”Über chinesische Postboten, Routenplaner und eingefärbte Landkarten“ – Eine Einführung in die Graphentheorie

Karima Benimmar, Tina Schmidt, Daniel Jungblut

Wie organisiert man Einbahnstraßensysteme sinnvoll? Ist es möglich über jede der sieben Königsberger Brücken genau einmal zu gehen und wieder zu Hause anzukommen? Warum ist in einer geschlossenen Gesellschaft die Anzahl der Personen, die einer ungeraden Anzahl an anderen Gästen die Hand schüttelt, immer gerade?

Diese und viele weitere realitätsnahe Probleme können mit Hilfe von graphentheoretischen Ansätzen mathematisch untersucht werden. Ein Graph besteht aus Knoten und Kanten, die diese Knoten miteinander verbinden. Beispielsweise werden bei der Routenplanung Städte durch Knoten und Straßen durch Kanten dargestellt. Beim „Königsberger Brückenproblem“ hingegen interpretiert man die Stadtteile als Knoten und die Brücken als Kanten. Mit Hilfe der Mathematik lassen sich zentrale Aussagen über Graphen beweisen, so dass diese praktisch motivierten Probleme gelöst werden können. Viele Algorithmen (genaue Rechenvorschriften) aus der Graphentheorie finden in unserem täglichen Leben Anwendung, wie zum Beispiel bei Routenplanern oder beim Erstellen von Stundenplänen. Einige dieser Algorithmen können leicht programmiert werden, was in diesem Kurs auch nachvollzogen wurde. Andere Probleme sind aber bis heute ungelöst und Bestandteil aktueller Forschung.

An dieser Stelle möchten wir uns bei unseren Teilnehmern recht herzlich bedanken für zwei wunderschöne Wochen Sommerakademie. Der dank unserer chinesischen Teilnehmer auf Englisch statt gefundene Kurs wird uns sicherlich lange in Erinnerung bleiben. Wir hoffen die Teilnehmer mit unserer Faszination für Mathematik angesteckt zu haben und würden uns freuen, wenn dieser Kurs sie dazu inspiriert hat, sich weiterhin mit Mathematik zu beschäftigen.

Vorbereitungswochenende

Maren Beck

Vom 13. Juni bis zum 15. Juni fand das Vorbereitungswochenende der Science Academy 2008 in Adelsheim statt.

Wie der Titel schon sagt, haben wir uns im Kurs „Graph Theory“ (Dt.: Graphentheorie) auf die Sommerakademie vorbereitet. Wir haben zunächst verschiedene Beweisverfahren kennen gelernt, die für die Sommerakademie nützlich waren:

- Direkter Beweis
- Indirekter Beweis
- Widerspruchsbeweis
- Äquivalenzbeweis
- Zirkelschluss
- Vollständige Induktion

Die chinesischen Gastschüler waren zwar noch nicht da, doch der Kurs fand bereits die meiste Zeit auf Englisch statt. Um im Sommer graphentheoretische Algorithmen programmieren zu können, bekamen wir beim Vorbereitungswochenende eine kleine Einweisung in die Programmiersprache C++. In der Zeit bis zur Sommerakademie erhielten wir im Abstand von 14 Tagen Arbeitsblätter. Diese beinhalteten jeweils Aufgaben zum Programmieren und den gelernten Beweistechniken.

Die Atmosphäre im Kurs

Maren Beck

Der Unterricht

Der Kurs wurde wegen unseren chinesischen Gastschülern auf Englisch gehalten. Am Anfang war es eine Umstellung, doch mit der Zeit wurde es ganz normal. Wir waren uns einig, die Präsentationen (Rotation, Abschluss) auf Englisch zu halten, denn im Deutschen hätten uns manchmal einfach die Worte gefehlt. Auch die Dokumentation wurde großteils auf Englisch verfasst. Einige Sachen wurden auch auf Deutsch erklärt, aber der restliche Kurs war auf Englisch. Allgemein war die Unterrichtsatmosphäre richtig gut und wir hatten auch oft viel zu Lachen.

Spaß im Kursraum.

Vor der Rotation und der Abschlusspräsentation war es dann ein bisschen anstrengender, da alles rechtzeitig fertig werden musste. Doch trotz aller Aufregung hat es gut geklappt.

Unsere chinesischen Gäste

Wir, die deutschen Teilnehmer, kannten uns schon vom Vorbereitungswochenende. Doch in der Sommerakademie kamen noch unsere chinesischen Freunde Stella, Blues, Jack und Susan dazu.

Unsere chinesischen Gastschüler bei der Arbeit.

Trotz der verschiedenen Kulturen fanden wir sehr schnell zusammen und unterhielten uns viel auf Englisch. Am Ende merkte man überhaupt nicht mehr, dass sie aus einem ganz anderen Land kommen.

Auch beim Ausflug nach Heidelberg, was natürlich für die Chinesen eine ganz besondere Erfahrung war, zeigten sie große Begeisterung. Ich glaube, wir werden unsere gemeinsame Zeit bei der Science Academy nie vergessen.

Wir – Der Graphentheoriekurs!

Lisa Bitterich

Die Leiter:

Sind alle drei ständig am Kekse Essen.

Daniel Jungblut – Unser Chef

Sein Unterricht war anspruchsvoll, trotzdem war er immer für einen Spaß zu haben. Selbst wenn wir uns über sein Tafelbild lustig machten, blieb er gut gelaunt. Hin und wieder vergaß er mal einen Buchstaben oder ein Wort, worauf wir ihn gerne hinwiesen. Dafür war er im Programmieren unschlagbar!

Tina Schmidt – Unser Englischwörterbuch

Wenn sie gerade nicht selbst an der Tafel stand, schrieb sie immer auf Plakaten mit, die sie dann später an die Wand hängte. Selbst wenn wir einen Beweis oder ein Problem nach dem dritten Mal Erklären nicht verstanden hatten, blieb sie verständnisvoll. Dass sie die Lage voll im Griff hatte, bewies sie nicht nur im Kurs, sondern auch als (unsere Graphentheorie-Lieblings-)Nachtaufsicht.

Karima Benimmar – Unsere Fotografin

Auch sie schrieb manchmal zu viele oder zu wenige Wörter an die Tafel. Sie blieb immer lustig und nett und was wir ihr ganz hoch anrechnen: Am Ausflugstag ging es ihr gar nicht

gut, sie fuhr aber trotzdem mit uns nach Heidelberg.

Die Teilnehmer

Maren Beck – Unsere Fußballerin

„Daniel, können wir jetzt Pause machen? Ich will wissen, wie Stuttgart gespielt hat.“ Ja, das ist die Maren. Immer voll dabei, wenn es um Sport geht, selbst wenn ihr Knie dick davon wird. An Diskussionen beteiligte sie sich gerne, aber lieber auf Deutsch als auf Englisch.

Marvin Becker – Unser Gerne-Lacher

Dass er ein ziemlich lustiger Typ ist, merkten wir schnell, da er gerne lachte oder zumindest vor sich hin grinste. In der Gemeinschaft half er viel mit, und selbst den Müll brachte er gerne raus.

Hanna Binder
– Unser Gute-Laune-Mensch
Sie lacht viel, gerne, oft und über alles. Nur über Bayern München kann sie nicht lachen, da sie ein eingefleischter VfB-Fan ist.

