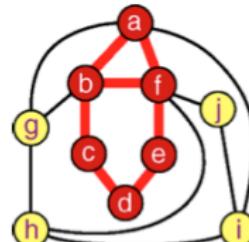
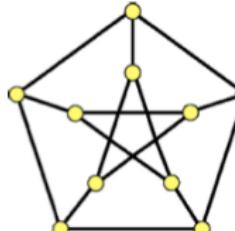
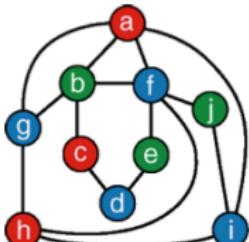


Algorithmische Graphentheorie

Vorlesung 1: Einführung in die Graphentheorie

Babeş-Bolyai Universität, Department für Informatik, Cluj-Napoca
csacarea@cs.ubbcluj.ro



GRAPHENTHEORIE

Vorbemerkungen

- Die Vorlesung wird **folienbasiert** gehalten;
- Die Folien enthalten **nur die wichtigsten Aspekte** (Definitionen, Sätze, knappe Beispiele, wichtige Bemerkungen);
- Alles was sonst eine Vorlesung ausmacht (Erläuterungen, ausführliche Beispiele, Beweise von Sätzen , Anwendungen, Querverweise auf andere Gebiete der Informatik, etc.) gibt es nur in der Vorlesung selbst.

Sprechstunden:

nach Vereinbarung



LITERATUR

Slides benutzen Lehrmaterial aus folgenden Lehrbücher:

- V. Turau - Algorithmische Graphentheorie
- R. Sedgewick - Algorithms
- V.K. Balakrishnan - Graph Theory
- R. Sedgewick - Algorithms in C++
- D. Joyner, M. Van Nguyen, N. Cohen, Algorithmic Graph Theory
- Nützlich wäre auch SAGE



GRAPHENTHEORIE

WIEDER EINE THEORIE???

Q: Wann fangen wir endlich an mit Programmieren?

A: Früh genug...

Q: Wozu ist denn diese Graphentheorie nützlich?

A:

- Straßen- und Verkehrsnetze;
- Computernetzwerke;
- elektrische Schaltpläne;
- Entity-Relationship Diagramme;
- Beweisbäume;
- endliche Automaten;
- Syntaxbäume für Programmiersprachen;
- Entscheidungsbäume;
- Petri-Netze und weitere Modellierungssprachen.

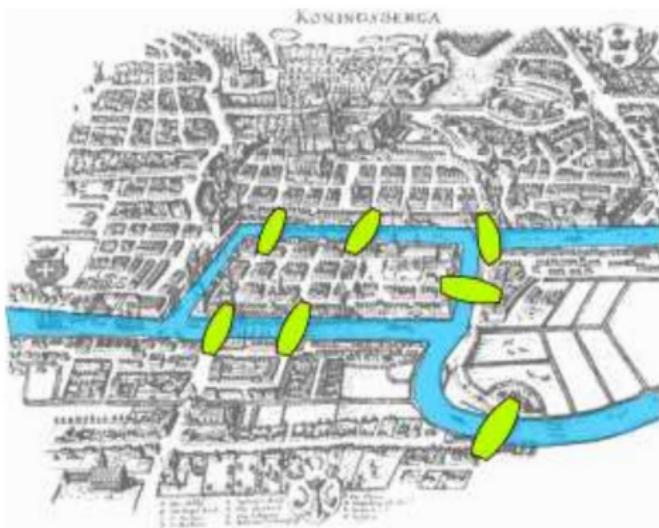


EINFÜHRUNG

Das Königsberger Brückenproblem

Beispiel 1.1. [Euler, 1736]

Gibt es einen Rundweg durch Königsberg, der jede der sieben Brücken genau einmal überquert?

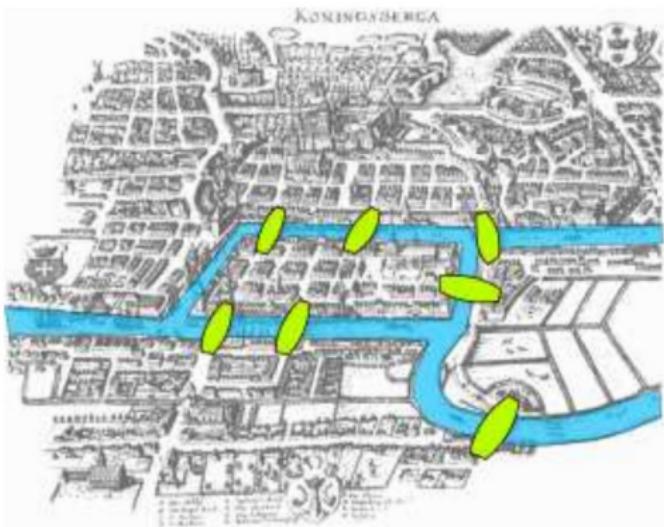


EINFÜHRUNG

Das Königsberger Brückenproblem

Beispiel 1.1. [Euler, 1736]

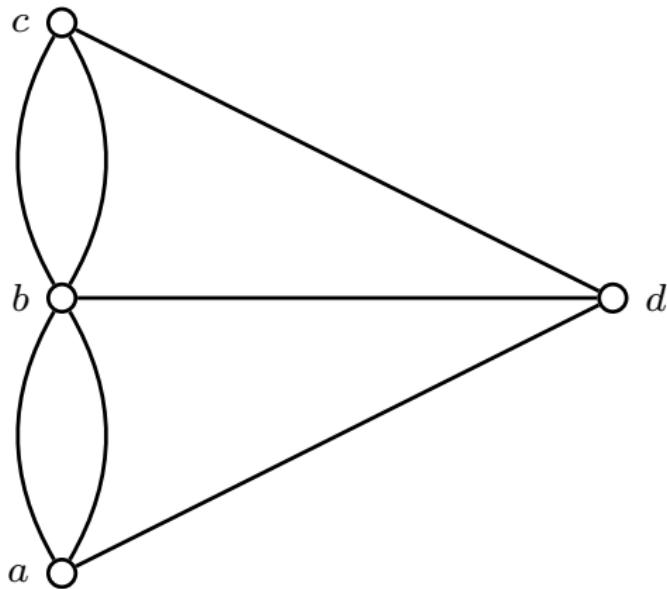
Gibt es einen Rundweg durch Königsberg, der jede der sieben Brücken genau einmal überquert?



Brückenproblem

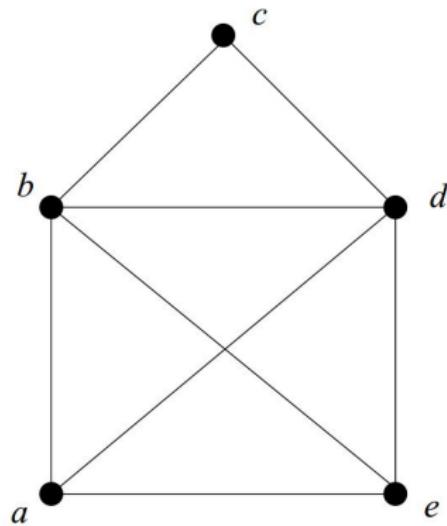


EINFÜHRUNG



EIN LUSTIGES BEISPIEL

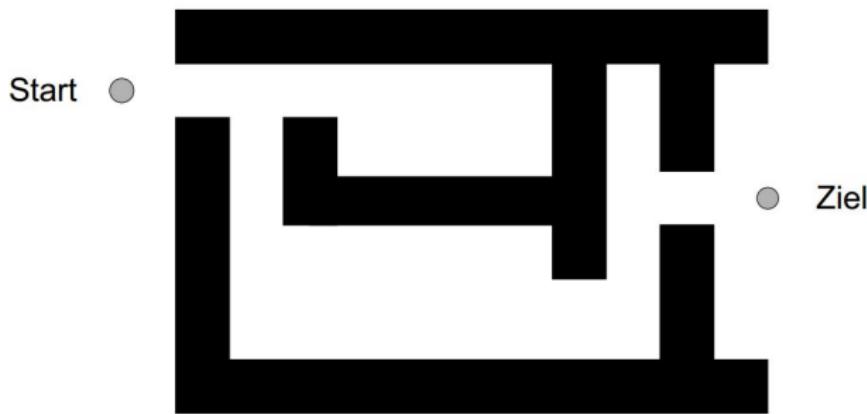
Das Haus vom Nikolaus



NOCH EIN BEISPIEL

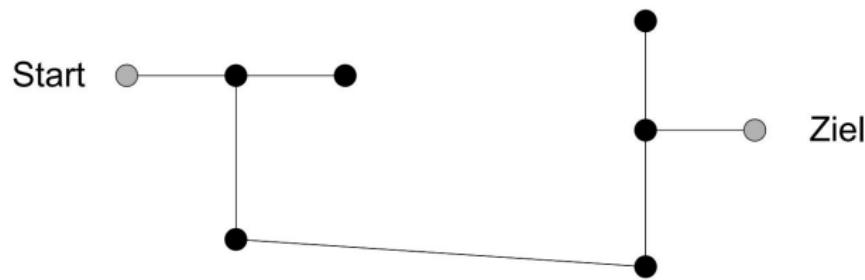
Labyrinth

Beispiel 1.2. Finde einen Weg vom *Start* zum *Ziel* durch das Labyrinth!



UND JETZT ALS GRAPH DARGESTELLT

Repräsentation als Graph:



EINLEITUNG

- In vielen praktischen und theoretischen Anwendungen treten Situationen auf, die durch ein System von Objekten und Beziehungen zwischen diesen Objekten charakterisiert werden können.
- Die Graphentheorie stellt zur Beschreibung von solchen Systemen ein Modell zur Verfügung: den Graphen.
- Die problemunabhängige Beschreibung mittels eines Graphen lässt die Gemeinsamkeit von Problemen aus den verschiedensten Anwendungsgebieten erkennen.



EINLEITUNG

- Die Graphentheorie ermöglicht somit die Lösung vieler Aufgaben, welche aus dem Blickwinkel der Anwendung keine Gemeinsamkeiten haben.
- Die algorithmische Graphentheorie stellt zu diesem Zweck Verfahren zur Verfügung, die problemunabhängig formuliert werden können.
- Ferner erlauben Graphen eine anschauliche Darstellung, welche die Lösung von Problemen häufig erleichtert.



BEISPIEL: VERFOLGUNG UND AUSFLUCHT SPIELE

2 Spieler Spiele

Ein Team von mobilen Agenten (**Polizisten**) verfolgen einen anderen mobilen Agent (**Räuber**).

Es gibt immer nur ein Gewinner.

- Combinatorisches Problem: Minimiere die Anzahl der notwendigen Ressourcen, um das Spiel zu gewinnen. Z.B. **minimiere die Anzahl der Polizisten**, die notwendig sind um den Räuber zu fangen.
- Algorithmisches Problem: **Berechne eine Gewinn-Strategie** (Folge von Spielzüge) für jeden Spieler. Z.B. berechne die Erfolgsstrategie für die Polizisten, um den Räuber zu fangen/Räuber entwischt.



BEISPIEL: VERFOLGUNG UND AUSFLUCHT SPIELE

Anwendungen:

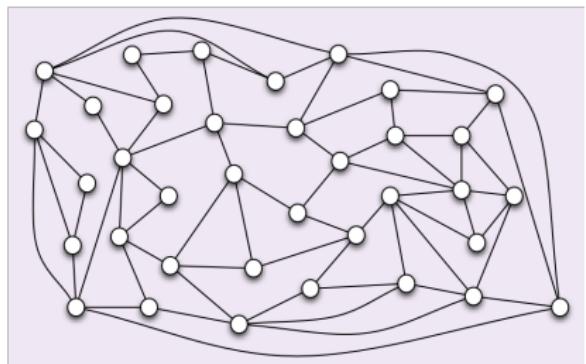
Robotik, Netzwerk Sicherheit, Information Retrieval, Graphen
Theorie, Modellierung, Logik, Routing, ...



POLIZISTEN UND RÄUBER SPIELE

[NOWAKOWSKI AND WINKLER; QUILLIOT, 1983]

Spielregeln

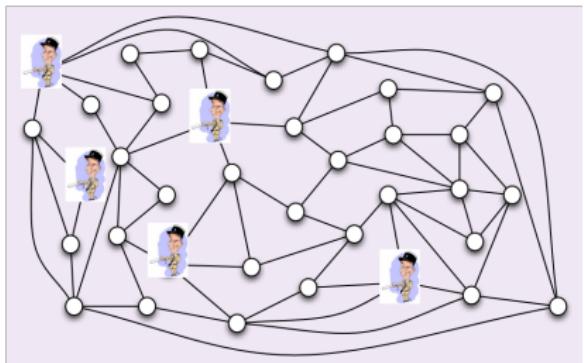


POLIZISTEN UND RÄUBER SPIELE

[NOWAKOWSKI AND WINKLER; QUILLIOT, 1983]

Spielregeln:

- 1 Platziere $k \geq 1$ Polizisten in den Knoten

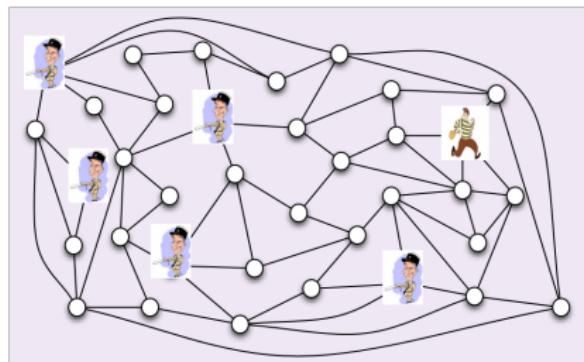


POLIZISTEN UND RÄUBER SPIELE

[NOWAKOWSKI AND WINKLER; QUILLIOT, 1983]

Spielregeln:

- ① Platziere $k \geq 1$ Polizisten in den Knoten
 - ② Einen Räuber \mathcal{R} in einem Knoten

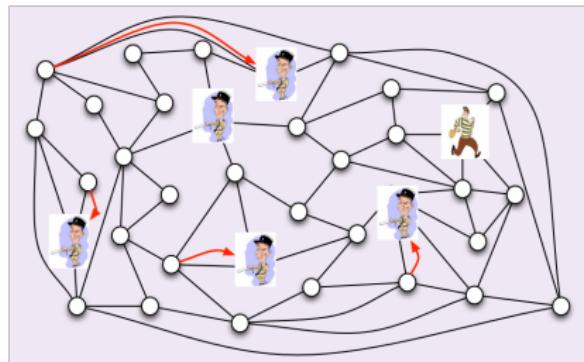


POLIZISTEN UND RÄUBER SPIELE

[NOWAKOWSKI AND WINKLER; QUILLIOT, 1983]

Spielregeln:

- ① Platziere $k \geq 1$ Polizisten in den Knoten
 - ② Einen Räuber \mathcal{R} ein einem Knoten
 - ③ Zug bei Zug
 - ④ Jeder Polizist \mathcal{P} rückt entlang einer Kante

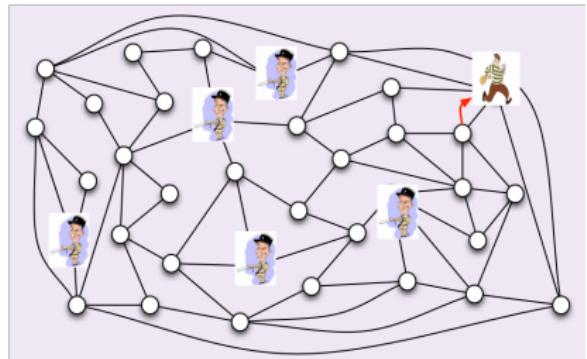


POLIZISTEN UND RÄUBER SPIELE

[NOWAKOWSKI AND WINKLER; QUILLIOT, 1983]

Spielregeln:

- ① Platziere $k \geq 1$ Polizisten in den Knoten
 - ② Einen Räuber \mathcal{R} ein einem Knoten
 - ③ Zug bei Zug
 - ① Jeder Polizist \mathcal{P} rückt entlang höchstens einer Kante
 - ② Der Räuber \mathcal{R} rückt ebenfalls entlang höchstens einer Kante

