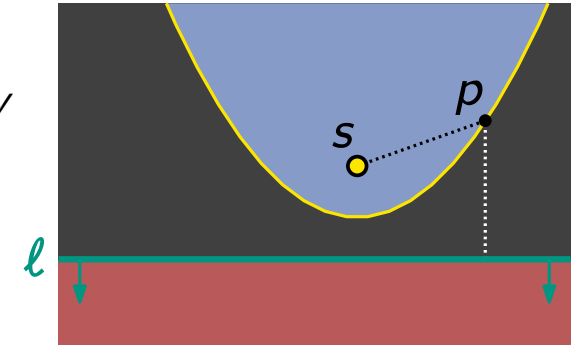


- die Grenze zwischen bekanntem Bereich und Graubereich heißt **Beach-Line**
- jede Teilkurve der Beach-Line gehört zu einem Standort
- gleich: jede Teilkurve ist eine Parabel

Welche Form hat die Beach-Line?

Relevante Koordinaten

- $s = (s_x, s_y)$
- $p = (p_x, p_y)$
- y -Koordinate von l : l_y



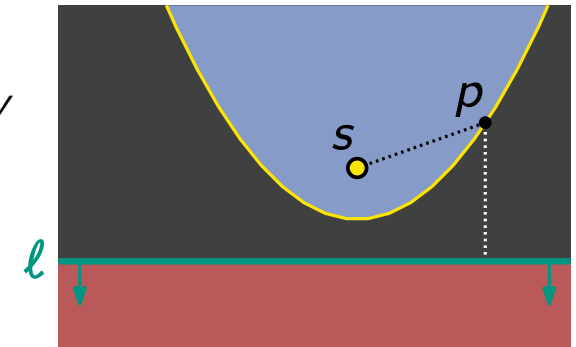
Welche Form hat die Beach-Line?

Relevante Koordinaten

- $s = (s_x, s_y)$
- $p = (p_x, p_y)$
- y -Koordinate von l : l_y

Distanzen

- $\text{dist}(s, p) = \sqrt{(p_x - s_x)^2 + (p_y - s_y)^2}$
- $\text{dist}(l, p) = p_y - l_y$



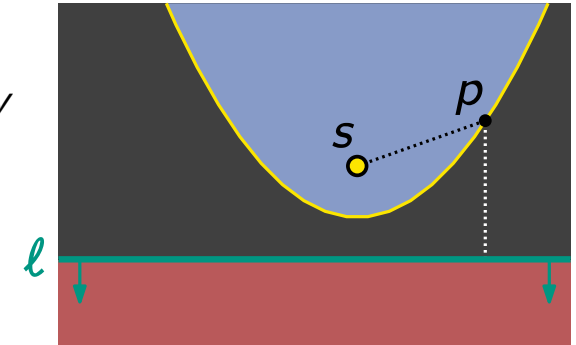
Welche Form hat die Beach-Line?

Relevante Koordinaten

- $s = (s_x, s_y)$
- $p = (p_x, p_y)$
- y -Koordinate von ℓ : ℓ_y

Distanzen

- $\text{dist}(s, p) = \sqrt{(p_x - s_x)^2 + (p_y - s_y)^2}$
- $\text{dist}(\ell, p) = p_y - \ell_y$
- p liegt auf der Beach-Line $\Leftrightarrow \text{dist}(s, p) = \text{dist}(\ell, p)$



Welche Form hat die Beach-Line?

Relevante Koordinaten

- $s = (s_x, s_y)$
- $p = (p_x, p_y)$
- y -Koordinate von l : l_y

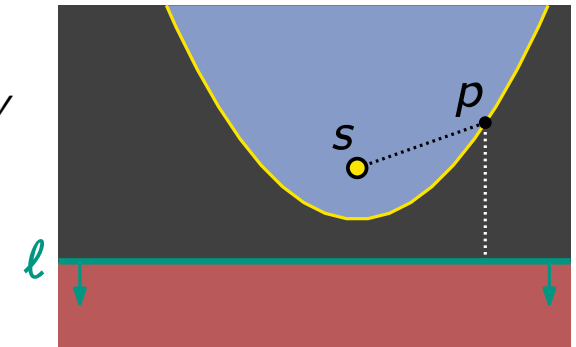
Distanzen

- $\text{dist}(s, p) = \sqrt{(p_x - s_x)^2 + (p_y - s_y)^2}$

- $\text{dist}(l, p) = p_y - l_y$

- p liegt auf der Beach-Line $\Leftrightarrow \text{dist}(s, p) = \text{dist}(l, p)$

$$(p_x - s_x)^2 + (p_y - s_y)^2 = (p_y - l_y)^2$$



Welche Form hat die Beach-Line?

Relevante Koordinaten

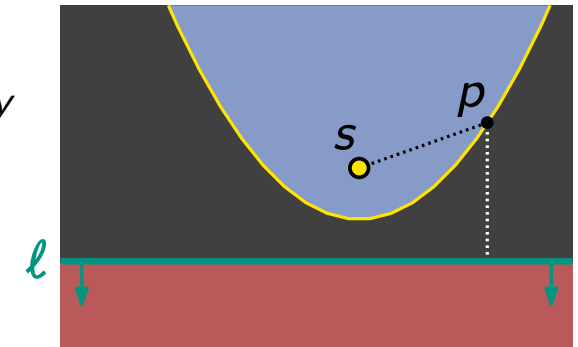
- $s = (s_x, s_y)$
- $p = (p_x, p_y)$
- y -Koordinate von l : l_y

Distanzen

- $\text{dist}(s, p) = \sqrt{(p_x - s_x)^2 + (p_y - s_y)^2}$
- $\text{dist}(l, p) = p_y - l_y$
- p liegt auf der Beach-Line $\Leftrightarrow \text{dist}(s, p) = \text{dist}(l, p)$

$$(p_x - s_x)^2 + (p_y - s_y)^2 = (p_y - l_y)^2$$

$$(p_x - s_x)^2 + p_y^2 - 2p_y s_y + s_y^2 = p_y^2 - 2p_y l_y + l_y^2$$



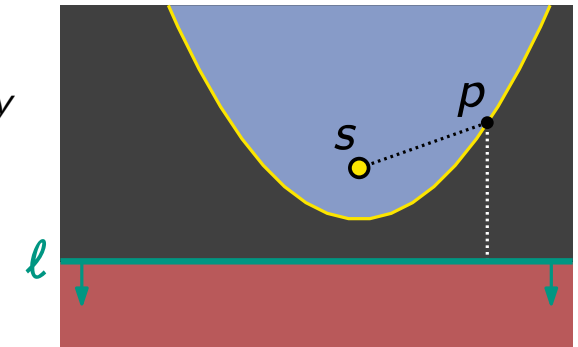
Welche Form hat die Beach-Line?

Relevante Koordinaten

- $s = (s_x, s_y)$
- $p = (p_x, p_y)$
- y -Koordinate von l : l_y

Distanzen

- $\text{dist}(s, p) = \sqrt{(p_x - s_x)^2 + (p_y - s_y)^2}$
- $\text{dist}(l, p) = p_y - l_y$
- p liegt auf der Beach-Line $\Leftrightarrow \text{dist}(s, p) = \text{dist}(l, p)$



$$(p_x - s_x)^2 + (p_y - s_y)^2 = (p_y - l_y)^2$$

$$(p_x - s_x)^2 + p_y^2 - 2p_y s_y + s_y^2 = p_y^2 - 2p_y l_y + l_y^2$$

$$(p_x - s_x)^2 + s_y^2 - l_y^2 = 2p_y s_y - 2p_y l_y$$

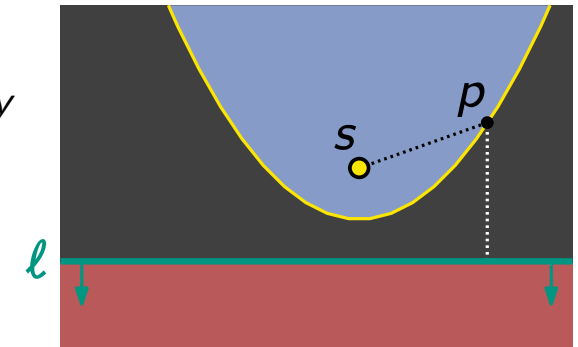
Welche Form hat die Beach-Line?

Relevante Koordinaten

- $s = (s_x, s_y)$
- $p = (p_x, p_y)$
- y -Koordinate von l : l_y

Distanzen

- $\text{dist}(s, p) = \sqrt{(p_x - s_x)^2 + (p_y - s_y)^2}$
- $\text{dist}(l, p) = p_y - l_y$
- p liegt auf der Beach-Line $\Leftrightarrow \text{dist}(s, p) = \text{dist}(l, p)$



$$(p_x - s_x)^2 + (p_y - s_y)^2 = (p_y - l_y)^2$$

$$(p_x - s_x)^2 + p_y^2 - 2p_y s_y + s_y^2 = p_y^2 - 2p_y l_y + l_y^2$$

$$(p_x - s_x)^2 + s_y^2 - l_y^2 = 2p_y s_y - 2p_y l_y$$

$$\frac{(p_x - s_x)^2}{2s_y - 2l_y} + \frac{s_y^2 - l_y^2}{2s_y - 2l_y} = p_y$$

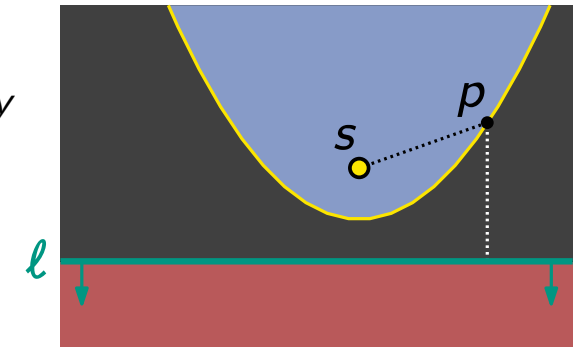
Welche Form hat die Beach-Line?

Relevante Koordinaten

- $s = (s_x, s_y)$
- $p = (p_x, p_y)$
- y -Koordinate von l : l_y

Distanzen

- $\text{dist}(s, p) = \sqrt{(p_x - s_x)^2 + (p_y - s_y)^2}$
- $\text{dist}(l, p) = p_y - l_y$
- p liegt auf der Beach-Line $\Leftrightarrow \text{dist}(s, p) = \text{dist}(l, p)$



$$\begin{aligned}
 (p_x - s_x)^2 + (p_y - s_y)^2 &= (p_y - l_y)^2 \\
 (p_x - s_x)^2 + p_y^2 - 2p_y s_y + s_y^2 &= p_y^2 - 2p_y l_y + l_y^2 \\
 (p_x - s_x)^2 + s_y^2 - l_y^2 &= 2p_y s_y - 2p_y l_y \\
 \frac{(p_x - s_x)^2}{2s_y - 2l_y} + \frac{s_y^2 - l_y^2}{2s_y - 2l_y} &= p_y \\
 \frac{(p_x - s_x)^2}{2s_y - 2l_y} + \frac{s_y + l_y}{2} &= p_y
 \end{aligned}$$

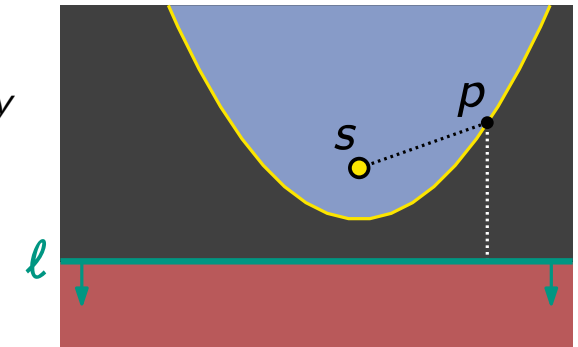
Welche Form hat die Beach-Line?

Relevante Koordinaten

- $s = (s_x, s_y)$
- $p = (p_x, p_y)$
- y -Koordinate von l : l_y

Distanzen

- $\text{dist}(s, p) = \sqrt{(p_x - s_x)^2 + (p_y - s_y)^2}$
- $\text{dist}(l, p) = p_y - l_y$
- p liegt auf der Beach-Line $\Leftrightarrow \text{dist}(s, p) = \text{dist}(l, p)$



$$\begin{aligned}
 (p_x - s_x)^2 + (p_y - s_y)^2 &= (p_y - l_y)^2 \\
 (p_x - s_x)^2 + p_y^2 - 2p_y s_y + s_y^2 &= p_y^2 - 2p_y l_y + l_y^2 \\
 (p_x - s_x)^2 + s_y^2 - l_y^2 &= 2p_y s_y - 2p_y l_y \\
 \frac{(p_x - s_x)^2}{2s_y - 2l_y} + \frac{s_y^2 - l_y^2}{2s_y - 2l_y} &= p_y
 \end{aligned}$$

Verschiebung in x -Richtung	$\frac{(p_x - s_x)^2}{2s_y - 2l_y}$	+	$\frac{s_y + l_y}{2}$	=	p_y	Parabel
						Verschiebung in y -Richtung

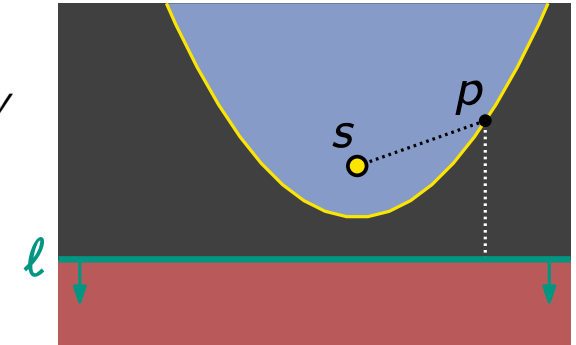
Welche Form hat die Beach-Line?

Relevante Koordinaten

- $s = (s_x, s_y)$
- $p = (p_x, p_y)$
- y -Koordinate von ℓ : ℓ_y

Distanzen

- $\text{dist}(s, p) = \sqrt{(p_x - s_x)^2 + (p_y - s_y)^2}$
- $\text{dist}(\ell, p) = p_y - \ell_y$
- p liegt auf der Beach-Line $\Leftrightarrow \text{dist}(s, p) = \text{dist}(\ell, p)$



$$\begin{aligned} (p_x - s_x)^2 + (p_y - s_y)^2 &= (p_y - \ell_y)^2 \\ (p_x - s_x)^2 + p_y^2 - 2p_y s_y + s_y^2 &= p_y^2 - 2p_y \ell_y + \ell_y^2 \\ (p_x - s_x)^2 + s_y^2 - \ell_y^2 &= 2p_y s_y - 2p_y \ell_y \\ \frac{(p_x - s_x)^2}{2s_y - 2\ell_y} + \frac{s_y^2 - \ell_y^2}{2s_y - 2\ell_y} &= p_y \end{aligned}$$

Verschiebung in x -Richtung $\frac{(p_x - s_x)^2}{2s_y - 2\ell_y} + \frac{s_y + \ell_y}{2} = p_y$ Parabel

Verschiebung in y -Richtung

- Parabel mit Scheitelpunkt auf halber Strecke zwischen s und ℓ
- je größer $\text{dist}(s, \ell)$, desto weiter die Parabel

Wann verändert sich die Beach-Line?

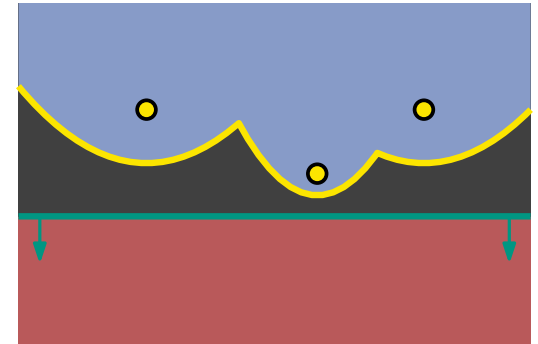
Sweep-Line Algo

- Sweep-Line Status: Zustand der Beach-Line
- Events: Zeitpunkte an denen sich die BL ändert

Wann verändert sich die Beach-Line?

Sweep-Line Algo

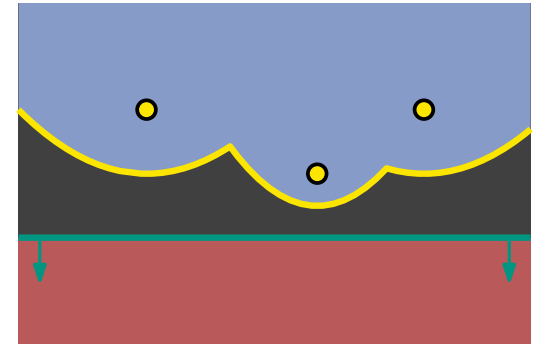
- Sweep-Line Status: Zustand der Beach-Line
- Events: Zeitpunkte an denen sich die BL ändert
- Was heißt „ändert“ in diesem Zusammenhang?



Wann verändert sich die Beach-Line?

Sweep-Line Algo

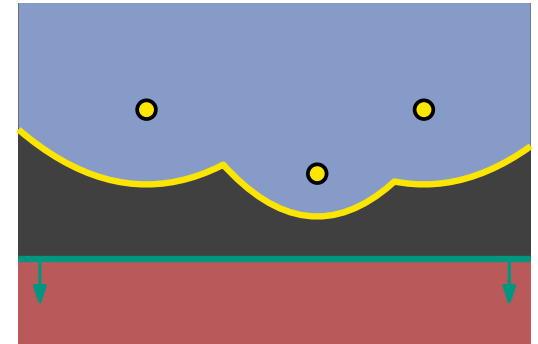
- Sweep-Line Status: Zustand der Beach-Line
- Events: Zeitpunkte an denen sich die BL ändert
- Was heißt „ändert“ in diesem Zusammenhang?



Wann verändert sich die Beach-Line?

Sweep-Line Algo

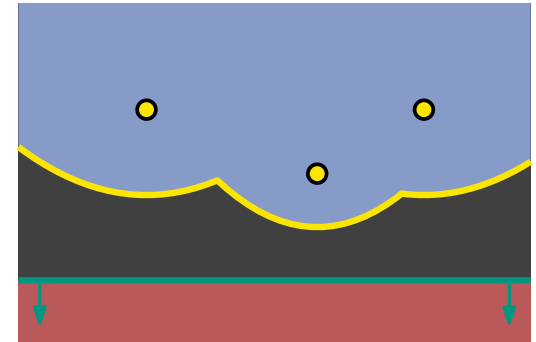
- Sweep-Line Status: Zustand der Beach-Line
- Events: Zeitpunkte an denen sich die BL ändert
- Was heißt „ändert“ in diesem Zusammenhang?



Wann verändert sich die Beach-Line?

Sweep-Line Algo

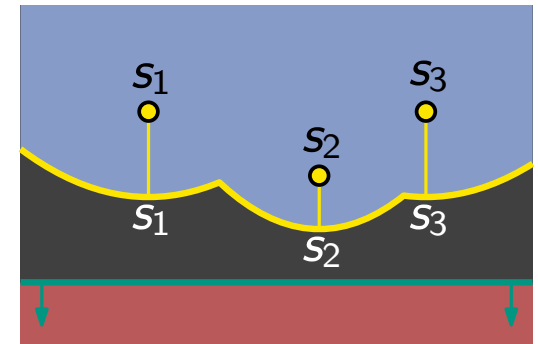
- Sweep-Line Status: Zustand der Beach-Line
- Events: Zeitpunkte an denen sich die BL ändert
- Was heißt „ändert“ in diesem Zusammenhang?
 - ändert sich irgendwie ständig



Wann verändert sich die Beach-Line?

