Graphentheorie

Vorlesung

Sommersemester 2003

Johannes Waldmann, Institut für Informatik
Universität Leipzig, joe@informatik.uni-leipzig.de

Kapitel 1

Vorwort

Zitiert aus Bollobas: Modern Graph Theory [Bol98]:

> ... Ein weiterer Grund für die Bedeutung der Graphentheorie im Lehrplan ist ihr Status als derjenige Zweig der reinen Mathematik, der der Informatik am nächsten ist — wovon beide Disziplinen profitieren. Einerseits ist die Graphentheorie eine der Grundlagen der theoretischen Informatik, andererseits sind Fragen aus der Informatik und anderen Anwendungsgebieten von großem Einfluß auf die Richtung, in der sich die Graphentheorie weiterenwickelt. Wir werden jedoch ... Anwendungen nicht betonen, sondern Graphentheorie als aufregenden Zweig der reinen Mathematik präsentieren, voller eleganter und innovativer Ideen.
>
> Ich hoffe, daß der (Hörer) die Schönheit und Wichtigkeit der wesentlichen Resultate und Beweise schätzen lernen wird. Für den werdenden Graphentheoretiker ist es jedoch der Kampf mit ungezählten harten Aufgaben genauso wichtig. (...) Den meisten Problemen außerhalb der reinen Mathematik fehlt es an klarer Struktur und offensichtlicher Lösungsmethode. Darin stimmen sie mit vielen Problemen der Graphentheorie überein: zu ihrer Lösung braucht man originelle Einfälle und Geschick. Deswegen werden sich die Mühe, die man zu ihrer Lösung aufwendet, und die Fähigkeiten, die man dabei erwirbt, nicht nur in der Graphentheorie selbst, oder in anderen Bereichen der Mathematik auszahlen, sondern auch in weiteren Wissenschaften.

Inhaltsverzeichnis

Literaturverzeichnis

[BLS00] Andreas Brandstädt, V. Bang Le, , and Jeremy P. Spinrad. *Graph Classes: A Survey*. Society for Industrial and Applied Mathematics, Philadelphia, 2000. http://www.informatik.uni-rostock.de/~ab/survey/survey.html.

[Bol79] Belá Bollobás. *Graph Theory — An Introductory Course*. Springer, 1979.

[Bol98] Belá Bollobás. *Modern Graph Theory*. Springer, 1998.

[Bra94] Andreas Brandstädt. *Graphen und Algorithmen*. Teubner, Stuttgart, 1994.

[BW02] Christian Biemann and Johannes Waldmann. Non-embedding primes and squares. http://www.informatik.uni-leipzig.de/~joe/projekte/wqo/paper/, 2002.

[CLR91] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, 1991. http://theory.lcs.mit.edu/~clr/.

[Die96] Reinhard Diestel. *Graphentheorie*. Springer, 1996. http://www.math.uni-hamburg.de/home/diestel/books/graph.theory/.

[Hal89] Rudolf Halin. *Graphentheorie*. Akademie–Verlag Berlin/Wissenschaftliche Buchgesellschaft Darmstadt, 1989.

[Tho96] Neil Robertson; Daniel P. Sanders; Paul Seymour; Robin Thomas. A new proof of the four-colour theorem, 1996. http://www.ams.org/journal-getitem?pii=S1079-6762-96-00003-0.

[Wal00] Johannes Waldmann. Kombinatorik auf endlichen strukturen (skript). http://www.informatik.uni-leipzig.de/~joe/edu/ss00/kombinat/, 2000.

[Wes01] Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, 2001. http://www.math.uiuc.edu/~west/igt/.

Kapitel 2

Einleitung

(Vorlesung vom 11. 4. – Literatur: jedes Graphentheoriebuch, Abschnitt 1)

2.1 Definition

Graph, Knoten, Kanten

Nachbarschaft $G(x)$, Knoten-Grad $\deg_G(x)$

2.2 Prädikate

isomorph $\cong$, Teilgraph $\subseteq$, induzierter Teilgraph

2.3 Elementare Graphen

vollständig (Clique) K_n, unabhängig (leer) E_n, Pfad P_n, Kreis C_n.

2.4 Operationen

Vereinigung $\cup$, disjunkte Summe $+$, Produkt $*$, Komplement $\overline{G}$.

selbst-komplementäre Graphen (Aufgabe).

Einschränkung $G\mid_{V'}$

2.5 Beweismethode: double counting

(Literatur: [Bol98], Abschnitt ??)

Satz 1. *Wenn* $|E| > |V|^2/4$, *dann* $K_3 \subseteq G$.

Beweis: $\sum_{xy \in E}(\deg x + \deg y) \leq n$ und Ungleichung von Cauchy-Schwarz.

Definition 2. *Ein* ebener Graph *ist eine Zeichung eines Graphen in der Ebene. Für ebenes* G *bezeichnet* $F(G)$ *die Menge der Länder von* G.

Jede Zeichnung enthält genau ein unendliches Land (den umgebenden Ozean).

Satz 3 (Euler). *Für zusammenhängende ebene Graphen gilt* $|V| + |F| - |E| = 2$.

Beweis durch Induktion nach $|F|$.

Definition 4. *Ein Graph heißt* planar *(plätt-bar), wenn er eine ebene Zeichnung besitzt.*

Beispiel 5. K_4 *ist planar.*

Definition 6. $\mathrm{girth}(G)$ *(Taillenweite) ist kleinstes* k *mit* $C_k \subseteq G$.

Satz 7. *Für zusammenhängende planare Graphen* G *mit* $|V| \geq 3$ *und* $\mathrm{girth}(G) = g$ *gilt:* $|E| \leq g(|V| - 2)/(g - 2)$.

Beweis: summiere für alle Länder die Anzahl der Kanten.

Anwendungen: K_5, $E_3 * E_3$, Petersen-Graph sind nicht planar.

Kapitel 3

Bäume

(Vorlesung vom 25. 4. – Literatur: [Wes01], Abschnitte 1.2, 2.1, 2.2)

3.1 Wege, Pfade, Zusammenhang

Definition 8. *Ein* Weg *in G ist eine Folge $(x_1, x_2, \dots x_k)$ von Knoten, so daß $\forall 1 \leq i < k : x_i x_{i+1} \in E(G)$. Ein* Pfad *in G ist ein Weg über paarweise verschiedene Knoten. (Wir nennen diese Folge einen $x_1 - x_k$-Weg bzw. -Pfad.)*

Aufgabe 9. *Zerlege K_n in kanten-disjunkte P_n. Zerlege K_n in kanten-disjunkte C_n. (Für welche n ist das möglich?)*

Definition 10. *Knoten u und v sind* verbunden *in G, geschrieben $u \sim_G v$, falls ein u, v-Pfad in G existiert.*

Satz 11. *Für jedes G ist $\sim_G$ eine Äquivalenzrelation auf $V(G)$.*

Bemerkung zum Beweis (Transitivität): falls wir einen $x - y$-Pfad und einen $y - z$-Pfad haben, dürfen wir diese nicht einfach hintereinanderkleben, da dadurch nur ein Weg, aber kein Pfad entsteht. Wir benutzen aber

Satz 12. *Wenn es einen $u - v$-Weg in G gibt, dann auch einen $u - v$-Pfad.*

Proof. Falls der Weg kein Pfad ist, dann enthält er einen Knoten x mehrfach, d. h. die Folge hat die Form $u \dots x \dots x \dots v$. Dann schneiden wir das Stück zwischen den beiden x weg. □

Definition 13. *Die Äquivalenzklassen von $V(G)$ bezüglich $\sim_G$ heißen* Zusammenhangskomponenten.

Definition 14. *Der Graph G heißt* zusammenhängend*, wenn er nur eine Zusammenhangskomponente besitzt.*

Die Anzahl der Z-Komponenten nennen wir (nur für diesen Abschnitt) $z(G)$.

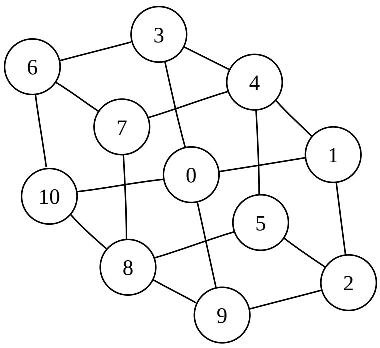
Definition 15. *Ein* Hamiltonkreis *in G ist ein Kreis C durch alle Knoten von G (d. h. $V(C) = V(G)$ und $E(C) \subseteq E(G)$). Die Menge aller Graphen mit Hamiltonkreis nennen wir* HC.

Es ist bekannt, daß die Frage $G \in$ HC schwer zu entscheiden ist (HC ist NP-vollständig). Es gibt jedoch einige notwendige Kriterien, zum Beispiel:

Satz 16. *Wenn $G \in \mathrm{HC}$, dann gilt für alle $\emptyset \neq S \subseteq V(G) : z(G - S) \leq |S|$.*

Proof. Weil $C \subseteq G$, gilt $z(G-S) \leq z(C-S)$. Ein Kreis zerfällt aber durch Löschen von k Knoten in höchstens k Komponenten. □

Aufgabe 17. *Beweise durch Angabe einer geeigneten Menge S: Der Herschel-Graph hat keinen HC:*



3.2 Bäume

Definition 18. *G heißt* Wald*, wenn G kreisfrei ist.*
G heißt Baum*, wenn G ein zusammenhängender Wald ist.*

Diese Bäume sind also nicht gerichtet, und haben keine ausgezeichnete Wurzel.

Satz 19. *Die folgenden Aussagen sind äquivalent:*

1. *G ist Baum*
2. *G ist minimal zusammenhängend,*
3. *G ist maximal kreisfrei,*
4. *zu je zwei Knoten $x, y \in V(G)$ gibt es* genau einen *$x - y$-Pfad.*

Hierbei bedeutet Minimalität: jeder G' mit $V(G') = V(G)$ und $E(G') \subset E(G)$ ist nicht zusammenhängend. (Entsprechend für Maximalität.)

Aufgabe 20. *Die folgenden Aussagen sind äquivalent:*

1. *G ist Baum,*
2. *G ist zusammenhängend und $|E| = |V| - 1$,*
3. *G ist kreisfrei und $|E| = |V| - 1$.*

3.3 Spannbäume

Definition 21. *Graph G' ist* aufspannender Teilgraph *von G, falls $V(G') = V(G)$ und $E(G') \subseteq E(G)$.*

Definition 22. *Graph T ist* Gerüst *(spanning tree) von G, falls T Baum und T aufspannender Teilgraph von G.*

Satz 23. *Jeder zusammenhängende Graph besitzt ein Gerüst.*

Beweis-Idee: lösche solange Kanten, bis ein Baum übrigbleibt.

(Später behandeln wir eine Verallgemeinerung: es ist zusätzlich eine Kantenbewertung $f : E(G) \to \mathbb{N}$ gegeben, und man sucht ein Gerüst mit kleinstmöglichen Gesamtkosten.)

Satz 24. *Für alle Gerüste T und T' von G gilt: für jedes $e \in E(T) \setminus E(T')$ existiert ein $e' \in E(T') \setminus E(T)$, so daß $T - e + e'$ ein Gerüst für G ist.*

Vorstellung: Durch das fehlende e geht T kaputt, aber durch e' wird er repariert.

Proof. Wir nennen $e = xy$. Der Graph $T - e$ besteht aus zwei Komponenten V_x, V_y mit $x \in V_x, y \in V_y$. Weil T' ein Gerüst ist, gibt es in T' (genau) einen $x - y$-Pfad. $x = z_1, z_2, \ldots, z_k = y$. Dann existiert ein i, für das $z_i \in V_x, z_{i+1} \in V_y$. Wir nehmen $e' = z_i z_{i+1}$. □

Das Shannon-Switching-Game ist ein Spiel für zwei Personen (Connector (rot) und Cutter (blau)). Gespielt wird auf einem Graphen G, mit zwei ausgezeichneten Knoten s, t. Die Kanten von G sind zunächst alle schwarz. In einem Spielzug darf:

- der Cutter eine schwarze Kante blau färben,
- der Connector eine schwarze Kante rot färben.

Beide ziehen abwechselnd, der Cutter beginnt. Der Connector gewinnt, sobald es einen roten $s - t$-Pfad gibt. (ansonsten gewinnt Cutter).

Satz 25 (Lehmann). *Connector hat eine Gewinnstrategie genau dann, wenn es G zwei kantendisjunkte Gerüste besitzt.*

Proof. Wenn die Gerüste existieren, dann gewinnt Connector so: während des Spiels gibt es immer zwei Gerüste T, T' für G, die nur rote oder schwarze Kanten enthalten, und die keine gemeinsamen schwarzen Kanten besitzen. Nach Spielregel löscht Cutter immer eine schwarze Kante, und nach Satz 24 kann Connector das durch eine rote Kante reparieren.

Aufgabe 26. *Vervollständige den Beweis. Wie gewinnt Cutter, wenn die disjunkten Gerüste nicht existieren?*

□

3.4 Graceful Labellings

Definition 27. *Für einen Graphen G heißt eine Abbildung (labelling) $f : V \to \{0, 1 \ldots, |E|\}$* graceful*, wenn f injektiv ist und $\{|f(u) - f(v)| : uv \in E\} = \{1, 2, \ldots, |E|\}$.*

d. h. alle Differenzen an den Kanten sind paarweise verschieden.

Definition 28. *Ein Graph G heißt* graceful*, wenn für ihn eine graceful labelling f existiert.*

Beispiel: K_4 ist graceful, benutze $f(V) = \{0, 1, 4, 6\}$.

Beispiel: Jeder Pfad ist graceful, numeriere so: $0, (n-1), 1, (n-2), 2, (n-3), \ldots$.