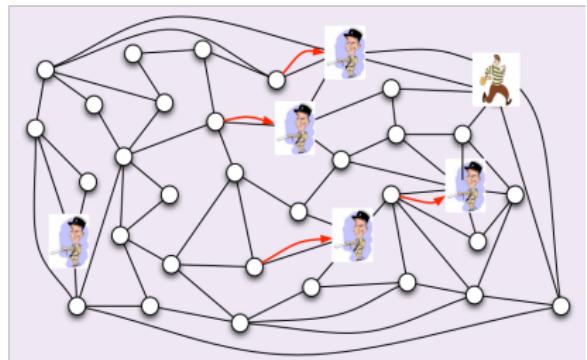


POLIZISTEN UND RÄUBER SPIELE

[NOWAKOWSKI AND WINKLER; QUILLIOT, 1983]

Spielregeln:

- ① Platziere $k \geq 1$ Polizisten in den Knoten
 - ② Einen Räuber \mathcal{R} ein einem Knoten
 - ③ Zug bei Zug
 - ① Jeder Polizist \mathcal{P} rückt entlang höchstens einer Kante
 - ② Der Räuber \mathcal{R} rückt ebenfalls entlang höchstens einer Kante

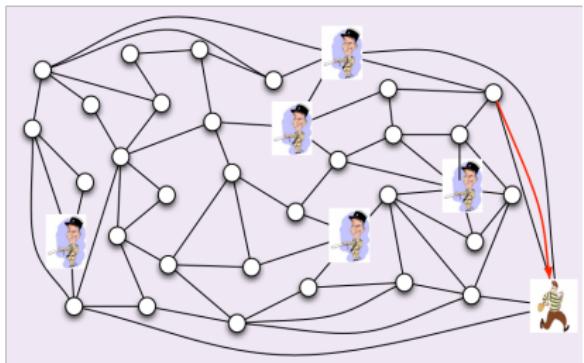


POLIZISTEN UND RÄUBER SPIELE

[NOWAKOWSKI AND WINKLER; QUILLIOT, 1983]

Ziel des Spiels:

- Räuber muss den Polizisten entwischen

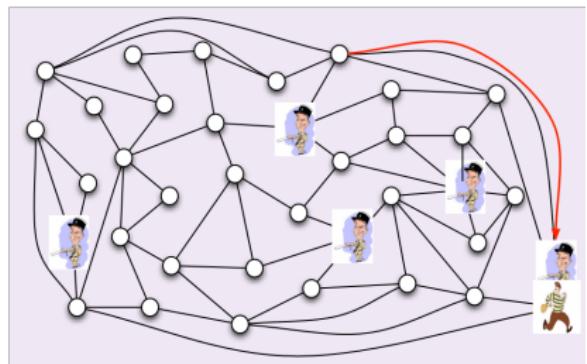


POLIZISTEN UND RÄUBER SPIELE

[NOWAKOWSKI AND WINKLER; QUILLIOT, 1983]

Ziel des Spiels:

- Räuber muss den Polizisten entwischen
 - Polizisten müssen den Räuber erwischen (d.h. sich im selben Knoten befinden)



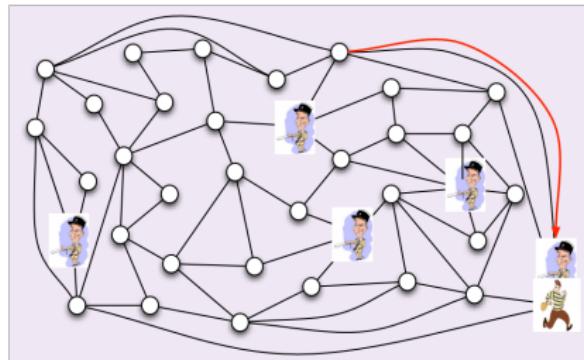
POLIZISTEN UND RÄUBER SPIELE

[NOWAKOWSKI AND WINKLER; QUILLIOT, 1983]

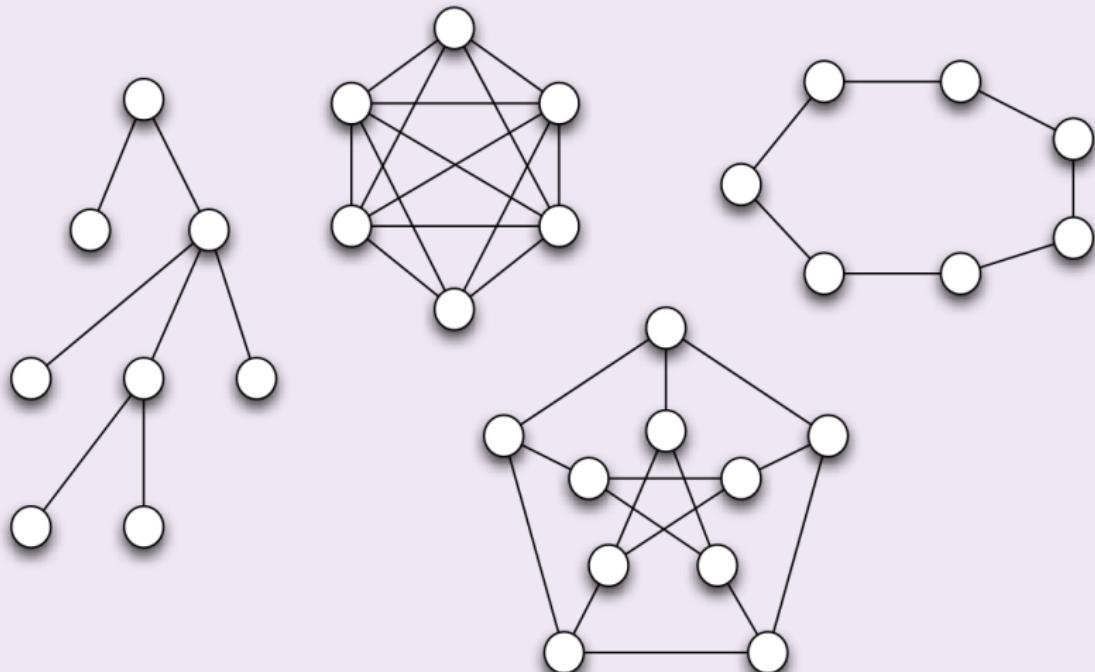
Ziel des Spiels:

- Räuber muss den Polizisten entwischen
 - Polizisten müssen den Räuber erwischen (d.h. sich im selben Knoten befinden)

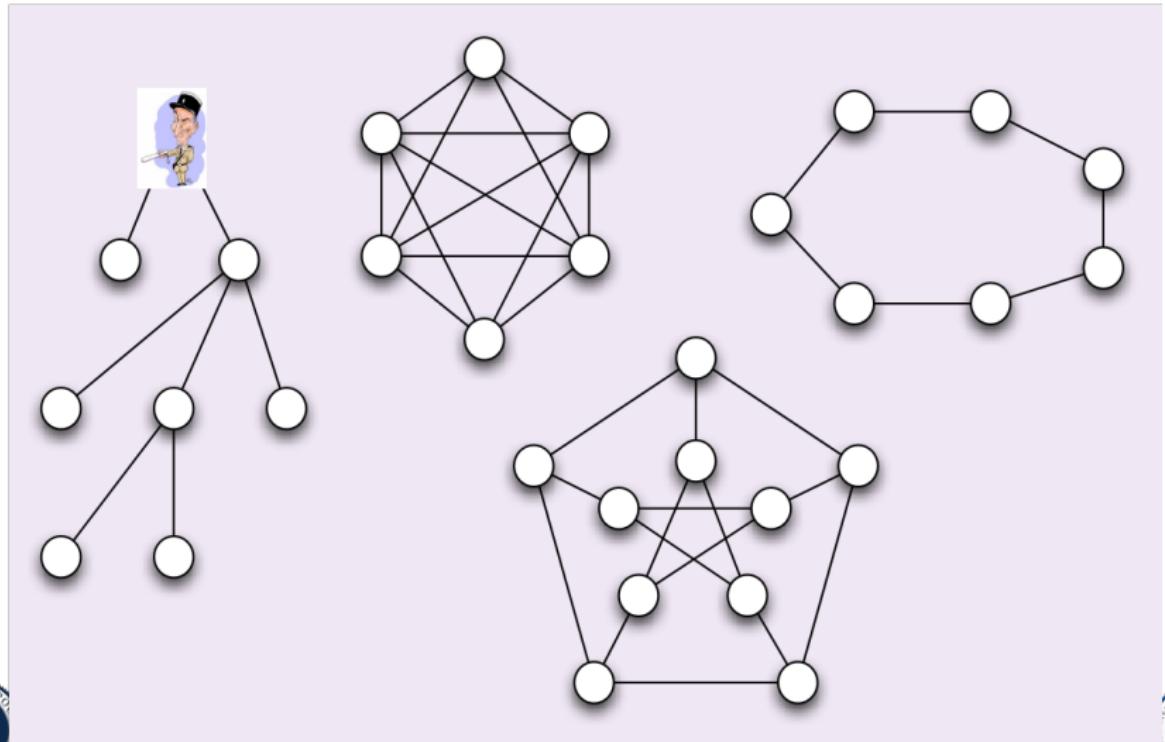
Cop Number eines Graphen
 $cn(G)$: minimale Anzahl von
Polizisten, um zu gewinnen.



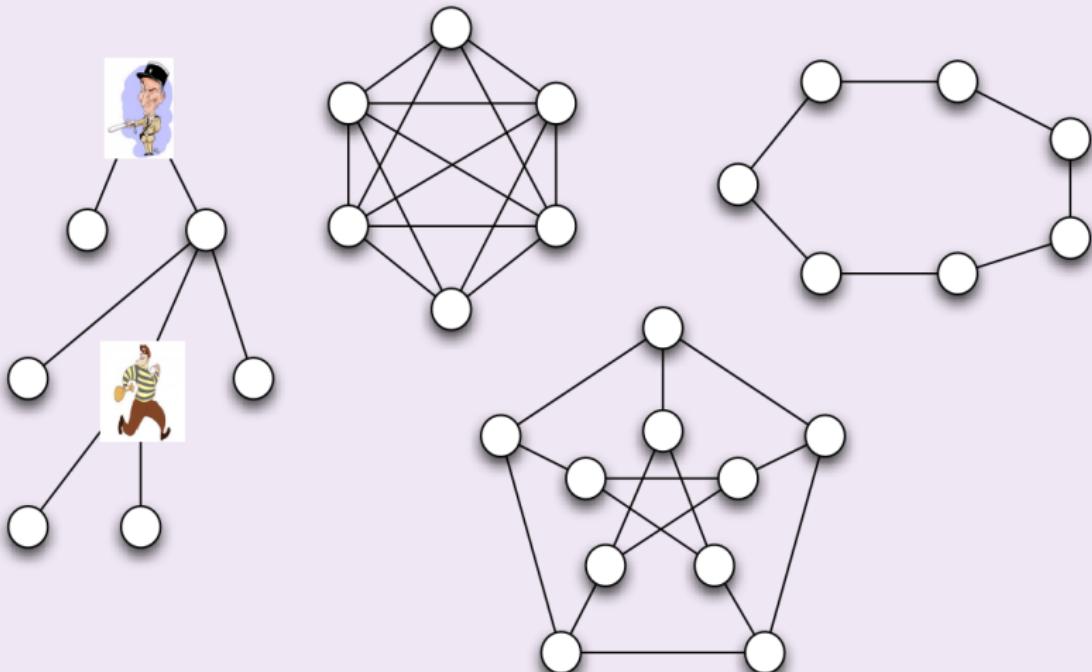
LASST UNS SPIELEN



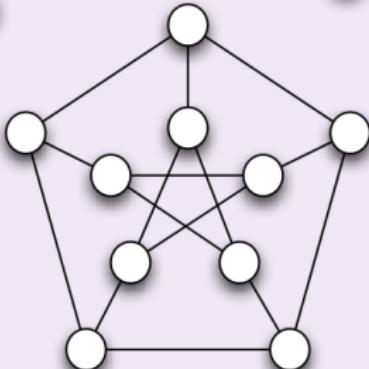
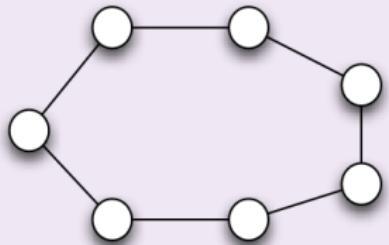
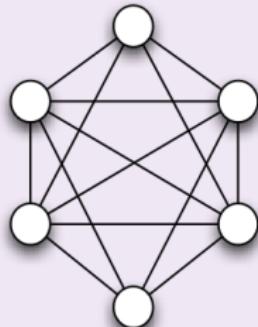
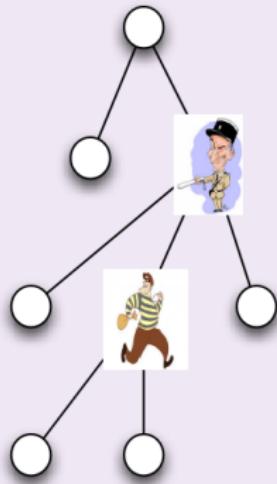
LASST UNS SPIELEN



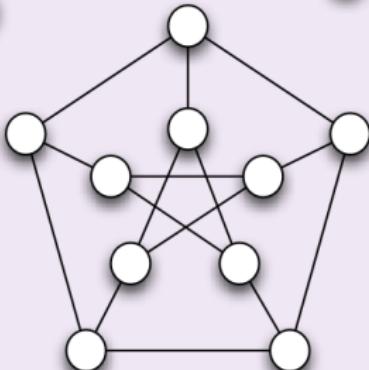
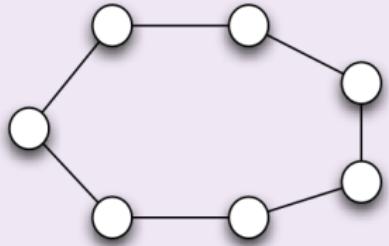
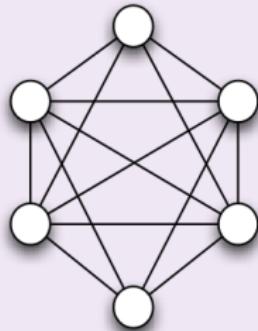
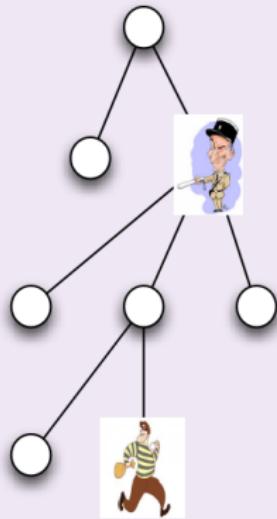
LASST UNS SPIELEN



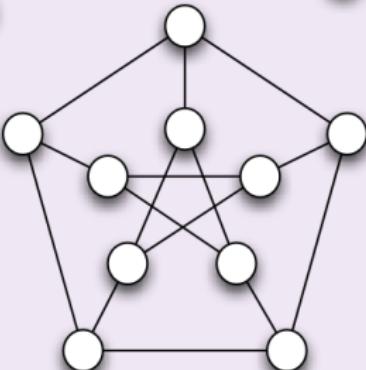
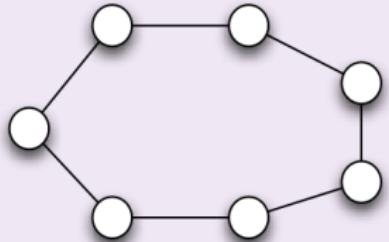
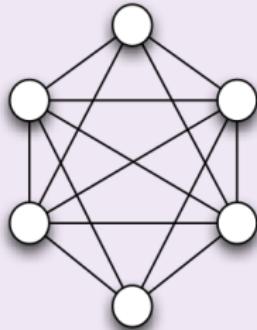
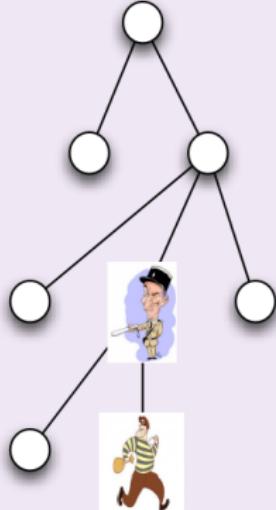
LASST UNS SPIELEN



