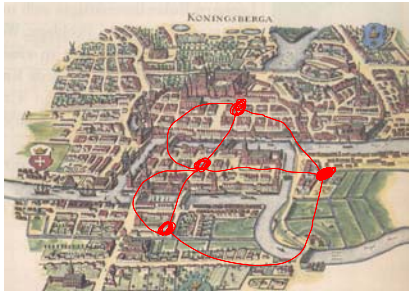
# Graphentheorie


Gibt es in Königsberg einen Spaziergang, bei dem man jede der sieben Pregelbrücken genau einmal überquert?




# Königsberg Bridge Problem

In the year 1736  Leonhard Euler found a general solution.



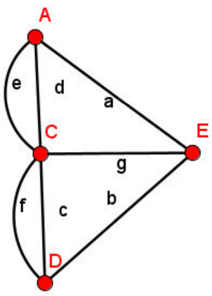

That's Königsberg, now Kaliningrad, a famous town near the Baltic Sea. Does there exist a walk which crosses over every bridge exactly once?



# Was ist ein Graph?

Im Jahre 1736  Leonhard Euler löste das Problem allgemein und begründete damit die Graphentheorie

Jede Kante verläuft von Ecke zu Ecke.




Ein Graph besteht aus einer Eckenmenge und einer Kantenmenge

Ecken= vertices V
Kanten= edges E

$$G=(E,K)$$

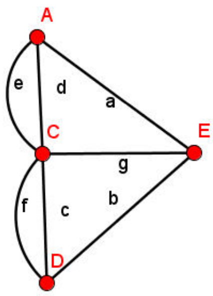
Mindestens eine Ecke



# What's a Graph?

In the year 1736  Leonhard Euler found a general solution. and this way he found the graph theory

Every edge is defined by a pair of vertices.




A graph is made of a set of vertices V and a set edges E.

Ecken= vertices V
Kanten= edges E

$$G=(V,E)$$

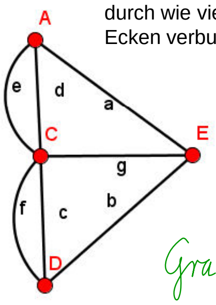
At least one vertex.



# Gundbegriffe

Die **Adjazenz-Matrix** gibt an, durch wie viele Kanten die Ecken verbunden sind.

Der **Grad einer Ecke** ist die Anzahl der abgehenden Kanten.




| | A | C | E | D |
|---|---|---|---|---|
| A | 0 | 2 | 1 | 0 |
| C | 2 | 0 | 1 | 2 |
| E | 1 | 1 | 0 | 1 |
| D | 0 | 2 | 1 | 0 |

↓+

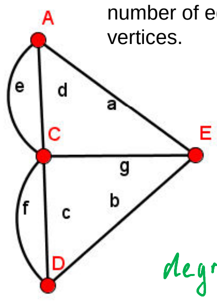
Grad 3 5



# Basic Concepts

The **adjacency matrix** gives the number of edges between two vertices.

The **degree of a vertex** is the number of edges which start at the vertex.




| | A | C | E | D |
|---|---|---|---|---|
| A | 0 | 2 | 1 | 0 |
| C | 2 | 0 | 1 | 2 |
| E | 1 | 1 | 0 | 1 |
| D | 0 | 2 | 1 | 0 |

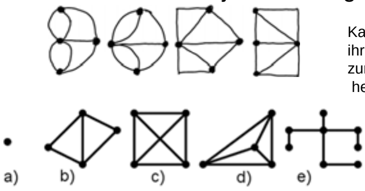
↓+

degree 3 5

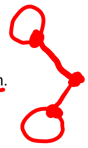
# Eulersche Begriffe

Im Jahre 1736 — Leonhard Euler löste das Problem allgemein

In einem **Eulerschen Weg** kommt jede Kante genau einmal vor.




Kanten, die zu ihrer Startecke zurückkehren, heißen Schlingen.



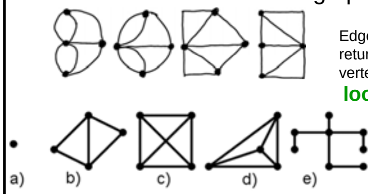
Ein geschlossener Eulerscher Weg heißt **Eulerscher Kreis**.



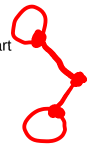
# Euler‘s Concepts

In the year 1736 — Leonhard Euler found a general solution.

An **Eulerian Path** uses each edge precisely once.




