Informatik-Propädeutikum

Dozentin: Dr. Claudia Ermel

Betreuer: Sepp Hartung, André Nichterlein, Clemens Hoffmann

Sekretariat: Christlinde Thielcke (TEL 509b)

TU Berlin
Institut für Softwaretechnik und Theoretische Informatik
Prof. Niedermeier
Fachgruppe Algorithmik und Komplexitätstheorie
http://www.akt.tu-berlin.de

Wintersemester 2013/2014

Gliederung

Abstraktion

Franz Marc (1880–1916), Kämpfende Formen

Abstraktion

Adolf Hölzel: Abstraktion II, 1915/16

Abstraktion

Wikipedia: Das Wort Abstraktion (lat. abstractus – „abgezogen“, Partizip Perfekt Passiv von abs-trahere – „abziehen, entfernen, trennen“) bezeichnet meist den induktiven Denkprozess des **Weglassens von Einzelheiten** und des **Überführens auf etwas Allgemeineres oder Einfacheres**.

Beispiel:

Karikaturen abstrahieren!

Autos und Abstraktion

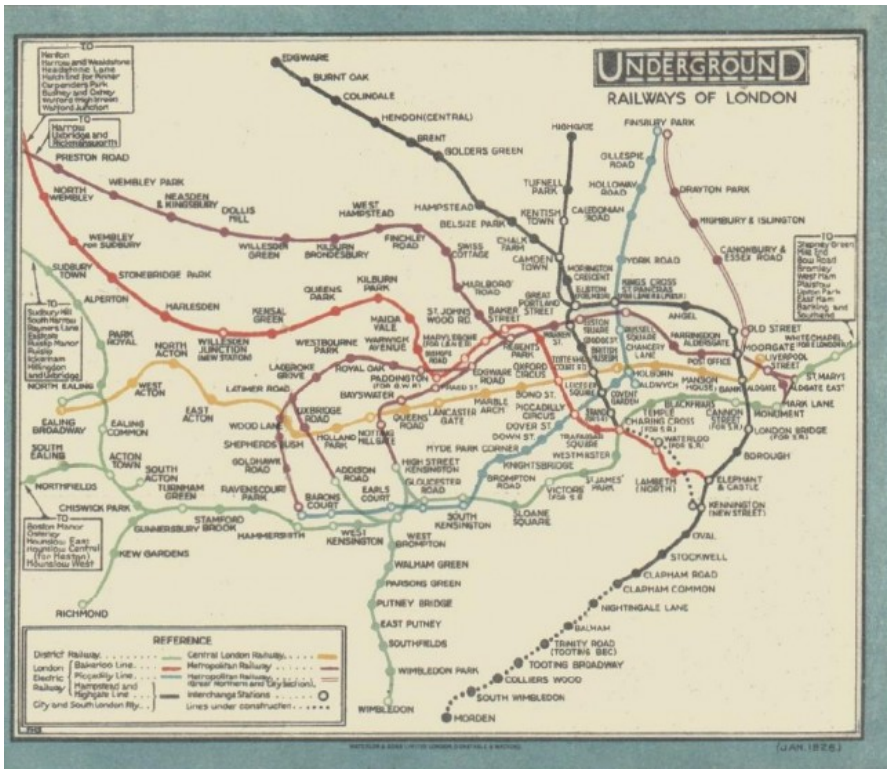
Ein klassisches Beispiel für Abstraktion: **Nachdenken über Autos**.

Reparieren: Bei der Reparatur eines Autos denkt man es sich (zunächst) nicht bestehend aus Schrauben, Muttern, Kabel, Gummi, Plastik, ..., sondern bestehend aus **Komponenten** wie Motor, Kraftstoffeinspritzung, Bremsen, Filter, Batterie, etc.
Das ist Abstraktion!

Fahren: In den Frühtagen des Automobils mussten die Fahrer zugleich nahezu Mechaniker sein. Später bedurften Autos zumindest auch des gekonnten Umgangs mit Kupplung und Gangschaltung; Autos mit Automatikgetriebe erfordern i.w. nur noch die Kenntnis von Gas und Bremse...
Das ist Abstraktion! (Nun kann jede(r) fahren(?)!)

Wichtig: **Schnittstellen** wie Gas- und Bremspedal, Lenkrad, ...

