Technische Universität Braunschweig

Parametrisierte Algorithmen

Arne Schmidt

Vorstellung

Interessen

- Geometrische Optimierung,
- Programmierbare Materie,
- Komplexitätstheorie

#GernePerDu

Computing MaxMin Edge Length Triangulations

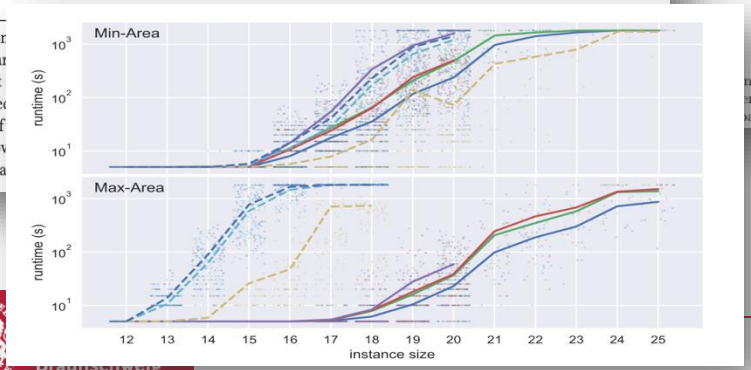
Sándor P. Fekete * Winfried Hellmann* Michael Hemmer* Arne Schmidt*
Julian Troegel*

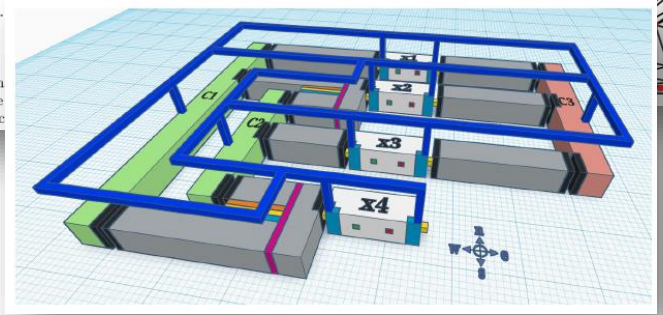
Abstract

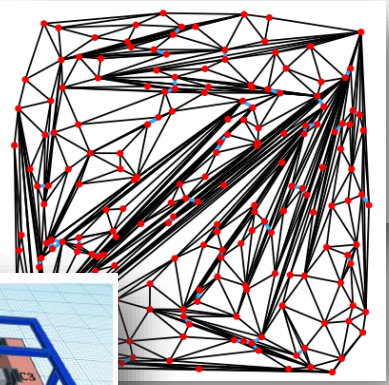
Particle-Based Assembly
Using Precise Global Control*

Computing Area-Optimal Simple Polygonizations

SÁNDOR P. FEKETE, ANDREAS HAAS, PHILLIP KELDENICH, MICHAEL PERK, and ARNE SCHMIDT, Department of Computer Science, TU Braunschweig







