Algorithmische Anwendungen 2005/2006

Thema: Plane Sweep Technik

Dennis Scheffler 11029640
Gruppe D_lila

1 Einleitung

In dieser Ausarbeitung stelle ich eine leistungsfähige Technik für das Lösen von geometrischen Probleme vor. Es handelt sich dabei um die „Plane Sweep"-Technik, die erstmals 1978 von Shamos-Hoey eingesetzt wurde. Die Plane Sweep Technik ist Grundlage für viele leistungsfähige und recht einfachen Algorithmen für geometrische Probleme. In vielen Fällen sind die einzigen benötigten Datenstrukturen Bäume.

2 Das Finden der Schnittpunkte von Linien in der Ebene

Als konkrete und klassische Anwendung der Plane Sweep Technik, betrachten wir das Problem zum finden der Schnittpunkte von Linien. Wir haben eine Menge S = {L1,L2... L_n } von n-Linien in der Ebene.
Unsere Aufgabe ist es alle Paare (L_i , L_j), $i \neq j$ von Linien, die sich überschneiden zu berechnen. Die Brute-Force Methode zu diesem Problem betrachtet alle Paare (L_i , L_j), $i \neq j$ und überprüft bei jedem Paar, ob sich die zwei Linien überschneiden.
Das ergibt folgende worst-case Laufzeit:

Paare = $n\frac{(n-1)}{2}$

Durchlaufen aller Paare in einer Schleife und Überprüfung auf Überschneidung: $O(n^2)$
Da jedes Paar auf Überschneidung geprüft werden muss erhalten wir einen worst-case von $O(n^2)$

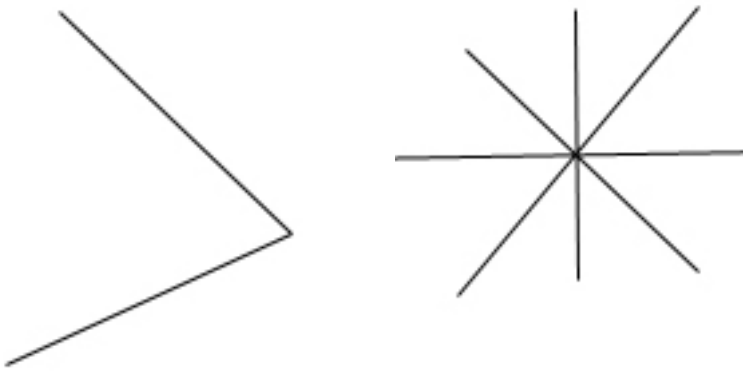
Ein Laufzeit bei diesem Algorithmus wäre dann optimal, wenn sich alle Linien überschneiden würden.

Ein Algorithmus dessen Laufzeit nicht nur von n abhängt sondern auch von der Anzahl der Überschneidungen ist mit der Plane Sweep Technik realisierbar. Den Algorithmus, den ich hier vorstellen möchte, wurde 1979 von Bentley und Ottmann erfunden. Seine Laufzeit beträgt O((n + k) log n), wobei k die Anzahl der Überschneidungen ist.

3 Der Algorithmus von Bentley-Ottmann

Ich lege folgende Vorraussetzungen fest:

1. Es gibt keine vertikalen Linien
2. Linien schneiden sich nicht an ihren Endpunkten
3. Nicht mehr als drei Linien schneiden sich an dem gleichen Punkt
4. Alle Endpunkte der Linien und alle Schnittpunkte haben unterschiedliche x-Koordinaten
5. Keine zwei Linien überlappen sich



Um alle Überschneidungen der Linien zu berechnen, verschieben wir die vertikale „Sweep“-Linie SL von links nach rechts und beginnen auf der linken Seite.