Das ist nur ein Spezialfall von:

Definition 29. *Ein Baum T heißt* Raupe *(caterpillar), falls nach Löschen aller Blätter von T ein Pfad entsteht.*

Aufgabe 30. *Jede Raupe ist graceful.*

Das wiederum ist nur ein Spezialfall von:

Vermutung 31. *Jeder Baum ist graceful.*

Diese Vermutung ist seit ca. 1970 *offen*. Das kann doch gar nicht sein, denkt man. Daß es nicht so leicht ist, sieht man z. B. hier:

Aufgabe 32. *Jeder vollständige binäre Baum ist graceful.*

Schon der nächste Schritt ist unklar:

Definition 33. *Ein Pfad ist eine 0-Raupe. Ein Baum T ist eine $(k+1)$-Raupe, wenn nach Löschen aller Blätter von T eine k-Raupe entsteht.*

Aufgabe 34. *Sind 2-Raupen graceful?*

Definition 35. *Ein Baum T mit Wurzel x heißt* grad-regulär*, falls $\forall y, z \in V(T) : \mathrm{dist}(x, y) = \mathrm{dist}(x, z) \Rightarrow \deg(y) = \deg(z)$.*

Beispiel 36. *Jeder Pfad ist grad-regulär. Jeder vollständige binäre Baum ist grad-regulär.*

Satz 37. *Jeder grad-reguläre Baum besitzt ein graceful labelling, bei dem die 0 in der Wurzel steht.*

(Vorlesung vom 2. 5.)

3.5 Minimale Gerüste und Wege

Literatur: [Wes01] Abschnitt 2.3 und 8.2, [Bra94] Kapitel 4.

Gegeben: zshg. Graph $G = (V, E)$, Kantengewichte $c : E \to \mathbb{Q}_{\geq 0}$.

Für $G' \subseteq G$ setzen wir $c(G) = \sum_{e' \in E(G')} c(e')$.

Definition 38. *T ist c-minimales Gerüst von G, falls T Gerüst von G und für alle Gerüste T' von G gilt: $c(T) \leq c(T')$.*

Satz 39. *Für Graphen G, Kostenfunktion c, Menge $U \subseteq V(G)$: für jede Kante e zwischen U und $V(G) \setminus U$ mit minimalem Gewicht gilt: G besitzt ein c-minimales Gerüst, das e enthält.*

Proof. Sei T irgendein minimales Gerüst von G mit $e \notin T$. Dann enthält $T \cup e$ einen Kreis. Auf diesem liegt eine weitere Kante e' zwischen U und $V(G) \setminus U$, so daß $T' = T \cup e \setminus e'$ ein Gerüst von G ist. Es gilt $c(T') \leq c(T)$. □

3.5.1 Kruskal

Algorithmus von Kruskal (1956). Eingabe (G, c).

Wir berechnen eine Folge S_i von spannenden Teilgraphen von G mit $\emptyset = E(S_0) \subset E(S_1) \subset \ldots \subset E(S_n) \subseteq E(G)$. Dabei ist (genau) S_n zusammenhängend.

Jedes $S \subseteq G$ definiert die Menge $\mathrm{cut}(S)$ aller Kanten von G, die zwischen verschiedenen Zusammenhangskomponenten von S verlaufen. Wir setzen $S_{i+1} = S_i \cup e_i$, wobei e_i eine c-minimale Kante aus $\mathrm{cut}(S_i)$ ist. (Beachte: $\mathrm{cut}(S_0) = E(G)$.)

Satz 40. *Für jeden zusammenhängenden Graphen G gilt: es gibt solche Folgen, und für jede so konstruierte Folge gilt: S_n ist c-minimales Gerüst von G.*

Proof. Solange S_k nicht zusammenhängt, ist $\mathrm{cut}(S_k) \neq \emptyset$, die Folge läßt sich also verlängern.

S_n ist nach Konstruktion zusammenhängend, kreisfrei und spannt G auf.

Minimalität folgt aus Satz 39 und den folgenden Betrachtungen zu Matroiden. □

Laufzeit: Kanten nach Gewicht sortieren: $O(|E| \log |E|)$, Komponenten bestimmen und schrittweise vereinigen (union-find-Struktur).

3.6 Greedy-Algorithmen

Eingabe: eine endliche Menge U (Universum), eine Kostenfunktion $c : U \to \mathbb{Q}_{\geq 0}$, ein Mengensystem $S \subseteq 2^U$ mit $\emptyset \in S$.

Ausgabe: eine Menge $M \in S$.

Wir konstruieren Folgen U_i, M_i mit $0 \leq i \leq |U|$ durch:

$M_0 = \emptyset, U_0 = U$.

$x_i \in U_i$ mit $c(x)$ maximal unter $\{c(x) : x \in U_i\}$.

$U_{i+1} = U_i \setminus x_i$, $M_{i+1} =$ wenn $M_i \cup x_i \in S$ dann $M_i \cup x_i$ sonst M_i.

3.7 Matroide

Whitney (1935) für Graphen, MacLane (1936) für geometrische Gitter, van der Waerden (1937) für Vektorräume.

Definition 41. *Ein Mengensystem $S \subseteq 2^U$ heißt erblich, wenn $\forall M \in S, M' \subseteq M : M' \in S$.*

Satz 42. *Für erbliche Mengensysteme $S \subseteq 2^U$ sind die folgenden Eigenschaften äquivalent, wobei $M = \{A : A \in S, \neg \exists B \in S : A \subset B\}$ das System der maximalen S-Mengen bezeichnet:*

1. *(Uniformität) für alle $C \subseteq U$: alle $A \in M, A \subseteq C$ sind gleichgroß.*
2. *(Vergrößerung) wenn $A, B \in S$ und $|A| < |B|$, dann $\exists x \in B \setminus A : A \cup x \in S$.*
3. *(Greedy) Für jede Kostenfunktion $c : U \to \mathbb{Q}_{\geq 0}$ endet jede Greedy-Folge in eine c-optimalen Menge $A \in S$.*

Definition 43. *Jedes erbliche Mengensystem $S \subseteq 2^U$, das eine (und damit alle) der obigen Bedingungen erfüllt, heißt* Matroid

Beispiele:

- für fixierten Graph G: nimm $U = E(G)$, und $M \in S \iff M$ ist kreisfreie Kantenmenge.
- Für Vektorräume U: nimm $M \in S \iff M$ ist linear unabhängig.

Aufgabe 44. *erbliches Mengensystem $S \subseteq 2^U$ ist Matroid $\iff$ (Basis-Tausch) für $A, B \in M$: $\forall x \in A \setminus B \exists y \in B \setminus A : A - x + y \in M$.*

Definition 45. *Der S-Rang* $\operatorname{rank}(E)$ *einer Menge $E \subseteq U$ ist* $\max\{|M| : M \subseteq E, M \in S\}$.

Satz 46. *Wenn $S \subseteq 2^U$ ein Matroid ist, I seine maximalen Mengen, und* rank *seine Rang-Funktion, dann gilt:*

1. *(Einschließung)* $0 \leq \operatorname{rank} A \leq |A|$,

2. $A \in I \iff |A| = \operatorname{rank} A$,

3. *(Monotonie)* $A \subseteq B \Rightarrow \operatorname{rank} A \leq \operatorname{rank} B$,

4. $\operatorname{rank} A \leq \operatorname{rank}(A \cup \{x\}) \leq \operatorname{rank} A + 1$

5. *(Submodularität) für* $X, Y \subseteq U$*:* $\operatorname{rank}(X \cap Y) + \operatorname{rank}(X \cup Y) \leq \operatorname{rank}(X) + \operatorname{rank}(Y)$.

Satz 47. *Wenn eine Funktion* $r : 2^U \to \mathbb{N}$ *die Eigenschaften (Einschließung), (Monotonie), (Submodularität) erfüllt, dann ist* $S = \{A \subseteq U \mid |A| = r(A)\}$ *ein Matroid.*

Aufgabe 48. *In Vektorräumen können wir zu jeder Menge* A *die Menge aller Linearkombinationen von Vektoren aus* A *bilden.*

Definiere allgemeine für Matroide $S \subseteq 2^U$ *einen Hüll-Operator* $\operatorname{hull} : 2^U \to 2^U$, *der diese als Speziallfall enthält. Benutze die* rank*-Funktion von* S.

Postulieren einige Eigenschaften von hull.

3.8 Union-Find-Strukturen

Beim Algorithmus von Kruskal ist es wichtig, schnell zu entscheiden, ob eine Kante zwei verschiedene Komponenten verbindet, oder einen Kreis innerhalb einer Komponente erzeugt.

Das Interface der Struktur ist:

```
data Pool s = ... -- Menge von Komponenten (Mengen)
units :: Set s -> Pool s -- alles Einermengen
find :: Pool s -> (s, s) -> Bool -- beide in gleicher Komponente?
unite :: Pool s -> (s, s) -> Pool s -- verschmelze beide Komponenten.
```

Ein schlechte Lösung wäre `Pool s = Set (Set s)`, denn dann ist `find` aufwendig (man muß den Punkt erst suchen).

Eine Tabelle (FiniteMap) zu verwalten, die jeden Knoten auf die Nummer seiner Komponente abbildet, liefert schnelles `find`, aber beim Verschmelzen von zwei Komponenten hätten wir dann für alle Punkte der einen Komponente diesen Eintrag zu ändern, also langsames `unite`.

Besser ist folgendes: jede Komponente ist ein (gerichteter) Baum, und

```
find p (u, v) = root p u == root p v
```

wobei `root p u` der Wurzelknoten für die Komponente von u ist. Diesen finden wir durch Verfolgen der Zeiger (jedes Knoten zu seinem Vater). Bei `unite p (u,v)` machen wir `root p u` zum neuen Kind von `root p v`.

Die Laufzeit für `unite` ist damit konstant 1, und die von `find` ist proportional zur Höhe der Bäume. Wenn wir immer den kleineren an den größeren hängen (`root p u` an `root p v` oder `root p v` an `root p u`), ist diese aber nur logarithmisch in der Knotenzahl. (Wir müssen dazu in jedem Knoten die Größe des Teilbaums mitführen, dessen Wurzel er ist, und das bei `unite` aktualisieren.)

Man kann weitere Verbesserungen erzielen, wenn man bei jedem `find` die Pfade komprimiert, d. h. von jedem Knoten auf einem Pfad (vom gesuchten Knoten) zur Wurzel den Vater-Zeiger direkt auf die Wurzel setzt.

Einzelheiten dazu [CLR91].

3.9 Minimale Gerüste: Prim

Jede Menge $S \subseteq V(G)$ definiert die Menge $\mathrm{cut}(S)$ aller Kanten von G, die zwischen S und $V(G) \setminus S$ verlaufen.

Das Folgende ist eine genauere Version von Satz 39.

Satz 49. *Für alle Graphen G, Kostenfunktionen $c : E(G) \to \mathbb{Q}_{\geq 0}$, c-minimale Gerüste T von G, Knotenmengen $S \subseteq E(G)$, und Kantenmengen $A \subseteq E(G)$ mit $A \cap \mathrm{cut}(S) = \emptyset$, und c-minimale Kanten $e \in \mathrm{cut}(S)$ gilt: es existiert ein c-minimales Gerüst T' von G, so daß $A \cup \{e\} \subseteq E(T')$.*

Proof. Betrachte $T \cup e$. Dieser Graph enthält einen Kreis. Dieser Kreis enthält eine weitere Kante $e' \in \mathrm{cut}(S)$. Wir wählen $T' = T \cup e \setminus e'$. Wegen $c(e) \leq c(e')$ ist $c(T') \leq c(T)$. Weil T minimal war, ist $c(T') = c(T)$ und auch T' ist minimal. □

Algorithmus von Prim (früher: Jarnik, 1930). Eingabe (G, c).

Wir wählen einen Knoten $v_0 \in G$ und eine Folge T_i von Teilbäumen von G mit $\{v_0\} = T_0 \subset T_1 \subset \ldots \subset T_n \subseteq E$. Dabei ist $V(T_n) = V(G)$.

Wir setzen $T_{i+1} = T_i \cup e_i$, wobei e_i eine c-minimale Kante aus $\mathrm{cut}(T)$ ist.

Satz 50. *Für jeden zusammenhängenden Graphen G gilt: es gibt solche Folgen, und für jede so konstruierte Folge gilt: T_n ist c-minimales Gerüst von G.*

Proof. Solange $V(T_k) \neq V(G)$, ist $\mathrm{cut}(S_k) \neq \emptyset$, die Folge läßt sich also verlängern.

Jedes T_k ist nach Konstruktion ein Baum, und T_n ein Gerüst für G.

Optimalität nach Satz 49. □

Laufzeit: wir benutzen eine Priority-Queue für die Nachbarn von T_i, die nicht in T_i selbst liegen. Wir brauchen $|E|$ mal `insert/update`, und $|V|$ mal `extract`. Genauere Betrachtung später (nach Dijkstra).

Kapitel 4

Wege, Matchings, Flüsse

(Vorlesung 9. 5.)

4.1 Kürzeste Wege von einem Knoten (Dijkstra)

Literatur: [CLR91]

Gegeben sind G und $c : E(G) \to \mathbb{Q}_{\geq 0}$ sowie ein Knoten $s \in G$.

Wir verzichten ganz auf G, indem wir für Nicht-Kanten $xy \notin E(G)$ setzen: $c(xy) = \infty$.

Gesucht ist für jedes $t \in V(G)$ die Länge eines c-minimalen $s - t$-Pfades.

Wir konstruieren eine Folge von Mengen $(S_1, S_2, \dots, S_n)$ und partiellen Funktionen $d_i : V(G) \to \mathbb{Q}_{\geq 0}$.

Start: $S_1 = \{s\}, d_1 = t \mapsto ($ wenn $t = s$ dann 0 sonst $c(st))$.

Ende: $S_n = V(G)$.

Schritt: t_{i+1} ist ein Knoten t in $V(G) \setminus S_i$, für den $d_i(t)$ minimal ist.

Dann $S_{i+1} = S_i \cup \{t_{i+1}\}$ und $d_{i+1} = t \mapsto \min\{d_i(t), d_i(t_{i+1}) + c(t_{i+1}t)\}$.

