4.9 Alg. (von Kruskal, greedy-Min):

Input: $G = ([n], E)$ zusammenhängend, $|E| = m$, Gewichte $c \in \mathbb{R}^E$

Output: $T^* = ([n], E^*) \in \operatorname{argmin}_{T \in \mathcal{T}_G} c(T)$

1. $E^* \leftarrow \emptyset$
2. Sortiere E nach aufsteigendem Gewicht s.d. $c_{e_1} \le c_{e_2} \le \ldots \le c_{e_m}$
3. for $i = 1, \ldots, m$ {
4. if ($E^* \cup \{e_i\}$ enthält keinen Kreis) {
5. $E^* \leftarrow E^* \cup \{e_i\}$
6. }

4.10 Satz: Alg 4.9 ist korrekt.

Bew.: Alg. 4.9 ist ein Spezialfall von Alg. 4.5. Sei $i_1 = 1 < \ldots < i_{n-1}$ mit $|E^* \cap \{e_1, \ldots, e_{i_j}\}| \neq |E^* \cap \{e_1, \ldots, e_{i_j - 1}\}|$, $j = 1, \ldots, n-1$. und $e_{i_j} = u_{i_j} v_{i_j}$, $j = 1, \ldots, n-1$. Dann wähle in Alg. 4.5.3 U_i mit $U_i \ni u_{i_j}$. □

4.7 Alg. (von Prim):

Input: $G = ([n], E)$ zusammenhängend, Gewichte $c \in \mathbb{R}^E$

Output: $T^* = ([n], E^*) \in \operatorname{argmin}_{T \in \mathcal{T}_G} c(T)$

1. $E^* \leftarrow \emptyset$
2. wähle $U \leftarrow \{u\}$, $u \in V$
3. for $i = 1, \ldots, n-1$ {
4. wähle $e_i = u_i v_i \in \operatorname{argmin}_{u_i \in U, v_i \notin U} c_{u_i v_i}$
5. $E^* \leftarrow E^* \cup \{e_i\}$
6. }

4.8 Satz Alg 4.7 ist korrekt.

Bew.: Wähle in Alg. 4.5.3 $U_i = U$. □

4.2 Bsp. (Wiederholung):

$c(T) = 15$ Prim Startknoten • Greedy-Min Greedy-Max

4.11 Alg. (Greedy-Max):

Input: $G = ([n], E)$ zusammenhängend, $|E| = m$, $c \in \mathbb{R}^E$

Output: $T^* = ([n], E^*) \in \operatorname{argmin}_{T \in \mathcal{T}_G} c(T)$

1. $E^* \leftarrow E$
2. Sortiere E nach absteigendem Gewicht s.d. $c_{e_1} \geq \dots \geq c_{e_m}$
3. for $i = 1, \dots, m$ {
4. if ($E^* \setminus \{e_i\}$ zusammenhängend) {
5. $E^* \leftarrow E^* \setminus \{e_i\}$
6. }.

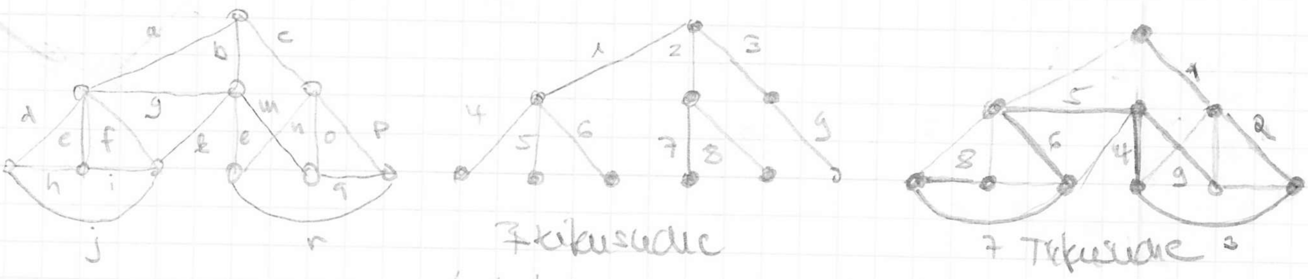
4.12 Satz: Alg 4.11 ist korrekt.

Bew.: Induktion über m.

