Computational complexity 101

The basics, $\mathcal{P}$ and $\mathcal{NP}$

Julian Lorenz

December 1, 2020

Goethe University

Table of contents

Motivation

Motivation

Alles beginnt mit einem Problem...

- Fokus auf *Klassifikations-* / *Entscheidungsproblemen*

Motivation

Alles beginnt mit einem Problem...

- Fokus auf *Klassifikations-* / *Entscheidungsproblemen*
- Zeit gemessen an der Anzahl elementarer Operationen

Motivation

Alles beginnt mit einem Problem...

- Fokus auf *Klassifikations-* / *Entscheidungsproblemen*
- Zeit gemessen an der Anzahl elementarer Operationen

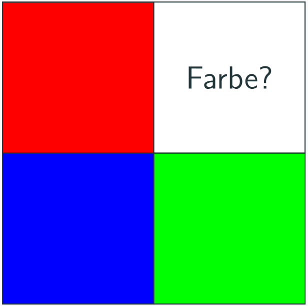
2 Probleme:

(1) Welche planaren Landkarten sind 3-färbbar?

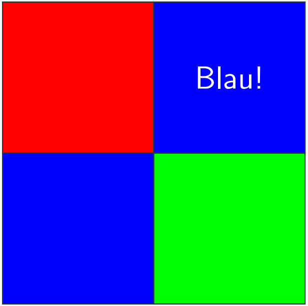
(2) Welche *Diophantische Gleichungen* der Form $Ax^2 + By + C = 0$ können durch positive Ganzzahlen gelöst werden?