Satz 51. *Für alle i und $x \in S_i, y \in V(G) \setminus S_i$ gilt $d_i(x) \leq d_i(y)$.*

Proof. Anfang: sicher wahr wegen $d_1(s) = 0$. Schritt: sei $t = t_{i+1} \in S_{i+1} \setminus S_i$. Für $x \in S_i, y \notin S_{i+1}$ gilt: $d_{i+1}(x) = \min(d_i(x), d_i(t) + c(tx)) \leq d_i(x) \leq d_i(t) = \min(d_i(t), d_i(t) + c(ty)) \leq \min(d_i(y), d_i(t) + c(ty)) = d_{i+1}(y)$ Für $x = t, y \notin S_{i+1}$ gilt $d_{i+1}(x) = d_{i+1}(t) = d_i(t) \leq \min(d_i(y), d_i(t) + c(ty))$ □

Satz 52. *Für alle i und alle $x \in S_i$ gilt: $d_i(x) = d_{i+1}(x) = \dots d_n(x)$*

Proof. Für $x \in S_i$ und $t = t_{i+1} \notin S_i$ gilt $d_{i+1}(x) = \min(d_i(x), d_i(t) + c(tx)) \leq d_i(x)$, und nach vorigem Satz $d_i(x) \leq d_i(t)$, also $d_{i+1}(x) = \min(d_i(x), d_i(t) + c(tx)) \geq d_i(x)$.. □

Satz 53. *Für alle i und alle $u \in S_i$: $d_i(u)$ ist die Länge eines minimalen $s - u$-Weges.*

Proof. Falls die Behauptung nicht gilt, sei u der erste Knoten (gezählt in der Reihenfolge der Einfügung in S), für den der Algorithmus eine falsche Weglänge bestimmt, d. h. es gibt einen $s-u$-Pfad mit $c(s\dots u) < d_n(u)$.

Dann gibt es ein i, so daß $u \in S_i \setminus S_{i+1}$, und alle Weglängen $d_n(x)$ für $x \in S_n$ sind korrekt.

Für jeden optimalen $s-u$-Weg gibt es eine Kante xy mit $x \in S_i, y \notin S_i$. Dann $c(s\dots u) = c(s\dots xy\dots u) \geq c(s\dots xy)$. Da x bereits in S_i ist, also vom Algorithmus gewählt wurde, und dieser nach Annahme bis dorthin korrekt ist, gilt $c(s\dots xy) = c(s\dots y) + c(xy) = d_i(x) + c(xy)$. Da nach Wahl von x die Kante xy betrachtet wurde, gilt auch $d_{i+1}(y) = d_i(x) + c(xy)$. Da der Algorithmus aber nicht y, sondern das minimale u wählte, gilt $d_i(u) \leq d_i(y)$, und das kann nicht sein. TODO. □

Laufzeit (vergleiche Prim): Priority-Queue für Schnittkanten, Operationen extract-min, insert, decrease-key.

Bemerkung: wenn wir mitschreiben, „woran die Knoten hängen“, dann erhält S eine Baumstruktur.

Aufgabe: das ist aber im Allgemeinen kein minimales Gerüst (Gegenbeispiel!)

Bemerkung: wenn $c : E \to \{1\}$, dann erhalten wir eine Breitensuche.

4.2 Prioritäts-Warteschlangen

4.2.1 Motivation, Interface

Beim Alg. von Dijkstra müssen wir Mengen von bewerteten Knoten verwalten. Als Interface haben wir

```
data Heap s = ...
empty :: Heap s -- leer
-- Schlüssel mit Wert einfügen
insert :: Heap s -> ( s, v ) -> ( Heap s, Pointer s )
decrease_key :: Heap s -> ( Pointer s , v) -> Heap s -- Wert verringern
extract_min :: Heap s -> ( s, Heap s ) -- ( Minimum und Rest )
```

Wir müssen uns vorstellen, daß wir beim insert einen "Zeiger in den Heap" erhalten, den wir zum späteren Zugriff (decrease) verwenden.

4.2.2 Naive Implementierungen

Wir könnten im einfachsten Fall alle Schlüssel-Wert-Paare in einer ungeordneten Liste aufbewahren. Dann geht `insert` trivial schnell, aber bei `extract_min` brauchen wir Linearzeit (in der Anzahl der Schlüssel).

Eine andere mögliche Implementierung ist eine nach Werten geordnete Liste. Dann geht `extract_min` trivial schnell, aber bei `insert` brauchen wir Linearzeit.

Weiter könnten wir einen balancierten Suchbaum nehmen (z. B. AVL-Baum), dann dauern `insert` und `find` beide logarithmische Zeit, da wir im Allgemeinen etwas Zeit zum Rebalancieren brauchen.

Dijkstras Algorithmus führt `insert` und `decrease_key` $|E(G)|$ mal aus (für jede Kante einmal), aber `extract_min` nur $|V(G)|$ mal. Wenn der Graph viele Kanten hat, können wir Zeit sparen, indem wir `decrease_key` auf Kosten von `extract_min` beschleunigen.

Ein wichtiger Gedanke ist nun der, daß wir die Paare gar nicht komplett sortieren müssen. Wir verwenden eine Baum mit *Heap-Ordnung*, d. h. in jedem Teilbaum hat die jeweilige Wurzel den kleinsten Wert (unter allen Knoten des Teilbaums).

(Vorlesung 16. 5.)

4.2.3 Binomial-Heaps

Literatur: [CLR91], Kapitel 20.

Wir verwenden *Binomial-Bäume* B_k.

Definition 54. *Die Wurzel eines* B_k *hat genau* k *Kinder, und zwar* $[B_0, B_1, \dots, B_{k-1}]$.

Beachte: Induktionsanfang: $k = 0$: Die Kinderliste ist leer (der Baum besteht nur aus der Wurzel)

Folgerung: Wenn man an die Wurzel eines B_k einen B_k (als neues Kind) anhängt, entsteht ein B_{k+1}.

Satz 55. *Jeder* B_k *hat genau* 2^k *Knoten.*

Wenn man das Anhängen schichtweise untersucht, erhält man:

Satz 56. *Jeder* B_n *hat in Schicht* k *(von der Wurzel aus gezählt) genau* $\binom{n}{k}$ *Knoten.*

daher kommt der Name.

Ein *Binomial-Heap* ist eine (der Größe nach geordnete) Liste von *paarweise verschieden* großen Binomial-Bäumen. (Beachte, daß jede Kinderliste eines Binomialbaums selbst ein Binomialheap ist.)

Satz 57. *Ein Binomial-Heap mit* n *Knoten besteht aus* $\leq \log_2 n$ *Bäumen, jeder vom Grad und Tiefe* $\leq \log_2 n$.

Vereinigung von zwei Binomial-Heaps entspricht binärer Addition mit Übertrag (Einfügen entspricht binärem Zählen). Laufzeit: proportional zur Länge der Wurzelliste, also $\log_2 n$.

`decrease_key` geht wie beim übliche Heapsort, indem der Knoten den Weg zur Wurzel hochrutscht. (Laufzeit: proportional zur Tiefe, also $\log_2 n$.)

4.2.4 Fibonacci-Heaps

Literatur: [CLR91], Kapitel 21

Hier ist nun die Stelle zum Optimieren: In der Anwendung (Dijkstras Algorithmus) ist ja von vornherein klar, daß wir $|E|$ mal `insert` und `decrease` ausführen, sowie $|V|$ mal `extract`. Wenn $|E|$ im Verhältnis zu $|V|$ groß ist (d. h. quadratisch), dann lohnt es sich, `decrease` zu verbilligen, und `extract` etwas zu verteuern.

Die Idee ist: man vermeidet bei `decrease` das Hoch-Rutschen, und *schneidet* stattdessen den Knoten aus dem Baum, und hängt ihn (den bei ihm beginnenden Teilbaum) direkt in die Wurzelliste. Erst beim nächsten `extract_min` wird in der Wurzelliste aufgeräumt (wie bei Binomial-Heaps: solange wir zwei Wurzeln vom gleichen Grad finden, hängen wir die eine als Kind in die andere).

Durch viele Schnitte kann die Wurzelliste aber inzwischen sehr lang geworden sein, wir müssen uns davon überzeugen, daß das selten genug vorkommt, und daß die Bäume nicht ganz beliebig unbalanciert werden.

Wir verlangen, daß durch Abschneiden ein Knoten x höchstens ein Kind verlieren darf. Wenn das zweite Kind von x entfernt werden soll, dann tun wir das, müssen aber danach x selbst aus der Kinderliste seines Vaters entfernen, usw. bis evtl. zur Wurzel.

Zu diesem Zweck gibt es in jedem Knoten eine Markierung (ein Bit), die anzeigt, ob bereits ein Kind entfernt wurde. Die Markierung von x wird *gelöscht*, wenn x selbst abgeschnitten wird, d. h. in die Wurzelliste wandert.

Definition 58. *Ein Fibonacci-Heap (kurz F-Heap) ist eine Folge von F-Bäumen. Ein F-Baum ist ein heap-geordneter Baum mit einem Markierungs-Bit in jedem Knoten.*

Die Analyse der Laufzeit führen wir *amortisiert*, d. h. nicht für einzelne Operationen, sondern für Folgen von Operationen. Wir benutzen dazu eine *Potential-Funktion* p (von Baum nach nichtnegativer Zahl), und setzen:

Definition 59. amortisierte Kosten *einer Operation* = tatsächliche Kosten + *Potential* nach *Operation* − *Potential* vor *Operation.*

Damit haben wir

Satz 60. tatsächliche *Kosten einer Folgen von Operationen* = *Summe der* amortisierten *Einzelkosten* + *Potential vor Folge* − *Potential nach Folge.*

Wenn das Potential am Anfang 0 (oder beschränkt) ist, haben wir damit eine Abschätzung für die tatsächlichen Kosten.

Definition 61. *Die Potentialfunktion für F-Heaps ist: Länge der Wurzelliste* $+2\cdot$ *Anzahl der markierten Knoten.*

Die amortisierten Kosten der Operationen sind:

`insert`: einen neuen Baum der Größe 1 in die Wurzelliste hängen: tatsächliche Kosten: 1 (durch Verbiegen von Zeigern, wir benutzen eine zweiseitig, ringförmig verkettete Liste), Potential: steigt um 1, amortisierte Kosten also: 2 (konstant)

`decrease`: rekursives Abschneiden, wie oben beschrieben. tatsächliche Kosten: Anzahl s der Schnitte. Potentialdifferenz: die Wurzelliste wird um s länger, die markierten Knoten um $s-1$ weniger, also: $s-2(s-1)$. amortisierte Kosten: $s+s-2s+2$ (konstant).

`extract`: zunächst das Minimum finden. das geht in konstanter Zeit, wenn wir zusätzlich zur Wurzelliste noch einen Zeiger auf die kleinste Wurzel mitführen. Den können wir immer in konstanter Zeit aktualisieren. Dann Kinder des Minimums mit restlicher Wurzelliste verketten (konstante Zeit, Zeiger-Verbiegen). Dann in neuer Wurzelliste aufräumen (Wurzelgrade disjunkt machen). tatsächliche Kosten: Länge der neuen Wurzelliste = Länge l der alten Wurzelliste + Anzahl der Kinder des Minimums $\leq l + D(n)$, wobei: $D(n)$ = Maximalgrad der Wurzel eines F-Baums mit n Knoten. Nach Aufräumen hat Wurzelliste die Länge $D(n)$, denn alle Wurzeln haben verschiedenen Grad (sonst wäre Aufräumen noch nicht fertig), und größere Wurzelgrade können nicht vorkommen (nach Definition von $D(n)$.) Potentialdifferenz: da wir an den Markierungen nichts ändern: $D(n) - (l + D(n))$. Amortisierte Kosten: $l + D(n) + D(n) - (l + D(n)) = D(n)$.

Es bleibt also die Aufgabe, das Wachstum der Funktion $D(\cdot)$ abzuschätzen. (Ziel: $D \in O(\log)$)

Satz 62. *Für jeden Knoten x eines F-Heaps mit $\deg(x) = d$: Sei $[y_1, y_2, \ldots, y_d]$ die Liste seiner Kinder, geordnet nach dem Zeitpunkt ihres Anhängens an x. Es gilt: $\deg(y_k) \geq k - 2$.*

Proof. Im Binomialheap (d. h. F-Heap ohne `decrease`) wäre $\deg(y_k) = k - 1$, aber hier kann noch ein Kind abgeschnitten werden (aber nicht mehr). □

Satz 63. *Für $\text{size}(d)$ = minimale Knotenzahl eines F-Heaps mit Wurzelgrad d gilt: $\text{size}(d) \geq 1 + \text{size}(0) + \text{size}(0) + \text{size}(1) + \ldots + \text{size}(d-2)$.*

Proof. 1 für die Wurzel, dann die vorigen Abschätzungen für die Kinder. □

Satz 64. $\text{size}(d) \geq F_{d+2}$*, wobei F die Fibonacci-Folge $F_0 = 0, F_1 = 1, F_{k+2} = F_k + F_{k+1}$.*

Beweis: Übung.

Satz 65. $D(n) \in O(\log(n))$

Beweis: Übung. Benutze $F_k = (q^k - q^{-k})/\sqrt{5}$ mit $q = (1 + \sqrt{5})/2$ (Goldener Schnitt, Turm des Alten Rathauses, usf. Es gibt tatsächlich eine ganze Zeitschrift *Fibonacci Quarterly* nur zu diesem Thema.)

Folgerung 66. *Algorithmus von Dijkstra erreicht mit Fibonacci-Heaps eine Laufzeit von $O(|E| + |V| \log |V|)$.*

Aufgabe 67. *Vergleiche mit Laufzeit des Alg. mit Binomialheaps.*

Wir haben den Wurzelgrad von F-Heaps nach oben abgeschätzt. Das verhindert flache, breite Bäume. In die andere Richtung folgt daraus gar nichts:

Aufgabe 68. *F-Bäume können sehr dünn und tief sein: Konstruiere eine Folgen von Operationen, die aus einem Einer-F-Heap einen F-Heap mit n Knoten und linearer Tiefe erzeugt.*

(Vorlesung 23. 5.)

4.3 Kürzeste Wege für alle Knotenpaare (Floyd/Warshall)

Eingabe: ein Graph $G = (V, E)$ und Kantengewichtsfunktion $c : E \to \mathbb{Q}_{\geq 0}$,

Ausgabe: Funktion d mit $d(uv)$ = Länge eines c-minimalen $u - v$-Weges.