Ind. Anf.: $m = 1$ ✓

Ind. Schritt: $m \to m+1$. e_1 ist eine Brücke ✓. Sei e_1 keine Brücke. Dann bestimmt der Greedy-Min Alg. 4.9 einen minimalen aufspannenden Baum in $G' = ([n], E \setminus \{e_1\})$, der auch ein min. aufspannender Baum in G ist. □

4.13 Bsp. (Breiten- und Tiefensuche)



Breitensuche   Tiefensuche

~~a b c d e f g k l m n o p h j i r q~~
1 2 3 4 5 6 7 8 9

~~a b c n o p q n l m k g d e f i h~~
1 2 3 4 9 5 6 7 8

4.14 Alg. (Breiten- und Tiefensuche)

Input: $G = ([n], E)$

Output: inklusionsmax. Wald $W = ([n], E^*)$

1. $E^* \leftarrow \emptyset$
2. $U \leftarrow V$ Menge der unbesuchten Knoten
   $F \leftarrow E$, unbenutzten Kanten

```
3.  forall v ∈ V {
4.      if (v ∈ U) {
5.          U ← U \ {v}          (v ist besucht)
6.          S ← S(v) = S(v) ∩ F
7.          while (S ≠ ∅) {
8.              wähle e = uw ∈ S  s.d. u ∈ U
9.              S ← S \ {uw} ,  F ← F \ {uw}
10.             if (w ∉ U) {
11.                 U ← U \ {w}
12.                 S ← S ∪ (S(w) ∩ F)
13.             }
14.     }
15. }
```

<u>4.15 Satz</u> Alg. 4.14 ist korrekt.

<u>Bew:</u> Sei $\overline{V} = V \setminus U$ die Menge der besuchten Knoten. Dann gilt in Alg 4.14.6 stets $S \supseteq S(\overline{V})$. In 4.14 7-13 wird $\overline{V}$ um einen benachbarten Knoten erweitert, solange dies möglich ist. In 4.14 3 wird ggf. eine neue Komponente eröffnet. □

<u>4.16 Bem.</u> Die Menge S kann mit verschiedenen Datenstrukturen implementiert werden.

a) S = First-in-First-out-Schlange (FIFO-Queue)
⇒ Alg. 4.14 = <u>Breitensuche</u>, engl. Breadth first search (BFS)

b) S = Last-in-First-out-Stapel (LIFO-Stack)
⇒ Alg. 4.15 = <u>Tiefensuche</u>, engl. Depth first search (DFS).

DFS läßt sich besonders einfach rekursiv programmieren und braucht nur wenig Speicherplatz.

5. Laufzeit von Algorithmen

5.1 Frage: Algorithmus A: Problem $\Pi = \{\text{Instanz}\} \to \{\text{Lösung}\}$

a) Existiert A zur Lösung von Π?

b) Welche Laufzeit hat A?

c) Wieviel Speicher benötigt A?

Klar: Die Laufzeit hängt von der Größe des Problems ab. Das Problem muss mit einem "Codierungsschema" codiert werden. Wir codieren Probleme als Folgen von Zahlen.

5.2 Def (Codierungslänge): Sei $z \in \mathbb{Z}$, $p/q \in \mathbb{Q}$ mit $p \in \mathbb{Z}$, $q \in \mathbb{N}$, p,q teilerfremd, $x \in \mathbb{Q}^n$, $A \in \mathbb{Q}^{m \times n}$, $G = ([n], E)$. Dann ist die Codierungslänge von z, $\frac{p}{q}$, x, A und G def. als

a) $\langle z \rangle := \underbrace{\lceil \log_2(|z|+1) \rceil}_{\text{Anzahl Bits}} + \underbrace{1}_{\text{Vorzeichen}}$

z	z_2	$\|z\|+1$	$\lceil \log_2(\|z\|+1) \rceil$	$\langle z \rangle$
0	0	1	0	1
1	1	2	1	2
2	10	3	2	3
3	11	4	2	3
4	100	5	3	4
5	101	6	3	4
6	110	7	3	4
7	111	8	3	4
8	1000	9	4	[illegible]

b) $\langle p/q \rangle := \langle p \rangle + \langle q \rangle$

c) $\langle x \rangle := \sum_{i=1}^{n} \langle x_i \rangle$

d) $\langle A \rangle := \sum_{i=1}^{m} \sum_{j=1}^{n} \langle a_{ij} \rangle$

29.04.13

e) Bei Graphen hängt die Codierungslänge vom Codierungsschema ab:

i) Knoten-Kanten-Inzidenzmatrix (a_{ve}): $\langle G \rangle = \langle (a_{ve}) \rangle = \underbrace{nm}_{\text{0en}} + \underbrace{2m}_{\text{1en}} = (n+2)m$

ii) Knoten-Knoten-Inzidenzmatrix (a_{uv}): $\langle G \rangle = \langle (a_{uv}) \rangle = n^2 + 2m$

iii) Kantenliste $E \subseteq \binom{[n]}{2}$: $\langle G \rangle \leq m \langle n \rangle \approx m \log_2 n$

f) $\{I \in \mathbb{Q}^k, k \in \mathbb{N}\} \supseteq \Pi$ Problem

$I \in \Pi$ Instanz mit wohldef. Codierungslänge

$L(I) \in \mathbb{Q}^k$, $k \in \mathbb{N}$ Lösung von I

5.3 Def. (Laufzeit und Speicherplatzbedarf von Algorithmen)

Ein Algorithmus $A: \Pi \to \{L(I) : I \in \Pi\}$, $I \mapsto L(I)$ zur Lösung eines Problems führt eine Abfolge folgender elementarer Operationen auf einer Random-Access-Maschine (Computer) auf dem in konstanter Zeit auf jeden Speicherplatz zugegriffen werden kann) aus:

i) Addition und Subtraktion

ii) Multiplikation und Division

iii) Lesen, Schreiben, Löschen eines Datums

iv) Vergleichen

a) Die Laufzeit von A zur Lsg. von $I \in \Pi$ ist die Anzahl elementarer Operationen, die A zur Berechnung von $L(I)$ durchführt, multipliziert mit der Codierungslänge der größten auftretenden Zahl.

b) Der Speicherplatzbedarf von A zur Lsg von $I \in \Pi$ ist die maximale Anzahl an Bits, die A zur Berechnung von $L(I)$ verwendet.

c) $f_A : \mathbb{N} \to \mathbb{N}$, $n \mapsto \max\{$Laufzeit von A zur Lsg von $I \in \Pi : \langle I \rangle \leq n\}$

Laufzeitfunktion von A

$f_A^{\#} : \mathbb{N} \to \mathbb{N}$, $n \mapsto \max\{\#$el. Op. " " $\}$

d) $S_A : \mathbb{N} \to \mathbb{N}$, $n \mapsto \max\{$Speicherpl. " $\}$

Speicherplatzfunktion von A.

e) A hat polynomielle Laufzeit / ist ein polynomieller Alg.

$:\Leftrightarrow \exists$ Polynom $p : \mathbb{N} \to \mathbb{N}$ mit $f_A(n) \leq p(n) \; \forall n \in \mathbb{N}$.

f) A hat polynomiellen Speicherplatzbed.

$:\Leftrightarrow \exists$ Polynom $q : \mathbb{N} \to \mathbb{N}$ mit $s_A(n) \leq q(n) \; \forall n \in \mathbb{N}$.

g) Seien $g, f : \mathbb{N} \to \mathbb{N}$ zwei Funktionen.

$g(n) = O(f(n)) :\Leftrightarrow \exists C \in \mathbb{N} : g(n) \leq C f(n) \; \forall n \in \mathbb{N}$

"Landausches O"   g ist von der (größten) Ordnung von f.

5.4 Beobachtung (Rechenregeln für Polynome in O-Notation):

a) $O(1) + O(1) = O(1)$

b) $O(1) + O(n) = O(1)$

c) $O(1) + O(n) + O(n^2) = O(n^2)$

5.5 Bem.: Man sagt

A hat quadratische Laufzeit $:\Leftrightarrow f_A(n) = O(n^2)$ usw.

