

# Algorithmische Geometrie

Wladimir Emdin und Martin Dames

Skript vom 3.6.2005

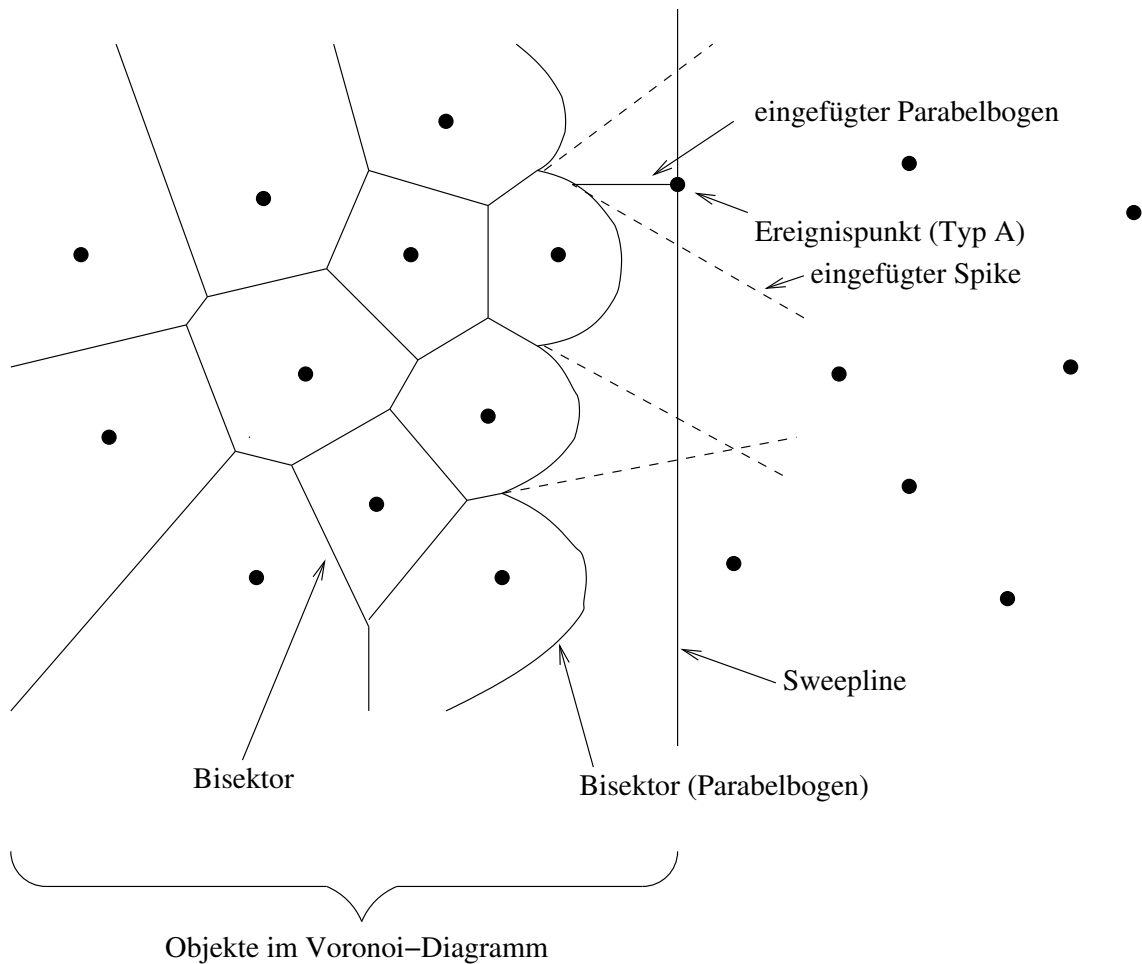
## 1 Wiederholung und Ergänzung zu Fortune Sweep