Während des Sweep Vorgangs werden die Linien in folgende Gruppen unterteilt:

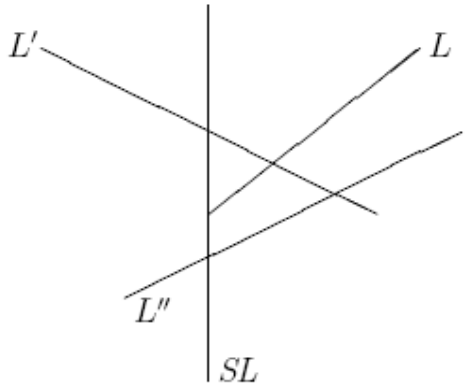
1. Tote Linien: dies sind die Linien, die vollständig auf der linken Seite von SL sind.
2. Aktive Linien: alle Linien, die gerade die SL schneiden.
3. Schlafende Linien: diese Linien befinden sich vollständig auf der rechten Seite von SL.

In einer Datenstruktur X sind die Endpunkte der einzelnen Lienen vor Beginn des Sweep-Vorganges gespeichert. Diese sind nach ihren x-Koordinaten aufsteigend sortiert. Während des Sweep-Vorgangs werden außerdem die gefundenen Schnittpunkte in X gespeichert. Die „aktiven Linien“ werden dabei in einer Datenstruktur Y gespeichert. Diese wird anhand der y-Koordinaten des Schnittpunktes zwischen der Linie und der Sweep-Linie sortiert.
Wandert die Sweep-Linie nun von links nach rechts und trifft auf einen Endpunkt oder einen Schnittpunkt werden folgende Ereignis ausgelöst.

Ereignis A
Die Sweep-Linie erreicht den linken Endpunkt einer Linie

A.1: Da Linie L aktiv wird, setzen wir es in die Datenstruktur Y ein.

A.2: Wir suchen nach der Linie L' in Y , welche direkt oberhalb von L ist (d.h., der Nachfolger von L). Nun suchen wir nach der Linie L" in Y, welche sofort unterhalb von L ist (d.h., der Vorgänger von L). Wir nehmen zur Einfachheit an, daß L' und L" existieren. Wir prüfen nun ob L und L' sich schneiden. Wenn sie dies tun, setzen wir den Schnittpunkt in die X-Struktur ein. Auch prüfen wir, ob L und L" sich überschneiden und setzen ihren Schnittpunkt gegebenenfalls in X ein.

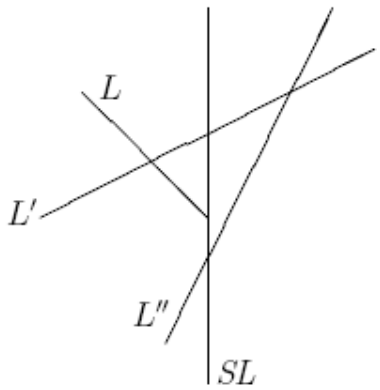


Ereignis B
Die Sweep-Linie erreicht den rechten Endpunkt einer Linie

B.1: Wir suchen in Y den Nachfolger L' und den Vorgänger L" von L. Wieder nehmen wir zur Einfachheit an, daß L' und L" existieren.

B.2: Wir löschen die Linie L aus der Datenstruktur Y.

B.3: Wir prüfen, ob L' und L" sich auf der rechten Seite der Sweep-Linie schneiden. Wenn sie dies tun, setzen wir ihren Schnittpunkt in die X ein.

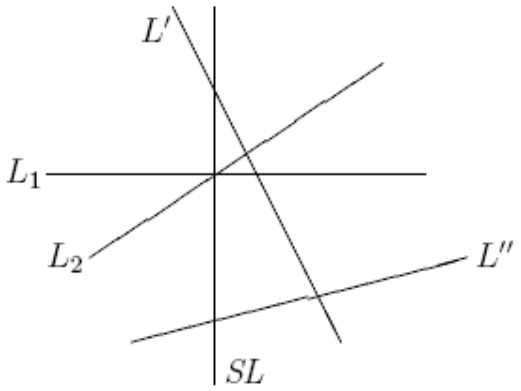


Ereignis C
Die Sweep-Linie trifft auf einen Schnittpunkt zweier Linien L1 und L2

C.1: Wir geben den Schnittpunkt (L1,L2) aus.

