# Übungsblatt 5

Julius Auer, Alexa Schlegel

**Aufgabe 1** (Suchen in ebenen Unterteilungen):

Eine ebene Unterteilung ist eine Partition von $\mathbb{R}^2$ in durch Strecken und Strahlen begrenzte Gebiete (Einbettung eines planaren Graphen $G$)

Wenn man durch jeden Knoten der ebenen Unterteilung eine vertikale Linie zieht und sich dann den Schnittpunkt nach oben und unten merkt, dann entstehen ganz viele Trapze, die man dann wieder in zwei Dreiecke unterteilen kann. Was man dann davon hat keine Ahnung. Und welche Datenstruktur hier Sinn macht mit ($O(log(n))$ weiß ich auch nicht. Vielleicht ein Baum? So eine ebene Unterteilung wenns dumm läuft kann doch dann ganz ganz viele Trapeze haben oder?

Ich verstehe die Aufgabe nicht so richtig. Wir hatten in der VL auch das Thema Suche in Ebenen Unterteilungen, da haben wir aber triangulierte Unterteilungen betrachtet, wo die Außenfacette ein Dreieck ist, das Suchproblem dafür gelöst und dann einfach um es Allgmeiner zu machen, um ebene Unterteilungen die nicht so sind, ein ganz ganz großes Dreieck rundrum konstruiert und dann den Rest trianguliert.

* einfache Datenstruktur beschreiben zum Suchen mit Anfragezeit $O(\log n)$
* Vorverarbeitungszeit
* Speicherplatz für Datensturktur

**Aufgabe 2** ($L_1$-Voronoi-Diagramme):

In $L_1$-Metrik beschreiben die Punkte mit Abstand $d$ von einem Punkt ein Quadrat mit Seitenlänge $2 \cdot d$ (Abb. 1).

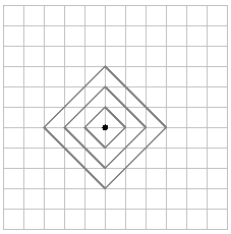

Abbildung 1: Punkte mit Abständen 1,2,3 von einem Punkt liegen auf den Ränder dieser Quadrate

Eine zwei Voronoi-Regionen trennende Kante hat nun stets eine von drei möglichen Formen, welche vom Verhältnis zwischen der X-Differenz und der Y-Differenz der Punkte bestimmt wird. Dominiert die X-Differenz, wird die Kante senkrecht "aussehen". Dominiert die Y-Differenz, wird die Kante waagerecht "aussehen".

Es sind im Folgenden die drei Fälle abgebildet, dass bei zwei Punkten die Y-Differenz (Abb. 2), die X-Differenz (Abb. 3) oder keine der beiden (Abb. 4) dominiert.

Jede Abbildung zeigt die drei untergeordneten Fälle, bei denen der in der dominanten Dimension größere Punkt in der rezesiven Dimension kleiner/gleich/größer dem anderen Punkt ist. Andere Fälle als die gezeigten sieben kann es nicht geben.

