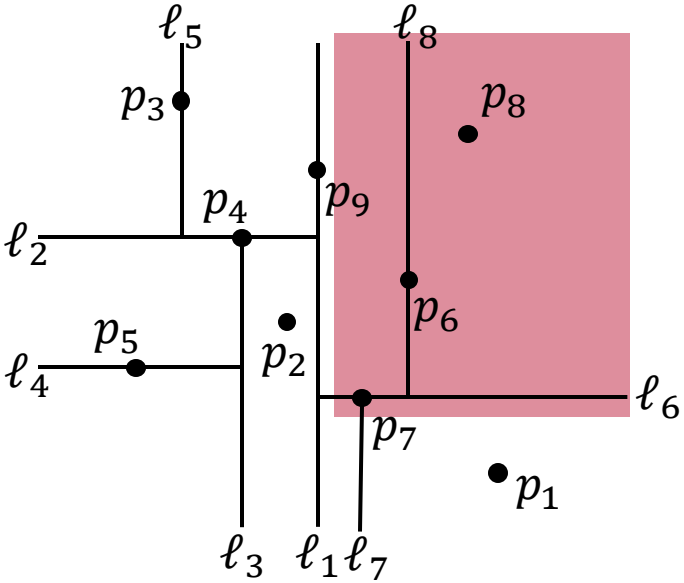
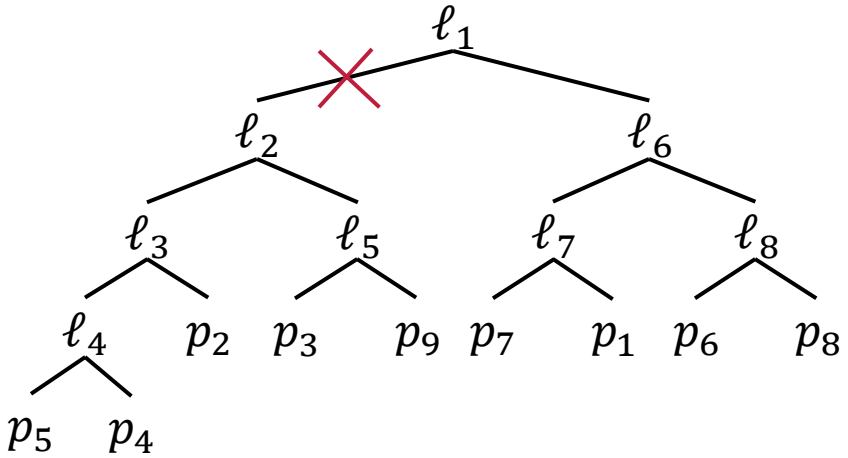
Technische Universität Braunschweig

Einführung in algorithmische Geometrie – Sweep Line

Arne Schmidt

Letztes Mal

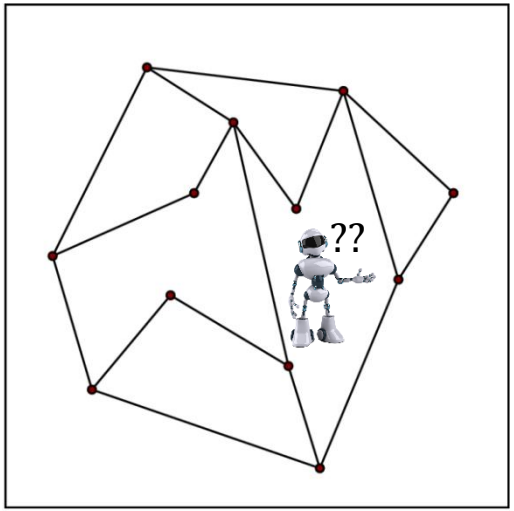




Welche Punkte liegen im Rechteck?

