Aufgabe 1

“In dem Tar-File strecken.tgz befinden sich Dateien mit jeweils 4 Koordinaten pro Zeile. Diese stellen jeweils die x- und y-Koordinaten eines Start- und Endpunktes einer Strecke dar. Lesen Sie die Datei Strecken.dat ein und ermitteln Sie die Anzahl der sich schneidenden (d.h. mindestens ein gemeinsamer Punkt) Strecken, indem Sie jedes Paar von Strecken gegeneinander testen. Messen Sie die pro Datei aufgewendete Zeit.”

Bei dieser Aufgabe gilt es über einen geeigneten Algorithmus herauszufinden, wieviele Schnittpunkte zwischen den Linien, die in den Files definiert sind, auftreten.

Der erste Schritt bildet hierbei das Errechnen des CCW (counterclockwise) Wertes.

CCW

Mit Hilfe des CCW Wertes ist es möglich zu errechnen ob ein Punkt rechts oder links eines Strecke liegt. Dies ist elementar für den Algorithmus für die Berechnung er Schnittpunkte. Liegt der Punkt links der Strecke so ist das aus den drei Punkten gebildete Dreieck gegen den Uhrzeigersinn orientiert (counterclockwise). Ist der Punkt rechts der Strecke so ist das gebildete Dreieck im Uhrzeigersinn orientiert (clockwise).

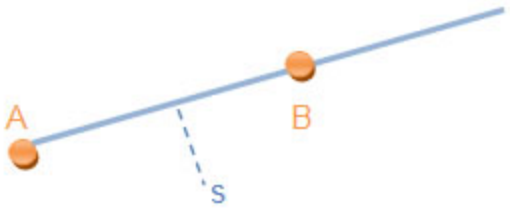

Abb. 1: Punkt neben einer Strecke

$$\mathrm{ccw}(p,q,r) := \begin{vmatrix} p_1 & p_2 & 1 \\ q_1 & q_2 & 1 \\ r_1 & r_2 & 1 \end{vmatrix} = \boxed{(p_1q_2 - p_2q_1) + (q_1r_2 - q_2r_1) + (p_2r_1 - p_1r_2)}$$

Schnittpunkt errechnen

Mit diesem Wissen als Grundlagen kann nun die Berechnung der Schnittpunkte beginnen. Das Prinzip ist das folgende:
Vergleicht man die eine Strecke mit einem Punkt der anderen Strecke und dann mit dem zweiten Punkt der Strecke so müssen, falsch ein Schnittpunkt vorhanden ist, die Punkte auf unterschiedlichen Seiten der Strecke liegen. Der erste Punkt muss also rechts der Strecke liegen und der andere Punkt links der Strecke.

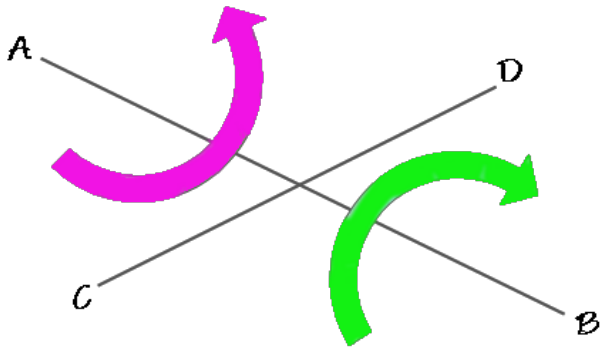

Abb 2: Orientierung der gebildeten Dreiecke

Somit weiß man, wenn für die beiden Punkte, gegen eine Strecke getestet, unterschiedliche Orientierungen errechnet werden, schneiden sich die beiden Strecken. Jedoch könnte es nun sein dass der Fall eintritt dass die Punkte zwar auf unterschiedlichen Seiten liegen, jedoch die Strecke so liegt dass kein Schnittpunkt zustanden kommt.

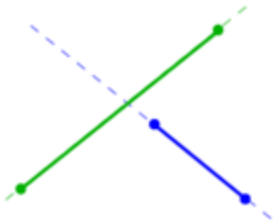

Abb. 3: Strecken ohne Schnittpunkt

Um dieses Problem zu lösen, wird nun die die andere Strecke gegen die anderen beiden Punkte getestet. Erhält man hier das selbe Ergebnis ist sicher gestellt dass sich die Strecken schneiden.

$$\mathrm{ccw}(p_1, p_2, q_1) \cdot \mathrm{ccw}(p_1, p_2, q_2) \leq 0$$
$$\wedge$$
$$\mathrm{ccw}(q_1, q_2, p_1) \cdot \mathrm{ccw}(q_1, q_2, p_2) \leq 0$$

Implementierung

1) Einlesen der Segmente in einen Vektor

```
vector<LineSegment*> lineSegments;
string line;
ifstream infile;
infile.open(file);
while (true) // To get you all the lines.
{
	getline(infile, line);
	if (line != "") {
		vector<string> lineParts = StringHelper::split(line, ' ');
		double p1 = atof(lineParts[0].c_str());
		double p2 = atof(lineParts[1].c_str());
		double p3 = atof(lineParts[2].c_str());
		double p4 = atof(lineParts[3].c_str());
		lineSegments.push_back(
				new LineSegment(Point(p1, p2), Point(p3, p4)));
	} else {
		break;
	}
}
```

2) Jedes Segment gegen jedes nachfolgende auf Schnittpunkt testen

```
LineSegment* segment = 0;
for (unsigned int i = 0; i < lineSegments.size(); i++) {
	segment = lineSegments.at(i);
	for (int current = interval_start; current < num_segments; current++) {
		LineSegment* segmentToCompare = lineSegments[current];
		if (segment->intersects(segmentToCompare)) {
			num_intersections++;
		}
	}
	interval_start++;
	progress.incrProgress();
}
```

3) Funktion: LineSegment.intersects()

(Siehe “CCW”)

```
bool LineSegment::intersects(LineSegment* other) {
	int test1 = ccw(&a, &b, other->getA()) * ccw(&a, &b, other->getB());
	int test2 = ccw(other->getA(), other->getB(), &a)
			* ccw(other->getA(), other->getB(), &b);
	return test1 <= 0 && test2 <= 0;
}
```

4) Ergebnis ausgeben

```
cout << "# of cuts between line segments: " << num_intersections << endl;
cout << "Total time needed: " << duration << "ms" << endl << endl;
```

Ergebnis

```
Starting benchmark for file [resources/Strecken_1000.txt] at 1405605781000
Read 1000 line segments from [resources/Strecken_1000.txt] at 1405605781000
# of cuts between line segments: 112117

Starting benchmark for file [resources/Strecken_10000.txt] at 1405605781000
Read 10000 line segments from [resources/Strecken_10000.txt] at 1405605781000
# of cuts between line segments: 11715703

Starting benchmark for file [resources/Strecken_100000.txt] at 1405605787000
Read 100000 line segments from [resources/Strecken_100000.txt] at 1405605788000
# of cuts between line segments: 1157481083
```