Abstraktion: London Underground - Liniennetzpläne



1928

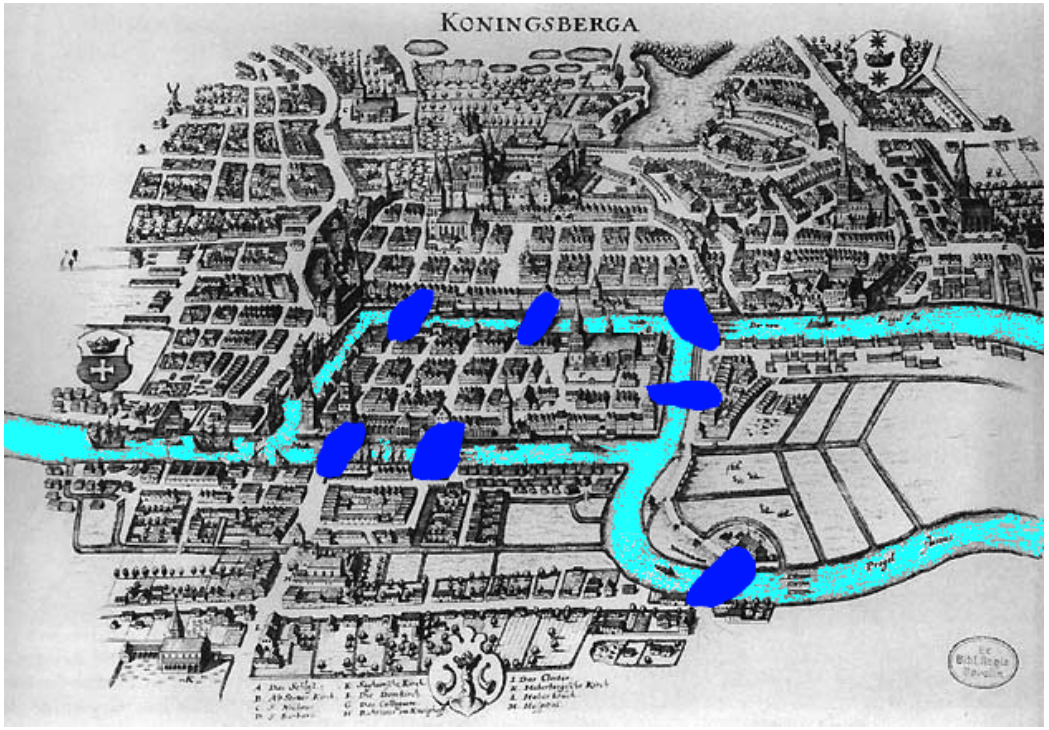
1933 (dank Harry Beck):

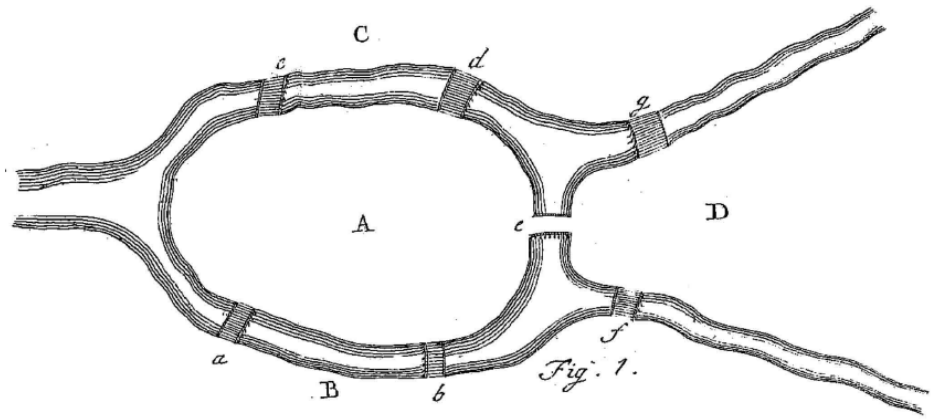
Wozu Abstraktion?

Vorteile des Abstrahierens:

- **Modellierung** und damit Abstraktion ist eine der zentralsten Informatiktechniken und unabdingbar für die Abbildung der realen Welt auf den Computer / die Software.
- Rechnen (Computing) ist letztlich nichts anderes als das Konstruieren, Manipulieren (Verarbeiten) und Folgern auf Basis von Abstraktionen.
- Mit Abstraktion lässt sich **Komplexität (besser) beherrschen**, z.B. komplexe Softwaresysteme.
- Der Umgang mit **mehreren Abstraktionsstufen** ist unerlässlich insbesondere beim Erstellen großer Softwaresysteme.
- Abstrahieren hilft beim Verallgemeinern von Lösungsansätzen und der Mehrfachverwendung von Softwarelösungen (vgl. Programmbibliotheken).
- Abstrahieren schärft den Blick aufs Wesentliche und macht damit erst den Weg für manche Lösungsstrategie frei.