Organisation

Vorlesung

- Grundlagen

+

Gr. Übung

- Live Coding
- Fragerunden

+

Kl. Übung

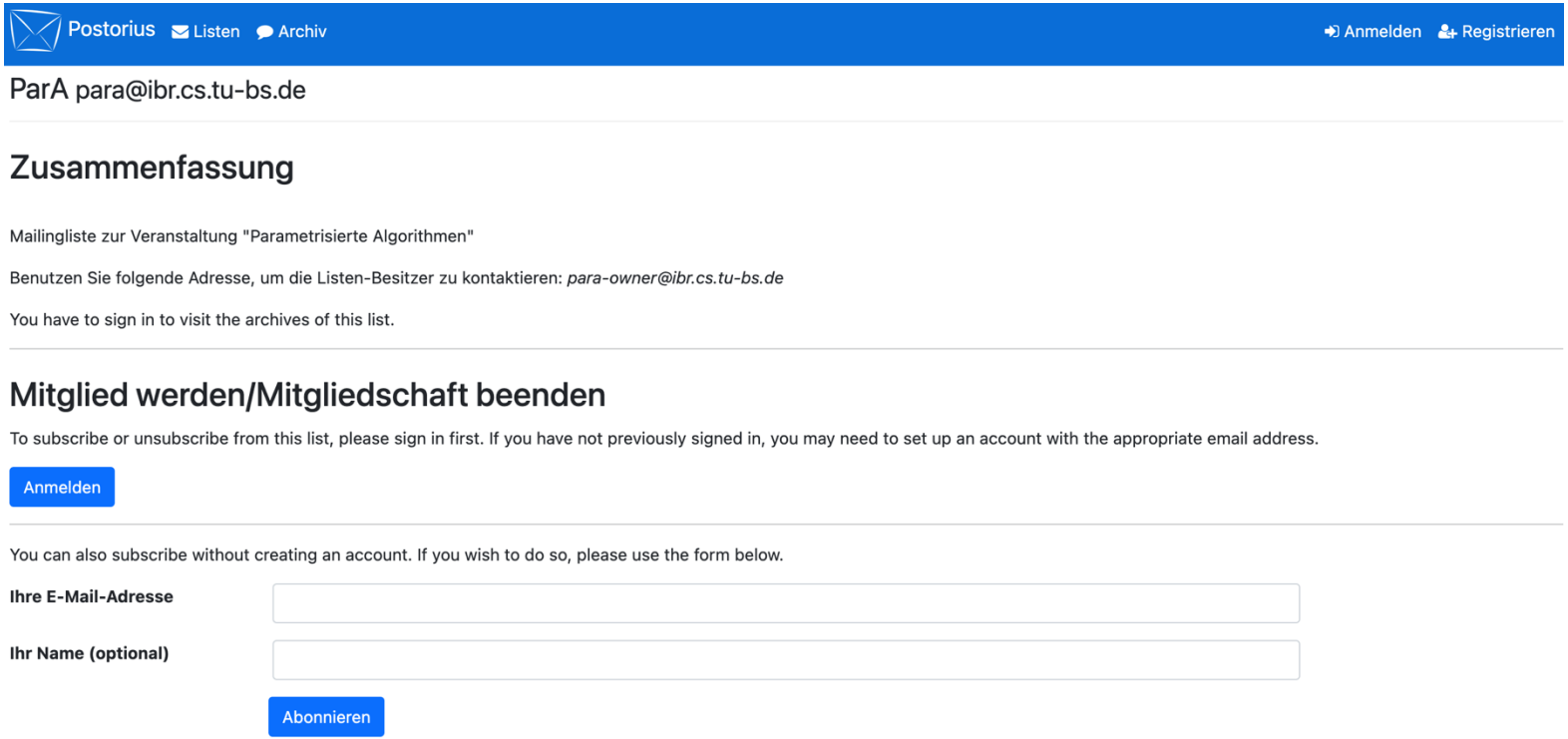
- Hausaufgaben

Arne Schmidt

Fragen

Inhaltlich oder allgemeiner Ablauf	Vorlesung / große Übung
Zu Übungsblättern	Kleine Übungen
Individuelle Fragen	**Immer per Mail** an aschmidt@ibr.cs.tu-bs.de
Sprechstunde	Montags, 09:45 Uhr im Raum IZ 333 (Am besten vorher per Mail ankündigen)

Mailingliste (https://lists.ibr.cs.tu-bs.de/postorius/lists/para.ibr.cs.tu-bs.de/)



Semesterplan (vorläufig)

Datum (VL / Ü)	VL / Ü Nummer	VL / Ü Inhalt	Aufgaben (Ausgabe)	Aufgaben (Besprechung)
22. Okt	- , -			
29. Okt	V1, -	Einleitung, Recap		
05. Nov	V2, U1	Bar Fight Problem		
12. Nov	V3, -	Kernelization	HA1	
19. Nov	V4, U2	Beispiele		
26. Nov	V5, K1	Matchings	HA2	HA1
03. Dez	V6, U4	Crown Decomposition		
10. Dez	V7, K2	Bounded Tree Search	HA3	HA2
17. Dez	V8, U5	Feedback Vertex Sets		
24. Dez	- , -			
31. Dez	- , -			
07. Jan	V9, K3	Iterative Compression	HA4	HA3
14. Jan	V10, U7	Tree Width		
21. Jan	V11, K4	Intractability	HA5	HA4
28. Jan	V12, U9	W-Hierarchie		
04. Feb	V13, K5	Zusammenfassung		HA5

Hausaufgaben

Insgesamt 5 Präsenzblätter.