Lisa Bitterich
– Unsere Chinesenunterhalterin
In einer Gruppenarbeit versuchte sie uns so gut wie möglich ans Ziel zu führen. Außerdem achtete sie darauf, dass auch die Chinesen einbezogen wurden und so viel wie möglich Englisch gesprochen wurde. Aber nicht nur bei der Gruppenarbeit war sie eine große Hilfe. Sie versuchte uns so gut es ging zu helfen, wenn wir Fragen hatten und schenkte jedem ein Lächeln, der sie anschaute.

Felix Dörre – Unser Programmiergenie
Er ist ein richtiger Computermensch und hatte auch schon Vorkenntnisse im Programmieren. Trotzdem half er den „weniger Computerbegabten“ gerne und bewies dabei, dass er sehr gut erklären kann.

Tobias Hocke
– Unser Colaflaschen- bzw. Laptopmann
Er ist extrem computerbegeistert und war auch für alles zuständig, was irgendwie mit Multi-Media zu tun hatte. Trotz seiner guten Vorkenntnisse brauchte er hin und wieder eine Gemüsesuppen- oder Kabapause.

Nico Modenese – Unser Harry Potter
Er war im Kurs eher ruhig, sobald er aber etwas erklärte, machte er es richtig gut. Er blieb immer freundlich und ging an die Aufgaben sehr ernst heran.

Marcel Schnirring – Unser Bayern-Fan
Mit ihm war der Kurs immer lustig. Beim Programmieren war er immer ganz vorne dabei, deshalb war es auch besser, wenn man ihn dabei nicht störte.

Carmen Streibel
– Unsere Colatrinkerin auf Kaffeeentzug
Sie lachte viel und gerne und manchmal konnte sie gar nicht mehr aufhören. Im Kurs hatte sie viele Ideen und arbeitete aktiv mit, nur Programmieren mochte sie nicht so gerne. Wenn ein Großteil von uns in den Computerraum ging, beschäftigte sie sich lieber weiter mit Mathematik.

Julia Werle – Unsere Blockanmalerin
Sie war auch eine von denen, die gerne Joggen gingen, wovon sie Muskelkater in den Armen bekam!? Während der Akademie war sie oft am Lachen und immer hilfsbereit.

Die Chinesen:

Waren alle vier total nett.

Stella – Unser Shoppingfreak
Sie ist eine ganz Liebe und Nette. Dass sie richtig gut Englisch kann, bewies sie uns bei den Präsentationen, in denen sie ihren Part immer verständlich erklärte. Nur die Absprache, wann Blues die Folien weiterschalten sollte, funktionierte nicht so gut.

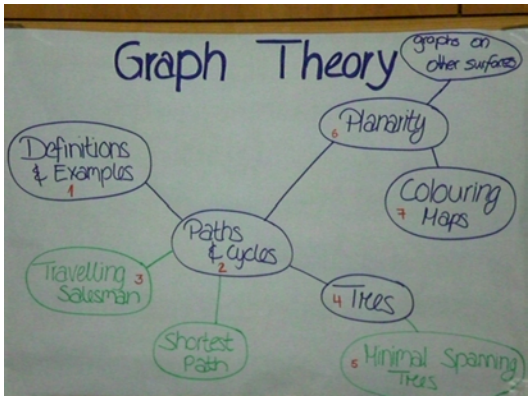
Susan – Unsere „Peace“-Wünscherin
Sie ist auch sehr sportlich (Tanzen, Kung Fu). Wenn man mit ihr redete, konnte man sicher sein, dass sie irgendwann „Oooh“ sagen würde, außerdem lachte sie sehr viel und war ständig am Fotografieren.

Blues – Unser Sänger
Er war der einzige Chinese, der mit uns programmierte. Außerdem kann er ein bisschen Pen-Spinning und lacht sehr gerne. „And don’t forget my elephant.“

Jack – Unser Süßigkeitenkäufer
Alarmanlagenauslöser, 5-fache Nachtischration-Bediener, Coole-Aussprache-Haber, Inden- unpassendsten- Momenten- aufs- Klo-Müsser. Er war immer für einen Lacher gut.

Basics

Carmen Streibel



Course contents represented as a graph.

In our course, we used graphs to solve abstract as well as practical problems.

What is a Graph?

A graph consists of points, called *vertices* (singular: a vertex) and lines, called *edges*. When there is more than one edge connecting two vertices, we call them *multiple edges*. An edge can also connect a vertex with itself, which is called a *loop*.

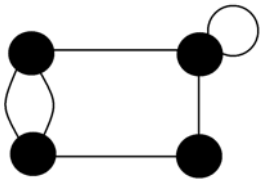


Figure 1: A graph with a multiple edge (left) and a loop (right).

Two vertices which are connected by an edge are called adjacent. An edge which starts or ends at a vertex is called incident to that vertex. The total number of incident edges of a vertex is called its *degree*. For example, a vertex which is incident to three edges has degree three; see figure 2 for an example. Two adjacent vertices are called neighbours.

An important basic result of graph theory is the Handshaking Lemma: In any graph, the sum of all vertex-degrees is an even number, namely twice the number of edges. This was proved by Leonhard Euler in 1736.

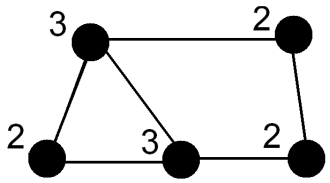


Figure 2: Vertex degrees.

Different Types of Graphs:

A graph is either *connected* or *disconnected*. A disconnected graph consists of at least two disjoint graphs, each of which is called a component. The vertices of different components are not connected by any edge. A connected graph cannot be expressed as the union of disjoint graphs.

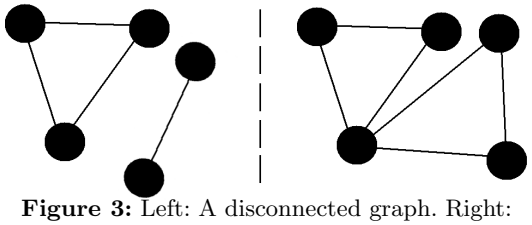


Figure 3: Left: A disconnected graph. Right: A connected graph.

A graph is called simple, if it contains no multiple edges and no loops. Otherwise, the graph is called non-simple.

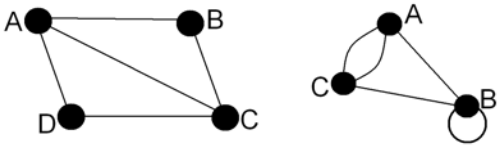


Figure 4: Left: A simple graph. Right: A non-simple graph.

Special Graphs

In a *complete graph*, each vertex is connected to every other vertex. K_n is used to denote the complete graph on n vertices.

In a *cycle graph*, all vertices have degree two. You can draw it like a real cycle, but also different as shown in figure 6:

In a *bipartite graph*, the set of vertices can be split into two subsets. To denote these two subsets, the vertices are coloured black and white. An edge of a bipartite graph always

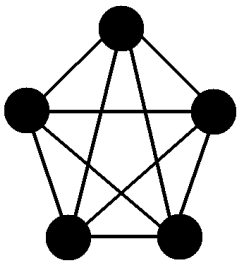


Figure 5: The complete graph on five vertices.