5.4 Bsp.:

a) Alg. (Fakultät)

Input: $k \in \mathbb{N}$

Output: $fak = k!$

Datenstrukturen: $\underbrace{k, i, fak}_{\text{Register}}$ $\overbrace{\text{lesen schreib. vgl. add mult. } \Sigma:}^{\text{elementare Operationen}}$

		lesen	schreib.	vgl.	add	mult.	Σ
1.	lese k, $fak \leftarrow 1$	1.(1	1				) = 2 = O(1)
2.	for $i = 1, \dots, k$ {	k·(		1	1		) = 2k = O(k)
3.	$fak \leftarrow fak \cdot i$	k·(				1	) = k = O(k)
4.	}						3k+2 = O(k)

Codierungslänge Input: $\langle k \rangle = n$

elementare Operationen: $O(k) = O(2^{\langle k \rangle}) = O(2^n)$

Codierungslänge größte Zahl: $\langle k! \rangle \approx \langle k^k e^{-k} \sqrt{2\pi k}/2 \rangle$

$\leq \langle k^k \rangle$

$\overset{\text{Übung}}{\leq} k \langle k \rangle = O(n 2^n)$

$f_{\text{Fakultät}}(n) = O(2^n)\, O(n 2^n) = O(n 4^n)$, d.h. die Laufzeit von Fakultät ist exponentiell.

b) Alg. (GGT)

Input: $a, b \in \mathbb{N}$, $b \geq a$

Output: $ggT(a,b)$

Datenstrukturen: $a, b, c \in \mathbb{N}$

1. do {
2. $c \leftarrow b$ $\quad O(1)$
3. $b \leftarrow a$ $\quad O(1)$
4. $a \leftarrow c \bmod b$ $\quad O(1)$
5. } while $(a \neq 0)$ $\quad O(1)$
6. output b $\quad O(1)$

Durchläufe = k

$a_0 = a$

$c = q_1 a_0 + a_1, \; 0 \leq a_1 < b$

$b = \underbrace{q_2}_{\geq 1} a_1 + a_2, \; 0 \leq a_2 < a_1$

$\geq 2 a_2$

$\Rightarrow a_0$ halbiert sich alle 2 Durchläufe

$\Rightarrow k \leq 2 \log_2 a_0 = O(\langle a \rangle)$. Sei $\langle b \rangle = n \geq \langle a \rangle$

$f_{ggT}(n) \geq 2\,O(\langle a \rangle) = O(n)$, d.h. die # el. Op ist linear

$\Rightarrow f_{ggT}(n) = O(\langle a \rangle \langle b \rangle) = O(n^2)$, die Laufzeit von ggT ist quadratisch.

c) Alg (Kreischeck)

Input: $G=(V,E)$, $f\in\binom{V}{2}$, $f\notin E$, $|V|=n$, $|E|=m$

Output: Ja/1, falls $E\cup\{f\}$ einen Kreis enthält, Nein/0 sonst

1. Sei $f = uv$ $O(1)$

2. Führe BFS in G mit Startknoten u aus (d.h. Zeile 4.14.3 durch 3. für $u\in V$) $O(\max\{m,n\})$

3. if ($v\in U$) {output Nein/0} $O(1)$

   else { " Ja/1 }

$O(\max\{m,n\})$

Codierungslänge Input: $\underbrace{\langle G\rangle}_{\text{Kantenliste}} = O(\overbrace{m\langle n\rangle}^{=k})$ $\Rightarrow |V| = O(|E|) = O(m)$

elementare Operationen: $O(|E|) = O(m)$

Codierungslänge größte Zahl: $\langle n\rangle = O(\langle n\rangle)$

$f_{\text{Kreischeck}}(k) = O(\langle n\rangle m) = O(k)$, d.h. linear.

$f^{\#}_{\text{Kreischeck}}(k) = O(m) = O\left(\frac{k}{\langle n\rangle}\right)$

d) Alg 4.19 von Kruskal

Sortieren: $O(m\log m) = O(m^2)$

Einfügen und Kreischeck $O(m\cdot m)$

größte Zahl: $O(C)$, $C = \max\{\langle n\rangle, \langle c_e\rangle, e\in E\}$

$f_{\text{Kruskal}}(\overbrace{mC}^{=k}) = O(m^2C) = O(k^2)$, d.h. quadratisch

$f^{\#}_{\text{Kruskal}}(k) = O(m^2) = O\left(\frac{k^2}{C^2}\right)$.

