

# Welfenlab Competition 2004

## Schülerwettbewerb Informatik

<http://www.gdv.uni-hannover.de/schools/competition04/>

### Worum geht es?

Wegen der vielen interessanten Arbeiten und des großen Interesses am Thema Knoten der letztjährigen Welfenlab Competition wollen wir uns auch dieses Jahr wieder mit Aufgaben aus der Knotentheorie beschäftigen. Es sind natürlich keine Vorkenntnisse nötig. Sicher sind Dir schon häufig Knoten begegnet (z.B. Krawattenknoten, Segelknoten oder einfach beim Schuhbinden). Was genau sind aber nun eigentlich Knoten? Wir verstehen unter einem Knoten keinen Schuhknoten, bei dem die zwei freien Enden einfach entlang ihres Weges zurückgeführt werden können. Bei unserem Knoten gibt es keine freien Enden, er besteht aus einem einzigen Stück Band, das immer geschlossen ist (einige Beispiele sind in Abbildung 1 zu finden).

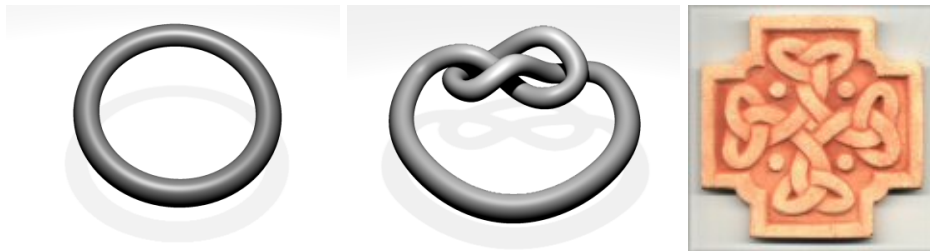


Abbildung 1: a) Unknoten b) Trefoil c) antike Zeichnung

Wenn man jetzt irgendwo an einem Teil des Knoten zieht, verändert das zwar sein Aussehen, aber nicht seine tatsächliche Verknotung. Mit anderen Worten: gleiche Knoten können sehr unterschiedlich aussehen (siehe Abbildung 2 auf der nächsten Seite). Es ist trotz vieler Forschung bis heute niemandem gelungen ein Programm zu schreiben, das zu zwei beliebigen gegebenen Knotendarstellungen feststellt, ob diese Darstellungen den gleichen Knoten repräsentieren. Bis zu einer gewissen Komplexität der Knoten ist dies aber immerhin mittels sogenannter Knoteninvarianten möglich. Hier kommst Du ins Spiel:

Ziel dieser Competition ist es, ein Programm zu schreiben, das Knoten einlesen kann. Außerdem soll Dein Programm für bestimmte Fälle feststellen können, ob zwei Knoten unterschiedlich sind. Doch mehr dazu später ...

### 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  |
|---|---|---|
|  |  |  |
| Pocket-PC<br>Toshiba PC e800 o.ä.   | mp3-Player<br>Philips HDD100 o.ä.   | DVD Brenner<br>Plextor PX-712A o.ä.   |



Abbildung 2: Trefoil unter verschiedenen Blickwinkeln

### Hintergrund

Die zu Anfang beschriebenen Knoten können natürlich beliebig komplex werden. Deshalb wollen wir Dir zunächst einige einfache Knoten vorgeben, an denen Du Dein Verfahren ausprobieren kannst. Die Knoten-Beispiele findest Du auf unseren Seiten im Internet:

<http://www.gdv.uni-hannover.de/schools/competition04/>

Natürlich sind unsere Knoten vereinfacht dargestellt, und zwar als sogenannte **Stock-Knoten**, die aus gradlinigen Strecken zusammengesetzt sind (siehe Abb. 3). In der Knotendatei werden immer nur die Anfangspunkte einer jeden Strecke, also die Ecken des Knotens, gespeichert.

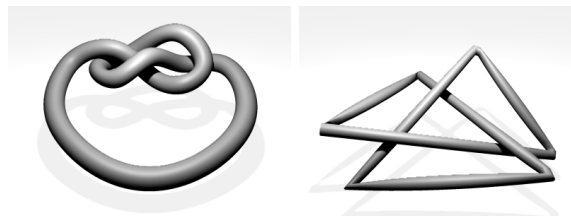


Abbildung 3: a) Trefoil b) Trefoil als Stock-Knoten

### Aufgabenstellung

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

1. Einlesen eines Stock-Knotens (geschlossenen Kantenzuges) aus einer Datei. Dabei stehen in jeder Zeile die drei Koordinaten  $(x,y,z)$  eines Eckpunktes. Z.B. :

```
12.222 -1.01 13.4
34 23.3 -17.93
...
```