Sweep-Line Algo

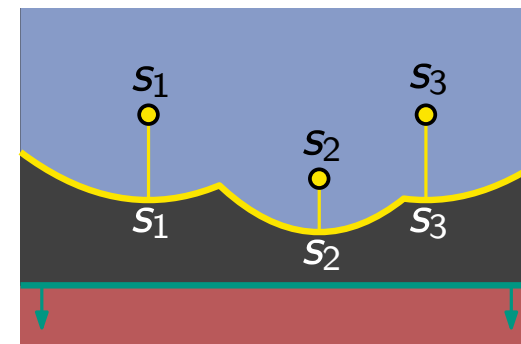
- Sweep-Line Status: Zustand der Beach-Line
- Events: Zeitpunkte an denen sich die BL ändert
- Was heißt „ändert“ in diesem Zusammenhang?
 - ändert sich irgendwie ständig
 - Sequenz zugehöriger Standorte ändert sich selten



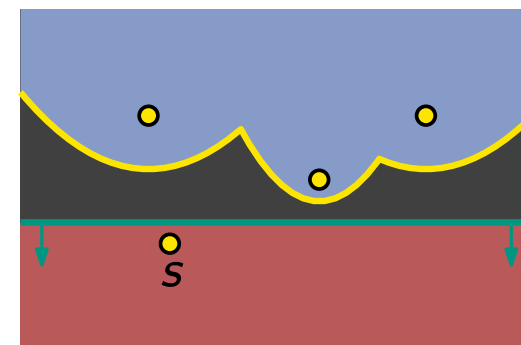
Wann verändert sich die Beach-Line?

Sweep-Line Algo

- Sweep-Line Status: Zustand der Beach-Line
- Events: Zeitpunkte an denen sich die BL ändert
- Was heißt „ändert“ in diesem Zusammenhang?
 - ändert sich irgendwie ständig
 - Sequenz zugehöriger Standorte ändert sich selten



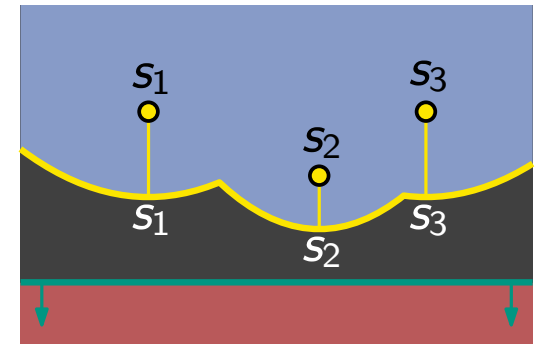
Event-Punkt: neuer Standort s



Wann verändert sich die Beach-Line?

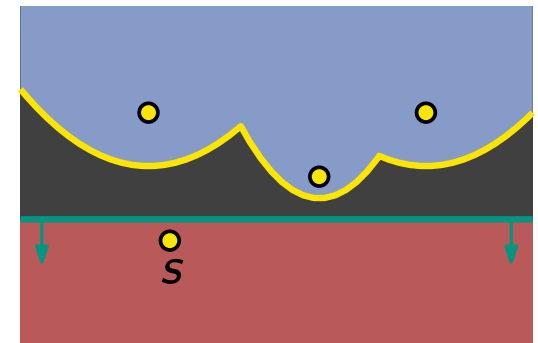
Sweep-Line Algo

- Sweep-Line Status: Zustand der Beach-Line
- Events: Zeitpunkte an denen sich die BL ändert
- Was heißt „ändert“ in diesem Zusammenhang?
 - ändert sich irgendwie ständig
 - Sequenz zugehöriger Standorte ändert sich selten



Event-Punkt: neuer Standort s

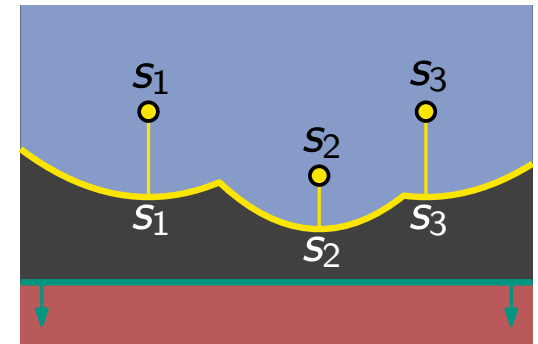
- neue Parabel wird in Beach-Line eingefügt



Wann verändert sich die Beach-Line?

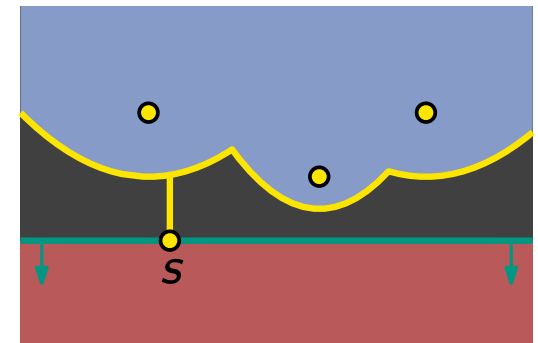
Sweep-Line Algo

- Sweep-Line Status: Zustand der Beach-Line
- Events: Zeitpunkte an denen sich die BL ändert
- Was heißt „ändert“ in diesem Zusammenhang?
 - ändert sich irgendwie ständig
 - Sequenz zugehöriger Standorte ändert sich selten



Event-Punkt: neuer Standort s

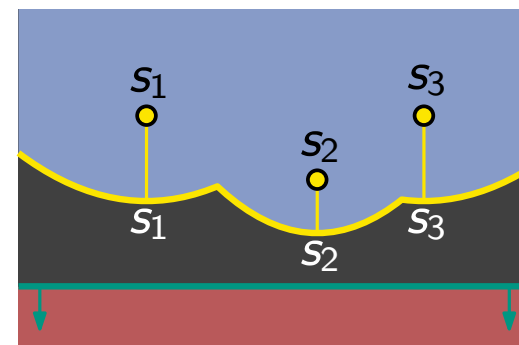
- neue Parabel wird in Beach-Line eingefügt



Wann verändert sich die Beach-Line?

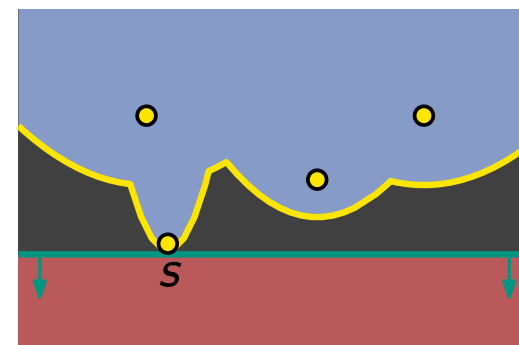
Sweep-Line Algo

- Sweep-Line Status: Zustand der Beach-Line
- Events: Zeitpunkte an denen sich die BL ändert
- Was heißt „ändert“ in diesem Zusammenhang?
 - ändert sich irgendwie ständig
 - Sequenz zugehöriger Standorte ändert sich selten



Event-Punkt: neuer Standort s

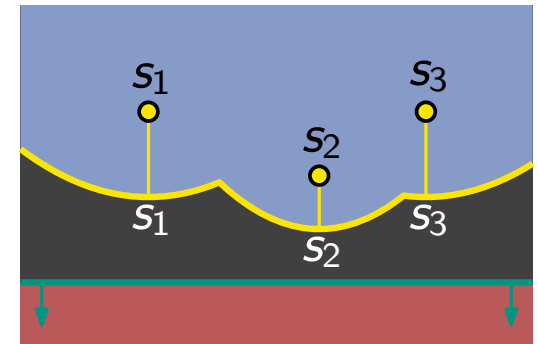
- neue Parabel wird in Beach-Line eingefügt



Wann verändert sich die Beach-Line?

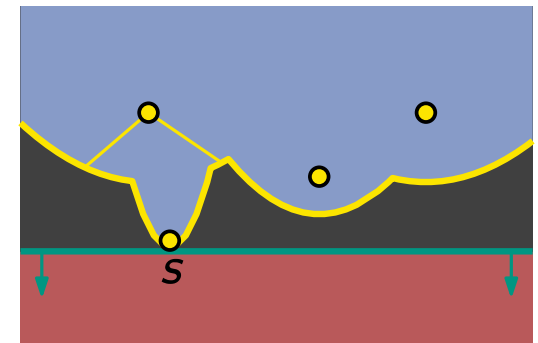
Sweep-Line Algo

- Sweep-Line Status: Zustand der Beach-Line
- Events: Zeitpunkte an denen sich die BL ändert
- Was heißt „ändert“ in diesem Zusammenhang?
 - ändert sich irgendwie ständig
 - Sequenz zugehöriger Standorte ändert sich selten



Event-Punkt: neuer Standort s

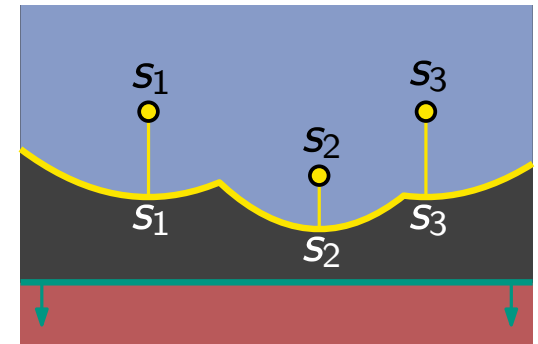
- neue Parabel wird in Beach-Line eingefügt
- spalte Parabel aus aktueller BL über s auf



Wann verändert sich die Beach-Line?

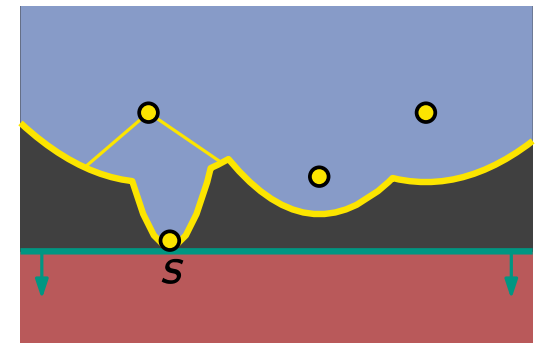
Sweep-Line Algo

- Sweep-Line Status: Zustand der Beach-Line
- Events: Zeitpunkte an denen sich die BL ändert
- Was heißt „ändert“ in diesem Zusammenhang?
 - ändert sich irgendwie ständig
 - Sequenz zugehöriger Standorte ändert sich selten



Event-Punkt: neuer Standort s

- neue Parabel wird in Beach-Line eingefügt
- spalte Parabel aus aktueller BL über s auf

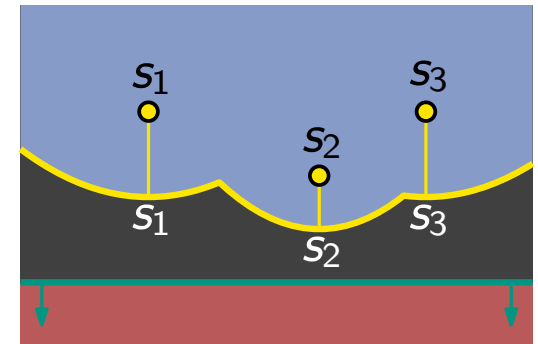


Event-Punkt: Parabelstück verschwindet

Wann verändert sich die Beach-Line?

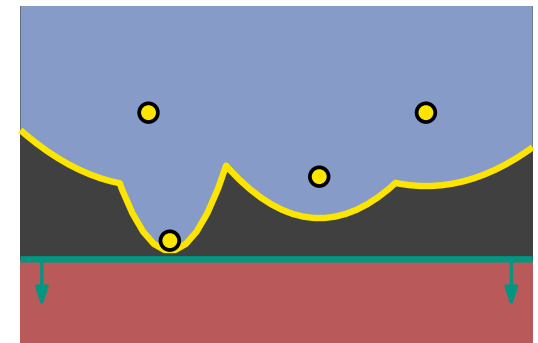
Sweep-Line Algo

- Sweep-Line Status: Zustand der Beach-Line
- Events: Zeitpunkte an denen sich die BL ändert
- Was heißt „ändert“ in diesem Zusammenhang?
 - ändert sich irgendwie ständig
 - Sequenz zugehöriger Standorte ändert sich selten



Event-Punkt: neuer Standort s

- neue Parabel wird in Beach-Line eingefügt
- spalte Parabel aus aktueller BL über s auf

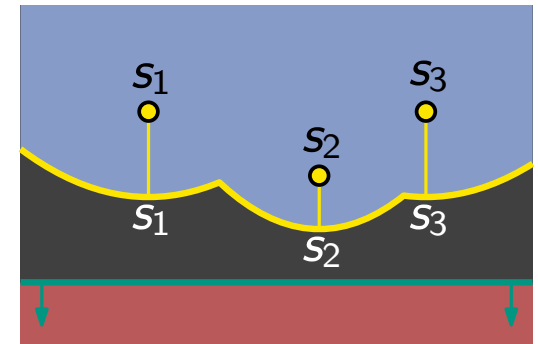


Event-Punkt: Parabelstück verschwindet

Wann verändert sich die Beach-Line?

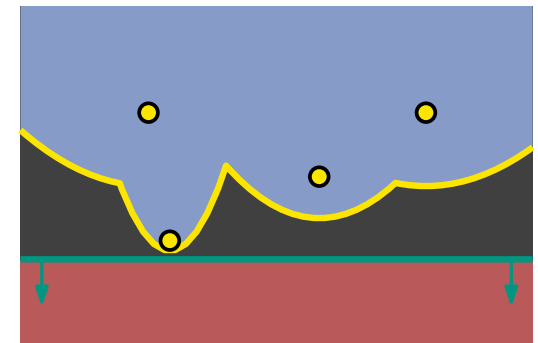
Sweep-Line Algo

- Sweep-Line Status: Zustand der Beach-Line
- Events: Zeitpunkte an denen sich die BL ändert
- Was heißt „ändert“ in diesem Zusammenhang?
 - ändert sich irgendwie ständig
 - Sequenz zugehöriger Standorte ändert sich selten



Event-Punkt: neuer Standort s

- neue Parabel wird in Beach-Line eingefügt
- spalte Parabel aus aktueller BL über s auf



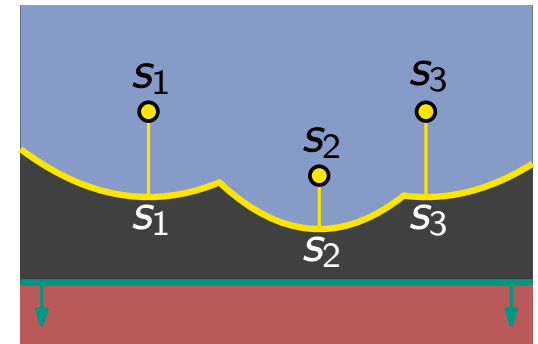
Event-Punkt: Parabelstück verschwindet

- Wie können wir wissen, wann das passiert?

Wann verändert sich die Beach-Line?

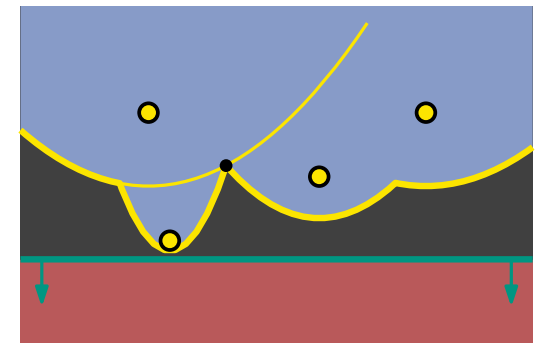
Sweep-Line Algo

- Sweep-Line Status: Zustand der Beach-Line
- Events: Zeitpunkte an denen sich die BL ändert
- Was heißt „ändert“ in diesem Zusammenhang?
 - ändert sich irgendwie ständig
 - Sequenz zugehöriger Standorte ändert sich selten



Event-Punkt: neuer Standort s

- neue Parabel wird in Beach-Line eingefügt
- spalte Parabel aus aktueller BL über s auf



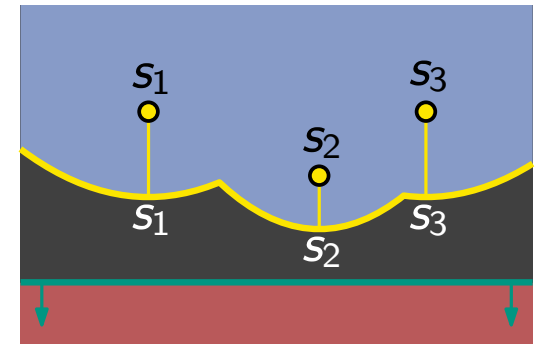
Event-Punkt: Parabelstück verschwindet

- Wie können wir wissen, wann das passiert?
- die Parabeln dreier Punkte schneiden sich dort

Wann verändert sich die Beach-Line?

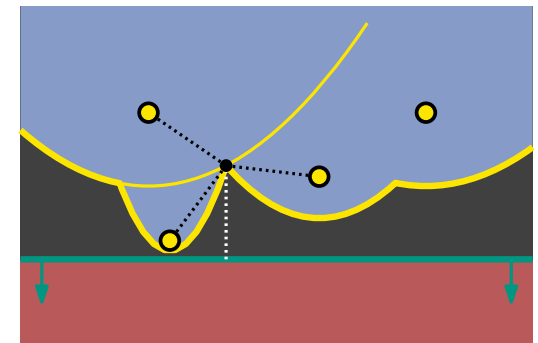
Sweep-Line Algo

- Sweep-Line Status: Zustand der Beach-Line
- Events: Zeitpunkte an denen sich die BL ändert
- Was heißt „ändert“ in diesem Zusammenhang?
 - ändert sich irgendwie ständig
 - Sequenz zugehöriger Standorte ändert sich selten



Event-Punkt: neuer Standort s

- neue Parabel wird in Beach-Line eingefügt
- spalte Parabel aus aktueller BL über s auf



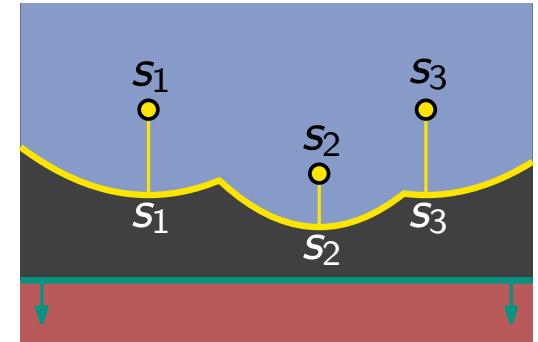
Event-Punkt: Parabelstück verschwindet

- Wie können wir wissen, wann das passiert?
- die Parabeln dreier Punkte schneiden sich dort
- Schnittpunkt hat von Standorten und Sweep-Line denselben Abstand

Wann verändert sich die Beach-Line?