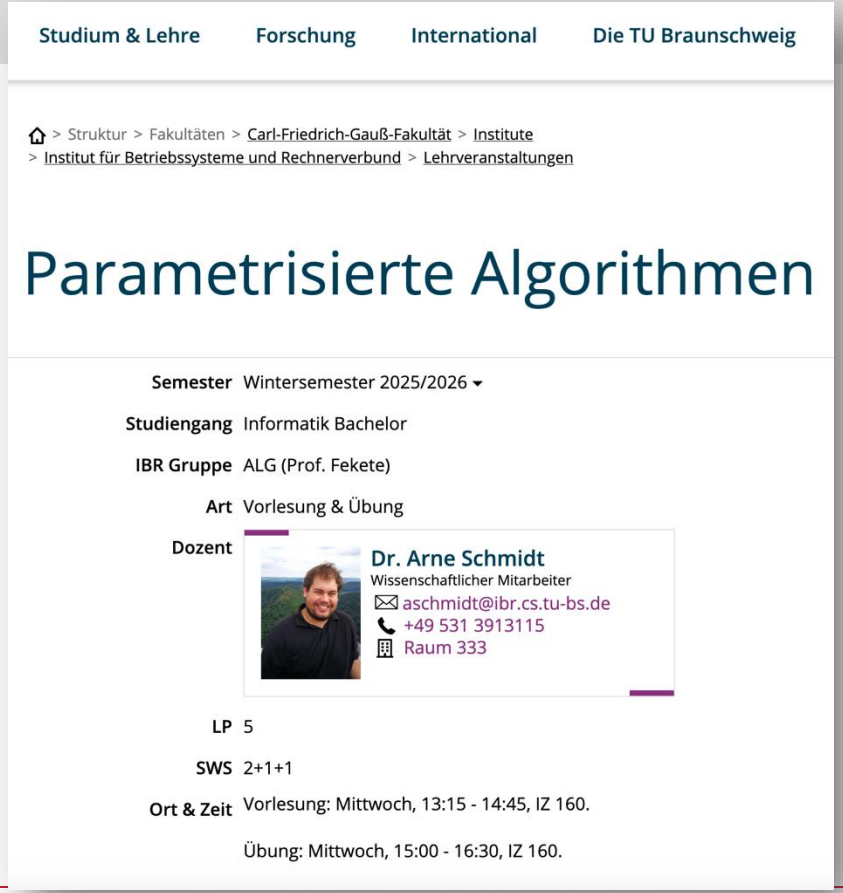
Studienleistung ist bestanden.

Aufgaben werden nur besprochen und diskutiert.

Unterlagen

- Öffentlich zugänglich
- Wird enthalten:
  - Slides (VL / UE)
  - Hausaufgaben
  - Klausurinformationen
  - ...

https://www.ibr.cs.tu-bs.de/courses/ws2526/para/



Klausur* – Vorläufig

*: Bei wenigen Teilnehmenden kann eine mündliche Prüfung erfolgen

Datum:
12.03.26

Uhrzeit:
13:00 – 15:00

Ort:
TBA

Inhalt:
Hauptsächlich
Vorlesung

Fragen?

Recap – Graphen

Graphen

Definition 0.1 (Graphen)

Ein **ungerichteter Graph** ist ein Tupel $G = (V, E)$ mit $E \subseteq \binom{V}{2}$. Dabei gilt

- Die Menge V heißt Knotenmenge, die Menge E heißt Kantenmenge.
- Die Nachbarschaft $N(v)$ eines Knotens $v \in V$ ist definiert durch $N(v) \coloneqq \{ w \in V \mid \{v, w\} \in E \}$
- Der Grad $\delta(v)$ eines Knotens $v \in V$ ist definiert durch $\delta(v) \coloneqq |N(v)|$

Ein Graph heißt **einfach**, wenn er keine Kante der Form $\{v, v\}$ für einen Knoten $v \in V$ besitzt.

Eine Folge $P \coloneqq (p_1, p_2, \dots, p_k)$ heißt

- **Kantenfolge**, wenn $\{p_i, p_{i+1}\} \in E$ für alle $1 \leq i < k$.
- **Weg**, wenn P eine Kantenfolge ist und jede Kante maximal einmal vorkommt.
- **Pfad**, wenn P ein Weg ist und jeder Knoten maximal einmal vorkommt.
- **Kreis**, wenn P ein Weg, P ohne p_k ein Pfad ist und $p_k = p_1$ gilt.

Digraphen

Definition 0.2 (Digraphen)
Ein **Digraph** (gerichteter Graph) ist ein Tupel $D = (V, A)$ mit $A \subseteq V \times V$. Dabei gilt

- Die Menge V heißt Knotenmenge, die Menge A heißt gerichtete Kantenmenge.
- $N^+(v) \coloneqq \{ w \in V \mid (v, w) \in A \}$ ist die positive Nachbarschaft eines Knotens $v \in V$
- $\delta^+(v) \coloneqq |N^+(v)|$ ist der Ausgangsgrad $\delta(v)$ eines Knotens $v \in V$
- Negative Nachbarschaft $N^-(v)$ und Eingangsgrad $\delta^-(v)$ sind analog definiert.