Antwort: p_7 p_6 p_8

Kapitel 2.2 – Punktlokalisierung

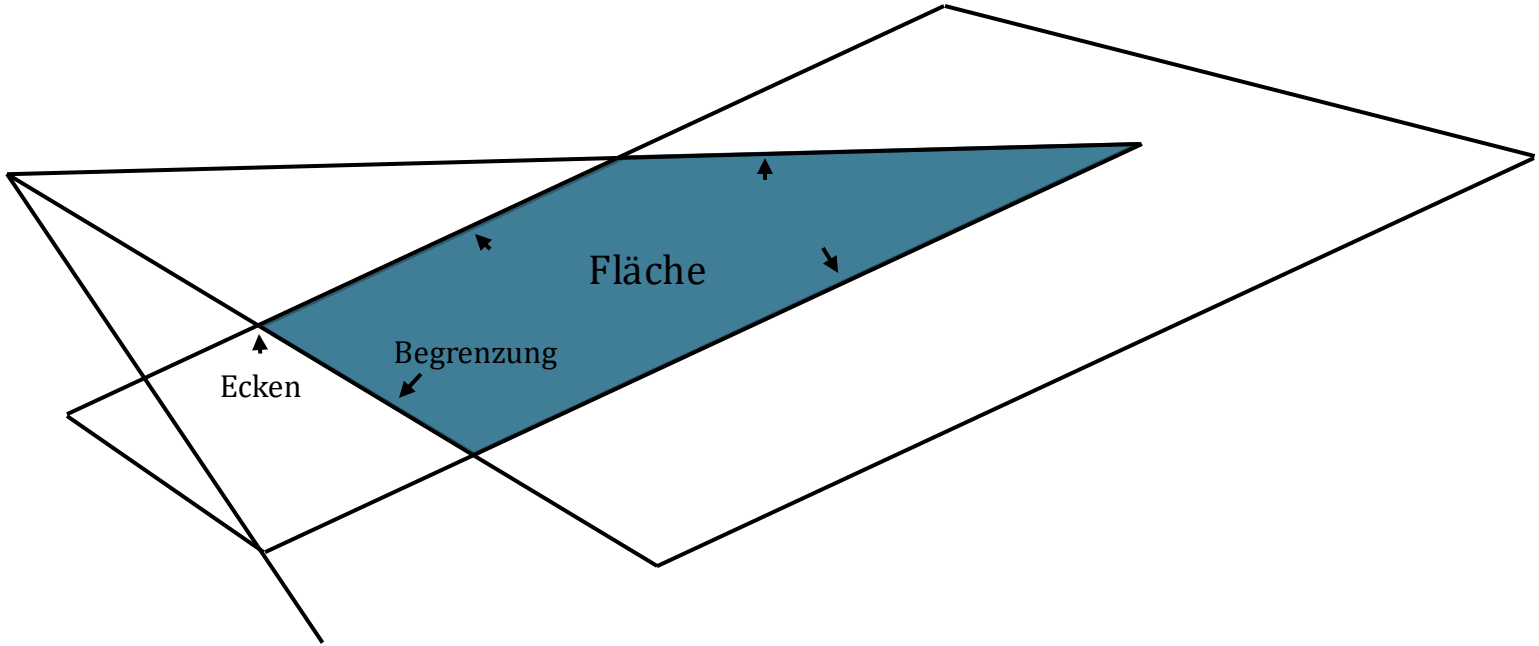


Überlegungen - Karten

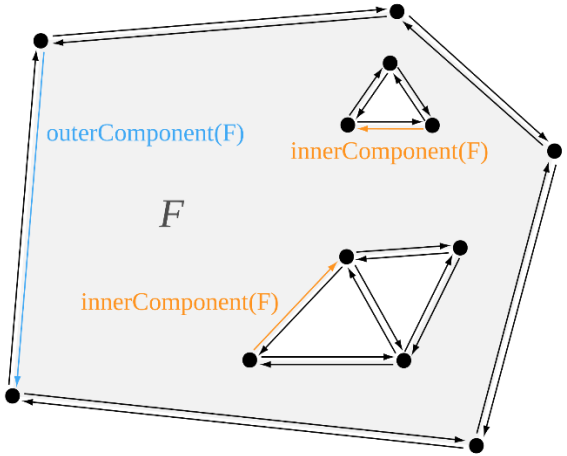
Karten sind eine Sammlung von Flächen.

Flächen sind eine Ansammlung von Linien.

Unterteilung der Ebene



Kapitel 2.2.1 – Arrangements



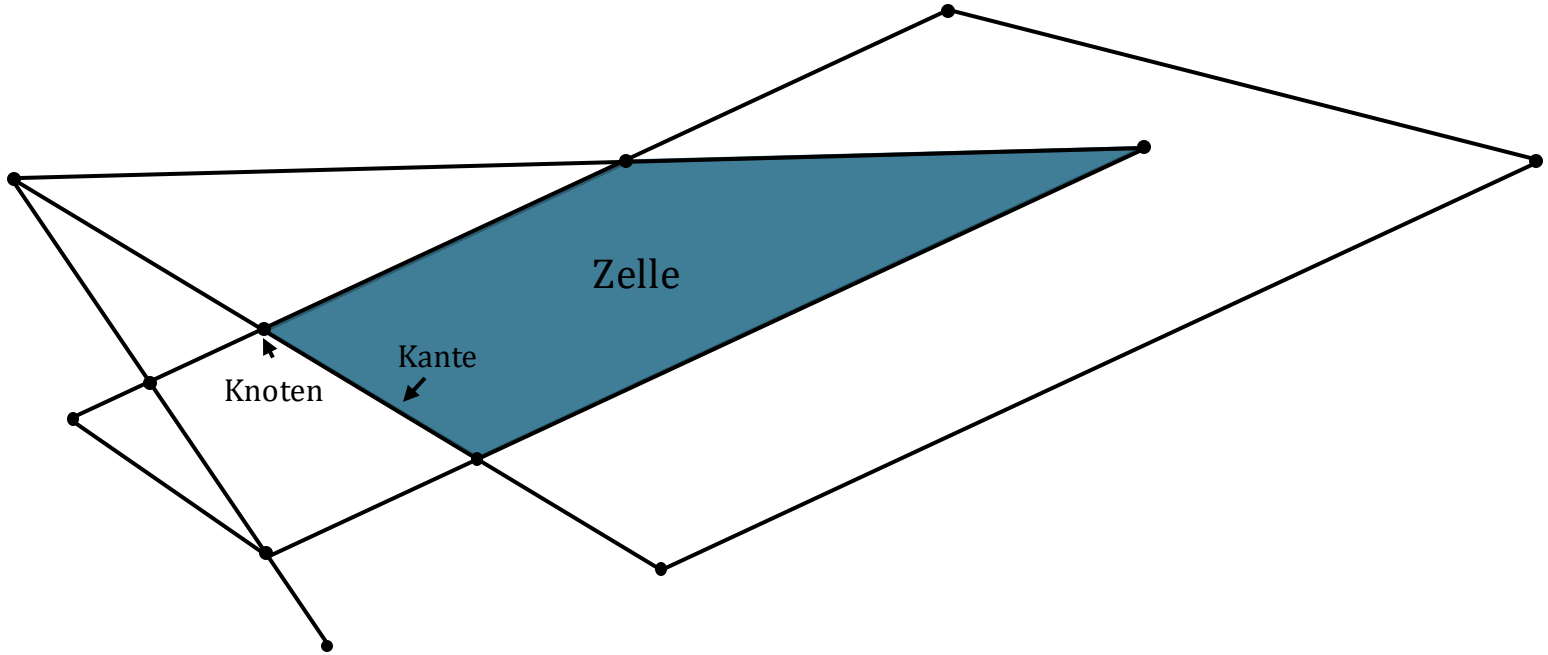
https://de.wikipedia.org/wiki/Half-Edge-Datenstruktur#/media/Datei:Dcel-components.svg