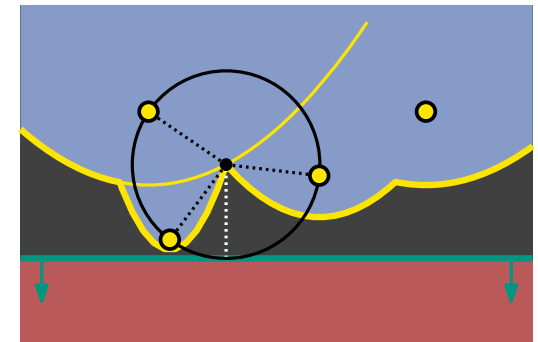
Sweep-Line Algo

- Sweep-Line Status: Zustand der Beach-Line
- Events: Zeitpunkte an denen sich die BL ändert
- Was heißt „ändert“ in diesem Zusammenhang?
 - ändert sich irgendwie ständig
 - Sequenz zugehöriger Standorte ändert sich selten



Event-Punkt: neuer Standort s

- neue Parabel wird in Beach-Line eingefügt
- spalte Parabel aus aktueller BL über s auf



Event-Punkt: Parabelstück verschwindet

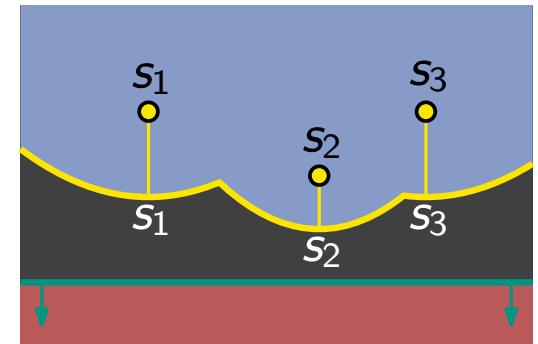
- Wie können wir wissen, wann das passiert?
- die Parabeln dreier Punkte schneiden sich dort
- Schnittpunkt hat von Standorten und Sweep-Line denselben Abstand
- kein anderer Standort ist dem Schnittpunkt näher

Warum nicht?

Wann verändert sich die Beach-Line?

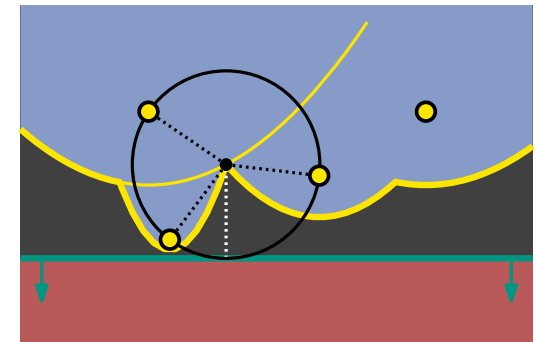
Sweep-Line Algo

- Sweep-Line Status: Zustand der Beach-Line
- Events: Zeitpunkte an denen sich die BL ändert
- Was heißt „ändert“ in diesem Zusammenhang?
 - ändert sich irgendwie ständig
 - Sequenz zugehöriger Standorte ändert sich selten



Event-Punkt: neuer Standort s

- neue Parabel wird in Beach-Line eingefügt
- spalte Parabel aus aktueller BL über s auf



Event-Punkt: Parabelstück verschwindet

- Wie können wir wissen, wann das passiert?
- die Parabeln dreier Punkte schneiden sich dort
- Schnittpunkt hat von Standorten und Sweep-Line denselben Abstand
- kein anderer Standort ist dem Schnittpunkt näher Warum nicht?
- diese Events heißen Kreis-Events
- definiert durch die Standorte dreier aufeinanderfolgender Parabelstücke

Zwischenstand

Zwischenstand

Event-Punkte: Veränderung der Beach-Line

- Standort-Event → neues Parabelstück einfügen
- Kreis-Event → Parabelstück verschwindet

(nicht bewiesen: das sind die einzigen Situationen, in denen sich die BL ändert)

Zwischenstand

Event-Punkte: Veränderung der Beach-Line

- Standort-Event → neues Parabelstück einfügen
 - Kreis-Event → Parabelstück verschwindet
- (nicht bewiesen: das sind die einzigen Situationen, in denen sich die BL ändert)

Kreis-Events finden

- bei jeder Veränderung der Beach-Line:
neue Tripel konsekutiver Parabelstücke → ggf. neues Kreis-Event

Zwischenstand

Event-Punkte: Veränderung der Beach-Line

- Standort-Event → neues Parabelstück einfügen
 - Kreis-Event → Parabelstück verschwindet
- (nicht bewiesen: das sind die einzigen Situationen, in denen sich die BL ändert)

Kreis-Events finden

- bei jeder Veränderung der Beach-Line:
neue Tripel konsekutiver Parabelstücke → ggf. neues Kreis-Event

Voronoi-Diagramm

- Kreis-Event → neuer Knoten im Voronoi Diagramm
- jeder Knoten wird so gefunden **Warum?**

Zwischenstand

Event-Punkte: Veränderung der Beach-Line

- Standort-Event → neues Parabelstück einfügen
 - Kreis-Event → Parabelstück verschwindet
- (nicht bewiesen: das sind die einzigen Situationen, in denen sich die BL ändert)

Kreis-Events finden

- bei jeder Veränderung der Beach-Line:
neue Tripel konsekutiver Parabelstücke → ggf. neues Kreis-Event

Voronoi-Diagramm

- Kreis-Event → neuer Knoten im Voronoi Diagramm
- jeder Knoten wird so gefunden **Warum?**

Offene Fragen

- Wie verwalten wir die Beach-Line?

Zwischenstand

Event-Punkte: Veränderung der Beach-Line

- Standort-Event → neues Parabelstück einfügen
 - Kreis-Event → Parabelstück verschwindet
- (nicht bewiesen: das sind die einzigen Situationen, in denen sich die BL ändert)

Kreis-Events finden

- bei jeder Veränderung der Beach-Line:
neue Tripel konsekutiver Parabelstücke → ggf. neues Kreis-Event

Voronoi-Diagramm

- Kreis-Event → neuer Knoten im Voronoi Diagramm
- jeder Knoten wird so gefunden **Warum?**

Offene Fragen

- Wie verwalten wir die Beach-Line?
- Wie verwalten wir die Events?

Zwischenstand

Event-Punkte: Veränderung der Beach-Line

- Standort-Event → neues Parabelstück einfügen
 - Kreis-Event → Parabelstück verschwindet
- (nicht bewiesen: das sind die einzigen Situationen, in denen sich die BL ändert)

Kreis-Events finden

- bei jeder Veränderung der Beach-Line:
neue Tripel konsekutiver Parabelstücke → ggf. neues Kreis-Event

Voronoi-Diagramm

- Kreis-Event → neuer Knoten im Voronoi Diagramm
- jeder Knoten wird so gefunden **Warum?**

Offene Fragen

- Wie verwalten wir die Beach-Line?
- Wie verwalten wir die Events?
- Wie bekommen wir am Ende das Voronoi-Diagramm?

Zwischenstand

Event-Punkte: Veränderung der Beach-Line

- Standort-Event → neues Parabelstück einfügen
 - Kreis-Event → Parabelstück verschwindet
- (nicht bewiesen: das sind die einzigen Situationen, in denen sich die BL ändert)

Kreis-Events finden

- bei jeder Veränderung der Beach-Line:
neue Tripel konsekutiver Parabelstücke → ggf. neues Kreis-Event

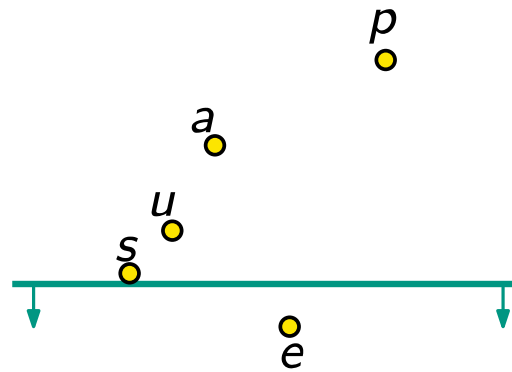
Voronoi-Diagramm

- Kreis-Event → neuer Knoten im Voronoi Diagramm
- jeder Knoten wird so gefunden **Warum?**

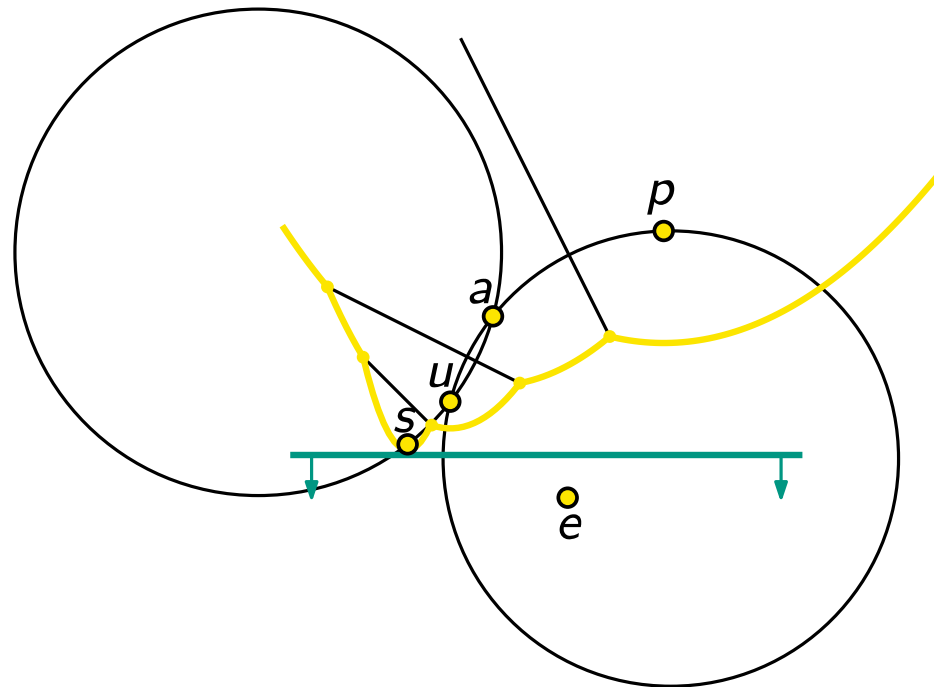
Offene Fragen

- Wie verwalten wir die Beach-Line?
- Wie verwalten wir die Events?
- Wie bekommen wir am Ende das Voronoi-Diagramm?
- Was machen wir mit Sonderfällen?
 - Standorte mit gleicher y -Koordinate
 - Knoten mit Grad > 3
 - sonstige?

Wie viele Events sind in der Queue?

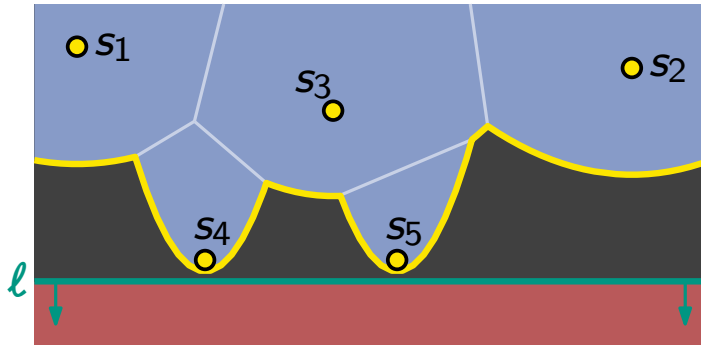


Wie viele Events sind in der Queue?



Antwort: 3 (ein Standort- und zwei Kreis-Events)

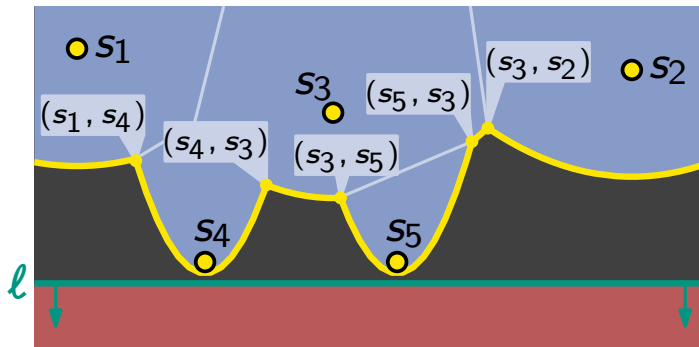
Verwaltung der Beach-Line



Beobachtung

- Parabeln können in mehrere Teile zerlegt sein
- es genügt nicht den Standort zu kennen

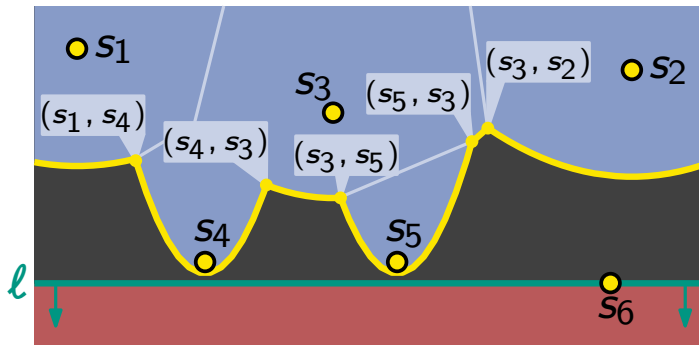
Verwaltung der Beach-Line



Beobachtung

- Parabeln können in mehrere Teile zerlegt sein
- es genügt nicht den Standort zu kennen
- speichere stattdessen eine Repräsentation der Schnittpunkte, die von l unabhängig ist

Verwaltung der Beach-Line

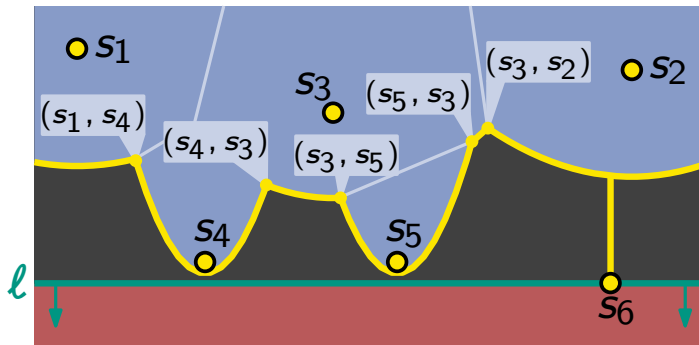


Beobachtung

- Parabeln können in mehrere Teile zerlegt sein
- es genügt nicht den Standort zu kennen
- speichere stattdessen eine Repräsentation der Schnittpunkte, die von l unabhängig ist

Standort-Event

Verwaltung der Beach-Line



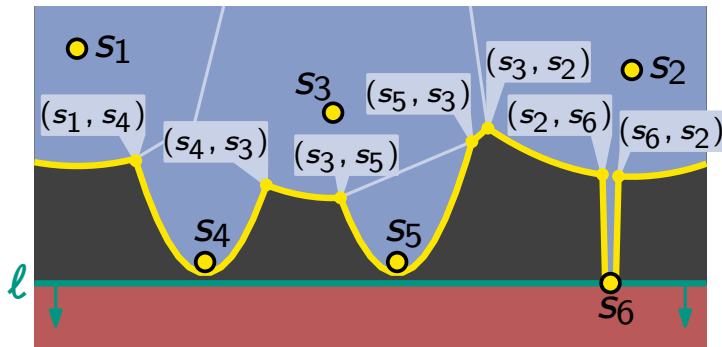
Beobachtung

- Parabeln können in mehrere Teile zerlegt sein
- es genügt nicht den Standort zu kennen
- speichere stattdessen eine Repräsentation der Schnittpunkte, die von ℓ unabhängig ist

Standort-Event

- finde x -Koordinate des neuen Standorts in der Menge der Schnittpunkte

Verwaltung der Beach-Line



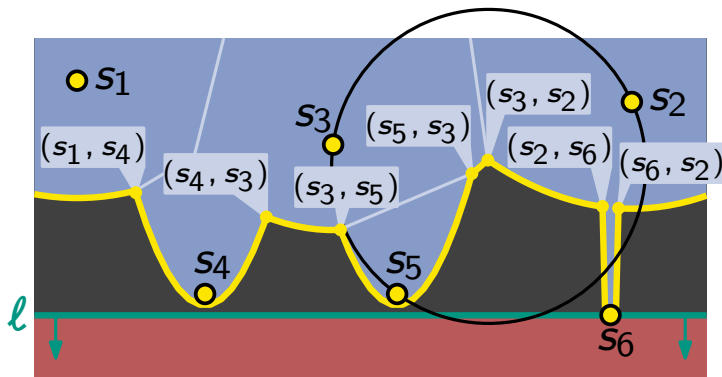
Beobachtung

- Parabeln können in mehrere Teile zerlegt sein
- es genügt nicht den Standort zu kennen
- speichere stattdessen eine Repräsentation der Schnittpunkte, die von ℓ unabhängig ist

Standort-Event

- finde x -Koordinate des neuen Standorts in der Menge der Schnittpunkte
- füge zwei neue Schnittpunkte ein

Verwaltung der Beach-Line



Beobachtung

- Parabeln können in mehrere Teile zerlegt sein
- es genügt nicht den Standort zu kennen
- speichere stattdessen eine Repräsentation der Schnittpunkte, die von l unabhängig ist

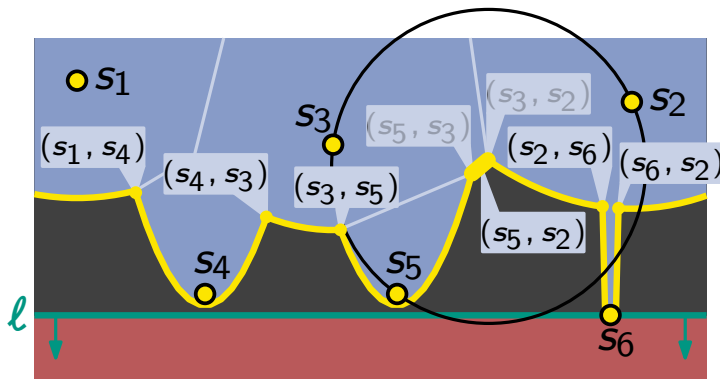
Standort-Event

- finde x -Koordinate des neuen Standorts in der Menge der Schnittpunkte
- füge zwei neue Schnittpunkte ein

Kreis-Event

- gehört zu zwei konsekutiven Schnittpunkten (hier: (s_5, s_3) und (s_3, s_2))

Verwaltung der Beach-Line



Beobachtung

- Parabeln können in mehrere Teile zerlegt sein
- es genügt nicht den Standort zu kennen
- speichere stattdessen eine Repräsentation der Schnittpunkte, die von ℓ unabhängig ist

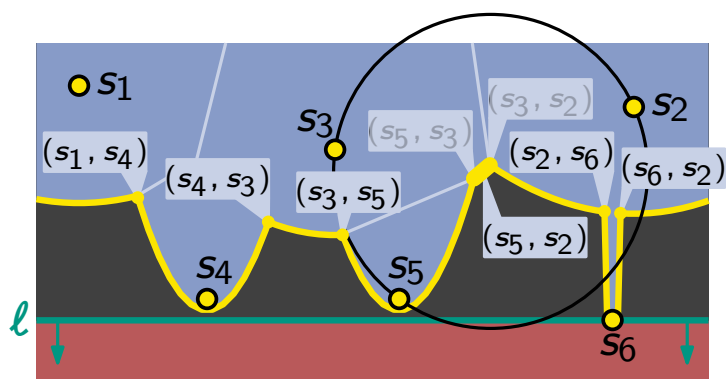
Standort-Event

- finde x -Koordinate des neuen Standorts in der Menge der Schnittpunkte
- füge zwei neue Schnittpunkte ein

Kreis-Event

- gehört zu zwei konsekutiven Schnittpunkten (hier: (s_5, s_3) und (s_3, s_2))
- entferne die beiden und füge entsprechenden neuen ein (hier: (s_5, s_2))

Verwaltung der Beach-Line



Beobachtung

- Parabeln können in mehrere Teile zerlegt sein
- es genügt nicht den Standort zu kennen
- speichere stattdessen eine Repräsentation der Schnittpunkte, die von ℓ unabhängig ist

Standort-Event

- finde x -Koordinate des neuen Standorts in der Menge der Schnittpunkte
- füge zwei neue Schnittpunkte ein

