

6 Abstraktion

Einführung in Graphentheorie

Marianne Maertens

Technische Universität Berlin

Wintersemester 2025/2026

Themen

- Wiederholung: RSA-Kryptosystem
- Leseauftrag: Suchalgorithmen
- Abstraktion
- Graphentheorie
 - Begriffe
 - Wege finden - Königsberger Brückenproblem
 - Eulerweg & Eulerkreis
 - Der Algorithmus von Fleury
 - Der Algorithmus von Hierholzer



foto of "The head of a woman" by Pablo Picasso,
exhibited in the Metropolitan Museum of Modern
Art, NYC

Themen

- Wiederholung: RSA-Kryptosystem
- Leseauftrag: Suchalgorithmen
- Abstraktion
- Graphentheorie
 - Begriffe
 - Wege finden - Königsberger Brückenproblem
 - Eulerweg & Eulerkreis
 - Der Algorithmus von Fleury
 - Der Algorithmus von Hierholzer



Suchprobleme als Beispiel intelligenten Verhaltens

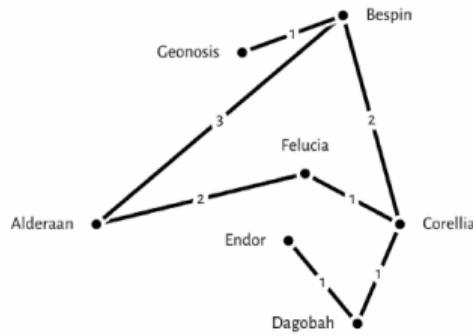
Leseauftrag

1. Lesen Sie bis zum Shuttle-Plan und zeichen Sie dann die Netzkarte (Erfahrung!!)
2. Wie funktioniert der Plan-basierte Algorithmus?
3. Wie funktioniert der heuristische Algorithmus?
4. Was ist eine Heuristik?



Netzkarte + Algorithmus

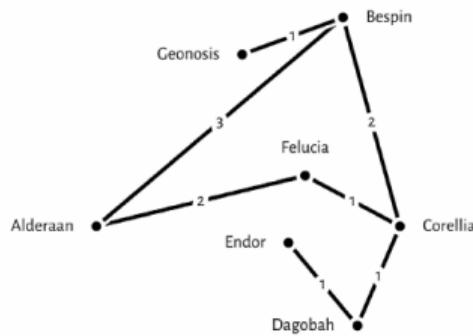
Wähle Zeile mit bislang kürzester Reisezeit und notiere alle von dort möglichen Weiterreisen



Alderaan → Endor

Netzkarke + Algorithmus

Wähle Zeile mit bislang kürzester Reisezeit und notiere alle von dort möglichen Weiterreisen

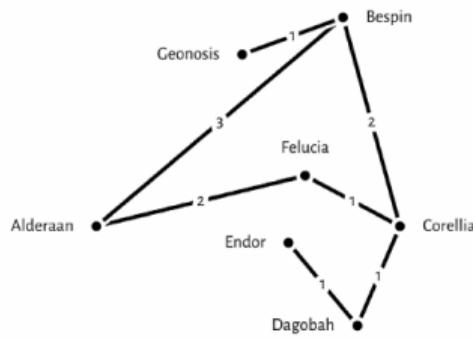


- 3 Alderaan - Bespin
- 2 Alderaan - Felucia

Alderaan → Endor

Netzkarte + Algorithmus

Wähle Zeile mit bislang kürzester Reisezeit und notiere alle von dort möglichen Weiterreisen

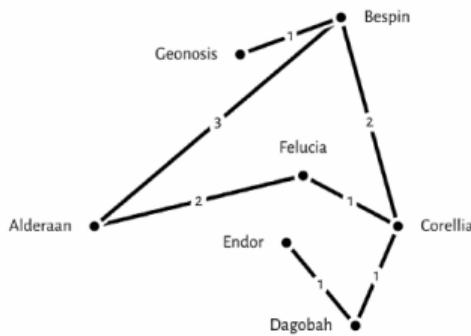


- 3 Alderaan - Bespin
- 3 Alderaan - Felucia - Corellia

Alderaan → Endor

Netzkarthe + Algorithmus

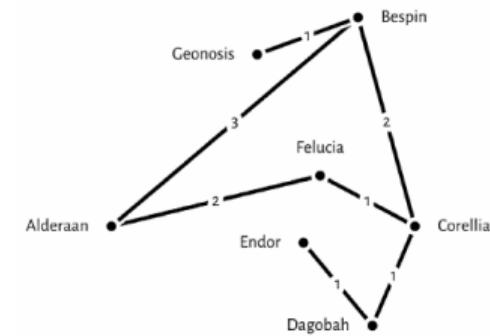
Wähle Zeile mit bislang kürzester Reisezeit und notiere alle von dort möglichen Weiterreisen



- 3 Alderaan - Bespin
- 3 Alderaan - Felucia - Corellia
- 5 Alderaan - Bespin - Corellia
- 4 Alderaan - Bespin - Geonosis (Sackgasse)
- 3 Alderaan - Felucia - Corellia

Alderaan → Endor

Netzkarte + Algorithmus



Wähle Zeile mit bislang kürzester Reisezeit und notiere alle von dort möglichen Weiterreisen

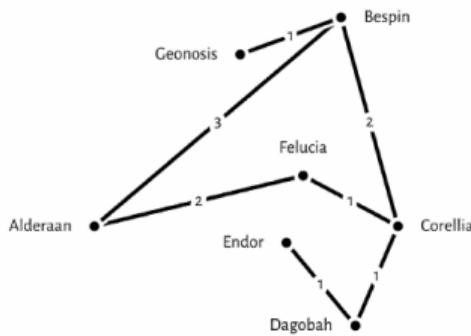
- 3 Alderaan - Bespin
- 3 Alderaan - Felucia - Corellia
- 5 Alderaan - Bespin - Corellia
- 4 Alderaan - Bespin - Geonosis (Sackgasse)
- 3 Alderaan - Felucia - Corellia

Alderaan → Endor

- 5 Alderaan - Bespin - Corellia
- 5 Alderaan - Felucia - Corellia - Bespin
- 4 Alderaan - Felucia - Corellia - Dagobah

Netzkarte + Algorithmus

Wähle Zeile mit bislang kürzester Reisezeit und notiere alle von dort möglichen Weiterreisen



- 3 Alderaan - Bespin
- 3 Alderaan - Felucia - Corellia
- 5 Alderaan - Bespin - Corellia
- 4 Alderaan - Bespin - Geonosis (Sackgasse)
- 3 Alderaan - Felucia - Corellia

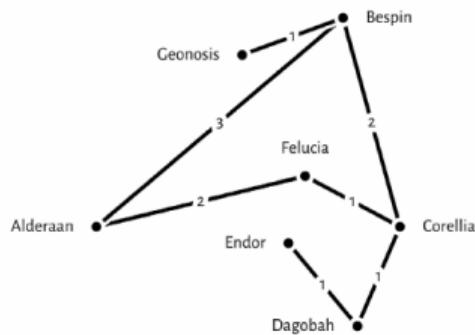
Alderaan → Endor

- 5 Alderaan - Bespin - Corellia
- 5 Alderaan - Felucia - Corellia - Bespin
- 5 Alderaan - Felucia - Corellia - Dagobah - Endor

Heuristischer Algorithmus

Wie machen Menschen das? → Distanzschätzung im Euklidischen Raum

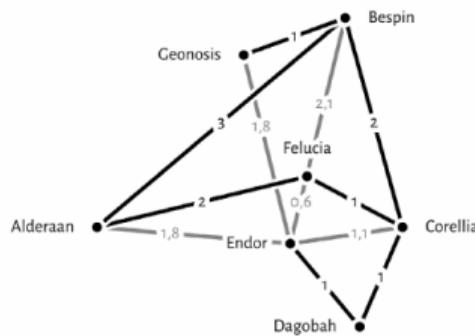
→ Algorithmus bekommt Koordinaten der Planeten - Zusatzinformation



Alderaan → Endor

<https://thedecisionlab.com/biases/heuristics>

Heuristischer Algorithmus



Wie machen Menschen das? → Distanzschätzung im Euklidischen Raum

→ Algorithmus bekommt Koordinaten der Planeten - Zusatzinformation

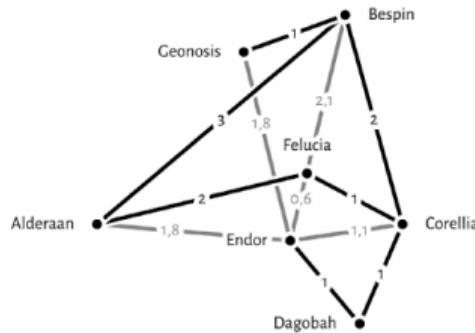
$$3+2,1 \text{ Alderaan - Bespin}$$

$$2+0,6 \text{ Alderaan - Felucia}$$

Alderaan → Endor

<https://thedecisionlab.com/biases/heuristics>

Heuristischer Algorithmus



Wie machen Menschen das? → Distanzschätzung im Euklidischen Raum

→ Algorithmus bekommt Koordinaten der Planeten - Zusatzinformation

$3+2,1$ Alderaan - Bespin

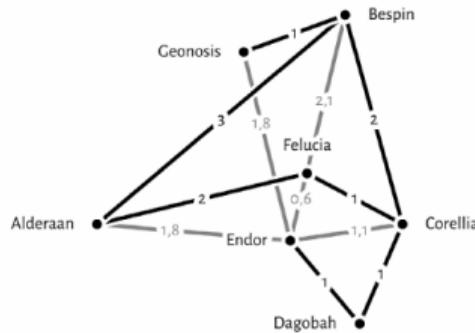
$2+0,6$ Alderaan - Felucia

Zusatzinfo der Restroute nutzen

Alderaan → Endor

<https://thedecisionlab.com/biases/heuristics>

Heuristischer Algorithmus



Wie machen Menschen das? → Distanzschätzung im Euklidischen Raum

→ Algorithmus bekommt Koordinaten der Planeten - Zusatzinformation

3+2,1 Alderaan - Bespin

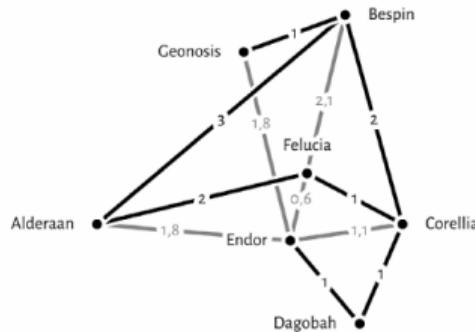
3+1,1 Alderaan - Felucia - Corellia

Zusatzinfo der Restroute nutzen

Alderaan → Endor

<https://thedecisionlab.com/biases/heuristics>

Heuristischer Algorithmus



Wie machen Menschen das? → Distanzschätzung im Euklidischen Raum

→ Algorithmus bekommt Koordinaten der Planeten - Zusatzinformation

3+2,1 Alderaan - Bespin

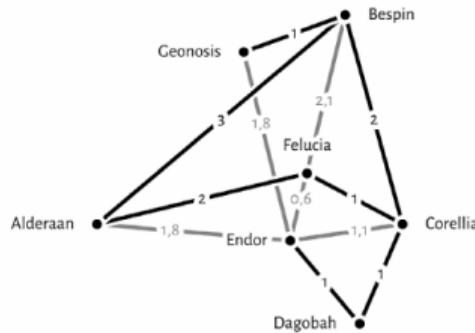
5+2,1 Alderaan - Felucia - Corellia - Bespin

Zusatzinfo der Restroute nutzen

Alderaan → Endor

<https://thedecisionlab.com/biases/heuristics>

Heuristischer Algorithmus



Wie machen Menschen das? → Distanzschätzung im Euklidischen Raum

→ Algorithmus bekommt Koordinaten der Planeten - Zusatzinformation

3+2,1 Alderaan - Bespin

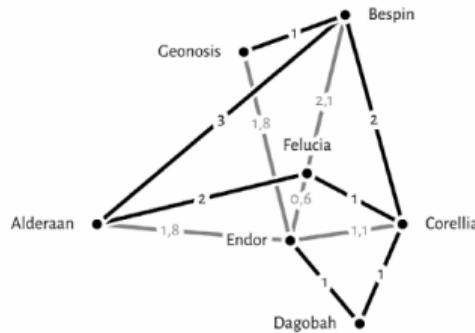
5+2,1 Alderaan - Felucia - Corellia - Bespin

5+0,0 Alderaan - Felucia - Corellia - Dagobah - Endor

Alderaan → Endor

<https://thedecisionlab.com/biases/heuristics>

Heuristischer Algorithmus



Wie machen Menschen das? → Distanzschätzung im Euklidischen Raum

→ Algorithmus bekommt Koordinaten der Planeten - Zusatzinformation

$3+2,1$ Alderaan - Bespin

$5+2,1$ Alderaan - Felucia - Corellia - Bespin

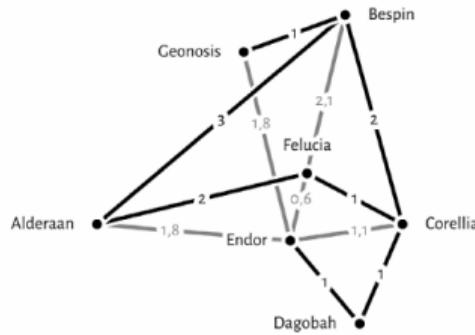
$5+0,0$ Alderaan - Felucia - Corellia - Dagobah - Endor

Alderaan → Endor

Suchalgorithmus, der für jede Route abschätzt, wie lange die Reise insgesamt dauern wird - **A***

<https://thedecisionlab.com/biases/heuristics>

Heuristischer Algorithmus



Wie machen Menschen das? → Distanzschätzung im Euklidischen Raum

→ Algorithmus bekommt Koordinaten der Planeten - Zusatzinformation

3+2,1 Alderaan - Bespin

5+2,1 Alderaan - Felucia - Corellia - Bespin

5+0,0 Alderaan - Felucia - Corellia - Dagobah - Endor

Alderaan → Endor

Suchalgorithmus, der für jede Route abschätzt, wie lange die Reise insgesamt dauern wird - **A***

