Komplexität von Problemen

TECHNISCHE UNIVERSITÄT DARMSTADT

Grundlagen der Informatik II
Sommersemester 2015

Karsten Weihe
Schnitt: Thomas Lüdecke

Asymptotische Komplexität ...

TECHNISCHE UNIVERSITÄT DARMSTADT

- ... von Algorithmen.
- ... von algorithmischen Problemstellungen.

➢ **Wie schnell kann ein Algorithmus für dieses Problem überhaupt sein?**

Untere Schranke Sortieren ...

TECHNISCHE UNIVERSITÄT DARMSTADT

... mit paarweisen Vergleichen:

- **Beste bekannte asymptotische Komplexität:**

$$O(n \log n)$$

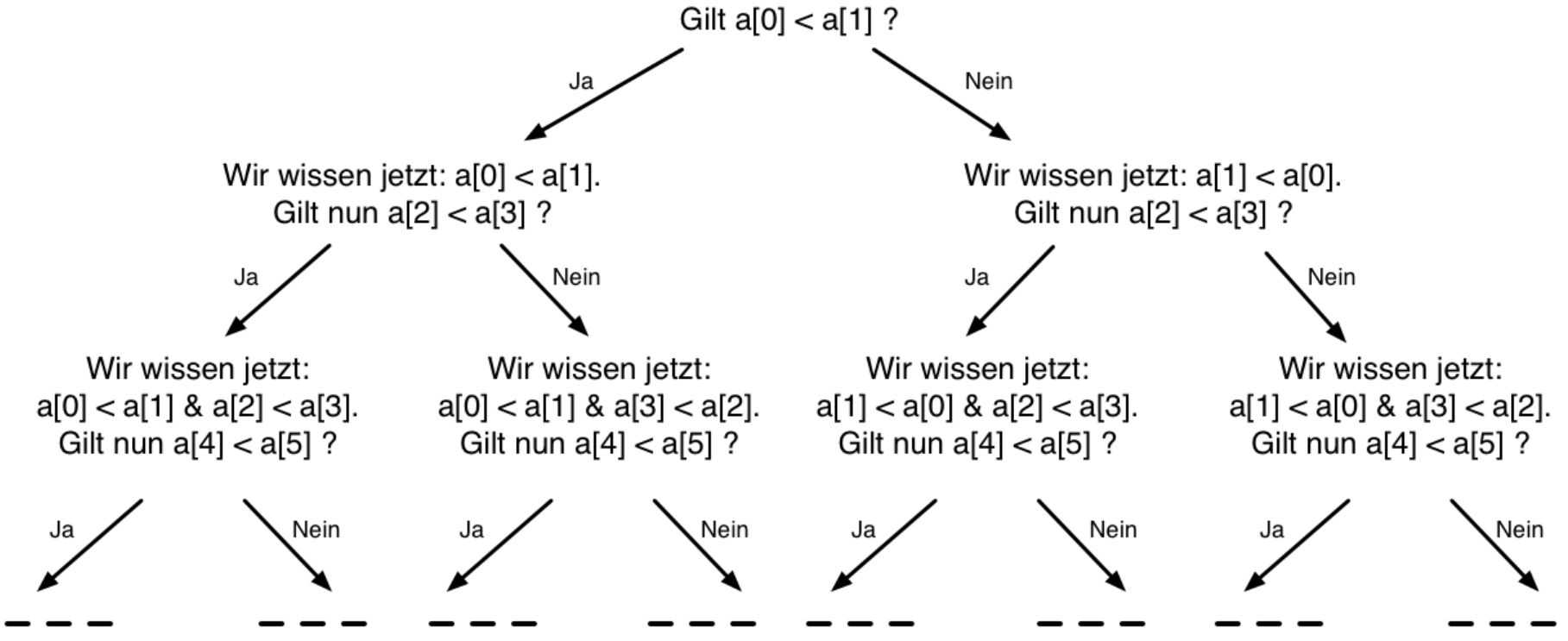
- **Ist auch bestmögliche!**
  - ➢ **Beweisskizze auf den nächsten Folien**

Für untere Schranke reicht, ...