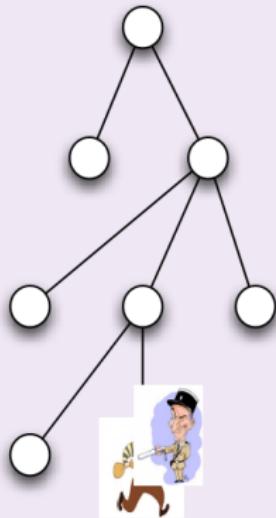
LASST UNS SPIELEN



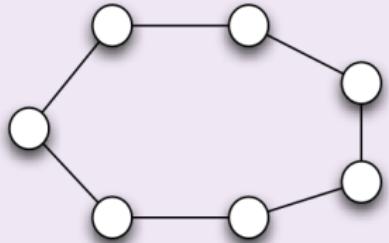
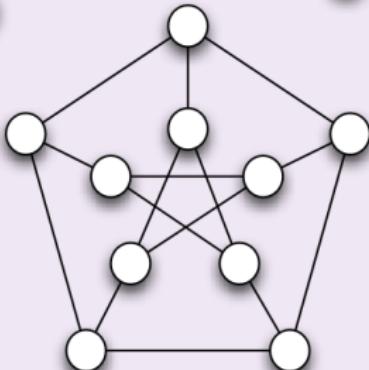
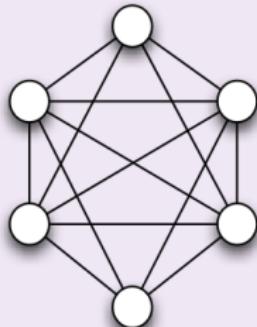
LASST UNS SPIELEN



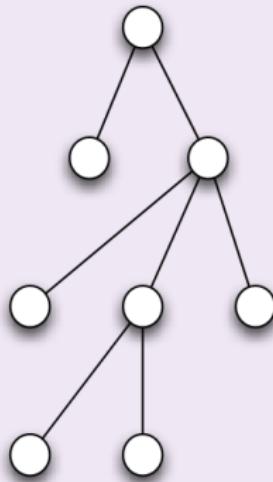
LASST UNS SPIELEN



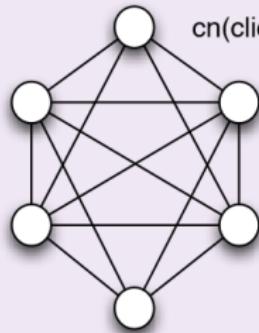
$$cn(tree)=1$$



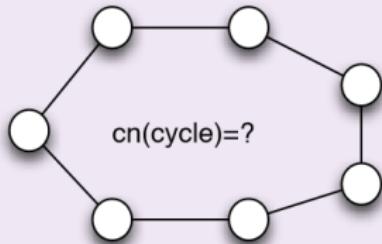
LASST UNS SPIELEN



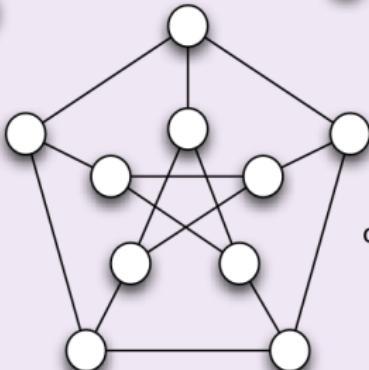
$$cn(tree)=1$$



$cn(\text{clique})=?$



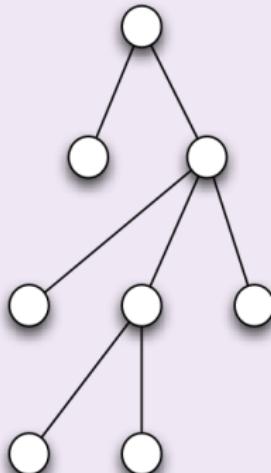
$cn(\text{cycle})=?$



$\text{cn}(\text{Petersen})=?$



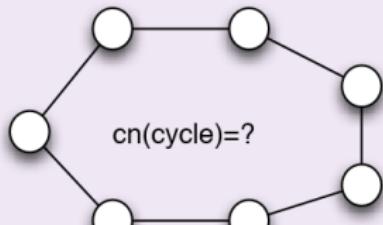
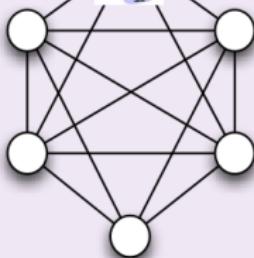
LASST UNS SPIELEN



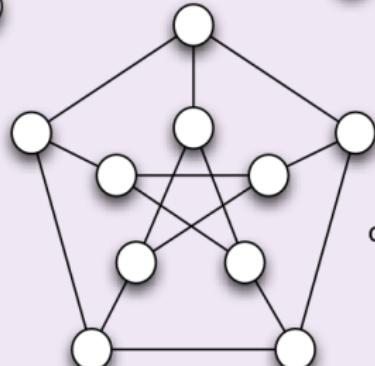
$\text{cn}(\text{tree})=1$



$\text{cn}(\text{clique})=?$



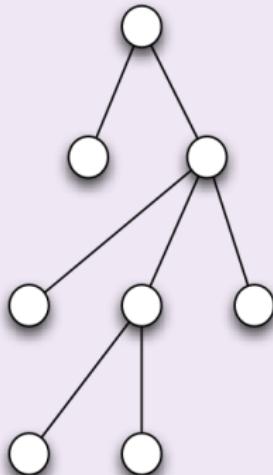
$cn(\text{cycle})=?$



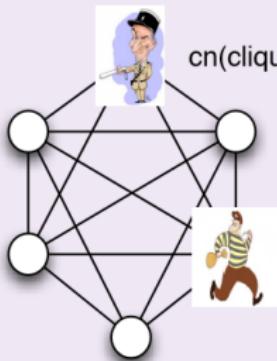
$\text{cn}(\text{Petersen})=?$



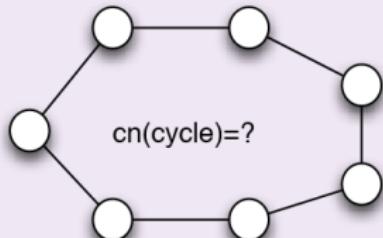
LASST UNS SPIELEN



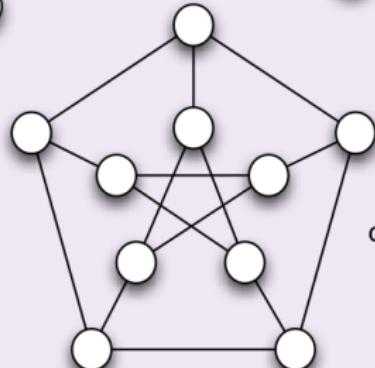
$cn(tree)=1$



$\text{cn}(\text{clique})=?$



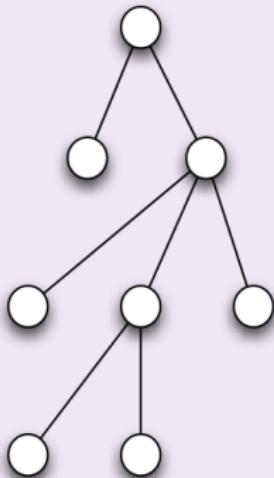
$cn(\text{cycle})=?$



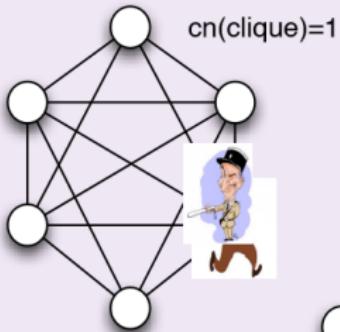
$\text{cn}(\text{Petersen})=?$



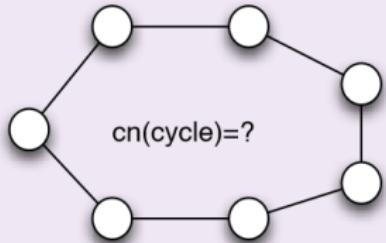
LASST UNS SPIELEN



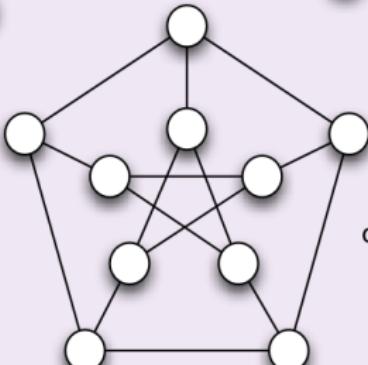
$cn(tree)=1$



$cn(clique)=1$



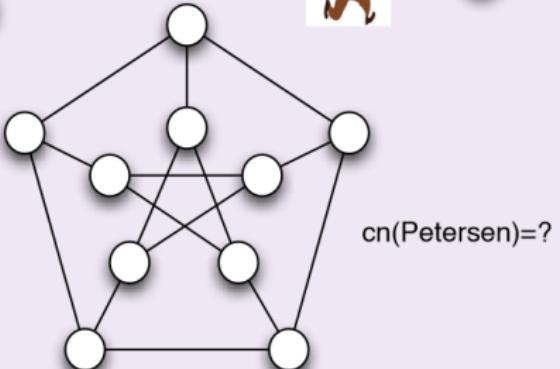
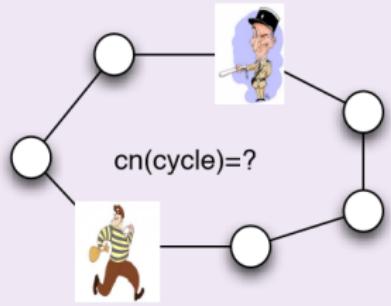
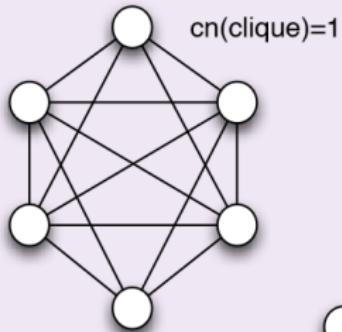
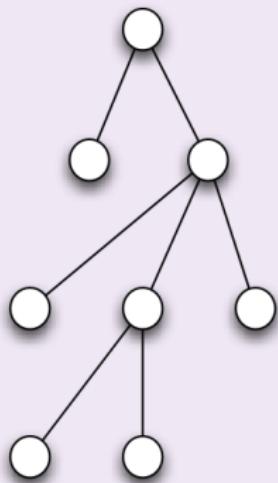
$cn(cycle)=?$



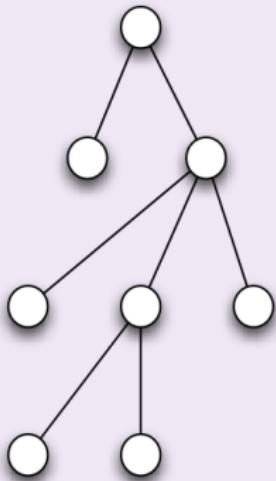
$cn(Petersen)=?$



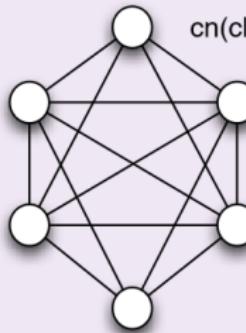
LASST UNS SPIELEN



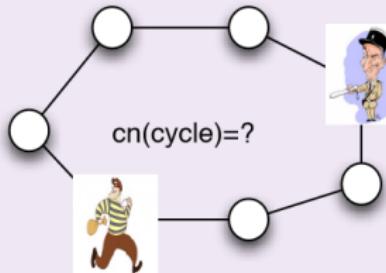
LASST UNS SPIELEN



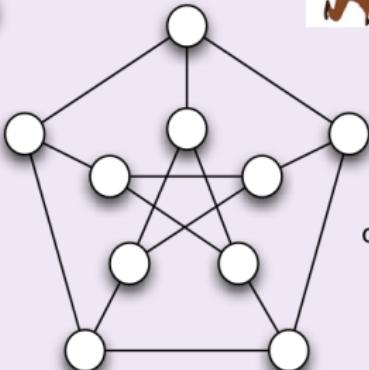
$cn(tree)=1$



$cn(clique)=1$



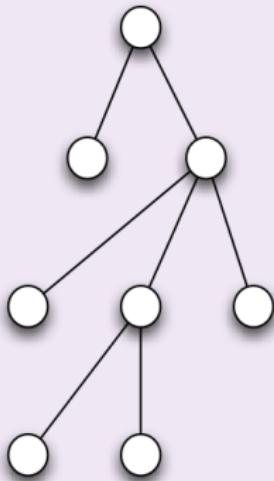
$cn(cycle)=?$



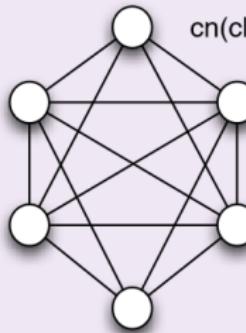
$cn(Petersen)=?$



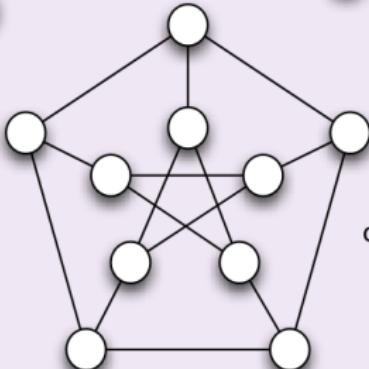
LASST UNS SPIELEN



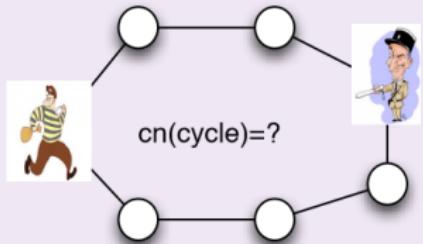
$cn(tree)=1$



$cn(clique)=1$



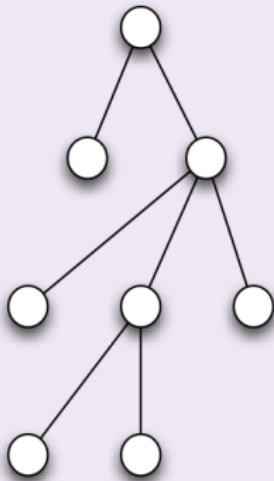
$cn(Petersen)=?$



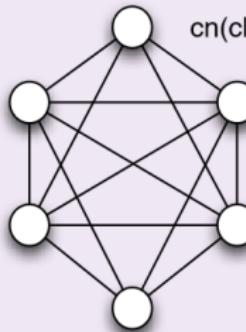
$cn(cycle)=?$



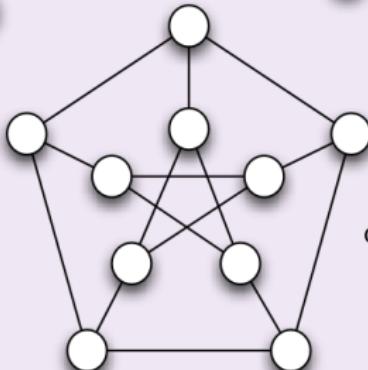
LASST UNS SPIELEN



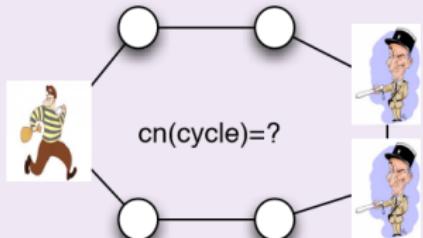
$cn(tree)=1$



$cn(clique)=1$



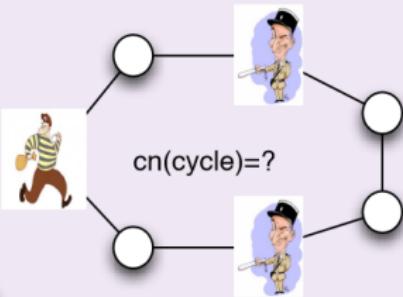
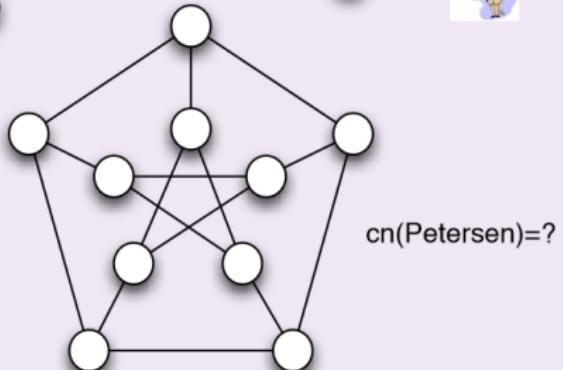
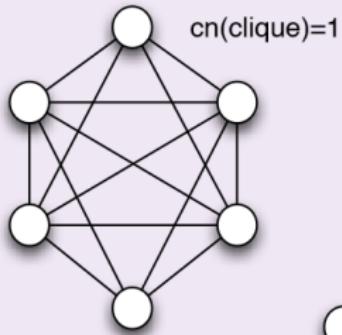
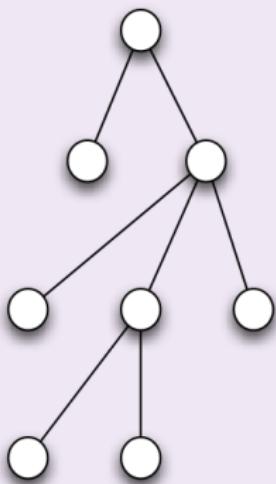
$cn(Petersen)=?$