Arrangement – Definition

Definition 2.17 (2D-Arrangement)
Für eine Menge von Segmenten S, sei $\mathcal{A}(S)$ das dazugehörige Arrangement. $\mathcal{A}(S)$ besteht dann aus:

- Zellen (**faces**): Komponenten von $\mathbb{R}^2 \setminus S$. Eine unbeschränkt große Zelle wird auch äußere Zelle (**outer face**) bezeichnet.
- Jede Zelle F wird von (Teil-)Segmenten beschränkt. Diese werden auch **Kanten** genannt.
- Endpunkten der Kanten, auch **Knoten** genannt.

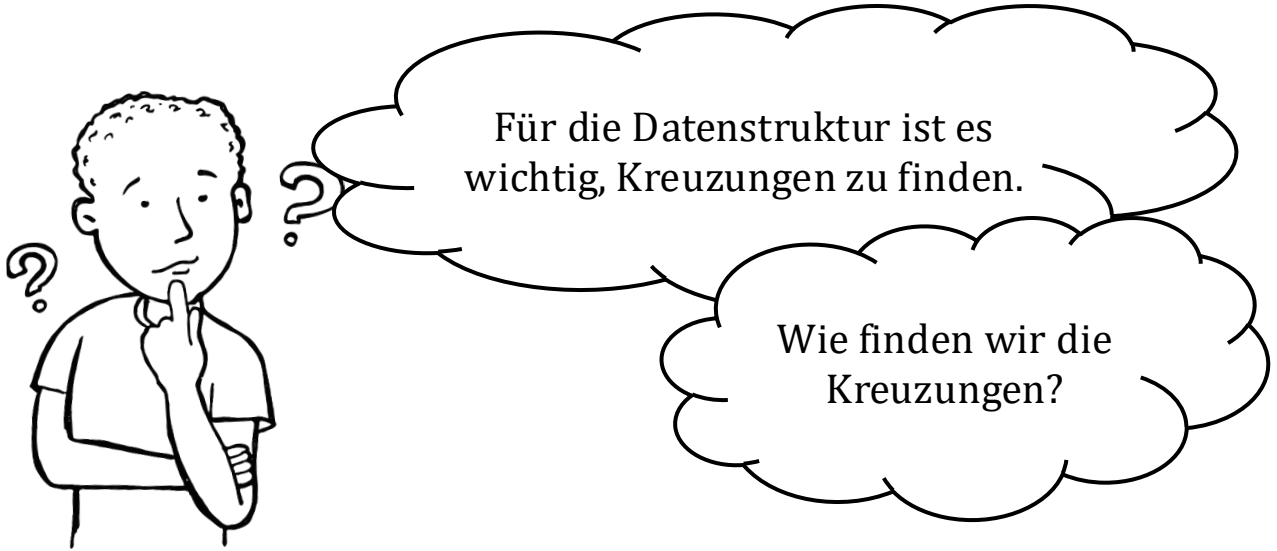
Unterteilung der Ebene



Komplexität eines Arrangements

Lemma 2.18:
Sei S eine Menge von Segmenten. Dann besitzt $\mathcal{A}(S)$ maximal $|S|^2$ viele Zellen.

Frage: Existiert eine Datenstruktur, welche in Zeit und mit Speicherbedarf $O(S^2)$ konstruiert werden kann?



