Ludwig-Maximilians-Universität München
Department "Institut für Informatik"
PD Dr. Peer Kröger
Gregor Jossé

München, 10.12.2013

Geo-Informationssysteme

WS 20013/14

Übungsblatt 5: Räumliche Anfragebearbeitung

Besprechung: 13.12.2013

Aufgabe 5-1 (Approximationen und topologische Prädikate)

Da komplexe Polygone in Geo-Datenbanken oft durch (achsenparallele) minimal umgebende Rechtecke (MUR) approximiert werden, ist es von Interesse, welche Aussagen man über Polygone aufgrund ihrer MUR machen kann. Welche Bedingungen müssen zwei MURs - falls möglich - erfüllen, damit die durch sie approximierten Polygone die topologischen Prädikate INSIDE bzw. DISJOINT (vgl. Skript)

(a) sicher erfüllen,

(b) sicher nicht erfüllen?

Aufgabe 5-2 (Nachste-Nachbarn-Anfrage / k-nächste-Nachbarn-Anfrage)

(a) Geben Sie an, wie die minimale Distanz zwischen einer Partition und einem Punkt berechnet werden kann (MinDist, Kap. 5, Folie 10). Gehen Sie dabei von euklidischer Distanz aus.

(b) Welche Datenstruktur eignet sich zur Verwaltung der PartitionList (Kap. 5, Folie 10)?

(c) Eine k-Nächste-Nachbarn-Anfrage liefert zu einem Anfrageobjekt o die k nächsten Nachbarn $p_1, \ldots, p_k$ aus der Datenbank zurück, sortiert nach aufsteigendem Abstand zum Objekt o.

Formulieren Sie einen Algorithmus für die k-Nächste-Nachbarn-Anfrage, indem Sie den Algorithmus für die Nächste-Nachbarn-Anfrage aus der Vorlesung (Kap. 5, Folie 11) geeignet erweitern bzw. ändern.

Lösungsvorschlag:

Algorithm 1 k-NN

Input: SpatialIndex über zu durchsuchenden Datensatz, Anfragepunkt QueryPoint

```
1: Initialisiere PartitionList mit den Root-Partitionen des SpatialIndex;
2: Sortiere die Einträge p der PartitionList nach MINDIST(p, QueryPoint);
3: KNNdist ← MAXREAL;                                                    ▷ ∞
4: KNN ← Liste mit fiktiven Elementen, dist = ∞: [(q_1, ∞), ..., (q_k, ∞)];
5: while PartitionList ≠ do
6:     Entferne erstes Element TopPart aus PartitionList;
7:     if TopPart ist ein Blatt des SpatialIndex then
8:         for all Entry ∈ TopPart do
9:             KNNC ← Entry.RETRIEVEEXACTGEOMETRY;
10:            if DIST(QueryPoint, KNNC) < KNNdist then
11:                Entferne letztes Element von KNN;
12:                Füge KNNC in KNN ein;
13:                sortiere Einträge x in KNN nach DIST(QueryPoint, x);
14:                KNNdist ← DIST(QueryPoint, grösstes Element von KNN);
15:            end if
16:        end for
17:        Entferne alle Elemente q aus der PartitionList für die gilt: MINDIST(q, QueryPoint) > KNNdist;
18:    else if TopPart liegt in einem inneren Knoten des SpatialIndex then
19:        ersetze TopPart durch seine Sohn-Partitionen;
20:        Sortiere die PartitionList erneut nach MINDIST(part,QueryPoint);
21:    end if;
22: end while;
23: return KNN;
```

Output: k-nächste Nachbarn von QueryPoint

Aufgabe 5-3 (Trapezzerlegung)

(a) Formulieren Sie einen einfachen Algorithmus zur Trapezzerlegung eines einfachen konvexen Polygons ohne Löcher. Funktioniert der gleiche Algorithmus auch für nicht konvexe Polygone und/oder Polygone mit Löchern? Wenn nein, welche Stellen im Polygon sind für den Algorithmus problematisch?

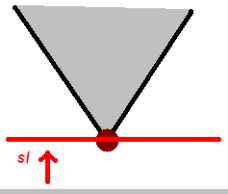
Hinweis:
Ein Polygon heißt konvex, wenn es gleich der konvexen Hülle seiner Eckpunkte ist.