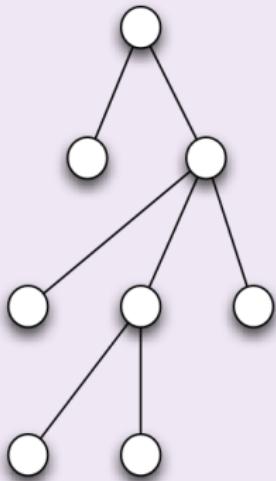
$cn(cycle)=?$



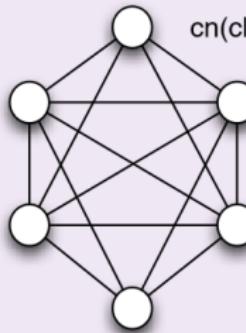
LASST UNS SPIELEN



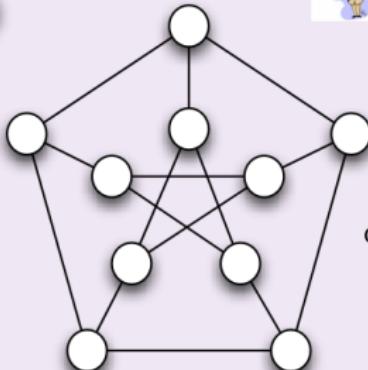
LASST UNS SPIELEN



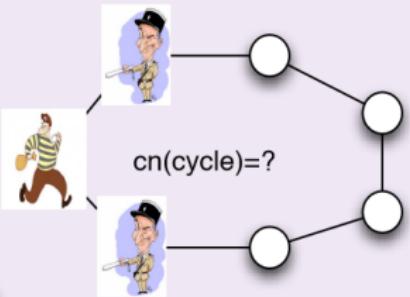
$cn(tree)=1$



$cn(clique)=1$



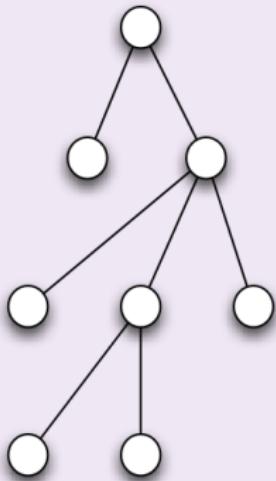
$cn(Petersen)=?$



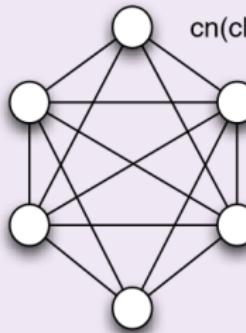
$cn(cycle)=?$



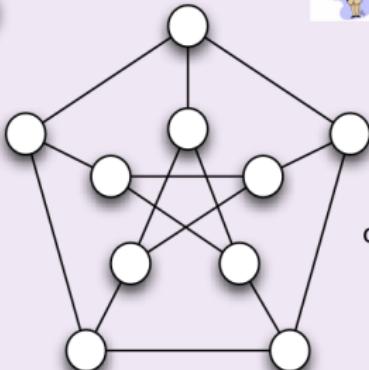
LASST UNS SPIELEN



$cn(tree)=1$



$cn(clique)=1$



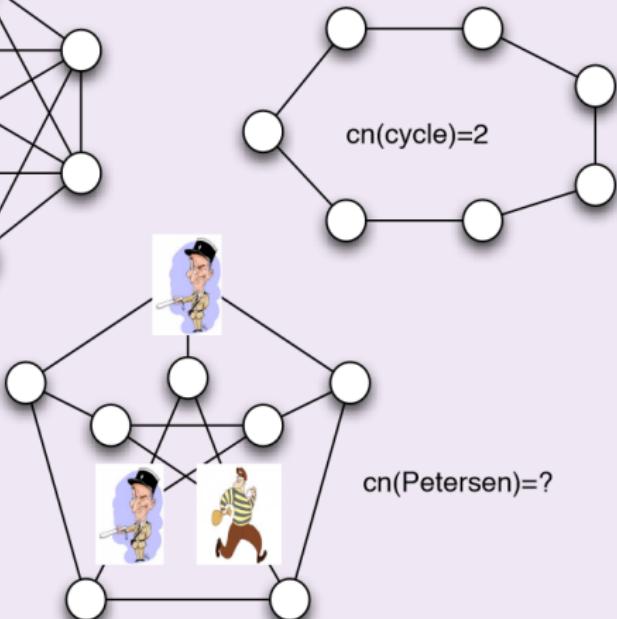
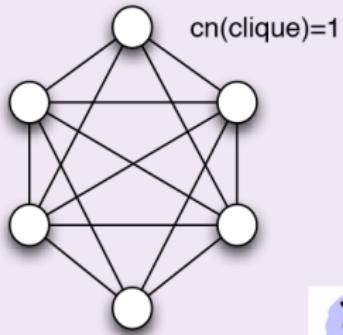
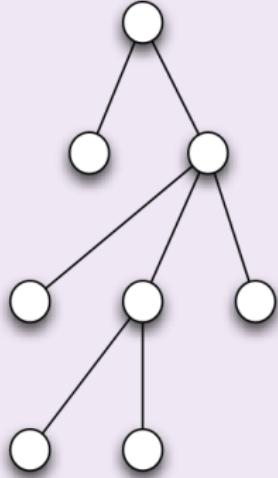
$cn(Petersen)=?$



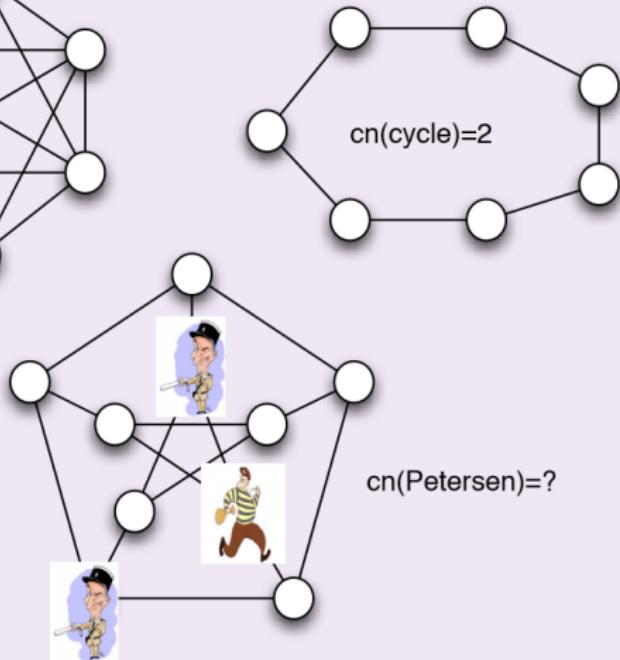
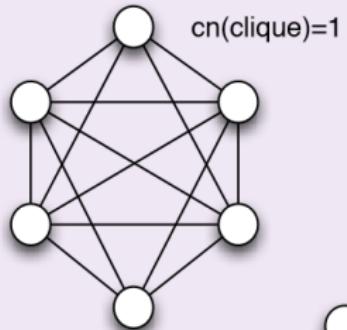
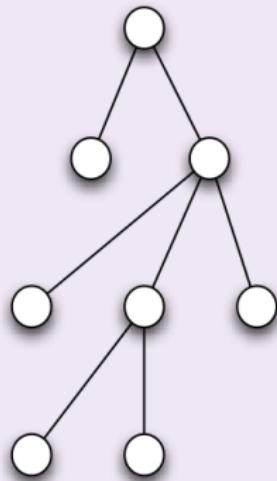
$cn(cycle)=2$



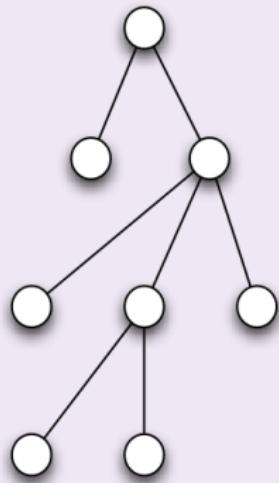
LASST UNS SPIELEN



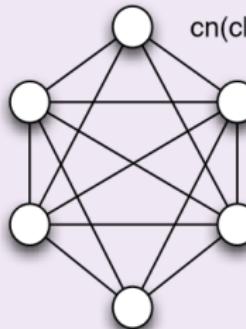
LASST UNS SPIELEN



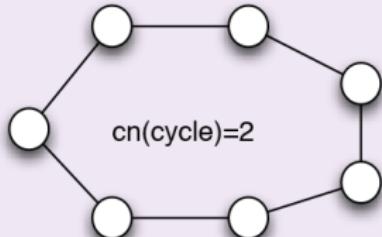
LASST UNS SPIELEN



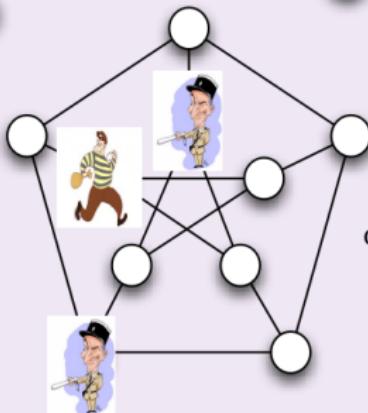
$cn(tree)=1$



$cn(clique)=1$



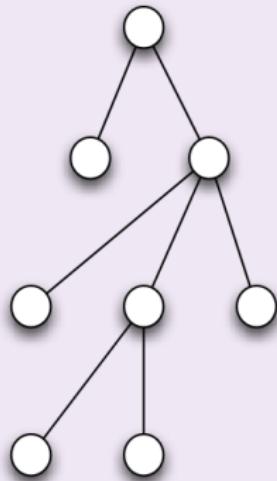
$cn(cycle)=2$



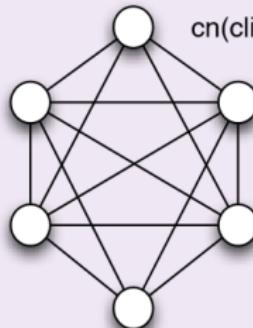
$cn(Petersen)=?$



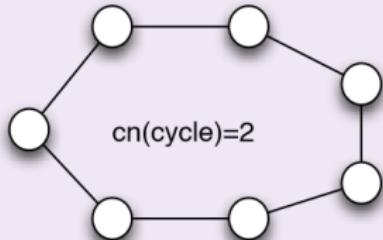
LASST UNS SPIELEN



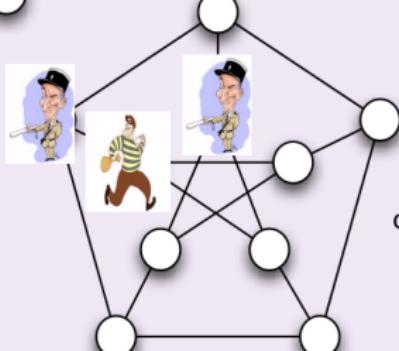
$cn(tree)=1$



$$cn(\text{clique})=1$$



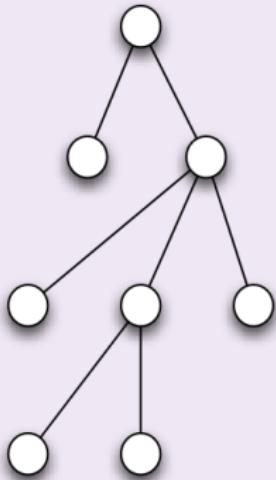
$\text{cn}(\text{cycle})=2$



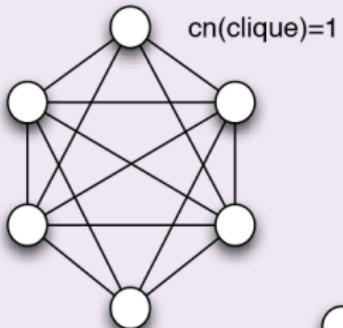
$\text{cn}(\text{Petersen})=?$



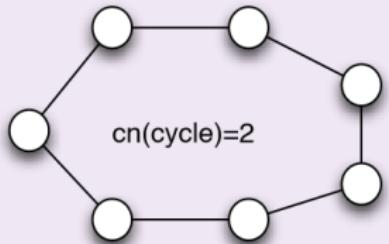
LASST UNS SPIELEN



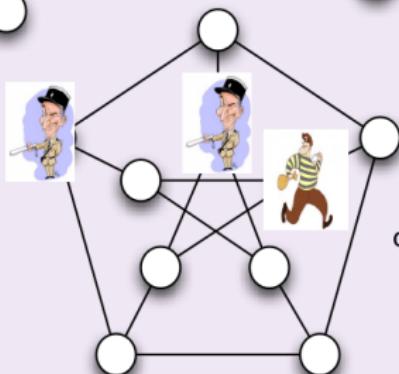
$cn(tree)=1$



$cn(clique)=1$



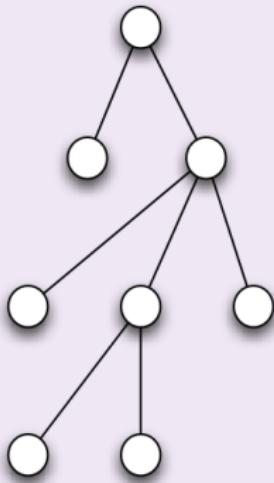
$cn(cycle)=2$



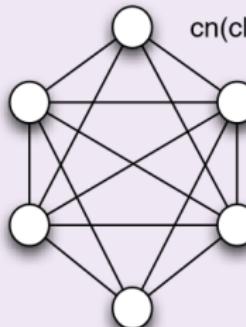
$cn(Petersen)=?$



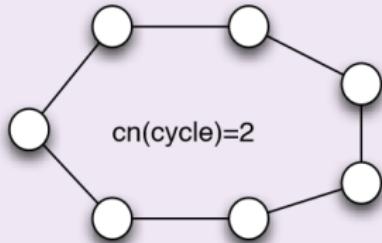
LASST UNS SPIELEN



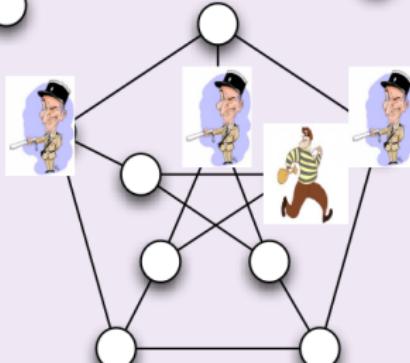
$cn(tree)=1$



$$cn(\text{clique})=1$$



$\text{cn}(\text{cycle})=2$

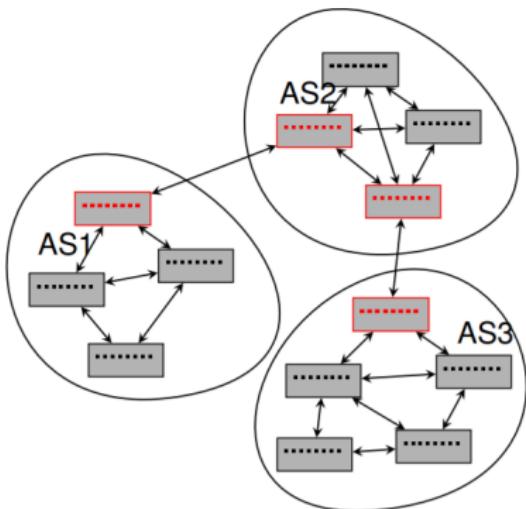


$\text{cn}(\text{Petersen})=3$



BEISPIEL: VERLETZLICHKEIT DER KOMMUNIKATIONSNETZE

- Ein Kommunikationsnetz ist ein durch Datenübertragungswege realisierter Verband mehrerer Rechner.
 - Es unterstützt den Informationsaustausch zwischen Benutzern an verschiedenen Orten.