Heuristik: mentale Abkürzungen (Shortcuts), helfen Probleme zu lösen, Wahrscheinlichkeiten abzuwägen, “Daumenregel” (rule of thumb)

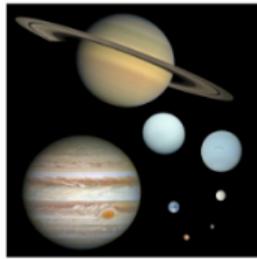
Abstraktion in der Informatik

Trennung der äußereren Eigenschaften eines Gegenstandes und seiner Funktion

Abstraktion in der Informatik

Trennung der äußereren Eigenschaften eines Gegenstandes und seiner Funktion

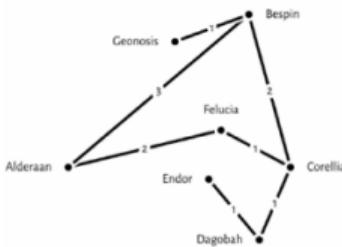
Realität/Welt



Abstraktion



Modell



Vorhersage



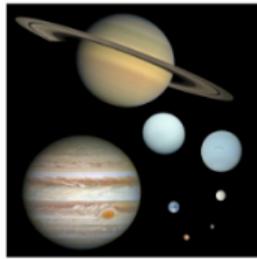
$$S \rightarrow [V_1(d_1, h_1), V_2(d_2, h_2), \dots, V_k(d_k, h_k)]$$

(S aktueller Ort, V_i – von hier zu erreichende Orte,
 d_i – Distanz bis hier, h_i – Entfernung zum Ziel)

Abstraktion in der Informatik

Trennung der äußereren Eigenschaften eines Gegenstandes und seiner Funktion

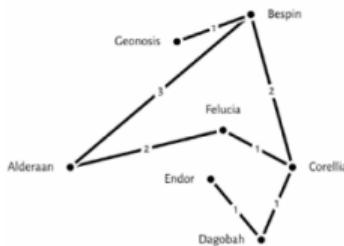
Realität/Welt



Abstraktion



Modell



Vorhersage



$$S \rightarrow [V_1(d_1, h_1), V_2(d_2, h_2), \dots, V_k(d_k, h_k)]$$

(S aktueller Ort, V_i – von hier zu erreichende Orte,
 d_i – Distanz bis hier, h_i – Entfernung zum Ziel)

Vorteile: Vereinfachung, Generalisierung

Nachteile: Entfremdung, Fehlschluss

Abstraktion

- abstractus (lat.) - “abgezogen” von abs-trahere - “abziehen, entfernen, trennen”
- *induktiver* Denkprozess, bei dem man Spezifika eines Problems weglässt und es auf etwas Allgemeineres oder Einfacheres überführt



Quellen:

<https://de.wikipedia.org/wiki/Abstraktion>,

[https://de.wikipedia.org/wiki/Induktion_\(Philosophie\)](https://de.wikipedia.org/wiki/Induktion_(Philosophie))

Abstraktion

- abstractus (lat.) - “abgezogen” von abs-trahere - “abziehen, entfernen, trennen”
- *induktiver* Denkprozess, bei dem man Spezifika eines Problems weglässt und es auf etwas Allgemeineres oder Einfacheres überführt
- Induktion (herbeiführen) = *abstrahierender* Schluss aus beobachteten Phänomenen auf allgemeinere Erkenntnis
- Deduktion (ableiten) = Schluss folgt zwingend/logisch aus bestimmten Annahmen

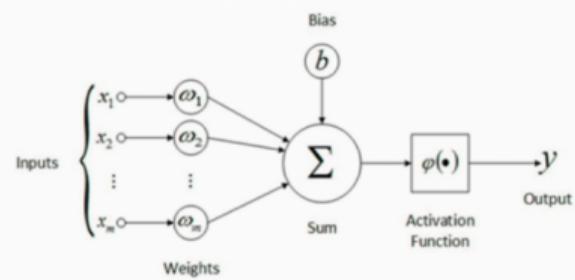
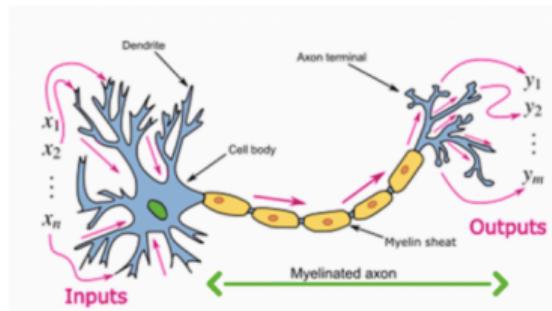
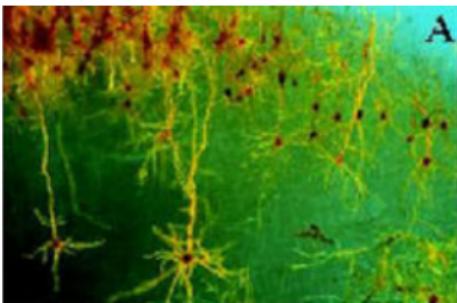


Quellen:

<https://de.wikipedia.org/wiki/Abstraktion>,

[https://de.wikipedia.org/wiki/Induktion_\(Philosophie\)](https://de.wikipedia.org/wiki/Induktion_(Philosophie))

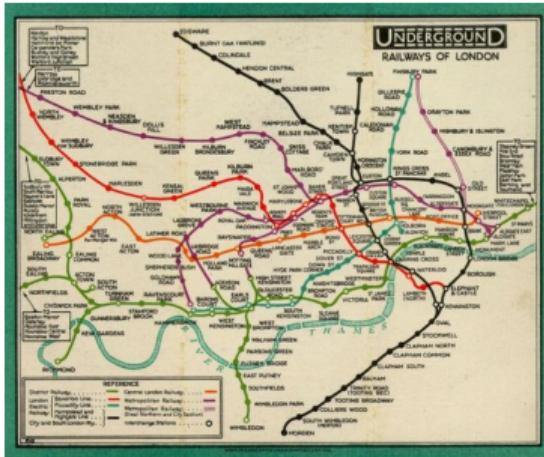
Beispiel: Neuronenmodell



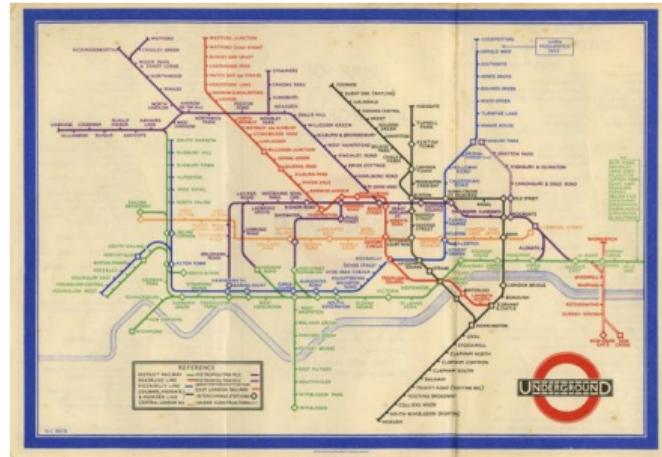
[http://www.scholarpedia.org/
article/Neuron#Dendritic_spikes](http://www.scholarpedia.org/article/Neuron#Dendritic_spikes)

<https://ttpsc.com/de/blog/maschinelles-lernen-thingworx-vs-covid-19/>

Beispiel: Lagedarstellungen



Fred Stingemore (1928) <https://www.ltmuseum.co.uk/collections/collections-online/maps/item/1991-245>



Henry C. Beck (1933) <https://www.ltmuseum.co.uk/collections/collections-online/maps/item/1999-321>

- realistische Lagebeziehungen unnötig → 45 deg Winkel der Linien erhöhen Übersichtlichkeit
- **Kunst des Weglassens und Vereinfachens** basiert auf Erfahrung

Abstraktion und die Kunst des Problemlösens

Algorithmen können Probleme lösen, die erstmal unterschiedlich aussehen, sich in ihrer Grundstruktur aber ähnlich sind und deshalb auf gleiche Weise gelöst werden können

Abstraktion und die Kunst des Problemlösens

Algorithmen können Probleme lösen, die erstmal unterschiedlich aussehen, sich in ihrer Grundstruktur aber ähnlich sind und deshalb auf gleiche Weise gelöst werden können

- Probleme lösen erfordert Abstraktion
- Phasen der Problemlösung (nach Polya, 1945)
 1. das Problem verstehen
 2. einen Plan zum Lösen des Problems entwerfen
 3. den Plan umsetzen
 4. die Lösung bewerten: wie gut passt sie und kann sie benutzt werden, um andere Probleme zu lösen (Generalisierbarkeit: abstrakt → konkret)

Abstraktion und die Kunst des Problemlösens

Algorithmen können Probleme lösen, die erstmal unterschiedlich aussehen, sich in ihrer Grundstruktur aber ähnlich sind und deshalb auf gleiche Weise gelöst werden können

- Probleme lösen erfordert Abstraktion
- Phasen der Problemlösung (nach Polya, 1945)
 1. das Problem verstehen
 2. einen Plan zum Lösen des Problems entwerfen
 3. den Plan umsetzen
 4. die Lösung bewerten: wie gut passt sie und kann sie benutzt werden, um andere Probleme zu lösen (Generalisierbarkeit: abstrakt → konkret)
- manchmal versteht man ein Problem erst, wenn man es gelöst hat

Beispiel: Ausgang offen

Person A hat die Aufgabe, das Alter der drei Kinder von Person B zu bestimmen. B teilt A mit, dass das **Produkt aus dem Alter der Kinder 36** Jahre beträgt. Nach einem Überlegen sagt Person A, dass sie einen weitereren Hinweis braucht. Also nennt B die **Summe des Alters der Kinder**. Wieder antwortet A, dass sie einen weiteren Hinweis benötigt und B erzählt, dass das **älteste Kind Klavier spielt**. Nachdem A diesen Hinweis gehört hat, sagt sie B das Alter der drei Kinder. Wie alt sind sie?

Beispiel: Ausgang offen

Person A hat die Aufgabe, das Alter der drei Kinder von Person B zu bestimmen. B teilt A mit, dass das **Produkt aus dem Alter der Kinder 36** Jahre beträgt. Nach einem Überlegen sagt Person A, dass sie einen weiteren Hinweis braucht. Also nennt B die **Summe des Alters der Kinder**. Wieder antwortet A, dass sie einen weiteren Hinweis benötigt und B erzählt, dass das **älteste Kind Klavier spielt**. Nachdem A diesen Hinweis gehört hat, sagt sie B das Alter der drei Kinder. Wie alt sind sie?

Dreier-Produkte = 36

(1,1,36) (1,6,6)

(1,2,18) (2,2,9)

(1,3,12) (2,3,6)

(1,4, 9) (3,3,4)

Beispiel: Ausgang offen

Person A hat die Aufgabe, das Alter der drei Kinder von Person B zu bestimmen. B teilt A mit, dass das **Produkt aus dem Alter der Kinder 36** Jahre beträgt. Nach einem Überlegen sagt Person A, dass sie einen weiteren Hinweis braucht. Also nennt B die **Summe des Alters der Kinder**. Wieder antwortet A, dass sie einen weiteren Hinweis benötigt und B erzählt, dass das **älteste Kind Klavier spielt**. Nachdem A diesen Hinweis gehört hat, sagt sie B das Alter der drei Kinder. Wie alt sind sie?

Dreier-Produkte = 36

(1,1,36) (1,6,6)

(1,2,18) (2,2,9)

(1,3,12) (2,3,6)

(1,4, 9) (3,3,4)

Summe der Triplets links

$1 + 1 + 36 = 38$ $1 + 6 + 6 = 13$

$1 + 2 + 18 = 21$ $2 + 2 + 9 = 13$

$1 + 3 + 12 = 16$ $2 + 3 + 6 = 11$

$1 + 4 + 9 = 14$ $3 + 3 + 4 = 10$

Beispiel: Ausgang offen

Person A hat die Aufgabe, das Alter der drei Kinder von Person B zu bestimmen. B teilt A mit, dass das **Produkt aus dem Alter der Kinder 36** Jahre beträgt. Nach einem Überlegen sagt Person A, dass sie einen weiteren Hinweis braucht. Also nennt B die **Summe des Alters der Kinder**. Wieder antwortet A, dass sie einen weiteren Hinweis benötigt und B erzählt, dass das **älteste Kind Klavier spielt**. Nachdem A diesen Hinweis gehört hat, sagt sie B das Alter der drei Kinder. Wie alt sind sie?

Dreier-Produkte = 36

(1,1,36) (1,6,6)

(1,2,18) (2,2,9)

(1,3,12) (2,3,6)

(1,4, 9) (3,3,4)

Summe der Triplets links

$1 + 1 + 36 = 38$ $1 + 6 + 6 = 13$

$1 + 2 + 18 = 21$ $2 + 2 + 9 = 13$

$1 + 3 + 12 = 16$ $2 + 3 + 6 = 11$

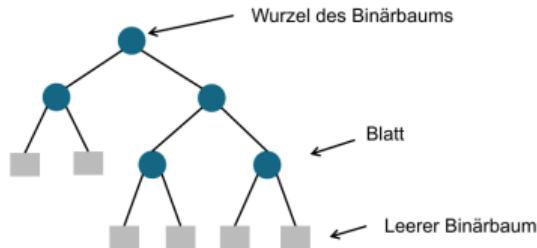
$1 + 4 + 9 = 14$ $3 + 3 + 4 = 10$

2, 2, 9

- erst beim Versuch das Problem zu lösen, kommt das Verständnis
- Inkubationszeit

Abstraktion in der Informatik

- Darstellung von Binärbäumen



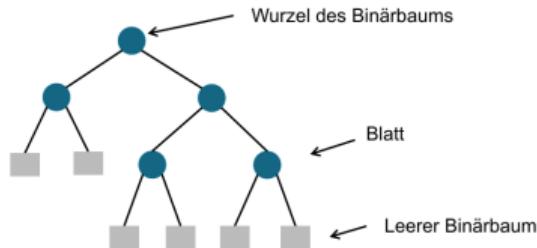
Bäume | Manfred Hauswirth | Einführung in die Programmierung, WS 25/26
Seite 10



- Algorithmen zum Sortieren von, Suchen in, Ändern der Struktur von Binärbäumen
- Abstraktion ist vorher passiert - welche Probleme lassen sich als Binärbäume abbilden?