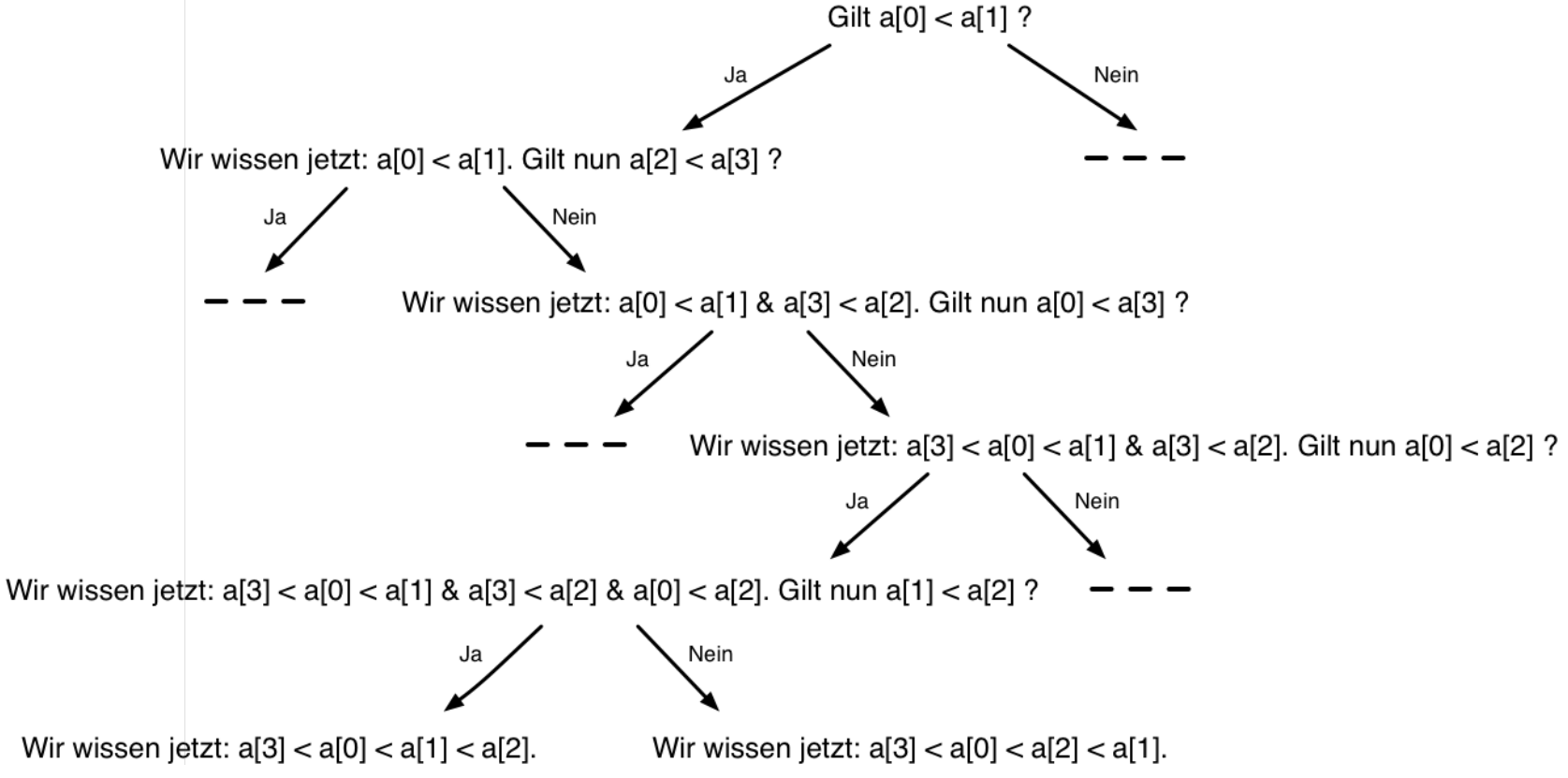
TECHNISCHE UNIVERSITÄT DARMSTADT

- **... die Gesamtzahl paarweiser Vergleiche zu berechnen.**
- **... nur die Permutationen von $\{1,\ldots,n\}$ als Inputs zu betrachten.**
- **... anstelle von Sortieren das algorithmische Problem „identifiziere die Input-Permutation“ zu analysieren.**

Entscheidungsbaum Mergesort n>4



Entscheidungsbaum Mergesort n=4



Permutation identifizieren

TECHNISCHE UNIVERSITÄT DARMSTADT

- In jedem Moment des Algos sind noch gewisse Permutationen möglich.
  - Noch *alle* vor dem ersten Vergleich.
  - Nur noch *eine* nach dem allerletzten Vergleich.
- Bei jedem nichtredundanten Vergleich werden die Permutationen in zwei Mengen zerlegt.

Mathematische Analyse

TECHNISCHE UNIVERSITÄT DARMSTADT

Es gibt $n!$ Permutationen von $\{1,\ldots,n\}$.

- **Maximale Anzahl Vergleiche ist in**

$$\Omega(\log(n!)\ )$$

Mathematische Analyse

TECHNISCHE UNIVERSITÄT DARMSTADT

Es gibt $n!$ Permutationen von $\{1,\ldots,n\}$.

- **Maximale Anzahl Vergleiche ist in**

$$\Omega(\log(n!))$$

Mathematische Analyse

TECHNISCHE UNIVERSITÄT DARMSTADT

$$\log(n!) = \log(1 \cdot 2 \cdot 3 \cdot \cdots \cdot n)$$

$$= \log 1 + \log 2 + \log 3 + \cdots + \log n$$

$$\geq \frac{n}{2} \cdot \log \frac{n}{2}$$

$$= \frac{1}{2} \cdot (n \log n) - n \cdot \text{const} \quad \epsilon \quad \Omega(n \log n)$$

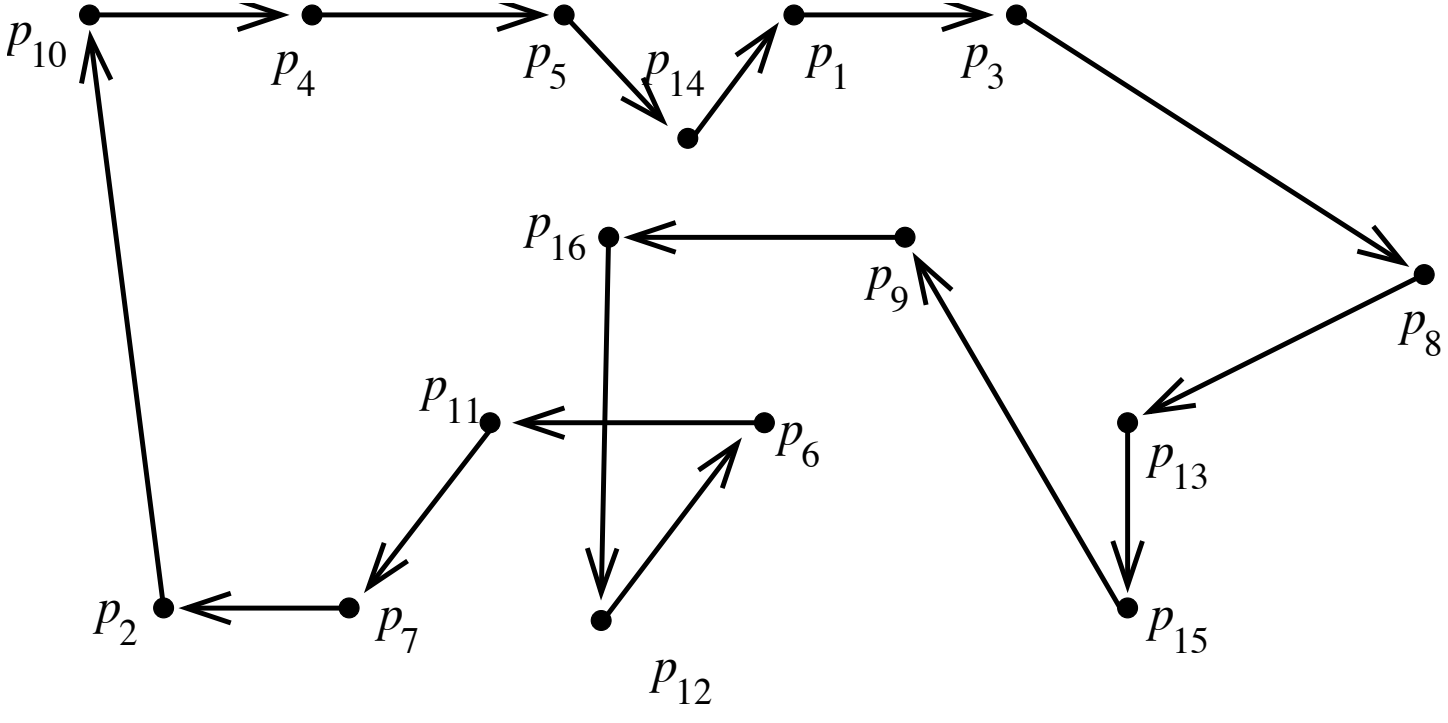
Wirklich schwere Probleme

TECHNISCHE UNIVERSITÄT DARMSTADT

- Handlungsreisendenproblem (TSP)
- Rucksackproblem
- Steinerbaumproblem
- Mengenpackung/-überdeckung
- Clique / Unabhängige Menge
- ...