(b) Formulieren Sie einen Plane-Sweep Algorithmus zur Trapezzerlegung eines einfachen Polygons mit Löchern. Welche Objekte werden im Event Point Schedule abgespeichert? Welche Objekte werden wann in den Sweep Line Status eingefügt oder gelöscht?

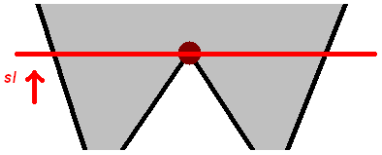
Lösungsvorschlag:

Event Point Schedule eps:	alle Eckpunkte
Sweep Line Status sls:	alle angefangenen Trapeze (untere Startpunkte, Verweise auf Strecken), sortiert nach Position auf der x-Achse

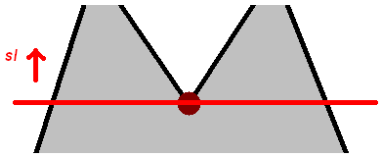
1. Sortiere alle Eckpunkte nach ihrem y-Wert ($O(n \log n)$) und initialisiere damit den eps.
2. Durchlaufe den eps. Interpretiere die Sweep Line sl als horizontalen Strahl und unterscheide folgende Fälle:

a) 

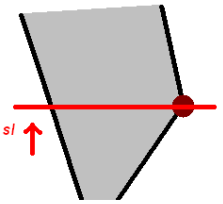
neues Trapez in sls einfügen

b) 

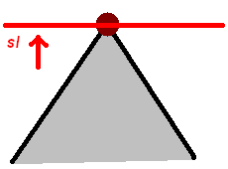
2 Trapeze abschließen und aus sls entfernen, 1 neues einfügen

c) 

1 Trapez abschliessen und löschen, 2 neue einfügen

d) 

1 abschliessen und löschen, 1 neues einfügen

e) 

1 Trapez abschliessen und aus sls entfernen

Diese Vorgehensweise funktioniert lediglich wenn keine zwei (unterschiedlichen) Punkte auf einer Höhe (y-Koordinate) liegen. Sonst wird die Behandlung weiterer Sonderfälle nötig.

Aufgabe 5-4 (Punkt-in-Polygon-Test)

Geben Sie einen Algorithmus für die erste Lösung des Punkt-in-Polygon-Tests (Jordansches Kurventheorem) an, der alle möglichen Sonderfälle berücksichtigt. Gehen Sie dabei von einem korrekt konstruierten einfachen Polygon mit Löchern aus.

Lösungsvorschlag:

Algorithm 2 Punkt-in-Polygon-Test

Input: Punkt Q, Polygon *Pol* über sein Kanten-Array.

```
1:  int s = 0;                                                          ▷ Zähler für Schnitte
2:  Ray g = von Q ausgehende Halbgerade nach rechts;                    ▷ Teststrahl für Q
3:  for i ∈ {1, ..., Kanten.LENGTH} do                                  ▷ Überprüfe für alle Kanten:
4:      if Kanten[i].CONTAINS(Q) then                                   ▷ Q liegt auf der Kante
5:          return true;
6:      else if ( !Kanten[i].ISHORIZONTAL ∧                             ▷ Kante nicht horizontal
            (S = Kanten[i].INTERSECT(g) ) ≠ null ∧                      ▷ Kante hat Schnittpunkt S mit g
            S.y ≠ MIN(Kanten[i].p1.y, Kanten[i].p2.y)) then             ▷ S ist nicht unterer Endpunkt der Kante
7:          s++;
8:      end if
9:  end for
10: if s.ISODD then                                                     ▷ Schnittzahl ungerade
11:     return true;
12: else                                                                ▷ Schnittzahl gerade
13:     return false;
14: end if
```

Output: Antwort ob Q *Pol* schneidet.

Betrachte Fallunterscheidungen auf Seite (Kap. 5 Folien 22 ff.). Bei Anwendung einer sich in horizontaler Richtung bewegenden Sweepline können diese verkürzt werden:

- ignoriere horizontale Kanten
- berücksichtige nur obere Kantenpunkte (untere gingen auch)

Somit wird in den ersten 3 Spezialfällen nur je 1 Schnitt pro Ereignis registriert, was korrekt einem Wechsel innen/aussen entspricht. Im letzen Fall registrieren wir entweder 0 (Strahl berührt Eckpunkt von unten) oder 2 Schnittpunkte (Strahl berührt Eckpunkt von oben) was keinen Wechsel zwischen Innen- und Aussenfläche nach sich zieht.