Welfenlab Competition
Schülerwettbewerb Informatik

http://www.gdv.uni-hannover.de/schools/competition05/

Worum geht es?

Nachdem wir uns in den letzten Jahren mit Knoten und ihren Eigenarten beschäftigt haben, widmen wir uns dieses Mal der Graphentheorie. Ein Graph ist dabei sehr theoretisch, man kann ihn jedoch durchaus zur Lösung ganz realer Probleme einsetzen. Dazu ein kleines Beispiel: Wir wollen in der Abbildung 1 links vom Punkt A den kürzesten Weg zum Punkt B bestimmen. Ungünstigerweise ist der direkte Weg versperrt, in diesem einfachen Beispiel ist einer der beiden kürzesten Wege jedoch problemlos durch hinschauen zu erkennen. In der Abbildung in der Mitte ist dies nicht mehr so einfach. Um gar auf einer Land- oder Stadtkarte den kürzesten Weg zu finden braucht man entweder viel Zeit oder einen Algorithmus, den auch ein Computer beherrscht. Um dem Computer unser Problem zugänglich zu machen und es gleichzeitig zu verallgemeinern, transformieren wir unsere Umgebung (hier eine Art Landkarte) in einen Graphen. Wir suchen also kürzeste Wege in einem Graphen.

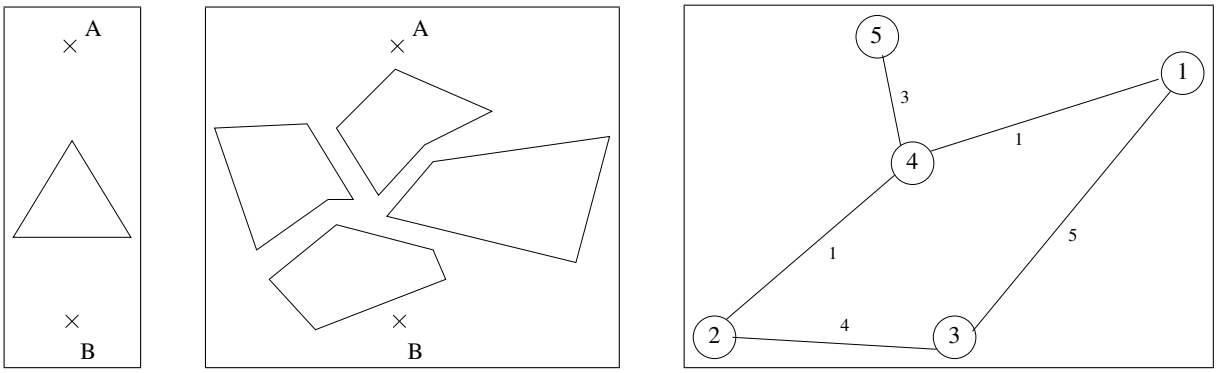

Abbildung 1: zwei relativ einfache Szenarien und ein Beispielgraph

Ziel der Competition ist es, ein Programm zu schreiben, das gewisse vorgegebene Szenarien in einen Graph transformiert, um in diesem Graph das Kürzeste-Wege-Problem mittels eines bereits bekannten Algorithmus zu lösen. Der kürzeste Weg im Graph entspricht dann dem kürzesten Weg im Szenario.

Und was gibt es zu gewinnen?

Unabhängig von viel Erfahrung, Ehre und Ruhm gibt es auch etwas Handfestes zu gewinnen:

1. Platz	2. Platz	3. Platz
		
Digicam Canon PowerShot Pro1 o.ä.	mp3-Player Apple iPod mini 6 GB o.ä.	Lautsprecher Logitech - X620 6.1 o.ä.

Was ist aber nun ein Graph rein mathematisch gesehen? Er besteht aus Ecken und Kanten, wobei eine Kante jeweils zwei Ecken verbindet. Ein gewichteter Graph ist ein Graph, bei dem jede Kante ein Gewicht oder auch sogenannte Kosten erhält. In dem Problem der kürzesten Wege ist es sinnvoll einem Ort eine Ecke zuzuordnen. Bei zwei Orten zwischen denen man eine direkte Verbindung hat, verbindet man die beiden die Orte repräsentierenden Ecken mit einer Kante. Als Gewicht erhält die Kante die Entfernung zwischen den beiden Orten. Als Konsequenz stecken alle für uns relevanten Informationen in den Kanten und ihren Gewichten. Die genaue Lage der Ecken (bzw. Orte) ist egal.