Handlungsreisendenproblem (TSP)

TECHNISCHE UNIVERSITÄT DARMSTADT



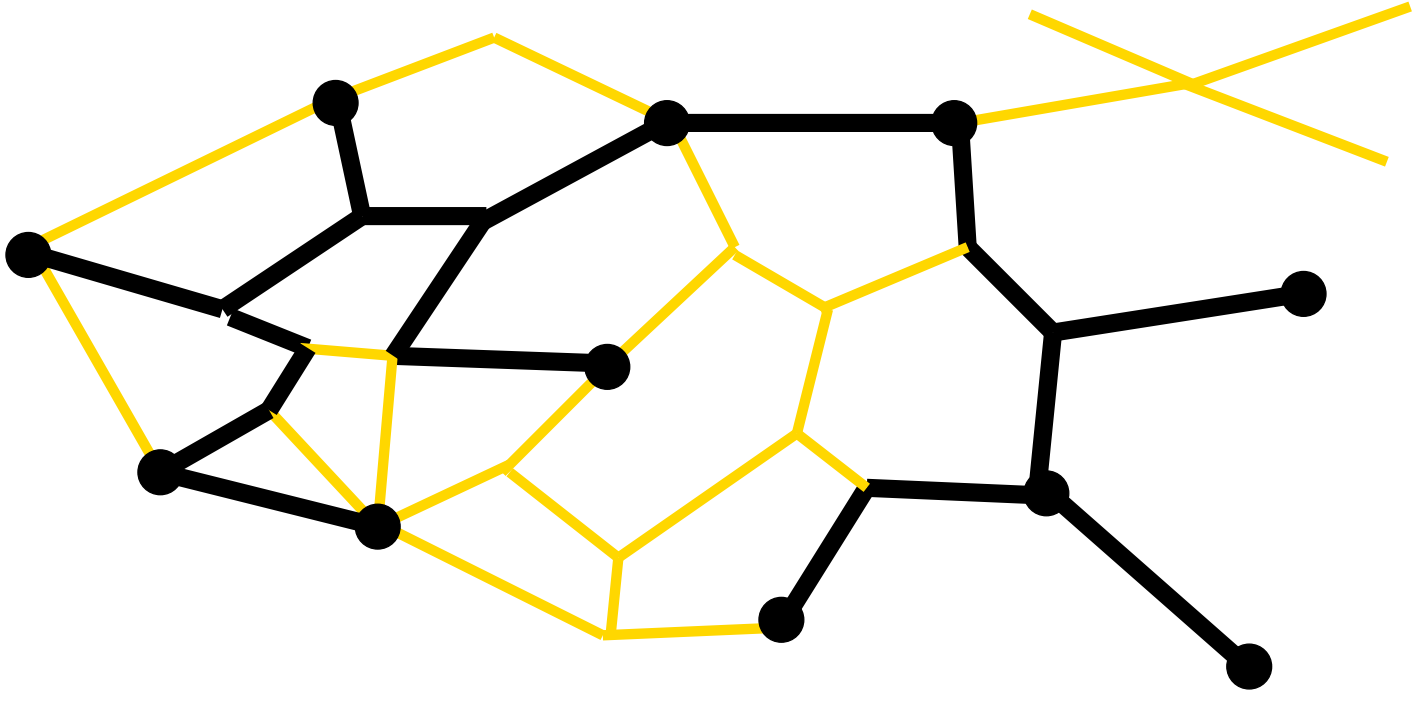
Rucksackproblem

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Gegeben:**
  - **Eine obere Gewichtsschranke *G*.**
  - **Eine Zahl *n* von Elementen.**
  - **Zu jedem Element *i* ein Gewicht g_i und Profit p_i.**
- **Gesucht: eine Auswahl von Elementen, so dass**
  - **das Gesamtgewicht *G* nicht überschritten und**
  - **der Gesamtprofit maximiert wird.**

Steinerbaumproblem

TECHNISCHE UNIVERSITÄT DARMSTADT



Mengenpackung/-überdeckung

TECHNISCHE UNIVERSITÄT DARMSTADT

- *Gegeben*: eine Grundmenge *G* und eine Menge *T* von Teilmengen von *G*.
- *Gesucht*: eine möglichst kleine Auswahl aus *T*, so dass jedes Element von *G* in

genau / mindestens

einer ausgewählten Teilmenge enthalten ist.

