Praktikums Protokoll

Teilnehmer
Sebastian Stumpf
Felix Schramm

Fach	Computational Geometry
Abgabe	Praktikumsabgabe 3 – Streckenschnittpunkte mit Line Sweep Algorithmus
Datum	04.07.15

Aufgabenbeschreibung

Implementieren Sie unter Zuhilfenahme der Funktionalität aus Aufgabe 1 zur Berechnung von Schnittpunkten zwischen Linien einen Sweep Line Algorithmus und vergleichen Sie die erzielten Laufzeiten. Verwenden Sie für die Laufzeitvergleiche neben den Daten aus der ersten Aufgabe. Vergleichen Sie ebenso die Laufzeiten für die Files s_1000_1.dat und s_1000_10.dat (s.u.) .

Lösung

- Als Programmiersprache verwenden wir JAVA. Zur Evaluation Matlab.
- Wir bauen auf unseren Primitives aus Aufgabe1 auf (LineSegment). Die Erweiterung von LineSegment, Neighbor enthält zusätzlich noch die Logik den exakten Schnittpunkt 2er Linien zu berechnen (Slope-Intercept-Form, Gleichung lösen)
- Da der Line Sweep Algorithmus auf Probleme stößt, wenn Events zur gleichen Zeit (gleicher x-Wert) auftritt, geben wir diese Fälle als problematisch in der Konsole aus. Wir brechen die Berechnung jedoch nicht ab, sondern fahren trotz den Gefahr eines falschen Ergebnisses fort. Die meisten der als problematisch erkannten Events führen zu keiner Verfälschung des Ergebnisses.
- Wir implementieren den Line Sweep Algorithmus grob über die 4 Datenstrukturen NeighborList, EventList, IntersectionList und SweepLine.
    - NeighborList: lineare Liste, enthält die LineSegments
    - EventList: lineare Liste, enthält die Events
    - IntersectionList: lineare Liste aller Intersections
    - SweepLine: kombiniert obige Datenstrukturen für komfortablen Zugriff und Verwendung

- Zur Evaluierung der Schnittpunkte gibt es die Möglichkeit, sich ein Matlab-Skript generieren zu lassen, welches den berechneten Schnittpunkt von 2 LineSegments überprüft. Nachträglich wurde dieser Generator, leicht abgewandelt auch in Abgabe 1 eingefügt, um deren Ergebnisse zu überprüfen.
  Durch die Matlab Evaluierung ist sichergestellt, dass gefundene Ergebnisse auch korrekt sind.
  - Unabhängiger Test des gefundenen Schnittpunktes in Matlab
  - Fehlermeldung bei fehlgeschlagener Assertion
  - Plot der Ergebnisse zur Veranschaulichung

Ergebnisse

	Intersections	Time
s_1000_1.dat	4	0.264
s_10000_1.dat	725	4.084
s_100000_1.dat	77105	350.859
s_1000_10.dat	796	0.277

Das Ergebnis der Referenzdaten s_1000_10.dat, welches keine problematischen Linien mehr enthält haben wir mit dem Simplen Algorithmus aus Aufgabe1 und dem Matlab Evaluierungsscript verifiziert.

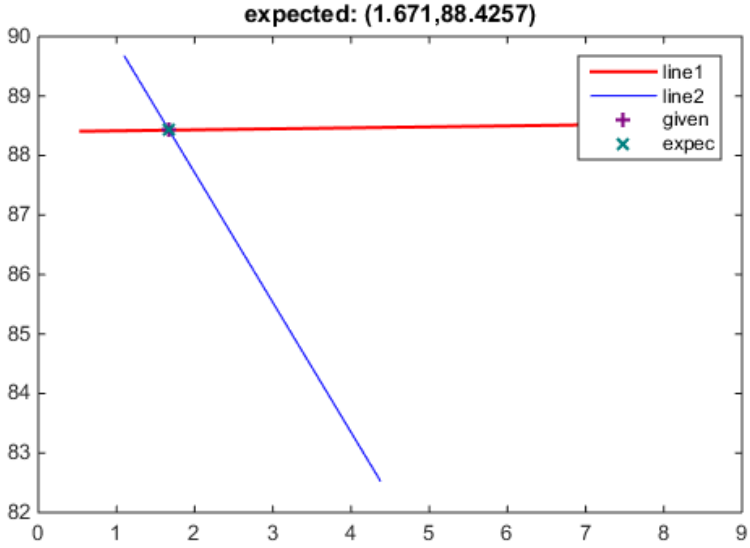
Die Ergebnisse der anderen Dateien haben wir mit Team Wurth/Weber abgeglichen.

Der eigentliche Laufzeit und Komplexitätsvorteil des Algorithmus kommt erst zum tragen, wenn die verwendeten Listenzugriffe O(n*log(n)) sind. Durch die Kapselung der Datenstrukturen dieser könnte man das noch relativ leicht ergänzen.

Die erhöhten Laufzeiten ggü. dem simplen Algorithmus lassen sich vermutlich durch die zusätzliche berechnung des exakten Schnittpunktes 2er Linien, sowie die komplexeren Datenstrukturen erklären.

Beispiele der Plots der Matlab Validation, erste 2 gefundenen Schnittpunkte in s_1000_10.dat:

[(0.545,88.407)|(8.4786,88.539)]X[(1.1,89.671)|(4.3714,82.5365)]
success
given: (1.671,88.4257)
expected: (1.671,88.4257)



[(0.545,88.407)|(8.4786,88.539)]X[(1.1,89.671)|(4.3714,82.5365)]
success
given: (1.671,88.4257)
expected: (1.671,88.4257)