- Gleichung mit ganzzahligen Koeffizienten und Lösungen

(1) 3-Färbbarkeit

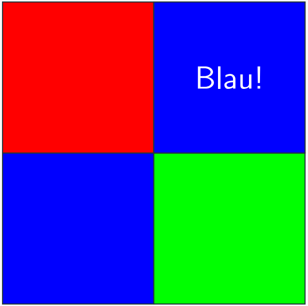


(1) 3-Färbbarkeit

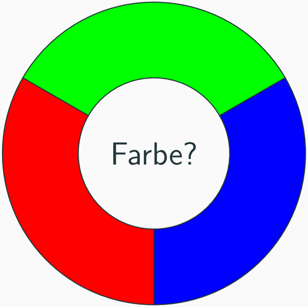


3-färbbar

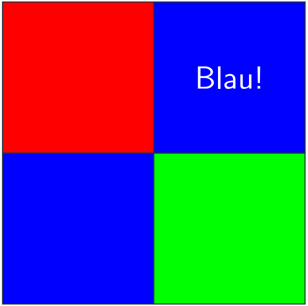
(1) 3-Färbbarkeit



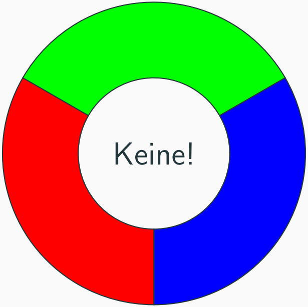
3-färbbar



(1) 3-Färbbarkeit



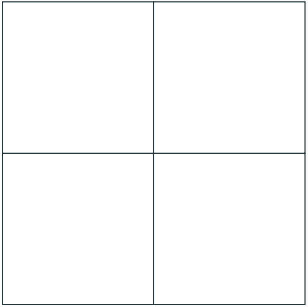
3-färbbar

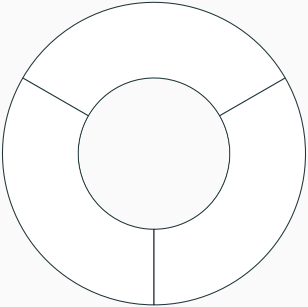


Nicht 3-färbbar

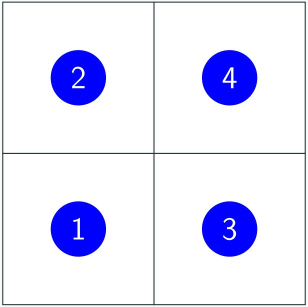
Darstellung

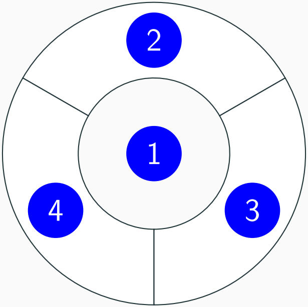
Darstellung



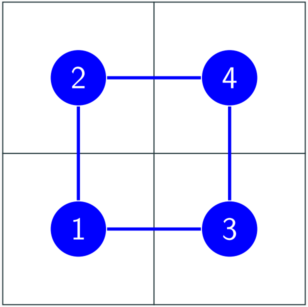


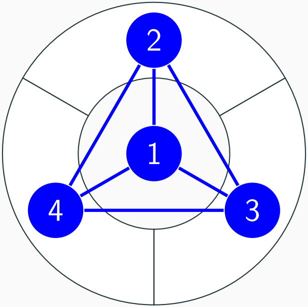
Darstellung



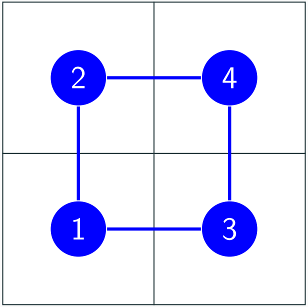


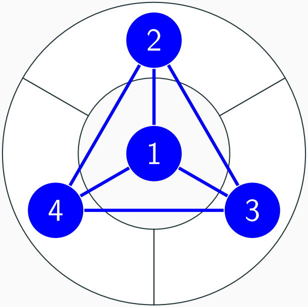
Darstellung





Darstellung





$$
\begin{aligned}
G_{\blacksquare} &= \{1 : [2,3], 2 : [1,4], 3 : [1,4], 4 : [2,3]\} \\
&= (V = (1,2,3,4), E = ((1,2),(1,3),(2,4),(3,4))) \\
&= \cdots
\end{aligned}
$$

Binäre Sequenzen

- Jedes endliche Objekt kann durch eine binäre Sequenz beschrieben werden

$\Rightarrow$ Beschreibung als Eingabe für Algorithmen

Binäre Sequenzen

- Jedes endliche Objekt kann durch eine binäre Sequenz beschrieben werden

$\Rightarrow$ Beschreibung als Eingabe für Algorithmen

Definition

Sei $\mathbb{I}$ die Menge aller binären Sequenzen über dem Alphabet $\{0, 1\}$ sowie $\mathbb{I}_n = \{0, 1\}^n$.
$\mathbb{I}$ kann als Menge der Eingaben aller Klassifikationsprobleme angesehen werden, wobei jede Teilmenge von $\mathbb{I}$ ein Klassifikationsproblem beschreibt.

Reduktion

Theorem

Probleme (1) *und* (2) *sind äquivalent.*

Reduktion

Theorem

Probleme (1) *und* (2) *sind äquivalent.*

- Berechenbare Funktionen $f, h : \mathbb{I} \to \mathbb{I}$:

$$
\begin{aligned}
(V, E) \in \mathbf{(1)} \quad &\Leftrightarrow \quad f(V, E) \in \mathbf{(2)} \quad \text{und} \\
(A, B, C) \in \mathbf{(2)} \quad &\Leftrightarrow \quad h(A, B, C) \in \mathbf{(1)}
\end{aligned}
$$

Reduktion

Theorem

Probleme (1) *und* (2) *sind äquivalent.*

- Berechenbare Funktionen $f, h : \mathbb{I} \to \mathbb{I}$:
$$\begin{aligned} (V, E) \in \mathbf{(1)} \quad &\Leftrightarrow \quad f(V, E) \in \mathbf{(2)} \quad \text{und} \\ (A, B, C) \in \mathbf{(2)} \quad &\Leftrightarrow \quad h(A, B, C) \in \mathbf{(1)} \end{aligned}$$
- f, h werden als **Reduktionen** bezeichnet
- Effizient berechenbar

Effiziente Berechnung und die Klasse $\mathcal{P}$

Effizient & Worst-case

- Grundgedanke der Industrie und Wirtschaft

Effizient & Worst-case

- Grundgedanke der Industrie und Wirtschaft
- Asymptotisches Verhalten als Funktion der Eingabelänge

(I) **Effizient:** Laufzeit bei Eingabelänge n ist durch eine *polynomielle* Funktion in n beschränkt

Effizient & Worst-case

- Grundgedanke der Industrie und Wirtschaft
- Asymptotisches Verhalten als Funktion der Eingabelänge

(I) **Effizient:** Laufzeit bei Eingabelänge n ist durch eine *polynomielle* Funktion in n beschränkt

(II) **Worst-case** Bedingung

Warum polynomiell?