Abstraktion in der Informatik

- Darstellung von Binärbäumen



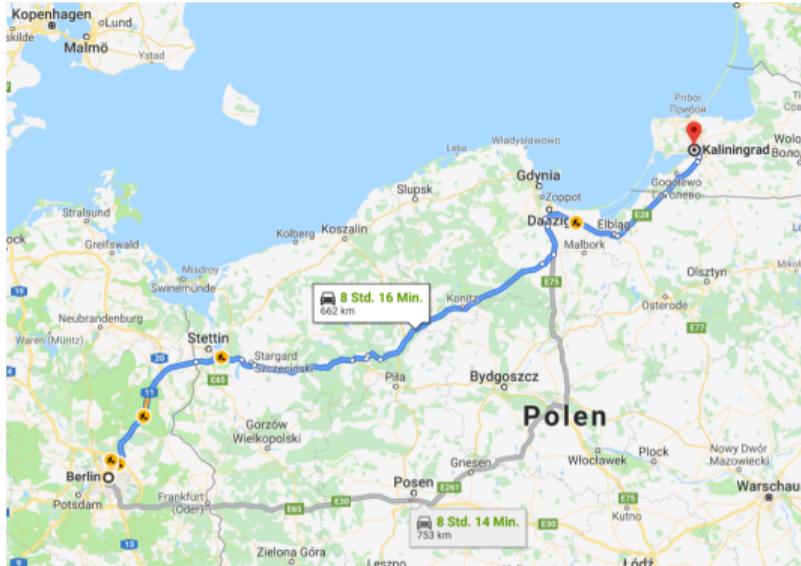
Bäume | Manfred Hauswirth | Einführung in die Programmierung, WS 25/26
Seite 10



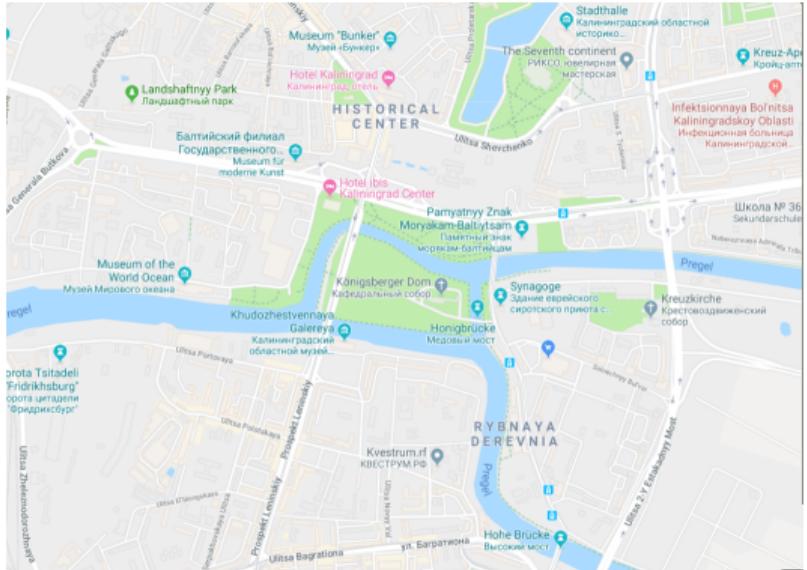
- Algorithmen zum Sortieren von, Suchen in, Ändern der Struktur von Binärbäumen
- Abstraktion ist vorher passiert - welche Probleme lassen sich als Binärbäume abbilden?

Ein Binärbaum ist ein gerichteter, azyklischer, zusammenhängender Graph, bei dem jeder Knoten höchstens zwei Kinder hat

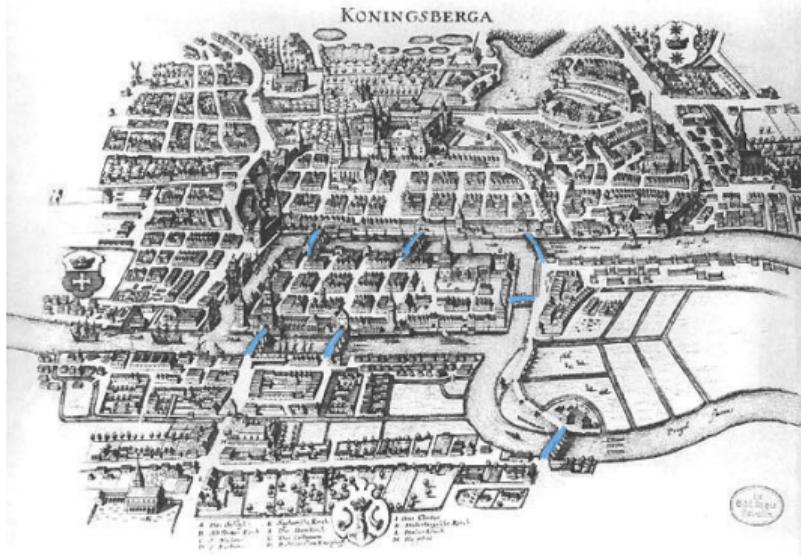
Abstraktionsklassiker: Königsberger Brückenproblem



Abstraktionsklassiker: Königsberger Brückenproblem



Abstraktionsklassiker: Königsberger Brückenproblem



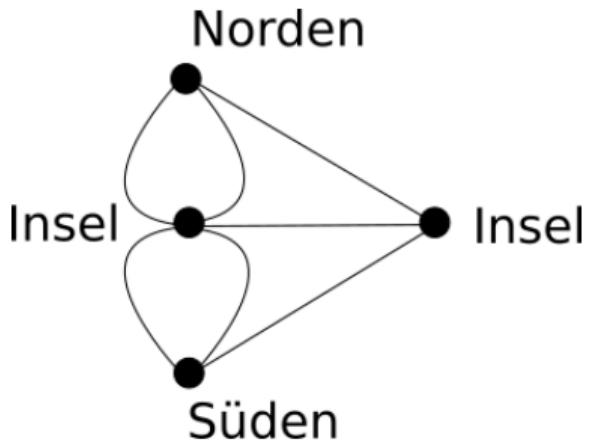
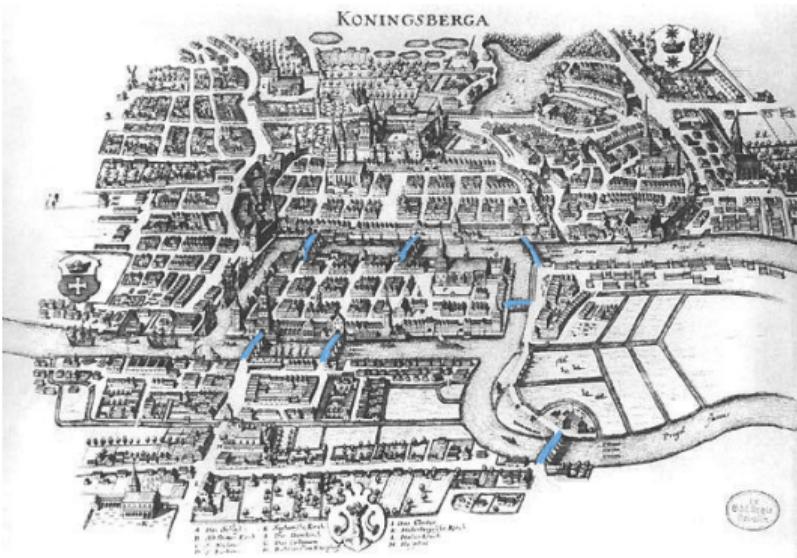
Im preußischen Königsberg gibt es eine Flussinsel, genannt der Kneiphof. Sie ist von zwei Flussarmen umgeben. Über die Arme dieses Flusses führen sieben Brücken. Es galt die Frage zu lösen, ob man einen Spazierweg so einrichten könne, dass man jede dieser Brücken einmal und nicht mehr als einmal überschreite.

– Leonhard Euler, 1707–1783

Gibt es einen Rundweg, der erlaubt, dass man

1. jede Brücke einmal überquert, und
2. am Ende wieder an den Ausgangspunkt zurückkommt?

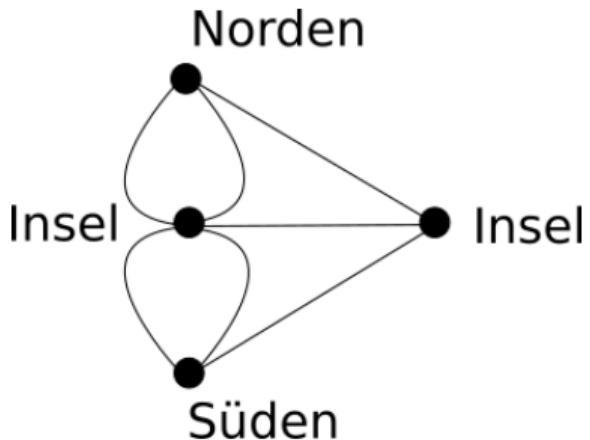
Königsberger Brückenproblem - Abstraktion



Beobachtungen

- Absolute Entfernungen und Positionen sind unwichtig!
- Lagen und relative Beziehungen wichtig!
- Euler lässt irrelevante Informationen weg und vereinfacht das Problem

Königsberger Brückenproblem - Abstraktion

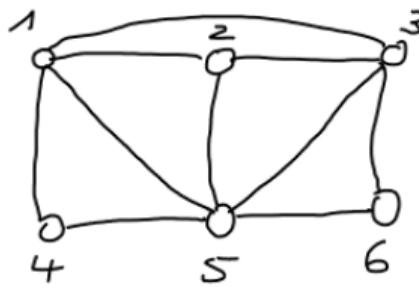


1736 erste Fragestellung der Graphentheorie, die damit von Euler begründet wird

Beobachtungen

- Absolute Entfernungen und Positionen sind unwichtig!
- Lagen und relative Beziehungen wichtig!
- Euler lässt irrelevante Informationen weg und vereinfacht das Problem

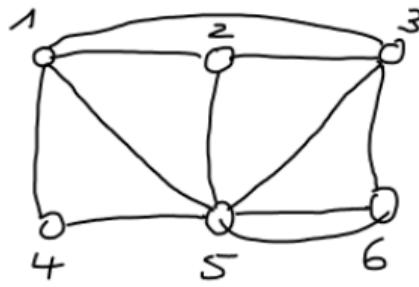
Graphentheorie: Terminologie



Graph repräsentiert paarweise Beziehungen zwischen Objekten

- Objekte = Knoten (Ecken), engl. **Vertices**, v_i z.B. Orte
- Beziehung zw. Objekten = Kanten, engl. **Edges** (v_i, v_{i+1}) z.B. Brücken
- Graph $G = (V, E)$

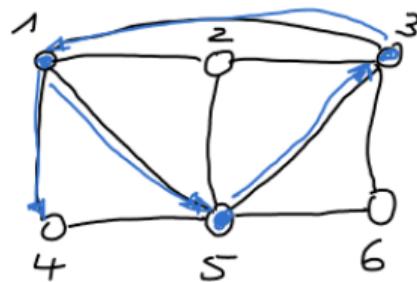
Graphentheorie: Terminologie



Graph repräsentiert paarweise Beziehungen zwischen Objekten

- Objekte = Knoten (Ecken), engl. **Vertices**, v_i z.B. Orte
- Beziehung zw. Objekten = Kanten, engl. **Edges** (v_i, v_{i+1}) z.B. Brücken
- Graph $G = (V, E)$
- “Multigraph”: mehr als 1 Verbindung zw. 2 Punkten

Graphentheorie: Terminologie

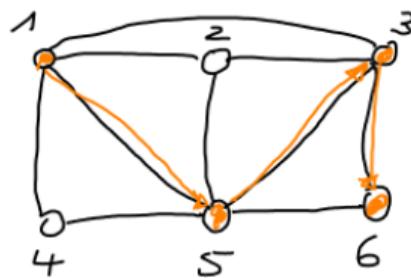


Graph repräsentiert paarweise Beziehungen zwischen Objekten

- Objekte = Knoten (Ecken), engl. **Vertices**, v_i z.B. Orte
- Beziehung zw. Objekten = Kanten, engl. **Edges** (v_i, v_{i+1}) z.B. Brücken
- Graph $G = (V, E)$
- “Multigraph”: mehr als 1 Verbindung zw. 2 Punkten

- **Weg** - Folge von Knoten v_1, \dots, v_n , in der je 2 aufeinanderfolgende Knoten durch eine Kante verbunden sind, z.B. $\{v_1, v_5\}, \{v_5, v_3\}, \{v_3, v_1\}, \{v_1, v_4\}$, d.h. $\{v_i, v_{i+1}\} \in E$ für alle $i < n$ (Kantenzug)

Graphentheorie: Terminologie

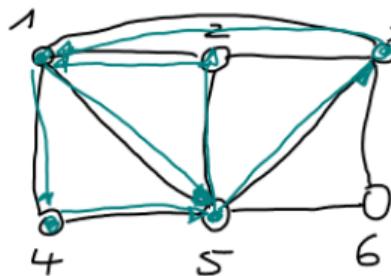


Graph repräsentiert paarweise Beziehungen zwischen Objekten

- Objekte = Knoten (Ecken), engl. **Vertices**, v_i z.B. Orte
- Beziehung zw. Objekten = Kanten, engl. **Edges** (v_i, v_{i+1}) z.B. Brücken
- Graph $G = (V, E)$
- “Multigraph”: mehr als 1 Verbindung zw. 2 Punkten

- **Weg** - Folge von Knoten v_1, \dots, v_n , in der je 2 aufeinanderfolgende Knoten durch eine Kante verbunden sind, z.B. $\{v_1, v_5\}, \{v_5, v_3\}, \{v_3, v_1\}, \{v_1, v_4\}$, d.h. $\{v_i, v_{i+1}\} \in E$ für alle $i < n$ (Kantenzug)
- Sind alle Knoten eines Weges verschieden, nennt man ihn **Pfad**.