Gegeben: eine endliche Menge an Punkten  $S \subset \mathbb{R}^2$

Gesucht: Das Voronoi-Diagramm von  $S$

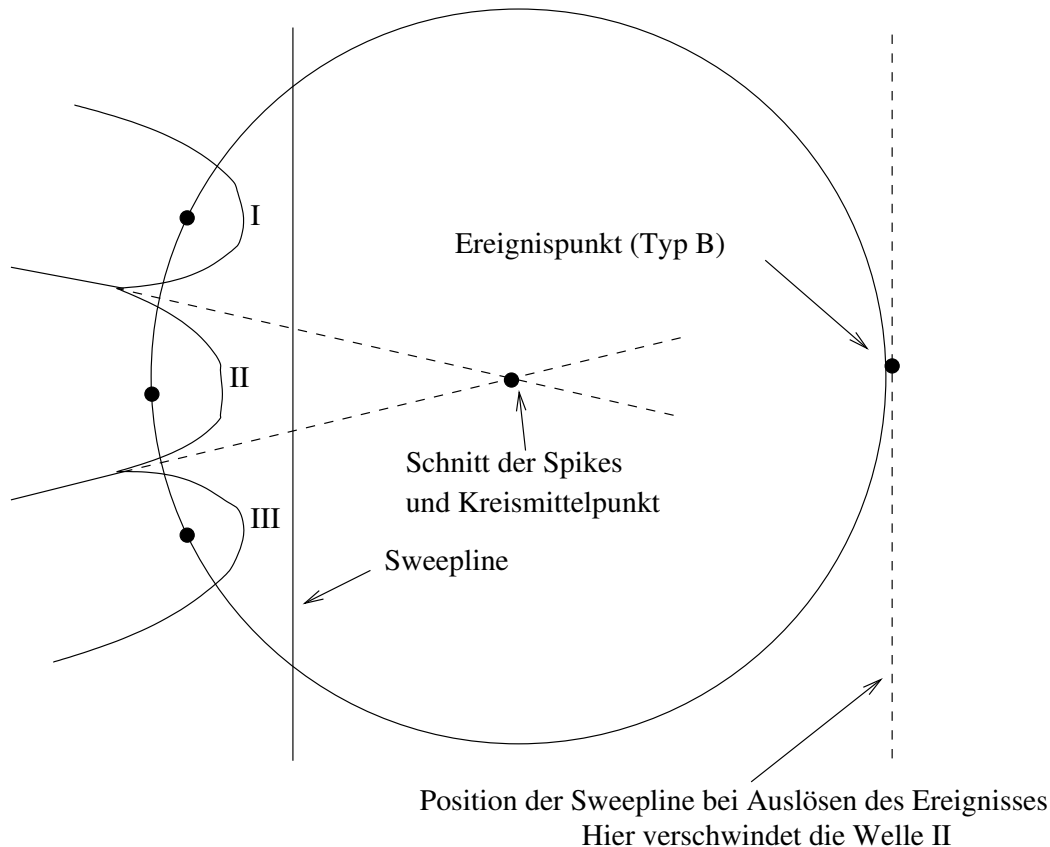
Es treten zwei verschiedene Ereignisse im EPS auf:

### 1.1 Typ A: Überstreichen der Sweepline eines Punktes aus $S$



Es wird eine neue Welle eingefügt (die alte Welle wird aufgeteilt). Das Einfügen kostet  $O(\log(n))$  Zeit, da der SLS in einer Suchstruktur (Wörterbuch) organisiert ist.  
 Es entsteht ein neuer Spike.  
 Es werden wenn nötig neue Ereignispunkte des Typs B eingefügt.

## 1.2 Typ B: Überstreichen eines hinzugefügten Punktes



Wenn die Sweepline ein Ereignis vom Typ B auslöst, wird die zu dem Ereignispunkt zugeordnete Welle aus dem SLS entfernt. Dieser Vorgang kostet  $O(\log(n))$  Zeit.  
 Das Ereignis wird auch Kreisevent oder Circleevent genannt.