- Abgeschlossen unter Addition, Multiplikation und Komposition
- Programme in Sequenz oder verschachtelt

Warum polynomiell?

- Abgeschlossen unter Addition, Multiplikation und Komposition
- Programme in Sequenz oder verschachtelt
- In Kontrast zu *exponentieller* Zeit

Warum polynomiell?

- Abgeschlossen unter Addition, Multiplikation und Komposition
- Programme in Sequenz oder verschachtelt
- In Kontrast zu *exponentieller* Zeit
- **AKS sorting Network** (1983)
- Benötigt $C \cdot \log(n)$ Schritte um n Schlüssel zu sortieren

Warum polynomiell?

- Abgeschlossen unter Addition, Multiplikation und Komposition
- Programme in Sequenz oder verschachtelt
- In Kontrast zu *exponentieller* Zeit

- **AKS sorting Network** (1983)
- Benötigt $C \cdot \log(n)$ Schritte um n Schlüssel zu sortieren
- C so groß, dass mergesort bis $\approx 1.2 \cdot 10^{52}$ besser ist

$\Rightarrow$ AKS somit in der Praxis bisher unbrauchbar

Warum Worst-case?

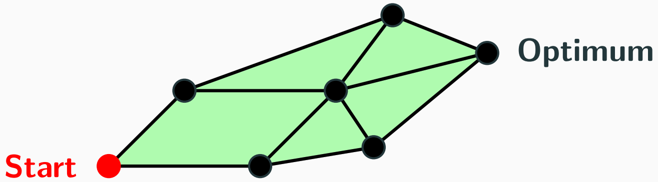
- Keine Sorgen um die Eingabe machen müssen
- Für alle Instanzen $\approx$ stärkere Aussage

Warum Worst-case?

- Keine Sorgen um die Eingabe machen müssen
- Für alle Instanzen $\approx$ stärkere Aussage
- Unbekannter Gegner generiert die Eingabe
- **Beispiel:** Optimale Strategie in *Nullsummenspielen*

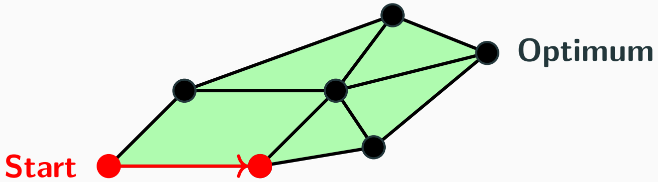
Der Simplex Algorithmus

- **Simplex Algorithmus**



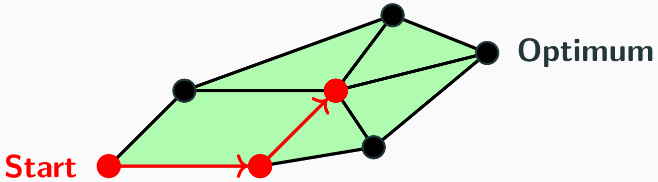
Der Simplex Algorithmus

- **Simplex Algorithmus**
- Klassische Methode zum Lösen Linearer Programme

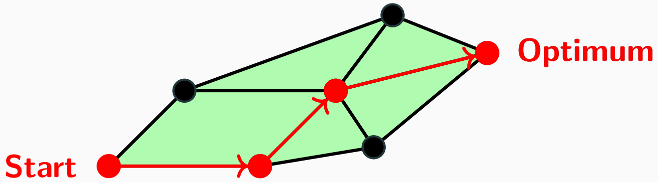


Der Simplex Algorithmus

- **Simplex Algorithmus**
- Klassische Methode zum Lösen Linearer Programme
- Effizient in der Praxis