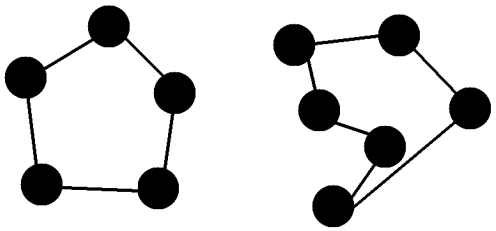


Figure 6: Different cycle graphs.

connects a black and a white vertex. There are no edges between white vertices and none between black vertices.

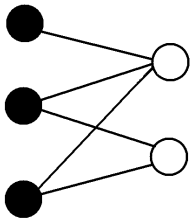


Figure 7: A bipartite graph.

In a *complete bipartite graph*, each white vertex is adjacent to each black vertex and vice versa. This graph is denoted by $K_{n,m}$. n and m denote the number of black and white vertices respectively.

In a *weighted graph*, each edge has an associated number, called its weight. Weighted graphs can be used to model practical problems, for example: To find the shortest way between two cities, a graph's vertices can stand for the cities and the weights on its edges denote the distances between them.

The vertices can also refer to subway stations and the weights to the price of the ticket between those stations. Then, the aim would be to find the cheapest way from one station to another one.

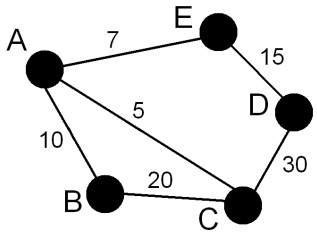


Figure 8: A weighted graph.

The so called "Travelling Salesman Problem" (TSP) is to find the shortest *tour* to visit every vertex of a complete weighted graph exactly once. The vertices represent airports for example, and you want to visit all cities, minimizing the travelling time.

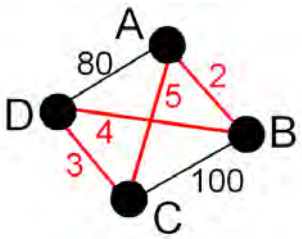


Figure 9: Solution of the Travelling Salesman Problem.

Paths and Cycles

Marcel Schnirring

In this part, different ways to "travel" through graphs are explained.

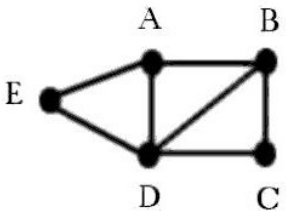


Figure 10: An example graph.

Walk

The simplest way to travel along the edges of a graph is a walk, where you can travel from one vertex to each adjacent vertex in an arbitrary order. For example in figure 10 $A \to D \to E \to A \to D \to B$.

Trail

A trail is a special walk, in which it is forbidden to use an edge twice, for example $A \to D \to B \to A \to E$. The example above for a walk is not a trail, since the edge joining A and D is used twice.

Path

A path is a special trail in which every vertex can only be used once except the initial vertex, which can be the end vertex as well, for example $A \to B \to C \to D$.

Closed Walks, Trails and Paths

Walks, trails and paths are called "closed", if the initial and the end vertex are the same.

$B \to D \to C \to B$ is a closed walk, a closed trail and a closed path, whereas $E \to A \to D \to B \to A \to D \to E$ is a closed walk, but not a closed trail, because vertex D and the edge joining A and D are used twice.

König's Theorem

A graph is bipartite, if and only if the length of each cycle is even. To prove the equivalence in König's Theorem, we have to prove the following two directions:

1. If G is bipartite, then each cycle is of even length („⇒").
2. If each cycle is of even length, then G is bipartite („⇐").

Proof:

„⇒" Let G be a bipartite graph. The vertex set can be split up into black and white vertices, so that each edge joins a black and a white vertex.

Walking along a cycle, one uses a black vertex after a white one was used and vice versa. To finish, one has to get back to the colour of the starting vertex. Therefore, the cycle must be of even length.

„⇐" **To show**: If every cycle in G is of even length, G is bipartite.

Assumption: G is connected. Otherwise, we can apply the proof to each component. Choose any vertex v. Look at the shortest paths from v to all other vertices. Colour a vertex black, if the length of its shortest path to v is even (including v); colour a vertex white, if its distance to v is odd. Choose two white vertices u and u'. Choosing two black vertices is similar to the following deduction.

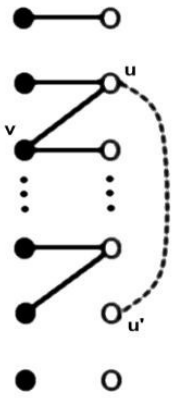


Figure 11

Assumption: G contains an edge uu'. Now look at the shortest paths from v to u and from v to u'. Let w be the first vertex which the paths uv and $u'v$ have in common. Now, we obtain the cycle: wu, uu', $u'w$. The lengths from w to u and from u' to w are either both even or both odd $\Rightarrow$ the distance from u to u' using the paths uw and wu' is always even. Therefore, the cycle must be of odd length because of the $+1$ from the edge uu', which contradicts the assumption, that every cycle is of even length. Hence, there cannot be an edge joining u and u' and the graph must be bipartite.

Theorem of Connectedness

We also proved in our course that, if there is a simple graph with n vertices, m edges and k components, then $n - k \leq m$.

Disconnecting Sets and Bridges

A disconnecting set (in figure 12 and figure 13 green) is any set of edges, whose removal disconnects the graph. A cutset (in figure 12 blue) is a minimal disconnecting set, so that none of its subsets is a disconnecting set.

A cutset with only one element is called a bridge (in figure 13 red).

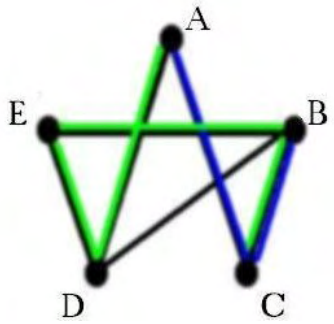


Figure 12

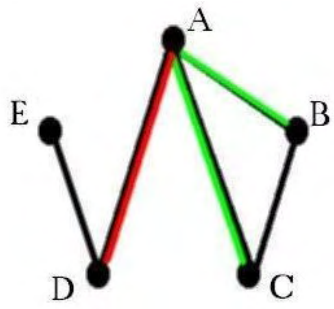


Figure 13

Eulerian Graphs

Marvin Becker

Euler

Leonhard Euler is one of the most important mathematicians of the eighteenth century. Among other things, he showed that it is not possible to walk through Königsberg with its seven bridges, while crossing each of the seven bridges exactly once and returning to the starting quarter.

Leonhard Euler's face can still be found on the ten Swiss Franc note.

Using graph theoretical methods, we proved his results during our course.

A graph is called Eulerian, if it contains a closed trail consisting of every edge exactly once. Practically, Eulerian means that you can draw a graph without lifting your pen and drawing every edge exactly once, finishing at the vertex, at which you started.

Euler's Theorem:

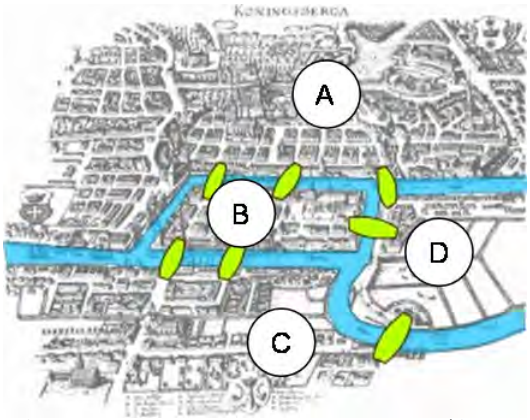
A graph is Eulerian, if and only if the degree of every vertex is even.