Ein Abstraktionsklassiker: Königsberger Brückenproblem I

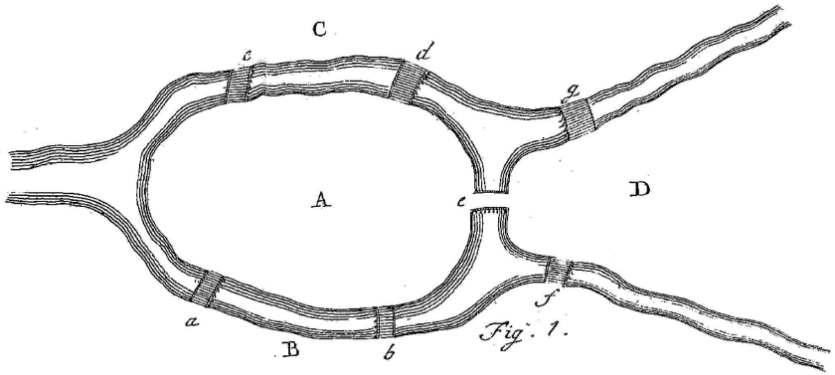


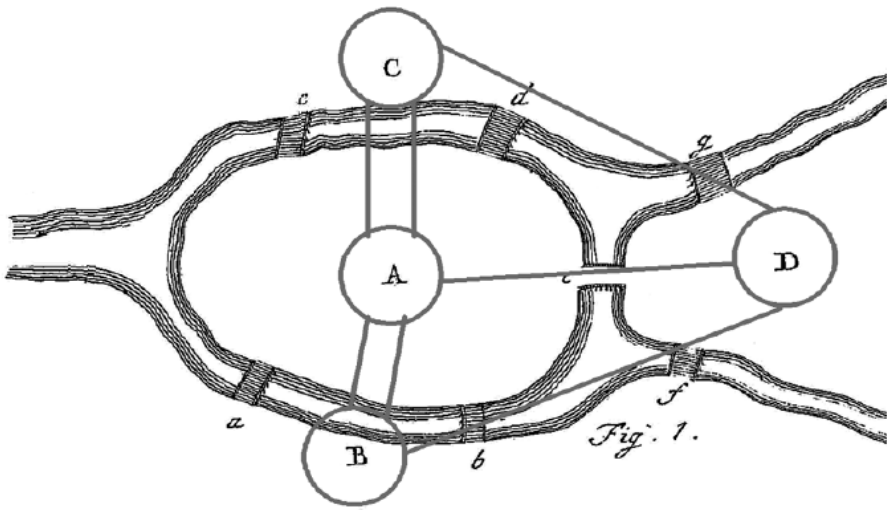


Im preußischen Königsberg gibt es eine Insel A, genannt der Kneiphof, und der Fluss, der sie umfließt, teilt sich in zwei Arme. Über die Arme dieses Flusses führen sieben Brücken a, b, c, d, e, f und g. Nun galt es die Frage zu lösen, ob jemand seinen Spazierweg so einrichten könne, dass er jede dieser Brücken einmal und nicht mehr als einmal überschreite.

– Leonhard Euler, 1707–1783

Eulers Königsberger Brückenproblem II





Beobachtungen:

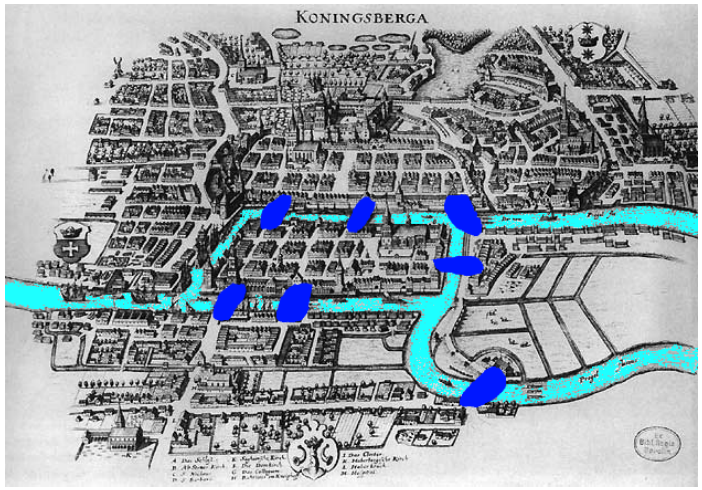
- Größen und Rechnen mit Größen irrelevant!
- Eigenschaften bzgl. Lagen und Beziehungen wichtig!

Modellierung:

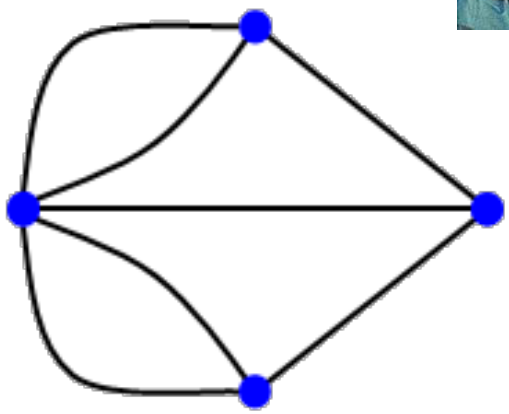
- Orte A, B, C, und D $\hat{=}$ Knoten;
- Brücken $\hat{=}$ Kanten.

Gesucht: Ein Weg durch alle Knoten so dass jede Kante genau einmal besucht wird!

Eulers Königsberger Brückenproblem III



$\Rightarrow$



Für einen zusammenhängenden (Multi-)Graphen gilt:

Satz: Ein zusammenhängender Graph enthält einen **Euler-Kreis** genau dann, wenn jeder Knoten geraden Grad (Anzahl anliegender Kanten) hat; falls genau zwei Knoten ungeraden Grad haben, so hat er einen **Euler-Pfad** aber keinen Euler-Kreis.