Kreuzungen finden

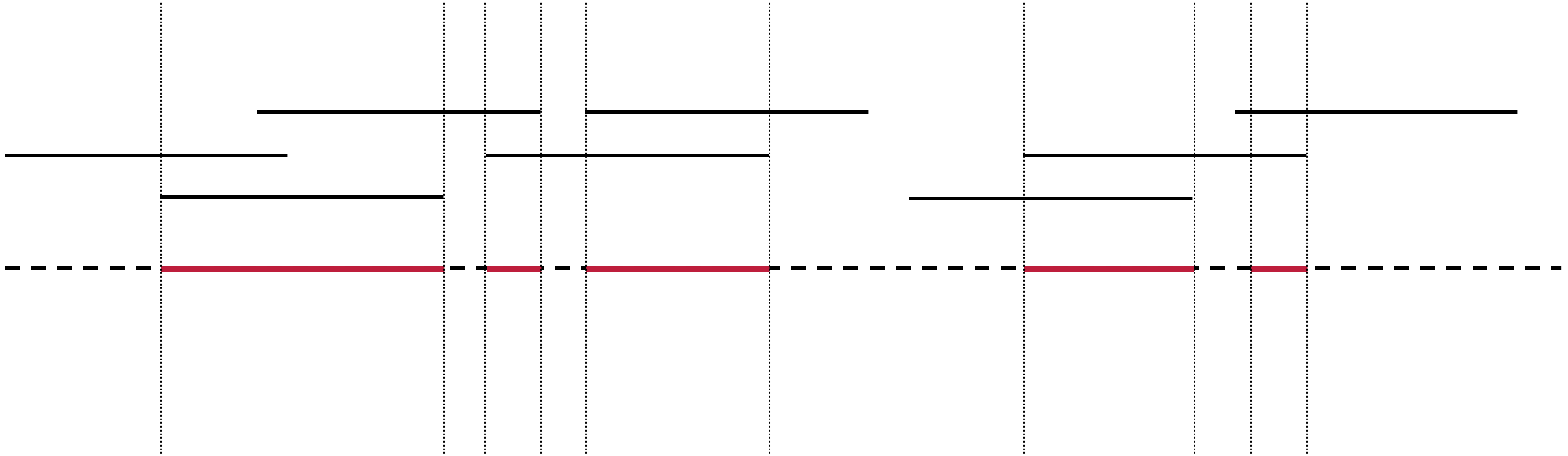
Es existiert ein Algorithmus mit Laufzeit $O(n^2)$, um alle Schnittpunkte von n Segmenten zu finden:
Für jedes Paar von Segmenten (S_1, S_2) teste, ob sie sich schneiden.

Vorteil:
- Einfach zu programmieren
- Funktioniert!

Nachteil:
- Nicht alle Segmente müssen sich schneiden.
  Wir testen zu viel!

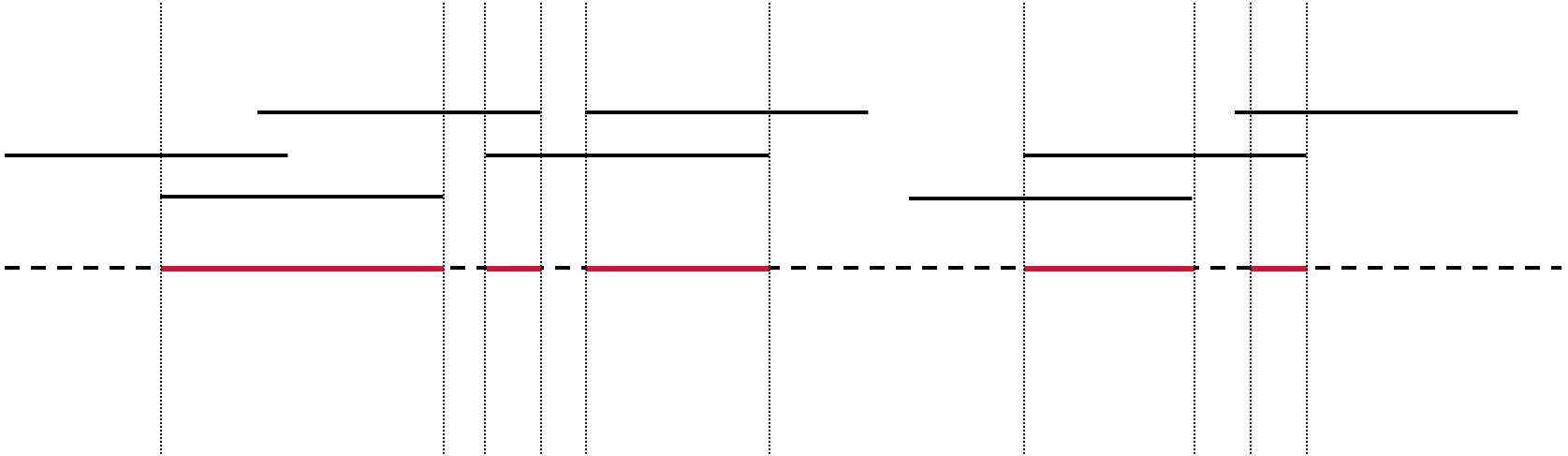
Gehen wir zunächst auf 1D:
Gegeben seien n Intervalle $I_1, \dots, I_n$.
Ausgegeben sollen die maximalen Intervalle aus $\bigcup_{i=1}^{n} \bigcup_{1=j\neq i}^{n} I_j \cap I_i$ in Zeit $O(n \log n)$.