- **Simplex Algorithmus**
- Klassische Methode zum Lösen Linearer Programme
- Effizient in der Praxis
- Klee-Minty Würfel zeigte 1973 exponentielle Laufzeit



Die Klasse $\mathcal{P}$

Definition (Die Klasse $\mathcal{P}$)

Eine Funktion $f : \mathbb{I} \to \mathbb{I}$ ist in der Klasse $\mathcal{P}$, falls ein Algorithmus, der f berechnet, und positive Konstanten A, c existieren, so dass für jedes n und jedes $x \in \mathbb{I}_n$ der Algorithmus, der $f(x)$ berechnet, maximal An^c elementare Operationen benötigt.

Beispiele: Lineare Programmierung, Planarität, ...

Effiziente Verifikation und die Klasse $\mathcal{NP}$

Effiziente Verifikation

- Sei $\mathcal{C} \subset \mathbb{I}$ ein Klassifikationsproblem

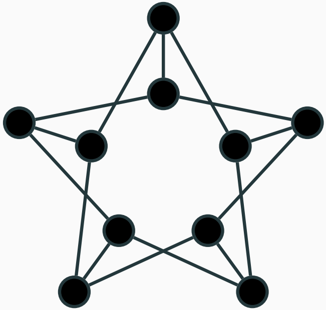
Effiziente Verifikation

- Sei $\mathcal{C} \subset \mathbb{I}$ ein Klassifikationsproblem
- Effizienter Algorithmus um x auf Eigenschaft $\mathcal{C}$ zu testen
- Halte für die Eingabe $x \in \mathbb{I}$, falls $x \in \mathcal{C}$

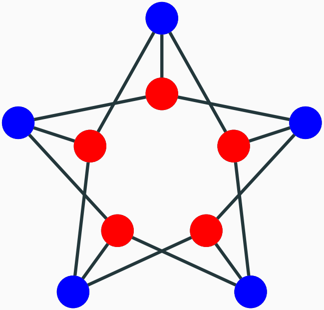
Effiziente Verifikation

- Sei $\mathcal{C} \subset \mathbb{I}$ ein Klassifikationsproblem
- Effizienter Algorithmus um x auf Eigenschaft $\mathcal{C}$ zu testen
- Halte für die Eingabe $x \in \mathbb{I}$, falls $x \in \mathcal{C}$
- *Orakel* / Lösung raten
- Jede Eingabe zu verifizieren ist exponentiell in n