Clique / Unabhängige Menge

TECHNISCHE UNIVERSITÄT DARMSTADT

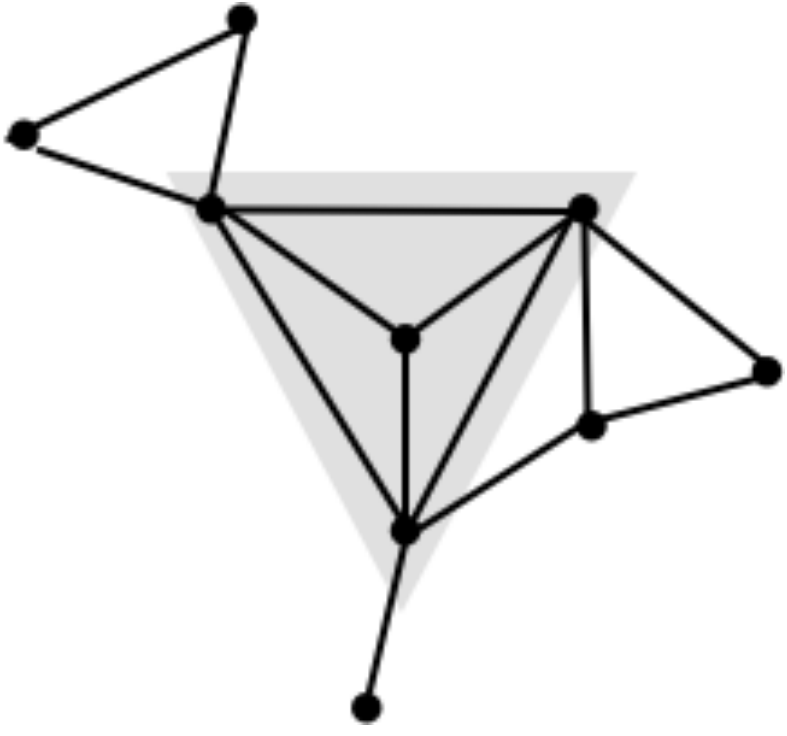
- *Gegeben*: ein ungerichteter Graph.
- *Gesucht*: eine maximale Auswahl von Knoten, so dass alle ausgewählten Knoten paarweise miteinander

  verbunden / unverbunden

  sind.

Maximale Clique

TECHNISCHE UNIVERSITÄT DARMSTADT



Was heißt schwer?

TECHNISCHE UNIVERSITÄT DARMSTADT

Antwort aus der Praxis:

- **Für jeden denkbaren Algo gibt es Beispiele von moderater Größe, auf denen der Algo mehr Laufzeit benötigt als das Universum.**
- **Algorithmische Herausforderung: Algos finden, die trotzdem bestmögliche Ergebnisse in vertretbarer Zeit finden.**

Ansatz aus der Theorie

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Unklar, ob überhaupt ein Algorithmus mit auch nur *irgendwie polynomieller* Komplexität im Worst Case existiert.**
- **Wenigstens das: Wenn ein Output tatsächlich eine zulässige Lösung ist, lässt sich das in jedem dieser Fälle in *polynomieller* Laufzeit verifizieren.**

Was ist polynomielle Laufzeit?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Die asymptotische Laufzeit von Algorithmen wird in einem oder mehreren Inputparametern gemessen.
- Gemeinsamer Nenner: Inputgröße.
  - Zahlen: logarithmische Größe.
  - Jede andere elementare Variable zählt eins.
  - Arrays, Sequenzen, Strings etc. nach ihrer Länge.

Arten von Problemen

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Entscheidungsproblem: Boolescher Output.**
- **Konstruktionsproblem: finde *irgendeine* Lösung.**
- **Aufzählungsproblem: finde *alle* Lösungen.**
- **Optimierungsproblem: finde eine *minimale* / *maximale* Lösung gemäß Zielfunktion.**

Entscheidungsproblem abgeleitet

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Konstruktionsproblem -> Entscheidungsproblem:**
  - **Gibt es überhaupt eine Lösung.**
- **Optimierungsproblem -> Konstruktionsproblem:**
  - **Zusätzlicher Input eine Zahl *k*, suche eine Lösung mit Wert mindestens bzw. höchstens *k***

Das Ausgangsproblem ist mindestens so schwer wie das so abgeleitete Entscheidungsproblem!

Zertifikate

TECHNISCHE UNIVERSITÄT DARMSTADT

Ein Zertifikat für ein Entscheidungsproblem ist

- **eine Datensequenz für jeden Ja-Input sowie**
- **ein Prüfalgorithmus, der mit dem Input und dieser Datensequenz bestätigen kann, dass dies tatsächlich ein Ja-Input ist.**

Zertifikate

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Beispiele:**
  - **Checksumme**
  - **Öffentlicher Schlüssel für Signatur**
- **Für abgeleitete Entscheidungsprobleme ist die Lösung selbst meist das beste Zertifikat.**

Zertifikat beim TSP

TECHNISCHE UNIVERSITÄT DARMSTADT

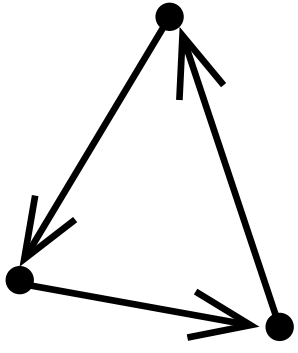
Lösungen kodiert ...

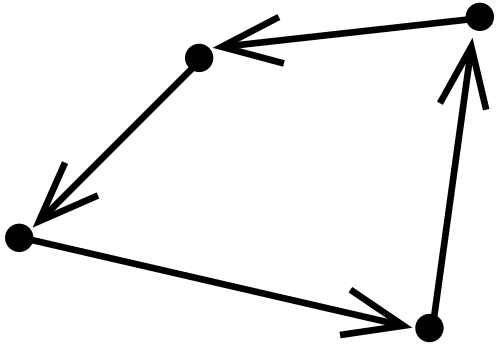
- **... als Permutation:**
  - ➢ **Jeder Punkt muss genau einmal vorkommen.**
- **... als Auswahl von Kanten:**
  - ➢ **Genau eine Kante in jeden / aus jedem Knoten.**
  - ➢ **In genau *n* Schritten zurück zum Start.**

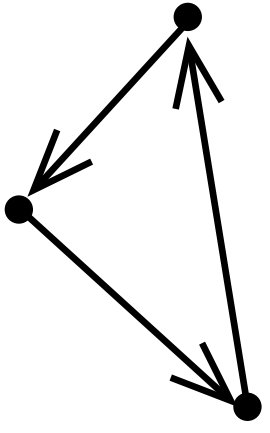
Zertifikat beim TSP

TECHNISCHE UNIVERSITÄT DARMSTADT

Warum in genau *n* Schritten zurück zum Start:







