Karlsruher Institut für Technologie
Fakultät für Informatik
ITI Wagner

Algorithmische Geometrie
Sommersemester 2012
Martin Nöllenburg/Andreas Gemsa

Übungsblatt 10 – Quadtrees

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° ist. Sei $\mathcal{T}$ eine Triangulierung einer endlichen Punktemenge $P \subset \mathbb{R}^2$ welche nur nicht-stumpfe Dreiecke enthält. Zeige, 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 Quadtrees

In dieser Aufgabe geht es um die Verringerung des Speicheraufwands für Quadtrees. 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 Kinds-Knoten 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 Kinds-Knoten von v zeigt.

a) Beweise, dass das beschriebene Verfahren tatsächlich den Speicheraufwand auf $\mathcal{O}(n)$ reduziert.

b) Wie ist die Laufzeit des Verfahrens, wenn man es auf einen Quadtree anwendet?

c) Ist es auch möglich den Aufwand von $\mathcal{O}((d+1)n)$ für die Konstruktion eines solchen Quadtrees zu verringern?

Lösung:
Zu a): Jeder Knoten im Quadtree hat wenigstens zwei Kindsknoten. Das heißt auf der untersten Stufe gibt es n Knoten, dadrüber $n/2$, dadrüber $n/4$ Knoten, usw. Damit ergibt sich linearer Speicherplatzbedarf.

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 Quadtrees in $\mathcal{O}(n \log n)$ zu konstruieren.

3 Balancierung

In der Vorlesung wurde als Balancierungs-Bedinung für Quadtrees 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 Quadtrees $[\mathcal{O}(m)]$? 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 Quadtrees

Man kann Quadtrees 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) Analysiere die worst-case Laufzeit für den Fall, dass R ein Rechteck mit Kanten Achsenparallelen 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 Kindsknoten 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 Quadtrees 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 Quadtrees die Knoten, die dieser Knoten abdeckt speichern [beachte: Der Speicheplatzverbrauch wird dadurch nicht erhöht]. Dann kann obige Anfrage in $\mathcal{O}(n)$ beantwortet werden.

Eine Seite eines Rechtecks schneidet maximal 2^d Knoten des Quadtrees. Offensichtlich schneiden dann die vier Seiten des Rechtecks auch $\mathcal{O}(2^d)$ Knoten des Quadtrees. 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.