Probleme in $\mathcal{NP}$

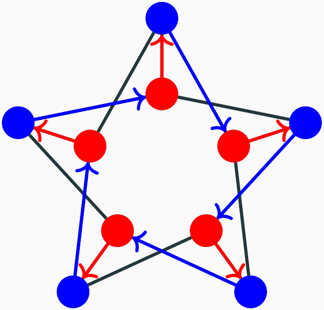


Probleme in $\mathcal{NP}$



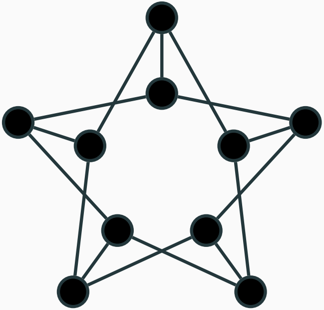
Bipartit

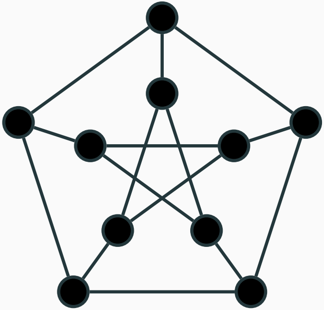
Probleme in $\mathcal{NP}$



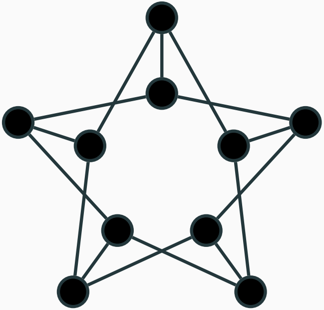
Hamiltion Kreis

Probleme in $\mathcal{NP}$

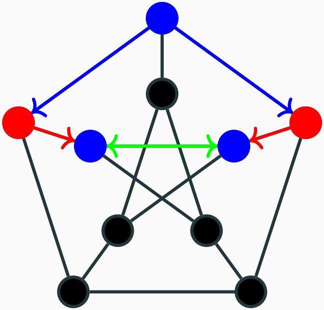




Probleme in $\mathcal{NP}$



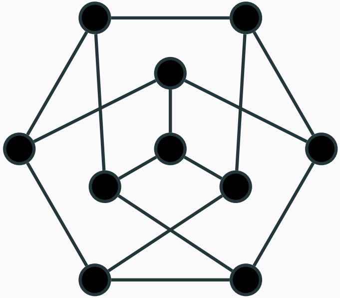
Bipartit

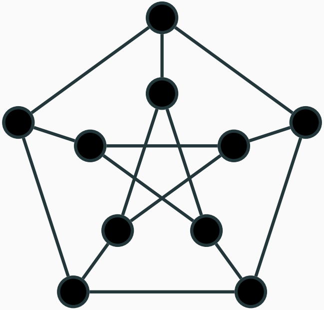


Nicht Bipartit

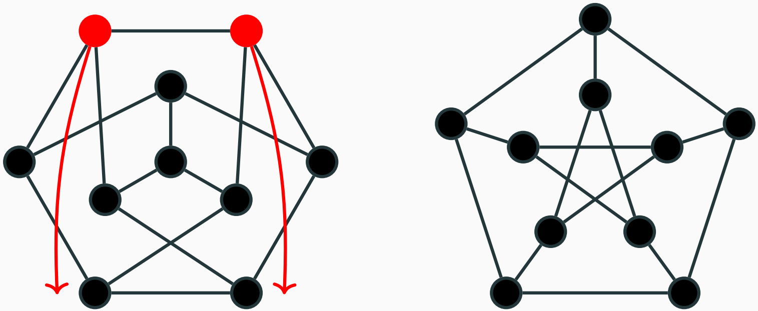
Nicht isomorph!

Probleme in $\mathcal{NP}$

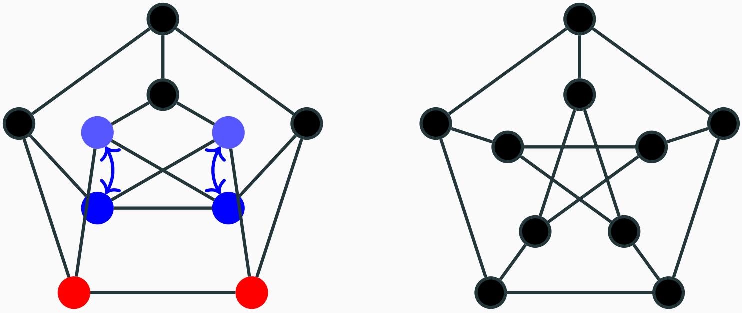




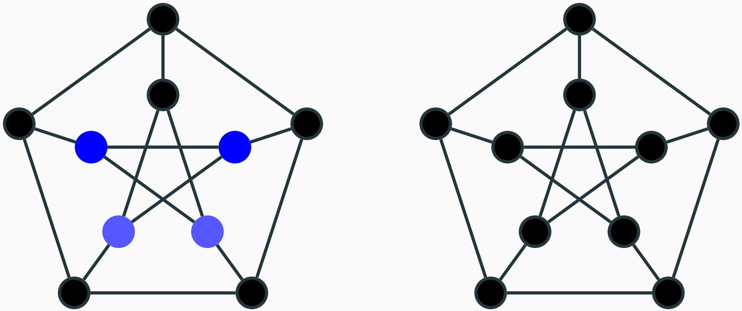
Probleme in $\mathcal{NP}$



Probleme in $\mathcal{NP}$



Probleme in $\mathcal{NP}$



Isomorph!