Proof:

„⇒“ Assume G is Eulerian. Hence, it contains a closed Eulerian trail. Walking along this trail, one has to leave each entered vertex. So walking d-times through a vertex v, using the Eulerian trail, means that v has degree $2d$.

Therefore, every vertex must have even degree. This is the easy direction of the proof. Euler also proved the other direction, as we did in our course: If the degree of every vertex is even, the graph is Eulerian.

Königsberg Bridges Problem



A map of Königsberg with its quarters (A, B, C and D), its bridges (green) and the river (blue).

To abstract a graph out of the map of Königsberg, we represent the quarters as vertices. For each bridge an edge is drawn between the vertices corresponding to the quarters, which are connected by the bridge. Applying this method, we obtain the graph shown in figure 14.

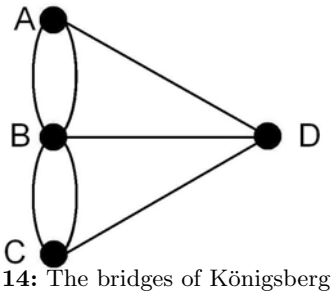


Figure 14: The bridges of Königsberg represented as a graph.

Looking at the degrees of the vertices, you can easily see that they are all odd. According to Euler's theorem, you cannot walk through Königsberg passing every bridge exactly once.

Semi-Eulerian Graphs

Almost same as Eulerian graphs are semi-Eulerian graphs, where you walk along every edge exactly once, but start and end-vertex do not have to be the same. The house of Santa Claus is an example for a semi-Eulerian graph, because you can find a trail through every vertex but the start and end vertex are different.

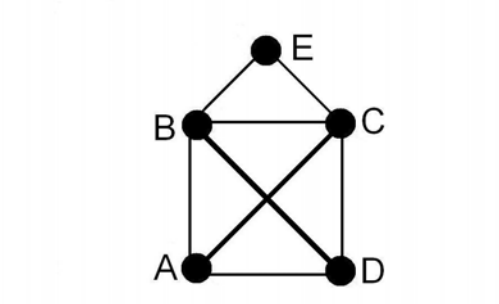


Figure 15: The house of Santa Claus.

A corollary derived from Euler's Theorem: A graph is semi-Eulerian, if and only if at most two vertices are of odd degree.

Proof

„⇒“ To show: If a graph is semi-Eulerian, it contains at most two vertices of odd degree.

If the graph is Eulerian, we are finished. In this case, all vertices have even degree. Otherwise, joining the start and the end vertex of the Eulerian trail by an edge, a closed Eulerian trail is obtained. Hence, all vertices of this graph are of even degree. By removing the added edge, we obtain the original graph, where exactly two vertices are of odd degree.

„⇐“ To show: If a graph has at most two vertices of odd degree, it is semi-Eulerian.

If all vertices are of even degree, the graph is Eulerian. Otherwise, connecting the two vertices of odd degree by an extra edge, yields an Eulerian graph, since all vertices of this graph are of even degree. Hence, this graph contains a closed Eulerian trail. Removing the extra edge leaves an Eulerian trail, which shows that the original graph is semi-Eulerian.

Hamiltonian Graphs

Nico Modenese

Hamiltonian Cycles

A closed path in a graph G, which passes every vertex exactly once, is called a Hamiltonian cycle.

Hamiltonian Graphs

A graph which contains a Hamiltonian cycle is called a Hamiltonian.

Examples

Unfortunately, there is no really "nice" characterization for Hamiltonian graphs as there is for Eulerian graphs. But there are some

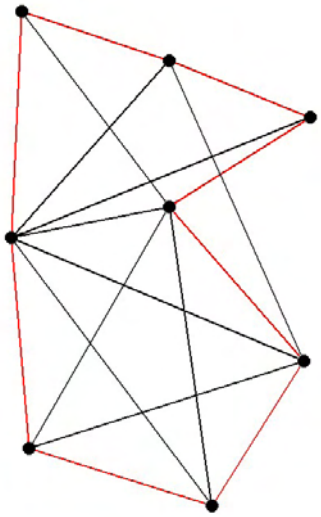


Figure 16: The red edges show a Hamiltonian cycle.

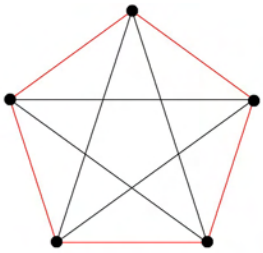


Figure 17: The red edges show a Hamiltonian cycle. The black edges show another Hamiltonian cycle.

theorems, which give sufficient conditions for a graph to be Hamiltonian.

Theorem (Ore):

Let G be a simple graph with $n \geq 3$ vertices. If $\deg(v) + \deg(w) \geq n$ for each pair of non-adjacent vertices v and w, then G is Hamiltonian.

Corollary (Dirac):

If G is a simple graph with $n \geq 3$ vertices, and if $\deg(v) \geq \frac{n}{2}$ for each vertex, then G is Hamiltonian.

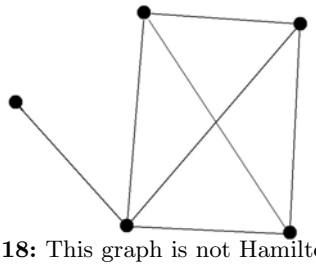


Figure 18: This graph is not Hamiltonian, since it contains a vertex of degree 1.

Shortest Path Algorithm

Nico Modenese

We are looking for the shortest path between two vertices in a weighted graph.

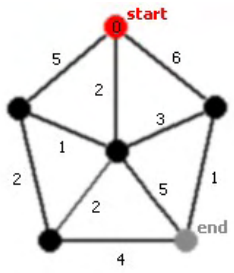


Figure 19

Idea of the Algorithm

We divide the vertices into two subsets: The visited vertices (red) and the non-visited vertices (black). At each step, the non-visited vertex with the shortest distance to the start vertex is added to the set of visited vertices. At the beginning, only the start vertex is in the set of visited vertices (coloured red). After adding a vertex v to the set of visited vertices, the distance from the start vertex to v is written in the vertex. This distance is the length of a shortest path from the start vertex to v. Since the number of vertices is finite, we must reach the end vertex eventually.

Example

First, we look for the shortest path in the graph, which starts at the start vertex. Since only the start vertex is visited in the beginning, the shortest edge, which is incident to it, is the first shortest path. We call this vertex v. We write the weight of the selected edge in v to denote the distance from the start vertex v.

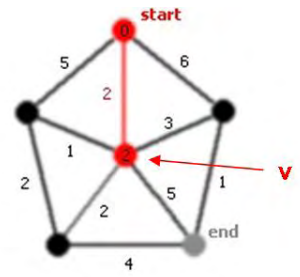


Figure 20

The next shortest path from the start vertex to any other unvisited vertex is obtained by using an edge, which is incident to v or the start vertex. The distance denoted in the end vertex of the next shortest path is calculated by adding the weight of the edge which connects this vertex to a visited vertex and the distance of the visited vertex.