5.5 Def. (Entscheidungsproblem): Ein Problem Π mit $L(I) \in \{0,1\}$ $\forall I \in \Pi$ heißt <u>Entscheidungsproblem</u>.
"nein" "ja"

5.7 Def. (Komplexitätsklassen P und NP) Sei Π ein Entscheidungsproblem.

a) Π gehört zur Klasse P, (der polynomial lösbaren Probleme), wenn es einen polynomialen Alg. zur Lsg von Π gibt.

b) Π gehört zur Klasse NP (der nicht deterministisch polynomial lösbaren Probleme), wenn es einen nicht-deterministischen polynomialen Algorithmus zur Lösung von Π gibt, d.h. einen Alg., der für jede Ja-Instanz $I \in \Pi$ mit $L(I) = 1$ ein Zusatzobjekt $Q(I)$ errät, mit dessen Hilfe die Korrektheit von $L(I)=1$ überprüft werden kann, und in einer Laufzeit, die polynomial in $\langle I \rangle$ ist, überprüft, ob $Q(I)$ ein Objekt ist, aufgrund dessen Existenz die Antwort $L(I)=1$ richtig ist.

30.04.13

5.7 Bsp. Folgende Entscheidungsprobleme sind in P

a) Ist ein Graph G zusammenhängend?

b) Ist ein Graph G eulersch?

Folgende Entscheidungsprobleme sind in NP:

c) Alle Probleme in P.

d) Enthält G einen Hamiltonkreis, d.h. einen Kreis, der jeden Knoten genau einmal besucht? (<u>Hamiltonsches Kreisproblem</u>)

e) Kann man die Knoten von G mit k Farben färben, so daß keine zwei benachbarten Knoten die gleiche Farbe haben? (<u>k-Färbungsproblem</u>)

f) <u>Faktorisierung</u>: Ist $n \in \mathbb{N}$ faktorisierbar (nicht prim?)

5.6 Bsp.: Sei $G=(V,E)$ ein Graph, $|V|=n$, $|E|=m$, $k\in\mathbb{N}$

Wir definieren folgende Entscheidungsprobleme

a) Hamiltonkreis:

Input: G

Output: $1 \Leftrightarrow \exists$ Hamiltonkreis $C \subseteq G$: $V(C)=V$

b) Clique

Input: G, k

Output: $1 \Leftrightarrow \exists$ Clique $Q \subseteq V$: $uv \in E\ \forall u,v \in Q$, $|Q| \geq k$.

e) SAT

Input: Boolsche Variablen $x_1, \dots, x_n$

Klauseln $C_i = \bigvee \neg x_i$

Boolscher Ausdruck $\bigwedge C_i$

Output: $1 \Leftrightarrow$ Truth assignment $x_i \mapsto \{0,1\}$: $\bigwedge C_i = 1$.

c) Stabile Menge

Input: G, k

Output: $1 \Leftrightarrow \exists$ stabile Menge $S \subseteq V$: $uv \notin E\ \forall u,v \in S$, $|S| \geq k$

d) Färbung

Input: G, k

Output: $1 \Leftrightarrow \exists$ Partition $V = \bigcup_{i=1}^{k} V_i$ in $\underbrace{\text{stabile Mengen}}_{\text{Farbklassen}}$

f) Faktorisierung

Input: $n \in \mathbb{N}$

Output: $1 \Leftrightarrow \exists a,b \in \{2,\dots,n\}: n=ab$

g) Primzahltest

Input: $n \in \mathbb{N}$

Output: $1 \Leftrightarrow \nexists a,b \in \{2,\dots,n\}: n=ab$

Bew.: a) ist polynomial lösbar mit BFS (Übung).
b) BFS + Knotengrade bestimmen
c) Zusatzobjekt $Q(I)$ wird nicht benötigt.
d) Enthält G einen Hamiltonkreis $H(G)$. Dann rate $Q(G) = H(G)$ und verifiziere in polynomialer Zeit, daß $H(G)$ ein Hamiltonkreis in G ist.
e) Errate eine Färbung $V = \dot{\bigcup}_{i=1}^{k} V_i$ mit k Farben und überprüfe, $uv \notin E$ $\forall u,v \in V_i$, $i=1,..,k$. in polynomialer Zeit.
f) Rate Primfaktorzerlegung und verifiziere sie.

5.8 Bem.: Es ist nicht bekannt (und eines der Millenium-Probleme), ob $P = NP$ oder $P \neq NP$ ist.

5.9 Bem.: a) $\Pi \in NP$ sagt nichts darüber aus, wie man verifiziert, ob $I \in \Pi$ eine Nein-Instanz ist, d.h. wenn es einen Beweis für Ja gibt, muß es noch lange keinen Beweis für Nein geben, z.B.:
i) Ist G nicht hamiltonsch?
ii) Ist $n \in \mathbb{N}$ prim?
b) $\Pi \in NP$ sagt auch nichts darüber aus, wie $Q(I)$ zu finden ist.