Wir konstruieren eine Folge von Funktionen (Matrizen) $d_i : V^2 \to \mathbb{Q}_{\geq 0}$ mit

(Anfang) $d_0 = (u, v) \mapsto$ wenn $u = v$ dann 0 sonst wenn $uv \in E$ dann $c(uv)$ sonst ∞

wähle eine beliebige Anordnung $v_1, v_2, \ldots, v_n$ von V.

(Schritt) Für $1 \leq i \leq |V| : d_i = (u, w) \mapsto \min(d_{i-1}(u, w), d_{i-1}(u, v_i) + d_{i-1}(v_i, w))$

Satz 69. *$d_{|V|}$ ist die gesuchten Funktion d.*

Proof. Die Invariante ist: für jedes i ist $d_i(u, w)$ die Länge eines c-minimalen Weges der Form $u - v_1 - v_2 - \ldots - v_k - w$ mit $v_1, v_2, \ldots, v_k \leq i$. (Falls $k = 0$, gibt es keine Hilfspunkte.) □

Laufzeit $O(|V|^3)$: wir müssen $|V|$-mal eine Matrix der Größe $|V|^2$ ausrechnen.

Aufgabe 70. *Vergleiche diese Laufzeit mit dem Verfahren, für jeden Startknoten den Algorithmus von Dijkstra auszuführen.*

4.3.1 Alg. von Kleene (Automaten-Analyse)

Vergleiche Floyd/Warshall auch mit Algorithmus von Kleene zur Auomaten-Analyse: gegeben ist endlicher Automat A, gesucht ist regulärer Ausdruck für $L(A)$.

Die Programmstruktur ist die gleiche, aber in diesem Fall benutzen wir nicht + und min (für Zahlen), sondern $\cdot$ und $\cup$ (für Mengen von Wörtern), und müssen auch Schleifen berücksichtigen:

$$L_i(p, q) = L_{i-1}(p, q) \cup L_{i-1}(p, i) \cdot L_{i-1}(i, i)^* \cdot L_{i-1}(i, q)$$

4.4 Flüsse

4.4.1 Definitionen

Literatur: [Bra94], Abschnitt 6.1, 6.2; [Bol98], Abschnitt III.1

Gegeben gerichteter Graph G, Quelle (source) $s \in V(G)$, Senke (target) $t \in V(G)$, Kapazitätsfunktion $c : E(G) \to \mathbb{Q}_{\geq 0}$.

Definition 71. *Für eine Abbildung* $f : E \to \mathbb{Q}_{\geq 0}$ *bezeichnen wir*

- *eingehender Strom in* x*:* $f^+(x) = \sum\{f(p, x) \mid p \in V, px \in E\}$,
- *von* x *ausgehender Strom:* $f^-(x) = \sum\{f(x, q) \mid q \in V, xq \in E\}$
- *in* x *überschüssiger (bzw. fehlender) Strom:* $f(x) = f^+(x) - f^-(x)$

Definition 72. *Ein* Fluß *in* G *ist eine Abbildung* $f : E(G) \to \mathbb{Q}_{\geq 0}$ *mit*

- *(Kapazität beachten)* $\forall xy \in E : 0 \leq f(x, y) \leq c(x, y)$
- *(lokale Fluß-Erhaltung)* $\forall x \in V \setminus \{s, t\} : f(x) = 0$.

Der Wert eines Flusses ist $f(s)$.

Definition 73. *Ein* Schnitt *in* G *ist ein Mengenpaar* $(S, \overline{S})$ *mit* $s \in S, t \in \overline{S} = V \setminus S$. *Wir schreiben* $(x, y) \in (S, \overline{S})$ *für* $x \in S, y \in \overline{S}, xy \in E$.

Für eine Funktion $f : E \to \mathbb{Q}_{\geq 0}$ *setzen wir* $f(S, \overline{S}) = \sum\{f(x, y) \mid (x, y) \in (S, \overline{S})\}$.

Insbesondere heißt $c(S, \overline{S})$ *die* Kapazität *von* S.

Der Wert des Flusses ist an der Quelle definiert, kann aber an *jedem* Schnitt festgestellt werden:

Satz 74. *Für jeden Fluß* f *mit Wert* v*, und jeden Schnitt* S *gilt:*

$$v = \sum\{f(e) \mid e \in (S, \overline{S})\} - \sum\{f(e) \mid e \in (\overline{S}, S)\}.$$

Proof. Addiere die Gleichungen $\forall x \in \overline{S} \setminus t : f(x) = 0$ zur Gleichung $f(s) = v$, und kürze. $\square$

4.4.2 Min-Cut-Max-Flow

Satz 75. *Für fixiertes G, s, t, c: Der maximale Wert eines Flusses ist die minimale Kapazität eines Schnittes.*

Proof. Es ist klar, daß der maximale Fluß nicht größer sein kann als die Kapazität eines beliebigen Schnittes. Zu zeigen ist, daß zu jedem maximalen Fluß f mit Wert v tatsächlich ein Schnitt S mit Kapazität v existiert.

Wir definieren die Menge S als die kleinste Menge mit den Eigenschaften:

- $s \in S$,
- (xy ist nützliche Vorwärtskante) wenn $x \in S$ und $f(x, y) < c(x, y)$, dann $y \in S$,
- (xy ist nützliche Rückwärtskante) wenn $y \in S$ und $0 < f(x, y)$, dann $x \in S$.

Wir zeigen, daß das ein Schnitt ist, d. h. $t \notin S$: falls $t \in S$, dann gibt es einen Weg (aus nützlichen Vorwärts- und Rückwärtskanten) von s nach t.

Wir bestimmen

$$\begin{aligned} m^+ &= \min\{c(x, y) - f(x, y) \mid xy \text{ ist nützliche Vorwärtskante}\} \\ m^- &= \min\{f(x, y) \mid xy \text{ ist nützliche Rückwärtskante}\} \\ m &= \min(m^+, m^-) \end{aligned}$$

Nach Definition von S ist m eine positive Zahl (als Minimum über eine endliche Menge von positiven Zahlen).

Wir können nun zu f auf jeder nützlichen Vorwärtskante m addieren, und auf jeder nützlichen Rückwartskante m subtrahieren. Dabei entsteht wieder ein Fluß f', mit um m größerem Wert, im Widerspruch zur Annahme, daß f maximal war. □

Aus dem Beweis folgt also nicht nur der Satz, sondern sogar eine Konstruktion für den maximalen Fluß: bestimme einen vergrößernden Weg, erhöhe den Fluß, iteriere solange wie nötig. Das ist der Algorithmus von Ford und Fulkerson, wobei aber noch nicht klar ist, wie wir solch einen Weg finden, und wie oft wir iterieren müssen.

Aufgabe 76. *(aus [Bra94]) $G = (V, E)$ mit $V = \{s, a, b, t\}$, Kapazitäten $c : sa \mapsto 100, sb \mapsto 100, ab \mapsto 1, at \mapsto 100, bt \mapsto 100$. Wir beginnen mit Fluß konstant 0.*

a) Benutzen $s-a-b-t$ als vergrößernden Weg (aus lauter Vorwärtskanten) Welcher Fluß entsteht?

b) Danach ist $s-b-a-t$ ein vergrößernden Weg (mit $b-a$ als nützlicher Rückwärtskante!). Welcher Fluß entsteht?

c) Dann immer so weiter: abwechselnd $s - a - b - t$ und $s - b - a - t$ benutzen. Wie lange dauert das?

d) Finde ein ähnliches Beispiel mit irrationalen Kapazitäten, für das eine unendliche *Folge von Flußvergrößerungen existiert.*

4.4.3 Kapazitäts-Schranken für Knoten

Literatur: [Bol98], Abschnitt III.3

In einigen Fällen haben wir Kapazitäts-Schranken nicht für Kanten, sondern für Knoten.

Ist eventuell nützlich für Havannah, dort setzen wir die Steine eben auf Knoten, und nicht auf Kanten.

Eingabe: gerichteter Graph G, Knoten $s, t \in V(G)$, Kapazitäten $c : V(G) \to \mathbb{Q}_{\geq 0}$.

Gesucht: maximaler Fluß mit Eigenschaft: $\forall x \in V : f(x) \leq c(x)$.

Wir konstruieren aus G einen Graphen G', in dem wir einen kanten-beschränkten Fluß suchen, der die Lösung des knotenbeschränkten Systems enthält.

$G' = (V', E')$ mit $V' = V \times \{1, 2\}$ und

$$E' = \{(x, 2)(y, 1)(x, y) \in E$$

sowie Kapazitäten $c' : E' \to \mathbb{Q}_{\geq 0}$ mit $c'((x, 2)(y, 1)) = \infty, c'((x, 1), (x, 2)) = c(x)$.

Es ist dann klar, daß jedem c'-kanten-beschränktem Fluß f' in G' ein c-knoten-beschränkter Fluß f in G entspricht, und umgekehrt.

Wir können analog zu den bereits betrachteten Kanten-Schnitten $(S, V \setminus S)$ auch *Knoten-Schnitte* betrachten. Das ist eine Menge $S \subseteq V$, so daß in $G \setminus S$ kein positiver $s - t$-Fluß existiert. Die Kapazität eines Knoten-Schnittes S ist $\sum_{x \in S} c(x)$, und wir haben ganz analog:

Satz 77. *Der maximale Wert eines knoten-beschränkten Flusses ist die minimale Kapazität eines Knoten-Schnittes.*

(Vorlesung 30. 5.)

4.4.4 Laufzeiten von Fluß-Algorithmen

Wir haben im Beispiel gesehen, daß bei ungünstiger Wahl des vergrößernden Pfades der Ford-Fulkerson-Algorithmus eventuell sehr lange braucht. Immerhin hält er.

Satz 78. *Wenn alle Kapazitäten ganzzahlig sind, dann gibt es einen überall ganzzahligen Maximalfluß, und jede Implementierung des Ford-Fulkerson-Algorithmus hält nach endlich vielen Schritten.*

Proof. Der Alg. beginnt mit einem ganzzahligen Fluß (nämlich 0). Auf jedem vergrößernden Weg ist die vergrößernde Differenz ganzzahlig und positiv, also wenigstens 1. Da alle Kapazitäten endlich sind, gibt es eine obere Schranke C für den Fluß. Nach höchstens C Schritten hält der Algorithmus (mit einem ganzzahligen Fluß, der ein Maximalfluß ist). □

Offenbar war es im Beispiel ungünstig, lange vergrößernde Wege zu wählen.

Algorithmus von Edmonds/Karp: wähle jeweils eine kürzesten vergrößernden Weg (bestimme diese mit Breitensuche).

Dieser Algorithmus erreicht Polynomialzeit (in der Knotenzahl).

Algorithmus von Dinitz: bestimme jeweils gleichzeitig eine Menge von kürzesten vergrößernden Wegen.

4.4.5 Satz von Menger

Definition 79. *Eine Knotenmenge C* trennt *Knoten s von t in G, falls s und t in verschiedenen Komponenten von $G - C$ liegen.*

Definition 80. *Zwei Wege von s nach t heißen* unabhängig*, falls sie außer s und t keine gemeinsamen Knoten enthalten.*

Satz 81. *(Menger) In jedem Graph G, für Knoten s, t: gilt: die kleinste Größe einer Menge, die s von t trennt, ist die größte Anzahl von unabhängigen $s - t$-Wegen.*

Proof. Wir konstruieren ein Netzwerk mit Knotenkapazitäten 1 überall (außer in s und t, dort unbeschränkt). Dann gilt: Schnitt im Netzwerk $\iff$ Knotenmenge, die s von t trennt; Fluß mit Wert $k \iff k$ unabhängige $s - t$-Pfade. Der Satz folgt damit aus Min-Cut-Max-Flow-Theorem. □

4.5 Matchings

Definition 82. *Ein* Matching *(eine Paarung) in G ist eine Menge $M \subseteq E(G)$ von* unabhängigen Kanten *(d. h. ohne gemeinsame Knoten)*

4.5.1 Matchings in Paaren Graphen

Definition 83. *Ein Graph $G = (V, E)$ heißt* bipartit *(paar), falls es eine disjunkte Zerlegung $V = V_1 \cup V_2$ gibt mit $E \subseteq V_1 \times V_2$. Wir schreiben dann $G = (V_1, V_2, E)$.*

D. h. wir partitionieren die Knoten in zwei Klassen, alle Kanten verlaufen zwischen den Klassen, keine innerhalb einer Klasse.

Definition 84. *Ein Matching M in $G = (V_1, V_2, E)$ heißt* vollständig von V_1 nach V_2, *wenn $\forall x \in V_1 \exists y \in V_2 : xy \in M$.*

Satz 85. *(Hall) $G = (V_1, V_2, E)$ besitzt genau dann ein vollständiges Matching von V_1 nach V_2, wenn $\forall A \subseteq V_1 : |A| \leq |G(A)|$, wobei $G(A) = \{y \mid x \in A, xy \in E(G)\}$.*

Proof. Die Bedingung ist offensichtlich notwendig. Zu zeigen ist, daß sie auch hinreicht.

Wir konstruieren aus G einen Graphen mit zwei neuen Knoten s, t, so daß s mit allen V_1 verbunden ist, und alle V_2 mit t. Ein vollständiges Matching entspricht einer Menge von $|V_1|$ unabhängigen $s - t$-Wegen.

Falls G jedoch kein vollst. Matching besitzt, dann gibt es nach Satz von Menger ein trennende Knotenmenge mit $< |V_1|$ Knoten. Diese Menge besteht aus $A_1 \subseteq V_1$ und $A_2 \subseteq V_2$. Also $|A_1| + |A_2| < |V_1|$.

Wir bezeichnen die Komplemente so: $B_i = V_i \setminus A_i$. Es gibt also in G keine Kante von B_1 nach B_2, deswegen $G(B_1) \subseteq A_2$. Also $|G(B_1)| \leq |A_2| < |V_1| - |A_1| = |B_1|$ im Widerspruch zur Annahme. □

(Vorlesung 6. 6.)