Das Gebiet der Graphen ist bereits gut erforscht und es gibt effiziente Algorithmen zur Bestimmung von kürzesten Wegen in Graphen. Daher beschäftigen wir uns mit der praktischen Komponente des Problems: Wie macht man aus einem realen Kürzeste-Wege-Problem einen Graph?

Hintergrund: Als Kürzeste-Wege-Problem dient uns ein Szenario, das einer Landkarte ähnelt. Wir haben einen Startpunkt und einen Zielpunkt. Weiterhin gibt es Hindernisse, die wir nicht überwinden können. Um ein solches Szenario zu speichern, benötigen wir eine Art Szenariodatei. In dieser Datei stehen alle nötigen Informationen, d.h. die Koordinaten der Start- und Zielpunkte und die Lage der Hindernisse, die bei uns durch geschlossene Polygone repräsentiert werden. (Ein geschlossenes Polygon ist eine Menge von Punkten bei denen jeder mit dem nächsten und der letzte mit dem ersten Punkt verbunden sind. Unsere Polygone haben keine Selbstüberschneidungen.) Um das Problem noch ein wenig komplexer zu machen, führen wir noch sogenannte Beamspots ein, an denen man sich zu einem anderen Spot beamen lassen kann. Beamspots treten immer gruppenweise auf (d.h. mindestens zwei), und innerhalb einer Gruppe kann man von jedem Spot zu jedem anderen reisen. Das folgende Beispiel zeigt eine solche Datei und das zugehörige Szenario.

```
A = 5.0 9.0                  (Startpunkt)
B = 5.0 1.0                  (Endpunkt)
P = 3.0 5.0;5.0 6.0;7.0 5.0 (dreieckiges Polygon als Hindernis)
P = ...                      (evtl. weitere Hindernisse)
BS = 1.0 9.0;9.0 1.0         (Gruppe von Beamspots)
BS = ...                     (evtl. weitere Beamspots)
```

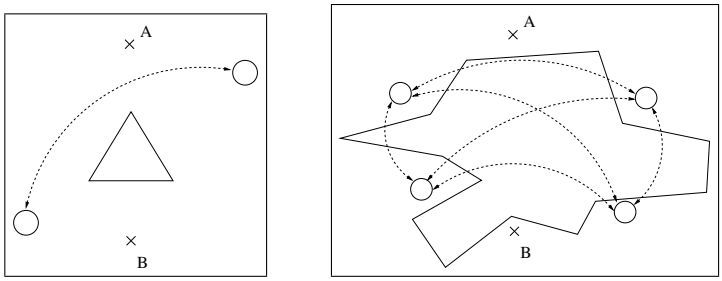

Abbildung 2: zwei denkbare Szenarien mit Beamspots

Selbstverständlich brauchen wir auch ein Dateiformat, um einen gewichteten Graphen abzuspeichern. Anhand des folgenden Beispiels sollte die Struktur klar sein. Die in der ersten Zeile angegebenen Ecken werden durchnummeriert beginnend bei 1.In der Datei wird im Weiteren die Nummer als Identifikation für die Ecke benutzt. Die Koordinaten der Ecken werden nur der Form halber mit abgespeichert.

```
5.0 9.0;5.0 1.0;3.0 5.0;5.0 6.0;7.0 5.0 (Koordinaten der Eckpunkte)
1 4 3.0              (Kante von 1. Ecke zu 4. Ecke mit Gewicht 3)
3 5 2.0              (Kante von 3. Ecke zu 5. Ecke mit Gewicht 2)
...                  (evtl. weitere Kanten)
```

Weitere Beispiele für Szenario- und Graphendateien gibt es im Internet:
http://www.gdv.uni-hannover.de/schools/competition05/

Aufgabenstellung
Es soll ein Programm entwickelt werden. In der dazugehörigen Dokumentation sollen Algorithmen und Programmaufbau verständlich dargestellt werden. Folgende Teilaufgaben sollen gelöst werden:

1. Überleg dir, wie man überprüft ob eine Strecke ein Polygon schneidet. Entwickle daraus einen Algorithmus. Als Eingabe erhält der Algorithmus ein geschlossenes Polygon und zwei Ecken (siehe Szenariodatei), und als Ausgabe erhält man die Antwort ob die direkte Verbindung der beiden Ecken das Polygon schneidet oder nicht. Dokumentiere deine Überlegungen und dein Vorgehen und beachte, dass der Algorithmus möglichst schnell/effizient ist, da er in der nächsten Teilaufgabe viel benutzt wird.