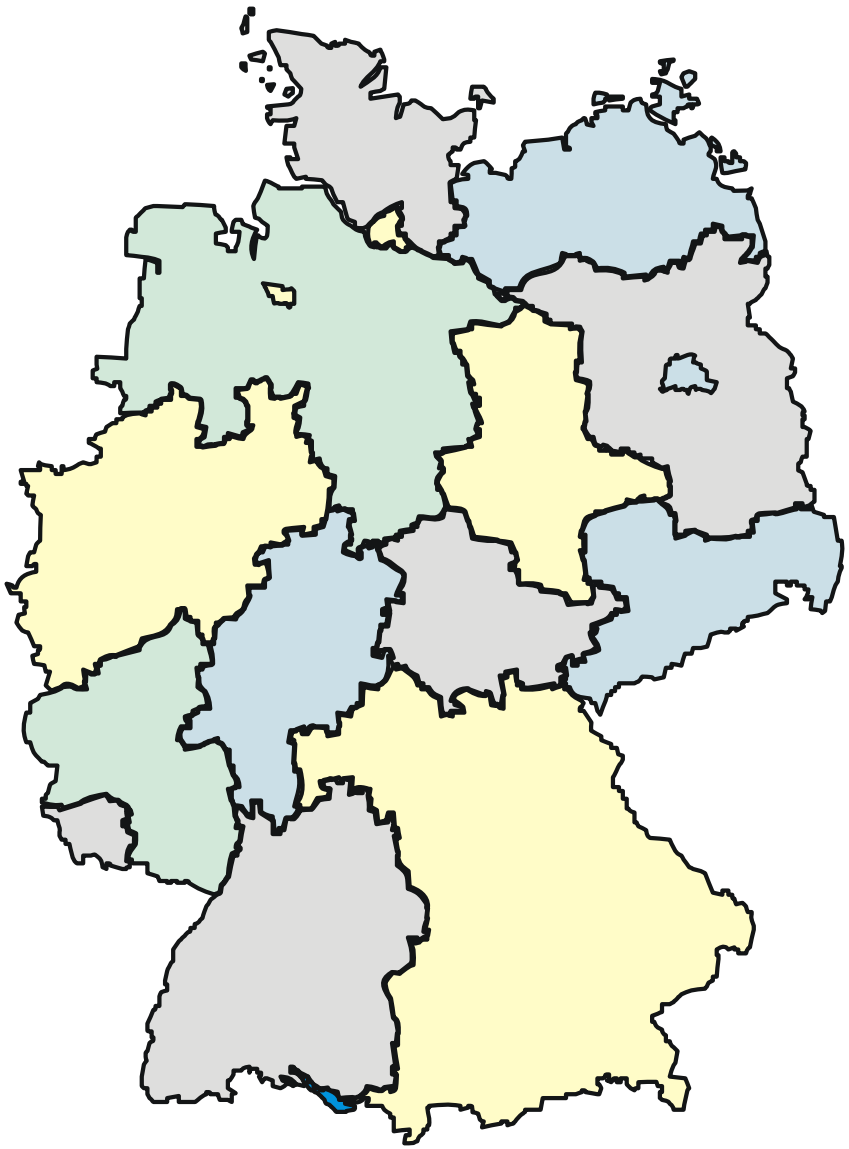
Eulers Königsberger Brückenproblem IV

Rückblick:

- Mit schneller Rechentechnik hätte man ohne Nachzudenken einfach alle möglichen Permutationen der Brücken (Kanten) überprüfen können?! **Ja, aber** bei m Kanten sind das $m!$ Möglichkeiten. Bei wachsendem m wächst $m!$ exponentiell schnell und kann schon für $m = 100$ nicht mehr berechnet werden.
- Eulers Lösung hingegen führt zu einem einfachen und effizienten Algorithmus, dessen Laufzeit nur linear in der Anzahl der Kanten wächst: Gehe alle Knoten durch und bestimme ihren Grad. Dies läuft in Sekundenschnelle selbst auf Graphen mit Milliarden von Kanten.

Bemerkung: In aktueller Forschung beschäftigen wir uns damit, Graphen durch Hinzufügen möglichst weniger Kanten „eulersch“ zu machen. Das könnte einmal der Berliner Stadtreinigung helfen...

Noch ein Abstraktionsklassiker: Landkartenfärben I

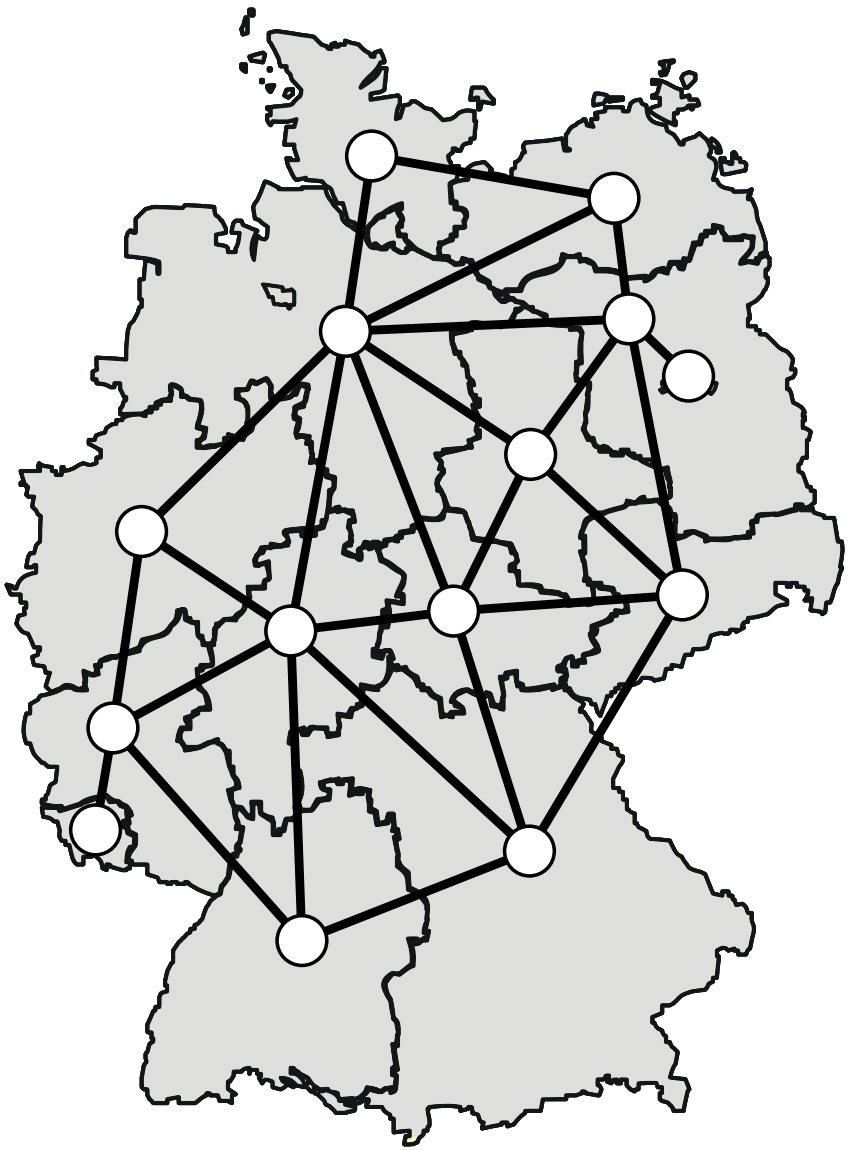


Aufgabe: Färbe Karte sodass zwei benachbarte Bundesländer unterschiedliche Farbe haben.