5.10 Def. (Polynomiale Transformation): Seien Π, Π' Entscheidungsprobleme. Ein Algorithmus $T: \Pi \to \Pi'$ mit $L(I) = L(T(I))$ $\forall I \in \Pi$ ist eine polynomiale Transformation von Π in Π'. Man sagt Π wird auf Π' reduziert, und schreibt $\Pi \propto \Pi'$.

5.11 Def. (NP-vollständige Probleme): Ein Entscheidungsproblem Π^* heißt NP-vollständig, wenn es für jedes Problem $\Pi \in NP$ eine polynomiale Transformation von Π in Π^* gibt.

5.12 Satz (Karp [1972]): Das hamiltonsche Kreisproblem und das Stabile-Mengen-Problem sind NP-vollst.
Bew.: ohne Bew.

5.13 Def (Minimierungs- und Maximierungsproblem):

Sei Π ein Problem, für $I \in \Pi$ sei $X(I)$ eine Menge von Lösungen, $C: \bigcup \{X : x \in X(I), I \in \Pi\} \to \mathbb{Q}$ eine Funktion, $L(I) = \operatorname{argmin}_{x \in X(I)} C(x)$. Dann heißt Π Minimierungsproblem. Π' mit Instanzen (I, z), $I \in \Pi$, $z \in \mathbb{Q}$, $L(I,z) = \begin{cases} 1 & L(I) \le z \\ 0 & \text{sonst} \end{cases}$ heißt das zu Π gehörige Entscheidungsproblem.

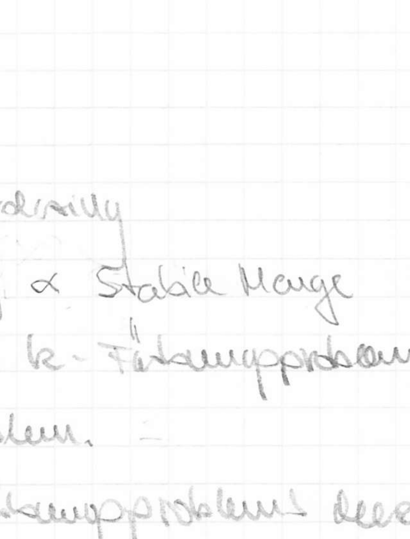
Π ist in P, wenn es einen polynomiellen Alg. zur Lösung von Π gibt. Π ist NP-hard, wenn Π' NP-schwer ist. Analog für Maximierungsprobleme.

5.14 Bsp.: Die Optimierungsversionen der Probleme aus Bsp. 5.6 heißen

a) Min-Hamiltonkreis = Traveling Salesman Problem (TSP)
b) Max-Clique
c) Max-Stabile Menge
e) Max-SAT
d) Min-Färbung = Coloring

5.15 Satz: k-Coloring $\propto$ Stabile Menge

Bew: Wir reduzieren das k-Färbungsproblem auf das Stabile-Mengen-Problem.
Sei eine Instanz des k-Färbungsproblems durch $G = (V, E)$ und $k \in \mathbb{N}$ gegeben.
Wir konstruieren eine Instanz des Stabile-Mengen-Problems wie folgt:



$G' = (V', E')$ mit
$V' = V \times [k]$
$E' = \{(v,i)(v,j) : 1 \le i < j \le k\} \cup \{(u,i)(v,i),\ uv \in E,\ i \in [k]\}$
$k' = V'$

$\Rightarrow$ $S' \subseteq V'$ stabil, $S' = \dot{\bigcup}_{i=1}^{k} S_i' = \dot{\bigcup}_{i=1}^{k} \{(v,i) \in S'\}$,
$S_i = \{v \in V : (v,i) \in S_i'\}$ stabil, $\dot{\bigcup} S_i = V$ k-Färbung von G. G' kann in polynomieller Zeit konstruiert werden. □

5.16 Satz: Stabile Menge $\propto$ k-coloring.

Bew.: Sei eine Instanz des Stabile Mengen Problems durch $G=(V,E)$ und $k \in \mathbb{N}$ gegeben. Dann gilt:

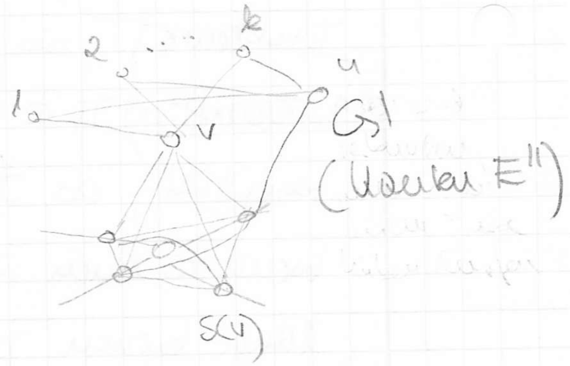
$$S \subseteq V \text{ stabil} \Leftrightarrow \forall uv \in E: u \notin S \text{ oder } v \notin S$$
$$\Leftrightarrow \forall uv \in E: \{u,v\} \cap V \setminus S \neq \emptyset$$

d.h. $V \setminus S$ überdeckt alle Kanten

$\Rightarrow \exists\, S \subseteq V$ stabil, $|S| \geq k \Leftrightarrow \exists\, \bar{S} \subseteq V$ Knotenüberdeckung, $|\bar{S}| \leq |V|-k$.

Wir zeigen, wie man eine solche Knotenüberdeckung durch Färben findet.
Konstruiere dazu folgenden Graphen $G'=(V',E')$:

$$V' = V \cup E \cup [k]$$



$$E'' = \{\{uv, vw\} : \underbrace{uv, vw \in E}_{\text{gemeinsamer Knoten}}\}$$
$$\cup\ \{\{v, vw\} : vw \in E\}$$
$$\cup\ \{\{i,v\} : i \in [k],\ v \in V\}$$
$$E' = \binom{V'}{2} \setminus E''$$

Dann sind die maximalen stabilen Mengen, die einen Knoten v bzw. i enthalten die Mengen $\{i,v\}$, $i \in [k]$, $\{v, S(v)\}$ bzw. $\{i,v\}$, $v \in V$.

Wir behaupten:

$$\exists\, |V|\text{-Färbung von } G' \Leftrightarrow \exists\, |V|-k \text{ Knotenüberdeckung } \bar{S} \text{ von } G$$
$$\Leftrightarrow \exists \text{ stabile } k\text{-Menge } S \text{ in } G.$$

"$\Leftarrow$" O.B.d.A. $|\bar{S}| = |V|-k$. Färbe den i-ten Knoten von $\bar{S}$ ($i=1,\dots,k$) mit Farbe i, ebenso alle seine noch ungefärbten inzidenten Kanten, und die restlichen k Knoten aus V mit den restlichen Farben, jeweils zusammen mit einem Knoten $i \in [k]$.

"$\Rightarrow$" Die Knoten $i \in [k]$ haben alle verschiedene Farben, jeder ist mit max. einem Knoten $v \in V$ gleichfarbig. Die restlichen $|V|-k$ Knoten V bestimmen die Farben für ihre inzidenten Kanten. $\square$

6. Matroide und Unabhängigkeitssysteme

6.1 Def. (Unabhängigkeitssystem): Sei M eine endliche Menge. Eine Mengenfamilie $\mathcal{I} \in 2^M$ heißt Unabhängigkeitssystem oder (abwärts-) monotones Mengensystem auf M wenn gilt:

a) $\emptyset \in \mathcal{I}$

b) $X \subseteq Y \in \mathcal{I} \Rightarrow X \in \mathcal{I}$.

6.3 Bem.: Basis- und Zirkuitssystem eines US sind Antiketten, d.h. Mengensyst. $\subseteq 2^M$ s.d. keine 2 Elemente aus $\mathcal{B}$ bzw. $\mathcal{C}$ ineinander enthalten sind.