Ein Digraph heißt **einfach**, wenn er keine Kante der Form (v, v) für einen Knoten $v \in V$ besitzt.

Gerichtete Kantenfolgen, Wege, Pfade und Kreise sind analog definiert.

Handschlag(di)lemma

Lemma 0.3
Für jeden einfachen, ungerichteten Graphen $G = (V, E)$ gilt $\sum_{v \in V} \delta(v) = 2|E|$.

Für jeden einfachen Digraphen $D = (V, A)$ gilt $\sum_{v \in V} \delta^-(v) = \sum_{v \in V} \delta^+(v) = |E|$.

Besondere Graphenklassen

Vollständiger Graph:

$$E = \binom{V}{2}$$

Bipartiter Graph:

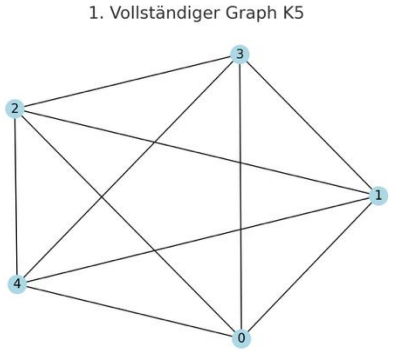
$V = V_1 \dot{\cup} V_2, E \subseteq \{\{v, w\} \mid v \in V_1, w \in V_2\}$

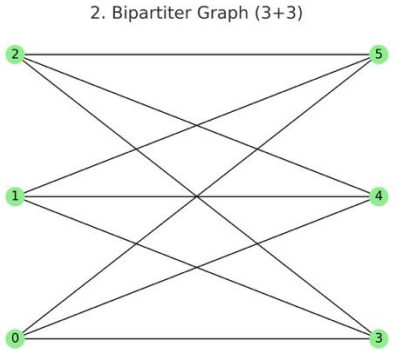
Baumgraph:

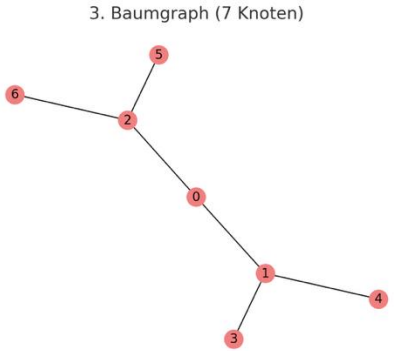
Besitzt keinen Kreis und ist zusammenhängend

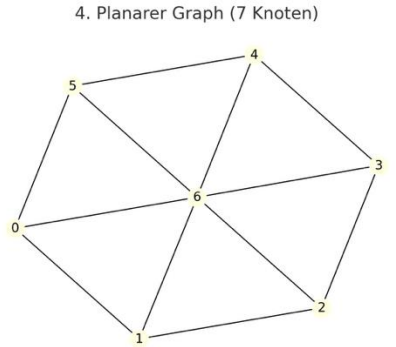
Planare Graphen:

Lassen sich ohne kreuzende Kanten zeichnen









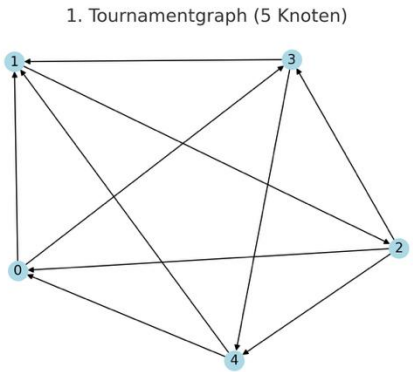
Besondere Digraphen

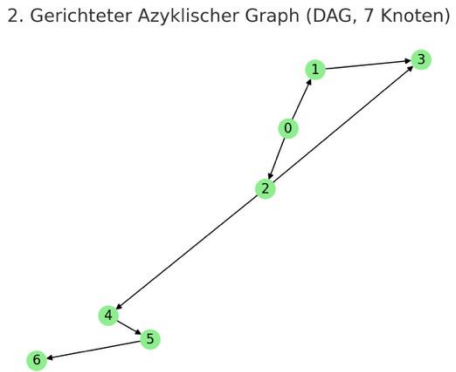
Tournamengraph:
Ungerichtet betrachtet, ist der Graph vollständig