Die Problemklassen $\mathcal{P}$ und $\mathcal{NP}$

TECHNISCHE UNIVERSITÄT DARMSTADT

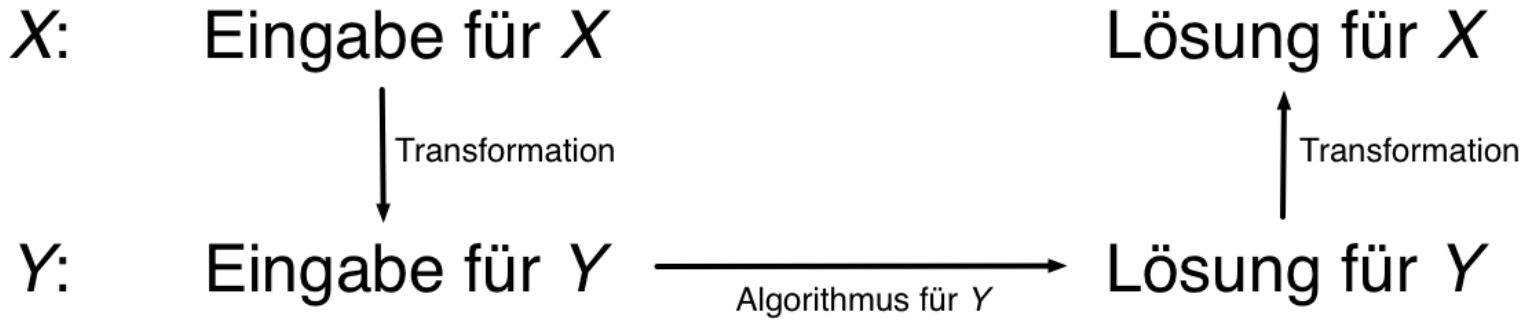
- **Definition:**
  - $\mathcal{P}$: alle polynomiell lösbaren Entscheidungsprobleme.
  - $\mathcal{NP}$: alle Entscheidungsprobleme, für die ein polynomiell überprüfbares Zertifikat existiert.
- **Einfache Folgerung:** $\mathcal{P} \subseteq \mathcal{NP}$.
- **Große ungelöste Frage: Gilt** $\mathcal{P} = \mathcal{NP}$?

Reduktion von Problemen

TECHNISCHE UNIVERSITÄT DARMSTADT

Problem *X* wird auf Problem *Y* *reduziert*.

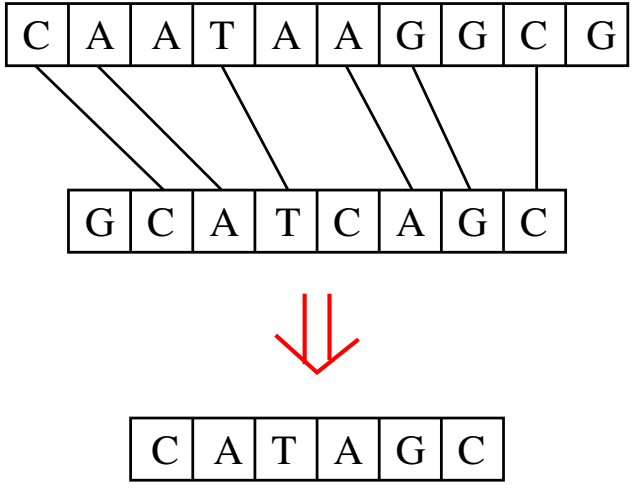
- ***Y* kann nicht einfacher als *X* sein.**