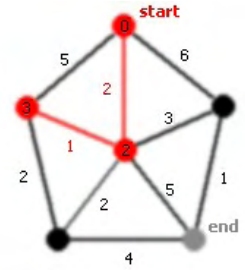


Figure 21

The next edge, which is coloured red in the example, is the one with the weight 1. Going on like this, the following shortest paths are obtained:

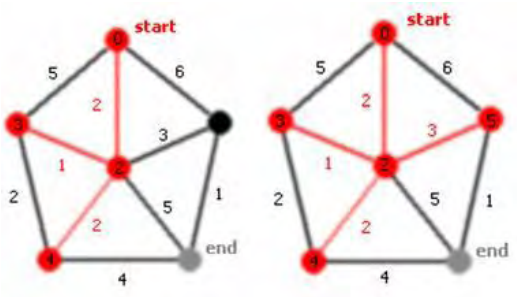


Figure 22

If a path reaches the end vertex, it is the required shortest path, because all shorter paths do not reach the end vertex. Other possible paths from the start to the end vertex must be longer. This algorithm is used in route planning systems for example.

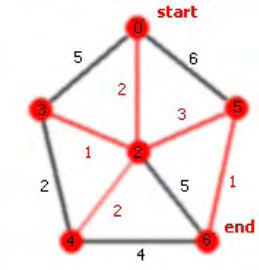


Figure 23

Travelling Salesman Problem

Felix Dörre

The Travelling Salesman problem is about a salesman who wants to find the shortest tour to visit a couple of cities, where he wants to sell and buy goods. In our course, we talked about a similar problem: We wanted to find the shortest tour to visit all course members exactly once and return home. To model this problem, a complete graph is used. The weights denote the quantity, which is supposed to be minimized, such as the time needed to travel from one city to another, the travelling costs or the geographic distance.

Nearest Neighbour Algorithm

The nearest neighbour algorithm is used to find a short tour, to compare its length to other possible tours. In this algorithm, we start at one vertex and use the shortest incident edge to reach one of its neighbours. Then at this vertex, we use the shortest incident edge to a not yet visited vertex. This step is repeated until all vertices are visited. After that, the algorithm returns to the start vertex.

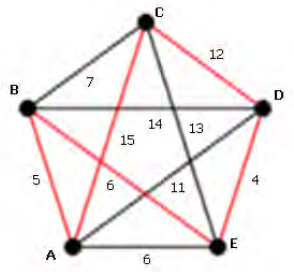


Figure 24: A weighted graph with the tour obtained by using the nearest neighbour algorithm starting at vertex A.

In the example shown in figure 24, A is the

starting vertex. The shortest edge incident to A has length 5 and joins vertex A to vertex B. Then, the edge incident to E is the shortest one, which is incident to B and an unvisited vertex. Next, the edge with weight 4 is chosen and determines D to be the next vertex of the tour. At D, we do not go back to A using the edge with weight 11, because A is already part of the tour. Instead, C is chosen to be the next vertex of the tour. From C we have to take the long distance of 15 back to A. This is the main problem of the algorithm: You have to use this last edge and its weight can be very high. Therefore, you cannot expect to get the shortest tour with the nearest neighbour algorithm.

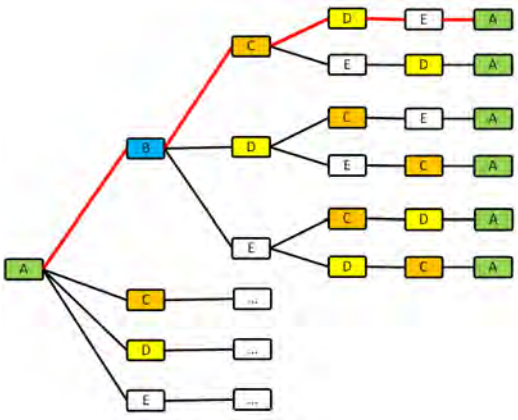
Solution

To determine the shortest tour, all possible tours need to be compared. Since there are

$$\frac{(n-1)!}{2} = \frac{(n-1)\cdot(n-2)\cdot\dots\cdot 2\cdot 1}{2}$$

possible tours in K_n, a computer is used to solve this problem. The computer checks all possible tours in a systematic way:

For example, from A one can go to B, C, D or E. Going to B, only C, D and E are left.



The searching tree with the actual shortest tour highlighted in red.

Comparing all possible tours needs a lot of time in relation to the number of cities involved.

Tour to Visit All Course Members

Using this procedure, we searched for the shortest tour to visit all our course members exactly once. So first, we have to create a complete graph whose vertices represent our home towns. Its weights are the distances between the home towns.

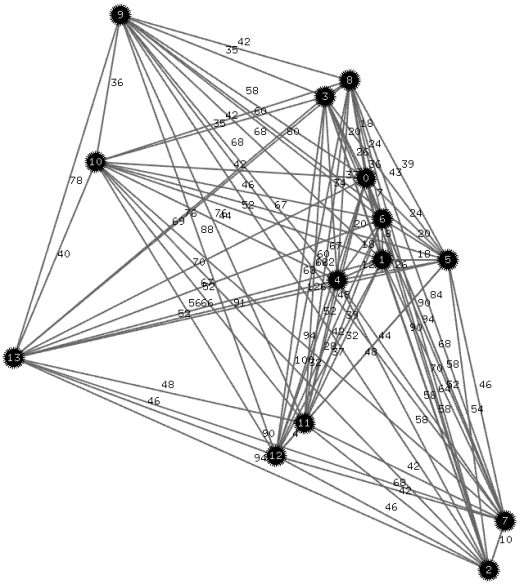


Figure 25: The complete graph corresponding to the map of Baden-Württemberg with our home towns.

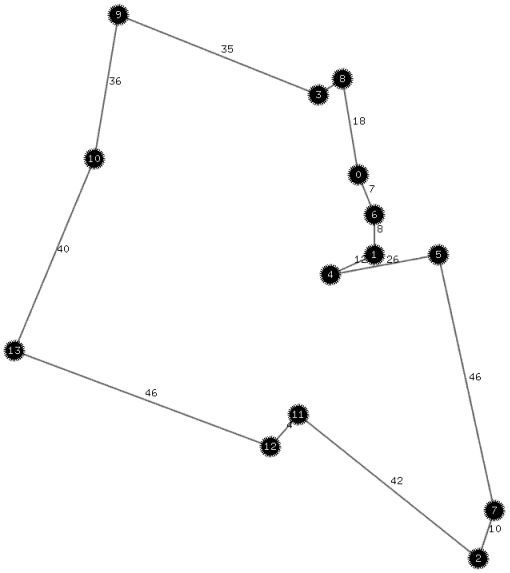


Figure 26: The shortest tour to visit all our course members; one of 3.113.510.400 possibilities.

To find this tour, the computer calculated 30 seconds, which is very long in terms of computers.

The map of Baden-Württemberg with our hometowns and the shortest tour.