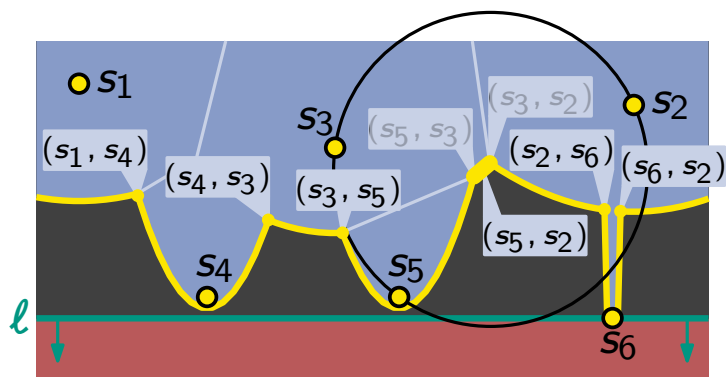
Kreis-Event

- gehört zu zwei konsekutiven Schnittpunkten (hier: (s_5, s_3) und (s_3, s_2))
- entferne die beiden und füge entsprechenden neuen ein (hier: (s_5, s_2))

Datenstruktur

- wir wollen suchen (bezüglich x), einfügen und löschen

Verwaltung der Beach-Line



Beobachtung

- Parabeln können in mehrere Teile zerlegt sein
- es genügt nicht den Standort zu kennen
- speichere stattdessen eine Repräsentation der Schnittpunkte, die von ℓ unabhängig ist

Standort-Event

- finde x -Koordinate des neuen Standorts in der Menge der Schnittpunkte
- füge zwei neue Schnittpunkte ein

Kreis-Event

- gehört zu zwei konsekutiven Schnittpunkten (hier: (s_5, s_3) und (s_3, s_2))
- entferne die beiden und füge entsprechenden neuen ein (hier: (s_5, s_2))

Datenstruktur

- wir wollen suchen (bezüglich x), einfügen und löschen
- Binärer Suchbaum bietet sich an ($O(\log n)$ pro Operation)

Verwaltung der Events

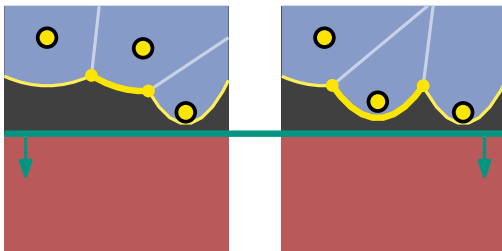
Beobachtung

- Standort-Events kennen wir alle zu Beginn
- Kreis-Events: gegeben durch benachbarte Schnittpunkten in der BL

Verwaltung der Events

Beobachtung

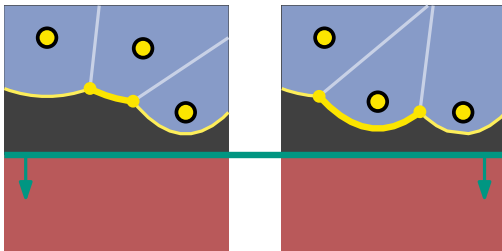
- Standort-Events kennen wir alle zu Beginn
- Kreis-Events: gegeben durch benachbarte Schnittpunkten in der BL
- nicht jedes Paar benachbarter Schnittpunkte führt zu einem Kreis-Event



Verwaltung der Events

Beobachtung

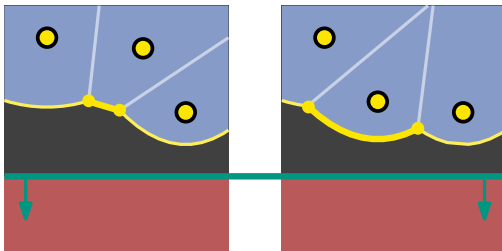
- Standort-Events kennen wir alle zu Beginn
- Kreis-Events: gegeben durch benachbarte Schnittpunkten in der BL
- nicht jedes Paar benachbarter Schnittpunkte führt zu einem Kreis-Event



Verwaltung der Events

Beobachtung

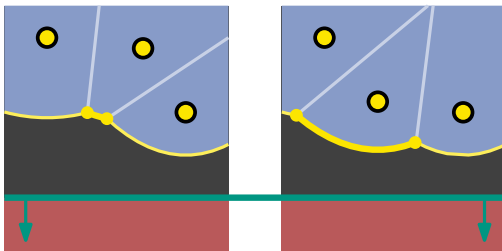
- Standort-Events kennen wir alle zu Beginn
- Kreis-Events: gegeben durch benachbarte Schnittpunkten in der BL
- nicht jedes Paar benachbarter Schnittpunkte führt zu einem Kreis-Event



Verwaltung der Events

Beobachtung

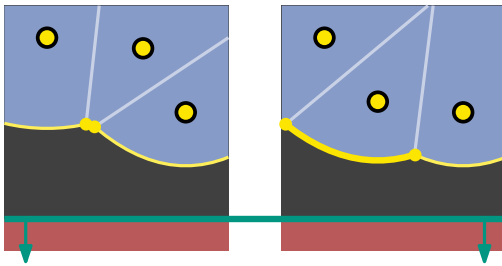
- Standort-Events kennen wir alle zu Beginn
- Kreis-Events: gegeben durch benachbarte Schnittpunkten in der BL
- nicht jedes Paar benachbarter Schnittpunkte führt zu einem Kreis-Event



Verwaltung der Events

Beobachtung

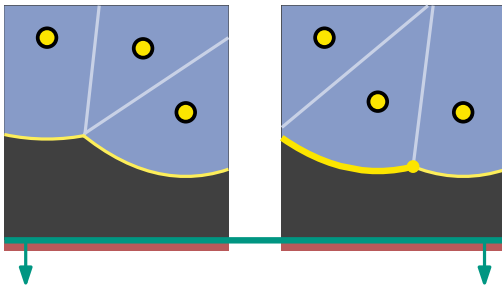
- Standort-Events kennen wir alle zu Beginn
- Kreis-Events: gegeben durch benachbarte Schnittpunkten in der BL
- nicht jedes Paar benachbarter Schnittpunkte führt zu einem Kreis-Event



Verwaltung der Events

Beobachtung

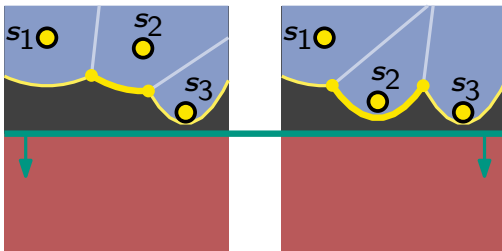
- Standort-Events kennen wir alle zu Beginn
- Kreis-Events: gegeben durch benachbarte Schnittpunkten in der BL
- nicht jedes Paar benachbarter Schnittpunkte führt zu einem Kreis-Event



Verwaltung der Events

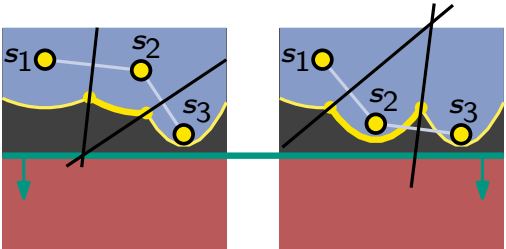
Beobachtung

- Standort-Events kennen wir alle zu Beginn
- Kreis-Events: gegeben durch benachbarte Schnittpunkten in der BL
- nicht jedes Paar benachbarter Schnittpunkte führt zu einem Kreis-Event
- ob wir ein Kreis-Event haben kann an den drei zugehörigen Punkten abgelesen werden



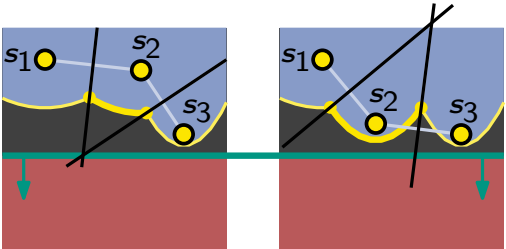
Verwaltung der Events

Beobachtung

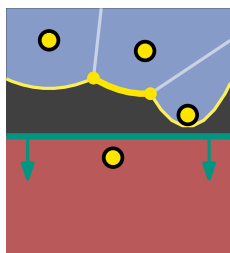
- Standort-Events kennen wir alle zu Beginn
 - Kreis-Events: gegeben durch benachbarte Schnittpunkten in der BL
 - nicht jedes Paar benachbarter Schnittpunkte führt zu einem Kreis-Event
- 
- ob wir ein Kreis-Event haben kann an den drei zugehörigen Punkten abgelesen werden
 - Mittelsenkrechten konvergieren (divergieren) → Parabelstück schrumpft (wächst) → (kein) Kreisevent

Verwaltung der Events

Beobachtung

- Standort-Events kennen wir alle zu Beginn
 - Kreis-Events: gegeben durch benachbarte Schnittpunkten in der BL
 - nicht jedes Paar benachbarter Schnittpunkte führt zu einem Kreis-Event
- 
- ob wir ein Kreis-Event haben kann an den drei zugehörigen Punkten abgelesen werden
 - Mittelsenkrechten konvergieren (divergieren) → Parabelstück schrumpft (wächst) → (kein) Kreisevent

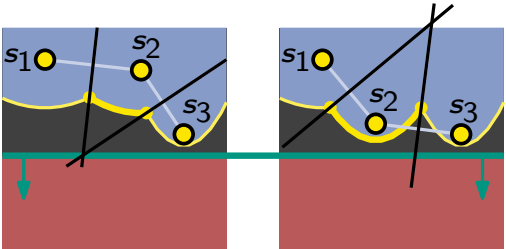
Falscher Alarm



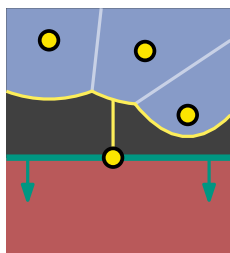
- ggf. findet ein Kreis-Event gar nicht statt

Verwaltung der Events

Beobachtung

- Standort-Events kennen wir alle zu Beginn
 - Kreis-Events: gegeben durch benachbarte Schnittpunkten in der BL
 - nicht jedes Paar benachbarter Schnittpunkte führt zu einem Kreis-Event
- 
- ob wir ein Kreis-Event haben kann an den drei zugehörigen Punkten abgelesen werden
 - Mittelsenkrechten konvergieren (divergieren) → Parabelstück schrumpft (wächst) → (kein) Kreisevent

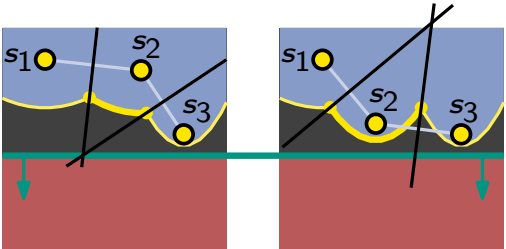
Falscher Alarm



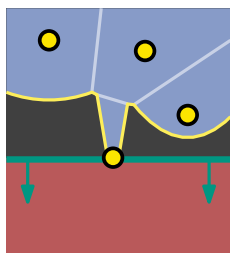
- ggf. findet ein Kreis-Event gar nicht statt

Verwaltung der Events

Beobachtung

- Standort-Events kennen wir alle zu Beginn
 - Kreis-Events: gegeben durch benachbarte Schnittpunkten in der BL
 - nicht jedes Paar benachbarter Schnittpunkte führt zu einem Kreis-Event
- 
- ob wir ein Kreis-Event haben kann an den drei zugehörigen Punkten abgelesen werden
 - Mittelsenkrechten konvergieren (divergieren) → Parabelstück schrumpft (wächst) → (kein) Kreisevent

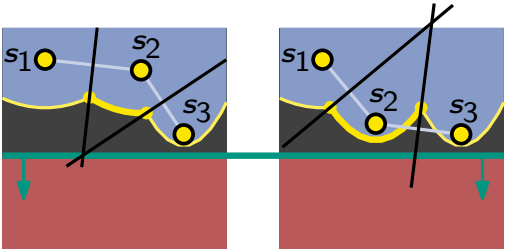
Falscher Alarm



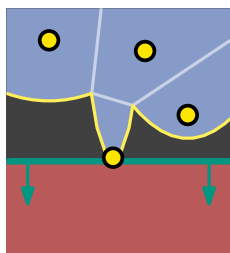
- ggf. findet ein Kreis-Event gar nicht statt

Verwaltung der Events

Beobachtung

- Standort-Events kennen wir alle zu Beginn
 - Kreis-Events: gegeben durch benachbarte Schnittpunkten in der BL
 - nicht jedes Paar benachbarter Schnittpunkte führt zu einem Kreis-Event
- 
- ob wir ein Kreis-Event haben kann an den drei zugehörigen Punkten abgelesen werden
 - Mittelsenkrechten konvergieren (divergieren) → Parabelstück schrumpft (wächst) → (kein) Kreisevent

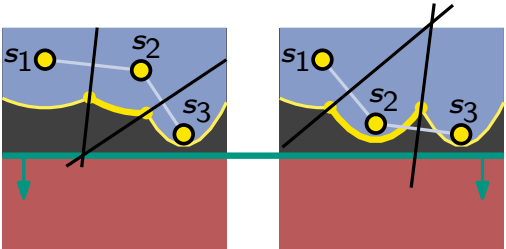
Falscher Alarm



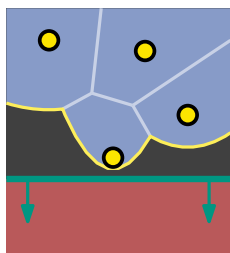
- ggf. findet ein Kreis-Event gar nicht statt

Verwaltung der Events

Beobachtung

- Standort-Events kennen wir alle zu Beginn
 - Kreis-Events: gegeben durch benachbarte Schnittpunkten in der BL
 - nicht jedes Paar benachbarter Schnittpunkte führt zu einem Kreis-Event
- 
- ob wir ein Kreis-Event haben kann an den drei zugehörigen Punkten abgelesen werden
 - Mittelsenkrechten konvergieren (divergieren) → Parabelstück schrumpft (wächst) → (kein) Kreisevent

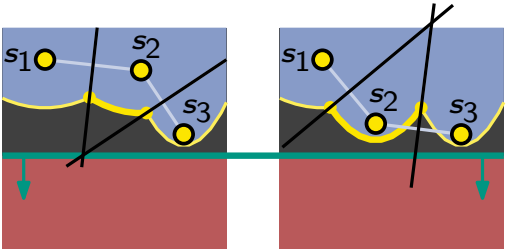
Falscher Alarm



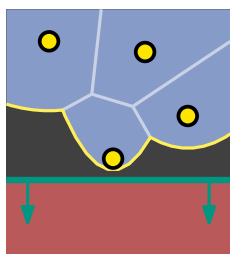
- ggf. findet ein Kreis-Event gar nicht statt

Verwaltung der Events

Beobachtung

- Standort-Events kennen wir alle zu Beginn
 - Kreis-Events: gegeben durch benachbarte Schnittpunkten in der BL
 - nicht jedes Paar benachbarter Schnittpunkte führt zu einem Kreis-Event
- 
- ob wir ein Kreis-Event haben kann an den drei zugehörigen Punkten abgelesen werden
 - Mittelsenkrechten konvergieren (divergieren) → Parabelstück schrumpft (wächst) → (kein) Kreisevent

Falscher Alarm

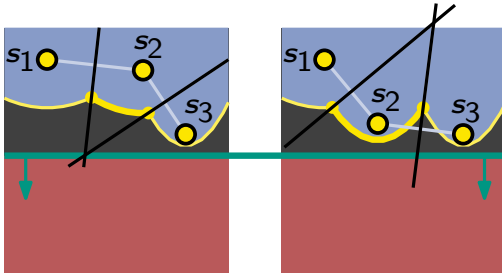


- ggf. findet ein Kreis-Event gar nicht statt
- bei jedem Standort-Event müssen wir das Kreis-Event für die aufgespaltene Parabel entfernen (falls es existiert)

Verwaltung der Events

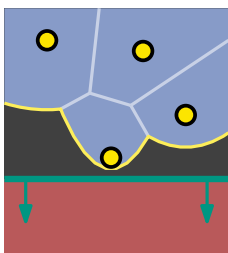
Beobachtung

- Standort-Events kennen wir alle zu Beginn
- Kreis-Events: gegeben durch benachbarte Schnittpunkten in der BL
- nicht jedes Paar benachbarter Schnittpunkte führt zu einem Kreis-Event



- ob wir ein Kreis-Event haben kann an den drei zugehörigen Punkten abgelesen werden
- Mittelsenkrechten konvergieren (divergieren) → Parabelstück schrumpft (wächst) → (kein) Kreisevent

Falscher Alarm



- ggf. findet ein Kreis-Event gar nicht statt
- bei jedem Standort-Event müssen wir das Kreis-Event für die aufgespaltene Parabel entfernen (falls es existiert)

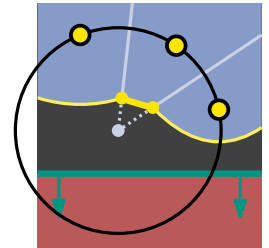
Alles kein Problem

- Benutze binären Suchbaum für die Queue
- jede der Operationen braucht $O(\log n)$ Zeit

Konstruktion des Voronoi-Diagramms

Schon gesehen

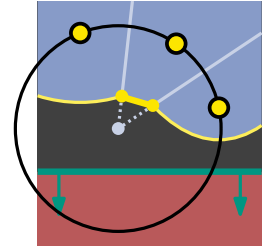
- Punkt p ist Knoten $\Leftrightarrow p$ ist Kreismittelpunkt eines Kreis-Events
- p liegt auf einer Kante $\Leftrightarrow p$ ist irgendwann Schnittpunkt von Parabelstücken



Konstruktion des Voronoi-Diagramms

Schon gesehen

- Punkt p ist Knoten $\Leftrightarrow p$ ist Kreismittelpunkt eines Kreis-Events
- p liegt auf einer Kante $\Leftrightarrow p$ ist irgendwann Schnittpunkt von Parabelstücken



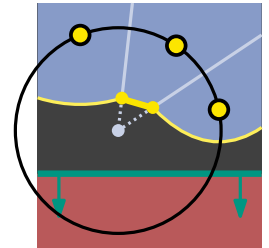
Plan

- berechne doppelt verkettete Kantenliste von $\text{Vor}(S)$ beim Sweepen
(unendlich lange Kanten muss man irgendwie handhaben; z.B. mit einer Bounding-Box)

Konstruktion des Voronoi-Diagramms

Schon gesehen

- Punkt p ist Knoten $\Leftrightarrow p$ ist Kreismittelpunkt eines Kreis-Events
- p liegt auf einer Kante $\Leftrightarrow p$ ist irgendwann Schnittpunkt von Parabelstücken



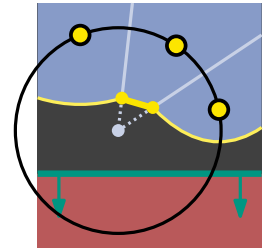
Plan

- berechne doppelt verkettete Kantenliste von $\text{Vor}(S)$ beim Sweepen
(unendlich lange Kanten muss man irgendwie handhaben; z.B. mit einer Bounding-Box)
- Standort-Event: neue Parabel mit Schnittpunkten \rightarrow neue Halbkanten

Konstruktion des Voronoi-Diagramms

Schon gesehen

- Punkt p ist Knoten $\Leftrightarrow p$ ist Kreismittelpunkt eines Kreis-Events
- p liegt auf einer Kante $\Leftrightarrow p$ ist irgendwann Schnittpunkt von Parabelstücken