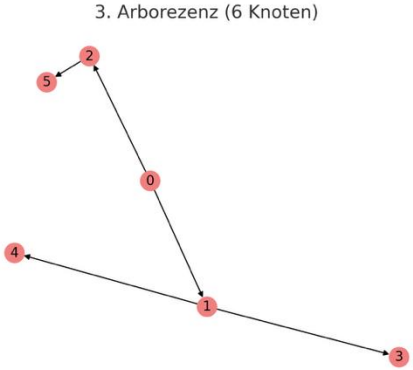
Azyklischer Graph
Besitzt keinen gerichteten Kreis

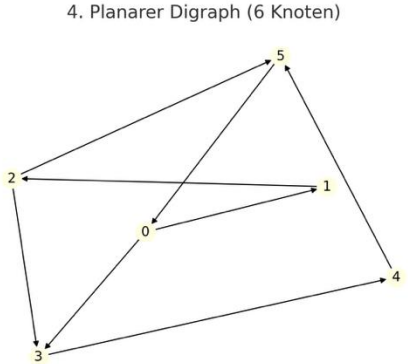
Arboreszenz:
Keine ungerichteten Kreise, ein Knoten kann alle anderen Knoten erreichen.

Planare Graphen:
Lassen sich ohne kreuzende Kanten zeichnen









Recap – NP-schwere

Entscheidungsprobleme

Definition 0.4
Sei $\mathcal{L} \subseteq \Sigma^*$ eine Sprache über ein Alphabet Σ.

Ein Entscheidungsproblem fragt, ob ein gegebenes Wort $w \in \Sigma^*$ zu $\mathcal{L}$ gehört.

Beispiel 0.5
Gegeben: Graph $G = (V, E)$
Frage: Ist G ein Baumgraph?

Für dieses Problem ist

$$\mathcal{L} = \{\text{Graph } G \mid G \text{ enthält keinen Kreis und ist zusammenhängend}\}$$

Lösen von Problemen

Beispiel 0.5
Gegeben: Graph $G = (V, E)$
Frage: Ist G ein Baumgraph?

Wie schnell lässt sich dieses Problem lösen?

Man kann zeigen:
G ist ein Baumgraph, gdw. G ist zusammenhängend und besitzt $|V| - 1$ Kanten.

Das Problem kann in Zeit $O(|V|)$ gelöst werden.
Wieso kann ich nicht direkt BFS ausführen, um Zusammenhang zu testen?

O-Notation

Definition 0.6
Seien $f, g: \mathbb{N} \to \mathbb{R}$ Funktionen. Dann ist

$$f(n) \in O\big(g(n)\big) \quad \Leftrightarrow \quad \exists c \in \mathbb{R}_{>0}, n_0 \in \mathbb{N}: \quad \forall n \geq n_0: \quad 0 \leq f(n) \leq c \cdot g(n)$$

Beispiele 7.2

a) $O(1) = O(5000) = O(k)$ für alle Konstanten $k \in \mathbb{N}$.

b) $O(\log n)$ ist die Klasse der logarithmischen Funktionen. Bspw. gilt $1 \in O(\log n)$, aber $\log n \notin O(1)$

c) $O(n), O(n^2), O(n^3)$, usw. sind die Klasse der linearen, quadratischen, kubischen, ... Funktionen. Zu jedem Polynom $p(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0$ gilt $p(n) \in O\big(n^k\big)$.

d) $\{n \mapsto 2^{f(n)} \mid f(n) \in O(n)\}$ ist die Klasse der exponentiellen Funktionen mit lin. Exponenten. Kurzschreibweise für solche Funktionen: $2^{O(n)}$

Ω-Notation

Definition 0.6
Seien $f, g: \mathbb{N} \to \mathbb{R}$ Funktionen. Dann ist

$$f(n) \in \Omega\big(g(n)\big) \quad \Leftrightarrow \quad \exists c \in \mathbb{R}_{>0}, n_0 \in \mathbb{N}: \quad \forall n \geq n_0: \quad 0 \leq c \cdot g(n) \leq f(n)$$

Beispiele 7.2

a) $\Omega(1) = \Omega(5000) = \Omega(k)$ für alle Konstanten $k \in \mathbb{N}$.

b) $\Omega(\log n)$ ist die Klasse der logarithmischen Funktionen. Bspw. gilt $1 \notin \Omega(\log n)$, aber $\log n \in \Omega(1)$

c) $\Omega(n), \Omega(n^2), \Omega(n^3)$, usw. sind die Klasse der linearen, quadratischen, kubischen, ... Funktionen. Zu jedem Polynom $p(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0$ gilt $p(n) \in \Omega\big(n^k\big)$.

