Sei $f(n) \geq \log_2 n$ (und so, dass es mit O(f(n)) Platz und Zeit berechnet werden kann)

**Satz D:** (Sprachkomplement)

$DTIME(f(n)) = \text{co-}DTIME(f(n))$ $DSPACE(f(n)) = \text{co-}DSPACE(f(n))$

$NSPACE(f(n)) = \text{co-}NSPACE(f(n))$

Dabei ist für Sprachklasse X die Sprachklasse co-X definiert als

$\text{co-}X = \{ \text{Sprache } L \mid \overline{L} \in X \}$



Sei $f(n) \geq \log_2 n$ (und so, dass es mit O(f(n)) Platz und Zeit berechnet werden kann)

**Satz E:** (Hierarchie) $f \in O(g)$

$DTIME(f(n)) \subseteq DTIME(g(n))$ $DSPACE(f(n)) \subseteq DSPACE(g(n))$

$NTIME(f(n)) \subseteq NTIME(g(n))$ $NSPACE(f(n)) \subseteq NSPACE(g(n))$

**Satz F:** (strikte Hierarchie)

$f(n) \in o(g(n)) \Rightarrow DSPACE(f(n)) \subsetneq DSPACE(g(n))$

$NSPACE(f(n)) \subsetneq NSPACE(g(n))$

$NTIME(f(n)) \subsetneq NTIME(g(n))$

$f(n) \in o(g(n)/\log g(n)) \Rightarrow DTIME(f(n)) \subsetneq DTIME(g(n))$



Sei $f(n) \geq \log_2 n$ (und so, dass es mit O(f(n)) Platz und Zeit berechnet werden kann)

Diese "constructibility"-Bedingung ist wesentlich! Ohne sie können recht eigenartige Phänomene auftreten:

**Satz Z:** (Borodin's Lückensatz)

Sei g() eine überall berechenbare rekursive Funktion.
Dann existiert eine überall berechenbare Funktion T() mit

$DTIME(S(n)) = DTIME(g(S(n)))$

z.B. für $g(n)=2^n$ beweist dieser Satz die Existenz einer Funktion S() mit

$DTIME(S(n)) = DTIME(2^{(S(n))})$,

also exponentiell mehr Zeit "bringt nichts".

Analoges gilt für NTIME, NSPACE, DSPACE.



Sei $f(n) \geq \log_2 n$ (und so, dass es mit O(f(n)) Platz und Zeit berechnet werden kann)

Diese "constructibility"-Bedingung ist wesentlich! Ohne sie können recht eigenartige Phänomene auftreten:

**Satz Y:** (Blum's Beschleunigungssatz)

Sei g() eine überall berechenbare rekursive Funktion.
Dann existiert eine TM-Sprache L sodass

$\forall$ TM M mit $L(M)=L$ $\exists$ TM M' mit $L(M')=L$ sodass $g(T_{M'}(n)) \leq_{fü} T_M(n)$

$T_M(n)$ ist dabei die worst-case Laufzeit von Turingmaschine M

z.B. für $g(n)=2^n$ beweist dieser Satz die Existenz einer Sprache L, für die jede sie akzeptierende Turingmaschine durch eine exponentiell schnellere ersetzt werden kann.

Analoges gilt für NTIME, NSPACE, DSPACE.

## Komplexitätsklassen

### Klasse P

$$\mathrm{P} = \bigcup_{k \geq 0} \mathrm{DTIME}(n^k)$$

In P sind Probleme, die sich in polynomieller Zeit deterministisch lösen lassen.



## Komplexitätsklassen

### Klasse P

$$\mathrm{P} = \bigcup_{k \geq 0} \mathrm{DTIME}(n^k)$$

In P sind Probleme, die sich in polynomieller Zeit deterministisch lösen lassen.

### Klasse NP

$$\mathrm{NP} = \bigcup_{k \geq 0} \mathrm{NTIME}(n^k)$$

In NP sind Probleme, deren Lösungen sich in polynomieller Zeit verifizieren lassen.