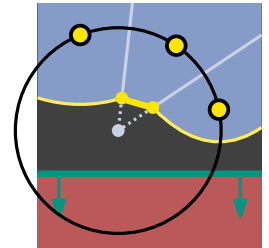
Plan

- berechne doppelt verkettete Kantenliste von $\text{Vor}(S)$ beim Sweepen
(unendlich lange Kanten muss man irgendwie handhaben; z.B. mit einer Bounding-Box)
- Standort-Event: neue Parabel mit Schnittpunkten \rightarrow neue Halbkanten
- Kreis-Event: neuer Knoten \rightarrow hänge Halbkanten an neuen Knoten
neuer Schnittpunkt \rightarrow neue Halbkante

Konstruktion des Voronoi-Diagramms

Schon gesehen

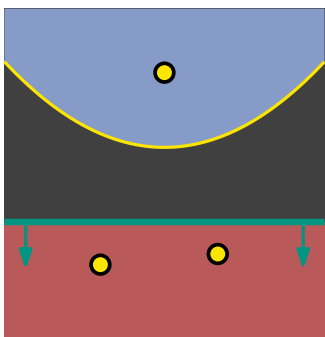
- Punkt p ist Knoten $\Leftrightarrow p$ ist Kreismittelpunkt eines Kreis-Events
- p liegt auf einer Kante $\Leftrightarrow p$ ist irgendwann Schnittpunkt von Parabelstücken



Plan

- berechne doppelt verkettete Kantenliste von $\text{Vor}(S)$ beim Sweepen
(unendlich lange Kanten muss man irgendwie handhaben; z.B. mit einer Bounding-Box)
- Standort-Event: neue Parabel mit Schnittpunkten \rightarrow neue Halbkanten
- Kreis-Event: neuer Knoten \rightarrow hänge Halbkanten an neuen Knoten
neuer Schnittpunkt \rightarrow neue Halbkante

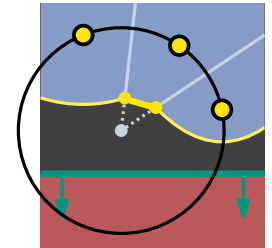
Beispiel



Konstruktion des Voronoi-Diagramms

Schon gesehen

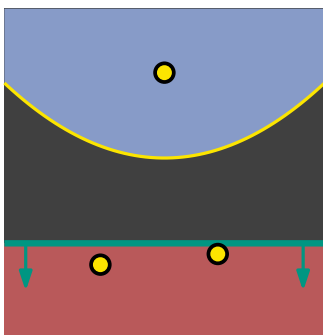
- Punkt p ist Knoten $\Leftrightarrow p$ ist Kreismittelpunkt eines Kreis-Events
- p liegt auf einer Kante $\Leftrightarrow p$ ist irgendwann Schnittpunkt von Parabelstücken



Plan

- berechne doppelt verkettete Kantenliste von $\text{Vor}(S)$ beim Sweepen
(unendlich lange Kanten muss man irgendwie handhaben; z.B. mit einer Bounding-Box)
- Standort-Event: neue Parabel mit Schnittpunkten \rightarrow neue Halbkanten
- Kreis-Event: neuer Knoten \rightarrow hänge Halbkanten an neuen Knoten
neuer Schnittpunkt \rightarrow neue Halbkante

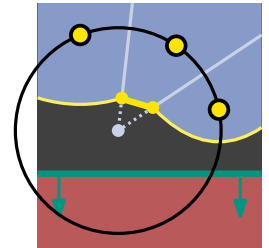
Beispiel



Konstruktion des Voronoi-Diagramms

Schon gesehen

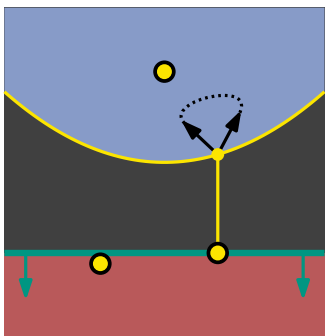
- Punkt p ist Knoten $\Leftrightarrow p$ ist Kreismittelpunkt eines Kreis-Events
- p liegt auf einer Kante $\Leftrightarrow p$ ist irgendwann Schnittpunkt von Parabelstücken



Plan

- berechne doppelt verkettete Kantenliste von $\text{Vor}(S)$ beim Sweepen
(unendlich lange Kanten muss man irgendwie handhaben; z.B. mit einer Bounding-Box)
- Standort-Event: neue Parabel mit Schnittpunkten \rightarrow neue Halbkanten
- Kreis-Event: neuer Knoten \rightarrow hänge Halbkanten an neuen Knoten
neuer Schnittpunkt \rightarrow neue Halbkante

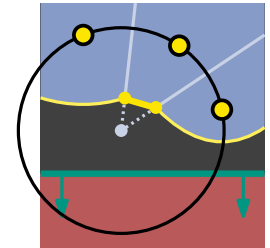
Beispiel



Konstruktion des Voronoi-Diagramms

Schon gesehen

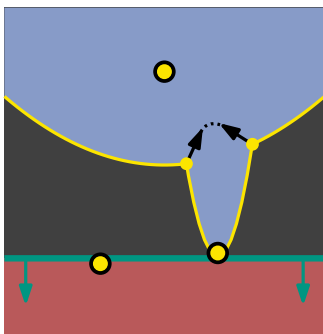
- Punkt p ist Knoten $\Leftrightarrow p$ ist Kreismittelpunkt eines Kreis-Events
- p liegt auf einer Kante $\Leftrightarrow p$ ist irgendwann Schnittpunkt von Parabelstücken



Plan

- berechne doppelt verkettete Kantenliste von $\text{Vor}(S)$ beim Sweepen
(unendlich lange Kanten muss man irgendwie handhaben; z.B. mit einer Bounding-Box)
- Standort-Event: neue Parabel mit Schnittpunkten \rightarrow neue Halbkanten
- Kreis-Event: neuer Knoten \rightarrow hänge Halbkanten an neuen Knoten
neuer Schnittpunkt \rightarrow neue Halbkante

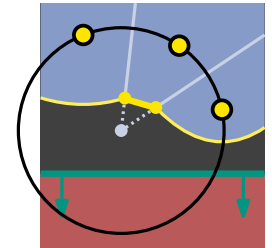
Beispiel



Konstruktion des Voronoi-Diagramms

Schon gesehen

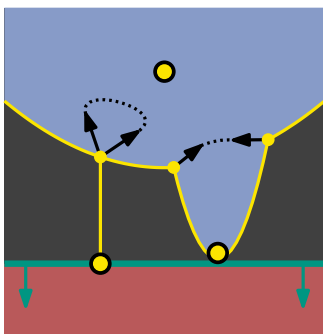
- Punkt p ist Knoten $\Leftrightarrow p$ ist Kreismittelpunkt eines Kreis-Events
- p liegt auf einer Kante $\Leftrightarrow p$ ist irgendwann Schnittpunkt von Parabelstücken



Plan

- berechne doppelt verkettete Kantenliste von $\text{Vor}(S)$ beim Sweepen
(unendlich lange Kanten muss man irgendwie handhaben; z.B. mit einer Bounding-Box)
- Standort-Event: neue Parabel mit Schnittpunkten \rightarrow neue Halbkanten
- Kreis-Event: neuer Knoten \rightarrow hänge Halbkanten an neuen Knoten
neuer Schnittpunkt \rightarrow neue Halbkante

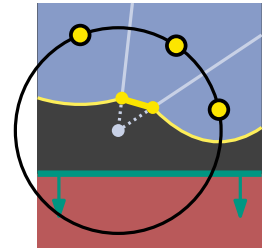
Beispiel



Konstruktion des Voronoi-Diagramms

Schon gesehen

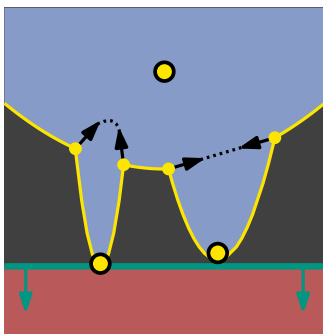
- Punkt p ist Knoten $\Leftrightarrow p$ ist Kreismittelpunkt eines Kreis-Events
- p liegt auf einer Kante $\Leftrightarrow p$ ist irgendwann Schnittpunkt von Parabelstücken



Plan

- berechne doppelt verkettete Kantenliste von $\text{Vor}(S)$ beim Sweepen
(unendlich lange Kanten muss man irgendwie handhaben; z.B. mit einer Bounding-Box)
- Standort-Event: neue Parabel mit Schnittpunkten \rightarrow neue Halbkanten
- Kreis-Event: neuer Knoten \rightarrow hänge Halbkanten an neuen Knoten
neuer Schnittpunkt \rightarrow neue Halbkante

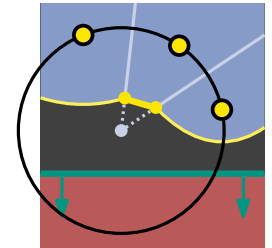
Beispiel



Konstruktion des Voronoi-Diagramms

Schon gesehen

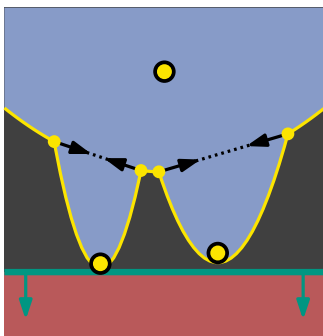
- Punkt p ist Knoten $\Leftrightarrow p$ ist Kreismittelpunkt eines Kreis-Events
- p liegt auf einer Kante $\Leftrightarrow p$ ist irgendwann Schnittpunkt von Parabelstücken



Plan

- berechne doppelt verkettete Kantenliste von $\text{Vor}(S)$ beim Sweepen
(unendlich lange Kanten muss man irgendwie handhaben; z.B. mit einer Bounding-Box)
- Standort-Event: neue Parabel mit Schnittpunkten \rightarrow neue Halbkanten
- Kreis-Event: neuer Knoten \rightarrow hänge Halbkanten an neuen Knoten
neuer Schnittpunkt \rightarrow neue Halbkante

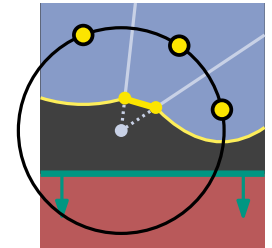
Beispiel



Konstruktion des Voronoi-Diagramms

Schon gesehen

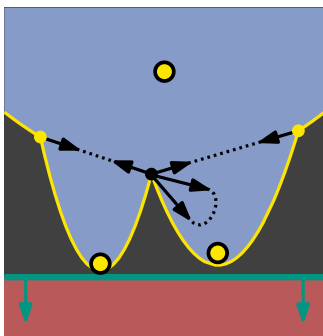
- Punkt p ist Knoten $\Leftrightarrow p$ ist Kreismittelpunkt eines Kreis-Events
- p liegt auf einer Kante $\Leftrightarrow p$ ist irgendwann Schnittpunkt von Parabelstücken



Plan

- berechne doppelt verkettete Kantenliste von $\text{Vor}(S)$ beim Sweepen
(unendlich lange Kanten muss man irgendwie handhaben; z.B. mit einer Bounding-Box)
- Standort-Event: neue Parabel mit Schnittpunkten \rightarrow neue Halbkanten
- Kreis-Event: neuer Knoten \rightarrow hänge Halbkanten an neuen Knoten
neuer Schnittpunkt \rightarrow neue Halbkante

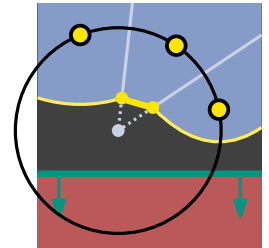
Beispiel



Konstruktion des Voronoi-Diagramms

Schon gesehen

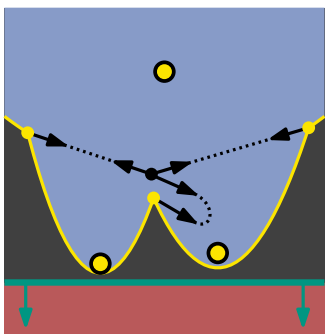
- Punkt p ist Knoten $\Leftrightarrow p$ ist Kreismittelpunkt eines Kreis-Events
- p liegt auf einer Kante $\Leftrightarrow p$ ist irgendwann Schnittpunkt von Parabelstücken



Plan

- berechne doppelt verkettete Kantenliste von $\text{Vor}(S)$ beim Sweepen
(unendlich lange Kanten muss man irgendwie handhaben; z.B. mit einer Bounding-Box)
- Standort-Event: neue Parabel mit Schnittpunkten \rightarrow neue Halbkanten
- Kreis-Event: neuer Knoten \rightarrow hänge Halbkanten an neuen Knoten
neuer Schnittpunkt \rightarrow neue Halbkante

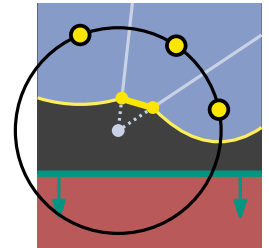
Beispiel



Konstruktion des Voronoi-Diagramms

Schon gesehen

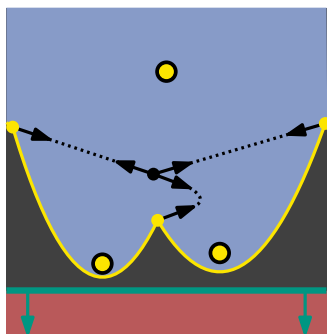
- Punkt p ist Knoten $\Leftrightarrow p$ ist Kreismittelpunkt eines Kreis-Events
- p liegt auf einer Kante $\Leftrightarrow p$ ist irgendwann Schnittpunkt von Parabelstücken



Plan

- berechne doppelt verkettete Kantenliste von $\text{Vor}(S)$ beim Sweepen
(unendlich lange Kanten muss man irgendwie handhaben; z.B. mit einer Bounding-Box)
- Standort-Event: neue Parabel mit Schnittpunkten \rightarrow neue Halbkanten
- Kreis-Event: neuer Knoten \rightarrow hänge Halbkanten an neuen Knoten
neuer Schnittpunkt \rightarrow neue Halbkante

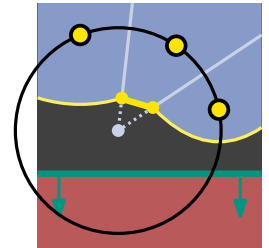
Beispiel



Konstruktion des Voronoi-Diagramms

Schon gesehen

- Punkt p ist Knoten $\Leftrightarrow p$ ist Kreismittelpunkt eines Kreis-Events
- p liegt auf einer Kante $\Leftrightarrow p$ ist irgendwann Schnittpunkt von Parabelstücken

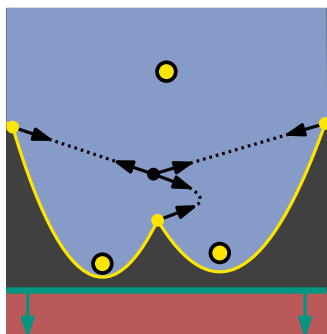


Plan

- berechne doppelt verkettete Kantenliste von $\text{Vor}(S)$ beim Sweepen
(unendlich lange Kanten muss man irgendwie handhaben; z.B. mit einer Bounding-Box)
- Standort-Event: neue Parabel mit Schnittpunkten \rightarrow neue Halbkanten
- Kreis-Event: neuer Knoten \rightarrow hänge Halbkanten an neuen Knoten
neuer Schnittpunkt \rightarrow neue Halbkante

Beispiel

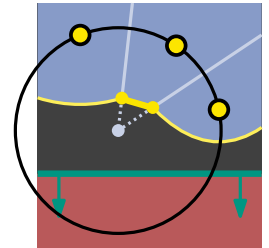
Unendlich lange Kanten



Konstruktion des Voronoi-Diagramms

Schon gesehen

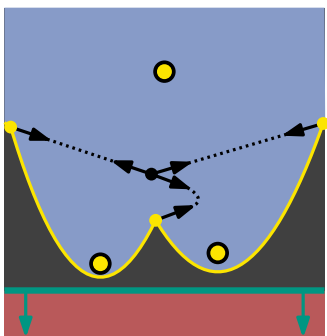
- Punkt p ist Knoten $\Leftrightarrow p$ ist Kreismittelpunkt eines Kreis-Events
- p liegt auf einer Kante $\Leftrightarrow p$ ist irgendwann Schnittpunkt von Parabelstücken



Plan

- berechne doppelt verkettete Kantenliste von $\text{Vor}(S)$ beim Sweepen
(unendlich lange Kanten muss man irgendwie handhaben; z.B. mit einer Bounding-Box)
- Standort-Event: neue Parabel mit Schnittpunkten \rightarrow neue Halbkanten
- Kreis-Event: neuer Knoten \rightarrow hänge Halbkanten an neuen Knoten
neuer Schnittpunkt \rightarrow neue Halbkante

Beispiel



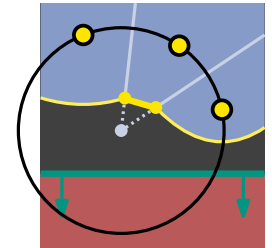
Unendlich lange Kanten

- manche Halbkanten werden nicht fertig, da ihr Parabelstück nicht verschwindet

Konstruktion des Voronoi-Diagramms

Schon gesehen

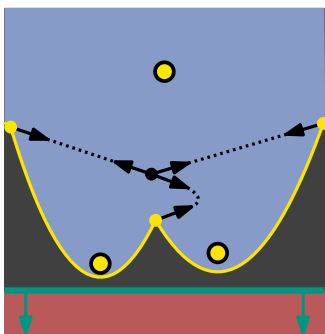
- Punkt p ist Knoten $\Leftrightarrow p$ ist Kreismittelpunkt eines Kreis-Events
- p liegt auf einer Kante $\Leftrightarrow p$ ist irgendwann Schnittpunkt von Parabelstücken



Plan

- berechne doppelt verkettete Kantenliste von $\text{Vor}(S)$ beim Sweepen
(unendlich lange Kanten muss man irgendwie handhaben; z.B. mit einer Bounding-Box)
- Standort-Event: neue Parabel mit Schnittpunkten \rightarrow neue Halbkanten
- Kreis-Event: neuer Knoten \rightarrow hänge Halbkanten an neuen Knoten
neuer Schnittpunkt \rightarrow neue Halbkante

Beispiel

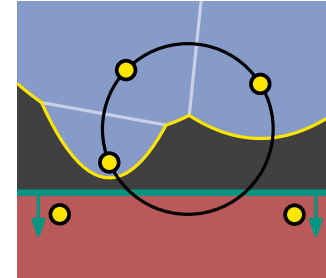


Unendlich lange Kanten

- manche Halbkanten werden nicht fertig, da ihr Parabelstück nicht verschwindet
- bei leerer Queue muss man die Schnittpunkte in der Beach-Line im Nachgang noch aufräumen
(abhängig vom Umgang mit unendlichen Kanten in der doppelt-verketteten Kantenliste)