Edges, which return to its own start vertex, are named **loops**.



Look at: Glossary of graph theory
From Wikipedia, the free encyclopedia

alphabetic glossery

A closed Eulerian path is named **Eulerian cycle**.

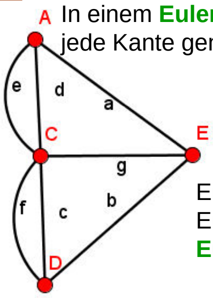


# Eulersche Begriffe

Im Jahre 1736 — Leonhard Euler löste das Problem allgemein

In einem **Eulerschen Weg** kommt jede Kante genau einmal vor.

Königsberg Graph




Ein geschlossener Eulerscher Weg heißt **Eulerscher Kreis**.

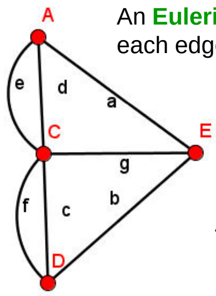


# Euler‘s Concepts

In the year 1736 — Leonhard Euler found a general solution.

An **Eulerian Path** uses each edge precisely once.

Königsberg graph



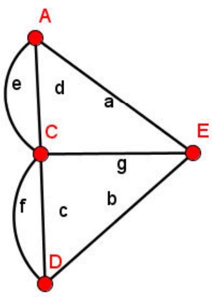

A closed Eulerian path is named **Eulerian cycle**.



# Eulers Lösung:

Im Jahre 1736 — Leonhard Euler löste das Problem allgemein

Im Königsberg-Graphen gibt es keinen Eulerschen Kreis.




Eulerscher Satz:

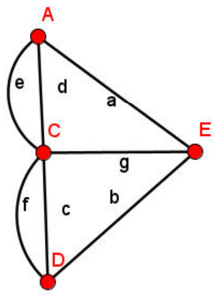
Einen Eulerschen Kreis gibt es genau dann, wenn alle Ecken einen geraden Grad haben.



# Euler‘s Solution:

In the year 1736 — Leonhard Euler found a general solution.

In the Königsberg graph there is no Eulerian cycle.

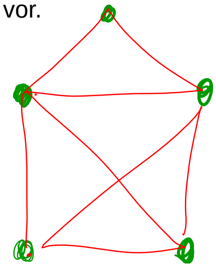



Euler‘s theorem:

An Eulerian cycle exists exactly in the case, when all vertices have an even degree.

# Das Haus des Nikolaus

In einem **Eulerschen Weg** kommt jede Kante genau einmal vor.



Grad:

Eulerscher Satz:

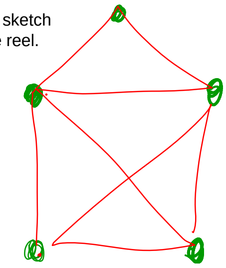
Einen offenen Eulerschen Weg gibt es genau dann, wenn genau zwei Ecken einen ungeraden Grad haben.



# The House of Nikolaus

An **Eulerian Path** uses each edge precisely once.

You can sketch in off the reel.



degree:

Euler's theorem:

An open Eulerian path exists exactly in the case, when presicely two vertices have an odd degree.



# Graphen in unserer Welt




# Graphs all over the World

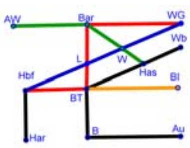



# Graphen in unserer Welt

Mit Graphen schafft man sich ein

Modell der Wirklichkeit,

das einen bestimmten Zusammenhang deutlich macht und andere Aspekte der Wirklichkeit ausblendet.




Die geometrische Lage und Form spielen bei Graphen eigentlich gar keine Rolle.

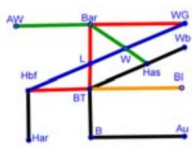
Bei Streckenplänen wird allerdings ganz grob die gegenseitige Lage wiedergegeben.



# Graphs all over the World

With graphs we are modelling reality.

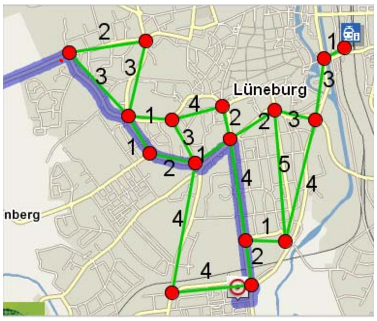
We reveal a certain context but we blind out other aspects of reality.




The geometrical situation and form are not important when we work with graphs.

But the position of locations is approximatly shown in traffic plans. Distances are not realistic.