Graphentheorie: Terminologie

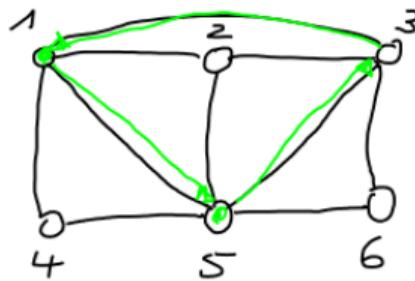


Graph repräsentiert paarweise Beziehungen zwischen Objekten

- Objekte = Knoten (Ecken), engl. **Vertices**, v_i z.B. Orte
- Beziehung zw. Objekten = Kanten, engl. **Edges** (v_i, v_{i+1}) z.B. Brücken
- Graph $G = (V, E)$
- “Multigraph”: mehr als 1 Verbindung zw. 2 Punkten

- **Weg** - Folge von Knoten v_1, \dots, v_n , in der je 2 aufeinanderfolgende Knoten durch eine Kante verbunden sind, z.B. $\{v_1, v_5\}, \{v_5, v_3\}, \{v_3, v_1\}, \{v_1, v_4\}$, d.h. $\{v_i, v_{i+1}\} \in E$ für alle $i < n$ (Kantenzug)
- Sind alle Knoten eines Weges verschieden, nennt man ihn **Pfad**.
- Ist $v_1 == v_n$ spricht man von einem **Zyklus**. $\{v_1, v_5\}, \{v_5, v_3\}, \{v_3, v_1\}, \{v_1, v_4\}, \dots, \{v_2, v_1\}$

Graphentheorie: Terminologie

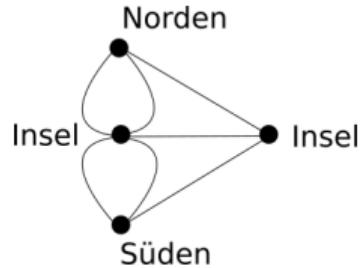
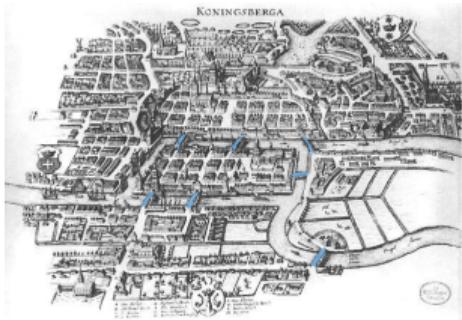


Graph repräsentiert paarweise Beziehungen zwischen Objekten

- Objekte = Knoten (Ecken), engl. **Vertices**, v_i z.B. Orte
- Beziehung zw. Objekten = Kanten, engl. **Edges** (v_i, v_{i+1}) z.B. Brücken
- Graph $G = (V, E)$
- “Multigraph”: mehr als 1 Verbindung zw. 2 Punkten

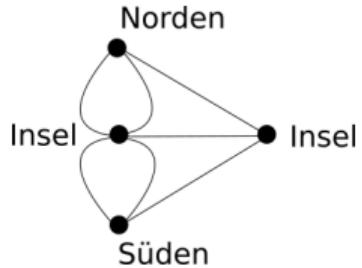
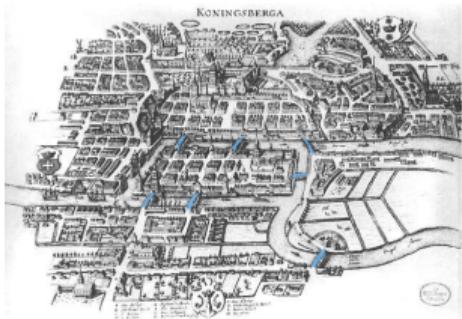
- **Weg** - Folge von Knoten v_1, \dots, v_n , in der je 2 aufeinanderfolgende Knoten durch eine Kante verbunden sind, z.B. $\{v_1, v_5\}, \{v_5, v_3\}, \{v_3, v_1\}, \{v_1, v_4\}$, d.h. $\{v_i, v_{i+1}\} \in E$ für alle $i < n$ (Kantenzug)
- Sind alle Knoten eines Weges verschieden, nennt man ihn **Pfad**.
- Ist $v_1 == v_n$ spricht man von einem **Zyklus**. $\{v_1, v_5\}, \{v_5, v_3\}, \{v_3, v_1\}, \{v_1, v_4\}, \dots, \{v_2, v_1\}$
- Zyklus, bei dem ausser v_1 und v_n alle Knoten verschieden sind, heißt **Kreis**

Zurück nach Königsberg: Eulerkreis & Eulerpfad



- jede Brücke einmal
- am Ende zurück zum Ausgang

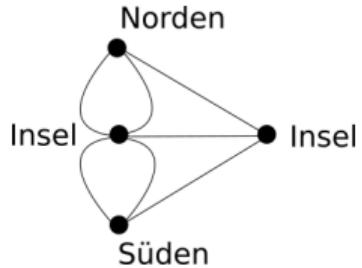
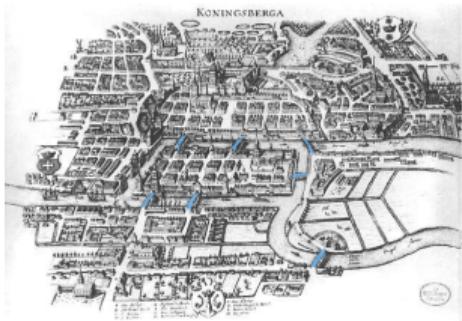
Zurück nach Königsberg: Eulerkreis & Eulerpfad



- jede Brücke einmal
- am Ende zurück zum Ausgang

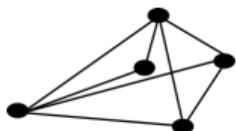
gesucht: ein Weg $\{v_1, \dots, v_n\}$, bei dem jede Kante des Graphen genau einmal als (v_i, v_{i+1}) vorkommt und jeder Knoten v_i mind. einmal besucht wird = **Eulerpfad**

Zurück nach Königsberg: Eulerkreis & Eulerpfad

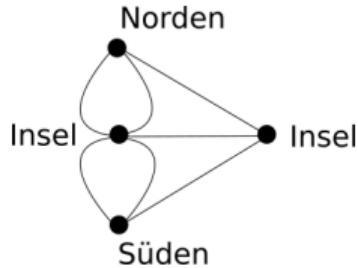


- jede Brücke einmal
- am Ende zurück zum Ausgang

gesucht: ein Weg $\{v_1, \dots, v_n\}$, bei dem jede Kante des Graphen genau einmal als (v_i, v_{i+1}) vorkommt und jeder Knoten v_i mind. einmal besucht wird = **Eulerpfad**

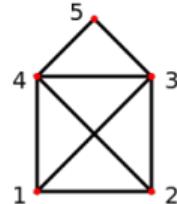
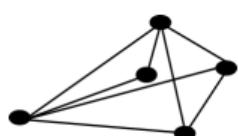


Zurück nach Königsberg: Eulerkreis & Eulerpfad

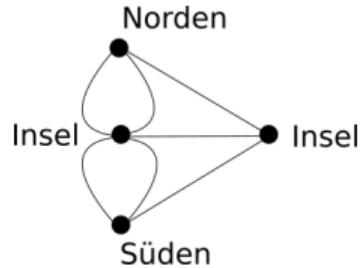


- jede Brücke einmal
- am Ende zurück zum Ausgang

gesucht: ein Weg $\{v_1, \dots, v_n\}$, bei dem jede Kante des Graphen genau einmal als (v_i, v_{i+1}) vorkommt und jeder Knoten v_i mind. einmal besucht wird = **Eulerpfad**

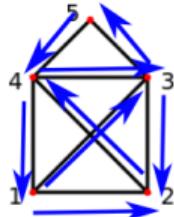
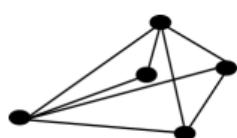


Zurück nach Königsberg: Eulerkreis & Eulerpfad



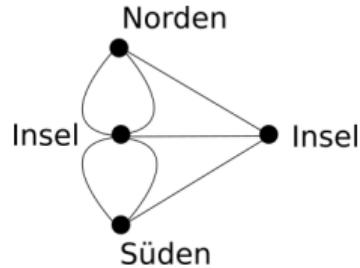
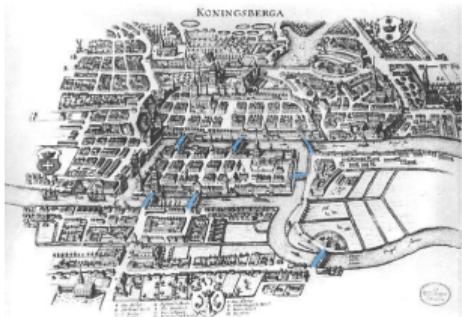
- jede Brücke einmal
- am Ende zurück zum Ausgang

gesucht: ein Weg $\{v_1, \dots, v_n\}$, bei dem jede Kante des Graphen genau einmal als (v_i, v_{i+1}) vorkommt und jeder Knoten v_i mind. einmal besucht wird = **Eulerpfad**



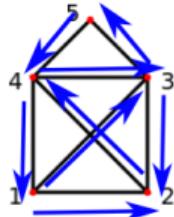
- Knoten können mehrmals vorkommen
- Ausgangs- \neq Endpunkt

Zurück nach Königsberg: Eulerkreis & Eulerpfad



- jede Brücke einmal
- am Ende zurück zum Ausgang

gesucht: ein Weg $\{v_1, \dots, v_n\}$, bei dem jede Kante des Graphen genau einmal als (v_i, v_{i+1}) vorkommt und jeder Knoten v_i mind. einmal besucht wird = **Eulerpfad**



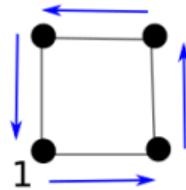
- Knoten können mehrmals vorkommen
- Ausgangs- \neq Endpunkt
- Eulerpfad, der gleichzeitig ein Zyklus ist, d.h.
 $v_1 == v_n$, heißt **Eulerkreis**

Eulerkreis durch Königsberger Brücken?

Wann gibt es einen Eulerkreis ($v_1 == v_n$) durch einen Graphen?

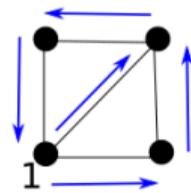
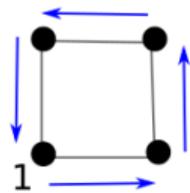
Eulerkreis durch Königsberger Brücken?

Wann gibt es einen Eulerkreis ($v_1 == v_n$) durch einen Graphen?



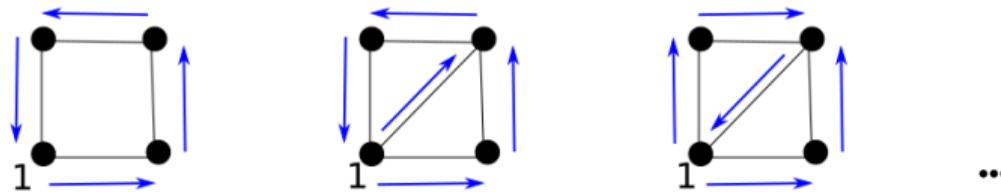
Eulerkreis durch Königsberger Brücken?

Wann gibt es einen Eulerkreis ($v_1 == v_n$) durch einen Graphen?



Eulerkreis durch Königsberger Brücken?

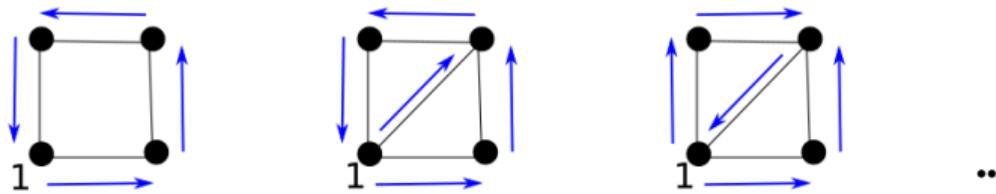
Wann gibt es einen Eulerkreis ($v_1 == v_n$) durch einen Graphen?



- Versuch-und-Irrtum (trial-and-error)
- Programm schreiben, das rumprobiert - *numerische Lösung*

Eulerkreis durch Königsberger Brücken?

Wann gibt es einen Eulerkreis ($v_1 == v_n$) durch einen Graphen?



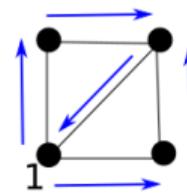
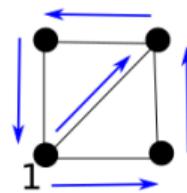
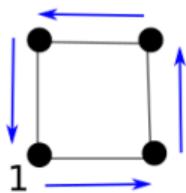
- Versuch-und-Irrtum (trial-and-error)
- Programm schreiben, das rumprobiert - *numerische Lösung*
- Dinge sind schwieriger als man ihnen auf den ersten Blick ansieht (Kombinatorik)



von https://de.wikipedia.org/wiki/Haus_vom_Nikolaus

Eulerkreis durch Königsberger Brücken?

Wann gibt es einen Eulerkreis ($v_1 == v_n$) durch einen Graphen?



...