Intervalle



Welche Eigenschaften besitzen die senkrechten Linien?

Intervalle

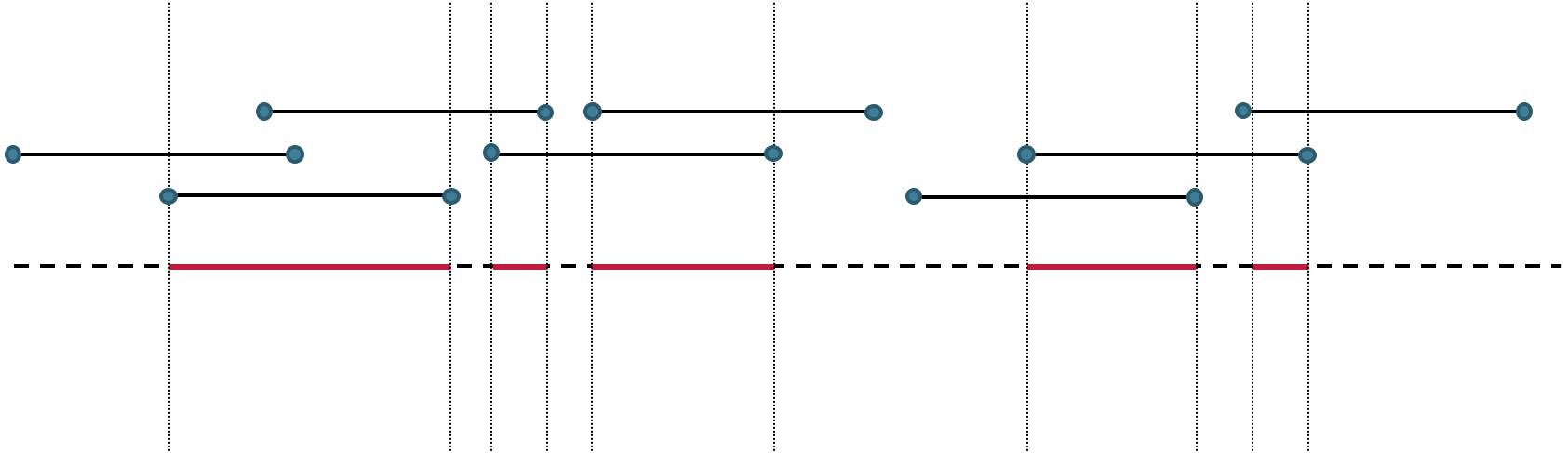


Welche Eigenschaften besitzen die senkrechten Linien?

- Sie trennen Gebiete mit mind. zwei überlappenen Intervallen und max. einem.
- Sie berühren immer einen Start- oder Enpunkt eines Intervalls!

Formalisieren wir einen Algorithmus an der Tafel!

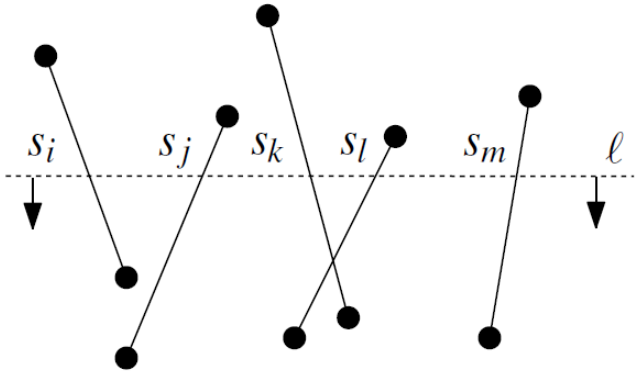
Intervalle

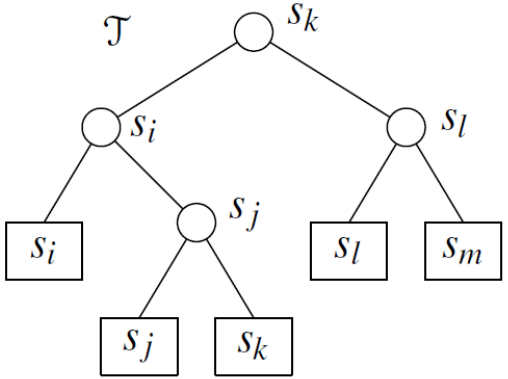


Recap:

- Sweep von links nach rechts.
- Nur am Start-/Endpunkt eines Intervalls ändert sich etwas → *Event-Point!*