VERLETZLICHKEIT DER KOMMUNIKATIONSNETZE

Verletzlichkeit

Die **Verletzlichkeit** eines Kommunikationsnetzes ist durch die Anzahl von Leitungen oder Rechnern gekennzeichnet, die ausfallen müssen, damit die Verbindung zwischen zwei beliebigen Benutzern nicht mehr möglich ist.

Häufig ist eine Verbindung zwischen zwei Benutzern über mehrere Wege möglich. Somit ist beim Ausfall einer Leitung oder einer Station die Verbindung nicht notwendigerweise unterbrochen.



VERLETZLICHKEIT DER KOMMUNIKATIONSNETZE

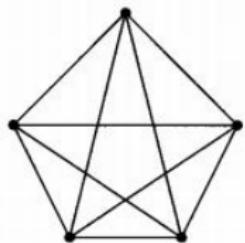
Welche Netzwerke sind verletzlicher?

Ein Netzwerk, bei dem schon der Ausfall einer einzigen Leitung oder Station gewisse Verbindungen **unmöglich** macht ist verletzlicher, als ein solches, wo dies nur beim Ausfall von mehreren Leitungen oder Stationen möglich ist.

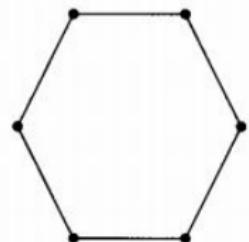
- Die **minimale Anzahl** von Leitungen und Stationen, deren Ausfall die Funktion des Netzwerkes beeinträchtigt, hängt sehr stark von der Beschaffenheit des Netzwerkes ab.
- Netzwerke lassen sich graphisch durch Knoten und Verbindungslinien zwischen den Knoten darstellen: Die Knoten entsprechen den Stationen, die Verbindungslinien den Datenübertragungswegen.



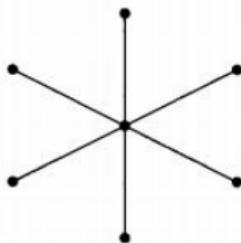
NETZWERKTOPOLOGIEN



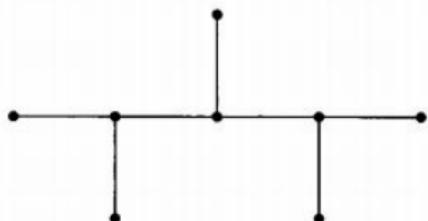
Vermischte Struktur



Ringstruktur



Sternstruktur



Busstruktur



ROUTING

- Das Internet gliedert sich in Bereiche unterschiedlicher administrativer Verantwortung (z.B. Verantwortung eines ISPs), sog. autonome Systeme (AS).
- Es muss unterschieden werden zwischen Routing innerhalb eines AS (intra-AS) und zwischen verschiedenen AS (inter-AS), da hier unterschiedliche Anforderungen an den Austausch von Routinginformation gestellt werden.
- Die Knoten an den Verbindungsstellen zwischen AS heißen Gateway Router.



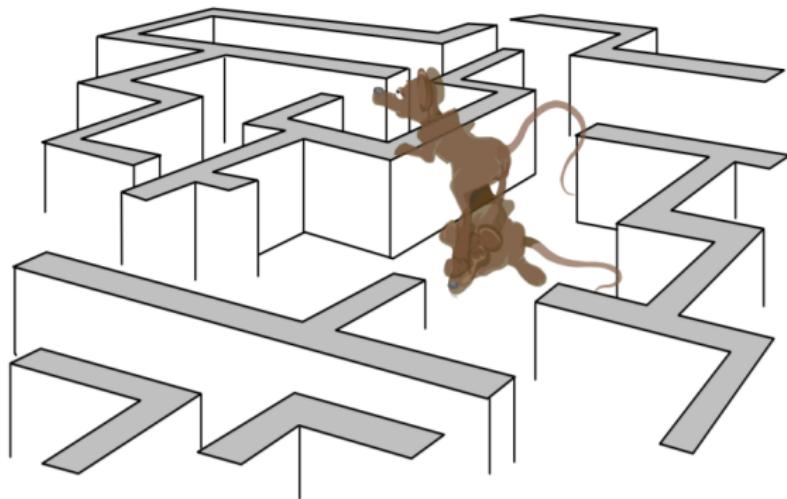
ÜBERSICHT ÜBER ROUTINGVERFAHREN

- **Distance Vector Algorithms:** Benutzt den distributed Bellman Ford Algorithmus, hat Probleme in großen Netzen. Kommunikation ist nur mit unmittelbaren Nachbarn notwendig.
- **Link State Algorithms:** Jeder Router bildet einen gewichteten Graph des gesamten Netzes. Die Routingtabelle ergibt sich durch den Shortest Path Algorithm. Das Verfahren hat Probleme in großen Netzen und erzeugt hohe Netz- und CPU-Last.
- **Path Vector Protocol:** Ähnlich dem Distance Vector Algorithms, aber es werden nur ausgewählte Hosts (Speaker Nodes) einbezogen.



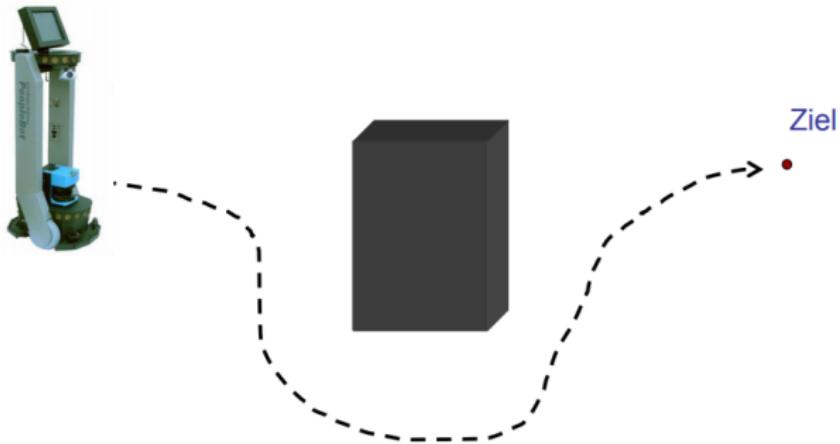
WEGPLANUNG FÜR ROBOTER

Es geht darum einen **kollisionsfreien** Weg in einer Umgebung zu finden.



WEGPLANUNG FÜR ROBOTER

Wie bewege ich mich sinnvoll in meiner Umgebung ?



3 Fragen müssen beantwortet werden:

1. Wo bin ich? (Selbstlokalisierung)
2. Wohin soll ich gehen?
3. Wie komme ich dahin? (**Wegplanung+ Bewegungsregelung**)



WEGPLANUNG FÜR ROBOTER

- Planung von kollisionsfreien Wegen für Roboter in ihrem Einsatzgebiet.
- Wie findet man die kürzesten Wege, auf denen der Roboter mit keinem Hindernis in Kontakt kommt?
- Zum Auffinden dieser Wege muß eine Beschreibung der Geometrie des Roboters und des Einsatzgebietes vorliegen.
- Ohne Einschränkungen der Freiheitsgrade des Roboters und der Komplexität des Einsatzgebietes ist dieses Problem praktisch nicht lösbar.



WEGPLANUNG FÜR ROBOTER

Ein Roboter soll

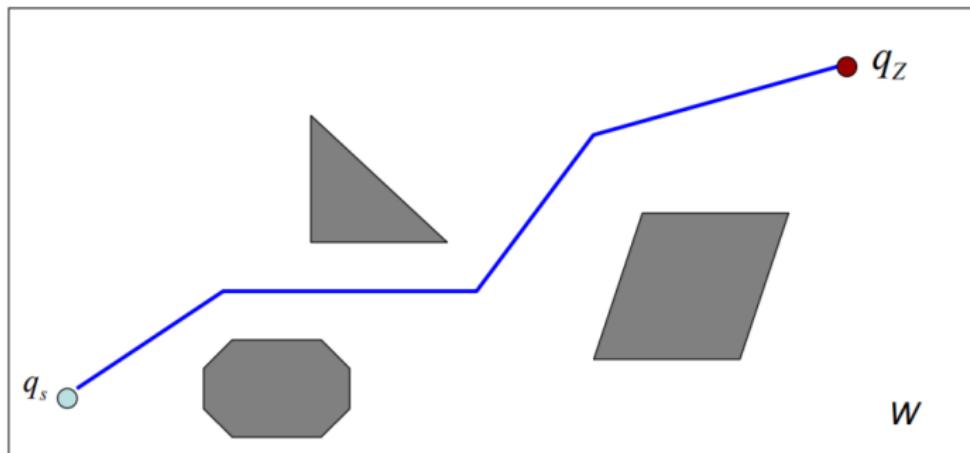
- kollisionsfrei,
- mit minimaler Fahrtzeit,
- mit minimaler Gesamtlnge, und
- mit minimaler Rechenzeit

sein Ziel erreichen.



PROBLEMSTELLUNG

Finde einen freien Pfad, also eine Folge von Konfigurationen $q_i \in C_{free}$, die die Startkonfiguration q_s mit der Zielkonfiguration q_Z verbinden.



WIE FINDET MAN EINE LÖSUNG?

- Modellieren!



WIE FINDET MAN EINE LÖSUNG?

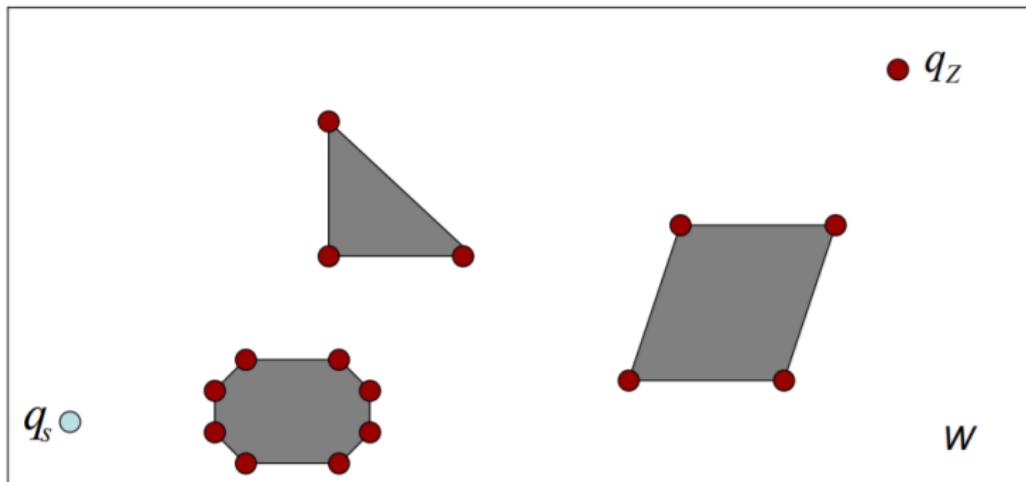
- Modellieren!
- In der Informatik reduziert sich nicht alles auf das Programmieren...
- Problem wird zur: Für eine Menge von Polygonen in der Ebene, einen Startpunkt s und einen Zielpunkt z ist der kürzeste Weg von s nach z gesucht, welcher die Polygone nicht schneidet



LÖSUNG

Knoten: Startpunkt, Zielpunkt, und die Eckpunkte der Hindernisse.

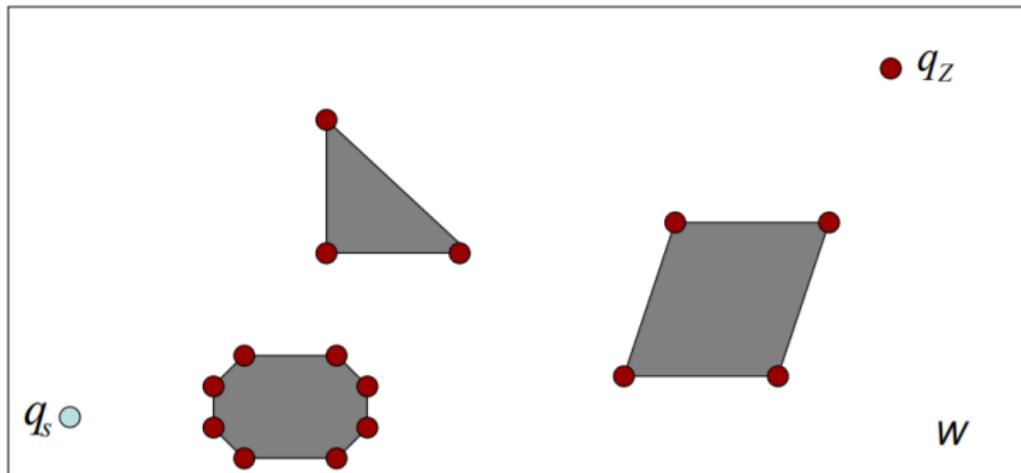
Der **Sichtgraph** Algorithmus **sucht** in der Ebene nach einem **Weg** anhand der **Knoten**.



LÖSUNG

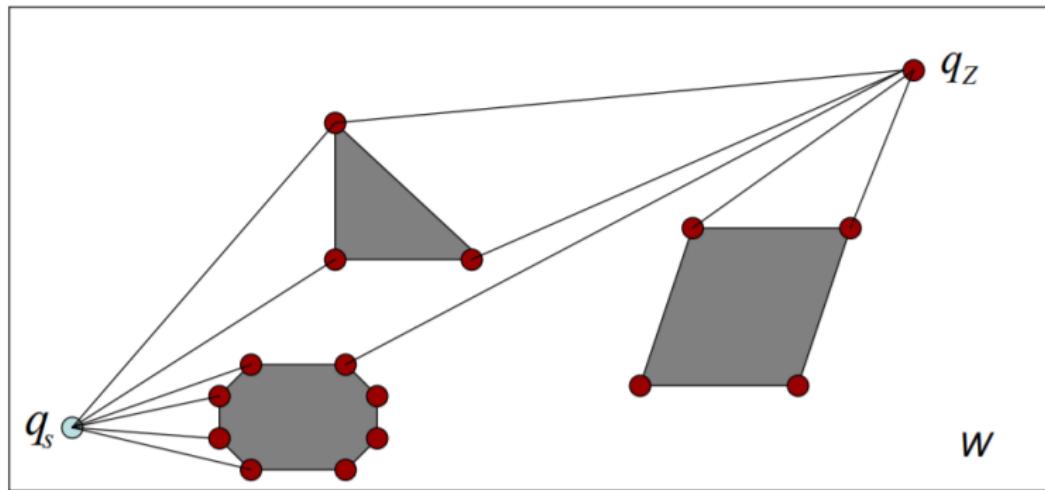
-die Knoten werden mit einer Kante verbunden, wenn sie sich gegenseitig sehen.

→ **Sehen bedeutet**, dass eine Strecke zwischen den beiden Knoten existiert, ohne ein Hindernis schneidet.