Triviale Lösung: Eigene Farbe für jedes Bundesland.

Ziel: Verwende möglichst wenige Farben!

Landkartenfärben II

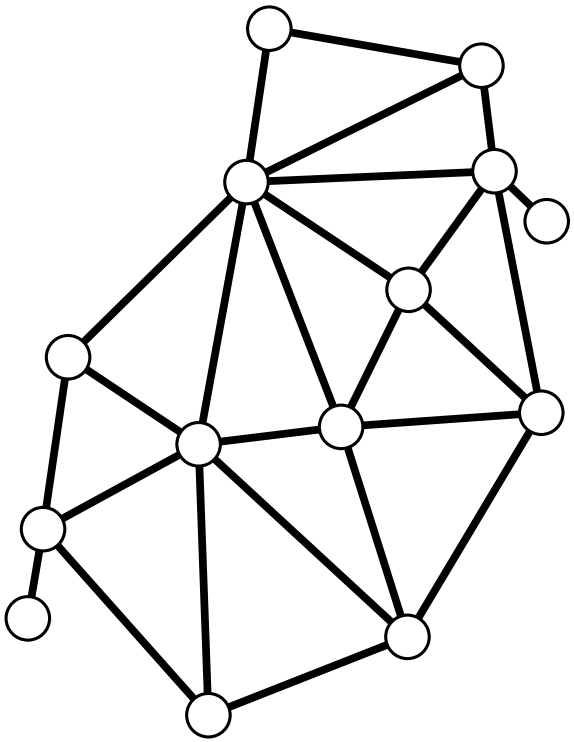


Beobachtung: Genaue Form der Bundesländer nicht wichtig, nur Nachbarschaftsrelation entscheidend.

Modellierung als Graphproblem:
Bundesländer $\hat{=}$ Knoten
Nachbarschaft $\hat{=}$ Kanten

Landkartenfärben III

Landkartenfärben ist ein Knotenfärbungsproblem in Graphen:



Problem: Färbe Knoten des Graphen sodass benachbarte („adjazente“) Knoten unterschiedliche Farben haben.

Beobachtung: Form der Bundesländer ist zusammenhängend $\rightsquigarrow$ Kanten kreuzen sich nicht, dies ergibt **planare Graphen**.

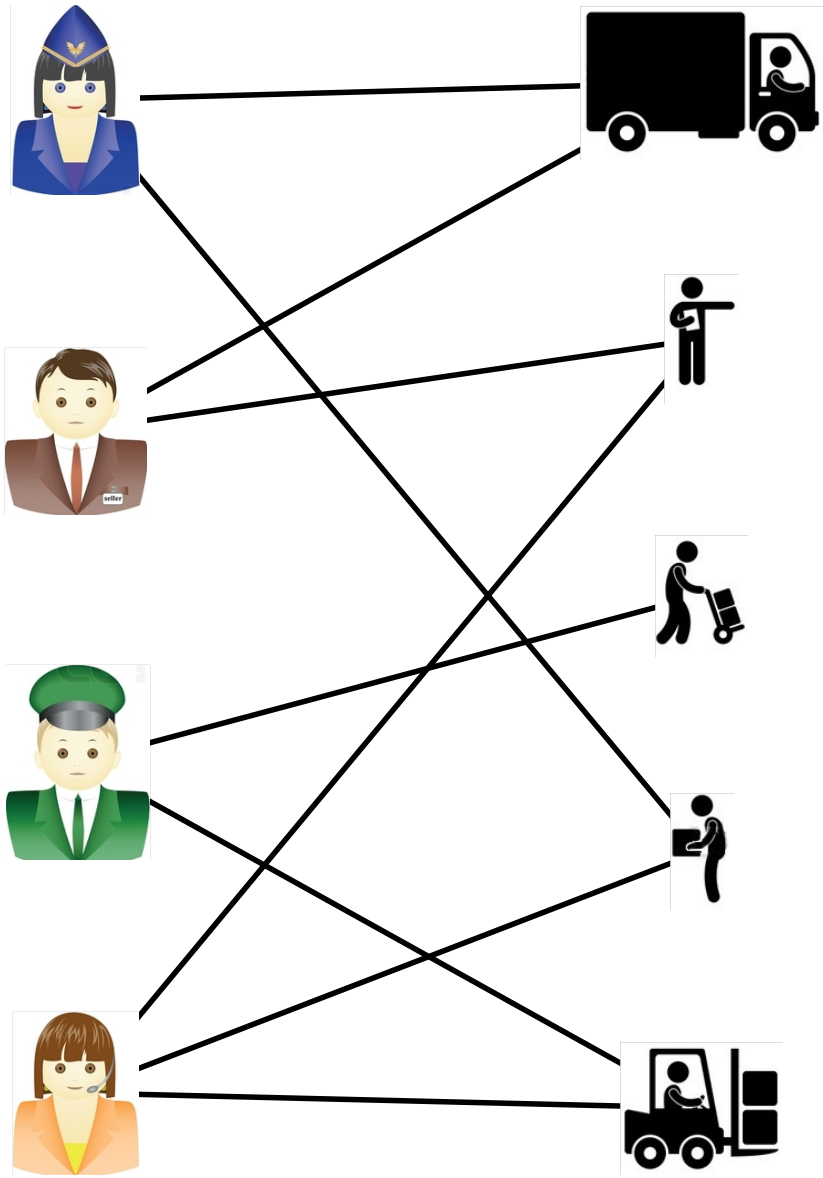
Vier-Farben-Satz: Vier Farben reichen um jeden planaren Graphen so zu färben, dass benachbarte Knoten unterschiedlich gefärbt sind.
Aufwändiger Beweis: Computergestützt! [Appel & Haken, 1976]

Zuordnungsproblem I

Problem: Unternehmen hat Mitarbeiter $m_1, m_2, \ldots, m_t$ und Aufträge $a_1, a_2, \ldots, a_s$ und weiß, ob Mitarbeiter m_i die Qualifikation hat Aufgabe a_j zu erfüllen.