Sonderfälle

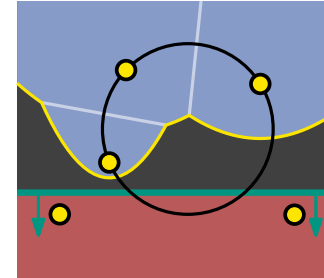
Events mit gleicher y -Koordinate



Sonderfälle

Events mit gleicher y -Koordinate

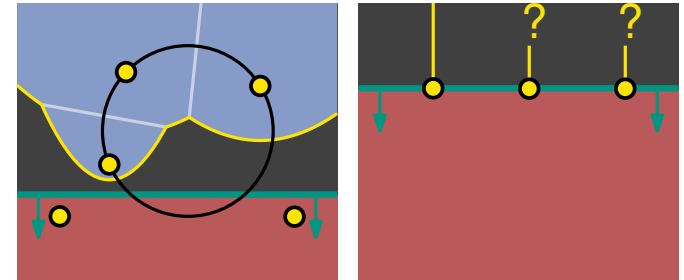
- behandle in beliebiger Reihenfolge



Sonderfälle

Events mit gleicher y -Koordinate

- behandle in beliebiger Reihenfolge
- Ausnahme: initiale Standort-Events
(noch keine Parabel über sich)

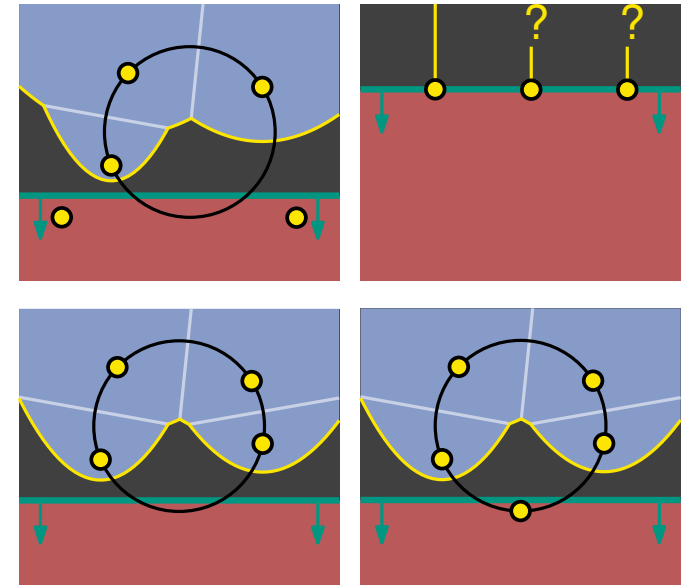


Sonderfälle

Events mit gleicher y -Koordinate

- behandle in beliebiger Reihenfolge
- Ausnahme: initiale Standort-Events
(noch keine Parabel über sich)

Gleiche x - und y -Koordinate



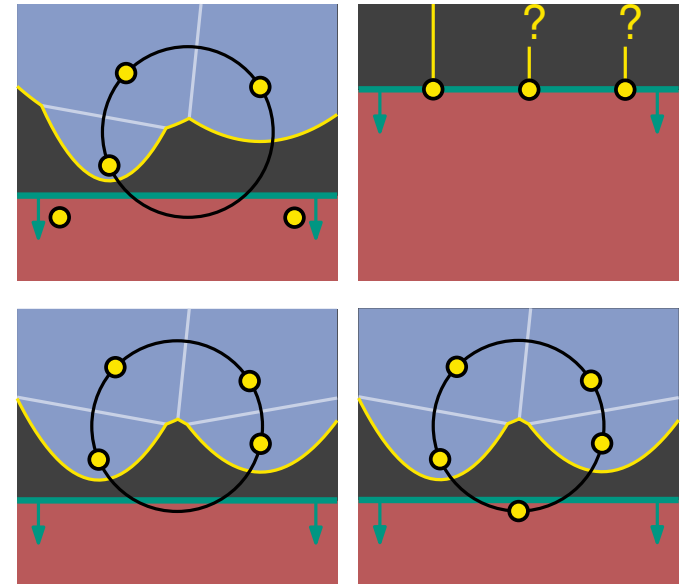
Sonderfälle

Events mit gleicher y -Koordinate

- behandle in beliebiger Reihenfolge
- Ausnahme: initiale Standort-Events
(noch keine Parabel über sich)

Gleiche x - und y -Koordinate

- Knoten in $\text{Vor}(S)$ mit $\text{Grad} > 3$



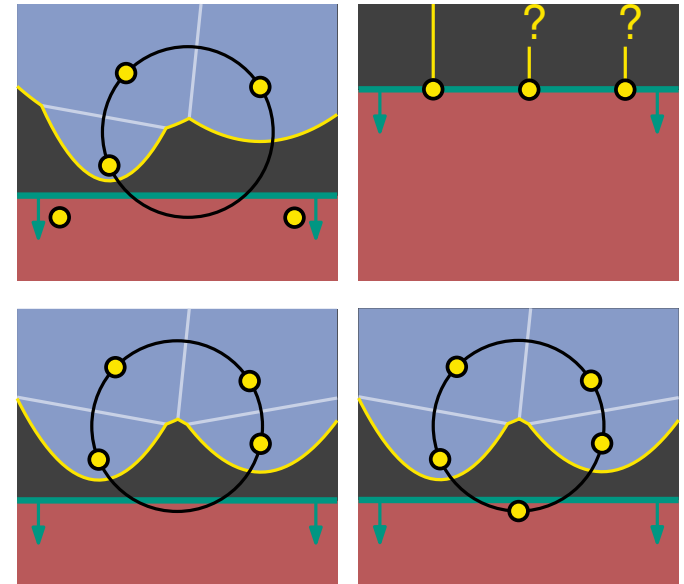
Sonderfälle

Events mit gleicher y -Koordinate

- behandle in beliebiger Reihenfolge
- Ausnahme: initiale Standort-Events
(noch keine Parabel über sich)

Gleiche x - und y -Koordinate

- Knoten in $\text{Vor}(S)$ mit $\text{Grad} > 3$
- Sonderbehandlung \rightarrow Knoten mit $\text{Grad} > 3$
- keine Sonderbehandlung \rightarrow mehrere Knoten mit $\text{Grad} 3$ und Kanten der Länge 0 dazwischen



Sonderfälle

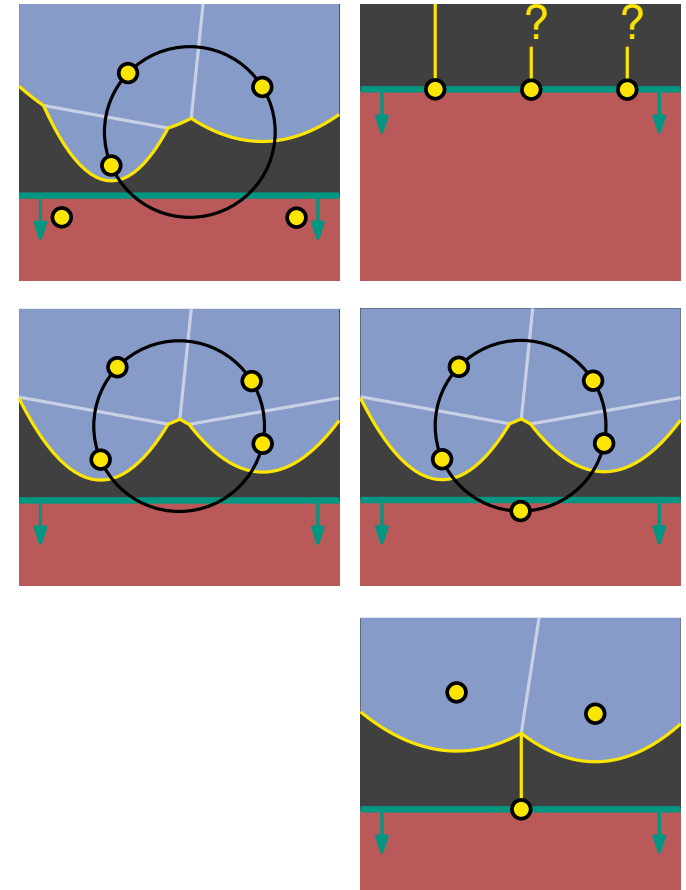
Events mit gleicher y -Koordinate

- behandle in beliebiger Reihenfolge
- Ausnahme: initiale Standort-Events
(noch keine Parabel über sich)

Gleiche x - und y -Koordinate

- Knoten in $\text{Vor}(S)$ mit $\text{Grad} > 3$
- Sonderbehandlung \rightarrow Knoten mit $\text{Grad} > 3$
- keine Sonderbehandlung \rightarrow mehrere Knoten mit $\text{Grad} 3$ und Kanten der Länge 0 dazwischen

Neue Parabel startet bei Schnittpunkt



Sonderfälle

Events mit gleicher y -Koordinate

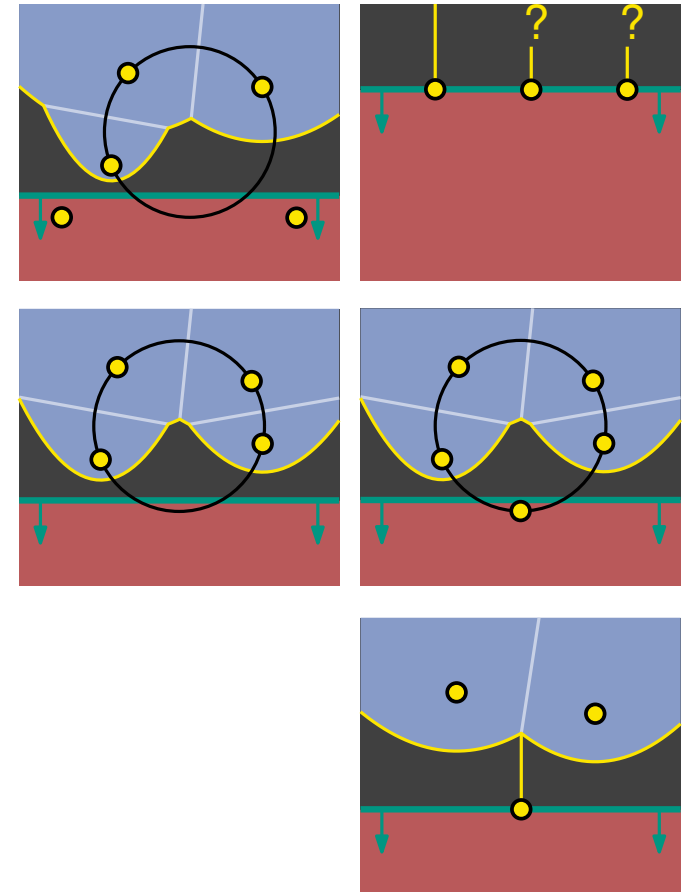
- behandle in beliebiger Reihenfolge
- Ausnahme: initiale Standort-Events
(noch keine Parabel über sich)

Gleiche x - und y -Koordinate

- Knoten in $\text{Vor}(S)$ mit $\text{Grad} > 3$
- Sonderbehandlung \rightarrow Knoten mit $\text{Grad} > 3$
- keine Sonderbehandlung \rightarrow mehrere Knoten mit $\text{Grad} 3$ und Kanten der Länge 0 dazwischen

Neue Parabel startet bei Schnittpunkt

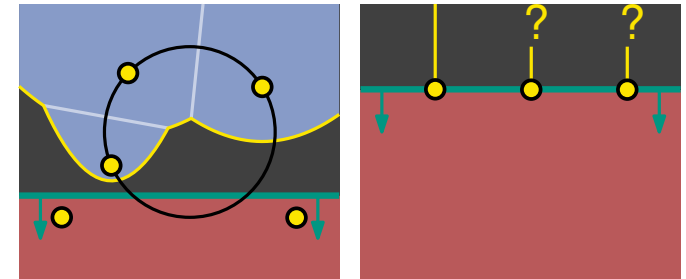
- an der Stelle sollte ein Knoten entstehen



Sonderfälle

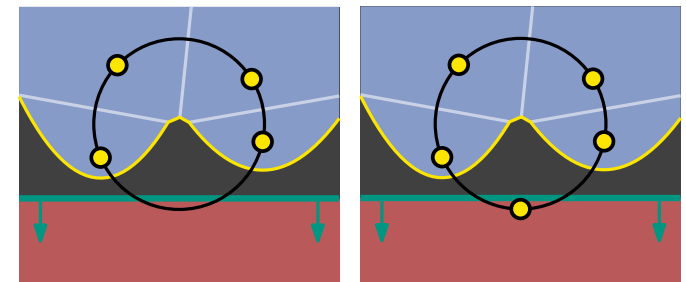
Events mit gleicher y -Koordinate

- behandle in beliebiger Reihenfolge
- Ausnahme: initiale Standort-Events
(noch keine Parabel über sich)



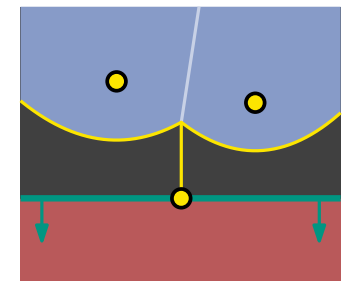
Gleiche x - und y -Koordinate

- Knoten in $\text{Vor}(S)$ mit $\text{Grad} > 3$
- Sonderbehandlung \rightarrow Knoten mit $\text{Grad} > 3$
- keine Sonderbehandlung \rightarrow mehrere Knoten mit $\text{Grad} 3$ und Kanten der Länge 0 dazwischen



Neue Parabel startet bei Schnittpunkt

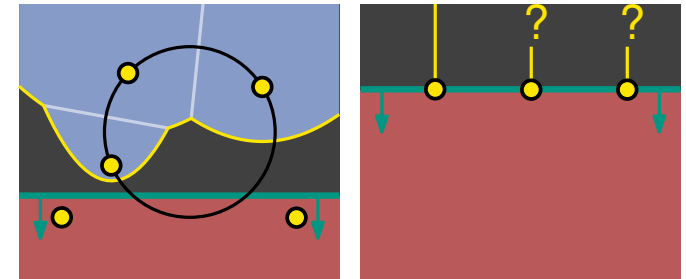
- an der Stelle sollte ein Knoten entstehen
- Sonderbehandlung: erstelle Knoten + Halbkanten
- keine Sonderbehandlung: zerteile eine der beiden Parabeln \rightarrow neues Stück der Länge 0 \rightarrow Kreisevent \rightarrow neuer Knoten



Sonderfälle

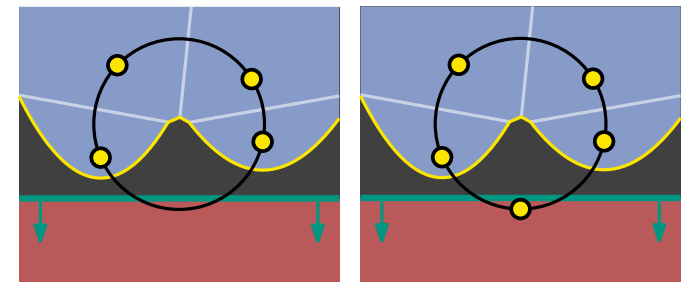
Events mit gleicher y -Koordinate

- behandle in beliebiger Reihenfolge
- Ausnahme: initiale Standort-Events
(noch keine Parabel über sich)



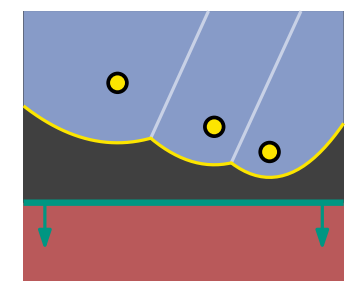
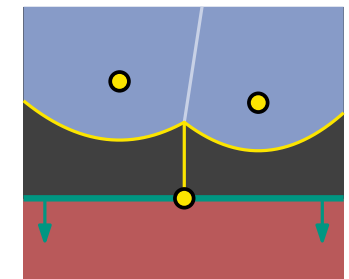
Gleiche x - und y -Koordinate

- Knoten in $\text{Vor}(S)$ mit $\text{Grad} > 3$
- Sonderbehandlung \rightarrow Knoten mit $\text{Grad} > 3$
- keine Sonderbehandlung \rightarrow mehrere Knoten mit $\text{Grad} 3$ und Kanten der Länge 0 dazwischen



Neue Parabel startet bei Schnittpunkt

- an der Stelle sollte ein Knoten entstehen
- Sonderbehandlung: erstelle Knoten + Halbkanten
- keine Sonderbehandlung: zerteile eine der beiden Parabeln \rightarrow neues Stück der Länge 0 \rightarrow Kreisevent \rightarrow neuer Knoten

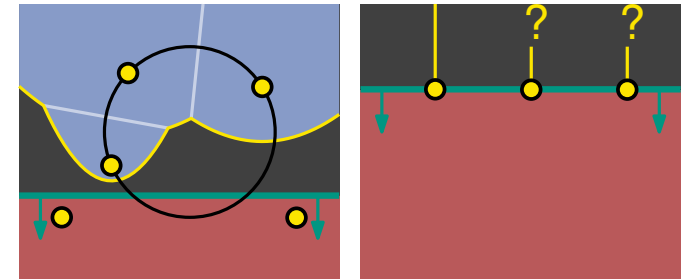


Kollineare Punkte mit konsekutiven Parabelstücken

Sonderfälle

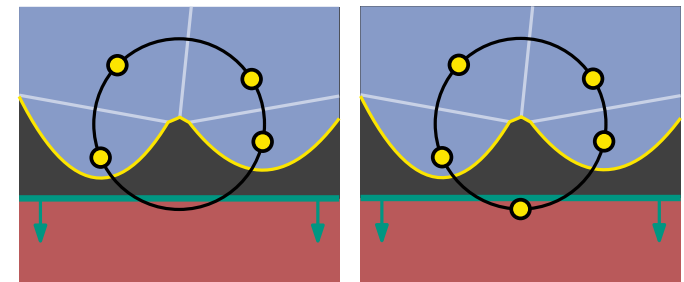
Events mit gleicher y -Koordinate

- behandle in beliebiger Reihenfolge
- Ausnahme: initiale Standort-Events
(noch keine Parabel über sich)



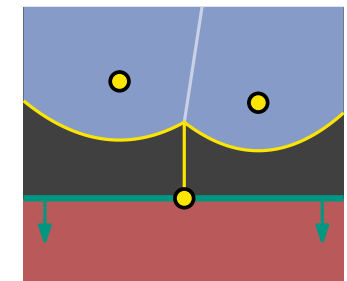
Gleiche x - und y -Koordinate

- Knoten in $\text{Vor}(S)$ mit $\text{Grad} > 3$
- Sonderbehandlung \rightarrow Knoten mit $\text{Grad} > 3$
- keine Sonderbehandlung \rightarrow mehrere Knoten mit $\text{Grad} 3$ und Kanten der Länge 0 dazwischen



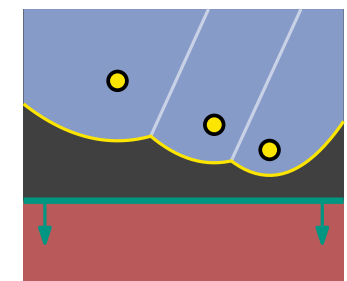
Neue Parabel startet bei Schnittpunkt

- an der Stelle sollte ein Knoten entstehen
- Sonderbehandlung: erstelle Knoten + Halbkanten
- keine Sonderbehandlung: zerteile eine der beiden Parabeln \rightarrow neues Stück der Länge 0 \rightarrow Kreisevent \rightarrow neuer Knoten