Jeder dieser Punkte gibt eine Ecke des Kantenzuges an. Die Kanten sind die geradlinigen Verbindungen der Ecken. Vom letzten Punkt des Kantenzuges führt wieder eine Kante zum ersten, um den Kantenzug zu schließen. Die erste Ecke wird nicht nochmal am Ende der Datei angegeben. Es können beliebig viele Freizeichen oder Tabs zwischen zwei Zahlen vorkommen. Wichtig ist nur, dass pro Zeile nur drei Zahlen stehen. Nach dem Einlesen soll der 3D Stock-Knoten so in den 2D projiziert werden (z.B. durch Weglassen einer der drei Koordinaten), dass nicht mehrere Kreuzungen oder ganze Kanten aufeinander fallen. Da im 2D nicht mehr zu erkennen ist, welche Kante über welcher anderen Kante liegt, muss an jeder Kreuzung gespeichert werden, welche Kante die Überführung und welche die Unterführung ist. Man nennt so eine zweidimensionale Darstellung **Projektion** oder **Diagramm** eines Knotens (siehe Abb. 4). Natürlich kann man zu einem Knoten viele verschiedene Diagramme angeben. Es ist häufig nicht einfach, sie ineinander zu überführen.

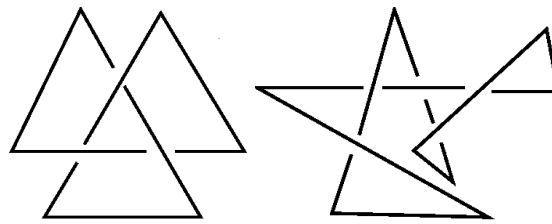


Abbildung 4: Projektionen vom a) Trefoil b) Unknoten

2. Um Knoten in einem einheitlichen Format auszugeben und einzulesen, wollen wir uns auf folgende **Kreuzungs-Notation** einigen: Wir wählen einen Startpunkt auf irgendeiner Kante und geben dieser Kante die Nummer 0. Dann verfolgen wir den Verlauf des Knotens bis zur ersten Kreuzung. Ab hier heisst die Kante 1, egal ob es eine Überführung oder Unterführung ist. Ab der nächsten Kreuzung wird die Zahl der Kante wieder erhöht, solange bis wir an der Startkante 0 ankommen. Schließlich schauen wir uns der Reihe nach die Kreuzungen an und geben für jede Kreuzung zwei Zahlen aus, nämlich zuerst die Kanten-Zahl an der Überführung von der aus wir den Knoten erreicht haben und dann die Kanten-Zahl an der Unterführung von der aus wir beim Knoten angekommen sind. Die wegführenden Kanten an jedem Knoten sind natürlich immer um genau 1 größer (außer natürlich bei der letzten Kreuzung, bei der der Nachfolger wieder 0 ist) und brauchen deshalb nicht angegeben werden.

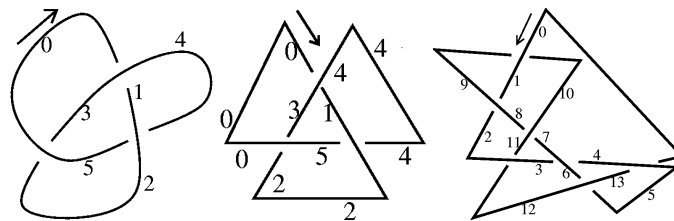


Abbildung 5: Beispiele zur Kreuzungsnotation

|            |                |   |   |    |   |    |
|------------|----------------|---|---|----|---|----|
| Beispiel 1 | Überführungen  | 3 | 1 | 5  |   |    |
|            | Unterführungen | 0 | 4 | 2  |   |    |
| Beispiel 2 | Überführungen  | 0 | 8 | 2  | 6 | 4  |
|            | Unterführungen | 9 | 1 | 11 | 3 | 13 |

Das Programm soll zu jedem gegebenen 3D Stock-Knoten diese Kreuzungs-Notation ausgeben. Außerdem soll es in der Lage sein, aus einer Datei Kreuzungs-Notationen zur weiteren Verarbeitung einzulesen. So eine Kreuzungs-Datei sieht ähnlich aus wie die Knoten-Datei, nur dass pro Zeile genau zwei ganze positive Zahlen stehen (nämlich die an einer Kreuzung ankommende Überführung und die ankommende Unterführung). Es fällt schnell auf, dass auf diese Weise immer eine gerade und eine ungerade Zahl gepaart werden (Gedankenanstoß: Warum eigentlich?).

3. Im weiteren Verlauf wollen wir uns nur noch mit Primknoten beschäftigen. Ein Primknoten ist ein Knoten der nicht aus zwei einfacheren Knoten zusammengesetzt ist. Wie findet man mit Hilfe unserer Kreuzungs-Notation heraus, ob ein Knoten ein Primknoten ist? Schreibe einen Algorithmus, der das nur anhand der Kreuzungs-Notation prüft.