Planarity and Colouring Maps

Julia Werle

Planarity

If a graph can be drawn in the plane without crossed edges, it is called a planar graph. Not every graph is planar. But sometimes, graphs which do not look planar are in fact planar since there is more than one way to draw a graph. Some examples:

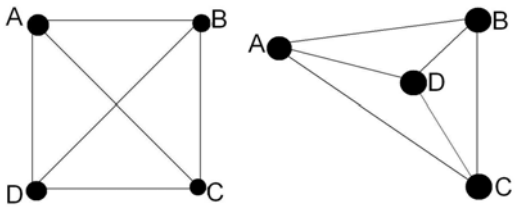


Figure 27: Different drawings of K_4.

Figure 27 shows two different drawings of the complete graph K_4. The first drawing contains crossed edges, but K_4 is planar, since there is a possibility to draw it without crossed edges. Such a drawing is called a plane drawing.

Of course, there are some graphs which are not planar. Kuratowski proved that a graph is planar, if and only if it contains no subgraph contractible to $K_{3,3}$ or K_5.

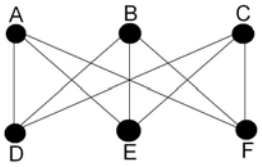


Figure 28: $K_{3,3}$

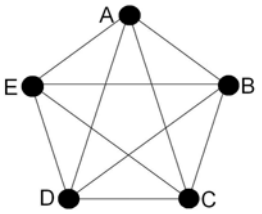


Figure 29: K_5

Contractible refers to contracting an edge, so that its two end vertices are joint together in one new vertex. All edges incident to the original vertices are now incident to the new vertex. For example, the Petersen graph is contractible to K_5 and hence is not planar: Contracting the edges AF, BG, CH, DI and EJ, K_5 is obtained.

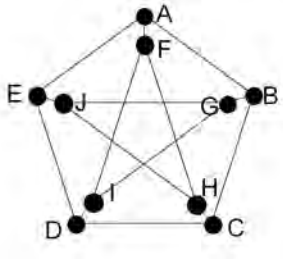


Figure 30: The Petersen graph.

Euler proved, that if G is a plane drawing of a connected graph and n, m and f denote the number of vertices, edges and faces of G respectively, then $n - m + f = 2$ is fulfilled. A face is the smallest region bounded by edges in a planar graph. There is at least one face in a graph – the infinite face. This face surrounds the graph. First of all, an example:

This theorem can be proved by induction on the number of edges (m) of G, which means, we prove it for the smallest possible number of edges first (induction base) and then for all the others (induction step).

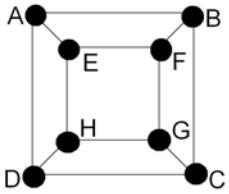
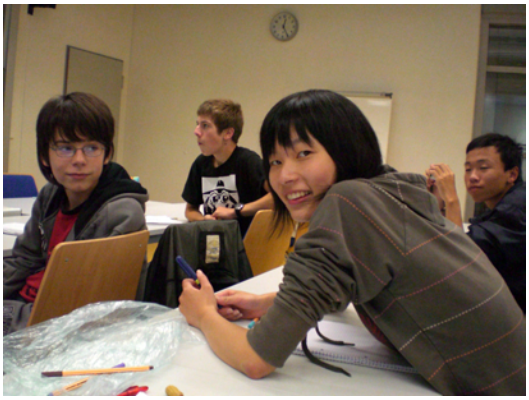


Figure 31: 8 vertices ($n = 8$); 12 edges ($m = 12$) and 6 faces ($f = 6$);
$\Rightarrow\ n - m + f = 8 - 12 + 6 = 2.$

Induction Base:
The lowest number of edges for which $n - m + f = 2$ can be fulfilled is zero. In such a graph, there are no edges and since G is connected, there can be at most one vertex and only one face. If there were two or more vertices and no edges, G would be disconnected. Plugging these values ($n = 1$; $m = 0$; $f = 1$) in Euler's formula results in $1 - 0 + 1 = 2$. So Euler's formula holds for $m = 0$.

Induction Step:
Now it is to show that the formula also holds for a graph G with m edges by using the induction hypothesis, which says that the formula is true for $m - 1$ edges.



We consider two cases:

(1) G is a tree $\Rightarrow$ G contains no cycles, since G is connected. The tree theorem states, that a tree with m edges has $n = m + 1$ vertices. Since there are no cycles in a tree, there can be only one face $\Rightarrow$ $f = 1$. Combining these numbers yields $n - m + f = 2$, as required.

(2) G is not a tree $\Rightarrow$ G contains cycles. Let e be an edge in one of the cycles of G. We delete the edge e in G and obtain the graph G' with $m' = m - 1$ edges, $n' = n$ vertices and $f' = f - 1$ faces, because two faces fuse together when deleting e. So

$$n' - m' + f = 2$$

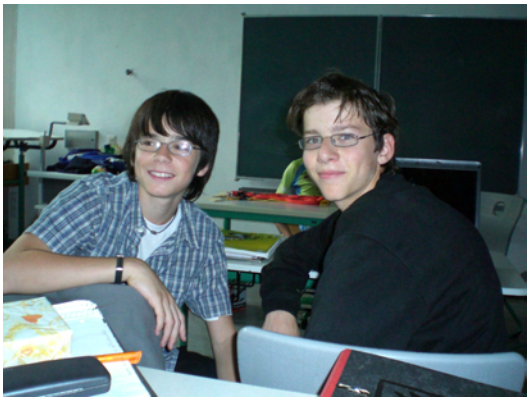
holds due to the induction hypothesis. Plugging in $m' = m - 1$, $n' = n$ and $f' = f - 1$ yields:

$$\begin{aligned} n - (m-1) + (f-1) &= 2 \\ \Rightarrow \quad n - m + f &= 2 \end{aligned}$$

The following two corollaries can be deduced from Euler's formula:

If G is a connected simple planar graph with $n \geq 3$ vertices and m edges, then $m \leq 3n - 6$ is fulfilled.

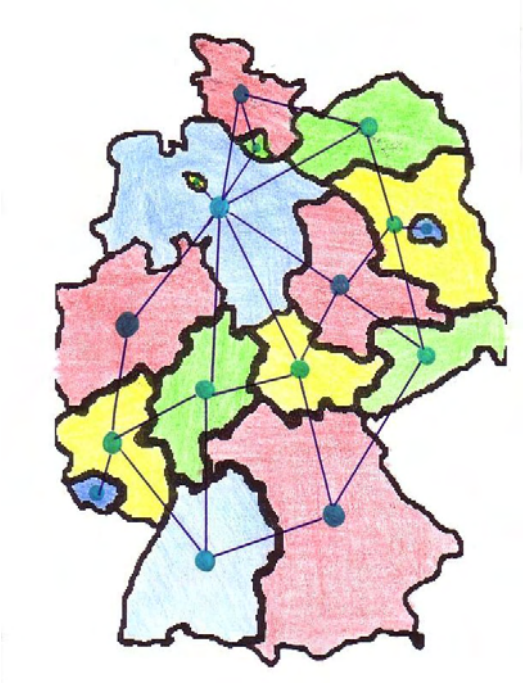
Every simple planar graph contains a vertex of degree at most 5.



Colouring Maps

The problem arose in 1852, when a British man, called Frank Guthrie, wanted to colour the British counties in a way that neighbouring counties are coloured with different colours, using as few colours as possible. We discussed this problem by using graph theoretical approaches. By representing every county as a vertex and drawing edges between neighbouring counties, we obtain a simple planar graph.