# Routenplaner und Graphen



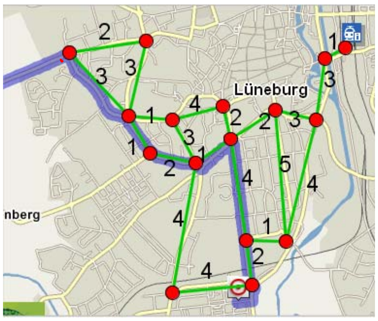

Die Routenplaner arbeiten mit bewerteten Graphen

Die Bewertung kann Entfernung, Zeit, Kosten .... bedeuten.

Erstmal leichtere Probleme:



# Route Planner and Graphs




The route planner works with weighted graphs

The weights can be distance, time, costs and so on.

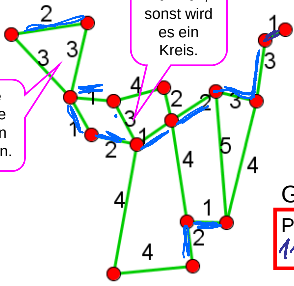
But at first easier problems:



# Optimierung und Graphen

Nicht nehmen, sonst wird es ein Kreis.

Eine Kante davon wählen.



Bewertung Baukosten

Radwegbelag möglichst billig so erneuern, dass jede Kreuzung auf neuem Belag erreichbar ist.

Greedy-Algorithmus

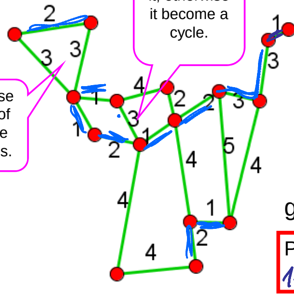
Protokoll
111112222



# Optimization and Graphs

Don't take it, otherwise it become a cycle.

choose one of these edges.



weighting building costs

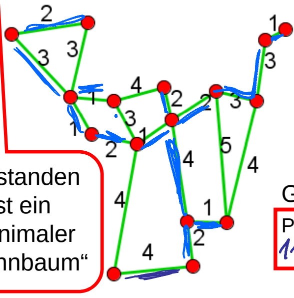
Goal: the bike path must be renewed as cheap as possible. Every crossing must be reachable by a renewed path.

greedy-algorithm

Protokoll
111112222



# Optimierung und Graphen



Entstanden ist ein „minimaler Spannbaum“

Bewertung Baukosten

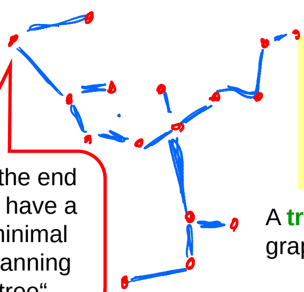
Radwegbelag möglichst billig so erneuern, dass jede Kreuzung auf neuem Belag erreichbar ist.

Greedy-Algorithmus

Protokoll
11111222233344



# Optimization and Graphs



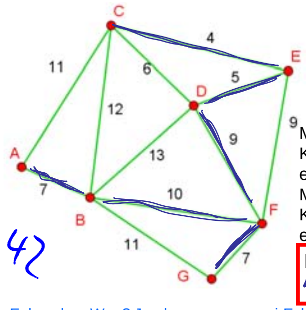
At the end we have a „minimal spanning tree“

weighting building costs

Goal: the bike path must be renewed as cheap as possible. Every crossing must be reachable by a renewed path.

A **tree** is a connected graph without cycles.

## Optimierung und Graphen



Bewertung Baukosten

Leitungsnetz verlegen, so dass jeder Knoten erreicht wird. Minimiere die Baukosten.

### Greedy-Algorithmus

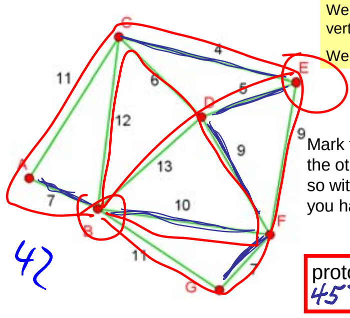
Markiere solange die billigsten Kanten, solange kein Kreis entsteht.
Mache dann mit einer nächst teuren Kante weiter, bis ein Spannbaum entsteht.

Protokoll
45779 10

Eulerschen Weg? Ja, denn genau zwei Ecken haben ungeraden Grad.



## Optimization and Graphs



We make a line network so that every vertex is reached.

We must have minimal building costs.