Ziel: Finde Zuweisung von Aufgaben an Mitarbeiter, sodass möglichst viele Aufträge erfüllt werden.

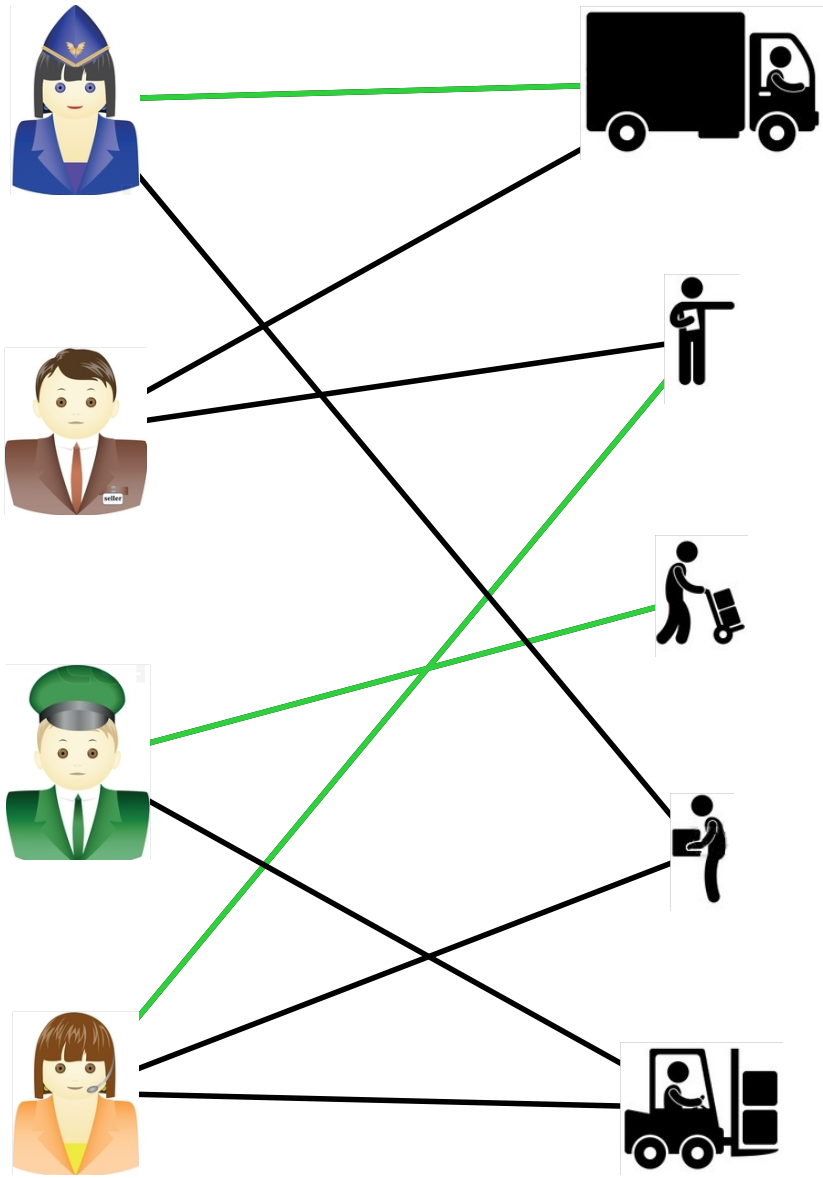
Annahme: Jeder Mitarbeiter kann nur einen Auftrag erfüllen und es braucht auch nur einen Mitarbeiter pro Auftrag.



Zuordnungsproblem II

Bipartites Graphproblem: Finde eine größtmögliche Kantenteilmenge M sodass kein Knoten Endpunkt von zwei oder mehr Kanten in M ist.

$\rightsquigarrow$ „**Bipartites Matching**“



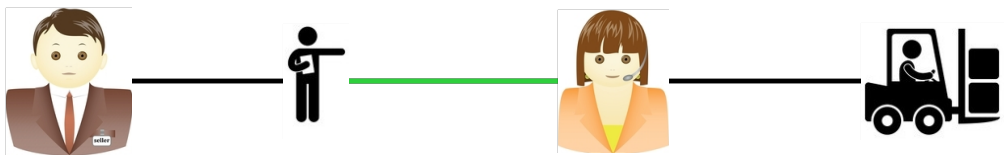
Zuordnungsproblem III

Lösungsansatz (Algorithmus) basiert auf folgender Beobachtung:

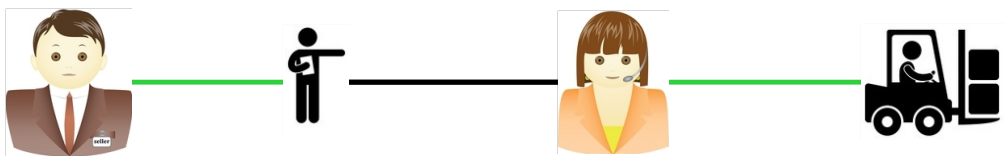
Satz [Berge 1957]: Eine Zuordnung (**Matching**) M ist größtmöglich genau dann, wenn es keinen **augmentierenden Pfad** gibt.

