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):

Für zwei Punkte $p = (p_x, p_y)$ und $q = (q_x, q_y)$ ist die L_1-Metrik wie folgt definiert:
$d_1 = |p_x - q_x| + |p_y - q_y|$

Es lassen sich drei Fälle unterscheiden, je nach dem in welchem Verhältnis die Differenz der x bzw. y Werte der Punkte p und q zueinander stehen.

Fall ①: $|p_x - q_x| < |p_y - q_y|$
Fall ②: $|p_x - q_x| > |p_y - q_y|$
Fall ③: $|p_x - q_x| = |p_y - q_y|$

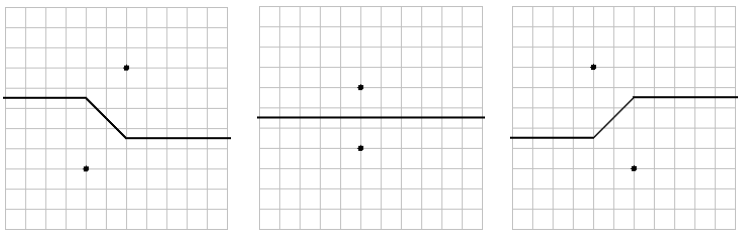

Abbildung 1: Fall 1: Bisektor verläuft in Richtung der x-Achse, horizontal

Aufgabe 3 (Suche in ebenen Unterteilungen - Verallgemeinerung):

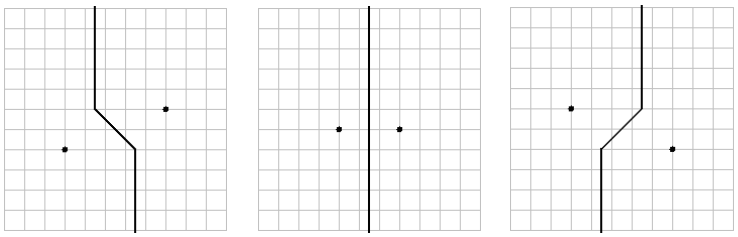

Abbildung 2: Fall 2: Bisektor verläuft in Richtung der y-Achse, vertikal

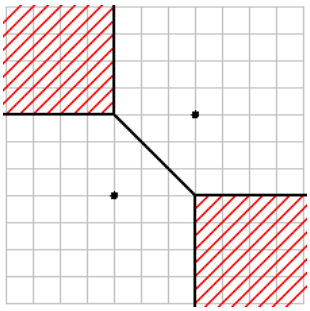

Abbildung 3: Fall 3: Bisektor besteht aus zwei unbeschränkten Bereichen und einer Strecke.

Die Lokalisierungsdatenstruktur (LDS) wurde in der Vorlesung als gerichteter azyklischer Graph (ADG) beschrieben und funktioniert für Triangulierung von G mit Dreieck als Außenfacette.

S_1 entspricht G, also der ebenen Unterteilung (eingebetteter Graph) S_n ist das äußere (umschließende) Dreieck

Ich habe es so verstanden dass in jedem Schritt die unabhänige Knotenmenge bestimmt und entfert wird, dass sind dann die S_i's' die zwischen 1 und n konstruiert werden.

Eine Knotenmenge I von G (Teilmenge) heißt unabhängig, wenn keine zwei Knoten aus I in G durch eine Kante verbungen sind. Es geht wohl in linearer Zeit zu berechnen, weil warum auch immer, wurde in VL bewiesen.

1. vorher das Ding Triangulieren
2. großes Dreieck drumrum bauen
3. wenn ein Strahl das äußerer Dreieck schneidet, dann dort einen neuen Knoten einfügen
4. Eckpunkte vom Dreieck mit Punkten verbinden oder einfach nochmal Triangulieren? oder voher schon Dreicke rumbasteln

h ... log(n)

* Erweiterung von LDS
* Algorithmen zur Suche und Konstruktion anpassen
* alle Unterteilungen den Ebene sollen unterstützt werden (mehrere unbeschränkte Facetten)
* Einzelheiten der Algorithmen beschreiben
*Vorverarbeitungszeit, Speicherbedarf, Anfragezeit, Anhängig von Anzahl der Knoten

Algorithm 1 Algorithmus zur Konstruktion

Sei $S_1 = G$
Für jedes $\triangle$ in G erzeuge einen Knoten in LDS(G)
i = 1
while $|S_i| > 3$ **do**
    Berechne I
    Entferne I aus S_i
    Trianguliere $S_i ohne I$, das ist dann $S_i + 1$
    für jedes neue $\triangle$ t in $S_i + 1$ erzeuge einen Knoten K(t)in LDS(G), füge eine Kante von K(t) zu allen Knoten von Dreiecken von S_i die von t geschnitten werden
    i++
end while

Algorithm 2 Lokalisiere (q, LDS(G))

teste ob $q \in S_h$ falls nein gib aussenfassette zurück
v=Wurzel(LDS(G))
while v hat Nachfolger **do**
    für jeden Nachfolger von u von v teste ob q im Dreiech u liegt.
    Falls ja, setze $v = u$ und wiederhole
    return q liegt im Dreieck von v
end while