### greedy-algorithm

Mark the cheapest edges one after the other while no cycle occurs. Do so with bigger weighted edges until you have a spanning tree.

protokol
45779 10

Eulerian path? Yes, because there are exactly two vertices with odd degree.




## Graphen-Theorie

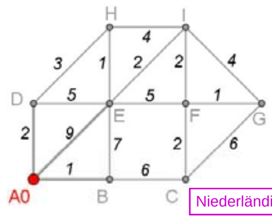
ist eins der spannendsten und dynamischsten mathematischen Themen zur Zeit.

Zwei Mathematiker greifen die Idee von „Sofies Welt" auf…….

http://www-m9.ma.tum.de/Ruth/WebHome



## Kürzeste-Wege-Bäume



Das ist das Routenplaner-Problem. Es ist schwieriger zu lösen.

Dijkstra-Algorithmus.

Wir suchen die kürzesten Wege von A aus zu allen anderen Ecken

Niederländischer Mathematiker Edsger Dijkstra, 1960

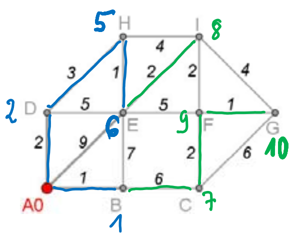
Sprich ij wie ei

Fertige Ecken: A
Aktive Ecke: A
Unfertige Ecken:
Unbetretene Ecken Ecken: B C D E F G H I

Lassen Sie sich im Folgenden lediglich auf den Grundgedanken ein.



## Kürzeste-Wege-Bäume



Das ist das Routenplaner-Problem. Es ist schwieriger zu lösen.

Dijkstra-Algorithmus.

Wir suchen die kürzesten Wege von A aus zu allen anderen Ecken

*Wir beschriften die Ecken mit ihren Entfernungen von A.*

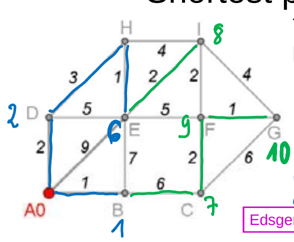
Bei kleinen Graphen findet man eine Lösung durch Hinsehen.

Achte auf die Grundidee

Klausurfrage!



## Shortest path trees



That's the route planning problem. It is more difficult than minimal cost spanning tree problem.

Dijkstra-Algorithmus.

We search a shortest path form A to each other vertex.

*We label the vertices by their distance to A.*

Edsger Dijkstra, 1960, dutch mathematician

speak ij wie ai

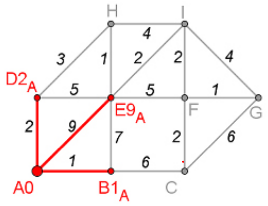
For small graphs you find a solution by head.

Remark only the basic idea .

Klausurfrage!

**The next slide in English is number 39.**

## Kürzeste-Wege-Bäume




Das ist das Routenplaner-Problem.

Dijkstra-Algorithmus.

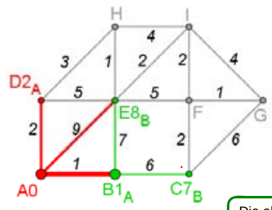
**Wir notieren an jeder Ecke Abstand von A und Vorgänger-Kante.**

Eine Ecke mit minimalem Wert wird neue aktive Ecke.

Ready vertices: A
Active vertex: B
Unready vertices: B1$_A$, E9$_A$, D2$_A$
Untrodden vertices: C F G H I



## Kürzeste-Wege-Bäume




Das ist das Routenplaner-Problem.

Dijkstra-Algorithmus.

**Wir notieren an jeder Ecke Abstand von A und Vorgänger-Kante.**

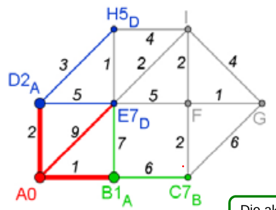
Die aktive Ecke ist fertig.

Eine Ecke mit minimalem Wert wird neue aktive Ecke.

Fertige Ecken: A, B1$_A$,
Aktive Ecke wird: D
Unfertige Ecken: E9$_A$, E8$_B$, D2$_A$, C7$_B$,
Unbetretene Ecken Ecken: F G H I



## Kürzeste-Wege-Bäume




Das ist das Routenplaner-Problem.

Dijkstra-Algorithmus.