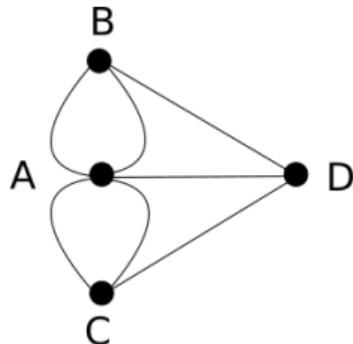
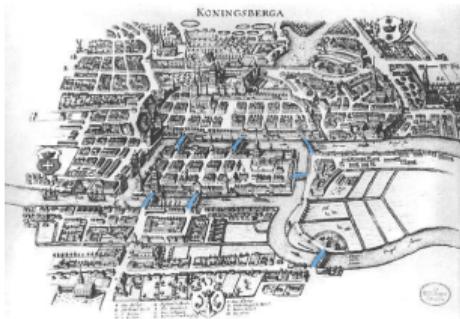
von https://de.wikipedia.org/wiki/Haus_vom_Nikolaus

- Versuch-und-Irrtum (trial-and-error)
- Programm schreiben, das rumprobiert - *numerische Lösung*
- Dinge sind schwieriger als man ihnen auf den ersten Blick ansieht (Kombinatorik)

→ Graphentheorie: *analytische Lösung*

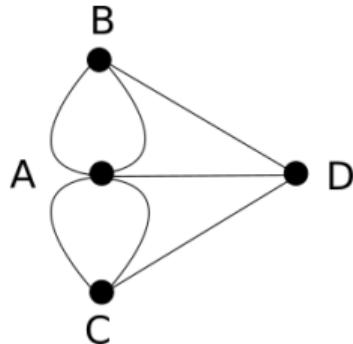
- **notwendige** und **hinreichende** Bedingung, dass Eulerkreis durch einen Graphen existiert
- leichter zu überprüfen als alle Möglichkeiten durchzutesten

Eulerkreis durch Königsberger Brücken?

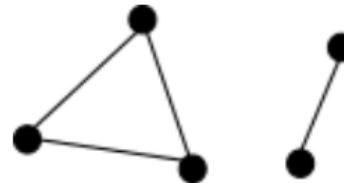


- jede Brücke einmal
- am Ende zurück zum Ausgang

Eulerkreis durch Königsberger Brücken?



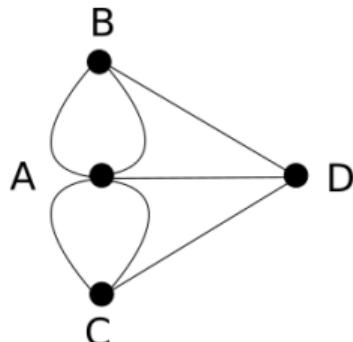
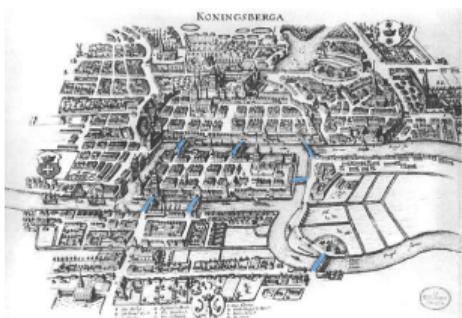
- jede Brücke einmal
- am Ende zurück zum Ausgang



notwendige Bedingung

1. Graph muß zusammenhängend sein

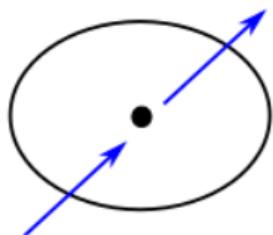
Eulerkreis durch Königsberger Brücken?



- jede Brücke einmal
- am Ende zurück zum Ausgang

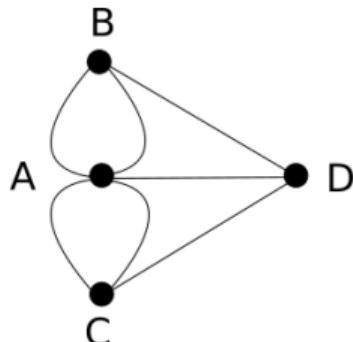
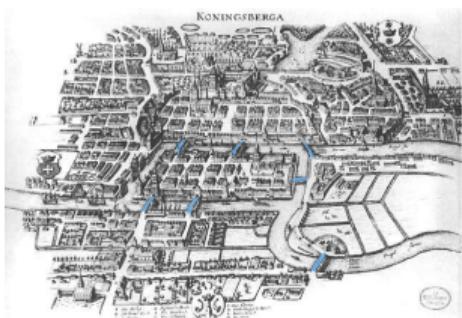


notwendige Bedingung



1. Graph muß zusammenhängend sein
2. jeder Knoten muß gerade Anzahl von Kanten haben, muß **geraden Grad** haben

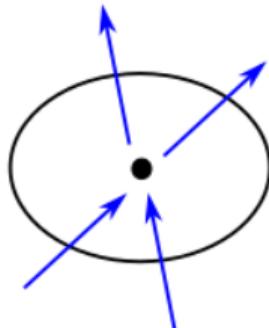
Eulerkreis durch Königsberger Brücken?



- jede Brücke einmal
- am Ende zurück zum Ausgang

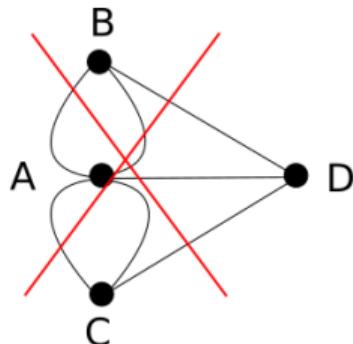
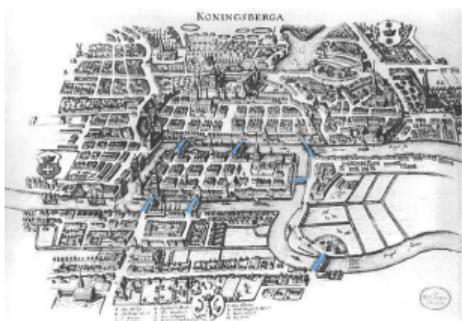


notwendige Bedingung



1. Graph muß zusammenhängend sein
2. jeder Knoten muß gerade Anzahl von Kanten haben, muß **geraden Grad** haben

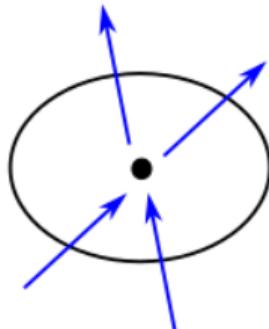
Eulerkreis durch Königsberger Brücken?



- jede Brücke einmal
- am Ende zurück zum Ausgang

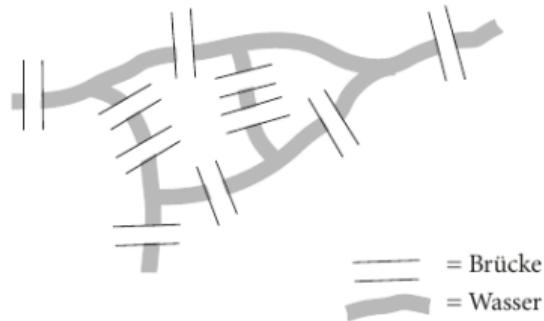


notwendige Bedingung



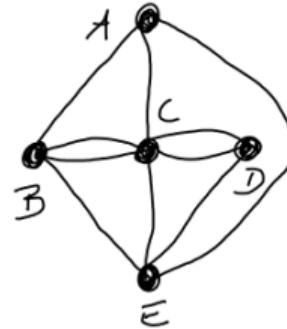
1. Graph muß zusammenhängend sein
2. jeder Knoten muß gerade Anzahl von Kanten haben, muß **geraden Grad** haben
3. wenn genau 2 Knoten ungeraden Grad haben, hat der Graph einen Eulerpfad aber keinen Eulerkreis

Aufgabe: Graphentheorie



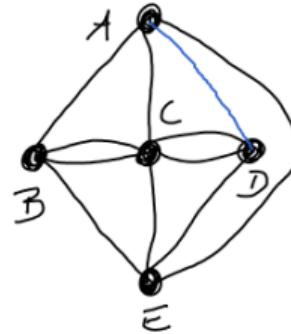
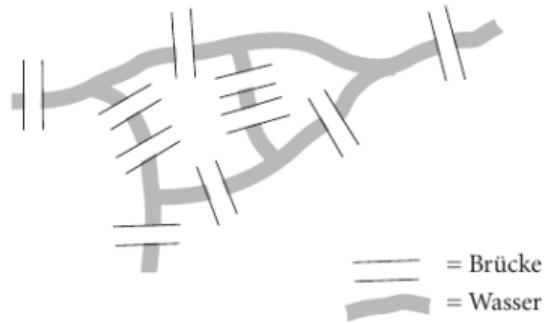
1. Zeichnen Sie zu dem Stadtplan den entsprechenden Graphen. Bestimmen Sie den Grad jedes Knoten.
2. Ist es in dieser Stadt möglich, einen Spaziergang zu machen, der über alle Brücken genau einmal führt? Begründen Sie. Beschreiben Sie einen möglichen Verlauf des Spaziergangs.
3. Bauen Sie eine oder mehrere Brücken, so dass ein Eulerkreis möglich ist. Beschreiben Sie einen möglichen Verlauf.

Aufgabe: Graphentheorie



1. Zeichnen Sie zu dem Stadtplan den entsprechenden Graphen. Bestimmen Sie den Grad jedes Knoten.
2. Ist es in dieser Stadt möglich, einen Spaziergang zu machen, der über alle Brücken genau einmal führt? Begründen Sie. Beschreiben Sie einen möglichen Verlauf des Spaziergangs.
3. Bauen Sie eine oder mehrere Brücken, so dass ein Eulerkreis möglich ist. Beschreiben Sie einen möglichen Verlauf.

Aufgabe: Graphentheorie



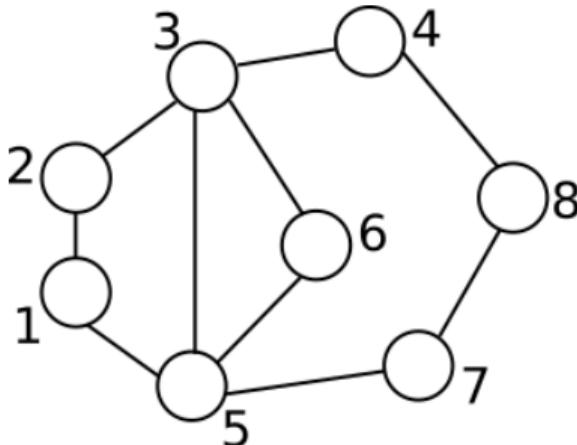
1. Zeichnen Sie zu dem Stadtplan den entsprechenden Graphen. Bestimmen Sie den Grad jedes Knoten.
2. Ist es in dieser Stadt möglich, einen Spaziergang zu machen, der über alle Brücken genau einmal führt? Begründen Sie. Beschreiben Sie einen möglichen Verlauf des Spaziergangs.
3. Bauen Sie eine oder mehrere Brücken, so dass ein Eulerkreis möglich ist. Beschreiben Sie einen möglichen Verlauf.

Der Algorithmus von Hierholzer, 1871

- findet Eulerkreis in einem Graphen

Vorüberlegung:

Zyklus ist ein Weg in einem Graphen, bei dem $v_1 == v_n$.



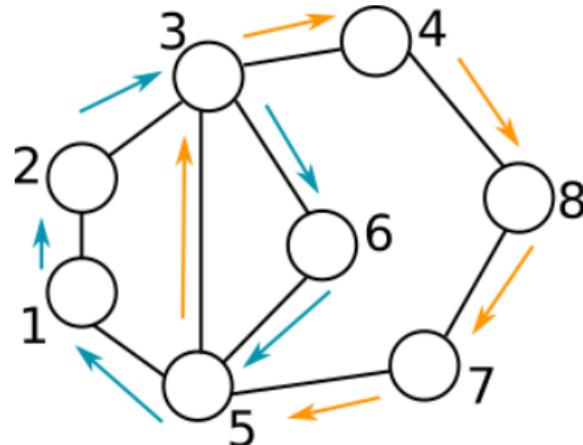
Der Algorithmus von Hierholzer, 1871

- findet Eulerkreis in einem Graphen

Vorüberlegung:

Zyklus ist ein Weg in einem Graphen, bei dem $v_1 == v_n$.

- wenn 1 Graph 2 Zyklen hat
z.B. (1,2,3,6,5,1) und (3,4,8,7,5,3)
- und wenn die beiden Zyklen mind. 1 Knoten teilen



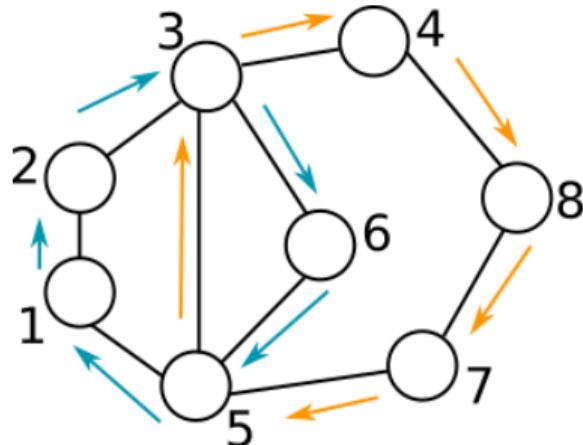
Der Algorithmus von Hierholzer, 1871

- findet Eulerkreis in einem Graphen

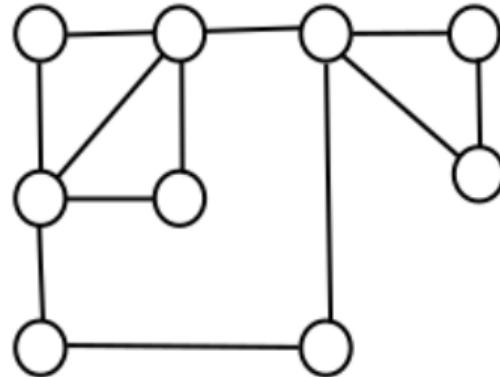
Vorüberlegung:

Zyklus ist ein Weg in einem Graphen, bei dem $v_1 == v_n$.