2. Entwickle einen Algorithmus der als Eingabe eine Szenariodatei erhält, diese einliest und als Ausgabe eine Graphendatei liefert und diese speichert. Dabei ist folgendes Vorgehen sinnvoll, damit sich ein Graph ergibt, der unser Szenario repräsentiert.

   - Start-, End- und jeder Eckpunkt eines Polygons wird zu einer Ecke im Graph.
   - Zwischen zwei Ecken gibt es genau dann eine Kante falls es im Szenario möglich ist eine direkte Verbindung zu ziehen, ohne ein Hindernis, d.h. ein Polygon zu schneiden. Die Kante erhält als Gewicht bzw. Kosten die Entfernung der Punkte im Szenario.

   Überlege dir und begründe warum dieses Vorgehen zweckmäßig ist.

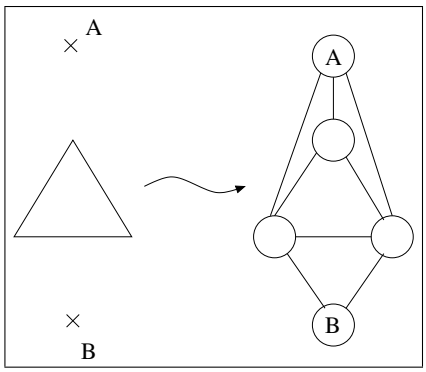

Abbildung 3: Transformation eines Szenarios in einen Graphen

   Der Algorithmus soll auch sehr komplexe Szenarien verarbeiten können, in denen eine Vielzahl von Polygonen als Hindernisse auftreten kann. Es ist daher entscheidend, einen effizienten, d.h. schnellen Algorithmus zu entwickeln. Um dies zu erreichen, können weitere Überlegungen zur Schnittproblematik gemacht werden. Entsprechende Erweiterungen des Algorithmus müssen verständlich dokumentiert werden.

   Die Beamspots sollen selbstverständlich ebenfalls berücksichtigt werden. Überleg dir wie man die Beamspots in den Graphen einbaut.

3. Es soll der kürzeste/billigste Weg in einem Graph gefunden werden. Da dieses Gebiet bereits gut erforscht ist wollen wir uns einiger Ergebnisse bedienen. Der Dijkstra-Algorithmus ist ein Algorithmus, der den kürzesten Weg in einem Graphen anschaulich konstruiert. Er ist relativ einfach, und es gibt sehr gute Bespiele im Internet (siehe Homepage).

Implementiere nun einen Algorithmus, der eine Graphendatei einliest und auf den Graph den Dijkstra-Algorithmus anwendet, um den kürzesten Weg von der Startecke (diejenige Ecke, die den Startpunkt des Szenarios im Graphen repräsentiert) zur Endecke zu bestimmen. Als Ergebnis soll der Algorithmus den genauen Weg (also eine Folge von Kanten im Graph) und die Länge des Weges ausgeben. Falls es mehrere gleichlange kürzeste Wege gibt, sollen alle ausgegeben werden.

4. Nachdem wir es geschafft haben den kürzesten Weg in sehr komplexen Szenarien zu bestimmen, wollen wir versuchen unsere Fragestellungen ein wenig zu verallgemeinern und fragen nun nach dem Weg vom Start- zum Zielpunkt mit minimaler Winkelsumme. Diese ergibt sich durch Aufaddieren aller beim Übergang von einer Kante zu einer anderen auftretenden Winkel, die sich durch die Richtungsänderung ergeben. Als Beispiel betrachte man folgende Abbildung rechts in denen zwei Wege (einmal als durchgezogene und einmal als gestrichelte Linie) eingezeichnet sind.

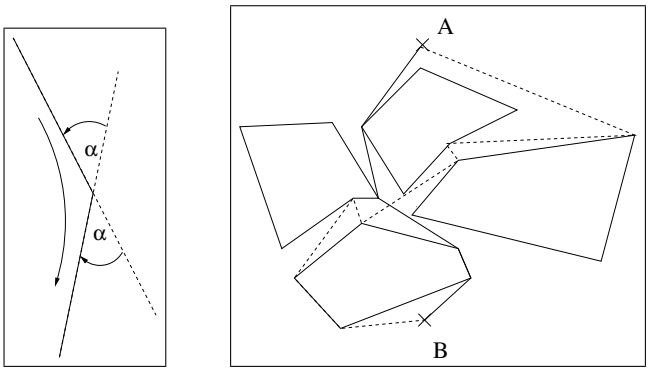

Abbildung 4: Messen der Winkel und Szenario mit zwei Wegen von A nach B

Der Weg mit der höheren Winkelsumme ist der zackigere Weg. Wir wollen also einen möglichst wenig zackigen Weg finden. Bei genauem Vergleich der Problemstellungen fällt auf, dass es sich wiederum um ein Minimierungsproblem in einem Graphen handelt. Es stellt sich die Frage, inwiefern unser bisheriges Vorgehen geeignet ist, um auch den Weg mit minimaler Winkelsumme zu bestimmen. Entwickle analog zum Vorgehen in den Teilaufgaben 1 bis 3 einen Algorithmus, der zu einer gegebenen Szenariodatei den Weg mit der geringsten Winkelsumme findet. Falls es mehrere gibt sollen selbstverständlich alle ausgegeben werden.