**Wir notieren an jeder Ecke Abstand von A und Vorgänger-Kante.**

*Der „Wert“ einer Ecke ist seine Entfernung von A.*

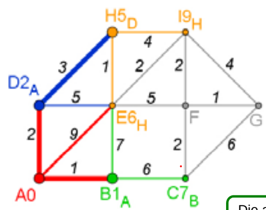
Die aktive Ecke ist fertig.

Eine Ecke mit minimalem Wert wird neue aktive Ecke.

Fertige Ecken: A0, B1$_A$, D2$_A$,
Aktive Ecke wird: H
Unfertige Ecken: E9$_A$, E8$_B$, C7$_B$, E7$_D$, H5$_D$,
Unbetretene Ecken Ecken: F G I



## Kürzeste-Wege-Bäume




Das ist das Routenplaner-Problem.

Dijkstra-Algorithmus.

**Wir notieren an jeder Ecke Abstand von A und Vorgänger-Kante.**

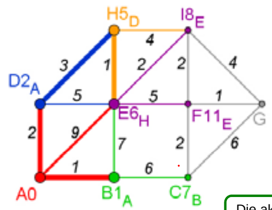
Die aktive Ecke ist fertig.

Eine Ecke mit minimalem Wert wird neue aktive Ecke.

Fertige Ecken: A0, B1$_A$, D2$_A$, H5$_D$
Aktive Ecke wird: E
Unfertige Ecken: E9$_A$, E8$_B$, C7$_B$, E7$_D$, E6$_H$, I9$_H$
Unbetretene Ecken Ecken: F G



## Kürzeste-Wege-Bäume




Das ist das Routenplaner-Problem.

Dijkstra-Algorithmus.

**Wir notieren an jeder Ecke Abstand von A und Vorgänger-Kante.**

*Der „Wert“ einer Ecke ist seine Entfernung von A.*

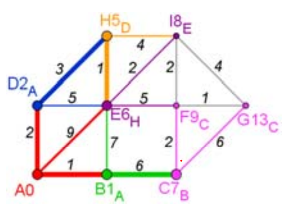
Die aktive Ecke ist fertig.

Eine Ecke mit minimalem Wert wird neue aktive Ecke.

Fertige Ecken: A0, B1$_A$, D2$_A$, H5$_D$, E6$_H$
Aktive Ecke wird: C
Unfertige Ecken: C7$_B$, I9$_H$, I8$_E$, F11$_E$
Unbetretene Ecken Ecken: G



## Kürzeste-Wege-Bäume




Das ist das Routenplaner-Problem.

Dijkstra-Algorithmus.

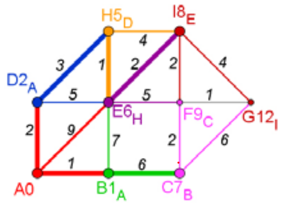
**Wir notieren an jeder Ecke Abstand von A und Vorgänger-Kante.**

Die aktive Ecke ist fertig.

Fertige Ecken: A0, B1$_A$, D2$_A$, H5$_D$, E6$_H$, C7$_B$
Aktive Ecke wird: I
Unfertige Ecken: I9$_H$, I8$_E$, F11$_E$, F9$_C$, G13$_C$,
Unbetretene Ecken Ecken:

Eine Ecke mit minimalem Wert wird neue aktive Ecke.

## Kürzeste-Wege-Bäume




Das ist das Routenplaner-Problem.

Dijkstra-Algorithmus.

**Wir notieren an jeder Ecke Abstand von A und Vorgänger-Kante.**

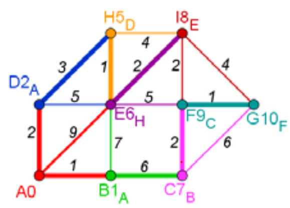
Die aktive Ecke ist fertig.

Fertige Ecken: A0, $B1_A$, $D2_A$, $H5_D$, $E6_H$, $C7_B$, $I8_E$
Aktive Ecke wird: F
Unfertige Ecken: $F11_E$, $F9_C$, $G13_C$, $G12_I$,
Unbetretene Ecken Ecken:

Eine Ecke mit minimalem Wert wird neue aktive Ecke.



## Kürzeste-Wege-Bäume




Das ist das Routenplaner-Problem.

Dijkstra-Algorithmus.

**Mit Wert und Vorgänger für jede Ecke haben wir den gesuchten Baum.**

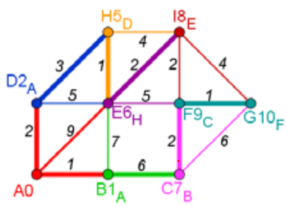
Die aktive Ecke ist fertig.