## Komplexitätsklassen

### Weitere wichtige Klassen

$$\mathrm{PSPACE} = \bigcup_{k \geq 0} \mathrm{DSPACE}(n^k)$$

$$\mathrm{NPSPACE} = \bigcup_{k \geq 0} \mathrm{NSPACE}(n^k)$$

$$\mathrm{L} = \mathrm{DSPACE}(\log \mathrm{n})$$

$$\mathrm{NL} = \mathrm{NSPACE}(\log \mathrm{n})$$



## Komplexitätsklassen

### Satz

*Es gilt*

$$\mathrm{L} \subseteq \mathrm{NL} \subseteq^* \mathrm{P} \subseteq^* \mathrm{NP} \subseteq^* \mathrm{PSPACE} = \mathrm{NPSPACE},$$

*wobei wenigstens eine der Inklusionen $\subseteq^*$ eine echte Inklusion $\subsetneq$ ist.*

### Beweis.

Benutze die Ergebnisse aus der 25. Vorlesung. □

## Komplexitätsklassen

- Besonders wichtig sind im Folgenden die Klassen P und NP.
- Probleme in P sind *effizient* lösbar.
  Bei Problemen außerhalb $P$ ist das vermutlich nicht möglich.

| | | |
|---|---|---|
| polynomiell | $\approx$ | „gut“ |
| suprapolynomiell | $\approx$ | „schlecht“ |



## Cliquen

**Definition**

Es sei $G = (V, E)$ ein ungerichteter Graph. Eine Eckenmenge $W \subseteq V$ heiße eine *Clique*, wenn für alle Ecken $u, v \in W$ gilt: $[u, v] \in E$.
(Die Ecken aus $W$ bilden einen vollständigen Untergraphen.)

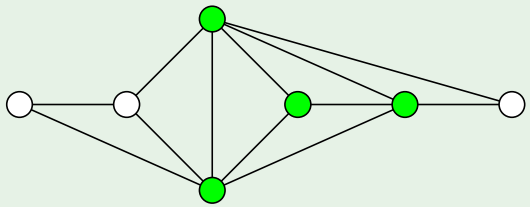


## Cliquen

**Definition**

Es sei $G = (V, E)$ ein ungerichteter Graph. Eine Eckenmenge $W \subseteq V$ heiße eine *Clique*, wenn für alle Ecken $u, v \in W$ gilt: $[u, v] \in E$.
(Die Ecken aus $W$ bilden einen vollständigen Untergraphen.)

**Beispiel**



Die markierten Ecken bilden eine Clique der Größe 4.



## Varianten des Clique-Problems

Es sei $G$ ein ungerichteter Graph.

| Variante | Gegeben | Frage/Aufgabe |
|---|---|---|
| Entscheidbarkeit | $G$, $k \in \mathbb{N}$ | Existiert eine Clique der Größe $k$? |
| Optimierung | $G$ | Bestimme maximales $k$, so dass $G$ eine Clique der Größe $k$ besitzt. |
| Optimierung/ Berechnung | $G$ | Berechne eine Clique maximaler Größe. |

## Wie hängen die Varianten zusammen?

**Klar ist**

1. Var. 2 in polyn. Zeit lösbar $\Longrightarrow$ Var. 1 in polyn. Zeit lösbar
2. Var. 3 in polyn. Zeit lösbar $\Longrightarrow$ Var. 2 in polyn. Zeit lösbar



## Wie hängen die Varianten zusammen?

**Klar ist**

1. Var. 2 in polyn. Zeit lösbar $\Longrightarrow$ Var. 1 in polyn. Zeit lösbar
2. Var. 3 in polyn. Zeit lösbar $\Longrightarrow$ Var. 2 in polyn. Zeit lösbar

**Lemma**

1. *Var. 1 in polyn. Zeit lösbar $\Longrightarrow$ Var. 2 in polyn. Zeit lösbar*
2. *Var. 2 in polyn. Zeit lösbar $\Longrightarrow$ Var. 3 in polyn. Zeit lösbar*