Die Klasse $\mathcal{NP}$

Definition (Die Klasse $\mathcal{NP}$)

Die Menge $\mathcal{C}$ ist in der Klasse $\mathcal{NP}$, falls eine Funktion $V_\mathcal{C} \in \mathcal{P}$ und eine Konstante k existieren, so dass gilt

- Falls $x \in \mathcal{C}$, dann $\exists y$ mit $|y| \leq k \cdot |x|^k$ und $V_\mathcal{C}(x, y) = 1$
- Falls $x \notin \mathcal{C}$, dann gilt $\forall y \; V_\mathcal{C}(x, y) = 0$

Beispiele: 3-Färbbarkeit, Hamiltionkreis, ...

$\mathcal{P}$ vs $\mathcal{NP}$

$\mathcal{P}$ vs $\mathcal{NP}$

Milleniumproblem

Korollar: $\mathcal{P} \subset \mathcal{NP}$

Offenes Problem: $\mathcal{P} = \mathcal{NP}$?

$\mathcal{P}$ vs $\mathcal{NP}$

Milleniumproblem

Korollar: $\mathcal{P} \subset \mathcal{NP}$

Offenes Problem: $\mathcal{P} = \mathcal{NP}$?

- Ersetze *polynomielle* durch *endliche Zeit* als Schranke:

$\Rightarrow$ $\mathcal{P}$ ist analog zu $\mathcal{R}$ also *Rekursiv* (Entscheidbar)

$\Rightarrow$ $\mathcal{NP}$ ist analog zu $\mathcal{RE}$ also *Rekursiv aufzählbar*

Milleniumproblem

Korollar: $\mathcal{P} \subset \mathcal{NP}$

Offenes Problem: $\mathcal{P} = \mathcal{NP}$?

- Ersetze *polynomielle* durch *endliche Zeit* als Schranke:

$\Rightarrow$ $\mathcal{P}$ ist analog zu $\mathcal{R}$ also *Rekursiv* (Entscheidbar)

$\Rightarrow$ $\mathcal{NP}$ ist analog zu $\mathcal{RE}$ also *Rekursiv aufzählbar*

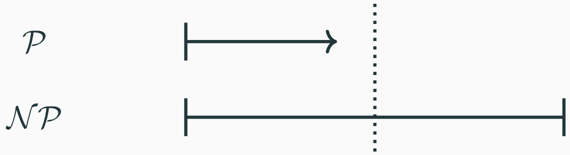
- $\mathcal{R} \neq \mathcal{RE}$ bereits gezeigt
- $\mathcal{NP} \subsetneq R \subsetneq RE$

Aktuelle Vermutung

Annahme: $\mathcal{P} \neq \mathcal{NP}$

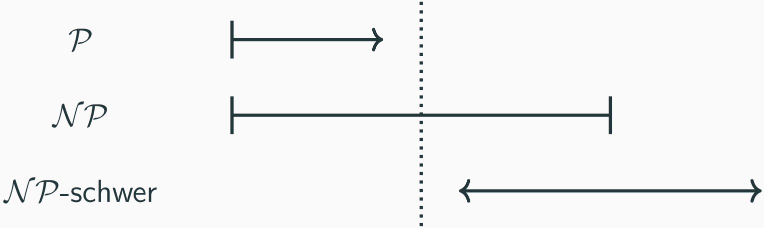
Aktuelle Vermutung

Annahme: $\mathcal{P} \neq \mathcal{NP}$



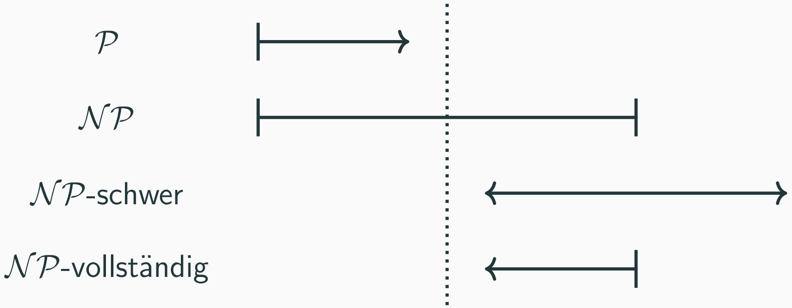
Aktuelle Vermutung

Annahme: $\mathcal{P} \neq \mathcal{NP}$



Aktuelle Vermutung

Annahme: $\mathcal{P} \neq \mathcal{NP}$



Vielen Dank!

Fragen?