Fertige Ecken: A0, $B1_A$, $D2_A$, $H5_D$, $E6_H$, $C7_B$, $I8_E$, $F9_C$, $G10_F$
Aktive Ecke wird: G
Unfertige Ecken: $G13_C$, $G12_I$, $G10_F$
Unbetretene Ecken Ecken:

Die letzte aktive Ecke ist fertig.



## Shortest path trees




That's the route planning problem.

Dijkstra-Algorithmus.

**Distance and antecedent at every vertex indicate the tree witch we have searched**

Fertige Ecken: A0, $B1_A$, $D2_A$, $H5_D$, $E6_H$, $C7_B$, $I8_E$, $F9_C$, $G10_F$

**Das ist nun ein kürzeste-Wege-Baum.**
**That's the shortest path tree.**

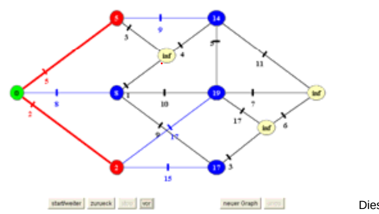
http://www-m9.ma.tum.de/Allgemeines/DijkstraApplet



## Kürzeste-Wege-Bäume

Das ist das Routenplaner-Problem. Es wird gelöst vom Dijkstra-Algorithmus.

http://www-m9.ma.tum.de/Allgemeines/DijkstraApplet



Interaktiv version at TU München

Dies ist Aufgabenblatt 6 bei der TUM



## Logistik

1. Modellierung des Problems mit Graphen
2. Bewertung des Graphen mit
   1. Fahrzeiten
   2. Fahrkosten
   3. Streckenlänge….

Lösung des Kürzeste-Wege-Problems



## Logistics

1. Modelling of the problem with graphs
2. Weighting of the graph with
   1. travel times
   2. travel costs
   3. distance….

solution of the shortest path problem

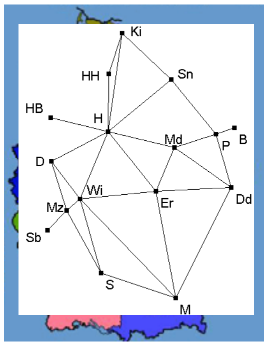
## Landkarten färben

Wie viele Farben braucht man, wenn benachbarte Länder verschieden gefärbt sein sollen?

Modellierung des Problems mit Graphen:



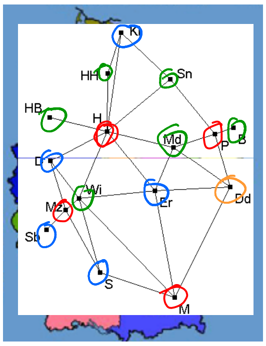
## Landkarten färben mit Graphentheorie




Wie viele Farben braucht man, wenn benachbarte Hauptstädte verschieden gefärbt sein sollen?



## Landkarten färben mit Graphentheorie




Wie viele Farben braucht man, wenn benachbarte Hauptstädte verschieden gefärbt sein sollen?

Vier-Farben-Satz

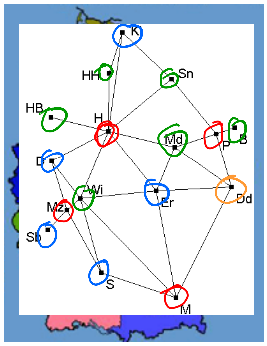
Es reichen immer vier Farben

Das gilt für planare Graphen.

Erst 1976 mit Computereinsatz bewiesen (Appel, Haken)



## Map Coloring by Graph Theory




How many colors are nessecary to color neighboring capitols distinct?

Four color theorem:

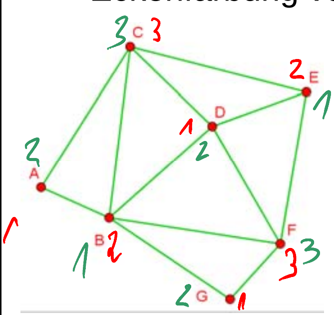
Four colors are enaught.

That's right for planar graphs.

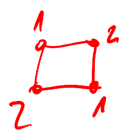
Prooved not until 1976 but unusually by computers (Appel, Haken)



## Eckenfärbung von Graphen




Die Ecken sollen so gefärbt werden, dass benachbarte Ecken verschiedene Farben haben

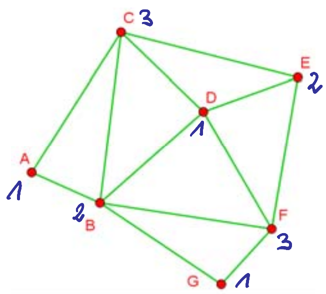


Oft gibt es mehrere Lösungen.