Typ B Ereignisse werden von Typ A Ereignissen berechnet. Es entsteht ein neuer Spike, der tangential zur Parabel liegt, welche von der neuen Welle getroffen wird. Die eventuellen Schnittpunkte des Spikes mit seinen Nachbarn bilden den Mittelpunkt eines Kreises. Der neue Ereignispunkt (Typ B) entsteht ganz rechts auf diesem Kreis.

## 1.3 Visualisierung des Fortune Sweep Verfahrens

Eine Visualisierung dieses Verfahrens kann man sich als Applet unter <http://www.diku.dk/hjemmesider/studerende/duff/Fortune/> anschauen.

## 1.4 Analyse

### 1.4.1 Verwendete Datenstrukturen

EPS: organisiert als Heap

SLS: organisiert als Suchstruktur (Wörterbuch)

### 1.4.2 Initialisieren des EPS

Der EPS (Heap) wird mit den X-Koordinaten der Punkte aus  $S$  initialisiert. Die Laufzeit hierfür beträgt  $O(n)$  mit  $|S| = n$ .

### 1.4.3 Typ A Ereignisse

- für jede entstehende Welle (Typ A Ereignis)
  - Die getroffene Welle wird gespalten und die neue Welle wird dazwischen in den SLS eingefügt. Laufzeit:  $O(\log(n))$
  - berechnen der Schnittpunkte der neuen Spikes mit Nachbarn und den dazugehörigen neuen Eventpoints (Typ B) in EPS einfügen. Laufzeit:  $O(\log(n))$

Die gesamte Anzahl der Wellen ist  $O(n)$ , da jeder überstrichene Punkt aus  $S$  die Anzahl der Wellen um höchstens zwei erhöht.

Es ergibt sich somit eine Laufzeit von  $O(n \cdot \log(n))$ .

### 1.4.4 Typ B Ereignisse

- für jedes Typ B Ereignis
  - Streichen einer Welle aus dem SLS. Laufzeit:  $O(\log(n))$

Für jeden Knoten im Voronoi-Diagramm tritt ein Typ B Ereignis auf. Da die Anzahl der Knoten im Voronoi-Diagramm  $O(n)$  beträgt, ergeben sich also  $O(n)$  Typ B Ereignisse. Es ergibt sich somit eine Laufzeit von  $O(n \cdot \log(n))$ .

### 1.4.5 Gesamtlaufzeit

Es ergibt sich also eine Laufzeit von  $O(n) + O(n \cdot \log(n)) + O(n \cdot \log(n))$

(Initialisierung + Typ A Ereignisse + Typ B Ereignisse).

Die Gesamtlaufzeit beträgt  $O(n \cdot \log(n))$ .

## 2 Kapitel 4 - Geometrische Datenstrukturen

Es wurde schon eine „geometrische Datenstruktur“ behandelt, nämlich die Suchstruktur von Kirkpatrick (Pointlocation).

### 2.1 Bereichsanfragen (range queries)

Problemstellung:

Sei  $\Gamma \subset 2^{R^k}$  (Potenzmenge).

- gegeben:
  - eine feste endliche Menge an Punkten  $S \subset R^k$
  - $U \in \Gamma$
- gesucht:
  - $S \cap U$  oder  $|S \cap U|$

Die Anwendungen von „range queries“ sind sehr zahlreich.


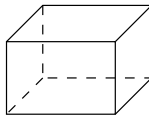
- Beispiele:
  - gib alle Orte zwischen dem 20. und 30. Längengrad und 10. und 20. Breitengrad
  - gib alle Einträge im Telefonbuch von B-K, die mit den Ziffern 3 bis 7 beginnen

Die Algorithmen für „range queries“ sind im Allgemeinen case-sensitive.

Eine wichtige Unterart der Bereichsabfragen sind „orthogonale Bereichsabfragen“.