Kollineare Punkte mit konsekutiven Parabelstücken

- Vorsicht bei der Berechnung des Kreises



Überblick

Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Überblick

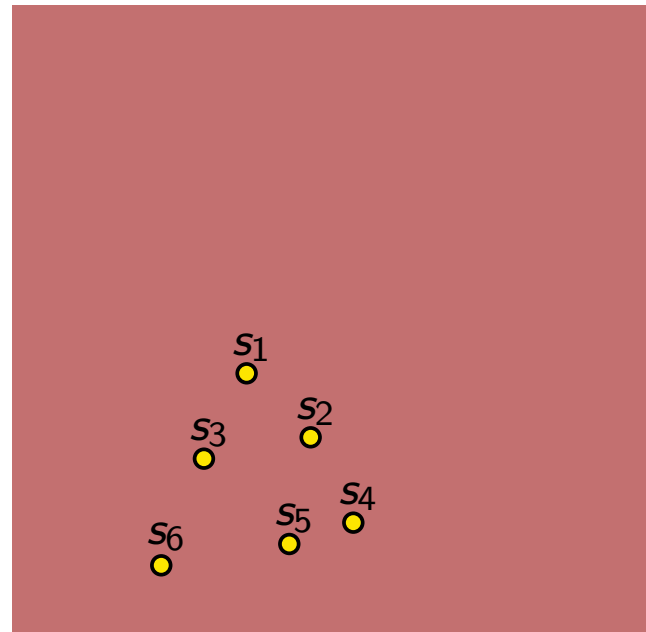
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line:

Überblick

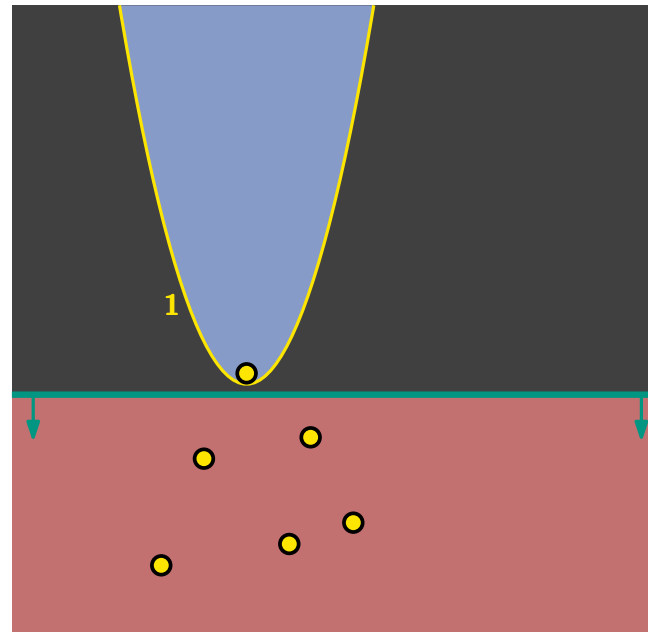
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line:

Überblick

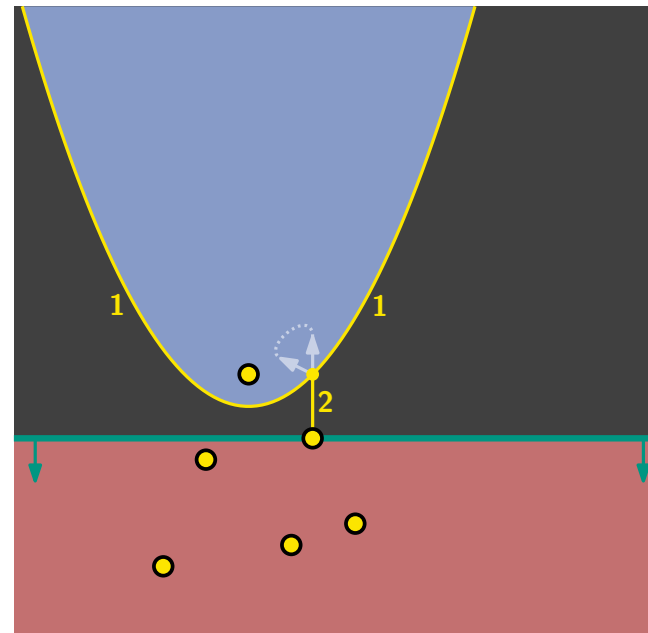
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line: (s_1, s_2) (s_2, s_1)

Überblick

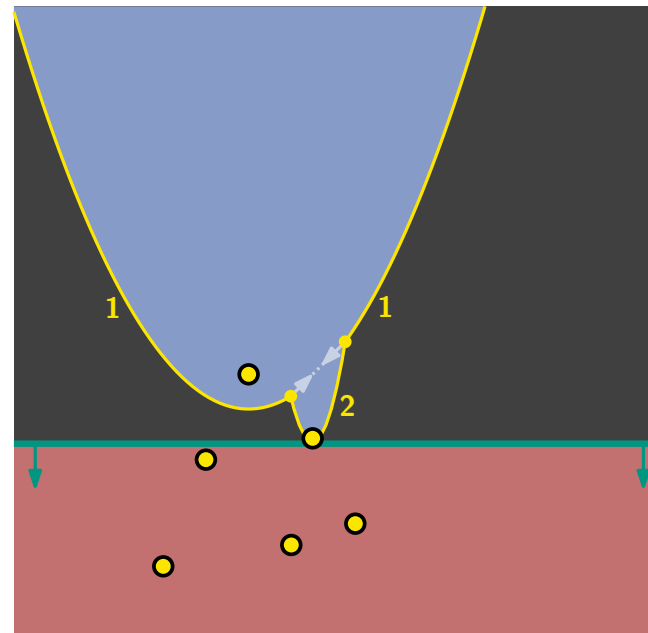
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line: (s_1, s_2) (s_2, s_1)

Überblick

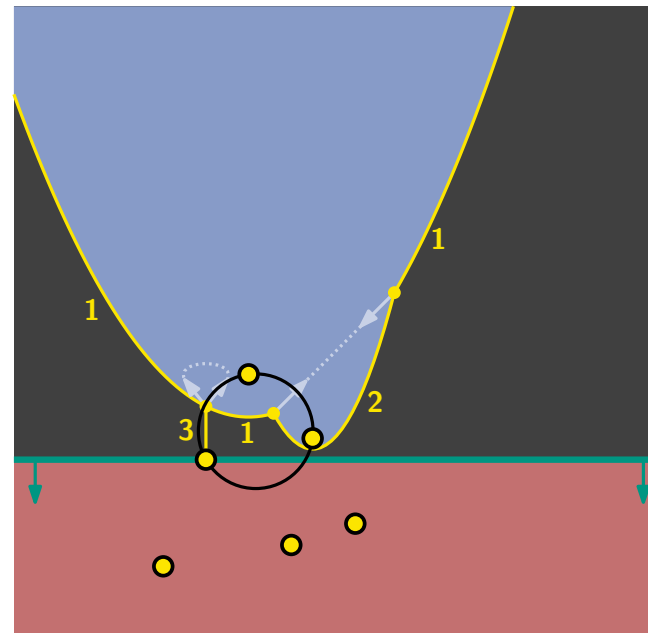
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line: (s_1, s_3) (s_3, s_1) (s_1, s_2) (s_2, s_1)

Überblick

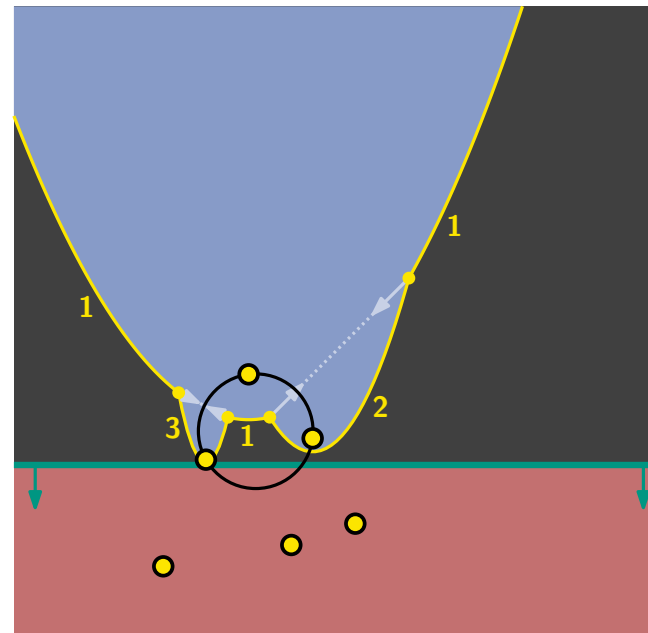
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line: (s_1, s_3) (s_3, s_1) (s_1, s_2) (s_2, s_1)

Überblick

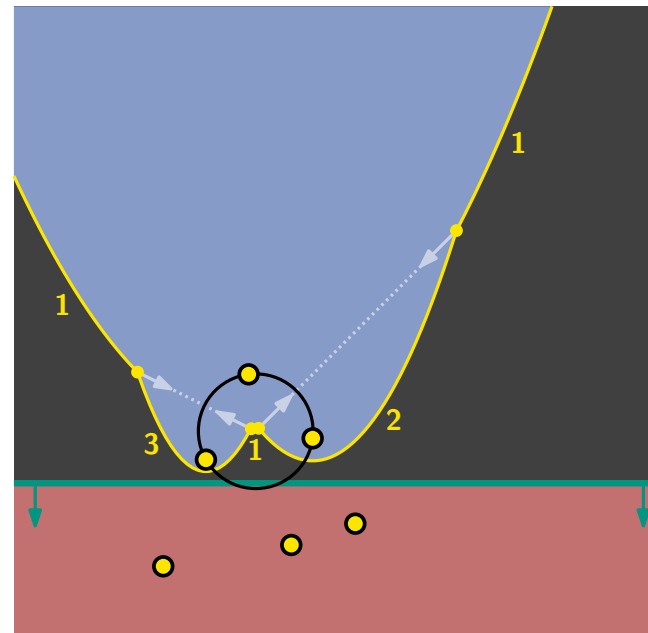
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line: (s_1, s_3) (s_3, s_1) (s_1, s_2) (s_2, s_1)

Überblick

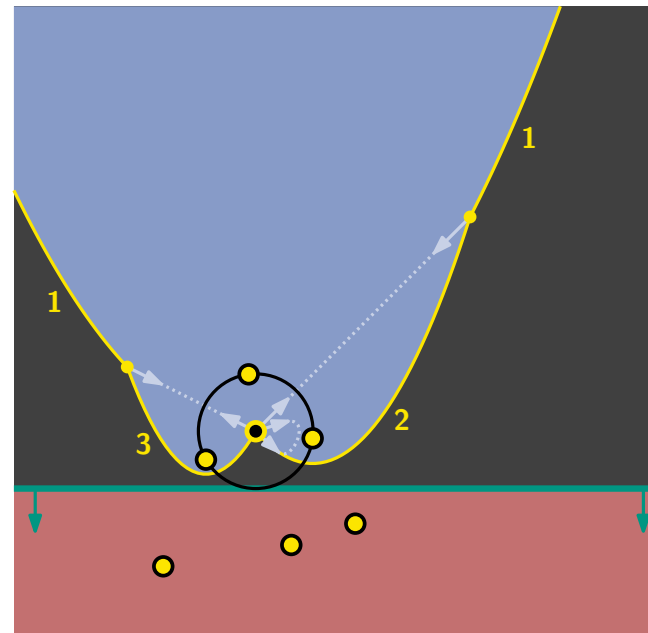
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line: (s_1, s_3) (s_3, s_2) (s_2, s_1)

Überblick

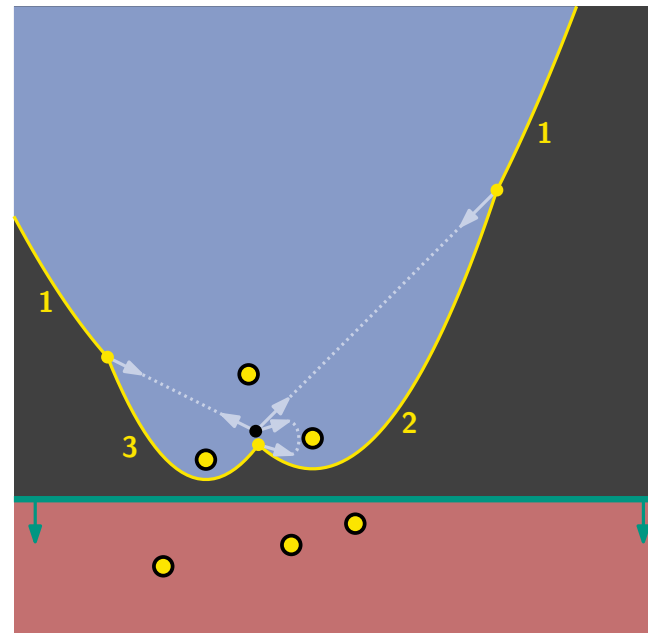
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line: (s_1, s_3) (s_3, s_2) (s_2, s_1)

Überblick

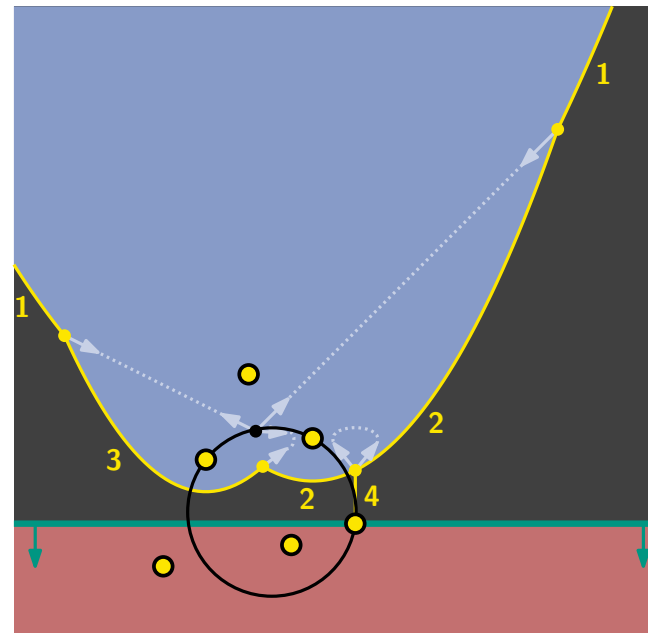
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line: (s_1, s_3) (s_3, s_2) (s_2, s_4) (s_4, s_2) (s_2, s_1)

Überblick

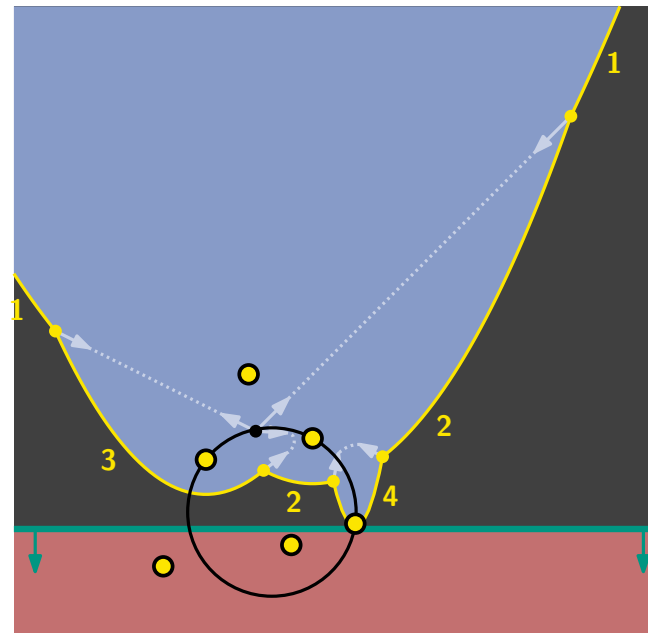
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line: (s_1, s_3) (s_3, s_2) (s_2, s_4) (s_4, s_2) (s_2, s_1)

Überblick

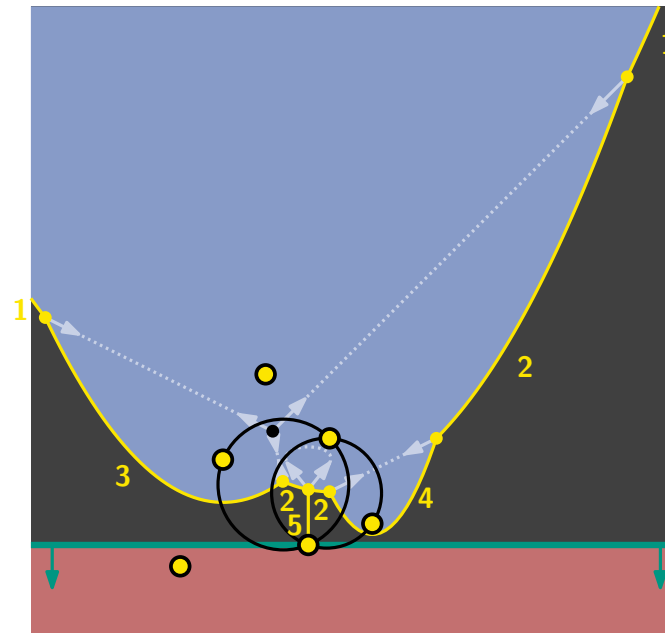
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line: (s_1, s_3) (s_3, s_2) (s_2, s_5) (s_5, s_2) (s_2, s_4) (s_4, s_2) (s_2, s_1)

Überblick

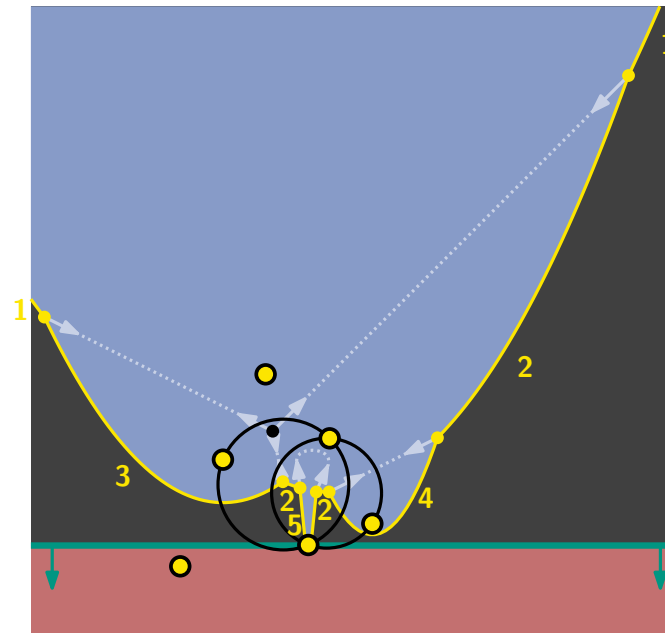
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line: (s_1, s_3) (s_3, s_2) (s_2, s_5) (s_5, s_2) (s_2, s_4) (s_4, s_2) (s_2, s_1)

Überblick

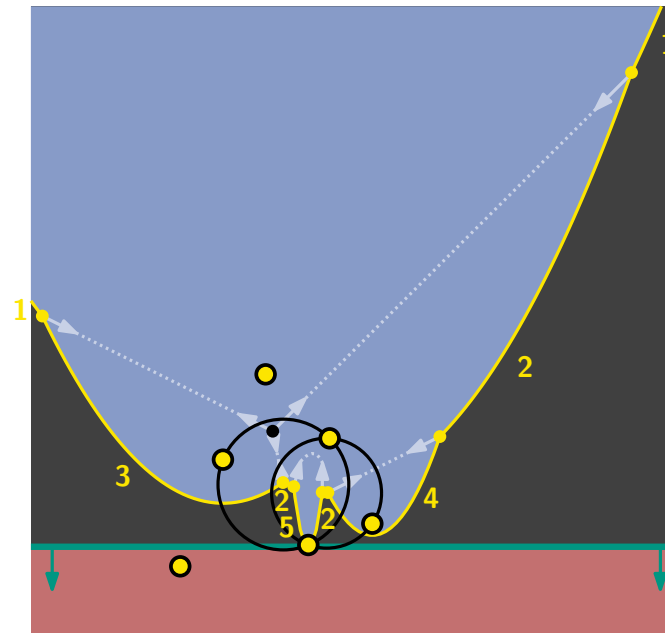
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line: (s_1, s_3) (s_3, s_2) (s_2, s_5) (s_5, s_2) (s_2, s_4) (s_4, s_2) (s_2, s_1)