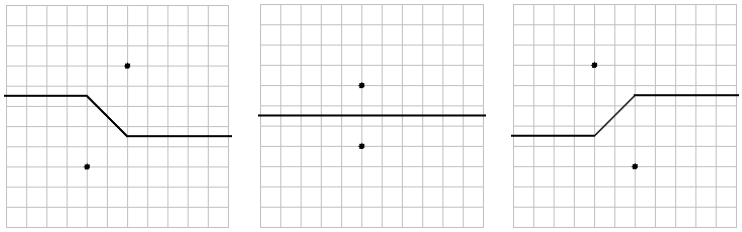

Abbildung 2: Fall 1: $\left|\frac{x_1-x_2}{y_1-y_2}\right| < 1$

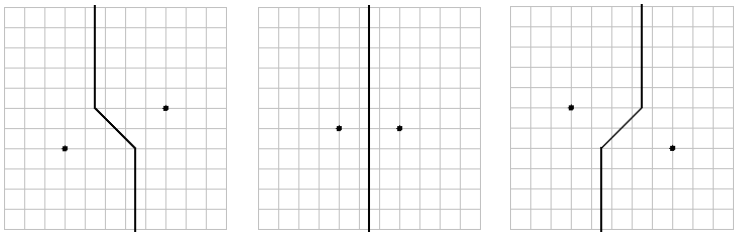

Abbildung 3: Fall 2: $\left|\frac{x_1-x_2}{y_1-y_2}\right| > 1$

Auffälig ist hier der letzte, entartete Fall, bei dem eine Menge von Punkten (rot) zu beiden Punkten equidistant ist und keine Kante sondern eine Fläche beschreibt. Wie dieser Fall im Kontext von Voronoi-Diagrammen behandelt werden sollte ist nicht aus deren Definition abzuleiten. In der Praxis sollte dieser Fall jedoch ohnehin nur relevant sein, wenn mit Ganzzahligen Koordinaten gerechnet wird.

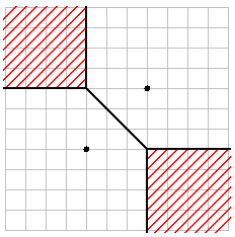

Abbildung 4: Fall 3: $\left|\frac{x_1-x_2}{y_1-y_2}\right| = 1$

**Aufgabe 3** (Suche in ebenen Unterteilungen - Verallgemeinerung):

Die Lokalisierungsdatenstruktur (LDS) wurde in der Vorlesung als gerichteter azyklischer Graph (DAG) beschrieben und funktioniert für ebene Unterteilungen, wo jede Facette ein Dreieck ist, auch die Außenfacette.

Wir werden nun im ersten Schritt aus der beliebigen Unterteilung der Ebene eine Triangulierung erzeugen einem Dreieck als Außenfacette. Im zweiten Schritt wenden wir darauf den in der Vorlsung vorgestellten Algorithmus zur Konstruktion der LDS an.

a) Um die vorhandene Unterteilung $U$ ein großes Dreieck $D$ legen, sodass alle Strecken im Inneren des Dreicks liegen.

b) Schnittpunkte von $D$ mit Strahlen der Unterteilung $U$ als Knoten in $U$aufnehmen.

c) Eckpunkte des Dreiecks auch in $U$ aufnehmen.

d) Inneres von $U$ Triangulieren $\rightarrow G$.

e) Algorithmus zur Konstruktion der LDS($G$) anwenden.

Folgendes wurde in der Vorlesung definiert:

- eingebetteter Graph $G$ entspricht $S_1$
- $S_h$ ist das äußere, umschließende Dreieck
- $h = O(\log n)$
- $I$ ist die unabhängige Knotenmenge

**Algorithm 1** Algorithmus zur Konstruktion der LDS($G$)

1: Sei $S_1 = G$
2: Für jedes $\triangle$ in $G$ erzeuge einen Knoten in LDS($G$)
3: i = 1
4: **while** $|S_i| > 3$ **do**
5: Berechne von $I$
6: Entferne $I$ aus $S_i$
7: Trianguliere $S_i \setminus I \rightarrow S_i + 1$
8: für jedes neue $\triangle$ $t$ in $S_i + 1$ erzeuge einen Knoten K($t$)in LDS($G$) und füge eine Kante von K($t$) zu allen Knoten von Dreiecken von $S_i$ die von $t$ geschnitten werden
9: i++
10: **end while**

**Algorithm 2** Lokalisiere ($q$, LDS($G$))

```
1: if q ∉ S_h then
2:    return q liegt in Außenfacette
3: end if
4: v=Wurzel(LDS(G))
5: while v hat Nachfolger do
6:    for all Nachfolger u von v do
7:       if q ist in Dreieck von u then
8:          v = u
9:       end if
10:   end for
11: end while
12: return q liegt im Dreieck von v
```

**Vorverarbeitungszeit** Ich schätze das Dreieck drumrumbauen dauert $O(n)$, weil man irgendwie das min-$x$, max-$x$ finden muss um das Dreieck zu konstruieren? Das Berechnen der Schnittpunkte kann man vernachlässigen.
$\rightarrow O(n)$

**Speicherbedarf** Es kommen nur konstant viele Knoten hinzu. 3 Knoten für das Dreieck und noch ein paar Schnittpunkte. Also bleibt es bei dem was in der Vorlesung gesagt wurde.
$\rightarrow O(n)$

**Anfragezeit** Bleibt gleich, oder?
$\rightarrow O(\log n)$