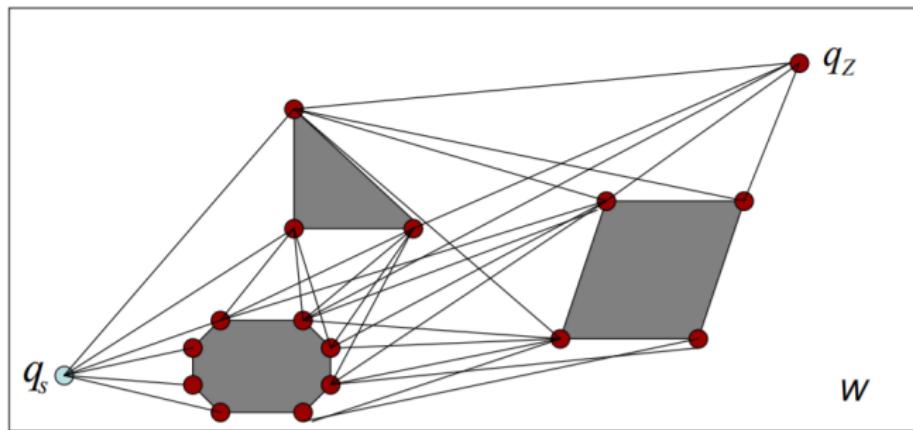
LÖSUNG

Zuerst werden Start q_s und Ziel q_z mit allen sichtbaren Ecken verbunden.



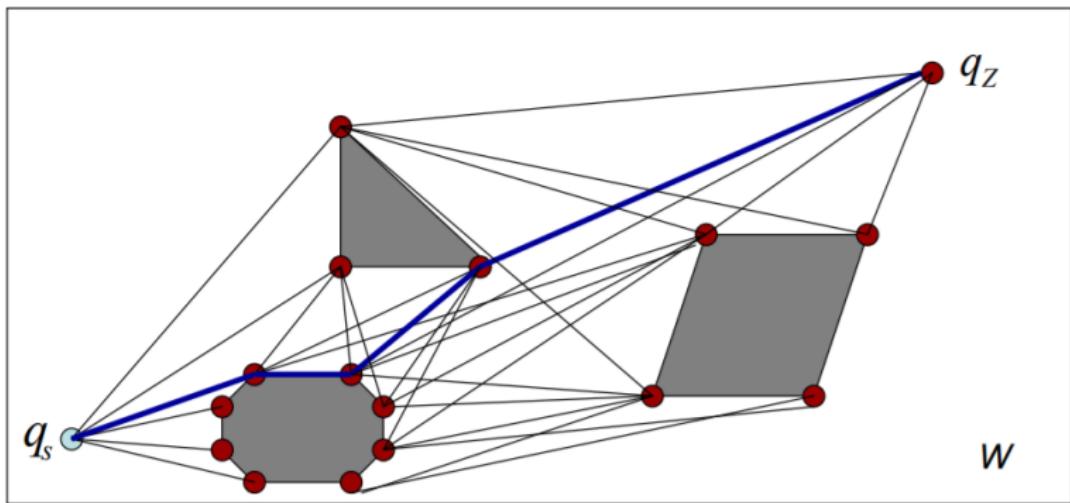
LÖSUNG

Die anderen Knoten werden mit Kanten verbunden, wenn sie sich gegenseitig sehen.

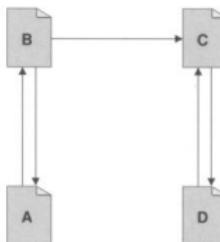


LÖSUNG

Der Algorithmus blickt von Knoten zu Knoten und baut so das Roadmap für den Weg auf (Dijkstra, A^* , usw.)



WWW ALS GERICHTETER GRAPH



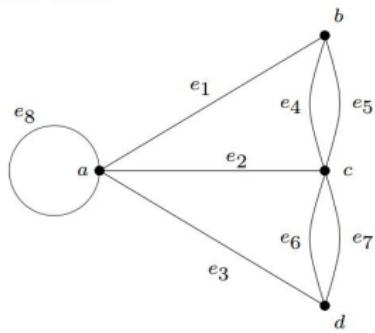
- Page Rank Algorithmus von Lawrence Page and Sergey Brin
- $LC(W)$ Anzahl der Verweise (Links), welche von Seite W ausgehen
- $P(W)$ Menge der Webseiten, welche einen Verweis auf Seite W enthalten
- Page Rank:

$$PR(W) = (1 - d) + d \cdot \sum_{D \in PL(W)} \frac{PR(D)}{LC(D)}, d \in [0, 1]$$



GRAPH - INFORMELL

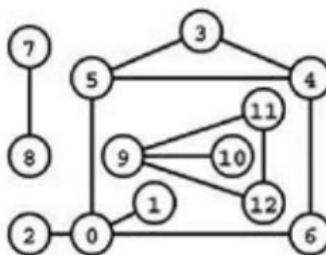
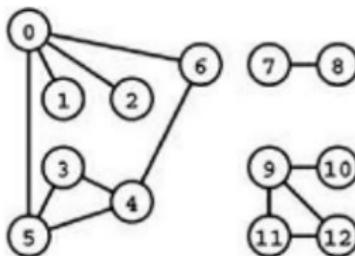
Beispiel 1.3.



- Ein Graph besteht aus Knoten und Kanten.
- a, b, c, d sind Knoten.
- Diese Knoten werden durch die Kanten e_1 bis e_8 miteinander verbunden.
- ☞ Ein Graph symbolisiert die max. zweistelligen Beziehungen zwischen Elementen einer Menge.



VERSCHIEDENE REPRÄSENTATIONEN



0-5	5-4	7-8
4-3	0-2	9-11
0-1	11-12	5-3
9-12	9-10	
6-4	0-6	



FORMALE DEFINITION EINES GRAPHS

Definition

Ein Graph $G = (V, E, \gamma)$ ist ein geordnetes Paar bestehend aus:

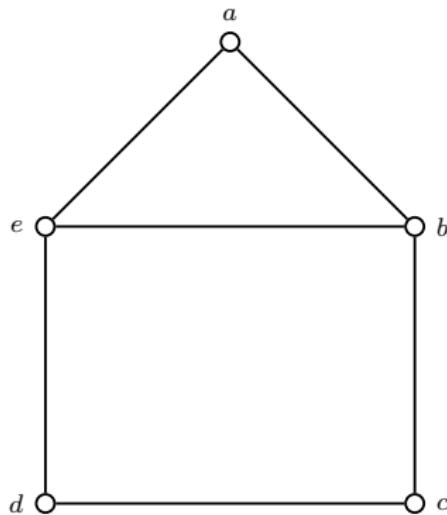
- einer nichtleeren Menge von Knoten (vertices), V ,
- einer Menge von Kanten (edges, arcs), E
- einer Funktion $\gamma: E \rightarrow \{X \mid X \subseteq V, 1 \leq |X| \leq 2\}$

Zwei Knoten $a, b \in V$ heißen adjazent (incident) falls $\exists e \in E$ mit $\gamma(e)$.

$a \in V$ und $e \in E$ heißen inzident (incident) gdw. $a \in \gamma(e)$.



HAUSGRAPH



- Knotenmenge $V = \{a, b, c, d, e\}$
- Kantenmenge $E = \{ab, ae, ba, bc, be, cb, cd, dc, de, ed, eb, ea\}$



HAUSGRAPH MIT SAGE

```
sage: G = Graph({ "a": ["b", "e"], "b": ["a", "c", "e"], "c": ["b", "d"],  
...   "d": ["c", "e"], "e": ["a", "b", "d"] })  
sage: G  
Graph on 5 vertices  
sage: G.vertices()  
['a', 'b', 'c', 'd', 'e']  
sage: G.edges(labels=False)  
[('a', 'b'), ('a', 'e'), ('b', 'e'), ('c', 'b'), ('c', 'd'), ('e', 'd')]
```

Frage: Weshalb werden von SAGE nur 6 Kanten angegeben, statt 12?



ADJAZENTE KNOTEN MIT SAGE

- Sei $v \in V$, dann ist $\text{adj}(v)$ die Menge der zu v adjazenten Knoten. Adjazente Knoten heißen auch **Nachbarn**.
- SAGE Funktion `G.neighbors`



FRAGESTELLUNGEN 1

- Welche Knoten oder Kanten haben die kleinste oder größte Anzahl von adjazenten Kanten?
- Was passiert wenn wir alle Kanten entfernen? Gibt es kantenfreie Graphen?
- Was passiert wenn wir alle Knoten entfernen? Gibt es knotenfreie Graphen?
- u.v.m.



FRAGESTELLUNGEN 2

Ein Graph ist bestimmt durch die Kanten, Knoten + Zuordnung Kanten - Endknoten

- Kann man die Kanten eines Graphen so durchlaufen, dass man jeden Knoten (oder jede Kante) genau einmal besucht?
- Wie viele Kanten muss man mindestens durchlaufen, um von einem Knoten zu einem anderen zu gelangen?
- Wie viele Knoten muss man mindestens besetzen, damit alle anderen Knoten die Graphen in der Nachbarschaft eines besetzten Knotens liegen?
- Gibt es zwischen je zwei Knoten einen Weg?

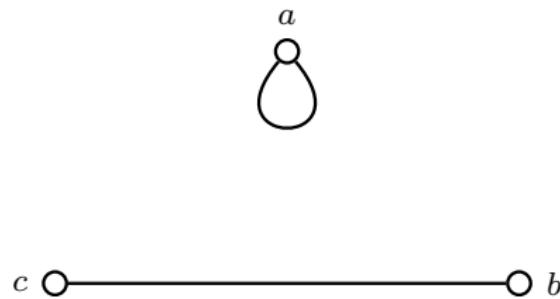


FRAGESTELLUNGEN 3

- Wie viele Kanten kann man aus dem Graphen auswählen, sodass keine zwei ausgewählten Kanten einen gemeinsamen Endknoten besitzen?
- Wie viele verschiedene Graphen mit einer gegebenen Knoten- und Kantenanzahl gibt es? Wie viele davon sind strukturell gleich?
- Wie viele Knoten und Kanten kann man aus einem Graphen entfernen, ohne dass dieser den Zusammenhang (oder andere wichtige Eigenschaften) verliert?
- Kann man einen gegebenen Graphen so in die Ebene zeichen, dass sich keine zwei Kanten überkreuzen?



BEISPIEL



Frage: Stellt diese Abbildung ein Graph dar?



SPEZIELLE GRAPHEN

Definition

Ein Graph $G = (V, E, \gamma)$ heißt **endlich (finite)** gdw. die Knotenmenge V und die Kantenmenge E endlich sind.

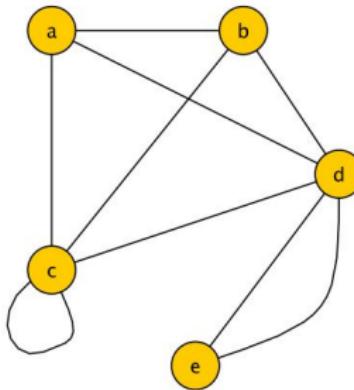
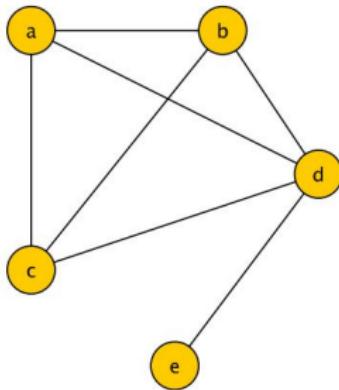
- $V(G), E(G)$
- **Schlinge:** $e \in E. \gamma(e) = \{v\}$
- **Parallele Kanten:** $e, f \in E. \gamma(e) = \gamma(f).$

Definition

Ein Graph, der weder Schlingen noch parallele Kanten besitzt, heißt **schlichter Graph**.



SCHLICHTE GRAPHEN - BEISPIEL



Der linke Graph ist schlicht, der rechte nicht.



SCHLICHTE GRAPHEN - FORMALE DEFINITION

Definition

Ein schlichter Graph $G = (V, E)$ wird beschrieben durch

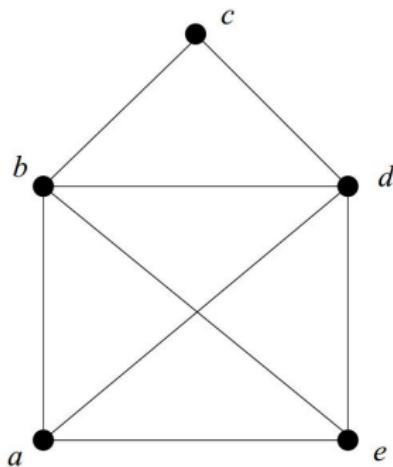
- eine Knotenmenge V und
- eine Kantemenge E , wobei E eine Menge zweielementiger Teilmengen von V ist, also

$$E \subseteq \{\{v, w\} \mid v, w \in V, v \neq w\}.$$

Im folgenden betrachten wir fast ausschließlich schlichte Graphen.



HAUS VOM NIKOLAUS ALS SCHLICHTER GRAPH



$$V = \{a, b, c, d, e\}$$

$$E = \{\{a, b\}, \{a, d\}, \{a, e\}, \{b, c\}, \\ \{b, d\}, \{b, e\}, \{c, d\}, \{d, e\}\}$$



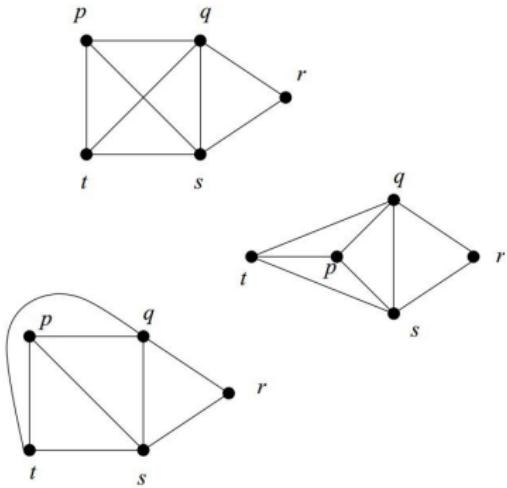
DIAGRAMME

- Graphen können durch Diagramme veranschaulicht werden.
- Der selbe Graph kann viele verschiedene Diagramme haben.

Beispiel $G = (V, E)$ mit

$$V = \{p, q, r, s, t\}$$

$$E = \{\{p, q\}, \{p, s\}, \{p, t\}, \{q, r\}, \\ \{q, s\}, \{q, t\}, \{r, s\}, \{s, t\}\}$$



KNOTENGRADE

Definition

Der *Grad (degree)* $\deg(v)$ eines Knoten $v \in V$ ist die Zahl der zu v inzidenten Kanten. Hierbei zählen Schlingen doppelt.

- Der *Maximalgrad* $\Delta(G)$ eines Graphen G ist definiert durch

$$\Delta(G) = \max\{\deg(v) \mid v \in V\}$$

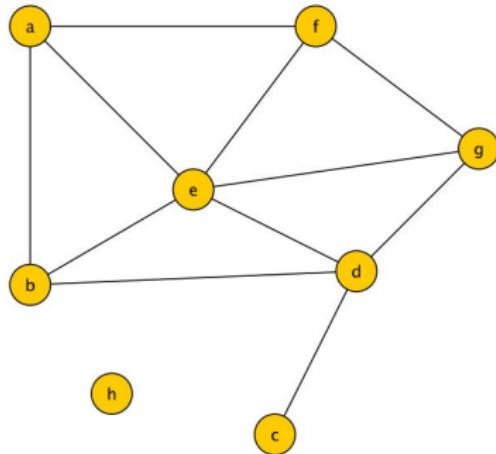
- Der *Minimalgrad* $\delta(G)$ eines Graphen G ist definiert durch

$$\delta(G) = \min\{\deg(v) \mid v \in V\}$$

- Ein Knoten $v \in V$ mit $\deg(v) = 0$ heißt *isoliert*.
- Ein Knoten $v \in V$ mit $\deg(v) = 1$ heißt *Blatt*.