4.5.2 Tuttes 1-Faktor-Satz

Nun wieder zu beliebigen (nicht notwendig paaren) Graphen.

Definition 86. *Ein k-Faktor in G ist ein aufspannender k-regulärer Teilgraph.*

Die Kanten eines 1-Faktors bilden also ein Matching, das jeden Knoten abdeckt.

Definition 87. *Wir nennen einen Graphen (un)gerade, wenn er (un)gerade Knoten-Anzahln besitzt; und bezeichnen mit $q(G)$ die Anzahl der ungeraden Komponenten von G.*

Satz 88. *(Tutte) G besitzt genau dann einen 1-Faktor, wenn für alle $S \subseteq V$ gilt: $q(G-S) \leq |S|$.*

Aufgabe 89. *Zeige, daß die Bedingung notwendig ist. Betrachte dazu die Lage der Kanten eines 1-Faktors. Fallunterscheidung nach geraden und ungeraden Komponenten von $G-S$.*

Proof. Zu zeigen ist noch das Hinreichen.

Dazu sei G ein Graph, der $\forall S \subseteq V : q(G-S) \leq |S|$ erfüllt.

Wir wählen S_0 als eine maximale (bzgl. Inklusion) unter allen Mengen, für die $q(G-S_0) = |S_0|$ gilt.

Solche Mengen existieren tatsächlich, beispielsweise Einermengen von Knoten (Übung).

Der Graph $G - S_0$ hat die ungeraden Komponenten $C_1, \dots, C_m$, und die geraden Komponenten $D_1, \dots, D_k$. Wir zeigen:

- Jedes D_i besitzt einen 1-Faktor, denn für alle $S \subseteq V(D_i)$ gilt $q(D_i - S) \leq |S|$ und wegen $|D_i| < |G|$ existiert Faktor nach Induktion.
- Für jedes C_i: für jedes $c \in C_i$: $C_i - c$ hat einen 1-Faktor: Indirekt: Wegen $|C_i - c| < |G|$ gilt nach Induktion: Wenn $C_i - c$ keinen 1-Faktor hat, dann $\exists S \subseteq V(C_i - c) : q(C_i - c - S) > |S|$. Man zeigt sogar $\geq |S| + 2$ und damit gilt für $S' = S \cup x \cup S_0$, daß $q(G - S') = |S'|$. Das ist Widerspruch zur Maximalität von S_0.
- G enthält m unabhängige Kanten (jeweils von $c_i \in C_i$ nach $s_i \in S_0$): Betrachte bipartiten Graphen $\Gamma = (\{1, \dots, m\}, S_0, \{is \mid \exists c \in C_i, s \in S_0 : cs \in E(G))$. Die Bedingung des Satzes von Hall sind erfüllt, denn $\forall A \subseteq \{1, \dots, m\}$ gilt $|A| \leq q(G - \Gamma(A)) \leq |\Gamma(A)|$.

Insgesamt können wir uns daraus einen 1-Faktor für G basteln. □

4.5.3 Satz von Petersen

Definition 90. *G heißt k-kanten(bzw. -knoten)-zusammenhängend (für $k > 1$), wenn nach Löschen von beliebigen $k-1$ Kanten (bzw. Knoten) immer ein zusammenhängender Graph entsteht.*

Aufgabe 91. *Es gibt Graphen, die k-Kanten-zshgd, aber nicht k-Knoten-zshgd sind.*

Die andere Richtung?

Definition 92. *Ein Graph heißt d-regulär, wenn* jeder *Knoten Grad d hat.*

Definition 93. *Der Kneser-Graph K_k^n hat als Knoten alle k-elementigen Teilmengen von $\{1, 2, \ldots, n\}$, und eine Kante AB genau dann, wenn $A \cap B = \emptyset$.*

Aufgabe 94. *Jeder K_k^n ist regulär. Welcher Grad?*

Definition 95. *Der Petersen-Graph ist K_2^5.*

Satz 96. *(Petersen) Jeder 2-kanten-zshg. 3-reguläre Graph enthält einen 1-Faktor.*

Aufgabe 97. *Finde einen 1-Faktor im Petersen-Graphen.*

Proof. Man rechnet nach, daß der Satz von Tutte anwendbar ist.

Betrachte $S \subseteq V(G)$, und eine (von m) ungerade Komponente C von $G-S$. Weil G 2-kanten-zshg, gehen wenigstens zwei Kanten von S nach C. Weil C ungerade und G 3-regulär, gehen sogar wenigstens 3 Kanten von S nach C (nachrechnen).

Jetzt zählen wir die Anzahl z *aller* Kanten von S nach nach den m ungeraden Komponenten. Wir haben $z \geq 3m$. Weil G 3-regulär, gilt $z \leq 3|S|$. Damit $q(G-S) = m \leq |S|$. □

Aufgabe 98. *Finde einen zshgd. 3-regulären Graphen ohne 1-Faktor.*

Kapitel 5

Färbungen und Perfekte Graphen

5.1 Knotenfärbungen

Definition 99. *Eine k-Knoten-Färbung von G ist eine Abbildung $c : V(G) \to \{1, 2, \dots, k\}$ mit $\forall xy \in E(G) : c(x) \neq c(y)$.*

Die chromatische Zahl $\chi(G)$ ist das kleinste k, für das G eine k-Färbung besitzt.

5.1.1 Farben und Cliquen: Konstruktion von Mycielski

Definition 100. *Die Cliquenzahl $\omega(G)$ ist die maximale Größe einer Clique in G*

Satz 101. $\omega(G) \leq \chi(G)$.

Das ist trivial (alle Knoten einer Clique müssen paarweise verschiedene Farben erhalten). Interessant ist:

Satz 102. *Für jedes $k \geq 2$ gibt es G_k, so daß $\omega(G_k) = 2$ und $\chi(G_k) = k$*

D. h. es gibt sogar dreiecksfreie Graphen mit beliebig hoher chromatischer Zahl

Aufgabe 103. *Finde solche. C_5 erreicht $\chi = 3$, wer schafft $\chi = 4$?*

Definition 104 (Mycielski). *Zu gegebenem Graphen $G = (V, E)$ konstruieren wir $G_M = (V_M, E_M)$ mit $V_M = V \times \{A, I\} \cup \{c\}$ (zwei Kopien von V und ein neuer Knoten v) sowie*

$$\begin{aligned} E_M &= \{(x, A)(y, A) \mid xy \in E\} \\ &\cup \{(x, A)(y, I) \mid xy \in E\} \\ &\cup \{(y, I)c \mid y \in V\} \end{aligned}$$

Beispiel 105. $(K_2)_M = C_5$

Definition 106. *Der Grötzsch-Graph ist $(C_5)_M$.*

Satz 107. *Wenn $K_3 \not\subseteq G$, dann $K_3 \not\subseteq G_M$.*

Proof. Falls es doch ein Dreieck in G_M gibt, dann kann keine seiner Ecken $= c$ sein, denn c ist nur mit $V \times \{I\}$ verbunden, aber diese untereinander nicht. Also liegt der K_3 in $V \times \{A, I\}$. Es können nicht alle drei Ecken in $V \times \{A\}$ liegen, denn dann wäre $K_3 \subseteq G$, im Widerspruch zur Voraussetzung (G dreiecksfrei). Für ein K_3 mit Ecken $(x, A), (y, A), (z, I)$, also mit Kanten $(x, A)(y, A), (x, A)(z, I), (y, A)(z, I)$ wären nach Konstruktion auch xy, xz, yz Kanten von G, im Widerspruch zur Voraussetzung. Mehr als eine Ecke des Dreiecks kann aber nicht in $V \times \{I\}$ liegen, denn die Menge $V \times \{I\}$ ist unabhängig in G_M. □

Satz 108. *Wenn $\chi(G) = k$, dann $\chi(G_M) = k + 1$.*

Proof. Aus einer k-Färbung f von G konstruiert man leicht eine $(k + 1)-$Färbung f_M von G_M:

$$f_M(x, A) = f_M(x, I) = f(x), \quad f(c) = k + 1.$$

Zu zeigen ist, daß aus einer $(k + 1)-$Färbung f_M von G_M eine k-Färbung f von G rekonstruierbar ist. O. B. d. A. ist $f_M(c) = k + 1$. Dann gilt $f_M(V \times \{I\}) \subseteq \{1, 2, \dots, k\}$, und deswegen ist die so definierte Funktion

$$f(x) = \text{ wenn } f_M(x, A) = k + 1 \text{ dann } f_M(x, I) \text{ sonst } f_M(x, A)$$

tatsächlich eine Abbildung in $\{1, 2, \dots, k\}$. Angenommen, es gibt $xy \in E$ mit $f(x) = f(y)$. Wenn $f(x) = f_M(x, A)$ und $f(y) = f_M(y, A)$, dann war bereits f_M keine korrekte Färbung von G_M. Falls $f(x) = f_M(x, I)$, dann ist $f_M(x, A) = k+1$. Damit muß $f_M(y, A) \neq k + 1$ gelten, also $f(y) = f_M(y, A)$. Die Kante $(x, I)(y, A)$ ist aber in G_M, und f_M ist eine korrekte G_M-Färbung, also $f_M(x, I) \neq f_M(y, A)$. □

Beispiel 109. *Der Grötzsch-Graph hat Cliquenzahl 2 (keine Dreiecke) und chromatische Zahl 4.*

Der Satz zeigt, daß man auch ohne Dreicke beliebig hohe chromatische Zahl erzwingen kann. Das kann man noch weiter treiben:

Definition 110. *Die* Taillenweite girth(G) *ist die Länge eines kürzesten Kreises in G.*

Beispiel 111. $K_3 \notin G \iff \text{girth}(G) \geq 4$.

Satz 112 (Erdös). *Zu alle Zahlen g, c gibt es G mit* $\text{girth}(G) \geq g, \chi(G) \geq c$.

Aufgabe 113. *Konstruiere ein Beispiel mit $g = 5, c = 4$.*

Der Parameter girth taucht auch in diesem Zusammenhang auf:

Definition 114. *Ein (r, g)-Käfig (*cage*) ist ein r-regulärer Graph G mit* $\text{girth}(G) =$ *g und minimaler Knotenzahl.*

Beispiel 115. *Der Petersen-Graph ist ein $(3, 5)$-Käfig.*

Es sind nur wenige Käfige explizit bekannt, siehe z. B. http://mathworld.wolfram.com/CageGraph.html.

5.1.2 Farben und Grade: Satz von Brooks

Definition 116. *Der Max(Min)imalgrad von G heißt $\Delta(G)$ ($\delta(G)$).*

Satz 117. $\chi(G) \leq \Delta(G) + 1$.

Proof. Wir ordnen die Knoten in irgendeiner Reihenfolge $x_1, x_2, \ldots, x_n$ und färben dann greedy: Sei N die Menge der Nachbarn von x_i. Einige davon (die mit kleinerem Index) sind bereits gefärbt. x_i bekommt die kleinste Farbe, die unter diesen Farben nicht vorkommt. Da jeder $\leq \Delta$ Vorgänger hat, benutzen wir höchstens die Farben bis $\Delta + 1$. □

(Wir könnten statt „die kleinste“ auch „irgendeine“ sagen, aber die gewählte Formulierung hat den Vorteil, daß sich ein deterministischer Algorithmus ergibt.)

Satz 118. *Wenn G zusammenhängend und* nicht *regulär, dann $\chi(G) \leq \Delta(G)$.*

Proof. Ähnliche Idee wie oben, aber mit geeignet gewählter Reihenfolge. Wir stellen sicher,

- daß $x_1, x_2, \ldots, x_{n-1}$ noch wenigstens einen Nachbarn mit größerem Index haben,
- daß'x_n echt weniger als Δ Nachbarn besitzt.

Dazu wählen wir x_n als einen Knoten mit $\deg(x_n) < \Delta$ (so ein Knoten existiert, weil G nicht regulär ist). Nun wählen wir irgendeinen aufspannenden Baum T für G (existiert, weil G zusammenhängend ist). Die Knoten-Reihenfolge (für die Färbung) nun so, daß x_n maximal und für jede Kante $xy \in E(T)$ gilt: $x < y$. (Mit anderen Worten: der Baum T definiert eine Halbordnung, und wir wählen eine Linearisierung.) Dann geht (außer von der Wurzel) immer eine Kante „nach oben “. □

Satz 119. *(Brooks) Wenn G zusammenhängend, und keine Clique und kein ungerader Kreis, dann $\chi(G) \leq \Delta(G)$.*

Proof. Nach dem vorigen ist die Behauptung nur noch für zusammenhängende reguläre Graphen zu zeigen. Die Idee ist wieder die Greedy-Färbung entlang einer geeigneten Reihenfolge.

Wenn G einen cut-vertex besitzt (einen Knoten x, so daß $G \setminus x$ in mehrere Komponenten zerfällt), dann wählen wir wie eben einen spannenden Baum mit x als Wurzel. Allerdings hat x schließlich eventuell Δ bereits gefärbte Vorgänger $y_1, \ldots, y_k$. Diese befinden sich jedoch in paarweise verschiedenen Komponenten von $G \setminus x$. Wir können diese Komponenten (unabhängig voneinander) so umfärben, daß jedes y_i die Farbe 1 bekommt, und dann x die Farbe 2.

Übrig bleiben die Fälle, in denen G wenigstens 2-zusammenhängend ist. In solche G suchen wir Tripel (x_1, x_2, x_n) mit

1. $x_1 x_2 \notin E, x_1 x_2 \in E, x_2 x_n \in E$.
2. $G \setminus \{x_1, x_2\}$ ist zusammenhängend

Dann konstruieren wir einen spannenden Baum für $G \setminus \{x_1, x_2\}$, mit Wurzel x_n, und legen dadurch die Reihenfolge $x_3, x_4, \dots, x_n$ fest. Wenn wir in dieser Reihenfolge greedy färben, beginnen wir bei x_1, x_2, denen wir dieselbe Farbe geben können (da sie nicht verbunden sind). Wenn wir schließlich bei x_n ankommen, der ja (wegen Regularität) Δ Nachbarn hat, finden wir darunter aber x_1 und x_2 mit der gleichen Farbe. Also ist für x_n noch eine Farbe übrig.