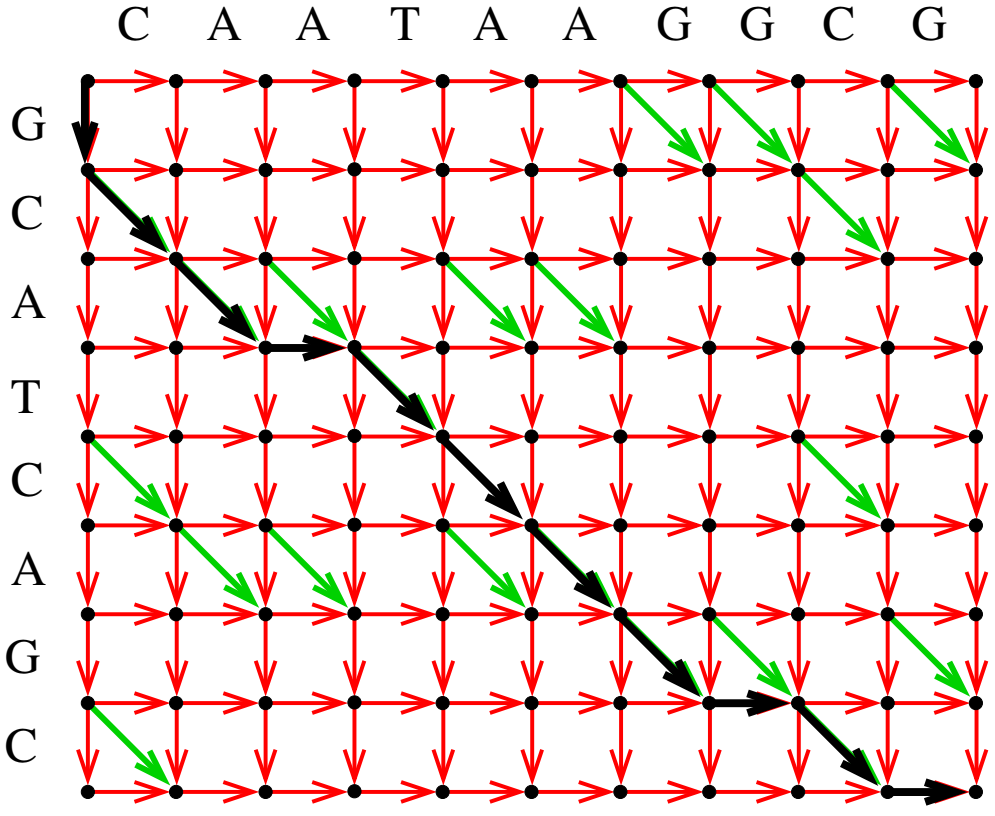


Reduktion auf kürzeste Pfade

TECHNISCHE UNIVERSITÄT DARMSTADT

Beispiel 1: DNA-Sequenzvergleich

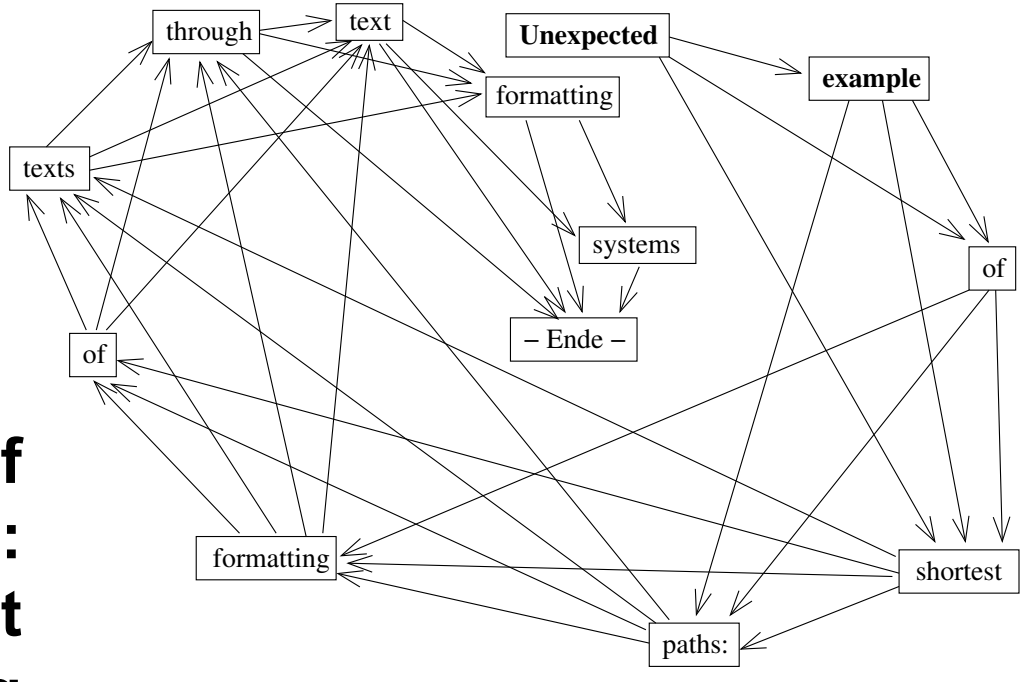




Reduktion auf kürzeste Pfade

TECHNISCHE UNIVERSITÄT DARMSTADT

Beispiel 2: Formatierung von Absätzen



Unexpected example of shortest paths: formatting of text through text formatting systems

Reduktion von Problemen

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Allgemein: Problem *X* wird auf Problem *Y* reduziert durch:**
  - **Einen Algorithmus zur Transformation jedes Inputs *I* für *X* zu einem Input *I'* für *Y*.**
  - **Einen Algorithmus zur Transformation einer Lösung für *I'* zu einer Lösung für *I*.**
- **Bei Entscheidungsproblemen ist die Rücktransformation der Lösung trivial.**

Polynomielle Reduktion

TECHNISCHE UNIVERSITÄT DARMSTADT

- ***X*** **ist auf** ***Y*** ***polynomiell*** **reduzierbar, wenn sowohl die Transformation des Inputs als auch die Rücktransformation des Outputs** ***polynomielle*** **Laufzeit haben.**

- **Bei Entscheidungsproblemen ist die Rück-transformation des Outputs wieder trivial.**

Die Klasse NPC

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Definition: Ein Entscheidungsproblem *X* in $\mathcal{NP}$ ist auch in $\mathcal{NPC}$, wenn alle Probleme in $\mathcal{NP}$ sich auf *X* polynomiell reduzieren lassen.**
- **Konsequenz: $\mathcal{P} = \mathcal{NP}$ gilt genau dann, wenn**

$$\mathcal{P} \cap \mathcal{NPC} \neq \emptyset$$

Circuit-SAT ist in NPC

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Problemstellung *Circuit-Satisfiability*:**
  - **Input: ein Boolescher Schaltkreis mit genau einem Eingangspin.**
  - **Output: ja genau dann, wenn es eine Belegung der Eingangspins mit 0/1-Werten gibt, so dass am Ausgangspin 1 herauskommt.**
- **Beobachtung: circuit-SAT ist in $\mathcal{NP}$.**
- **Behauptung: circuit-SAT ist in $\mathcal{NPC}$.**

Beweisskizze Circuit-SAT in NPC

TECHNISCHE UNIVERSITÄT DARMSTADT

- Sei $X \subseteq \mathcal{NP}$ ein Entscheidungsproblem.
- Für jeden Ja-Input *I* von *X* sei *Z*(*I*) ein Zertifikat.
- Sei *A* ein polynomieller Algorithmus mit *A*(*I*,*Z*(*I*))=1 für jeden Ja-Input *I*.
- Polynomiell heißt: *A* durchläuft nur polynomiell viele Taktzyklen in der Größe von *I*.