C.2: Wir suchen in Y den Nachfolger L' von L1 und den Vorgänger L" von L2. Wir nehmen zur Einfachheit an, daß L' und L" bestehen.

C.3: Wir ändern die Reihenfolge von L1 und L2 in Y. Falls L2 und L' sich auf der rechten Seite der Sweep-Linie schneiden, setzen wir ihren Schnittpunkt in die X ein. Falls L1 und L" sich auf der rechten Seite von SL schneiden, setzen wir ihren Durchschnitt in die Datenstruktur X ein.



Algorithm *Bentley Ottmann*(*S*)
initialisiere eine leere Datenstruktur *Y*;
sortiere die 2*n* Endpunkte aller Linien nach deren x-Koordinaten und
speichere sie in der Datenstruktur *X*;
(die Sweep-Linie *SL befindet sich an* Position $x = -\infty$)
while *X* $X \neq \emptyset$
do *min* = kleinste Datenstruktur in *X*;
lösche *min aus X*;
(*Die* Sweep-Linie *SL* ist nun auf position *x* = *min*)
if *min* ist linker Endpunkt einer Linie
then führe die Schritte in Ereignis A aus
endif;
if *min* ist rechter Endpunkt einer Linie
then führe die Schritte in Ereignis B aus
endif;
if *min* ist ein Schnittpunkt zweier Linien
then führe die Schritte in Ereignis C aus
endif
endwhile

4 Implementierung der Datenstrukturen X und Y als Bäume und die Laufzeit des Algorithmus

Bevor wir die Laufzeit des Algorithmus analysieren können, müssen wir einige Implementierungen spezifizieren. Wir nehmen für die Datenstruktur Y einen Rot-Schwarz Baum, in dem wir alle „aktiven Linien“ speichern, sortiert nach der y-Koordinate des Schnittpunktes der Linie L mit der Sweep-Line SL. Ein Linie L wird durch ihre zwei Endpunkte gegeben. Jede „aktive Linie“ wird mit ihren Endpunkten in einem Knoten des Baumes gespeichert.
Da es sich bei Y um einen Rot-Schwarz Baum handelt benötigen wir beim suchen, einfügen und löschen von Linien O(log n) Zeit. Auch das finden des Nachfolgers oder des Vorgängers eines Knotens kann in O(log n) realisiert werden.

Die Datenstruktur X enthält die Endpunkte einer Linie und die Schnittpunkte zwischen zwei Linien. Wir müssen Punkte einsetzen und löschen können. Außerdem müssen wir noch den Punkt mit der kleinsten x-Koordinate finden und löschen. Wir nehmen wieder einen Rot-Schwarz Baum, um diese Datenstruktur umzusetzen. Der Rot-Schwarz Baum wird höchstens 3n-1 Punkte enthalten und somit benötigt jede Operation O(log n) an Zeit.

Jetzt können wir die Laufzeit des Algorithmus Bentley Ottmann(S) bestimmen. Im Initialisierungsschritt sortieren wir die 2n Endpunkte der Linien anhand der x-Koordinaten und speichern sie im Rot-Schwarz Baum X. Dieses benötigt eine Zeit von O(n log n).

Lassen wir k die Anzahl der Schnittpunkte der Linien sein, dann benötigt die while-Schleife 2n+k Durchläufe. Bei jedem Schleifendurchlauf wird eine konstante Anzahl an Operationen bei den Rot-Schwarz-Bäumen X und Y ausgeführt. Also benötigt jeder Schleifendurchlauf O(log n) Zeit.

Daraus läßt sich folgende Laufzeit für den Algorithmus von Bentley und Ottmann errechnen.

$$O(n \log n) + (2n + k) * O(\log n)$$
$$=$$
$$O((n+k) \log n)$$

Quellen

Buch: M.T. Goodrich, R. Tamassia - Algorithm Design, John Wiley&Sons Inc.
Internet: http://www.geometryalgorithms.com/Archive/algorithm_0108/algorithm_0108.htm