## Wie hängen die Varianten zusammen?

**Klar ist**

1. Var. 2 in polyn. Zeit lösbar $\Longrightarrow$ Var. 1 in polyn. Zeit lösbar
2. Var. 3 in polyn. Zeit lösbar $\Longrightarrow$ Var. 2 in polyn. Zeit lösbar

**Lemma**

1. *Var. 1 in polyn. Zeit lösbar $\Longrightarrow$ Var. 2 in polyn. Zeit lösbar*
2. *Var. 2 in polyn. Zeit lösbar $\Longrightarrow$ Var. 3 in polyn. Zeit lösbar*

**Beweis.**

1. Teste für jedes $k$, $1 \leq k \leq |V|$, ob $G$ eine Clique der Größe $k$ enthält.

□



## Wie hängen die Varianten zusammen?

**Klar ist**

1. Var. 2 in polyn. Zeit lösbar $\Longrightarrow$ Var. 1 in polyn. Zeit lösbar
2. Var. 3 in polyn. Zeit lösbar $\Longrightarrow$ Var. 2 in polyn. Zeit lösbar

**Lemma**

1. *Var. 1 in polyn. Zeit lösbar $\Longrightarrow$ Var. 2 in polyn. Zeit lösbar*
2. *Var. 2 in polyn. Zeit lösbar $\Longrightarrow$ Var. 3 in polyn. Zeit lösbar*

**Beweis.**

1. Teste für jedes $k$, $1 \leq k \leq |V|$, ob $G$ eine Clique der Größe $k$ enthält.
2. 
   1. Bestimme die Größe $k$ der maximalen Clique.
   2. Entferne sukzessive Ecken und teste, ob der resultierende Graph immer noch eine Clique der Größe $k$ enthält.

□

# Clique-Problem

## Varianten des Clique-Problems

- Entscheidungsproblem
- Optimierungsproblem
- Berechnungsproblem

## Wir haben festgestellt:

Ist eine der Varianten in polynomieller Zeit berechenbar, dann auch jede der anderen.



# Clique-Problem (Fortsetzung)

## Frage:

Gibt es einen effizienten Algorithmus für das Clique-Problem?



# Clique-Problem (Fortsetzung)

## Frage:

Gibt es einen effizienten Algorithmus für das Clique-Problem?

## Vermutung:

Nein.

Aber: Das Clique-Problem (Entscheidungs-Variante) ist in NP.
D. h. es gibt einen nichtdet. Algorithmus, der in Polynomialzeit verifiziert, dass $G$ eine Clique der Größe $k$ hat.



# Clique-Problem (Fortsetzung)

## Frage:

Gibt es einen effizienten Algorithmus für das Clique-Problem?

## Vermutung:

Nein.

Aber: Das Clique-Problem (Entscheidungs-Variante) ist in NP.
D. h. es gibt einen nichtdet. Algorithmus, der in Polynomialzeit verifiziert, dass $G$ eine Clique der Größe $k$ hat.

## Bemerkung

Das ist ein Beispiel von vielen wichtigen Problemen, für die man eine Lösung effizient verifizieren kann, man aber nicht weiß, wie man eine Lösung effizient findet.

# Polynomialzeitreduktionen

**Definition**

Es seien $L, L' \in \Sigma^*$. Wir nennen $L$ *polynomiell leichter* als $L'$ bzw. $L$ *polynomiell reduzierbar* auf $L'$, wenn es eine Funktion $f : \Sigma^* \to \Sigma^*$ gibt mit

1. $\forall w \in \Sigma^* : w \in L \Leftrightarrow f(w) \in L'$,
2. $f$ ist in polynomieller Zeit berechenbar.

Wir schreiben $L \preccurlyeq_P L'$.



