

## Übungsblatt 11 – Quadrees (Lösung)

**Ausgabe:** 24. Juni 2014

**Abgabe:** 01. Juli 2014

### 1 Delaunay Triangulierungen - Meshing

Der in der Vorlesung vorgestellte Meshing-Algorithmus erzeugt nur nicht-stumpfe Dreiecke, d.h. Dreiecke die keinen Winkel haben der größer als  $90^\circ$  ist. Sei  $\mathcal{T}$  eine Triangulierung einer endlichen Punktmenge  $P \subset \mathbb{R}^2$  welche nur nicht-stumpfe Dreiecke enthält. Zeigen Sie, dass  $\mathcal{T}$  eine Delaunay Triangulierung von  $P$  ist.

**Lösung:**

Siehe Übungsfolien für eine Illustration. Beweis folgt nach Satz des Thales und Satz 5 der VL über Delaunay-Triangulierungen.

### 2 Komprimierte Quadrees

In dieser Aufgabe geht es um die Verringerung des Speicheraufwands für Quadrees. Seien dazu eine endlichen Menge  $P \subset \mathbb{R}^2$  von  $n$  Punkten sowie ein Quadtree  $\mathcal{Q}$  mit Tiefe  $d$  auf  $P$  gegeben. Ein gewöhnlicher Quadtree hat Speicheraufwand  $\mathcal{O}((d+1)n)$ . Es ist möglich diesen Aufwand auf  $\mathcal{O}(n)$  zu reduzieren in dem man jeden Knoten  $v$  entfernt, der nur einen Kindknoten hat in dem sich Knoten befinden. Dabei ändert man den Pointer vom Eltern-Knoten von  $v$  auf  $v$  so ab, dass er auf den einzig interessanten Kindknoten von  $v$  zeigt.

- Beweisen Sie, dass das beschriebene Verfahren tatsächlich den Speicheraufwand auf  $\mathcal{O}(n)$  reduziert.
- Wie ist die Laufzeit des Verfahrens, wenn man es auf einen Quadtree anwendet?
- Ist es auch möglich den Aufwand von  $\mathcal{O}((d+1)n)$  für die Konstruktion eines solchen Quadrees zu verringern?

**Lösung:**

Zu a): Sei  $m$  die Anzahl der Kanten, sei  $n_b$  die Anzahl der Blätter und sei  $n_i$  die Anzahl der inneren Knoten (ohne die Wurzel) in  $T'(P)$ . Zeige zunächst, dass  $2m \geq n_b + 3n_i$  gilt.

Jedes Blatt ist zu genau einer Kante inzident. Außerdem hat jeder innere Knoten mindestens zwei Kinder und damit mindestens Grad 3. Summiert man die Grade aller Knoten auf, so zählt man

jede Kante zweimal. Daraus folgt die Ungleichung  $2m \geq n_b + 3n_i$ . Außerdem gilt  $n = n_b + n_i + 1$  (das +1 wegen der Wurzel) und  $m = n - 1$ . Ersetzt man  $n$  in letzterer Formel erhält man  $m = n_b + n_i$ . Einsetzen in die erste Ungleichung ergibt  $2n_b + 2n_i \geq n_b + 3n_i$  und damit gilt  $n_b \geq n_i$ .

Da  $T'(P)$  gerade  $n_b = |P|$  Blätter enthält, folgt direkt  $|T'(P)| \in O(|P|)$ .

Zu b):  $\mathcal{O}((d+1)n)$ . Man muss jeden Knoten besuchen (maximal 4 mal).

Zu c): Offensichtliche Verfahren ex. nicht. Aber es ist möglich komprimierte Quadrees in  $\mathcal{O}(n \log n)$  zu konstruieren.

### 3 Balancierung

In der Vorlesung wurde als Balancierungs-Bedingung für Quadrees gefordert, dass adjazente Rechtecke sich höchstens um einen Faktor von zwei in ihrer Größe unterscheiden dürfen. Wir verschärfen nun diese Bedingung und fordern, dass alle Rechtecke exakt gleich groß sind. Wandeln wir nun einen Quadtree mit  $m$  Knoten in einen 'balancierten' Quadtree  $\mathcal{Q}$ , der obiger Bedingung genügt, um. Ist die Anzahl der Knoten in  $\mathcal{Q}$  dann immer noch linear in der Anzahl der Knoten des original Quadrees [ $\mathcal{O}(m)$ ]? Wenn das nicht der Fall ist kann man eine obere Schranke für die Anzahl der Knoten angeben?

**Lösung:**

Anzahl der Knoten nicht mer linear in  $m$ . Die Anzahl liegt nun in  $\mathcal{O}(4^d)$  (auf erster Stufe 1 Knoten, zweite Stufe 4 Knoten, dritte Stufe 16 Knoten usw.).

### 4 Range Queries mit Quadrees

Man kann Quadrees nutzen um Range Queries durchzuführen. Beschreibe einen möglichst effizienten Algorithmus der einen Quadtree  $\mathcal{Q}$  auf einer Punktmenge  $P$  nutzt um eine Query für eine Anfrage-Region  $R$  durchzuführen. Dabei sei der Quadtree  $\mathcal{Q}$  bereits gegeben.

- a) Analysieren Sie die worst-case Laufzeit für den Fall, dass  $R$  ein Rechteck mit Kanten Achsen-parallelen Kanten ist.
- b) Wie ändert sich die worst-case Laufzeit, wenn  $R$  eine Halbebene ist, die durch eine vertikale Gerade begrenzt ist.

**Lösung:**

Grundlegende Idee. Verwende das Prinzip der kd-Trees, das heißt: Wenn das Anfragerechteck alle Kinds-knoten eines inneren Knotens im Baum überdeckt, dann gebe alle Kinder aus. Schneidet das Anfragerechteck ein Blatt nur teilweise, dann prüfe ob der korrespondierende Knoten im Anfragerechteck liegt.

Zu a): Zunächst: Wenn der komplette Bereich es Quadrees in der Anfrageregion liegt, dann müssen alle Knoten besucht werden :  $\mathcal{O}((d+1)n)$

Nehmen wir jetzt an, dass wir in jedem Knoten des Quadrees die Knoten, die dieser Knoten abdeckt speichern [beachte: Der Speicherplatzverbrauch wird dadurch nicht erhöht]. Dann kann obige Anfrage in  $\mathcal{O}(n)$  beantwortet werden.

Eine Seite eines Rechtecks schneidet maximal  $2^d$  Knoten des Quadrees. Offensichtlich schneiden dann die vier Seiten des Rechtecks auch  $\mathcal{O}(2^d)$  Knoten des Quadrees. Das heißt der Gesamtaufwand liegt in  $\mathcal{O}(2^d + k)$ , wobei  $k$  die Anzahl der Knoten im Anfragerechteck ist.

Zu b): Ähnliche Argumentation liefert gleiches asymptotisches Ergebnis.