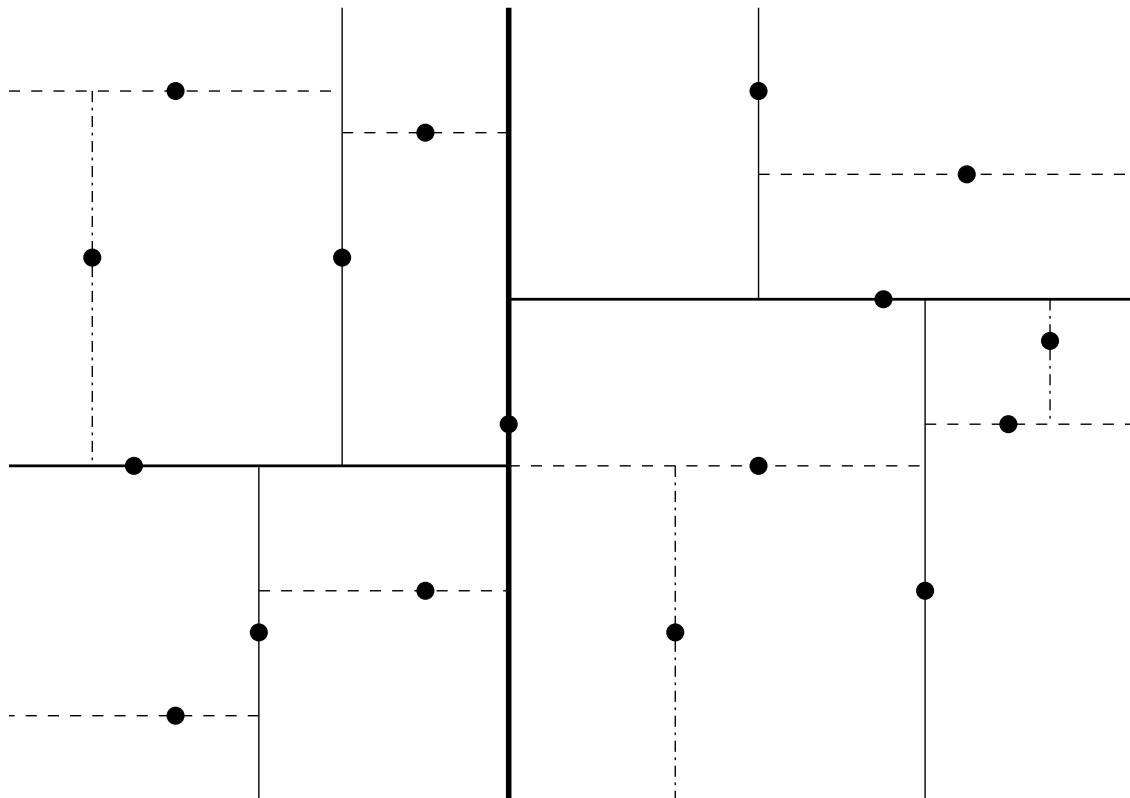
Hier gilt  $\Gamma = \text{Menge aller kartesischen Produkte von } k \text{ Intervallen}$ .






- Beispiele

- für  $k = 2$  
- für  $k = 3$  

Wobei die Seiten auch  $\infty$  lang sein können.

## 2.2 Beispiel in zwei Dimensionen ( $k = 2$ )



	$d(v)=1$	$j=1$
	$d(v)=2$	$j=2$
	$d(v)=3$	$j=1$
	$d(v)=4$	$j=2$
	$d(v)=5$	$j=1$

- Aufspalten der Punktmenge an einem Punkt in linke und rechte Hälfte  
(nach x-Koordinate  $j = 1 \wedge d(v) = 1$ )
- Aufspalten der zwei Punktmenge an einem Punkt in obere und untere Hälfte  
(nach y-Koordinate  $j = 2 \wedge d(v) = 2$ )
- Aufspalten der vier Punktmenge an einem Punkt in linke und rechte Hälfte  
(nach x-Koordinate  $j = 1 \wedge d(v) = 3$ )
- ...