Definition: Ein Pfad zwischen zwei Knoten heißt *augmentierend*, wenn beide Knoten nicht Endpunkt einer Kante in M sind und die Kanten des Pfades abwechselnd aus M und $\overline{M}$ (der Komplementmenge von M) sind.

Augmentierender Pfad (aus vorherigem Bsp.):



kann verbessert werden zu:



(Passives) Virales Marketing I

Wikipedia: Beim passiven viralen Marketing verbreiten Nutzer die Nachricht **allein durch die Nutzung** des Produkts.

Beispiel: Der kostenlose Dienst Hotmail fügte an jede versendete Nachricht „P.S.: Get your private, free email at Hotmail“.
Idee: Bekommen Nutzer *viele* solcher E-Mails, werden sie auch zu Hotmail wechseln. $\rightsquigarrow$ Dominoeffekt

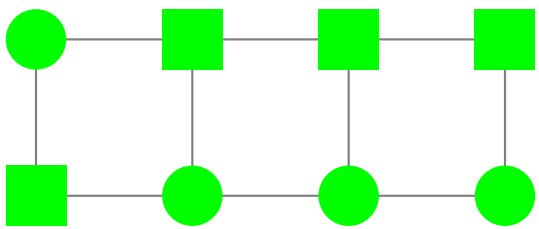
Es stellt sich die Frage: „Wer versendet die ersten E-Mails?“
Ziel: Möglichst wenige Benutzer auf herkömmlichen, teuren Weg von dem Produkt überzeugen, um möglichst viele Benutzer durch den „Dominoeffekt“ anzuwerben.

(Passives) Virales Marketing II

Modellierung als Graphproblem:

- Knoten $\hat{=}$ Personen
- Kanten $\hat{=}$ E-Mailkontakte zwischen Personen
- Knoten sind grün (nutzt Hotmail) und rot (nutzt Hotmail nicht) gefärbt; Zu Beginn sind alle Knoten rot.
- einfache Färbungsregel: Pro „Zeittakt" ändert ein roter Knoten seine Färbung zu grün falls die Mehrheit seiner Nachbarn grün ist.

Fragestellung: Färbe eine minimale Anzahl an Knoten grün, sodass nach wiederholter Anwendung obiger Färbungsregel alle Knoten grün gefärbt werden.



Virales Marketing III

Weitere Anwendungen obigen Modells:

- Verbreitung von ansteckenden Krankheiten:
- Fehlertoleranz in verteilten Systemen
- Verbreitung von Meinungen / Einstellungen in sozialen Netzwerken wie z.B. Facebook.

Beobachtung: Obiges Graphproblem ist also allgemein genug um in ganz verschiedenen Anwendungskontexten als Modell zu dienen...

Lobbying I

Natürlich sind nicht nur Graphen für die Modellbildung (Abstraktion) nützlich... Betrachte ein **Referendum zu mehreren Themen**:

Beispiel:

	Diplom statt Master	Direktwahl des Präsidenten	Internet-zensur
Wähler 1	✔	✔	✔
Wähler 2	✘	✔	✔
Wähler 3	✔	✘	✘
Wähler 4	✘	✔	✘
Wähler 5	✔	✘	✘
	⇓		
Ergebnis	✔(3:2)	✔(3:2)	✘(2:3)
Lobbyist	✘	✘	✔

Aufgabe:
Wähler so beeinflussen damit Ziel erreicht wird.

Triviale Lösung:
Mehr als Hälfte der Wähler beeinflussen.

Ziel:
Aus „Kostengründen“ so wenige Wähler wie möglich beeinflussen!

Lobbying II

Mathematische Modellierung als „Matrixmodifikationsproblem“:

	Diplom statt Master	Direktwahl des Präsidenten	Internet-zensur
Wähler 1	1	1	1
Wähler 2	0	1	1
Wähler 3	1	0	0
Wähler 4	0	1	0
Wähler 5	1	0	0
	⇓		
Ergebnis	1(3:2)	1 (3:2)	0 (3:2)
Lobbyist	0	0	1

Beobachtung:
Stimmabgaben haben nur **zwei** Werte: ✘ oder ✔.

Modellierung:
Referendum mit n Wählern und m Themen $\hat{=}$ binärer Matrix $\in \{0,1\}^{n \times m}$

Lobbying III

	Diplom statt Master	Direktwahl des Präsidenten	Internet-zensur
Wähler 1	1	1	1
Wähler 2	0	1	1
Wähler 3	1	0	0
Wähler 4	0	1	0
Wähler 5	1	0	0
	⇓		
Ergebnis	1(3:2)	1 (3:2)	0 (2:3)
Lobbyist	0	0	1

Problemstellung:
So **wenig Wähler wie möglich** beeinflussen um das Ziel des Lobbyisten zu erreichen.