Benutzt man einen Beamspot, guckt man nach dem Beamen in die gleiche Richtung wie vorher, d.h. Beamspots verkürzen zwar den Weg enorm, die Winkelsumme aber eventuell nicht.

Vergleiche nun bei gleicher Szenariodatei die Ergebnisse des kürzesten Weges und der minimalen Winkelsumme. Was lässt sich sagen, falls Beamspots beteiligt sind? Was, wenn nicht? Halte deine Ergebnisse fest und begründe deine Schlussfolgerungen oder liefere Gegenbeispiele.

Teilnahmebedingungen

- Anmeldeschluss ist der 30.11.2005.
- Gruppenmeldungen sind nicht möglich.
- Als Programmiersprache sind Pascal, C, C++ und Java zugelassen. Die Sprache muss bei der Anmeldung mit angegeben werden. Es dürfen nur die Standard-Bibliotheken und keine Codegeneratoren verwendet werden.
- Es muss eine 5 bis 10-seitige Ausarbeitung angefertigt werden, in der die benutzten Algorithmen erklärt werden, und in der das Programm ausführlich dokumentiert wird.
- Das erstellte Programm und die Ausarbeitung sowie ein Ausdruck eines Testdurchlaufs bzw. die Ergebnisse mit von uns gestellten Daten sind spätestens bis zum 20.02.2006 bei uns einzureichen.
- Es dürfen nur Schüler der Sekundarstufe I oder II allgemeinbildender Schulen aus Niedersachsen an dem Wettbewerb teilnehmen. Familienangehörige von Mitarbeitern des Fachgebietes Graphische Datenverarbeitung an der Universität Hannover sind leider ausgeschlossen.
- Der Rechtsweg ist ausgeschlossen.

Bewertungskriterien

- Lauffähigkeit und Korrektheit des Programms
- Gut verständliche Dokumentation der Algorithmen ggf. mit Ergebnissen von Beispieldaten
- Besondere eigenständige Ideen
- In Zweifelsfällen: Strukturierter Programmierstil
- ...

Anmelden kannst Du Dich im Internet oder per Post mit dem beiliegenden Anmeldebogen. Solltest Du es dann nicht schaffen, Dein Programm rechtzeitig abzugeben, verfällt einfach Deine Anmeldung.

So, das war's erstmal. Hoffentlich hast Du ein wenig Lust bekommen, bei diesem Gewinnspiel mitzumachen. Es geht bei dieser Competition nicht darum, alles perfekt zu machen. Wir freuen uns auch über Teillösungen. Wenn wir merken, dass Du Dich mit der Problematik beschäftigt hast, hier und da ein paar interessante eigene Ideen hattest und Deine Gedanken und Algorithmen dokumentierst, hast Du gute Chancen unter den drei Besten zu sein. Bei Rückfragen kannst du dich gerne bei uns melden.
E-Mail: competition@gdv.uni-hannover.de

Viel Erfolg,
Hannes Thielhelm
Dipl.-Math. Dennis Allerkamp
Prof. Dr. F.-E. Wolter
Lehrstuhl Graphische Datenverarbeitung

Welfenlab Competition 2005
Schülerwettbewerb Informatik

Registrierung

Auch Online unter http://www.gdv.uni-hannover.de/schools/competition05/

Person: ☐ weiblich ☐ männlich

Vorname:	
Nachname:	
E-Mail:	

Adresse:

Straße:	
Postleitzahl:	
Stadt:	
Telefon:	

Informationen zur Person:

Alter:	
Schule:	

Programmiersprache, die für dieses Projekt benutzt wird
(Zutreffendes bitte ankreuzen):

☐ Pascal ☐ C ☐ C++ ☐ Java

Datum Unterschrift

Anmeldung an:
Universität Hannover - Institut für Mensch-Maschine Kommunikation - FG Graphische Datenverarbeitung
Welfengarten 1, 30167 Hannover
Tel.: 0511-762-2910 Fax.: 0511-762-2911