Bleibt die Existenz eines geeigneten Tripels (x_1, x_2, x_n) zu zeigen. Wenn G nicht 3-zusammenhängend ist, dann gibt es Knoten a, b, für die $G \setminus \{a, b\}$ in mehr als eine Komponente $C_1, \dots$ zerfällt. Weil G aber 2-zusammenhängend ist, hat a Nachbarn in jedem C_i. Wir wählen $x_n := a$, $x_{1,2} :=$ ein Nachbar von a in $C_{1,2}$.

Falls G 3-zusammenhängend ist, dann wählen wir x_1, x_2 als zwei Knoten im Abstand 2 (existieren, weil G keine Clique ist), und x_n als einen gemeinsamen Nachbarn von x_1, x_2. Nach Voraussetzung (3-zshg) ist $G \setminus \{x_1, x_2\}$ zusammenhängend. □

5.1.3 Aufgaben zu Färbungen

aus [Wes01]

5.1.20. Betrachte Graphen G, in denen je zwei ungerade Kreise wenigstens einen Punkt gemeinsam haben. (Erfüllt jeder K_n diese Eigenschaft? Nein.) Zeige, daß $\chi(G) \leq 5$.

5.1.24. Über einen Graphen G mit Knoten $\{0, 1, \dots, 359\}$ ist bekannt:

1. G ist 20-regulär
2. Knoten mit Abstand 1 und 2 sind *nicht benachbart*
3. Knoten mit Abstand $\{3, 4, 5, 6\}$ *sind* benachbart.

(Dabei sind Abstände zyklisch zu messen, z. B. 357 und 3 haben Abstand 6.) Zeige: $\chi(G) \leq 19$.

5.1.25. Der Einheitsgraph in der Ebene: Die Knotenmenge ist $\mathbb{R}^2$, und $E = \{xy : |xy| = 1\}$. (Also ein unendliche Graph). Zeige $4 \leq \chi(G) \leq 7$. (Bessere Schranken sind bis heute nicht bekannt!)

5.1.33. Zeige, daß jeder Graph G eine Knotenreihenfolge besitzt, für die eine Greedy-Färbung tatsächlich nur $\chi(G)$ Farben benutzt.

5.1.40. Wir nennen $n = |V(G)|$. Zeige $\chi(G) \cdot \chi(\overline{G}) \geq n$ sowie damit $\chi(G) + \chi(\overline{G}) \geq 2\sqrt{2n}$. Finde Graphen G, für die die Schranke scharf ist.

5.1.41 Zeige $\chi(G) + \chi(\overline{G}) \leq n + 1$.

5.2 Perfekte Graphen

5.2.1 Definition

Wie wir gesehen haben, ist der simpelste Grund für $\chi(G) \geq c$, daß G eine c-Clique enthält. Falls keine weiteren Gemeinheiten passieren, sollte auch tatsächlich $\chi(G) = c$ sein. Wir nennen die Graphen, auf die das zutrifft, perfekt:

Definition 120. *Ein Graph G heißt* perfekt*, falls für jeden* induzierten *Untergraphen $H \subseteq G$ gilt: $\omega(H) = \chi(H)$.*

5.2.2 Beispiele: Bipartite Graphen und Komplemente

Satz 121. *Jeder bipartite Graph $G = (V_1, V_2, E)$ ist perfekt.*

Proof. Jeder induzierte Untergraph H eines bipartiten Graphen ist bipartit. Dann gilt $\chi(H) = 1$ oder 2, je nach dem, ob H eine Kante enthält. Ebenfalls gilt $\omega(H) = 1$ oder 2. □

Satz 122. *Für jeden bipartiten Graphen G gilt: $\overline{G}$ ist perfekt.*

Aufgabe 123. *Beweise!*

5.2.3 Beispiele: Split-Graphen

Definition 124. *Ein Graph G heißt* Split-Graph*, wenn $V = V_1 \cup V_2$ (disjunkt), so daß $G|_{V_1}$ eine unabhängige Menge ist und $G|_{V_2}$ eine Clique.*

(Jedes Komplement eines Split-Graphen ist selbst ein Split-Graph.)

Aufgabe 125. *(Aus [Wes01]) 8.1.19. Finde alle Bäume, die Split-Graphen sind. Finde zwei nicht isomorphe Split-Graphen mit identischer (absteigend geordneter) Grad-Sequenz.*

Satz 126. *Jeder Split-Graph ist perfekt.*

Aufgabe 127. *Beweise!*

5.2.4 Intervallgraphen

Definition 128. *Zu jedem Mengensystem $M \subseteq 2^U$ betrachten wir den* Durchschnittsgraphen *$S_M = (M, \{xy \mid x \cap y \neq \emptyset\})$.*

Definition 129. *Ein Graph G heißt* Intervallgraph*, wenn G der Durchschnittsgraph eines Systems von Intervallen von reellen Zahlen ist.*

Aufgabe 130. *Ist jeder Baum ein Intervallgraph?*

Satz 131. *Jeder Intervallgraph ist perfekt.*

Proof. Jeder induzierte Untergraph H eines Intervallgraphen ist selbst ein Intervallgraph. Wir zeigen, daß man für H eine optimale Färbung durch einen Greedy-Algorithmus erhält. Wir ordnen dazu die Knoten (Intervalle) nach ihren linken Endpunkten. Wenn ein Knoten in dieser Anordnung d Vorgänger hat, dann gehört er zu einer d-Clique. □

5.2.5 Das Perfekte-Graphen-Theorem

(Vorlesung vom 27. 6.)

Der Begriff „perfekter Graph“ erscheint implizit bereits in Arbeiten von Gallai und König (1932), und wurde explizit definiert von Claude Berge (1960). Es dauert 10 Jahre, bis Lovasz und Fulkerson (unabhängig voneinander) 1970 folgenden Satz zeigen konnten:

Satz 132 (WPGT - *weak* perfect graph theorem). *G ist perfekt $\iff$ $\overline{G}$ ist perfekt.*

Mittlerweile muß man das „weak“ dranschreiben, denn es gibt auch ein SPGT — siehe später.

Wir werden jetzt das WPGT beweisen. Die Darstellung folgt [Bol98], Abschnitt V.5.

Eine äquivalente Charakterisierung perfekter Graphen

Satz 133. *Für jeden Graphen G gilt: G ist genau dann perfekt, wenn für jeden induzierten $H \subseteq G$ gilt:*

Es gibt eine unabhängige Menge $I \subseteq V(H)$ mit $\omega(H - I) < \omega(H)$.

Anschaulich: I zerstört jede größte Clique in H. (Beachte: „größte“ bedeutet hier nicht: bezüglich der Inklusion auf Knotenmengen, sondern bezüglich der Anzahl der Knoten.)

Natürlich können wir in jedem Graphen die großen Cliquen dadurch zerstören, daß wir aus jeder einen Knoten wählen. Im Allgemeinen wird die so gewählte Menge aber nicht unabhängig sein.

Proof. $\rightarrow$) Wenn G perfekt ist, dann besitzt jeder $H \subseteq G$ eine $\omega(H)$-Färbung f. Wir wählen I als eine Farbklasse davon. Da f eine korrekte Färbung war, kann $H - I$ keine $\omega(H)$-Cliquen mehr enthalten.

$\leftarrow$) Induktion nach $\omega(G)$. Induktionsanfang: $\omega(G) = 1$. Dann ist G unabhängig, also perfekt. Induktionsschritt: Wir müssen für jeden induzierten $H \subseteq G$ zeigen, daß $\chi(H) \leq \omega(H)$. Nach Voraussetzung gibt es $I \subseteq V(H)$ mit $\omega(H - I) < \omega(H)$. Nach Induktion besitzt $H - I$ eine $\omega(H - I)$-Färbung f. Wir erweitern diese zu einer $\omega(H - I) + 1$-Färbung von H, indem alle $x \in I$ die (gleiche) neue Farbe erhalten. Dann $\chi(H) \leq \omega(H - I) + 1 \leq \omega(H)$. □

Aufgabe 134. *Wie sieht eine solche Menge I aus, wenn G bipartit ist?*

Das Ersetzungs-Lemma für perfekte Graphen

Definition 135. *Für einen Graphen G mit $V(G) = \{1, 2, \dots, n\}$ und eine Liste von Graphen $[G_1, G_2, \dots, G_n]$ mit (der Einfachheit halber) disjunkte Knotenmengen definieren wir eine Graphen $G^* = G[G_1, G_2, \dots, G_n]$ durch*

$$\begin{aligned} V(G^*) &= \bigcup_{1 \le i \le n} V(G_i) \\ E(G^*) &= \bigcup_{1 \le i \le n} E(G_i) \cup \bigcup_{ij \in E(G)} V(G_i) \times V(G_j) \end{aligned}$$

Anschaulich: wir ersetzen in G jeweils Knoten i durch Graphen G_i, und verbinden verschiedene Teilgraphen G_i, G_j uniform so, wie vorher die Knoten i, j in G verbunden waren.

Satz 136. *Wenn $G, G_1, G_2, \dots, G_n$ perfekt sind, dann ist auch $G^* = G[G_1, G_2, \dots, G_n]$ perfekt.*

Proof. Es genügt, die Aussage für eine einzige Ersetzung $G^* = G[x := G_x]$ zu zeigen. Wir wollen Satz 133 anwenden. Da jeder induzierte Teilgraph $H^* \subseteq G^*$ die Form $H^* = H[x := H_x]$ mit $H \subseteq G, H_x \subseteq G_x$ hat, genügt es, zu zeigen, daß G^* selbst eine passende unabhängige Menge I besitzt, die alle Cliquen der Größe $\omega(G^*)$ trifft.

Nach Voraussetzung (und Satz 133) besitzt G_x eine unabhängige Menge I_x mit $\omega(G_x - I_x) < \omega(G_x)$. Weil G perfekt ist, besitzt G eine $\omega(G)$-Färbung f. Wir bezeichnen mit I die Farbklasse von x, setzen $J = I \setminus x \cup I_x$, und behaupten, daß J die gewünschten Eigenschaften hat.

Nach Konstruktion ist J unabhängig. Zu zeigen ist, daß J jede $\omega(G^*)$-Clique K trifft. Die Menge K zerfällt in einen Anteil $\subseteq (G - x)$ und einen Anteil $K_x \subseteq G_x$.

Falls K_x leer ist, dann ist K bereits eine Clique in $G - x$, und sogar eine größte, wird also von $I - x \subseteq J$ getroffen.

Falls K_x nicht leer ist, dann ist K_x sogar eine größte Clique in G_x, und wird deswegen von $I_x \subseteq J$ getroffen. □

Der Beweis des WPGT

Da es hier um Komplemente geht, messen wir nicht nur Cliquen, sondern auch unabhängige Mengen:

Definition 137. *$\alpha(G)$ bezeichnet die maximale Größe eine unabhängigen Menge in G.*

(Eselsbrücke: α/ω, erster/letzer Buchstabe des Alphabets, leerer/voller Graph.) Es gilt $\alpha(G) = \omega(\overline{G})$.

Aufgabe 138. *Für* jeden *Graphen G gilt: $\chi(G) \geq |V(G)|/\alpha(G)$.*

Proof. (von Satz 132). Wir benutzen Induktion nach $n = |V(G)|$. Für $n = 1$ gilt die Behauptung.

Für $n > 1$ ist nach Satz 133 zu zeigen, daß $\overline{G}$ eine unabhängige Menge I mit $\omega(G - I) < \omega(I)$ besitzt.

Eine unabhängige Menge I im Komplement entspricht einer Clique im Original. Wir suchen also eine Clique $K \subseteq G$, für die $\alpha(G - K) < \alpha(G)$.

Wir nehmen an, daß es kein solches K gibt. Wir bezeichnen mit $[K_1, K_2, \ldots, K_t]$ die Liste *aller* Cliquen in G. Dann gibt es zu jedem K_r eine unabhängige Menge I_r der Größe $\alpha(G)$, so daß $V(K_r) \cap V(I_r) = \emptyset$.

Für jedes $x \in G$ bezeichnen wir mit $i(x)$ die Anzahl der I_r mit $x \in I_r$. Wir konstruieren einen Graphen G^*, indem wir in G jedes x durch eine Clique der Größe $i(x)$ ersetzen. Nach Satz 136 ist G^* perfekt.

Wir rechnen andereseits $\omega(G^*)$ und $\chi(G^*)$ aus und erhalten damit einen Widerspruch:

Jede größte Clique in G^* entspricht einer Clique K_r in G (K_r ist nicht notwendige eine größte Clique in G). Damit gilt $\omega(G^*) = \sum_{x \in K_r} i(x)$. Diese Anzahl ist aber gleich $\sum_{s=1}^{t} |K_r \cap I_s|$, denn alle diese Durchschnitte haben Größe 0 oder 1. (Warum?) Nun ist diese Summe aber $\leq t-1$, da wenigstens I_r mit K_r leeren Durchschnitt hat. Also $\omega(G^*) \leq t - 1$.