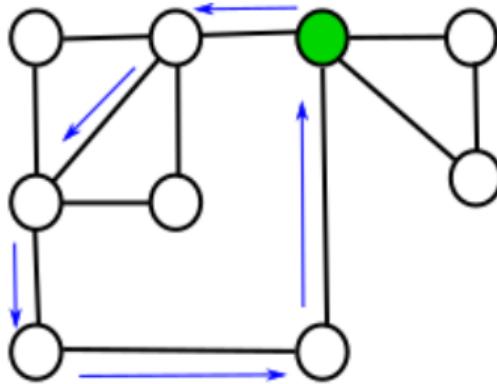
- wenn 1 Graph 2 Zyklen hat
z.B. (1,2,3,6,5,1) und (3,4,8,7,5,3)
- und wenn die beiden Zyklen mind. 1 Knoten teilen
- ⇒ dann kann man aus beiden Zyklen einen neuen Zyklus bauen, der beide Kantenmengen vereinigt z.B. (1,2,3,4,8,7,5,3,6,5,1)



Der Algorithmus von Hierholzer, 1871

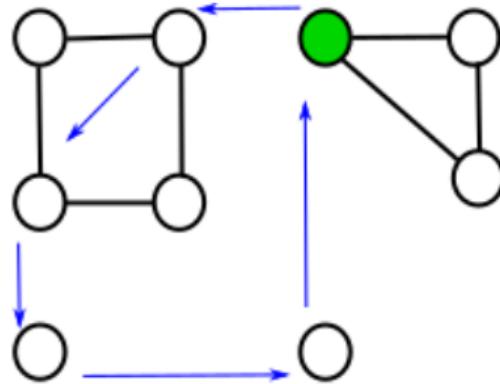


Der Algorithmus von Hierholzer, 1871



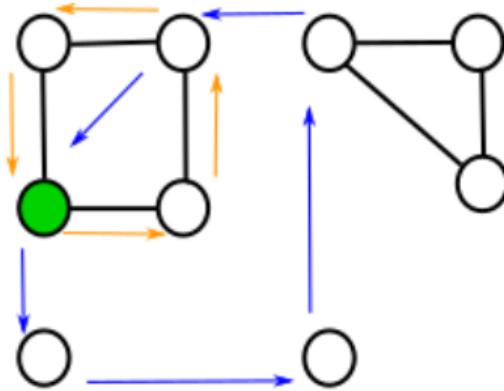
1. nimm beliebigen Startknoten & laufe los bis Du wieder am Startknoten an kommst \Rightarrow Zyklus

Der Algorithmus von Hierholzer, 1871



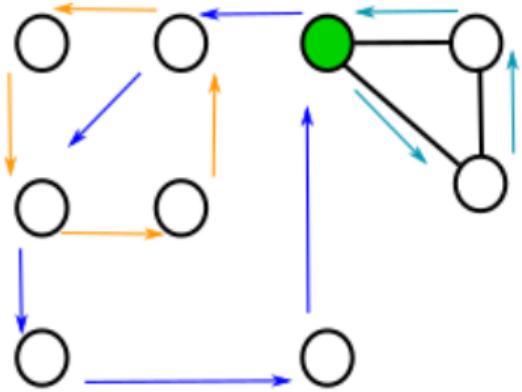
1. nimm beliebigen Startknoten & laufe los bis Du wieder am Startknoten an kommst \Rightarrow Zyklus \rightarrow entferne alle Kanten

Der Algorithmus von Hierholzer, 1871



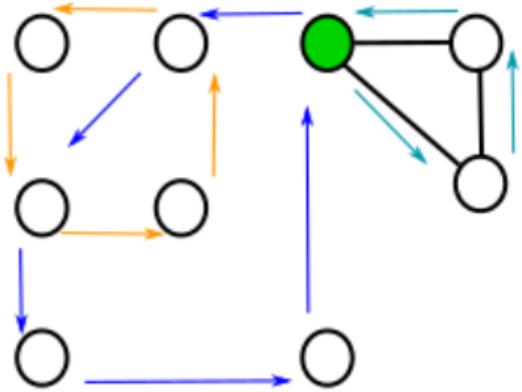
1. nimm beliebigen Startknoten & laufe los bis Du wieder am Startknoten an kommst \Rightarrow Zyklus \rightarrow entferne alle Kanten
2. starte von einem Knoten, von dem noch Kanten ausgehen und laufe auf unbefahrenen Kanten bis zum Startknoten

Der Algorithmus von Hierholzer, 1871



1. nimm beliebigen Startknoten & laufe los bis Du wieder am Startknoten an kommst \Rightarrow Zyklus \rightarrow entferne alle Kanten
2. starte von einem Knoten, von dem noch Kanten ausgehen und laufe auf unbenutzten Kanten bis zum Startknoten \Rightarrow Zyklus' \rightarrow entferne alle Kanten

Der Algorithmus von Hierholzer, 1871



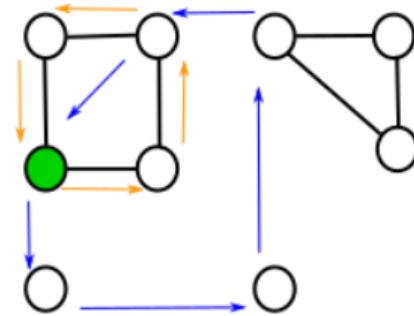
1. nimm beliebigen Startknoten & laufe los bis Du wieder am Startknoten an kommst \Rightarrow Zyklus \rightarrow entferne alle Kanten
2. starte von einem Knoten, von dem noch Kanten ausgehen und laufe auf unbenutzten Kanten bis zum Startknoten \Rightarrow Zyklus' \rightarrow entferne alle Kanten
3. vereinige die Zyklen und wiederhole 2.

Der Algorithmus von Hierholzer, 1871

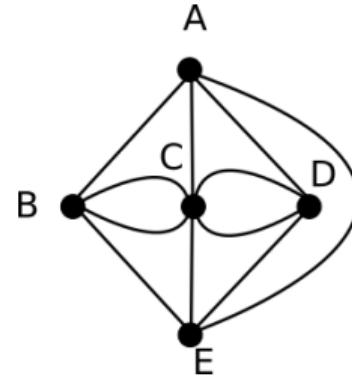
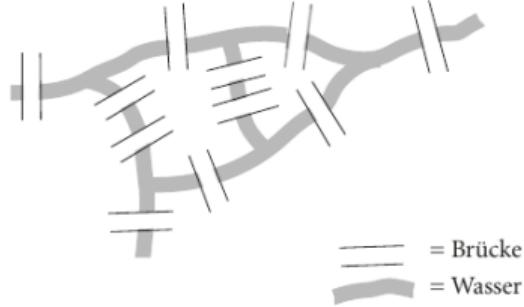
Voraussetzung:

$G = (V, E)$ ist **zusammenhängend**, alle Knoten haben **geraden Grad**

1. Wähle beliebigen Knoten v_0 in G . Konstruiere von v_0 ausgehend einen Unterkreis K in G , der keine Kante in G zweimal durchläuft
2. Wenn K ein Eulerkreis in G ist, brich ab,
sonst: lösche nun alle Kanten des Unterkreises K .
3. am ersten Knoten von K , dessen Grad größer 0 ist, starte weiteren Unterkreis K' , der keine Kante in G zweimal durchläuft
4. füge K' in K ein, indem der Startknoten von K' durch alle Knoten von K' in der richtigen Reihenfolge ersetzt wird. Nenne den so erhaltenen Kreis K . Fahre bei Schritt 2 fort.

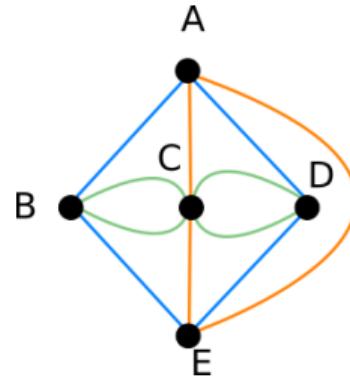


Aufgabe: Eulerkreis mit Algorithmus von Hierholzer



Finden Sie einen Eulerkreis in dem Graphen mit dem Algorithmus von Hierholzer.

Aufgabe: Eulerkreis mit Algorithmus von Hierholzer



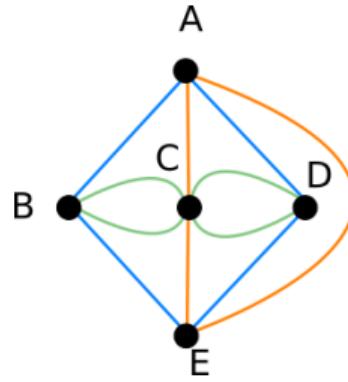
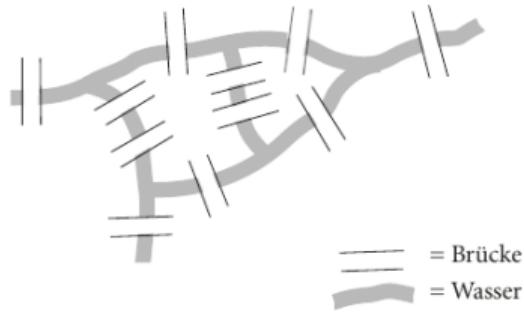
Finden Sie einen Eulerkreis in dem Graphen mit dem Algorithmus von Hierholzer.

Beispiel:

Kreis K: A C E A

Kreis K': C D C B C

Aufgabe: Eulerkreis mit Algorithmus von Hierholzer



Finden Sie einen Eulerkreis in dem Graphen mit dem Algorithmus von Hierholzer.

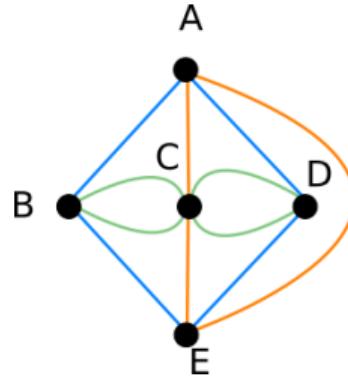
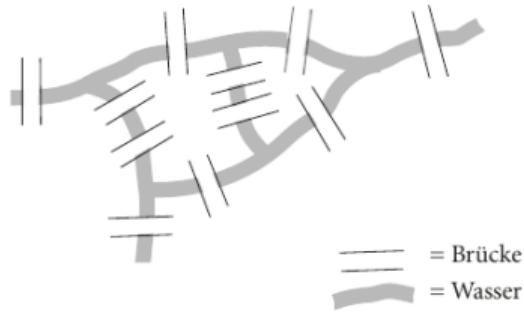
Beispiel:

Kreis K: A C E A

Kreis K': C D C B C

neuer Kreis K: A C D C B C E A

Aufgabe: Eulerkreis mit Algorithmus von Hierholzer



Finden Sie einen Eulerkreis in dem Graphen mit dem Algorithmus von Hierholzer.

Beispiel:

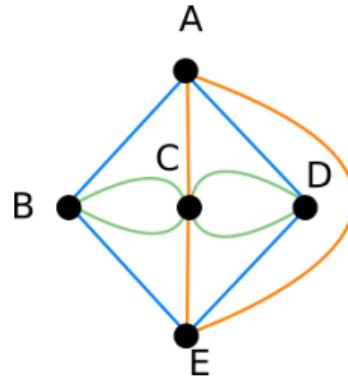
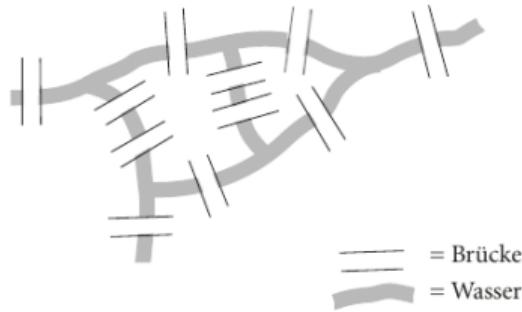
Kreis K: **A C E A**

Kreis K': **C D C B C**

neuer Kreis K: **A C D C B C E A**

neuer Kreis K': **A B E D A**

Aufgabe: Eulerkreis mit Algorithmus von Hierholzer



Finden Sie einen Eulerkreis in dem Graphen mit dem Algorithmus von Hierholzer.

Beispiel:

Kreis K: **A C E A**

Kreis K': **C D C B C**

neuer Kreis K: **A C D C B C E A**

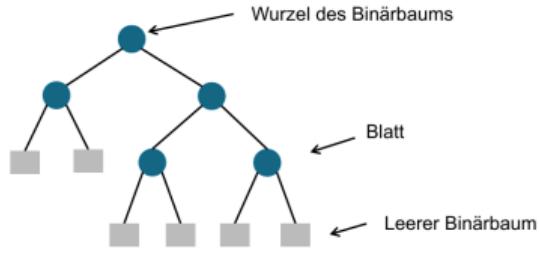
neuer Kreis K': **A B E D A**

neuer Graph K: **A C D C B C E A B E D A**

Binärbäume sind spezielle Graphen

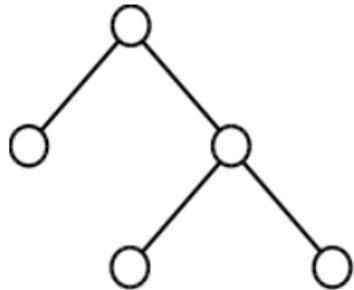
Binärbäume

- Darstellung von Binärbäumen

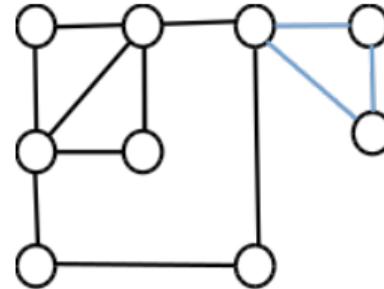


Binärbäume sind spezielle Graphen

Binärbäume



Graphen



- gerichteter, azyklischer, zusammenhängender Graph
- jeder Knoten hat höchstens zwei Kinder

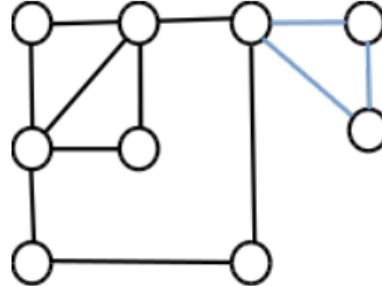
- Zyklen (blau)
- beliebig viele Kanten pro Knoten
- keine Hierarchie, keine Wurzel, keine Kinder/Eltern

Binärbäume sind spezielle Graphen

Graphalgorithmen

- Traversierungsalgorithmen (Durchlaufen)
 - Hierholzer
 - Fleury
 - ...
- Suchalgorithmen
 - Tiefensuche
 - Breitensuche
 - A*
 - ...