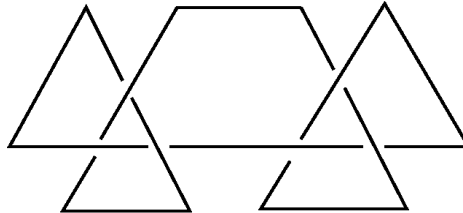


Abbildung 6: Kein Primknoten, da zusammengesetzt

4. Bei dieser Aufgabe wirst Du zum Maler, denn nun wollen wir loslegen und unsere Knoten einfärben. Dazu soll jede Kante mit einer von drei Farben (z.B. rot, grün und blau oder einfach 0,1,2) versehen werden. Man nennt das Tricolorisation. Allerdings muss eine gültige Färbung einigen Bedingungen genügen:
- (i) an jeder einzelnen Kreuzung muss die Überführung die Farbe beibehalten
  - (ii) an jeder einzelnen Kreuzung müssen entweder alle drei Farben oder nur eine einzige Farbe verwendet werden
  - (iii) der gesamte Knoten darf am Ende nicht nur in einer Farbe gefärbt sein.

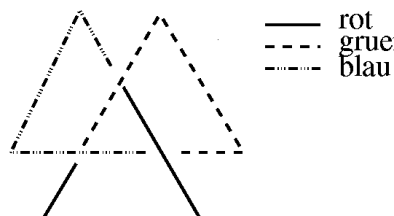


Abbildung 7: Eine von sechs Färbungen des Trefoil

Das Programm soll nicht nur prüfen, ob es eine solche Färbung gibt, sondern gleich mitschreiben, wieviele gültige Färbungen existieren. Es ist schnell klar, dass z.B. der Unknoten keine gültige Färbung besitzt. Zur Info: man kann zeigen, dass die Anzahl der Färbungen für einen Knoten immer gleich bleibt, egal mit welcher Projektion bzw. Diagramm man es gerade zu tun hat. Man nennt das dann einen Knoten-Invariante. Mit anderen Worten: Hat man bei zwei Diagrammen unterschiedliche Färbungs-Anzahlen, dann kann es sich nicht um denselben Knoten handeln! Der Trefoil von oben hat z.B. mehrere Färbungen, er kann daher also niemals entknotet werden, egal wie lange wir probieren. Es gibt also tatsächlich echte Knoten (die nicht der Unknoten sind).

## Teilnahmebedingungen

- Anmeldeschluss ist der 10.12.04.
- 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 21.02.05 bei uns einzureichen. Es ist möglich anstelle eines Ausdrucks die Dokumente auch als PDF mit dem Programm per Mail abzugeben.
- Es dürfen nur Schüler der Sekundarstufe I oder II an allgemeinbildenden Schulen sowie an Fachgymnasien aus Niedersachsen an dem Wettbewerb teilnehmen. Die Teilnehmer dürfen max. 19 Jahre alt sein. Familienangehörige von Mitarbeitern des Fachgebiets 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, an diesem Wettbewerb teilzunehmen. 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 beschäftigst, hier und da ein paar interessante eigene Ideen hattest und Deine Gedanken und Algorithmen dokumentierst, hast Du gute Chancen unter den Besten zu sein. Bei Rückfragen kannst Du Dich gerne bei uns melden.

E-Mail: [competition@gdv.uni-hannover.de](mailto:competition@gdv.uni-hannover.de)

Viel Erfolg,  
i.A. Dipl. - Math. Martin Reuter  
Lehrstuhl Graphische Datenverarbeitung  
Prof. Dr. F. - E. Wolter

# Welfenlab Competition 2004

## Schülerwettbewerb Informatik

### Registrierung

Auch Online unter <http://www.gdv.uni-hannover.de/schools/competition04/>

Person:

|                          |          |                          |          |
|--------------------------|----------|--------------------------|----------|
| <input type="checkbox"/> | weiblich | <input type="checkbox"/> | männlich |
|--------------------------|----------|--------------------------|----------|

|           |  |
|-----------|--|
| Vorname:  |  |
| Nachname: |  |
| E-Mail:   |  |

Adresse:

|            |  |        |  |
|------------|--|--------|--|
| Straße:    |  |        |  |
| PLZ Stadt: |  |        |  |
| Telefon:   |  | Alter: |  |

Informationen zur Schule:

|            |  |           |  |
|------------|--|-----------|--|
| Schule:    |  | Schultyp: |  |
| Straße:    |  |           |  |
| PLZ Stadt: |  |           |  |

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

|                          |        |                          |   |                          |     |                          |      |
|--------------------------|--------|--------------------------|---|--------------------------|-----|--------------------------|------|
| <input type="checkbox"/> | Pascal | <input type="checkbox"/> | C | <input type="checkbox"/> | C++ | <input type="checkbox"/> | Java |
|--------------------------|--------|--------------------------|---|--------------------------|-----|--------------------------|------|

---

Datum

Unterschrift

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