Übungsblatt 2

Besprechung der Aufgaben am 20.05.

Präsenzaufgabe 1 (Line Segment Intersection):

a) Führe den Algorithmus FindIntersections aus der Vorlesung auf den Liniensegmenten in Abbildung 1 aus. Hierbei soll folgendes angegeben werden:

- Die Event-Queue zu Beginn vom Algorithmus, sowie nach jedem vollständig bearbeiteten Event. Benenne dabei, wie in der Vorlesung, Events resultierend aus Endpunkten von Liniensegmenten mit I_k und Events resultierend aus Schnittpunkten mit X_k. Markiere zusätzlich alle Events in Abbildung 1 und benenne sie wie oben beschrieben.
- Die auf Schnittpunkte geprüften Paare von Liniensegmenten während jedes Events.

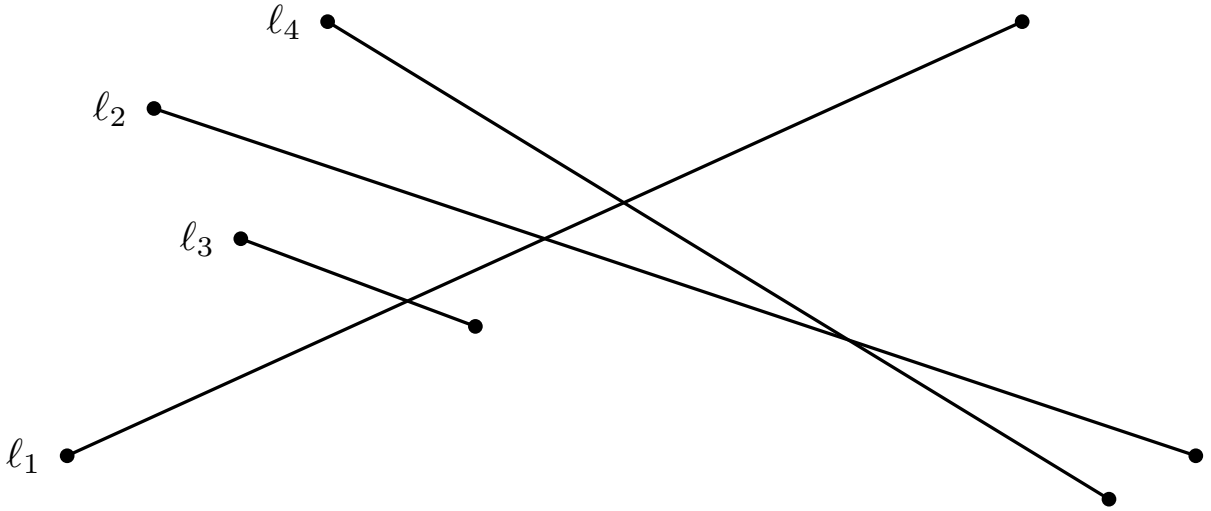

Abbildung 1

b) Gegeben sei eine Menge S von n Liniensegmenten in der Ebene. Anstatt alle Schnittpunkte zu berechnen möchten wir nur wissen, ob es zwei sich schneidende Liniensegmente in S gibt. Wie kann das Problem mit einer Laufzeit in $O(n \log n)$ gelöst werden?

Präsenzaufgabe 2 (Doubly-Connected Edge List):

a) Warum muss für Halbkanten in einer DCEL kein Zeiger auf den Zielknoten gespeichert werden?

b) Entscheide, ob die Aussage *immer* wahr ist. Falls nicht, gib ein Gegenbeispiel an.

   (i) $\vec{e}.\,\mathrm{Prev}().\,\mathrm{Next}() = \vec{e}$

   (ii) $\vec{e}.\,\mathrm{IncidentFace}() \neq \vec{e}.\,\mathrm{Twin}().\,\mathrm{IncidentFace}()$

   (iii) $\vec{e}.\,\mathrm{Twin}().\,\mathrm{Prev}().\,\mathrm{Twin}() = \vec{e}.\,\mathrm{Next}()$

   (iv) $\vec{e}.\,\mathrm{Prev}().\,\mathrm{IncidentFace}() = \vec{e}.\,\mathrm{Next}().\,\mathrm{IncidentFace}()$

Präsenzaufgabe 3 (Punkte und Dreiecke):
Betrachte folgendes Problem: Gegeben seien n disjunkte Dreiecke und n Punkte in der Ebene. Gesucht sind alle Punkte, die in *keinem* Dreieck liegen.

Entwickle einen *Plane-Sweep* Algorithmus, der das Problem mit einer Laufzeit in $O(n \log n)$ löst. Gehe dabei besonders auf die folgenden Punkte ein:

- Welche Eventtypen gibt es und wie werden sie behandelt?
- Welche Datenstrukturen sind notwendig und wie werden sie verwaltet?

Hinweis: Zur Lösung der Aufgabe darf das Prädikat Orientation(p, T) (mit konstantem Zeitverbrauch) verwendet werden. Hierbei ist p ein Punkt und T ein Dreieck, sodass eine vertikale Linie ℓ existiert mit $p \in \ell$ und $T \cap \ell \neq \emptyset$. Liegt p über (oder unter) allen Punkten in $T \cap \ell$, dann wird `above` (oder `below`) zurückgegeben, sonst `inside`.

In Abbildung 2 wäre zum Beispiel Orientation(p, T_1) = `below`, Orientation(p, T_2) = `inside` und Orientation(p, T_3) = `above`.

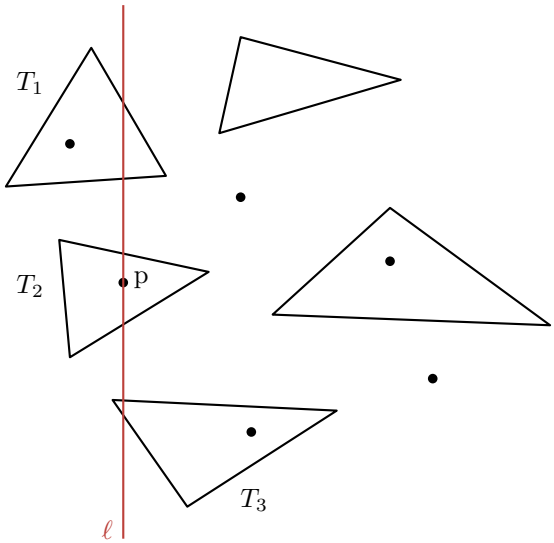

Abbildung 2