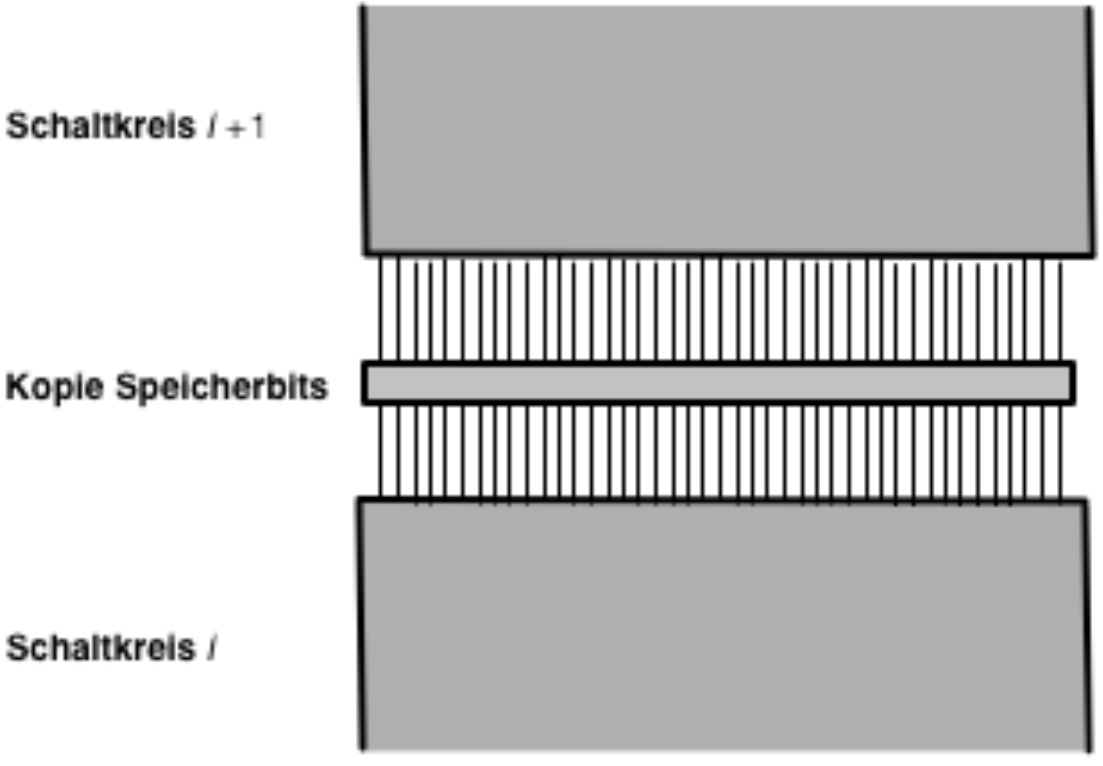
Beweisskizze Circuit-SAT in NPC

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Ein Taktzyklus = ein Schaltkreis.**
  - **OBdA ist der Teil, den A für (I,Z(I)) in einem Taktzyklus durchläuft, polynomial in der Größe von I.**
  - **Lauf von A für (I,Z(I)) = Konkatenation von polynomial vielen Schaltkreisen von jeweils polynomialer Größe.**

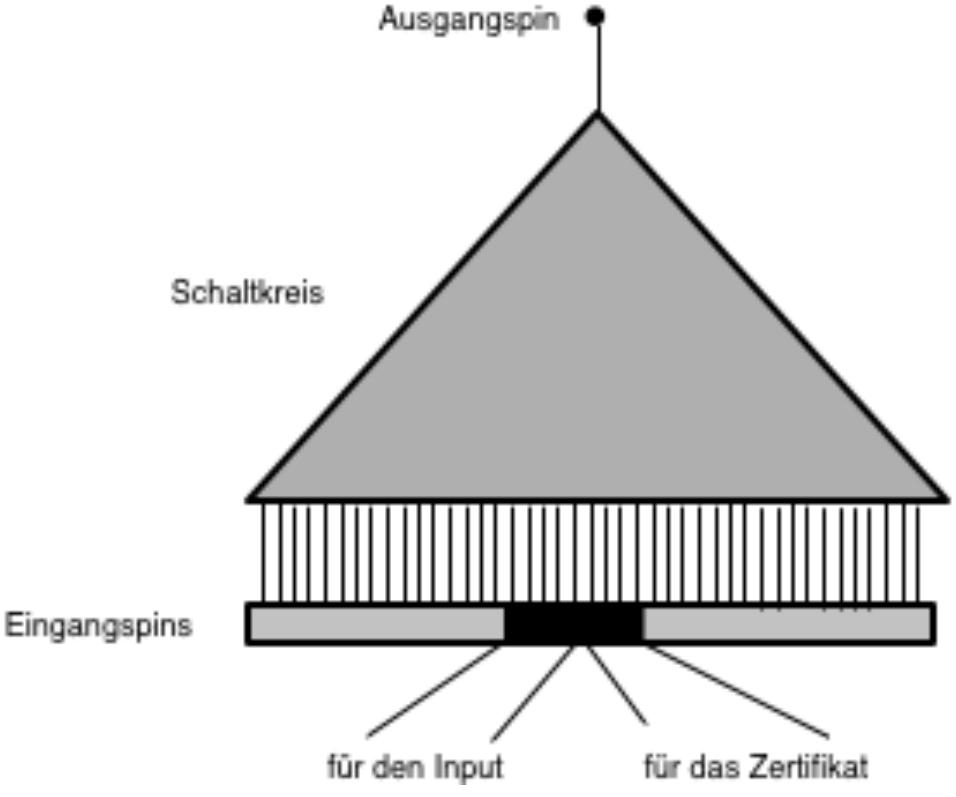
Beweisskizze Circuit-SAT in NPC

TECHNISCHE UNIVERSITÄT DARMSTADT



Beweisskizze Circuit-SAT in NPC

TECHNISCHE UNIVERSITÄT DARMSTADT



Beweisskizze Circuit-SAT in NPC

TECHNISCHE UNIVERSITÄT DARMSTADT

- Falls I ein Ja-Input ist.
  - ➢ Ausgangsbit des Schaltkreises ist 1.
  - ➢ Schaltkreis ist erfüllbar.
- Falls Schaltkreis von I erfüllbar ist.
  - ➢ Die Werte der für das Zertifikat reservierten Eingabepins bilden ein Zertifikat.

Auch SAT ist in NPC

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Problemstellung Satisfiability:**
  - **Input: eine Boolesche Formel.**
  - **Output: ja genau dann, wenn es eine Belegung der Booleschen Variablen gibt, so dass die Formel erfüllt ist.**
- **Beobachtung: SAT ist in NP.**
- **Behauptung: SAT ist in NPC.**

Beweisskizze SAT in NPC

TECHNISCHE UNIVERSITÄT DARMSTADT

- Nimm einen beliebigen Schaltkreis.
- Entwickle 1:1 die Boolesche Formel dieses Schaltkreises.

➢ Also: Alle Probleme in $\mathcal{NP}$ lassen sich polynomiell (transitiv via Circuit-SAT) auf SAT reduzieren.

Auch 3-CNF ist in NPC

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Problemstellung:**
  - **Input: eine Boolesche Formel in CNF mit maximal drei Literalen in jeder Klausel.**
  - **Output: ja genau dann, wenn es eine Belegung der Booleschen Variablen gibt, so dass die Formel erfüllt ist.**
- **Beobachtung: 3-CNF ist in $\mathcal{NP}$.**
- **Behauptung: 3-CNF ist in $\mathcal{NPC}$.**