Graphen



- Zyklen (blau)
- beliebig viele Kanten pro Knoten
- keine Hierarchie, keine Wurzel, keine Kinder/Eltern

Anwendungen der Graphentheorie

- Graphentheorie Werkzeug NACHDEM die Abstraktion stattgefunden hat
- Schnittfeld von Mathematik und Informatik
- Graphen sind mathematische Modelle für netzartige Strukturen in der Welt

Anwendungen der Graphentheorie

- Graphentheorie Werkzeug NACHDEM die Abstraktion stattgefunden hat
- Schnittfeld von Mathematik und Informatik
- Graphen sind mathematische Modelle für netzartige Strukturen in der Welt
- Soziale Netzwerke (V - Personen, E - Beziehungen)



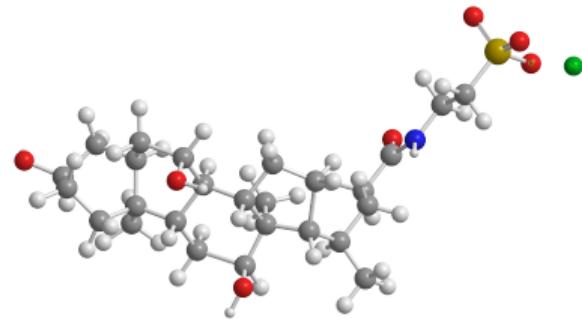
Anwendungen der Graphentheorie

- Graphentheorie Werkzeug NACHDEM die Abstraktion stattgefunden hat
- Schnittfeld von Mathematik und Informatik
- Graphen sind mathematische Modelle für netzartige Strukturen in der Welt
- Soziale Netzwerke (V - Personen, E - Beziehungen)
- Spielpläne (V - Teams, E - Begegnungen)



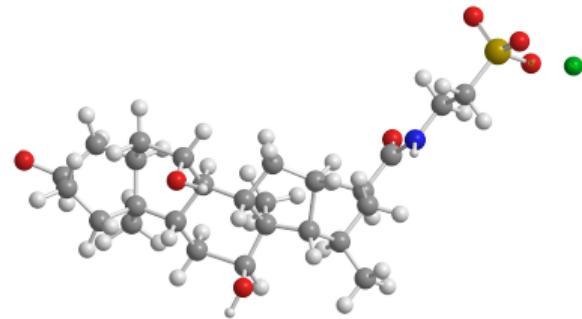
Anwendungen der Graphentheorie

- Graphentheorie Werkzeug NACHDEM die Abstraktion stattgefunden hat
- Schnittfeld von Mathematik und Informatik
- Graphen sind mathematische Modelle für netzartige Strukturen in der Welt
- Soziale Netzwerke (V - Personen, E - Beziehungen)
- Spielpläne (V - Teams, E - Begegnungen)
- Verarbeitung von Molekülstrukturen (V - Atome, K - Bindungen)



Anwendungen der Graphentheorie

- Graphentheorie Werkzeug NACHDEM die Abstraktion stattgefunden hat
- Schnittfeld von Mathematik und Informatik
- Graphen sind mathematische Modelle für netzartige Strukturen in der Welt
- Soziale Netzwerke (V - Personen, E - Beziehungen)
- Spielpläne (V - Teams, E - Begegnungen)
- Verarbeitung von Molekülstrukturen (V - Atome, K - Bindungen)
- Personalplanung, Kommunikationsnetze, Zuordnungen ...



Validität

Leseauftrag: Balzert

1. Was bedeutet Validität?
2. Was sind mögliche Fehlerquellen?



Validität

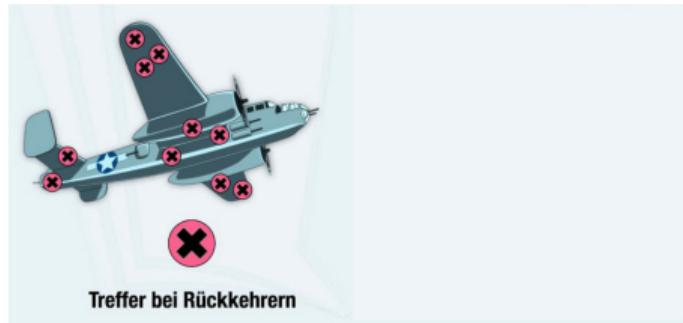
Leseauftrag: Balzert

1. Was bedeutet Validität?
2. Was sind mögliche Fehlerquellen?
 1. eine Messung ist valide, wenn sie das misst, was gemessen werden soll
 2. Stichprobe zu klein oder falsch



Welches Validitätsproblem besteht hier?

Im Zweiten Weltkrieg wurde die US-amerikanische Fliegerstaffel regelmässig analysiert, um die Flugzeuge zu optimieren. Unter anderem wollte man besonders anfällige Stellen mit einer besseren Panzerung ausstatten. Hierzu untersuchte man Flugzeuge, die aus dem Gefecht zurückgekehrt waren: Welche Stellen waren besonders oft von Schüssen durchlöchert worden? Die Idee dahinter: Dort, wo sie die meisten Einschusslöcher hatten, sollten die Flugzeuge stärker gepanzert werden.

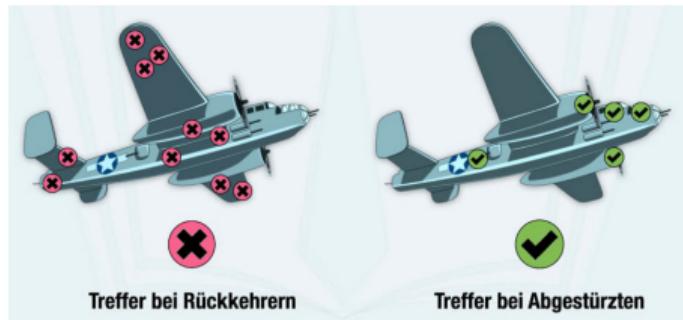


<https://www.hirnliga.ch/de/artikel-das-gehirn-slider/stolperfallen>

<https://karrierebibel.de/survivorship-bias/>

Welches Validitätsproblem besteht hier?

Im Zweiten Weltkrieg wurde die US-amerikanische Fliegerstaffel regelmässig analysiert, um die Flugzeuge zu optimieren. Unter anderem wollte man besonders anfällige Stellen mit einer besseren Panzerung ausstatten. Hierzu untersuchte man Flugzeuge, die aus dem Gefecht zurückgekehrt waren: Welche Stellen waren besonders oft von Schüssen durchlöchert worden? Die Idee dahinter: Dort, wo sie die meisten Einschusslöcher hatten, sollten die Flugzeuge stärker gepanzert werden.

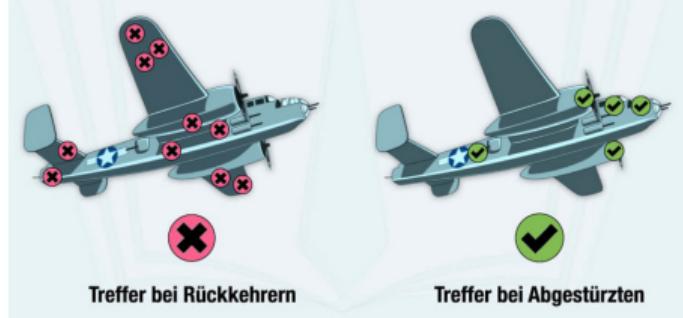


		Grundgesamtheit	
		zurückgekehrt	abgeschossen
Teile	beschädigt	X	X
	intakt	X	

<https://www.hirnliga.ch/de/artikel-das-gehirn-slider/stolperfallen>

<https://karrierebibel.de/survivorship-bias/>

Survivorship bias



		Grundgesamtheit	
		zurückgekehrt	abgeschossen
Teile	beschädigt	x	x
	intakt	x	

<https://www.hirnliga.ch/de/artikel-das-gehirn-slider/stolperfallen>

<https://karrierebibel.de/survivorship-bias/>

Kognitive Verzerrung bei der sichtbare Informationen überbewertet werden und/oder weniger sichtbare vernachlässigt oder übersehen werden

Zusammenfassung

1. verschiedene Probleme in der Welt können abstrakt mit ähnlichen Strukturen beschrieben werden:
Zuordnung, Wege, ...
2. Abstraktion eines der wichtigsten Werkzeuge in der Informatik
3. Königsberger Brückenproblem: Graphentheoretische Konzepte: Weg, Pfad, Zyklus, Kreis, ...
4. Eulerweg & Eulerkreis: Algorithmus von Hierholzer
5. Problem bei Abstraktion: Validität
6. Survivorship Bias



Abstraktion in der Informatik

1. Funktionen sind wichtige Abstraktionstechnik in Programmen.

Abstraktion in der Informatik

1. Funktionen sind wichtige Abstraktionstechnik in Programmen.
2. Modellierung ist eine zentrale Abstraktionstechnik und unabdingbar für die Abbildung der realen Welt auf den Computer / die Software.

Abstraktion in der Informatik

1. Funktionen sind wichtige Abstraktionstechnik in Programmen.
2. Modellierung ist eine zentrale Abstraktionstechnik und unabdingbar für die Abbildung der realen Welt auf den Computer / die Software.
3. Rechnen (Computing) ist nichts anderes als das Konstruieren, Manipulieren (Verarbeiten) und Folgern auf Basis von Abstraktionen.

Abstraktion in der Informatik

1. Funktionen sind wichtige Abstraktionstechnik in Programmen.
2. Modellierung ist eine zentrale Abstraktionstechnik und unabdingbar für die Abbildung der realen Welt auf den Computer / die Software.
3. Rechnen (Computing) ist nichts anderes als das Konstruieren, Manipulieren (Verarbeiten) und Folgern auf Basis von Abstraktionen.
4. Mit Abstraktion lassen sich komplexe System (besser) beherrschen, z.B. komplexe Softwaresysteme.

Abstraktion in der Informatik

1. Funktionen sind wichtige Abstraktionstechnik in Programmen.
2. Modellierung ist eine zentrale Abstraktionstechnik und unabdingbar für die Abbildung der realen Welt auf den Computer / die Software.
3. Rechnen (Computing) ist nichts anderes als das Konstruieren, Manipulieren (Verarbeiten) und Folgern auf Basis von Abstraktionen.
4. Mit Abstraktion lassen sich komplexe System (besser) beherrschen, z.B. komplexe Softwaresysteme.
5. Durch Abstrahieren kann man Lösungsansätze verallgemeinern und Softwarelösungen mehrfach verwenden.

Abstraktion in der Informatik

1. Funktionen sind wichtige Abstraktionstechnik in Programmen.
2. Modellierung ist eine zentrale Abstraktionstechnik und unabdingbar für die Abbildung der realen Welt auf den Computer / die Software.
3. Rechnen (Computing) ist nichts anderes als das Konstruieren, Manipulieren (Verarbeiten) und Folgern auf Basis von Abstraktionen.
4. Mit Abstraktion lassen sich komplexe System (besser) beherrschen, z.B. komplexe Softwaresysteme.
5. Durch Abstrahieren kann man Lösungsansätze verallgemeinern und Softwarelösungen mehrfach verwenden.
6. Abstrahieren schärft den Blick aufs Wesentliche. Damit werden Lösungsstrategien sichtbar.

Weitere Beispiele für Problemlösen: “einen Fuß in die Tür kriegen”

A, B, C und D laufen ein Rennen. Vorher haben Sie folgende Vorhersagen gemacht:

- A hat vorhergesagt, dass B gewinnt
- B hat vorhergesagt, dass D Letzte wird
- C hat vorhergesagt, dass A Dritte wird
- D hat gesagt, dass A's Vorhersage richtig ist.

Nur eine dieser Vorhersagen ist richtig und zwar die der Gewinnerin. In welcher Reihenfolge haben A, B, C und D das Rennen beendet?

Weitere Beispiele für Problemlösen: “einen Fuß in die Tür kriegen”

A, B, C und D laufen ein Rennen. Vorher haben Sie folgende Vorhersagen gemacht:

- A hat vorhergesagt, dass B gewinnt
- B hat vorhergesagt, dass D Letzte wird
- C hat vorhergesagt, dass A Dritte wird
- D hat gesagt, dass A's Vorhersage richtig ist.

Nur eine dieser Vorhersagen ist richtig und zwar die der Gewinnerin. In welcher Reihenfolge haben A, B, C und D das Rennen beendet?

- A und D machen dieselbe Vorhersage, aber nur eine Vorhersage ist richtig, d.h. weder A noch D haben gewonnen und ihre Vorhersagen sind falsch

Weitere Beispiele für Problemlösen: “einen Fuß in die Tür kriegen”

A, B, C und D laufen ein Rennen. Vorher haben Sie folgende Vorhersagen gemacht:

- A hat vorhergesagt, dass B gewinnt
- B hat vorhergesagt, dass D Letzte wird
- C hat vorhergesagt, dass A Dritte wird
- D hat gesagt, dass A's Vorhersage richtig ist.

Nur eine dieser Vorhersagen ist richtig und zwar die der Gewinnerin. In welcher Reihenfolge haben A, B, C und D das Rennen beendet?