Komplexitätsklassen

Definition 0.7
Sei Π ein Entscheidungsproblem. Dann gilt:

- **$\Pi \in$ P**, wenn Π in polynomieller Laufzeit entschieden werden kann.
- **$\Pi \in$ NP**, wenn Π nicht-deterministisch in polynomieller Laufzeit entschieden werden kann.
- **Π ist NP-schwer**, wenn jedes Problem aus NP auf Π in polynomieller Zeit reduziert werden kann.
- **Π ist NP-vollständig**, wenn Π NP-schwer ist und in NP enthalten ist.

Definition 0.8
Seien $A, B \subseteq \Sigma^*$ zwei Entscheidungsprobleme. Eine berechenbare Funktion $f: \Sigma^* \to \Sigma^*$ ist eine **Polynomialzeitreduktion**, wenn

1. Es gilt genau dann $w \in A$, wenn $f(w) \in B$
2. Ein Algorithmus der f berechnet, besitzt polynomielle Laufzeit.

Millenniums-Problem

Gilt P ≠ NP?

Man vermutet, dass die Aussage gilt. Damit könnten viele interessante Probleme deterministisch nicht in polynomieller Zeit gelöst werden.

Was kann man in diesem Fall tun?


HELP ME THEORY
YOU'RE MY ONLY HOPE
imgflip.com

Allgemeine Techniken in AuD 2

Brute Force:
Teste alle Möglichkeiten durch.

Dynamic Programming, Branch&Bound:
Etwas intelligenteres Brute Force.

Approximationen:
Für Optimierungsprobleme, finde eine gute Lösung. “Semi-Entscheidet”

Heuristiken:
Finde schnell eine mögliche Lösung. ”Semi-Entscheidet”

“Semi-Entscheidet”: Falls ein solcher Algorithmus “ja” zurückgibt, ist die Instanz eine “ja”-Instanz. Andernfalls, ist es nicht klar.

Parametrisierte Algorithmen

In der Praxis tauchen Worst-Case-Instanzen nicht immer auf. Damit kann man sich folgende Fragen stellen:

Kann man Instanzen klassifizieren, die einfach sind?

Kann man gewisse Eigenschaften von Instanzen nutzen, um effizientere Algorithmen zu entwerfen?

Wenn wir Eigenschaft / Parameter k fixieren, können wir solche Instanzen effizient lösen?

Was macht schwere Instanzen überhaupt schwer?

Semesterplan (vorläufig)

	Datum (VL / Ü)	VL / Ü Nummer	VL / Ü Inhalt	Aufgaben (Ausgabe)	Aufgaben (Besprechung)
	22. Okt	- , -			
Einführung	**29. Okt**	V1, -	Einleitung, Recap		
Einführung	**05. Nov**	V2, U1	Bar Fight Problem		
Allgemeine Techniken	**12. Nov**	V3, -	Kernelization	HA1	
Allgemeine Techniken	**19. Nov**	V4, U2	Beispiele		
Allgemeine Techniken	**26. Nov**	V5, K1	Matchings	HA2	HA1
Allgemeine Techniken	**03. Dez**	V6, U4	Crown Decomposition		
Allgemeine Techniken	**10. Dez**	V7, K2	Bounded Tree Search	HA3	HA2
Allgemeine Techniken	**17. Dez**	V8, U5	Feedback Vertex Sets		
	24. Dez	- , -			
	31. Dez	- , -			
Allgemeine Techniken	**07. Jan**	V9, K3	Iterative Compression	HA4	HA3
Allgemeine Techniken	**14. Jan**	V10, U7	Tree Width		
Schwere von schweren Problemen	**21. Jan**	V11, K4	Intractability	HA5	HA4
Schwere von schweren Problemen	**28. Jan**	V12, U9	W-Hierarchie		
Schwere von schweren Problemen	**04. Feb**	V13, K5	Zusammenfassung		HA5

Nächstes Mal

Wie kann ich schnell als Türsteher nur Personen hereinlassen, die keine Probleme verursachen?

Ich darf aber maximal k Personen abweisen. Wie berechne ich das?