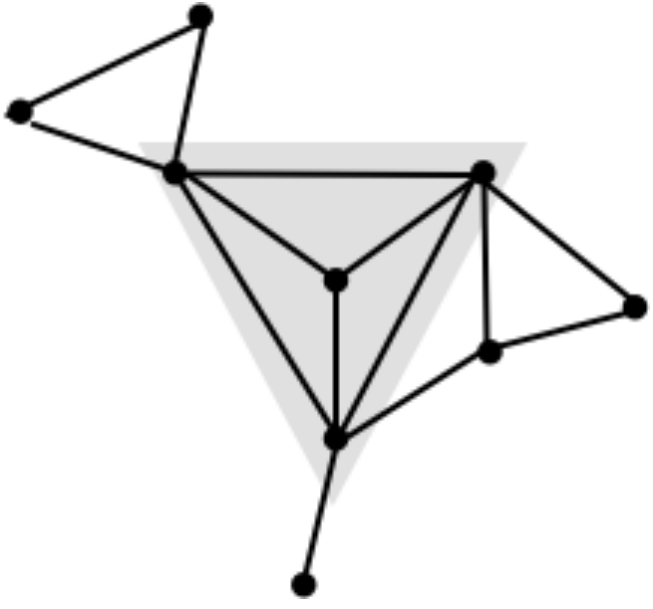
Beweis 3-CNF in NPC

TECHNISCHE UNIVERSITÄT DARMSTADT

- Sei *F* eine beliebige Boolesche Formel.
- Wandle *F* in eine äquivalente Formel *F'* in CNF um ➔ polynomiell.
- Solange es Klausel $K = l_1 \wedge l_2 \wedge \cdots \wedge l_n$ in *F'* mit $n > 3$ Literalen gibt:
  - Führe eine neue Boolesche Variable x ein.
  - Ersetze K durch $K' = l_1 \wedge \cdots \wedge l_{n-2} \wedge x$ und $K'' = l_{n-1} \wedge l_n \wedge \bar{x}.$

Auch CLIQUE ist in NPC

TECHNISCHE UNIVERSITÄT DARMSTADT



Auch CLIQUE ist in NPC

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Entscheidungsproblem zu MAX-CLIQUE:**
  - **Input: ungerichteter Graph $G=(V,E)$, natürliche Zahl k.**
  - **Ja genau dann, wenn es eine Clique in G mit mindestens k Knoten gibt.**
- **Beobachtung: CLIQUE ist in $\mathcal{NP}$.**
- **Behauptung: CLIQUE ist in $\mathcal{NPC}$.**

Beweisskizze MAX-CLIQUE in NPC

TECHNISCHE UNIVERSITÄT DARMSTADT

- Nimm eine beliebige Formel *F* in konjunktiver Normalform her.
- Definiere $G=(V,E)$ aus *F*:
  - Für jedes Literal *x* in einer Klausel *K* von *F*, ein Knoten in *V* als Repräsentant für (x,K).
  - Eine Kante zwischen (x_1,K_1) und (x_2,K_2) genau dann, wenn $K_1 \neq K_2$ und x_2 nicht die Negation von x_1 ist.

Beweisskizze MAX-CLIQUE in NPC

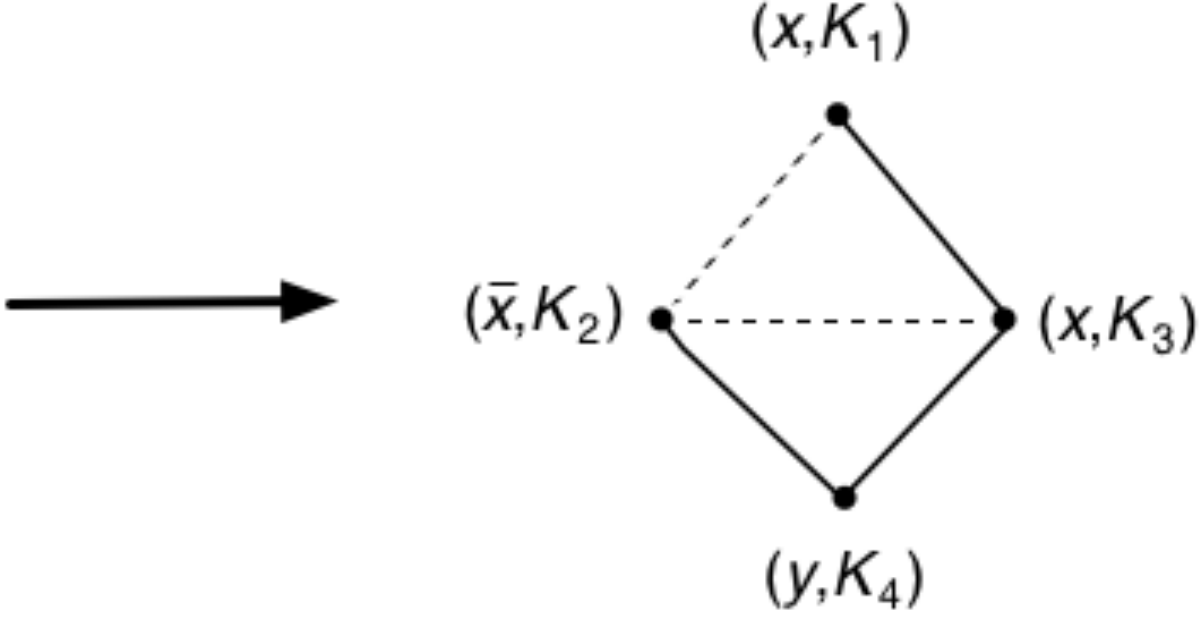
TECHNISCHE UNIVERSITÄT DARMSTADT

$K_1 = (\ldots \wedge x \wedge \ldots)$

$K_2 = (\ldots \wedge \bar{x} \wedge \ldots)$

$K_3 = (\ldots \wedge x \wedge \ldots)$

$K_3 = (\ldots \wedge y \wedge \ldots)$



Beweisskizze MAX-CLIQUE in NPC

TECHNISCHE UNIVERSITÄT DARMSTADT

- Definiere *k* als die Anzahl der Klauseln in *F*.
- *Beobachtung*: Die erfüllenden Variablenbelegungen entsprechen den *k*-Cliquen in *G*.
  - Clique der Größe *k* ➔ Setze alle Literale in der Clique auf true.
  - Erfüllende Belegung ➔ Je ein true-Literal aus jeder Klausel ergeben eine *k*-Clique.

Grundprinzip der Argumentation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ein erstes Element von NPC wird identifiziert.
- Dann wird Circuit-SAT auf andere algorithmische Probleme polynomiell reduziert.
- Dann werden weitere algorithmische Probleme auf letztere polynomiell reduziert.
- Und so weiter ...
- Auch TSP, Steinerbaum usw. werden so polynomial reduziert.