- A und D machen dieselbe Vorhersage, aber nur eine Vorhersage ist richtig, d.h. weder A noch D haben gewonnen und ihre Vorhersagen sind falsch
- A's Vorhersage, dass B gewinnt, ist falsch → C ist Gewinnerin

Weitere Beispiele für Problemlösen: “einen Fuß in die Tür kriegen”

A, B, C und D laufen ein Rennen. Vorher haben Sie folgende Vorhersagen gemacht:

- A hat vorhergesagt, dass B gewinnt
- B hat vorhergesagt, dass D Letzte wird
- C hat vorhergesagt, dass A Dritte wird
- D hat gesagt, dass A's Vorhersage richtig ist.

Nur eine dieser Vorhersagen ist richtig und zwar die der Gewinnerin. In welcher Reihenfolge haben A, B, C und D das Rennen beendet?

- A und D machen dieselbe Vorhersage, aber nur eine Vorhersage ist richtig, d.h. weder A noch D haben gewonnen und ihre Vorhersagen sind falsch
 - A's Vorhersage, dass B gewinnt, ist falsch → C ist Gewinnerin
- A ist Dritte (laut C's Vorhersage)

Weitere Beispiele für Problemlösen: “einen Fuß in die Tür kriegen”

A, B, C und D laufen ein Rennen. Vorher haben Sie folgende Vorhersagen gemacht:

- A hat vorhergesagt, dass B gewinnt
- B hat vorhergesagt, dass D Letzte wird
- C hat vorhergesagt, dass A Dritte wird
- D hat gesagt, dass A's Vorhersage richtig ist.

Nur eine dieser Vorhersagen ist richtig und zwar die der Gewinnerin. In welcher Reihenfolge haben A, B, C und D das Rennen beendet?

- A und D machen dieselbe Vorhersage, aber nur eine Vorhersage ist richtig, d.h. weder A noch D haben gewonnen und ihre Vorhersagen sind falsch
- A's Vorhersage, dass B gewinnt, ist falsch → C ist Gewinnerin
 - A ist Dritte (laut C's Vorhersage)
 - C B A D oder C D A B?

Weitere Beispiele für Problemlösen: “einen Fuß in die Tür kriegen”

A, B, C und D laufen ein Rennen. Vorher haben Sie folgende Vorhersagen gemacht:

- A hat vorhergesagt, dass B gewinnt
- B hat vorhergesagt, dass D Letzte wird
- C hat vorhergesagt, dass A Dritte wird
- D hat gesagt, dass A's Vorhersage richtig ist.

Nur eine dieser Vorhersagen ist richtig und zwar die der Gewinnerin. In welcher Reihenfolge haben A, B, C und D das Rennen beendet?

- A und D machen dieselbe Vorhersage, aber nur eine Vorhersage ist richtig, d.h. weder A noch D haben gewonnen und ihre Vorhersagen sind falsch
- A's Vorhersage, dass B gewinnt, ist falsch → C ist Gewinnerin
 - A ist Dritte (laut C's Vorhersage)
 - C B A D oder C D A B? → **C D A B**

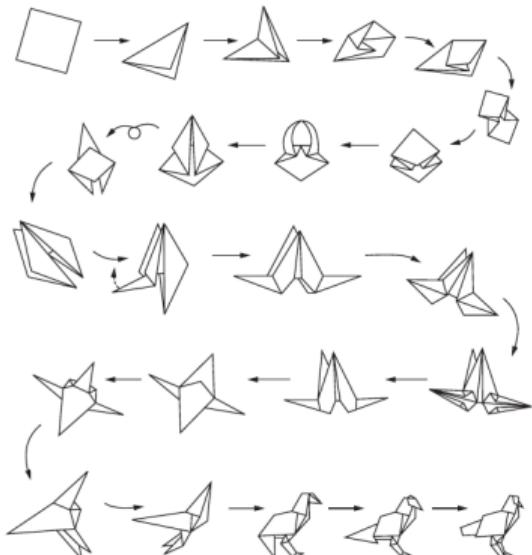
Problemlöse-Strategien

- ein Problem zu lösen, auch nur Ansatz zu finden, erfordert Erfahrung/Übung, Geduld, Vertrauen in die eigenen Fähigkeiten → Kreativität
- es gibt ein paar allgemeine Strategien

Problemlöse-Strategien

- ein Problem zu lösen, auch nur Ansatz zu finden, erfordert Erfahrung/Übung, Geduld, Vertrauen in die eigenen Fähigkeiten → Kreativität
- es gibt ein paar allgemeine Strategien

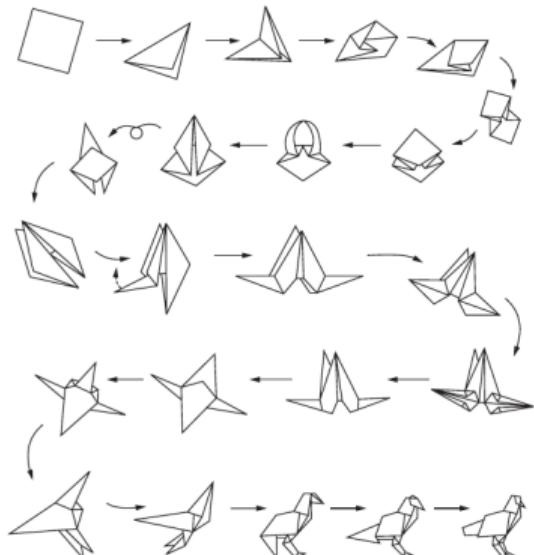
1. von hinten anfangen z.B. aus gegebenem Input soll bestimmter Output erzeugt werden, eventuell beim Output anfangen und sich zum Input zurückarbeiten



Problemlöse-Strategien

- ein Problem zu lösen, auch nur Ansatz zu finden, erfordert Erfahrung/Übung, Geduld, Vertrauen in die eigenen Fähigkeiten → Kreativität
- es gibt ein paar allgemeine Strategien

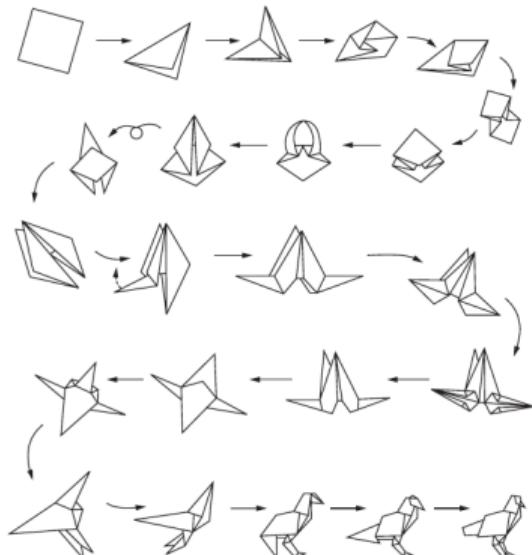
1. von hinten anfangen z.B. aus gegebenem Input soll bestimmter Output erzeugt werden, eventuell beim Output anfangen und sich zum Input zurückarbeiten
2. verwandtes Problem finden, das einfacher ist und auf den Kontext anwenden



Problemlöse-Strategien

- ein Problem zu lösen, auch nur Ansatz zu finden, erfordert Erfahrung/Übung, Geduld, Vertrauen in die eigenen Fähigkeiten → Kreativität
- es gibt ein paar allgemeine Strategien

1. von hinten anfangen z.B. aus gegebenem Input soll bestimmter Output erzeugt werden, eventuell beim Output anfangen und sich zum Input zurückarbeiten
2. verwandtes Problem finden, das einfacher ist und auf den Kontext anwenden
3. Zerlegen in Teilprobleme
! Entscheiden, welche Strategie passt

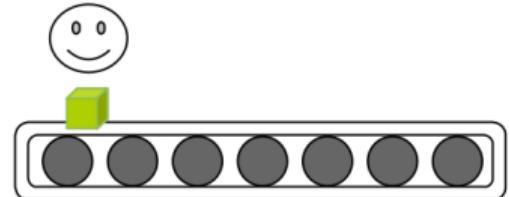


Beispiel: Perspektive

Ein Fluss fließt mit einer Geschwindigkeit von 24km pro Stunde relativ zum Ufer. Eine Rudermannschaft übt das Rudern und rudert zunächst flussaufwärts (gegen den Strom). Bei dieser Geschwindigkeit können sie relativ zum Ufer nur 10km pro Stunde zurücklegen. Der Typ am hinteren Ende des Bootes trägt zu Beginn einen Hut, aber nach einer Weile fällt sein Hut ins Wasser (und schwimmt flussabwärts). Es dauert 15 Minuten bis sie es bemerken. Sie kehren augenblicklich um und rudern zurück, um den Hut einzuholen, wobei sie mit der gleichen Kraft rudern wie zuvor. Wie lange werden sie brauchen, um den Hut einzuholen, der von der Strömung flussabwärts getrieben wird?

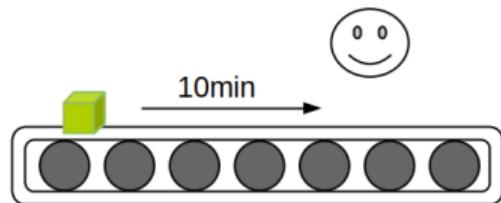
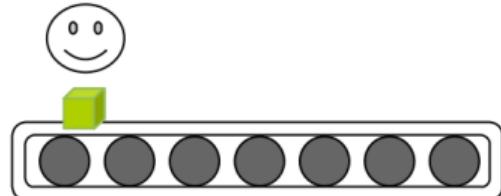
Beispiel: Perspektive

Ein Fluss fließt mit einer Geschwindigkeit von 24km pro Stunde relativ zum Ufer. Eine Rudermannschaft übt das Rudern und rudert zunächst flussaufwärts (gegen den Strom). Bei dieser Geschwindigkeit können sie relativ zum Ufer nur 10km pro Stunde zurücklegen. Der Typ am hinteren Ende des Bootes trägt zu Beginn einen Hut, aber nach einer Weile fällt sein Hut ins Wasser (und schwimmt flussabwärts). Es dauert 15 Minuten bis sie es bemerken. Sie kehren augenblicklich um und rudern zurück, um den Hut einzuholen, wobei sie mit der gleichen Kraft rudern wie zuvor. Wie lange werden sie brauchen, um den Hut einzuholen, der von der Strömung flussabwärts getrieben wird?



Beispiel: Perspektive

Ein Fluss fließt mit einer Geschwindigkeit von 24km pro Stunde relativ zum Ufer. Eine Rudermannschaft übt das Rudern und rudert zunächst flussaufwärts (gegen den Strom). Bei dieser Geschwindigkeit können sie relativ zum Ufer nur 10km pro Stunde zurücklegen. Der Typ am hinteren Ende des Bootes trägt zu Beginn einen Hut, aber nach einer Weile fällt sein Hut ins Wasser (und schwimmt flussabwärts). Es dauert 15 Minuten bis sie es bemerken. Sie kehren augenblicklich um und rudern zurück, um den Hut einzuholen, wobei sie mit der gleichen Kraft rudern wie zuvor. Wie lange werden sie brauchen, um den Hut einzuholen, der von der Strömung flussabwärts getrieben wird?



Vom Speziellen zum Allgemeinen

Betrachte folgende Stromtarife. Beide Tarife bestehen aus einer monatl. Grundgebühr und einem Teil, der sich nach dem Verbrauch in Kilowattstunden (kWh) richtet.

	Grundgebühr pro Monat	Verbrauch pro Einheit (kWh)
Tarif "Billig-Strom"	4,90EUR	0,19EUR
Tarif "Watt für wenig"	8,20EUR	0,16EUR

1. Schreibe ein Programm, das den Monatsverbrauch in kWh akzeptiert und den im Tarif "Billig-Strom" zu zahlenden monatlichen Rechnungsbetrag berechnet.
2. Schreibe ein Programm, das den Monatsverbrauch in kWh akzeptiert und den im Tarif "Watt für wenig" zu zahlenden monatlichen Rechnungsbetrag berechnet.

Vom Speziellen zum Allgemeinen

Betrachte folgende Stromtarife. Beide Tarife bestehen aus einer monatl. Grundgebühr und einem Teil, der sich nach dem Verbrauch in Kilowattstunden (kWh) richtet.

	Grundgebühr pro Monat	Verbrauch pro Einheit (kWh)
Tarif "Billig-Strom"	4,90EUR	0,19EUR
Tarif "Watt für wenig"	8,20EUR	0,16EUR

1. Schreibe ein Programm, das den Monatsverbrauch in kWh akzeptiert und den im Tarif "Billig-Strom" zu zahlenden monatlichen Rechnungsbetrag berechnet.
2. Schreibe ein Programm, das den Monatsverbrauch in kWh akzeptiert und den im Tarif "Watt für wenig" zu zahlenden monatlichen Rechnungsbetrag berechnet.

```
define billig_strom(verbrauch):  
    return 4.9 + .19* verbrauch
```

```
define watt_für_wenig(verbrauch):  
    return 8.2 + .16* verbrauch
```

Vom Speziellen zum Allgemeinen

Betrachte folgende Stromtarife. Beide Tarife bestehen aus einer monatl. Grundgebühr und einem Teil, der sich nach dem Verbrauch in Kilowattstunden (kWh) richtet.

	Grundgebühr pro Monat	Verbrauch pro Einheit (kWh)
Tarif "Billig-Strom"	4,90EUR	0,19EUR
Tarif "Watt für wenig"	8,20EUR	0,16EUR

1. Schreibe ein Programm, das den Monatsverbrauch in kWh akzeptiert und den im Tarif "Billig-Strom" zu zahlenden monatlichen Rechnungsbetrag berechnet.
2. Schreibe ein Programm, das den Monatsverbrauch in kWh akzeptiert und den im Tarif "Watt für wenig" zu zahlenden monatlichen Rechnungsbetrag berechnet.

```
define alle_tarife(grundgebühr, pro_einheit, verbrauch):  
    return grundgebühr + pro_einheit * verbrauch
```