We can colour the vertices, which are adjacent, with different colours and the ones, which are not adjacent with the same colour. In this way, you can obtain the general result, that every map can be coloured with four different colours. In general, less colours are not possible, because if a graph contains a subgraph like the one in figure 32, at least 4 colours are needed.



Map of Germany coloured with four colours.

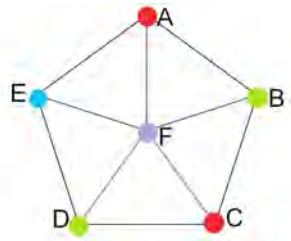


Figure 32

With the first colour, you can colour F, then none of the other vertices can be coloured with that colour. With the second colour A and C can be coloured; with the third B and D. For colouring the last vertex E, a fourth colour will be needed. Every simple planar graph is 4-colourable. Until today, this result was only proved with the help of a computer. But in our course, we proved the easier theorem, that every simple planar graph is 5-colourable.

Trees and Minimal Spanning Trees

Tobias Hocke

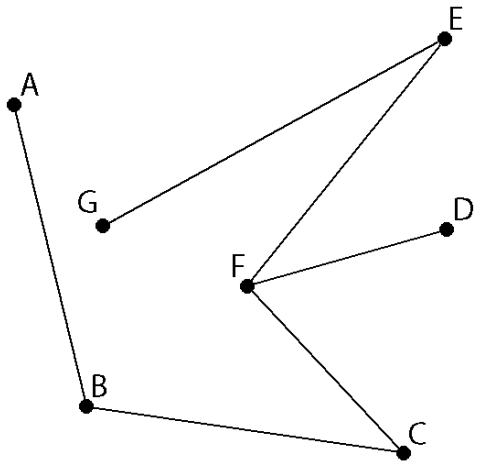


Figure 33: A tree.

Definition

A tree is a connected graph, which contains no cycles. The union of trees is called a forest.

Note: Every tree is a planar, bipartite graph.

Tree Theorem

The following statements about a graph T are equivalent. Let n be the number of vertices of T:

1. T is a tree.
2. T contains no cycles and has $n-1$ edges.
3. T is connected and has $n-1$ edges.
4. T is connected and each edge is a bridge.
5. Any two vertices of T are connected by exactly one path.
6. T contains no cycles, but the addition of any edge creates one cycle.

Proof:

During our course, we proved these equivalences, using a circular argument.

„$1 \Rightarrow 2$“: If T is a tree, T contains no cycles and has $n-1$ edges. By definition, T contains no cycles. In the following, T is constructed gradually. Starting with one isolated vertex,

every other vertex can be added by inserting it with one incident edge, that connects it to the already constructed tree. Hence, T contains $n-1$ edges.

„$2 \Rightarrow 3$“: If T contains no cycles and has $n-1$ edges, T is connected and has $n-1$ edges. Assume T is disconnected. So T consists of at least two components, which are trees. Consequently, T has at most $n-2$ edges, which contradicts the assumption.

„$3 \Rightarrow 4$“: If T is connected and has $n-1$ edges, then T is connected and each edge is a bridge. Assume e is not a bridge. Let T' be the graph obtained from T by removing e. Hence, T' has $n-2$ edges and is connected, contradicting the theorem of connectedness.

„$4 \Rightarrow 5$“: If T is connected and each edge is a bridge, any two vertices are connected by exactly one path. If two vertices were connected by two different paths, then these paths would enclose a cycle. Removing any edge of this cycle does not disconnect the graph, which contradicts the assumption, that every edge is a bridge.

„$5 \Rightarrow 6$“: If any two vertices are connected by exactly one path, T contains no cycles, but the addition of any edge creates one cycle. Adding one edge between the vertices u and v creates a cycle together with the existing path between u and v.

„$6 \Rightarrow 1$“: If T contains no cycles but the addition of any edge creates one cycle, then T is a tree. T contains no cycles by assumption. If T was disconnected, adding an edge between two of its components would not create a cycle, contradicting the assumption.

Spanning Trees

A spanning tree T of a connected graph G is a tree composed of all vertices of G and some of the edges of G.

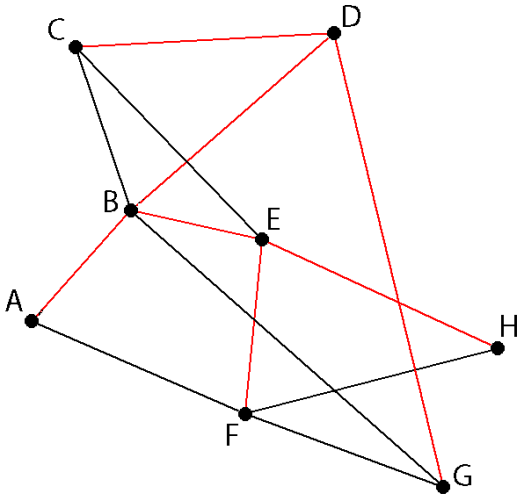


Figure 34: A spanning tree (red).

Minimal Spanning Trees (MST)

A minimal spanning tree is a spanning tree of a weighted graph, for which the sum of all weights of its edges is minimal.

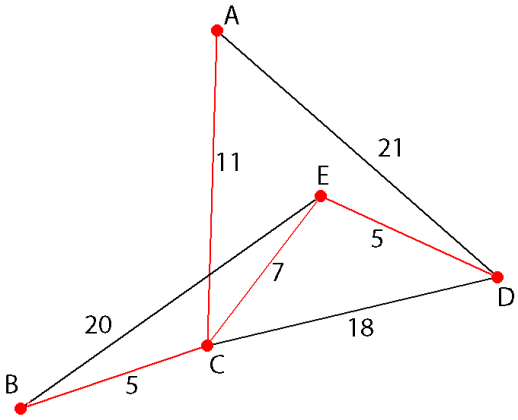


Figure 35: A minimal spanning tree (red).

During our course, we discussed two algorithms to compute a minimal spanning tree of a given weighted graph.

Kruskal's Algorithm: This algorithm creates the MST by first sorting all edges of the graph G in increasing order of their weights. Then it selects the edge with

the lowest weight, which is not yet selected and whose selection does not create a cycle with the other selected edges. This step is repeated until $n-1$ edges are selected, where n denotes the number of vertices.

Prim's Algorithm: At the beginning, this algorithm chooses an arbitrary starting vertex. Then the edge with the lowest weight, joining an unselected and a selected vertex, and the unselected vertex incident to this edge are selected. After repeating this step until $n-1$ edges are selected, the selected edges form a minimal spanning tree.

Rotation

Hanna Binder

Die Rotation diente dazu, andere Kurse über unsere bisherige Arbeit zu informieren und für die Abschlusspräsentation zu üben. Wir teilten uns in vier Gruppen auf und bereiteten eine PowerPoint Präsentation vor. Dazu erstellte je ein Mitglied jeder Gruppe die Folien zu einem Thema. Zusammen gestalteten wir noch das Layout und setzten die verschiedenen Folien zusammen. Außerdem bereiteten die chinesischen Schüler unseres Kurses auf einer Baden-Württemberg-Karte das Schaubild zur „Rundtour" zwischen allen Wohnorten der Kursmitglieder vor.

Am Donnerstag stellten die entsprechenden Gruppen dann ihre bisherige Kursarbeit in etwa 15 Minuten vor und gingen in den restlichen fünf Minuten auf Fragen ein.