Wir können auch $\chi(G^*)$ abschätzen: es gilt $|V(G^*)| = \sum_{x \in G} i(x) = \sum_{r=1}^{t} |I_r|$, weil wir einmal über die x, das andere mal über die r zählen. Die letzte Summe ist aber $= t\alpha(G)$, da nach Konstruktion alle I_r die Größe $\alpha(G)$ haben. Andererseits ist $\alpha(G^*) = \alpha(G)$, da wir, um G^* aus G zu erhalten, Knoten durch Cliquen ersetzt haben, wobei keine größeren unabhängigen Mengen entstehen können. Nun gilt $\chi(G^*) \geq |V(G^*)|/\alpha(G^*)$ (siehe Aufgabe oben). Daraus folgt aber $\chi(G^*) \geq t\alpha(G^*)/\alpha(G^*) = t$,

Damit gilt $\chi(G^*) > \omega(G^*)$, also ist G^* nicht perfekt. Aus der Annahme (zu jedem K_r existiert ein I_r, ...) folgt aber die Perfektion von G^*, also ist die Annahme falsch, und damit nach Kontraposition der Satz wahr. □

Aufgabe 139. *Wo wird in diesem Beweis die Induktionssannahme (Gültigkeit der Behauptung für $< n$) benutzt?*

Aufgabe 140. *Wieso sind die I_r paarweise verschieden?*

5.3 Chordale Graphen

5.3.1 Definition

Definition 141. *Ein Graph G heißt* chordal*, wenn es kein $k \geq 4$ gibt, so daß C_k induzierter Untergraph von G ist.*

D. h. in G hat jeder Kreis der Länge ≥ 4 wenigstens eine Sehne.

Beispiel 142. *Bäume sind chordal. Split-Graphen sind chordal.*

Definition 143. *Eine Menge $S \subseteq V(G)$ heißt (a, b)-Separator, falls a und b in verschiedenen Komponenten von $G - S$ liegen.*

Eine Menge $S \subseteq V(G)$ heißt minimaler Separator, falls es $a, b \in V(G)$ gibt, für die S ein (bezüglich Inklusion) minimaler (a, b)-Separator ist.

Satz 144. *Ein Graph ist genau dann chordal, wenn alle minimalen Separatoren Cliquen sind.*

Satz 145. *Intervallgraphen sind chordal.*

Satz 146. *Chordale Graphen sind perfekt.*

5.3.2 Lex-BFS

Definition 147. *Eine Knotenreihenfolge* $[v_1, v_2, \ldots, v_n]$ *heißt* perfekte Eliminations-Ordnung, *falls für alle i gilt: Die Nachbarschaft von* v_i *in* $G[v_{i+1}, \ldots, v_n]$ *ist eine Clique.*

Satz 148. *Ein Graph ist genau dann chordal, wenn er eine perfekte Eliminationsordnung besitzt.*

Der folgende Algorithmus konstruiert eine solche Ordnung.

Eingabe: Ein Graph $G = (V, E)$
Ausgabe: $\sigma : \{1, 2, \ldots, |V|\} \to V$

begin
  $n \leftarrow |V|$;
  for $v \in V$ **do** $L_{n+1}(v) \leftarrow [\,]$ **od**;
  for $i \leftarrow n$ **downto** 1 **do**
    wähle einen Knoten $v \in V$ mit größtem L_{i+1};
    $\sigma(i) \leftarrow v$; $V \leftarrow V \setminus \{v\}$;
    for $u \in V$ **do**
      if $u \in N(v)$ **then** $L_i(u) \leftarrow L_{i+1}(u) + [i]$
               else $L_i(u) \leftarrow L_{i+1}(u)$
      fi
    od
  od
end.

e

Kapitel 6

Minoren und Planare Graphen

6.1 Definitionen, Einfache Aussagen

Definition 149. *Eine* Zeichnung *eines Graphen G ist eine Abbildung, die jedem Knoten einen Punkt der Ebene zuweist, und jeder Kante ein Kurvenstück, das die zu den Knoten gehörenden Punkte verbindet.*

Wir lasse als Kurvenstücke nur Polygonzüge zu, weil wir Ausflüge in die Topologie oder Differentialgeometrie vermeiden wollen.

Definition 150. *Eine Zeichnung von G heißt* kreuzungsfrei*, wenn die Kurvenstücke zu verschiedenen Kanten keine gemeinsamen inneren Punkte besitzen.*

Definition 151. *Ein Graph heißt* planar*, wenn er eine kreuzungsfreie Zeichnung besitzt.*

Bemerkung 152. *Zu jeder kreuzungsfreien Zeichnung von G gibt es sogar eine topologisch äquivalente, deren sämtliche Kurvenzüge Strecken sind.*

Für das weitere benötigen wir:

Satz 153. *K_5 und $K_{3,3}$ sind nicht planar.*

Proof. Beweis durch Eulerschen Polyeder-Satz. Siehe Einleitung (Seite 8). □

Aufgabe 154. *Zeige, daß $\overline{C_7}$ nicht planar ist. (Reicht Satz 7?)*

Aufgabe 155. *Finde eine Zeichnung des $\overline{C_7}$ mit möglichst wenig Kreuzungen.*

Definition 156. *Die* Kreuzungszahl *(crossing number) eines Graphen G ist die kleinstmögliche Zahl von Kreuzungen in Zeichnungen von G.*

6.1.1 Topologische Einbettungen

Definition 157. *Der Graph* $G' = (V', E')$ *entsteht aus* $G = (V, E)$ *durch* Unterteilung *der Kante* xy*, wenn* $z \notin V$, $V' = (V \cup \{z\}$ *und* $E' = E \setminus xy \cup \{xz, zy\}$.

Definition 158. *Die Menge* TG *ist die Menge aller Graphen* G'*, die aus* G *durch eine Folge von Kanten-Unterteilungen entstehen.*

Definition 159. *Der Graph* G *ist* topologisch eingebettet *in den Graphen* H*, geschrieben* $G \leq_T H$*, falls* $\exists G' \in TG : G' \subseteq H$.

Beispiel 160. *Für alle* $3 \leq m \leq n : C_m \leq_T C_n$.

Satz 161. *Wenn* H *planar und* $G \leq_T H$*, dann* G *planar.*

Proof. Wir können aus einer kreuzungsfreien Zeichnung von H leicht eine kreuzungsfreie Zeichnung von G konstruieren. □

6.1.2 Minoren

Definition 162. *Der Graph $G' = (V', E')$ entsteht aus $G = (V, E)$ durch* Kontraktion *der Kante xy, geschrieben $G' = G/xy$, wenn $z \notin V$, $V' = (V \setminus x, y \cup \{z\}$ und $E' = E(G - xy) \cup \{zq \mid q \in G(x) \cup G(y) \setminus x, y\}$*

Der neue Punkt z erbt alle Nachbarn von x *und* y.

Definition 163. *Die Menge MG ist die Menge aller Graphen G', aus denen man G durch eine Folge von Kanten-Kontraktionen erhalten kann.*

Definition 164. *Der Graph G ist ein* Minor *des Graphen H, geschrieben $G \leq_M H$ (manchmal nur $G \leq H$), falls $\exists G' \in MG : G' \subseteq H$.*

Satz 165. *Wenn $G \leq_T H$, dann $G \leq_M H$.*

Proof. Wir können jede Kanten-Unterteilung durch eine Kontraktion aufheben. □

Die Umkehrung dieses Satzes gilt *nicht*:

Satz 166. *Es gibt Graphen G, H mit $G \leq_M H \wedge G \not\leq_T H$.*

Proof. $G = (\{1, 2, 3, 4, 5\}, \{15, 25, 35, 45\}), H = (\{1, 2, 3, 4, 5, 6\}, \{15, 25, 36, 46, 56\})$. Es gilt $G = H/56$. □

In einigen Fällen stimmen $G \leq_T H$ und $G \leq_M H$ jedoch überein:

Satz 167. *Wenn $\Delta(G) \leq 3$, dann $G \leq_T H \iff G \leq_M H$.*

($\Delta(G)$ ist der maximale Knotengrad in G.)

Proof. Die Richtung $G \leq_T H \Rightarrow G \leq_M H$ ist trivial. Sei nun G durch eine Folge von Kontraktionen aus einem Teilgraphen von H entstanden. Die letzte dieser Kontraktionen sei $G'/xy = G$, der aus xy entstandene Knoten heiße z. Nach Voraussetzung hat z höchstens drei Nachbarn in G. Deswegen hatte wenigstens einer der Knoten x, y, sagen wir x, in G' einen Grad ≤ 2. Wenn $\delta_{G'}(x) = 1$, dann ist $G \subseteq G'$, und wenn x in G' einen weiteren Nachbarn x' (außer y) besitzt, dann ist G' durch Unterteilung der Kante zx' aus G entstanden. □

Satz 168. *Wenn H planar und $G \leq_M H$, dann G planar.*

Proof. Wir können aus einer kreuzungsfreien Zeichnung von H leicht eine kreuzungsfreie Zeichnung von G konstruieren. □

6.2 Satz von Kuratowski

Gegenstand des Kapitels ist der Beweis von

Satz 169 (Kuratowski). *Ein Graph G ist genau dann planar, wenn $K_5 \not\leq_M G$ und $K_{3,3} \not\leq G$.*

Wir zeigen zunächst, daß wir in der Formulierung des Satzes $\leq_M$ und $\leq_T$ austauschen können:

Satz 170. *Für alle G gilt: $K_5 \not\leq_M G \wedge K_{3,3} \not\leq_M G$ genau dann, wenn $K_5 \not\leq_T G \wedge K_{3,3} \not\leq_T G$.*

Proof. Wegen Satz 165 ist nur $(K_5 \not\leq_T G \wedge K_{3,3} \not\leq_T G) \Rightarrow (K_5 \not\leq_M G \wedge K_{3,3} \not\leq_M G)$ zu zeigen. Wegen Satz 167 und $\Delta(K_{3,3}) = 3$ ist schließlich nur $(K_5 \not\leq_T G \wedge K_{3,3} \not\leq_T G) \Rightarrow K_5 \not\leq_M G$ zu zeigen.

Falls nun doch $K_5 \leq_M G$, dann gibt es in G also 5 Knotenmengen $[A_1, \ldots, A_5]$, so daß aus A_i durch Kontraktionen die Ecke i eines K_5 entsteht. Es gibt also für alle $i < j$ (wenigstens) einen Pfad zwischen A_i und A_j. A_i selbst ist auch zusammenhängend, also gibt es für alle $j \neq k$ Pfade von A_j nach A_k, die durch A_i verlaufen.

Wir sagen, A_i gehört zu Fall 1, falls alle diese Pfade in A_i eine gemeinsamen Punkt besitzen, sonst zu Fall 2. Falls alle $A_1, \ldots, A_5$ zu Fall 1 gehören, dann gibt es offensichtlich einen TK_5 in G.

Falls wenigstens ein A_i zu Fall 2 gehört, sagen wir A_1, dann gibt es zwei Knoten x, y in A_i, so daß x auf $A_2 - A_3$ und y auf $A_4 - A_5$ liegt, aber nicht andersherum. Diese Punkte $x, y \in A_1$, sowie passend gewählte Punkte aus A_2, A_3, A_4, A_5, bilden eine $TK_{3,3}$ in G. $\square$

Wir zeigen nun den Satz von Kuratowski zunächst für 3-zusammenhängende Graphen. Dazu eine Vorbertrachtung (die nichts mit Planarität zu tun hat).

Satz 171. *Wennn $|V(G)| > 4$ und G 3-zusammenhängend, dann gibt es eine Kante $e \in E(G)$, so daß G/e 3-zusammenhängend ist.*

Proof. Angenommen, für kein e ist G/e 3-zshg. Dann gibt es zu jedem $e = xy \in E(G)$ einen Knoten z, so daß $G - \{x, y, z\}$ zerfällt. Wir wählen unter allen solchen Tripeln (x, y, z) und Komponenten C von $G - \{x, y, z\}$ ein solches mit kleinstem $|C|$.

Jeder von x, y, z besitzt in jeder Komponente von $G - \{x, y, hz\}$ wenigstens einen Nachbarn. (Sonst wäre G nicht 3-zshg.) Sei v der Nachbar von z in C.

Nun gibt es auch zur Kante zv eine Knoten w, so daß $G - \{z, v, w\}$ zerfällt, und z, v, w haben Nachbarn in jeder der dabei entstehenden Komponenten.

Weil $xy \in E$, gibt es eine Komponente D von $G - \{z, v, w\}$, die weder x noch y enthält. Wir zeigen, daß D eine echte Teilmenge von C ist (nämlich eine Teilmenge von $C \setminus v$):

Wegen $v \in C$ liegt jeder D-Nachbar von v ebenfalls in C. (Davon gibt es wenigstens einen, da v ja Nachbarn in jeder Komponente hat.) D besteht dann aus allen von diesen Nachbarn (von v in C) erreichbaren Punkten, wobei die Pfade nicht über x, y gehen (und auch nicht über z). Damit ist D eine echte Teilmenge von C. $\square$

Satz 172. *Wenn G 3-zusammenhängend und wenn $K_5 \not\leq_M G$ und $K_{3,3} \not\leq_M G$, dann ist G planar.*

Proof. Wenn $|V(G)| = 4$, dann ist G ein K_4 und damit planar. Wenn $|V(G)| > 4$, dann gibt es eine Kante xy, so daß G/xy 3-zshg ist. Nach Induktion ist G/xy planar.

Wir nenne den durch Kontraktion aus xy entstandenen Punkt z. Die Nachbarn von z in G/xy bilden einen Kreis (um z) in einer kreuzungsfreien Zeichnung von G/xy.

Man muß nun alle mögliche Verteilungen (der auf dem Kreis angeordneten Nachbarn von z auf die Nachbarn von x und y) betrachten und stellt fest, daß man entweder die Zeichung von G/xy zu einer Zeichnung von G erweitern kann, oder in G einen MK_5 oder $MK_{3,3}$ findet. $\square$

6.3 Wohl-Quasi-Ordnungen

Literatur: Kapitel 10 aus [Die96], Kapitel III in [Wal00].

6.3.1 Definitionen, Beispiele

Die Relation $\leq_M$ erscheint in einem der schwersten Sätze der Graphentheorie – der Wagnerschen Vermutung. Zur knappen Formulierung benötigen wir einige Definitionen der Ordnungstheorie:

Definition 173. *Eine* Quasi-Ordung *ist eine transitive und reflexive Relation* $\leq$. *Wir schreiben* $x > y$ *für* $y \leq x \wedge x \not\leq y$.

Definition 174. *Eine Folge* $G_1, G_2, \ldots$ *heißt* absteigende Kette*, falls* $\forall i < j : G_i > G_j$.

Definition 175. *Eine Menge* C *heißt* Antikette*, falls für alle* $\{G_1, G_2\} \subseteq C$ *gilt:* $G_1 \not\leq_M G_2 \wedge G_2 \not\leq_M G_1$.