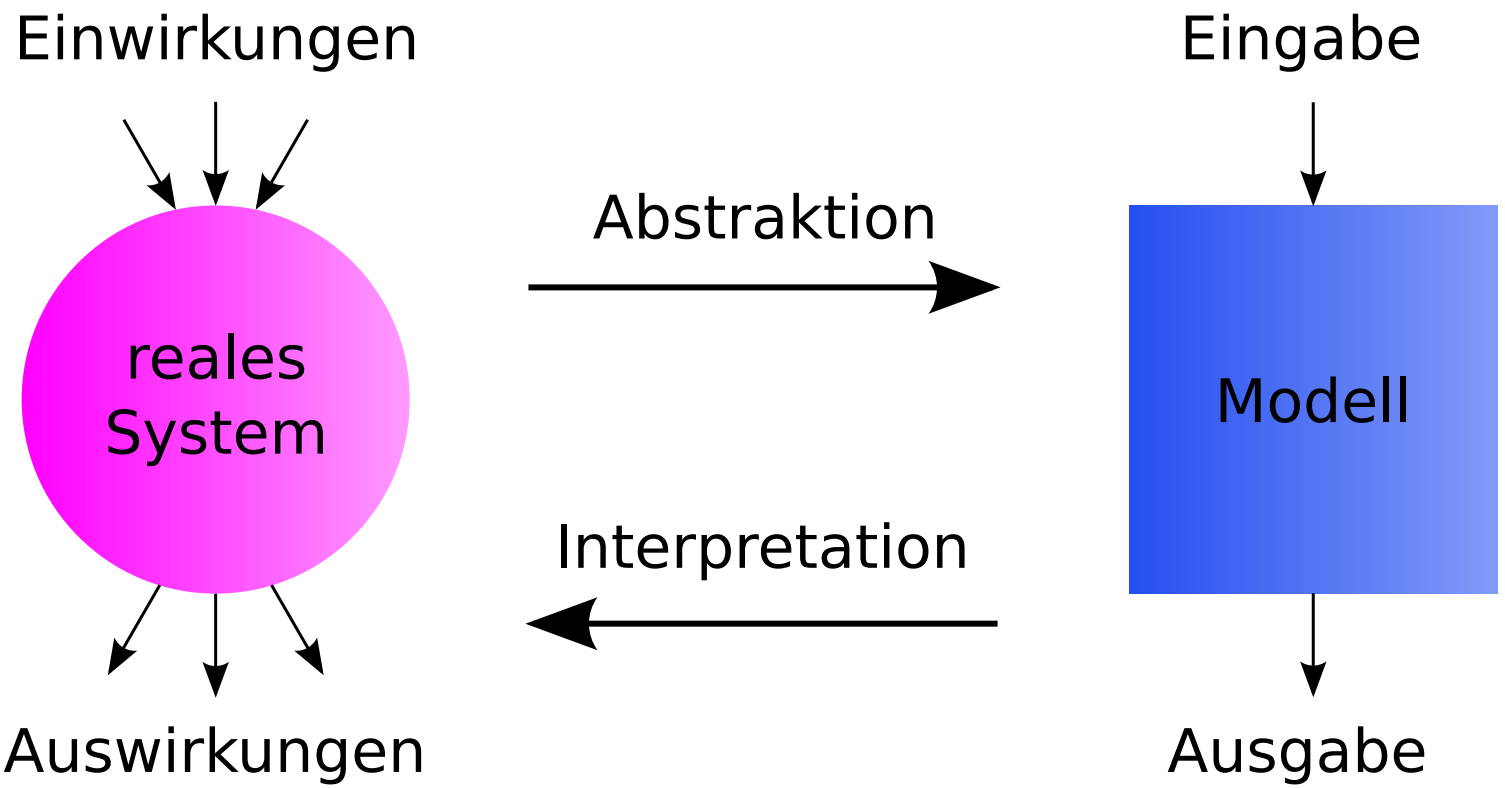
Bemerkung: Das Problem ist „**NP-schwer**“, d.h. es existiert wahrscheinlich kein effizienter Lösungsalgorithmus.

Bemerkung: In aktueller Forschung konnten wir einen Lobbying-Algorithmus entwerfen, der für den Spezialfall von höchstens vier abzustimmenden Themen das Problem effizient löst!

Fazit: Immer darauf achten, ob vielleicht auch Spezialfälle des Modells bereits ausreichend „reich“ für die Anwendung in der Praxis sind.

Fazit: Abstraktion in der Informatik

Allgemeines Bild:



Modellierung realer Strukturen nach Reinhard Wilhelm (1996)

Abstraktion beim Softwareentwurf

- Von der Maschinensprache zu Hochsprachen und Skriptsprachen ($\rightsquigarrow$ Abstraktionshierarchien).
- Modularität: Programmmodule.
- Datenabstraktion: Datentypen etc.
- Objektorientierte Konzepte: Zusammenspiel kooperierender „Objekte“ mit Attributen und Methoden...
- Wiederverwendung von Software; Softwarebibliotheken...
- Entwurfsmuster („design patterns“) zur Softwaremodellierung...
- ...

Abschließende Zitate zur Abstraktion in der Informatik

„Computer Science and Engineering is a field that attracts a different kind of thinker. I believe that one who is a natural computer scientist thinks algorithmically. Such people are especially good at dealing with situations where different rules apply in different cases; they are individuals who can rapidly change levels of abstraction, simultaneously seeing things 'in the large' and 'in the small'." – Donald E. **Knuth**

„One of the defining characteristics of computer science is the immense difference in scale of the phenomena computer science deals with. From the individual bits of program and data in the computers to billions of operations per second on this information by the highly complex machines, their operating systems and the various languages in which the problems are described, the scale changes through many orders of magnitude" . – Juris **Hartmanis**

„We all know that the only mental tool by means of which a very finite piece of reasoning can cover a myriad cases is called 'abstraction'; as a result the effective exploitation of her/his powers of abstraction must be regarded as one of the most vital activities of a competent programmer." – Edsger W. **Dijkstra**