# Polynomialzeitreduktionen

**Definition**

Es seien $L, L' \in \Sigma^*$. Wir nennen $L$ *polynomiell leichter* als $L'$ bzw. $L$ *polynomiell reduzierbar* auf $L'$, wenn es eine Funktion $f : \Sigma^* \to \Sigma^*$ gibt mit

1. $\forall w \in \Sigma^* : w \in L \Leftrightarrow f(w) \in L'$,
2. $f$ ist in polynomieller Zeit berechenbar.

Wir schreiben $L \preccurlyeq_P L'$.

**Beispiel**

Jede Variante des Clique-Problems ist polynomiell reduzierbar auf jede Variante des Clique-Problems.



# Polynomialzeitreduktionen

**Lemma**

*Es seien $L, L' \in \Sigma^*$. Dann gilt*

$$L \preccurlyeq_P L' \text{ und } L' \in \mathrm{P} \Rightarrow L \in \mathrm{P}.$$

*Die Aussage ist auch richtig, wenn man* P *durch* NP *ersetzt.*



# Polynomialzeitreduktionen

**Lemma**

*Es seien $L, L' \in \Sigma^*$. Dann gilt*

$$L \preccurlyeq_P L' \text{ und } L' \in \mathrm{P} \Rightarrow L \in \mathrm{P}.$$

*Die Aussage ist auch richtig, wenn man* P *durch* NP *ersetzt.*

**Beweis.**

- $L \preccurlyeq_P L'$ : Es gibt TM $F$, die $f : \Sigma^* \to \Sigma^*$ berechnet mit $w \in L \Leftrightarrow f(w) \in L'$, Laufzeit $O(n^k)$.
- $L' \in \mathrm{P}$ : Es gibt (det.) TM $M'$, die $L'$ entscheidet, Laufzeit $O(n^l)$.

□

## Polynomialzeitreduktionen

**Lemma**

*Es seien $L, L' \in \Sigma^*$. Dann gilt*

$$L \preccurlyeq_P L' \text{ und } L' \in \mathrm{P} \Rightarrow L \in \mathrm{P}.$$

*Die Aussage ist auch richtig, wenn man* P *durch* NP *ersetzt.*

**Beweis.**

- $L \preccurlyeq_P L'$ : Es gibt TM $F$, die $f : \Sigma^* \to \Sigma^*$ berechnet mit $w \in L \Leftrightarrow f(w) \in L'$, Laufzeit $O(n^k)$.
- $L' \in \mathrm{P}$ : Es gibt (det.) TM $M'$, die $L'$ entscheidet, Laufzeit $O(n^l)$.
- Wir bauen eine (det.) TM $M$ für $L$ (Eingabe $w$):
  1) Berechne $f(w)$ mit F.
  2) Teste mit $M'$ ob $f(w) \in L'$ ist.

□



## Polynomialzeitreduktionen

**Lemma**

*Es seien $L, L' \in \Sigma^*$. Dann gilt*

$$L \preccurlyeq_P L' \text{ und } L' \in \mathrm{P} \Rightarrow L \in \mathrm{P}.$$

*Die Aussage ist auch richtig, wenn man* P *durch* NP *ersetzt.*

**Beweis.**

- $L \preccurlyeq_P L'$ : Es gibt TM $F$, die $f : \Sigma^* \to \Sigma^*$ berechnet mit $w \in L \Leftrightarrow f(w) \in L'$, Laufzeit $O(n^k)$.
- $L' \in \mathrm{P}$ : Es gibt (det.) TM $M'$, die $L'$ entscheidet, Laufzeit $O(n^l)$.
- Wir bauen eine (det.) TM $M$ für $L$ (Eingabe $w$):
  1) Berechne $f(w)$ mit F.
  2) Teste mit $M'$ ob $f(w) \in L'$ ist.
- $M$ entscheidet $L$, Laufzeit: $O(n^k) + O(|f(w)|^l) \in O(n^{kl})$

□



## Polynomialzeitreduktionen

**Lemma (Transitivität)**

*Es seien $L, L', L'' \in \Sigma^*$. Dann gilt*

$$L \preccurlyeq_P L' \text{ und } L' \preccurlyeq_P L'' \Rightarrow L \preccurlyeq_P L''.$$