## 2.3 Definition von k-d-Bäumen

Sei  $S$  eine Teilmenge im  $k$ -dimensionalen Raum ( $S \subset R^k \wedge |S| = n$ )

Ein  $k$ -d-Baum  $T$  für  $S$  (beginnend mit Koordinate  $i$ ) ist definiert durch:

1. Falls  $k = n = 1$ , d.h.  $S = \{x\}$ , dann ist  $T$  ein einzelnes Blatt, das  $x$  enthält.
2. Falls  $k > 1 \vee n > 1$ , dann ist die Wurzel von  $T$  markiert mit  $x \in R$  und hat drei Unterbäume  $T_{<}$ ,  $T_{=}$  und  $T_{>}$ .
  - $T_{<}$  ist ein  $k$ -d-Baum beginnend mit Koordinate  $j$ , wobei

$$j = \begin{cases} i + 1 & \text{für } i < k \\ 1 & \text{für } i = k \end{cases}$$

für die Menge  $S_{<} = \{(x_1, \dots, x_k) \in S \mid x_i < x\}$

- $T_{>}$  ist ein  $k$ -d-Baum beginnend mit Koordinate  $j$ , wobei

$$j = \begin{cases} i + 1 & \text{für } i < k \\ 1 & \text{für } i = k \end{cases}$$

für die Menge  $S_{>} = \{(x_1, \dots, x_k) \in S \mid x_i > x\}$

- $T_{=}$  ist ein  $(k-1)$ -d-Baum für die Menge  $S_{=} = \{(x_1, \dots, \hat{x}_i, \dots, x_k) \in S \mid x_i = x \wedge x_i \neq \hat{x}_i\}$ .  
 $\hat{x}_i$  ist der Punkt auf dem die Teilung vorgenommen wurde. Er ist nicht in  $S_{=}$  enthalten.

## 2.4 Bezeichnungen

$T$  sei ein  $k$ -d-Baum und  $v$  ein Knoten von  $T$ .

$s(v)$  = Menge aller Blätter im Teilbaum mit Wurzel  $v$

$d(v)$  = Tiefe des Knotens  $v$

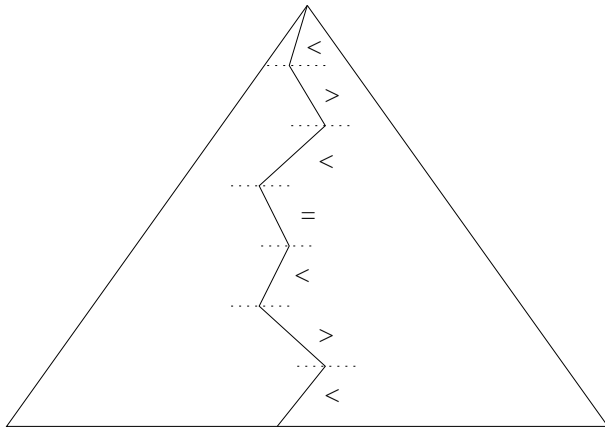
$Reg(v)$  = zu  $v$  gehörender „orthogonaler Bereich“, wobei  $Reg(Wurzel) = R^k$

$T$  heißt (perfekt) balanciert gdw.

$|S(x)| \leq \frac{|S(v)|}{2}$  für alle Knoten  $v$  und das „<-Kind und „>-Kind  $x$  von  $v$ .

## 2.5 Lemma

Die Höhe eines balancierten k-d-Baumes ist höchstens  $k + \log(n)$ .



Es gibt höchstens  $\log(n)$  „<“-Zeiger und „>“-Zeiger auf einem Weg durch den Baum. Es gibt höchstens  $k$  „=“-Zeiger auf einem Weg durch den Baum.