Überblick

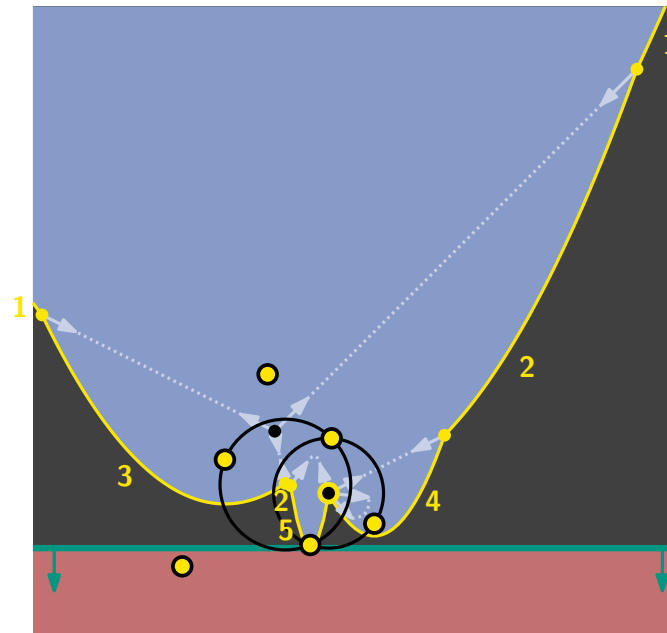
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line: (s_1, s_3) (s_3, s_2) (s_2, s_5) (s_5, s_4) (s_4, s_2) (s_2, s_1)

Überblick

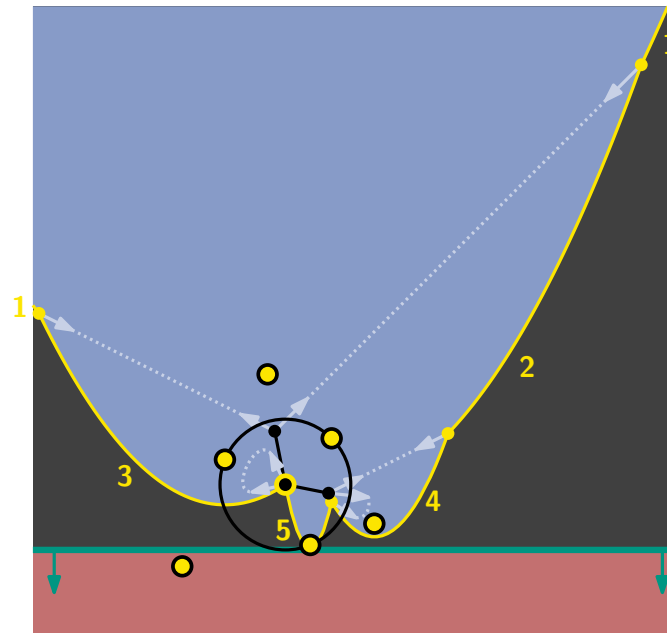
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line: (s_1, s_3) (s_3, s_5) (s_5, s_4) (s_4, s_2) (s_2, s_1)

Überblick

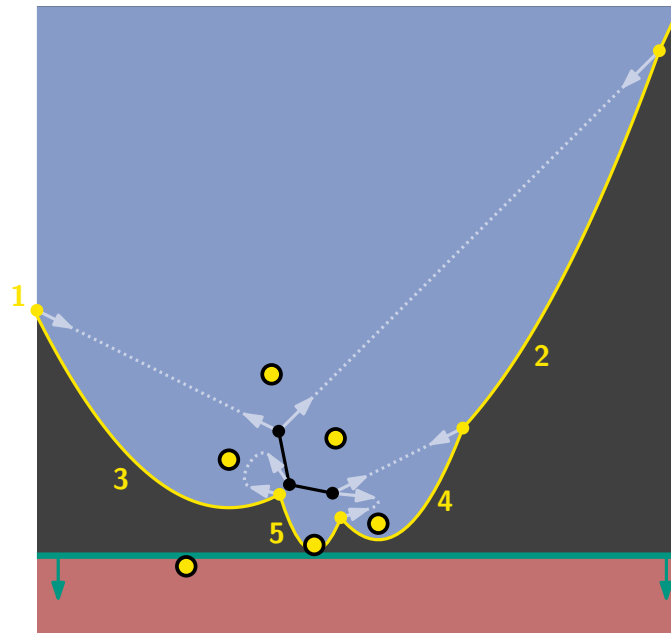
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line: (s_1, s_3) (s_3, s_5) (s_5, s_4) (s_4, s_2) (s_2, s_1)

Überblick

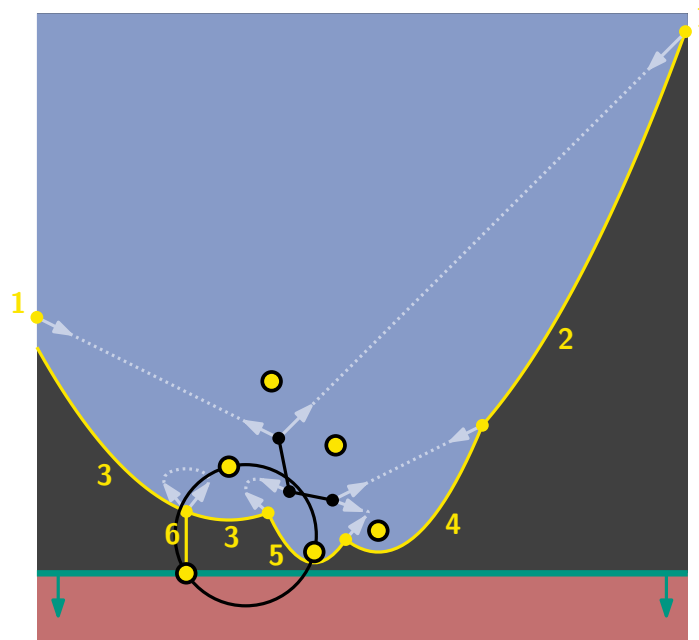
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line: (s_1, s_3) (s_3, s_6) (s_6, s_3) (s_3, s_5) (s_5, s_4) (s_4, s_2) (s_2, s_1)

Überblick

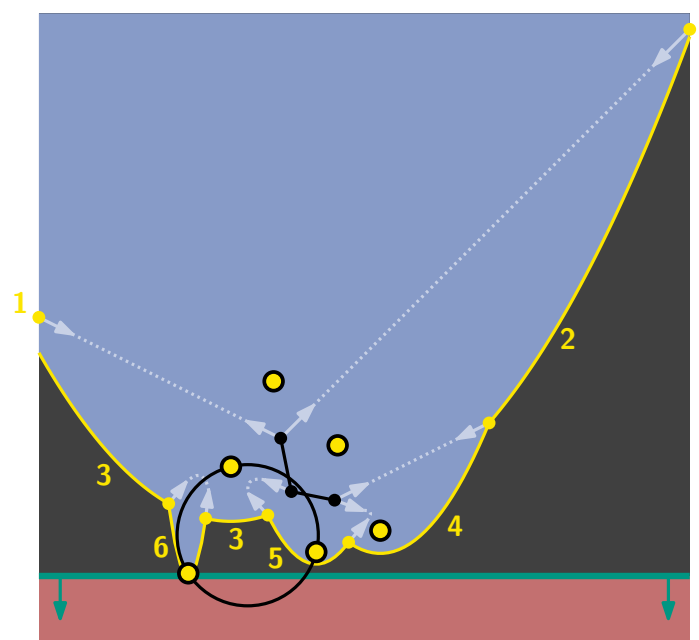
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line: (s_1, s_3) (s_3, s_6) (s_6, s_3) (s_3, s_5) (s_5, s_4) (s_4, s_2) (s_2, s_1)

Überblick

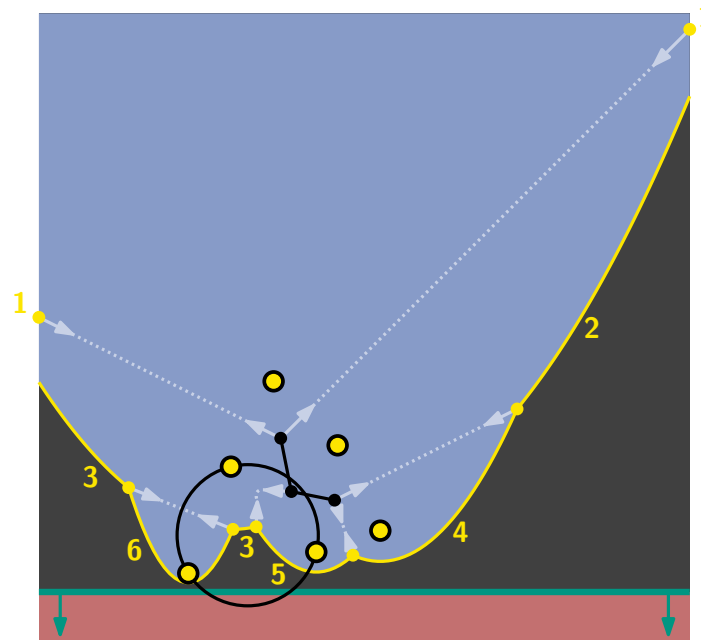
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line: (s_1, s_3) (s_3, s_6) (s_6, s_3) (s_3, s_5) (s_5, s_4) (s_4, s_2) (s_2, s_1)

Überblick

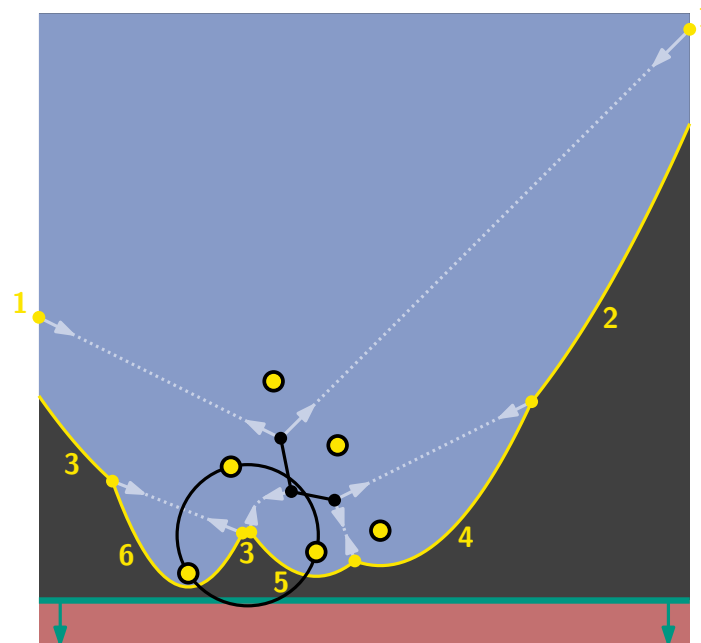
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line: (s_1, s_3) (s_3, s_6) (s_6, s_3) (s_3, s_5) (s_5, s_4) (s_4, s_2) (s_2, s_1)

Überblick

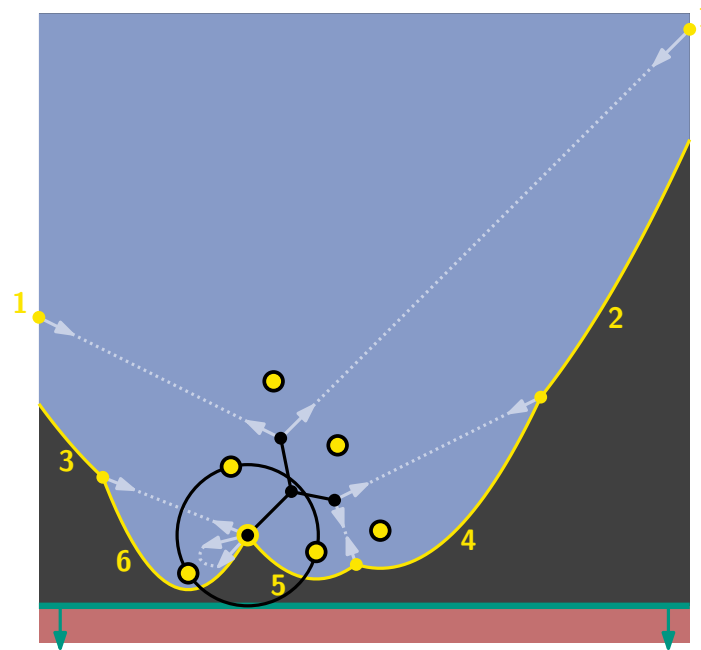
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line: (s_1, s_3) (s_3, s_6) (s_6, s_5) (s_5, s_4) (s_4, s_2) (s_2, s_1)

Überblick

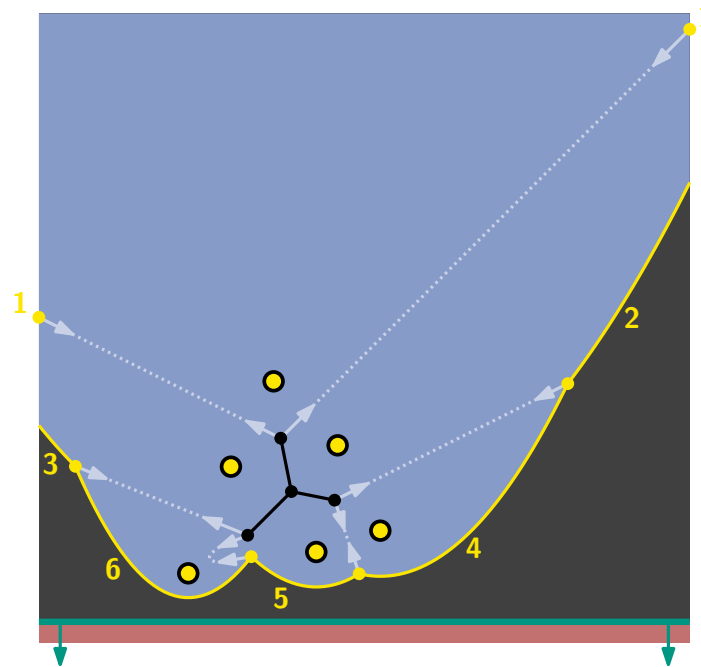
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line: (s_1, s_3) (s_3, s_6) (s_6, s_5) (s_5, s_4) (s_4, s_2) (s_2, s_1)

Überblick

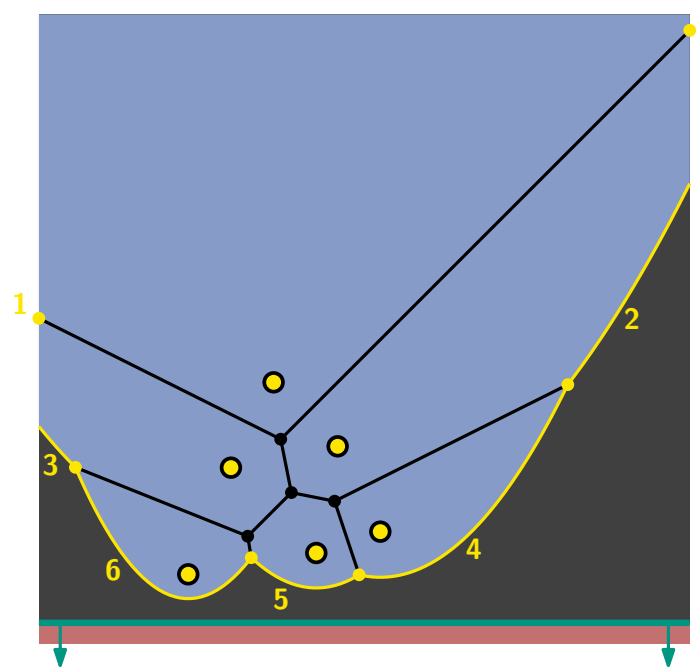
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line: (s_1, s_3) (s_3, s_6) (s_6, s_5) (s_5, s_4) (s_4, s_2) (s_2, s_1)

Überblick

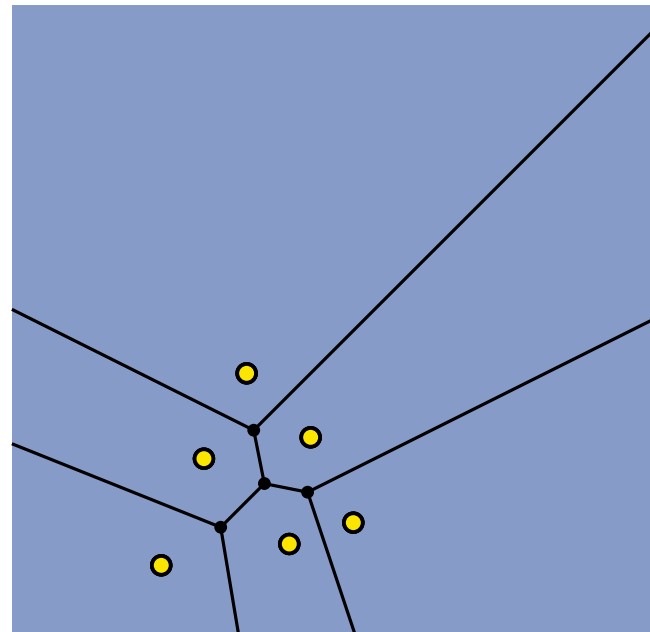
Standort-Event (Standort s)

- finde s_x in der Beach-Line
- füge zwei neue Schnittpunkte ein
- behandle falschen Alarm (für zugeh. Parabel)
- teste neue Schnittpunkte auf Kreis-Events
- neues Paar Halbkanten für die Schnittpunkte

Kreis-Event (für Schnittpunkte (s_i, s_j) , (s_j, s_k))

- entferne (s_i, s_j) und (s_j, s_k)
- füge (s_i, s_k) ein
- teste (s_i, s_k) mit Nachbarn (in BL) auf Kreis-Events
- erstelle neuen Knoten v
- hänge Halbkanten von (s_i, s_j) und (s_j, s_k) an v
- neues Paar Halbkanten: eine an v , eine an (s_i, s_k)

Beispiel



Beach-Line:

Theorem

Das Voronoi-Diagramm von n Punkten kann in $O(n \log n)$ Zeit mit $O(n)$ Speicherplatz berechnet werden.

Zusammenfassung

Theorem

Das Voronoi-Diagramm von n Punkten kann in $O(n \log n)$ Zeit mit $O(n)$ Speicherplatz berechnet werden.

Was gibt es sonst noch?

- man kann diverse Varianten betrachten
 - höhere Dimensionen
 - andere Metriken
 - gewichtete Standorte
 - Standorte, die nicht nur Punkte sind (beispielsweise Strecken)
 - Voronoi Diagramme höherer Ordnung

Zusammenfassung

Theorem

Das Voronoi-Diagramm von n Punkten kann in $O(n \log n)$ Zeit mit $O(n)$ Speicherplatz berechnet werden.

Was gibt es sonst noch?

- man kann diverse Varianten betrachten
 - höhere Dimensionen
 - andere Metriken
 - gewichtete Standorte
 - Standorte, die nicht nur Punkte sind (beispielsweise Strecken)
 - Voronoi Diagramme höherer Ordnung
- Visualisierung: <http://www.raymondhill.net/voronoi/rhill-voronoi.html>