BEISPIEL



- $\delta(G) = 0$ (für Knoten h)
- $\Delta(G) = 5$ (für Knoten e)
- Knoten c ist ein Blatt.
- Knoten h ist ein isolierter Knoten.



HANDSCHLAGLEMMA

- Jede Kante liefert genau zweimal einen Beitrag zu der Summe der Grade über alle Knoten.
- Dies gilt auch für Schlingen.

Lemma (Handschriftaglemma)

Für jeden Graphen $G = (V, E, \gamma)$ gilt

$$\sum_{v \in V} \deg(v) = 2 |E| .$$



HANDSCHLAGLEMMA 2

Korollar *Jeder Graph hat eine gerade Anzahl an Knoten mit ungeradem Grad.*

Beweis:

$$\begin{array}{lcl} V_g &:=& \{v \in V \mid \deg(v) \text{ ist gerade}\} \\ V_u &:=& \{v \in V \mid \deg(v) \text{ ist ungerade}\} \\ 2|E| &=& \sum_{v \in V} \deg(v) \\ &=& \sum_{v \in V_g} \deg(v) + \sum_{v \in V_u} \deg(v) \end{array} \quad \left| \begin{array}{l} \text{Definition} \\ \text{Definition} \\ \text{Handsaglemma} \\ \text{weil } V = V_g + V_u \end{array} \right.$$

$$\begin{array}{lcl} \Rightarrow \sum_{v \in V_u} \deg(v) \text{ ist gerade} && \left| \begin{array}{l} \text{weil } 2|E| \text{ und } \sum_{v \in V_g} \deg(v) \text{ gerade} \end{array} \right. \\ \Rightarrow |V_u| \text{ ist gerade} && \left| \begin{array}{l} \text{weil alle Summanden ungerade} \end{array} \right. \end{array}$$



Satz

Jeder Graph $G = (V, E)$ mit mindestens zwei Knoten enthält zwei Knoten, die den gleichen Grad haben.

Der Beweis dieses Satzes erfolgt mit Hilfe eines wichtigen kombinatorischen Prinzips, dem **Schubfachprinzip**.



SCHUBFACHPRINZIP

Theorem (Schubfachprinzip)

Es seien n Elemente auf m (paarweise disjunkte) Mengen verteilt und es gelte $n > m$. Dann gibt es mindestens eine Menge, die **mindestens zwei Elemente** enthält.

Beweis.

Wenn jede der m Mengen höchstens ein Element enthalten würde, dann gäbe es insgesamt höchstens m Elemente.

Widerspruch zu $n > m$.



Andere Bezeichnungen für das Schubfachprinzip:
Taubenschlagprinzip, pigeonhole principle.



BEISPIEL

Herr Müller hat in seiner Sockenkiste weiße, schwarze und grüne Socken.

Wenn er vier Socken aus der Kiste nimmt, hat er mindestens zwei Socken mit der gleichen Farbe.

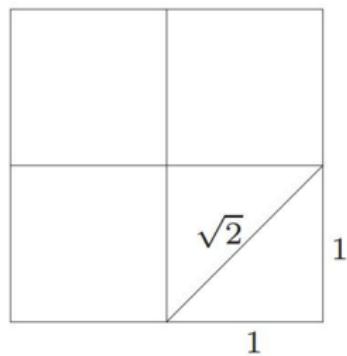
$n = 4$ Elemente verteilt auf $m = 3$ Mengen.



SCHUBFACHPRINZIP

Unter je fünf Punkten, die in einem Quadrat der Seitenlänge 2 liegen, gibt es stets zwei, die einen Abstand $\leq \sqrt{2}$ haben.

- Wir unterteilen das Quadrat durch halbieren der Seitenlänge in vier Unterquadrate mit Seitenlänge 1.
- $n = 5$ Punkte verteilen sich auf $m = 4$ Unterquadrate.
- Dann muss mindestens ein Unterquadrat zwei Punkte enthalten.



BEWEIS

$$\begin{array}{ll} n & := |V| \\ V_i & := \{v \in V \mid \deg(v) = i\} \text{ für } i = 0, \dots, n-1 \end{array}$$

Definition
Definition

Damit haben wir aber genauso viele Knoten wie Mengen, das Schubfachprinzip ist noch nicht anwendbar. Deshalb Fallunterscheidung:

- | | |
|--|--------------------------------|
| G hat keinen isolierten Knoten | Annahme für den 1. Fall |
| $\Rightarrow V_0 = \emptyset$ | nach Ann. und Def. von V_0 |
| $\Rightarrow n$ Knoten verteilen sich auf die $m = n - 1$ Mengen V_1, \dots, V_{n-1} | weil $V_0 = \emptyset$ |
| $\Rightarrow \exists i : V_i \geq 2$ | Schubfachprinzip |



BEWEIS

- G hat einen isolierten Knoten
- \Rightarrow Es existiert kein Knoten, der zu allen anderen Knoten adjazent ist
- $\Rightarrow V_{n-1} = \emptyset$
- \Rightarrow n Knoten verteilen sich auf die $m = n - 1$ Mengen V_0, \dots, V_{n-2}
- $\Rightarrow \exists i : |V_i| \geq 2$
- 2. Fall**, komplementär zum 1.
kein Knoten kann zu einem isolierten Knoten adjazent sein
ein Knoten $v \in V_{n-1}$ wäre adjazent zu allen anderen Knoten
weil $V_{n-1} = \emptyset$
- Schubfachprinzip



GERICHTETE GRAPHEN

Bislang spielte die **Richtung** einer Kante überhaupt keine Rolle.
Im Hausgraph war die Kante ae gleich der Kante ea .

Definition

Eine Kante ab heißt **gerichtet**, falls a der **Anfangsknoten** und b der **Endknoten** ist. In diesem Fall ist die Menge der Kanten $E \subseteq V \times V$. Eine gerichtete Kante ist vom Anfangsknoten zum Endknoten gerichtet. Ein **gerichteter Graph (Digraph)** ist ein Graph dessen Kanten alle gerichtet sind.

Gerichtete Kanten sind als Pfeile dargestellt.



GRAD IN DIGRAPHEN

Definition

Sei $G = (V, E)$ ein gerichteter Graph.

- Der **Eingangsgrad** eines Knoten $v \in V$, $id(v)$, ist die Anzahl der Vorgänger von v , also die Anzahl der Kanten deren Endknoten v ist.
- Der **Ausgangsgrad** eines Knoten $v \in V$, $od(v)$, ist die Anzahl der Nachfolger von v , also die Anzahl der Kanten deren Anfangsknoten v ist.
- Der **Grad** von v ist die Summe der Eingangs- und Ausgangsgrade von v : $\deg(v) = id(v) + od(v)$.



MULTIGRAPHEN

→ nicht schlichte Graphen

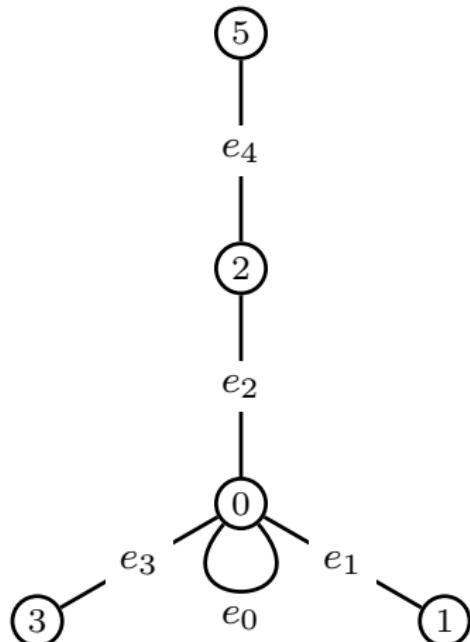
→ SAGE kann auch mit Multigraphen arbeiten.

```
sage: G = Graph({0:{0:'e0'},1:'e1',2:'e2',3:'e3'}, 2:{5:'e4'})  
sage: G.show(vertex_labels=True, edge_labels=True, graph_border=True)  
sage: H = DiGraph({0:{0:"e0"}}, Loops=True)  
sage: H.add_edges([(0,1,'e1'), (0,2,'e2'), (0,2,'e3'), (1,2,'e4'), (1,0,'e5')])  
sage: H.show(vertex_labels=True, edge_labels=True, graph_border=True)
```



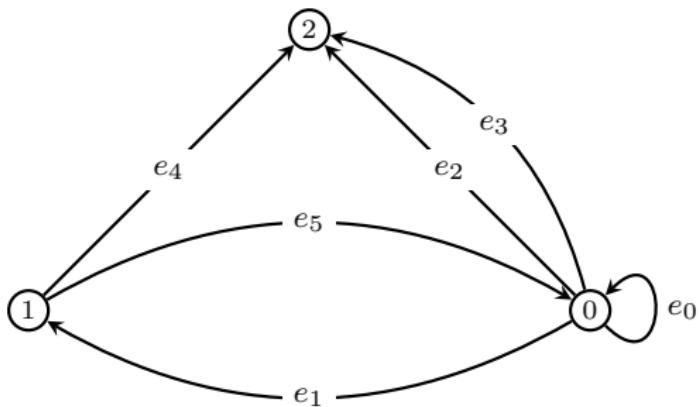
BEISPIEL

MULTIGRAPH 1

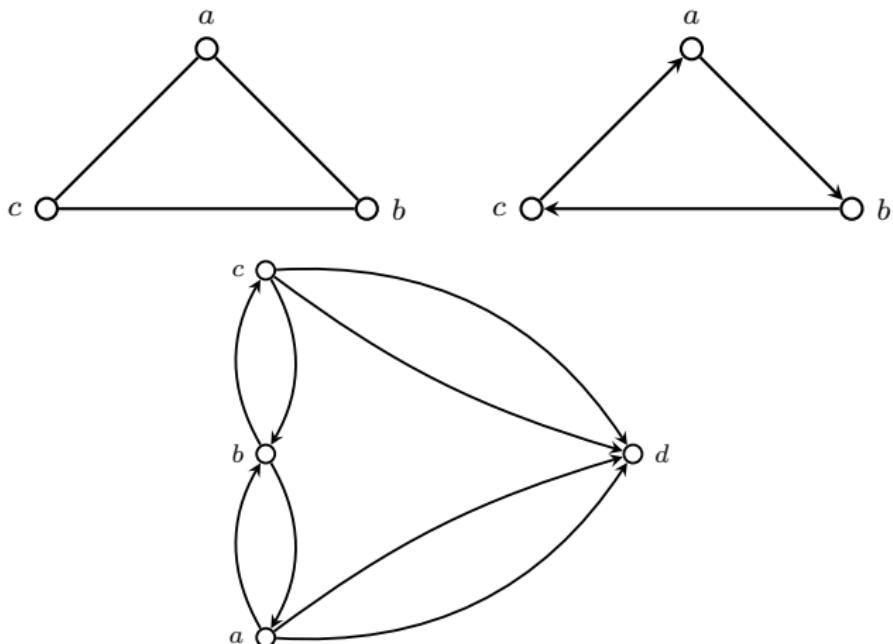


BEISPIEL

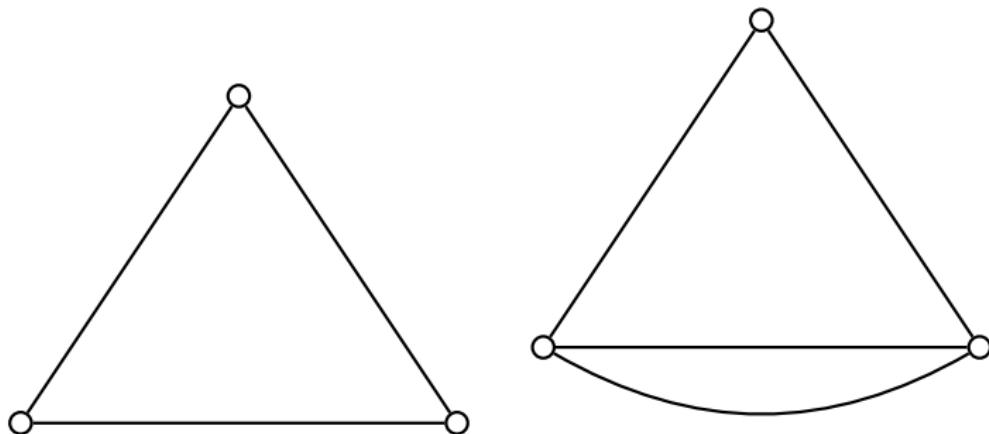
MULTIGRAPH 2



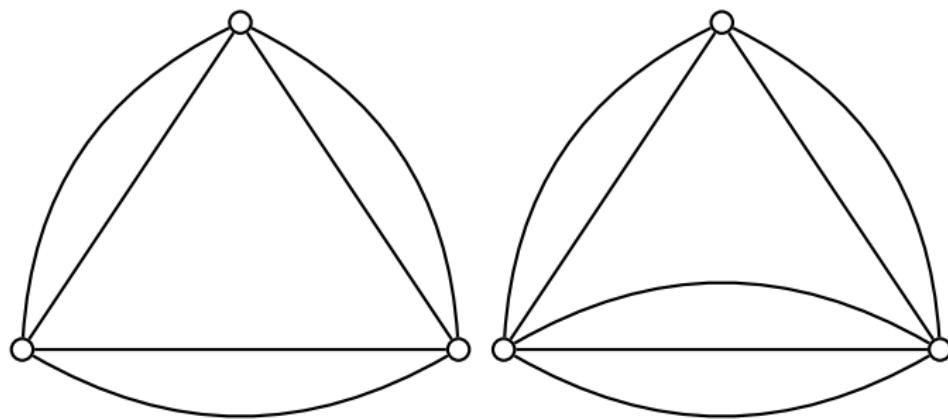
SCHLICHTE GRAPHEN, GERICHTETE GRAPHEN, MULTIGRAPHEN



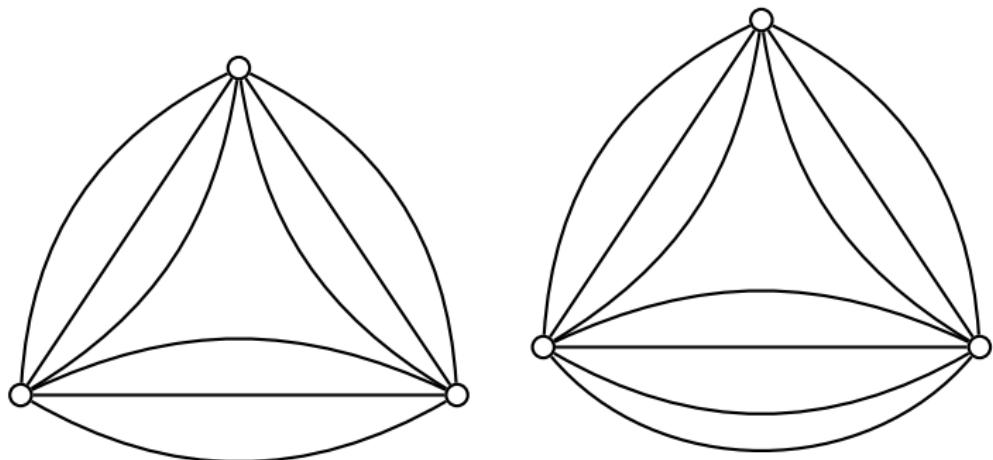
DIE SHANNON MULTIGRAPHEN FAMILIE $Sh(n)$ FÜR $n = 2, 3$



DIE SHANNON MULTIGRAPHEN FAMILIE $Sh(n)$ FÜR $n = 4, 5$



DIE SHANNON MULTIGRAPHEN FAMILIE $Sh(n)$ FÜR $n = 6, 7$



VOLLSTÄNDIGE GRAPHEN

Definition

Sei $G = (V, E)$ ein Graph. Gilt $\{v, w\} \in E$ für alle $v, w \in V, v \neq w$, dann heißt G vollständig. Wir bezeichnen mit K_n den vollständigen Graphen mit n Knoten.

- In einem vollständigen Graphen sind je zwei Knoten adjazent!
- Der K_n hat



VOLLSTÄNDIGE GRAPHEN

Definition

Sei $G = (V, E)$ ein Graph. Gilt $\{v, w\} \in E$ für alle $v, w \in V, v \neq w$, dann heißt G vollständig. Wir bezeichnen mit K_n den vollständigen Graphen mit n Knoten.