Achtung: Der 4-Farbensatz gilt nur für Graphen ohne Kantenkreuzungen.



## Vertex Coloring of Graphs

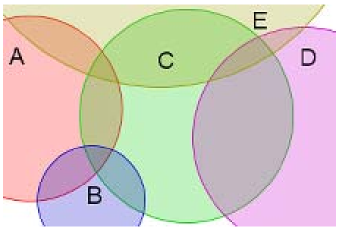


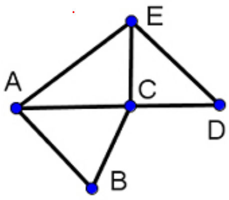

It is a way of coloring the vertices of a graph such that no two neighboring vertices share the same color.

Often there are more solutions.

Attention: The four color theorem is only correct for graphs without cossings.

## Mobilfunk-Konfliktgraphen

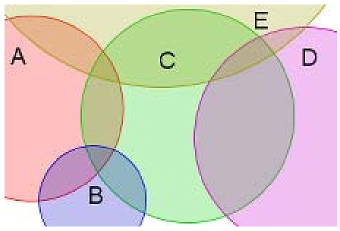


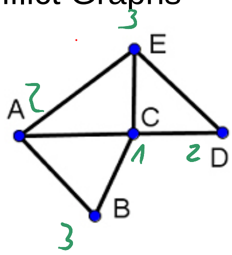




Überlappende Handybereiche brauchen verschiedene Sendefrequenzen. Eine Eckenfärbung des Konflikt-graphen zeigt, wie man Frequenzen zuordnen kann.



## Radio Transmitter Conflict Graphs



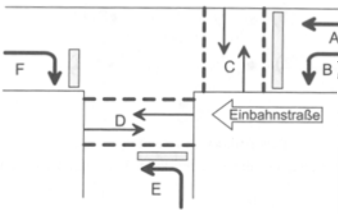




Overlapping mobile regions need distict transmitter frequencies. A vertex coloring of the canflict graph shows wich of the transmitters can have the same frequence.

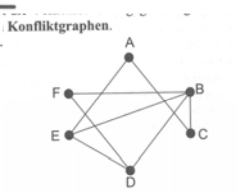
Here A and D kann have the same frequency and E and B can have another,



## Konflikt-Graphen




Eine verkehrsreiche Einmündung

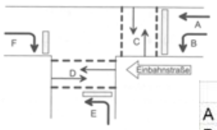



Der Konfliktgraph der Einmündung

Die Verkehrsströme werden Ecken.
Wenn zwei in Konflikt geraten, werden sie durch eine Kante verbunden.



## Konflict Graphs

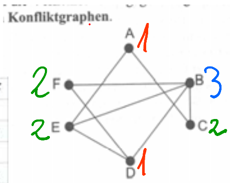



Eine verkehrsreiche Einmündung

traffic at a street crossing

adjacency matrix

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | | | x | | x | |
| B | | | x | x | x | x |
| C | x | x | | | | |
| D | | x | | | x | x |
| E | x | x | | x | | |
| F | | x | | x | | |




Der Konfliktgraph der Einmündung

conflict graph of the crossing

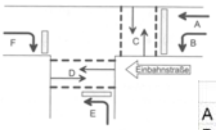
Every traffic stream becomes a vertex. You make an edge in case of two streams conflict. A proper vertex coloring of the graph shows which streams can have „green" by the traffic signal at once.

http://en.wikipedia.org/wiki/Graph_colouring

Mehr dazu im Buch Nitzsche: Graphentheorie, tw.moodle Kap 11



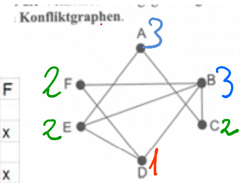
## Konflikt-Graphen




Eine verkehrsreiche Einmündung

Adjazenzmatrix dazu

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | | | x | | x | |
| B | | | x | x | x | x |
| C | x | x | | | | |
| D | | x | | | x | x |
| E | x | x | | x | | |
| F | | x | | x | | |



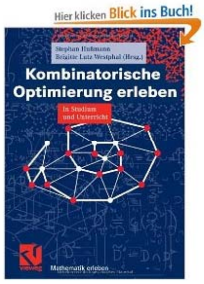

Der Konfliktgraph der Einmündung

Die Verkehrsströme werden Ecken. Wenn zwei in Konflikt geraten, werden sie durch eine Kante verbunden.
Eine zulässige Eckenfärbung des Graphen zeigt: Verkehrsströme mit der gleichen Farbe dürfen gleichzeitig „Grün" an ihrer Ampel haben.