6.2 Def. (Zirkuit, Basis, Rang): Sei $\mathcal{I}$ ein Unabh.-sys. auf M, $X, Y \subseteq M$.

a) $X \left\{\begin{matrix}\text{un}\\ -\end{matrix}\right\}$ abhängige Menge $:\Leftrightarrow X \left\{\begin{matrix}\in\\ \notin\end{matrix}\right\} \mathcal{I}$

b) X Basis von Y $:\Leftrightarrow$ X max. unabhängige Teilmenge von Y
$\Leftrightarrow X \in \mathcal{I}$ und $X' \in \mathcal{I}$, $X \subseteq X' \subseteq Y \Rightarrow X = X'$

c) $\mathcal{B} := \mathcal{B}(\mathcal{I}) := \{X \subseteq M : X$ Basis von $M\}$ Basissystem von $\mathcal{I}$

d) X Zirkuit $:\Leftrightarrow X$ min. abhängige Menge
$\Leftrightarrow X \notin \mathcal{I}$ und $X \supseteq Y \notin \mathcal{I} \Rightarrow Y = X$

e) $\mathcal{C} := \mathcal{C}(\mathcal{I}) := \{X \subseteq M : X$ Zirkuit von $\mathcal{I}\}$ Zirkuitsystem von $\mathcal{I}$

6.4 Def.: Opt. prob.

f) $\text{rang}(X) :\Leftrightarrow \max\limits_{B \text{ Basis von } X} |B|$ Rangfunktion von $\mathcal{I}$

g) u-rang (X) $:\Leftrightarrow \min\limits_{B \text{ Basis von } X} |B|$ untere Rangfunktion von $\mathcal{I}$

6.5 Bsp.:

a) $G = (V, E)$ zusammenh., $M = E$, $\mathcal{I} = \{W \subseteq E : W$ Wald$\}$

$\Rightarrow \mathcal{B} = \{$aufspannende Bäume$\}$, $\mathcal{C} = \{$Kreise$\}$ $\max\limits_{X \in \mathcal{B}} c(B)$ max aufsp. Baum, $\max\limits_{X \in \mathcal{I}} c(X)$ max. Wald

$\Rightarrow$ rang $B = |V| - 1$ $\forall B$ Basis, rang $X = |V| - \#$Komp. in (V, X)

b) $M = \{x_1, \dots, x_k \in \mathbb{R}^n\}$, $\mathcal{I} = \{\{x_i, i \in [k]\} : x_i, i \in I$ lin. unabh.$\}\}$

$\Rightarrow \mathcal{B} = \{x_i, i \in [k]$, Basis von span $\bigcup_{i=1}^{k} x_i\}$ $\mathcal{C} = \{$min. Linearkombinationen$\}$

$\Rightarrow$ rang $X = \dim M$ $\forall X$ Basis, rang $X = \dim X$, $\max\limits_{X \in \mathcal{I}} c(X)$?

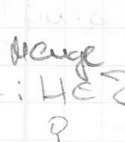
c) $G = (V, E)$, $M = V$, $\mathcal{I} = \{S \subseteq V : S$ stabil$\}$

$\Rightarrow \mathcal{B} = \{$max. stabile Mengen$\}$ $\mathcal{C} = \{\{u, v\} : uv \in E\}$

rang $X = \alpha(G[X])$ $\max\limits_{X \in \mathcal{I}} c(X) = \max\limits_{X \in \mathcal{B}} c(X)$ max. stabile Menge

d) $K_n = (V, E)$, $\mathcal{H} = \{H \subseteq E : H$ Hamiltonkreis$\}$, $\mathcal{I} = \{I \subseteq H : H \in \mathcal{H}\}$

$\Rightarrow \mathcal{B} = \mathcal{H}$, $\mathcal{C} = \{\{wx, wy, wz\} \subseteq E,\ w, x, y, z \in V\}$



Sei $c' := K \cdot \mathbb{1} - c > 0$.

$\Rightarrow$ rang $H = |V|$ $\forall H \in \mathcal{B}$. $\min\limits_{H \in \mathcal{B}} c(H) = \max\limits_{H \in \mathcal{B}} c'(H) = \max\limits_{H \in \mathcal{I}} c'(H)$ TSP

6.6 Proposition (Eigenschaften der Rangfunktion): Für die Rangfunktion eines US $\mathcal{I}$ auf einer Menge M gilt:

a) rang $I \leq |I|$ $\forall I \in \mathcal{I}$ (Subkardinalität)

b) $I \subseteq J \Rightarrow$ rang $I \leq$ rang J $\forall I, J \in \mathcal{I}$ (Monotonie)

c) $k\,\text{rang}\, I_i \leq \sum\limits_{i \in K} \text{rang}\, I_i$ $\forall I \subseteq M$, $I_i \in I$, $i \in K$ (Starke Subadditivität) $|\{i \in K : I_i \supseteq x\}| = k$ $\forall x \in M\}$