- In einem vollständigen Graphen sind je zwei Knoten adjazent!
- Der K_n hat

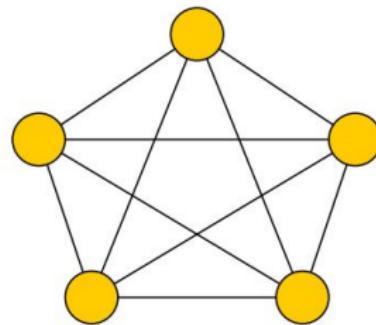
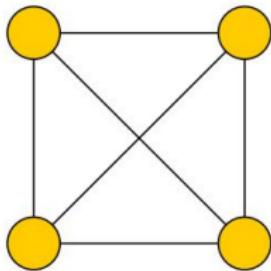
$$C_n^2 = \frac{n(n - 1)}{2}$$

Kanten.



BEISPIEL

Die vollständigen Graphen K_4 und K_5 :



ISOMORPHIE

Definition

Zwei Graphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ heißen *isomorph* gwd. es eine bijektive Abbildung $\phi: V_1 \rightarrow V_2$ gibt, so dass folgendes gilt:

$$\forall v, w \in V_1: \{v, w\} \in E_1 \Leftrightarrow \{\phi(v), \phi(w)\} \in E_2.$$

Wir nennen ϕ dann einen *Isomorphismus* von G_1 auf G_2 .



ISOMORPHIE (2)

- Zwei Graphen sind genau dann isomorph, wenn der eine Graph aus dem anderen Graphen durch Umbenennung der Knoten hervorgeht.
- Isomorphe Graphen haben die gleichen graphentheoretischen Eigenschaften

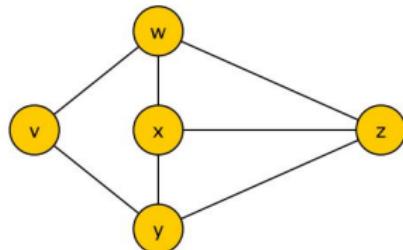
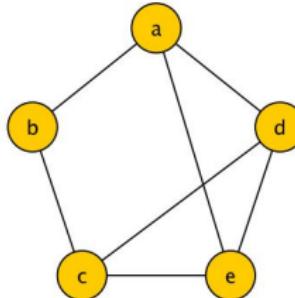


ISOMORPHIE (3)

Die Abbildung

$$\varphi = \{a \rightarrow w, b \rightarrow v, c \rightarrow y, d \rightarrow x, e \rightarrow z\}$$

ist ein Isomorphismus für die folgenden Graphen.



ISOMORPHIE (4)

Definition

Wenn $G_1 = G_2$ in der Definition der Isomorphie, dann ist ϕ ein Automorphismus.

Ein Automorphismus für einen Graphen G ist eine strukturerhaltende Abbildung von G auf sich selbst.

Beispiel

Für den linken Graphen ist

$$\{a \rightarrow c, b \rightarrow b, c \rightarrow a, d \rightarrow e, e \rightarrow d\}$$

ein Automorphismus.



BEISPIEL ISOMORPHER GRAPHEN

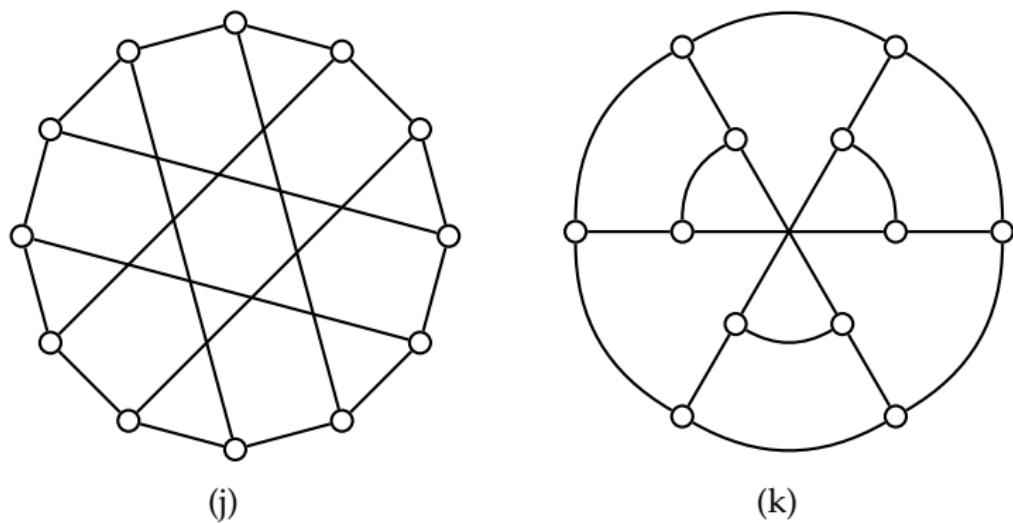
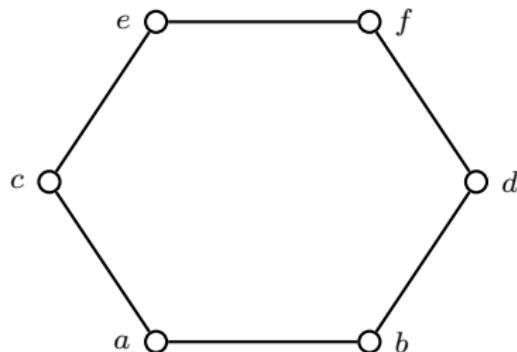


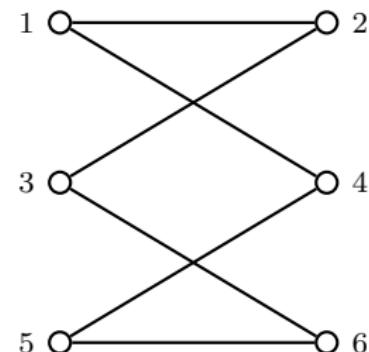
Abbildung 1: Zwei Darstellungen des Franklin Graphs.



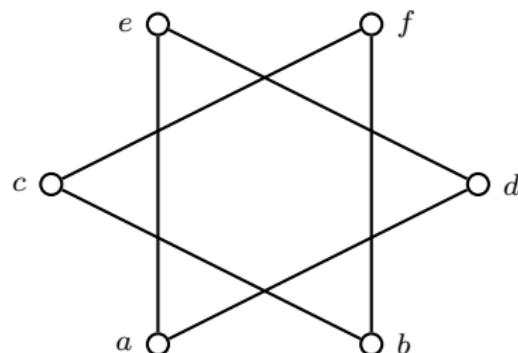
ISOMORPHE UND NICHTISOMORPHE GRAPHEN



(a) C_6



(b) G_1



ISOMORPHIE MIT SAGE

→ Methode `G.is_isomorphic()`.

```
sage: C6 = Graph({ "a": ["b", "c"], "b": ["a", "d"], "c": ["a", "e"], \
...   "d": ["b", "f"], "e": ["c", "f"], "f": ["d", "e"] })
sage: G1 = Graph({1:[2,4], 2:[1,3], 3:[2,6], 4:[1,5], \
...   5:[4,6], 6:[3,5] })
sage: G2 = Graph({ "a": ["d", "e"], "b": ["c", "f"], "c": ["b", "f"], \
...   "d": ["a", "e"], "e": ["a", "d"], "f": ["b", "c"] })
sage: C6.is_isomorphic(G1)
True
sage: C6.is_isomorphic(G2)
False
sage: G1.is_isomorphic(G2)
```



INVARIANZ

Ein wichtiger Begriff ist **Invarianz**. Ein **Invariant** ist ein Objekt, welches mit einem Graphen G assoziiert wird. Falls G und H isomorphe Graphen sind, dann gilt

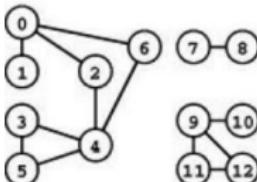
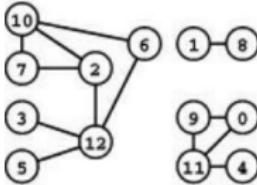
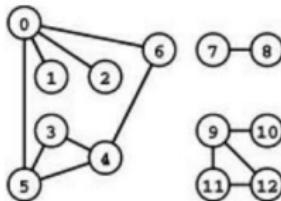
$$f(G) = f(H).$$

Z.B. ist die Anzahl der Knoten eines Graphen ein Invariant,
 $f(G) = |V(G)|$.



ISOMORPHIE (5)

Zwei ja, eins nein... Welcher und wie? :P



GRADFOLGEN

- Sei G ein Graph mit n Knoten. Eine **Gradfolge** von G ist eine absteigende Folge der Knotengrade von G .
- Die Gradfolge des Hausgraphs ist $3, 3, 2, 2, 2$.
- Für den zyklischen Graphen C_n , $n \geq 3$ ist die Gradfolge

$$\underbrace{2, 2, 2, \dots, 2}_{n \text{ mal}}.$$

- Der vollständige Graph K_n , $n \geq 1$ hat die Gradfolge

$$\underbrace{n - 1, n - 1, n - 1, \dots, n - 1}_{n \text{ mal}}.$$



GRADFOLGEN

Definition

Sei S ein absteigende Folge natürlicher Zahlen. S wird als *graphisch* bezeichnet, falls es die Gradfolge irgendeines Graphen ist. Falls G ein Graph ist mit Gradfolge S ist, sagen wir, dass S von G *realisiert* wird.

Sei $S = (d_1, d_2, \dots, d_n)$ eine graphische Folge, d.h. $d_i \geq d_j$ für alle $i \leq j, 1 \leq i, j \leq n$. Aus dem Handschlaglemma folgt, dass $\exists k \geq 0$ mit

$$\sum_{d_i \in S} d_i = 2k.$$

Die Summe einer graphischen Folge ist gerade und verschieden von Null.



THEOREM VON ERDÖS UND GALLAI, 1961

Theorem

Sei $d = (d_1, d_2, \dots, d_n)$ eine Folge positiver Zahlen mit $d_i \geq d_{i+1}$.
Die Folge d wird durch einen schlichten Graphen **realisiert** gdw.
 $\sum_i d_i$ gerade ist und

$$\sum_{i=1}^k d_i \leq k(k+1) + \sum_{j=k+1}^n \min\{k, d_j\}$$

für alle $1 \leq k \leq n-1$.

→ Dieses Ergebnis ist ein reines Existenzergebnis und erlaubt keinen Algorithmus aufzubauen, um einen schlichten Graphen mit Gradfolge d zu bestimmen.



THEOREM VON HAVEN UND HAKIMI

Theorem

Sei $S_1 = (d_1, d_2, \dots, d_n)$ eine absteigende Folge natürlicher Zahlen mit $n \geq 2$ und $d_1 \geq 1$. Dann ist S_1 graphisch genau dann, wenn die Folge

$$S_2 = (d_2 - 1, d_3 - 1, \dots, d_{d_1+1} - 1, d_{d_1+2}, \dots, d_n)$$

graphisch ist.



Beweis (1)

Angenommen S_2 ist graphisch. Sei $G_2 = (V_2, E_2)$ der Graph der Ordnung $n - 1$ mit Knotenmenge $V_2 = \{v_2, v_3, \dots, v_n\}$ und

$$\deg(v_i) = \begin{cases} d_i - 1, & \text{falls } 2 \leq i \leq d_1 + 1, \\ d_i, & \text{falls } d_1 + 2 \leq i \leq n. \end{cases}$$



BEWEIS (2)

Wir erzeugen einen neuen Graphen G_1 mit graphischer Folge S_1 wie folgt.

- Füge einen Knoten v_1 in V_2 ein und erweitere E_2 mit den Kanten v_1v_i für $2 \leq i \leq d_1 + 1$.
- $\deg(v_1) = d_1$ und $\deg(v_1) = d_i$ für $2 \leq i \leq n$.
- G_1 hat die graphische Folge S_1 .



BEWEIS (3)

Angenommen S_1 ist eine graphische Folge und G_1 ein realisierender Graph, so dass

- ① Der Graph G_1 hat Knotenmenge $V(G_1) = \{v_1, v_2, \dots, v_n\}$ und $\deg(v_i) = d_i$ für $i = 1, \dots, n$.
- ② Die Summe der Grade aller benachbarten Knoten von v_1 ist maximal.

Wir nehmen jetzt an, dass v_1 ist nicht benachbart mit Knoten vom Grad

$$d_2, d_3, \dots, d_{d_1+1}.$$

Dann existieren Knoten v_i und v_j mit $d_j > d_i$, so dass $v_1v_i \in E(G_1)$ aber $v_1v_j \notin E(G_1)$.



BEWEIS (3)

Angenommen S_1 ist eine graphische Folge und G_1 ein realisierender Graph, so dass

- ① Der Graph G_1 hat Knotenmenge $V(G_1) = \{v_1, v_2, \dots, v_n\}$ und $\deg(v_i) = d_i$ für $i = 1, \dots, n$.
- ② Die Summe der Grade aller benachbarten Knoten von v_1 ist maximal.

Wir nehmen jetzt an, dass v_1 ist nicht benachbart mit Knoten vom Grad

$$d_2, d_3, \dots, d_{d_1+1}.$$

Dann existieren Knoten v_i und v_j mit $d_j > d_i$, so dass $v_1v_i \in E(G_1)$ aber $v_1v_j \notin E(G_1)$. Weil $d_j > d_i$, existiert ein Knoten v_k so dass $v_jv_k \in E(G_1)$ aber $v_iv_k \notin E(G_1)$.



BEWEIS (4)

Wir ersetzen nun die Kanten v_1v_i und v_jv_k mit v_1v_j und v_iv_k und erhalten einen neuen Graphen H mit graphischer Folge S_1 .



BEWEIS (4)

Wir ersetzen nun die Kanten v_1v_i und v_jv_k mit v_1v_j und v_iv_k und erhalten einen neuen Graphen H mit graphischer Folge S_1 . Allerdings aber, die Summe der Grade zu v_1 benachbarten Knoten ist größer als die entsprechende Summe in G_1 , Widerspruch zu Eigenschaft (2) von G_1 .



BEWEIS (4)

Wir ersetzen nun die Kanten v_1v_i und v_jv_k mit v_1v_j und v_iv_k und erhalten einen neuen Graphen H mit graphischer Folge S_1 .

Allerdings aber, die Summe der Grade zu v_1 benachbarten Knoten ist größer als die entsprechende Summe in G_1 , Widerspruch zu Eigenschaft (2) von G_1 .

Es folgt, dass v_1 ist benachbart mit d_1 Knoten mit maximalem Grad. Dann ist S_2 graphisch, da G_1 ohne v_1 die Gradfolge S_2 hat.



VOM SATZ ZUM ALGORITHMUS



VOM SATZ ZUM ALGORITHMUS

- **Test 1:** G ist ein einfacher Graph
 $\Rightarrow \forall v \in V(G), \deg(v) \leq |V(G)|$
- **Test 2:** Aus dem Handschlaglemma folgt: die Summe aller Folgenglieder ist gerade
- Nach der Erfüllbarkeit dieser beiden Teste, wird die Methode aus dem Theorem benutzt, um die Folge iterativ zu verkleinern.



HAVEL-HAKIMI TEST FÜR DIE, VON SCHLICHTEN GRAPHEN REALISIERTEN FOLGEN

Input: A nonincreasing sequence $S = (d_1, d_2, \dots, d_n)$ of nonnegative integers,
where $n \geq 2$.

Output: True if S is realizable by a simple graph; False otherwise.

```
1 if  $\sum_i d_i$  is odd then
2   return False
3 while True do
4   if  $\min(S) < 0$  then
5     return False
6   if  $\max(S) = 0$  then
7     return True
8   if  $\max(S) > \text{length}(S) - 1$  then
9     return False
10   $S \leftarrow (d_2 - 1, d_3 - 1, \dots, d_{d_1+1} - 1, d_{d_1+2}, \dots, d_{\text{length}(S)})$ 
11  sort  $S$  in nonincreasing order
```

