# Computational Geometry Aufgabe 3

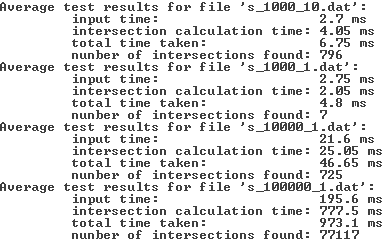
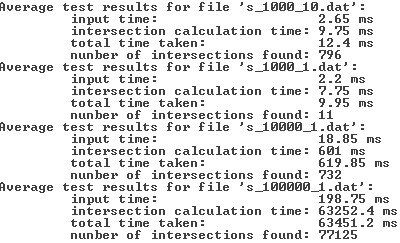
## Aufgabenstellung:

Ziel der Aufgabe ist es ein Programm zu schreiben, in welches gegebene Liniensegmente eingelesen werden können, welches dann die Anzahl an Schnittpunkten liefert. Hierbei soll ein Sweepline-Algorithmus zum Einsatz kommen.

## Vorgehensweise:

Wie in Aufgabe 1 wird zunächst die Datei mit den Liniensegmenten Zeilenweise eingelesen. Allerdings werden die einzelnen Koordinaten hier bereits für jedes Segment nach X-Koordinate sortiert. Dies erspart eine spätere Sortierung beim Erstellen der Start/Ende Events.  
Nach dem Einlesen werden Events für den Start und das Ende eines Liniensegments erstellt. Alle Events werden in X-Richtung sortiert. Die Datenstruktur dafür ist ein std::set, welches als Event Queue dient. Für die Sweepline-Struktur wird ein std::vector verwendet. Diese Struktur unterstützt den notwendigen „swap“ von zwei Elementen, jedoch leider nicht das Einfügen in O ( 1 ), sondern nur in O ( n ), allerdings ist die Sweepline in der Regel deutlich kleiner als die gesamte Anzahl an Liniensegmenten, weshalb dies nicht allzu sehr ins Gewicht fällt.

## Auswertung:

Es ergeben sich folgende Messergebnisse (20 Durchläufe, gemitteltes Resultat):   
Links ist das Ergebnis des Sweepline-Algorithmus zu sehen, rechts das des „naiven“ Ansatzes.  
Für die Dateien s\_1000\_1, s\_10000\_1 und s\_100000\_1 können über die Sweepline nicht alle Schnittpunkte ermittelt werden, denn bei den restlichen Schnittpunkten handelt es sich um Grenzfälle, welche in der hier Implementierten Lösung nicht behandelt wurden. Lediglich in s\_1000\_10 liefern beide das gleiche Ergebnis. Es ist anzunehmen, dass dort die Grenzfälle nicht vorhanden sind, welche bei der Sweepline zu Problemen führen (z.B. Vertikale Liniensegmente, Liniensegmente bestehend aus einem Punkt, …). Bei 100000 Liniensegmenten wird das Ausmaß des Zeit-Komplexitätsunterschiedes der beiden Algorithmen deutlich: Während mit der Sweepline die Schnittpunkte innerhalb von einer Sekunde gefunden werden können, so benötigt der „naive“ Ansatz schon über eine Minute.