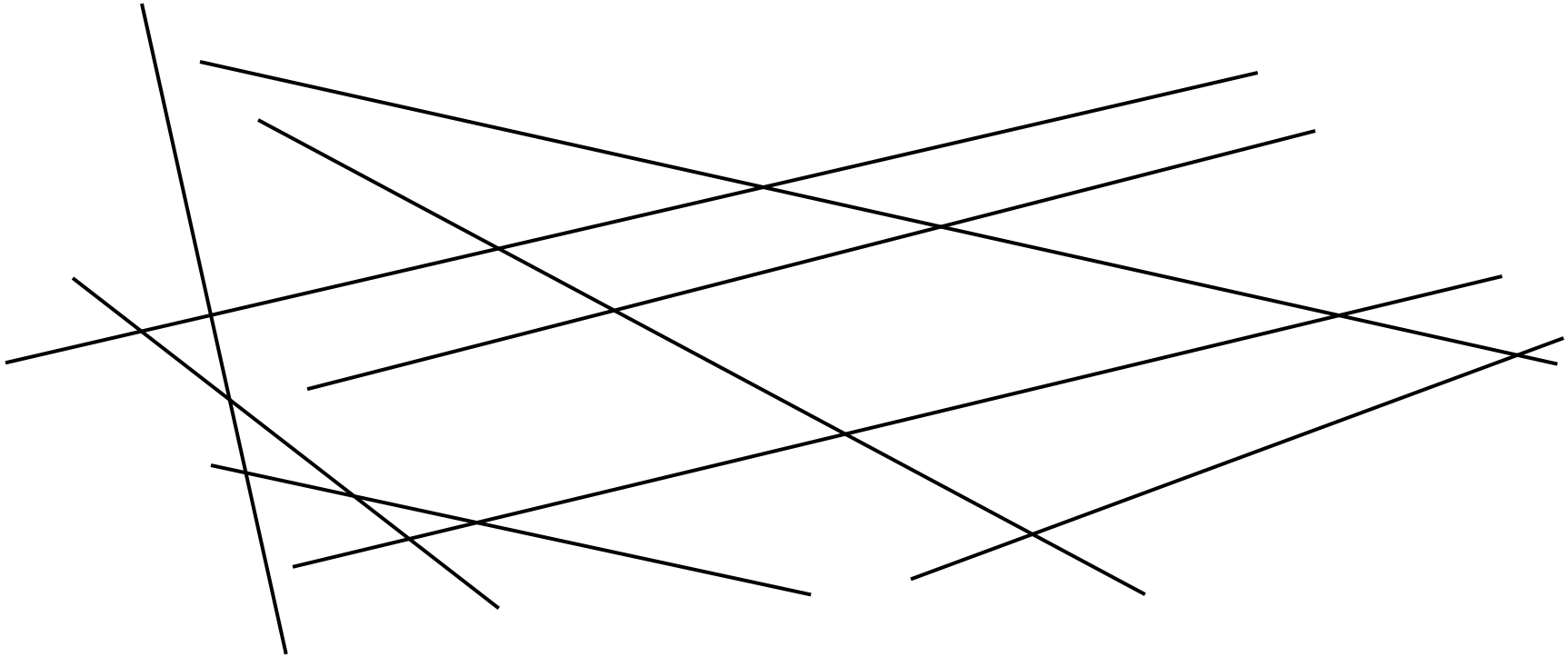
Kapitel 2.2.2 – Sweep-Line-Algorithmen



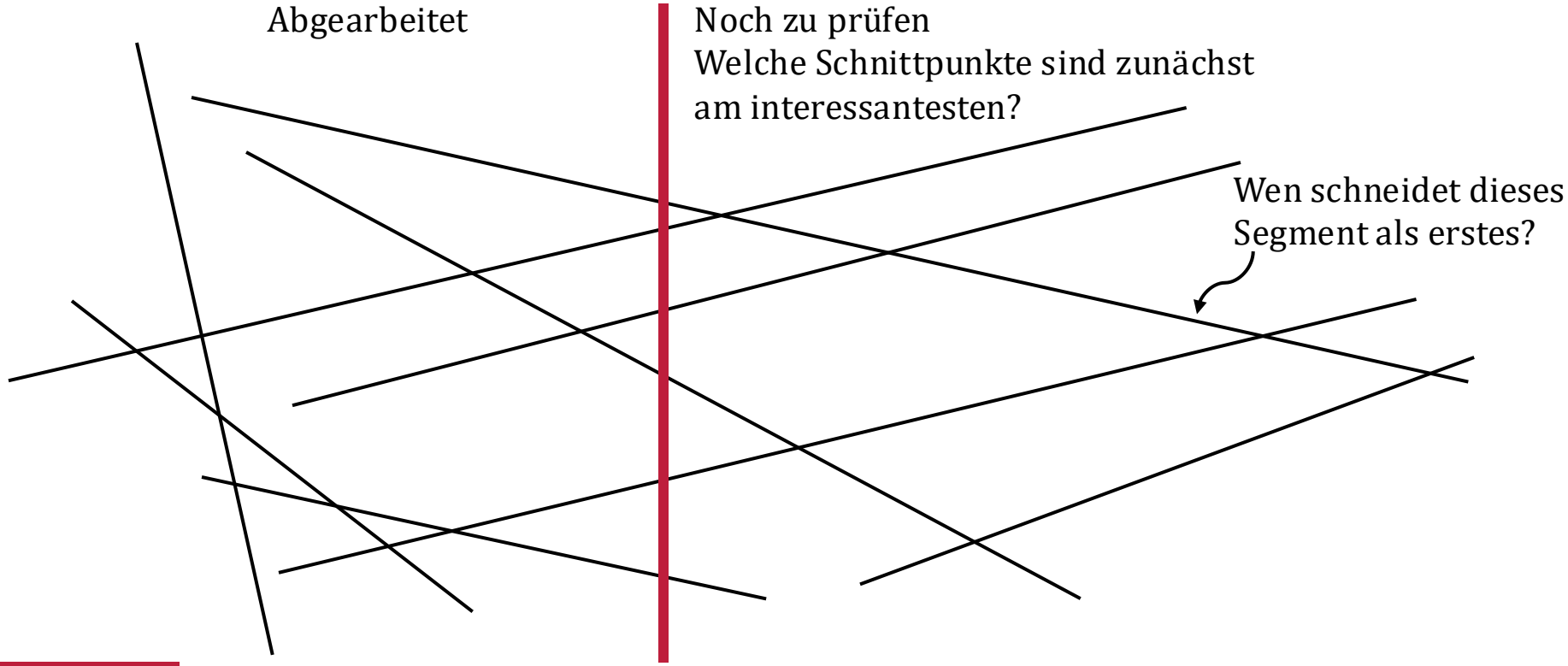


de Berg et al.: Computational Geometry – Algorithms and Applications

Algorithmus-Idee



Algorithmus-Idee



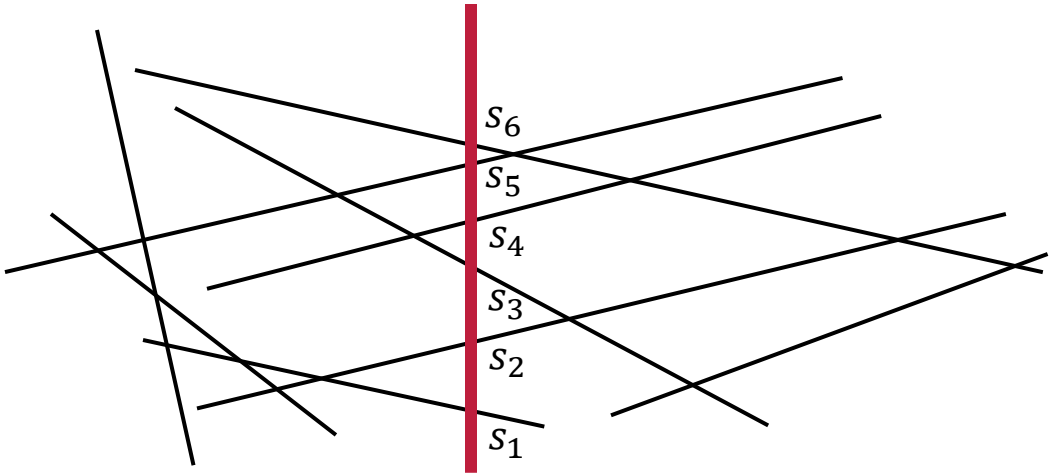
Idee Zusammenfassung

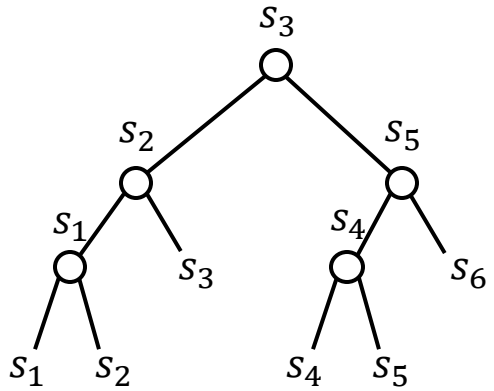
- Gehe die Segmente von links nach rechts durch.
- Berechne dabei, welche Schnittpunkte am interessantesten sind
  - Nur „benachbarte“ Segmente sind interessant.
    - Speichere diese in einer geeigneten Datenstruktur
  - Nachbarn können sich nur ändern, wenn:
    - Zwei Segmente sich schneiden
    - Ein Segment beginnt oder endet

    } Event Points

- Zwischen diesen *Event Points* passiert nichts Spannendes!
- Welche Datenstruktur benutzen wir für benachbarte Segmente?
  - Hinzufügen / Löschen / Tauschen von Segmenten muss schnell sein.
  - Tauschen kann durch Löschen + Hinzufügen simuliert werden.