Mehr dazu im Buch Nitzsche: Graphentheorie, tw.moodle Kap 11



## Graphentheorie in Büchern



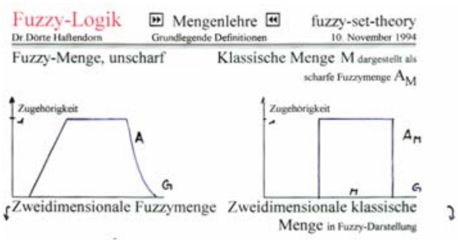

**Kombinatorische Optimierung erleben:**
**Im Studium und Unterricht**
Stephan Hußmann (Autor),
Brigitte Lutz-Westphal (Autor)


Manfred Nitzsche


Dörte Haftendorn

# Fuzzy logic

„weak logic“

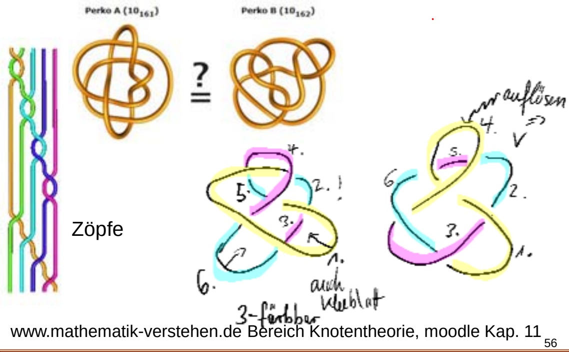



www.mathematik-verstehen.de Bereich Algebra, Logik

55

Prof. Dr. Dörte Haftendorn, Leuphana Universität Lüneburg, 2013 http://www.leuphana.de/matheomnibus

# Knot theory

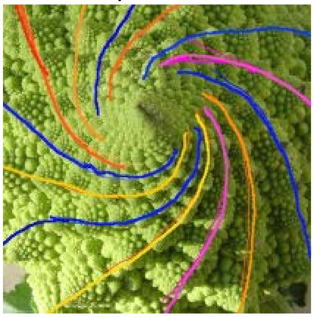



www.mathematik-verstehen.de Bereich Knotentheorie, moodle Kap. 11

56

Prof. Dr. Dörte Haftendorn, Leuphana Universität Lüneburg, 2013 http://www.leuphana.de/matheomnibus

# Fraktals, Chaos Theory

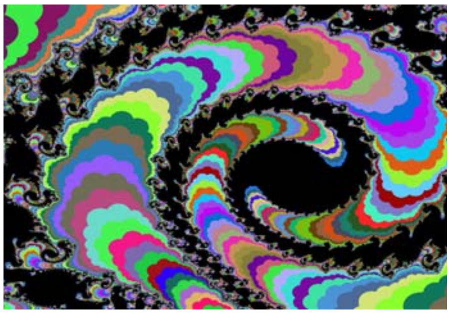


www.mathematik-verstehen.de Bereich Fraktale

57

Prof. Dr. Dörte Haftendorn, Leuphana Universität Lüneburg, 2013 http://www.leuphana.de/matheomnibus

# Fraktale, Chaostheorie

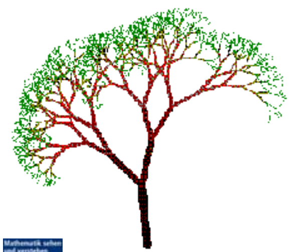


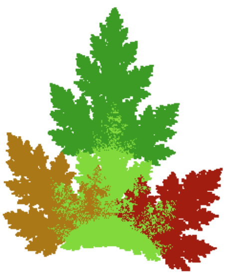
www.mathematik-verstehen.de Bereich Fraktale

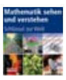
58

Prof. Dr. Dörte Haftendorn, Leuphana Universität Lüneburg, 2013 http://www.leuphana.de/matheomnibus

# Fraktale, Chaostheorie







www.mathematik-verstehen.de Bereich Fraktale, moodle Kap.10

59

Prof. Dr. Dörte Haftendorn, Leuphana Universität Lüneburg, 2013 http://www.leuphana.de/matheomnibus