Definition 176. *Eine Quasi-Ordnung heißt* wohlfundiert*, falls jede absteigende Kette endlich ist.*

Definition 177. *Eine Quasi-Ordnung heißt* Wohl-Quasi-Ordnung*, wenn jede absteigende Kette und jede Antikette endlich ist.*

Aufgabe 178. *Die Relation „ist ein Teiler von " auf den natürlichen Zahlen ist wohlfundiert, aber keine Wohl-Quasi-Ordung. Finde eine unendliche Antikette. (Hinweis: Primzahlen)*

Aufgabe 179. *Die Relation* $\subseteq$ *(Teilgraph) ist* keine *Wohl-Quasi-Ordnung. Finde eine unendliche Antikette! (Hinweis: Kreise)*

6.3.2 Satz von Kruskal

mit einem bekannten Beweis (ca. 1960) von Nash-Williams, auf dem z. B. viele Methoden für Terminationsbeweise in Termersetzungssystemen beruhen.

Definition 180. *Für Wörter* $u, v \in \Sigma^*$ *schreiben wir* $u \subseteq v$ *(*u *ist* Teilwort *von* v*), falls* $\exists l, r \in \Sigma^* : l \cdot u \cdot r = v$.

Satz 181. $(\Sigma^*, \subseteq)$ *ist wohlfundierte Halbordnung, aber keine Wohl-Quasi-Ordnung.*

Proof. Es gibt unendliche Antiketten, z. B. ab^*c. □

Definition 182. *Für Wörter* $u = u_1 u_2 \ldots u_n, v \in \Sigma^*$ *schreiben wir* $u \leq v$ *(*u *ist* verstreutes *Teilwort von* v*), falls* $\exists c_0, c_1 \ldots c_n \in \Sigma^* : c_0 u_1 c_1 u_2 \ldots u_n c_n = v$.

Satz 183 (Kruskal). $(\Sigma^*, \leq)$ *ist Wohl-Quasi-Ordnung.*

Satz 184. *Wenn* $(U, \leq)$ *WQO ist, dann gilt für alle* $M \subseteq U$*:* $\min_{\leq}(M)$ *ist endlich.*

Proof. Die Menge $\min_{\leq}(M)$ ist eine Antikette. □

Aufgabe 185. *Bestimme die Menge* $\min_{\leq}(L)$ *für* $L =$ *die Menge der Dezimaldarstellungen von Primzahlen. Beginnt so:* $\{2, 3, 5, 7, 11, 19, 41, 64, \dots\}$*. Desgleichen für Quadratzahlen. Siehe [BW02].*

Die Anwendung von WQOs bei Terminations-Beweisen ergibt sich aus folgendem

Satz 186. *Jede Erweiterung einer WQO ist wieder eine WQO, und damit wohlfundiert.*

(Eine QO $\leq'$ ist eine Erweiterung einer QO $\leq$, falls $\forall x, y : x \leq y \Rightarrow x \leq' y$.)

Proof. Bei einer Erweiterung darf man (einige) Antiketten ordnen. Da es nach Voraussetzung keine unendlichen Antiketten gibt, entstehen daraus keine unendlichen absteigenden Ketten. □

6.3.3 Wagnersche Vermutung

Die Relation $\leq$ auf Strings entspricht genau der Minoren-Relation auf (markierten) Pfaden. Nach dem Satz von Kruskal ist $\leq$ eine WQO für (Pfade und) Bäume. Und weiter?

Satz 187 (Wagnersche Vermutung). *Die Relation* $\leq_M$ *ist eine Wohl-Quasi-Ordnung auf Graphen.*

Die Wagnersche Vermutung wurde ca. 1990 von Robertson und Seymour bewiesen. Der Beweis erschien in einer Folge von ca. 10 Artikeln, verteilt über 10 Jahre, in Fachzeitschriften, und umfaßt insgesamt mehrere hundert Seiten.

Äquivalent dazu ist die Aussage:

Satz 188. *Jede bezüglich* $\leq_M$ *nach unten abgeschlossene Menge* C *von Graphen läßt sich durch* endlich viele *verbotene Minoren charakterisieren:*

Wenn $\forall H \in C : \forall G \leq_M H : G \in C$*, dann existiert eine endliche Menge* F *mit* $\forall G : G \in C \iff \nexists H \in F : H \leq_M G$.

Wegen Satz 168 erfüllt die Menge der planaren Graphen die Voraussetzungen. Tatsächlich ist in diesem Fall durch den Satz von Kuratowski die Menge der verbotenen Minoren sogar explizit bekannt (und schon vor dem Beweis der Vermutung): $\{K_5, K_{3,3}\}$.

6.4 Der Vier-Farben-Satz

(Literatur: [Tho96] und viele andere Darstellungen.)

Die Frage nach der chromatischen Zahle von planaren Graphen hat die Entwicklung der Graphentheorie seit über 150 Jahren wesentlich beeinflußt.

Ähnlich wie beim Fermatschen Satz stellt die Antwort selbst (4) mittlerweile höchstens ein Kuriosum dar; viel wichtiger sind die Methoden, die zu ihrem Beweis entwickelt wurden — auch die, welche für diese Anwendung nicht erfolgreich waren.

Wir können den Beweis hier nicht führen, denn man muß eine riesige Fallunterscheidung verifizieren, was nur mit Computerhilfe funktioniert. Wir können jedoch einige Ideen verstehen.

6.4.1 Sechs Farben

Satz 189 (Euler). *Für jeden planaren Graphen G gilt $\chi(G) \leq 6$.*

Proof. Durch Induktion nach der Knotenzahl. Wenn $|V(G)| \leq 6$, dann trivial.

Nach dem Satz von Euler gibt es in jedem planaren Graphen wenigstens einen Knoten x mit $\delta(x) \leq 5$. Nach Induktion gibt es für $G - x$ eine 6-Färbung. Die Nachbarn von x haben darin höchstens 5 verschiedene Farben, so daß für x wenigstens eine Farbe übrigbleibt. □

6.4.2 Fünf Farben

Aufgabe 190. *Gibt es planare Graphen, bei denen jeder Knoten den Grad* 5 *besitzt?*

Satz 191 (Kempe, Heawood). *Für jeden planaren Graphen G gilt $\chi(G) \leq 5$.*

Proof. G enthält einen Knoten x vom Grad ≤ 5. Nach Induktion gibt es eine 5-Färbung f von $G - x$. Die Nachbarn von x nennen wir $x_1, x_2, \ldots, x_5$, und wir nehmen an, daß $f(x_i) = i$. (Falls nicht alle der Farben von x_i verschieden sind, dann können wir x selbst die fehlende Farbe geben.)

Für Farben i, j betrachten wir den Teilgraphen $G_{i,j} = (G - x)|_{f^{-1}(i) \cup f^{-1}(j)}$.

Für jedes $i \neq j$, und jede Komponente C von G_{ij} gilt: Wenn wir in C die Farben i und j vertauschen, erhalten wir wieder eine 5-Färbung von $G - x$.

Falls x_1 und x_3 in verschiedenen Komponenten von G_{13} liegen, färben wir Komponente von x_3 um und erhalten wieder eine Färbung, bei der x_1 und x_3 beide die Farbe 1 haben. Damit ist für x selbst die Farbe 3 freigeworden.

Falls x_2 und x_4 in verschiedenen Komponenten von G_{24} liegen, dann vertauschen wir ein der Komponente von x_4 die Farben 2 und 4 und färben x mit 4.

Übrig bleiben die Graphen, bei denen x_1 und x_3 in G_{13} verbunden sind, und x_2 mit x_4 in G_{24}.

Solche gibt es aber nicht, da sich der Pfad zwischen x_1 und x_3 sowie der zwischen x_2 und x_4 in einem gemeinsamen Punkt treffen müssen. □

Aufgabe 192. *Wo wird hier die Planarität von G benutzt?*

6.4.3 Vier Farben

Satz 193 (Appel, Haken, 1976). *Für jeden planaren Graphen G gilt $\chi(G) \leq 4$.*

Die Idee mit den Kempe-Ketten funktioniert leider nicht für Vier-Färbungen (TODO: Beispiel aus West), sie läßt sich aber verallgemeinern.

Wir betrachten dazu *Konfigurationen*, das sind planara Graphen mit „Kanten nach außen“, d. h. an den äußeren Knoten stehen noch Zahlen, die anzeigen, wieviele Kanten hier noch eintreffen.

Definition 194. *Eine Menge M von Konfigurationen heißt* unvermeidbar*, falls jeder planare Graph wenigstens ein Element aus M enthält.*

Beispiel 195. *Die Menge der Punkte mit 3,4,5 Beinen ist unvermeidbar (Satz von Euler).*

Definition 196. *Eine Konfiguration K heißt* reduzierbar*, wenn man für jeden Graphen G, der K enthält, aus einer Färbung von $G - K$ eine Färbung von G konstruieren kann.*

Beispiel 197. *Im Beweis des 5-Farb-Satzes wurde die Konfiguration „Punkt mit 5 Beinen“ reduziert (durch Umfärben des gesamten Graphen entlang der Kempe-Ketten).*

Der Vier-Farb-Satz folgt aus der Existenz einer unvermeidbaren Menge M von reduzierbaren Konfigurationen. Appel und Haken haben 1976 eine solche Menge (von ca. 2000 Konfigurationen) angegeben und mit einem Computer verifiziert. Robertson, Sanders, Seymour und Thomas haben 1996 eine unvermeidbare Menge von 633 Konfigurationen gefunden. Zu ihrer Verifikation ist ebenfalls Computer-Hilfe nötig, die Autoren haben die Quelltexte der entsprechenden Programme veröffentlicht, damit sich jeder Interessent von deren Korrektheit überzeugen kann.

6.5 Die Hadwiger-Vermutung

Literatur: [Die96], Abschnitt 6.4

Der Vier-Farb-Satz beschreibt die chromatische Zahl planarer Graphen. Nach dem Satz von Kuratowski lassen sich planare Graphen leicht durch verbotene Minoren charakterisieren. Daher die Idee, nach Zusammenhängen zwischen (verbotenen) Minoren und chromatischer Zahl zu suchen. *Außerdem* möchte man damit geometrischen Überlegungen (zu Planarität, kreuzungsfreien Wegen etc.) ganz aus dem Weg gehen, um evtl. neue Zugänge zum Vier-Farben-Problem zu finden.

Vermutung 198 (Hadwiger, 1943). *Für jedes $n \in \mathbb{N}$ und jeden Graphen G gilt: $\chi(G) \geq r \Rightarrow K_r \leq_M G$.*

Diese Vermutung gilt als eines der tiefsten offenen Probleme der Graphentheorie.

Aufgabe 199. *Zeige die Hadwiger-Vermutung für $r \leq 2$.*

Aufgabe 200. *Die Hadwiger-Vermutung ist wahr für $r = 3$.*

Lemma 201. *Wenn G 3-zshg, dann $K_4 \leq G$.*

Proof. Wähle zwei Knoten $a, b \in V(G)$ und drei unabhängige $a - b$-Wege p_1, p_2, p_3 in G (die existieren nach Satz von Menger). Da es keine parallelen Kanten gibt, ist o. B. d. A. höchstens p_1 eine Kante, und die anderen Wege p_2, p_3 enthalten je wenigstens einen echten inneren Knoten $x_i \in p_i$. Nach Voraussetzung ist $G - \{a, b\}$ zusammenhängend, also gibt es in $G - a, b$ einen Pfad von x_2 nach x_3. Damit ist aber $\{a, b, x_2, x_3\}$ ein eingebetteter K_4 in G. □

Satz 202. *Die Hadwiger-Vermutung ist wahr für $r = 4$.*

Proof. Wir betrachten G mit $\chi(G) \geq 4$. Falls G 3-zshg ist, dann gilt die Behauptung nach dem Lemma. Falls G nur 2-zshg ist, dann gibt es $x, y \in V(G)$, so daß $G - \{x, y\}$ in mehrere Komponenten zerfällt. Es gibt wenigstens eine Komponente H, für die $\chi(H \cup xy) \geq 4$. (Falls überall $\chi \leq 3$, dann könnten wir sogar eine 3-Färbung für G konstruieren.) Nach Induktion gibt es einen $K_4 \leq H \cup xy$. Falls $xy \in E(G)$, sind wir fertig, ansonstens gibt es aber eine $x - y$-Pfad durch eine andere Komponente von G, und auch dann haben wir den $K_4 \leq G$. □

Mit etwas mehr Mühe stellt man fest:

Satz 203. *Für $r \in \{5, 6\}$ ist die Hadwiger-Vermutung äquivalent zum 4-Farb-Satz.*

Für $r \geq 7$ ist die Vermutung nach wie vor offen.

Wir können jedoch wenigstens die folgende Abschwächung der Vermutung für alle r zeigen:

Satz 204 (Wagner, 1964). *Für alle G: $\chi(G) \geq 2^r \Rightarrow K_r \leq_M G$.*

(ist Übungsaufgabe 6.10 in [Die96], Lösung in [Hal89], Abschnitt 6.6)

Proof. Induktion nach r. Für $r = 1$ ist die Aussage trivial.

Falls G mit $\chi(G) \geq 2^{r+1}$, dann wähle ein $x \in G$ und betrachte die Graphen G_d für $d \geq 1$ mit $G_d = G|\{y \mid x \neq y, \mathrm{dist}(x, y) = d\}$. (Punkte in Abstand d von x).

Wenigstens ein G_d hat chromatische Zahl $\geq 2^r$. (Falls für alle d: $\chi(G_d) \leq 2^r - 1$, dann könnte man G selbst mit $1+(2^r-1)+(2^r-1) < 2^{r+1}$ Farben färben: eine Farbe für x, $2^r - 1$ Farben für $G_1 \cup G_3 \cup \ldots$, weil die ungeraden Schichten untereinander keine Kanten haben, ebenso $2^r - 1$ Farben für die geraden Schichten.)

Deswegen enthält ein G_d nach Annahme einen K_r als Minor. Jeder $y \in G_d$ hat einen Nachbarn in G_{d-1}. Wir kontrahieren nun alle Schichten $x \cup G_1 \cup \ldots \cup G_{d-1}$ auf einen einzigen Punkt, und haben damit einen K_{r+1} als Minor. $\square$