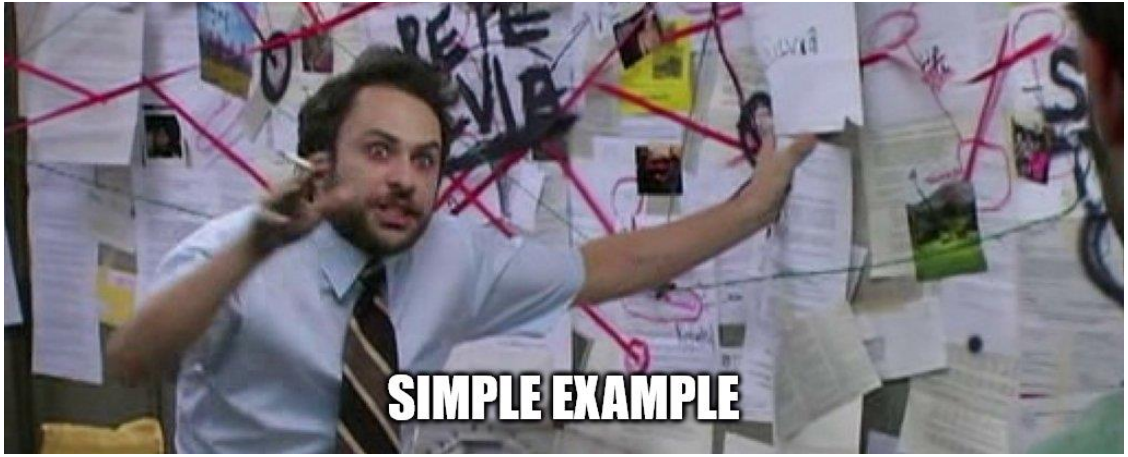
Datenstruktur – Balancierte Suchbäume!





Einfügen / Löschen von Segmenten im Suchbaum benötigt $O(\log n)$ Zeit.

Sweep-Line Algorithmus an der TAFEL!



Sweep Line – Wrapper

Algorithmus 2.19a (FINDINTERSECTIONS)
Gegeben: Menge S von Segmenten.
Ausgabe: Menge I von Schnittspunkten von S

1. Initialisiere eine Event-Queue Q (Priorität: x-Koordinate)
2. Füge alle Endpunkte von S zu Q hinzu.
3. Initialisiere einen balancierten Suchbaum $\mathcal{T}$
4. **While** (Q nicht leer) **do**
   a) Wähle nächsten Event Point $q \in Q$
   b) HANDLEEVENTPOINT(q)

Sweep Line – Handle Event Points

Algorithmus 2.19b (HANDLEEVENTPOINTS)
Gegeben: Balancierter Suchbaum $\mathcal{T}$, Punkt q.
Ausgabe: Menge I von Schnittspunkten von S

1. Sei $L(q)$ die Menge aller Segmente, dessen linker Knoten q ist.
2. Finde alle Segmente in $\mathcal{T}$, die q enthalten (sind in $\mathcal{T}$ benachbart!).
   Seien dazu $R(q)$ die Segmente mit rechtem Endknoten und $C(q)$ die Segmente auf welchen q liegt.
3. **If** $(|L(q) \cup R(q) \cup C(q)| \geq 2)$ **Then** Gib q als Schnittpunkt aus.
4. Lösche $R(q) \cup C(q)$ von $\mathcal{T}$.
5. Füge $L(q) \cup C(q)$ in $\mathcal{T}$ hinzu.
6. **If** $(L(q) \cup C(q) = \emptyset)$ **Then** Füge Schnittpunkt von Nachbarn von q in $\mathcal{T}$ zu Q hinzu.*
7. **Else:**
   a) Füge Schnittpunkt von linkestem Segment aus $L(q) \cup C(q)$ und seinem linken Nachbarn zu Q hinzu.*
   b) Füge Schnittpunkt von rechtestem Segment aus $L(q) \cup C(q)$ und seinem rechten Nachbarn zu Q hinzu.*

*: Der Schnittpunkt muss natürlich existieren und rechts von q liegen.

Laufzeit Kreuzungen finden

Theorem 2.20:
Algorithmus 2.19a gibt alle Schnittpunkte in Zeit $O(n \log n + I \log n)$ aus, wobei I die Anzahl an Schnittpunkten der n Segmente ist.

Ein für den Beweis nützliches Resultat, ist die sogannte Polyederformel:

Lemma 2.21 (Polyederformel)
Sei $\mathcal{A}$ ein Arrangement mit Zellen F, Kanten E und Knoten V. Dann gilt:

$$|V| + |F| - |E| \geq 2$$

Weiter folgt daraus

$$|E| \leq 3|V| - 6$$

Theorem 2.22
Alle Schnittpunkte lassen sich in $O(n \log n + I \log n)$ Zeit mit $O(n)$ Speicherbedarf finden, wobei I die Anzahl an Schnittpunkten von n Segmenten ist.

Sweep Line – Allgemein

Wie soll gesweept werden (entlang einer Achse, um einen Punkt herum, ...)?

Was sind die Event-Points? Gibt es verschiedene Event-Typen?

Wie werden Event-Points behandelt?

Welche Datenstrukturen werden benutzt?

Nächste Woche

Wie kann man nun effizient Arrangements speichern?