Diese Themen stellten wir vor

Grundlagen der Graphentheorie: Wie sieht ein Graph aus? Was kann man mit Graphentheorie machen?

Königsberger Brückenproblem: Warum kann man keinen „Rundgang" durch die Königsberger Stadtviertel finden, bei dem jede der sieben Brücken genau einmal benutzt wird?
Bei welchen Graphen geht das?

Shortest Tour Algorithmus: In welcher Reihenfolge kann man alle Kursmitglieder auf dem kürzesten Weg der Reihe nach besuchen? Wie sieht dieser Algorithmus als Computerprogramm aus?

Wir hielten die ganze Präsentation auf Englisch, da wir alle Themen auf Englisch bearbeitet hatten und so nicht „umschalten" mussten. Auch Fachausdrücke wurden auf Englisch eingeführt und so verstanden auch die chinesischen Schüler alles. Bei jeder Präsentation waren Kursleiter anwesend, die sich Notizen machten. Nach der Rotation besprachen die Leiter die Präsentationen der einzelnen Kurse. Nachmittags fand dann wieder Kurs statt und wir erhielten ein Feedback über unsere Präsentationen. Danach arbeiteten wir weiter an unseren Kursthemen. Abends feierten wir noch die erste „Party" der Akademie 2008, das Bergfest, symbolisch so genannt da es genau zur Halbzeit der Akademie stattfand. Wir überlegten uns einen lustigen Wettbewerb. Die beste Mannschaft erhielt einen genialen Preis: Ein wunderschönes Lebkuchenherz.

Abschlusspräsentation

Hanna Binder

Die Abschlusspräsentation fand am vorletzten Tag der Akademie statt. Wir fanden uns wieder in vier Gruppen zusammen und bereiteten in Expertengruppen die Präsentation der Rotation mit neuen Themen auf. Dann probten wir diese sehr ausführlich und verbesserten sie.

Folgende Themen kamen dazu:

Kürzester Weg: Wie kommt man in einem gewichteten Graphen am schnellsten von

einem Knoten zum anderen?

Einfärben von Landkarten: Wie viele Farben reichen aus, um eine beliebige Landkarte so zu färben, dass aneinander grenzende Länder nie dieselbe Farbe haben?

Am Tag der Präsentation begrüßten wir alle unsere Familien und stellten ihnen unsere Projekte vor. Hier gewährten wir ihnen einen Einblick in die wichtigsten Themen unserer Kursarbeit. Nach der Präsentation konnten sie noch selbst einige Aufgaben zu unseren Kursthemen bearbeiten, die wir vorbereitet hatten. Zum Beispiel sollten sie versuchen, eine Deutschlandkarte mit vier Farben so anzumalen, dass benachbarte Bundesländer nie die gleiche Farbe hatten. Außerdem sollten sie auf einem Arbeitsblatt herausfinden, ob bestimmte Graphen „Eulersch“ waren, also eine, Tour enthalten, die jede Kante genau einmal benutzt. Sehr interessant waren auch die Abschlusspräsentationen einiger anderer Kurse, die wir noch besuchen konnten. Die Teilnehmer der anderen Kurse führten neben ihren PowerPoint Präsentationen ebenfalls Anschauungsmaterial zu ihren Themen vor.

Unser Ausflugstag

Lisa Bitterich

Am 1. September war es so weit: Der Graphentheoriekurs unternahm seine Exkursion nach Heidelberg. Es ging schon ziemlich früh los: Gegen 8 Uhr wurden wir zum Bahnhof in Adelsheim gebracht, von wo aus wir dann zum Interdisziplinären Zentrum für wissenschaftliches Rechnen (kurz IWR) an der Universität Heidelberg fuhren. Dort hielt Dr. Michael Winckler für uns einen interessanten Vortrag. Unter anderem zeigte er uns, wie man einen dreidimensional dargestellten Graphen auf eine Ebene projizieren und wie man die Kanten eines Dodekaeders mit nur drei Farben einfärben kann, ohne dass an einer Ecke eine Farbe mehrmals vorkommt, was perfekt auf unseren Kurs abgestimmt war. Außerdem zeigte Dr. Michael Winckler uns mit welchen Anwendungsgebieten man ein Mathematikstudium verbinden kann. Nach dem Vortrag bot Daniel uns an, das Heidelberger Schloss zu besichtigen. Wer das Schloss schon kannte, konnte sich auch selbstständig die Heidelberger Innenstadt anschauen. Vor allem das Shoppen mit Jack war ein Erlebnis für sich. Was man vielleicht auch noch sagen muss: An diesem Tag ging es unserer Schülermentorin Karima überhaupt nicht gut, trotzdem fuhr sie mit uns mit. Karima, das fanden wir alle ganz toll!

Bericht der chinesischen Gastschüler

Shengjie Zhu (Stella), Zhou Bao (Blues), Nichen Wu (Susan), Kaiyuan Zhu (Jack)

We have been in the graph theory course for nearly two weeks during the Science Academy. Here we experienced a lot of new things, which were exciting and interesting. We have learned the differences between western and oriental cultures through our course and the free time we spent together with our German friends and teachers. It is our honour to write this essay to illustrate our thoughts and feelings.

We were impressed by the teaching methods

here, which are quite different from our previous experiences from China. To study the theorems and definitions, we played a memory game. You need to find two matching cards: on one there is a technical term or the name of a theorem and the other one describes it. We have never learned things in such a way. Our teacher prepared many posters to help us remember the definitions and theorems. There were three teachers, each of whom presents one part of the contents. We could raise our hands whenever we had questions and were confused by some mathematical proof and the teachers would come to our seats to explain it in an easy-understanding way. The atmosphere during the course was not as stressful as we had imagined, there was not so much pressure on the students as it is in our schools. We enjoyed our time in the course, since the teachers have a good sense of humour.

Though the course was new for all of us, the process of teaching was not too fast and we could follow easily. The contents were taught step by step, so it was easy for us to learn all the new things. The students here are relaxed and active. They sit wherever they want, talk whenever they want and ask questions whenever they need. Whenever something was said in German, they always helped us by translating it into English. The most different thing was that the amount of the students is much smaller than in China. We always have a class with approximately fifty students while there are only fourteen students in our course. We sit in a cycle around the blackboard, which makes us closer to each other and helps us to get to know the other students quickly. It also was easy for us to find new friends over here.

We do not take any exams about our course, but there is a thing called “rotation”. We enjoyed the time we spent together to prepare the rotation. We discussed the topics, searched for some reference websites and made PowerPoint presentations in groups, combining our own knowledge together. The different expectations about the presentations were a culture shock for us. We like to decorate our slides and show every item after a click with an animation. But Germans prefer to do PowerPoint in a more straight way: They show all the contents directly without using fancy decorations or animations. Finally, we Chinese and Germans united our opinions in a harmonious way after plenty of communication and argument. We do not have much group work or rotation in China, so it is a good time for us to learn the real meaning of “cooperation”. Through this rotation we got to know the basic structure of a presentation and how to show our ideas to our classmates and those who have not learned the knowledge before.

We are really grateful to the people here, who give us this chance and teach us more than the academic knowledge during this period. We will regard this experience as a treasure in our life.


UMBRO
STANFORD