2. Klausur zur Vorlesung
Algorithmentechnik
Wintersemester 2009/2010

Lösung!

Beachten Sie:

- Bringen Sie den Aufkleber mit Ihrem Namen und Matrikelnummer auf diesem Deckblatt an und beschriften Sie jedes Aufgabenblatt mit Ihrem Namen und Matrikelnummer.
- Schreiben Sie die Lösungen auf die Aufgabenblätter und Rückseiten. Zusätzliches Papier erhalten Sie bei Bedarf von der Aufsicht.
- Zum Bestehen der Klausur sind **20** der möglichen **60** Punkte hinreichend.
- Es sind keine Hilfsmittel zugelassen.

Aufgabe	Mögliche Punkte					Erreichte Punkte				
	a	b	c	d	Σ	a	b	c	d	Σ
1	4	-	-	-	4		-	-	-	
2	2	3	1	-	6				-	
3	1	3	4	-	8				-	
4	2	1	4	-	7				-	
5	5	-	-	-	5		-	-	-	
6	2	2	3	2	9					
7	3	-	-	-	3		-	-	-	
8	1	3	4	-	8				-	
9	10x1				10					
Σ					60					

Problem 1: B-Zähler 4 Punkte

Ein B-Zähler ist ein Array $A[0 \dots k-1]$ mit $k \in \mathbb{N}$, $k > 0$ und $A[i] \in \{0, 1\}$ für $i = 0, \dots, k-1$. Zu Beginn sei $A[i] = 0$ für alle $i = 0, \dots, k-1$. Wir interpretieren den Inhalt von A als eine binär dargestellte Zahl z, d.h.

$$z := \sum_{i=0}^{k-1} A[i] \cdot 2^i.$$

Die einzige erlaubte Operation ist ERHÖHE. Dabei wird z um 1 erhöht. Das Array sei so implementiert, dass gilt:

- Das **Setzen einer 1** an Stelle i des Arrays verursacht **Kosten von $i+1$**.
- Das Setzen beliebig vieler Stellen des Arrays auf 0 verursacht konstante Kosten von 1.

Zeigen Sie, dass die amortisierten Kosten für n ERHÖHE-Operationen in $O(n)$ liegen. Setzen Sie voraus, dass die Länge des Arrays für die Darstellung der Zahl $z + n$ ausreicht.

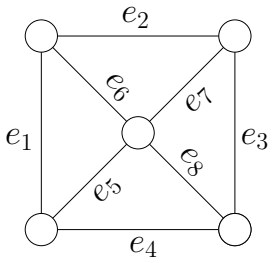
Lösung: Kosten einer Erhöhung:
Eine Erhöhung setzt ein 1-Bit an Stelle i und setzt i Nullbits an den Stellen 0 bis $i-1$. Damit sind die Kosten gleich 1 für $i = 0$, und gleich $(i+1)+1$ sonst.

Für $i > 0$ stehen vor der Erhöhung i 1-Bits im Array (auf den Stellen 0 bis $i-1$), von denen jedes eine vorangegangene Erhöhung impliziert. Jede dieser Erhöhungen hat bereits Kosten 1 für Nullbits bezahlt. Veranschlage also Kosten 2 für das Setzen von Nullbits (davon wird Kosten 1 aufgespart für das Setzen des nächsten 1-Bits bei der nächsten Erhöhung) und Kosten 1 für das Setzen eines 1-Bits. Amortisiert verursacht dann eine Erhöhung Kosten von $1 + 2 = 3$.

Problem 2: Kreisbasen 2 + 3 + 1 = 6 Punkte

(a) Untenstehende Abbildung zeigt den Graphen $G := (V, E)$. Alle Kanten in G haben Gewicht 1.

Geben Sie eine **minimale Kreisbasis** B von G an. Stellen Sie die Basiskreise als Teilmengen von E dar. Zeigen Sie außerdem, dass die von Ihnen angegebene Kreismenge wirklich eine minimale Kreisbasis ist (zu zeigen sind die Minimalität *UND* die Basis-Eigenschaft).



Lösung: Der Baum $\hat{T} := \{e_5, e_6, e_7, e_8\}$ spannt G auf und induziert somit die Fundamentalbasis $B := \{C_1, C_2, C_3, C_4\}$ mit $C_1 := \{e_1, e_5, e_6\}$, $C_2 := \{e_2, e_6, e_7\}$, $C_3 := \{e_3, e_7, e_8\}$, $C_4 := \{e_4, e_5, e_8\}$. Das Gewicht eines minimalen Kreises in G ist 3. Da alle Kreise in B Gewicht 3 haben, ist diese Basis minimal.

(b) Widerlegen Sie folgende Aussage:
Es gibt eine **minimale Kreisbasis** zu G, die von mehreren verschiedenen aufspannenden Bäumen induziert wird.

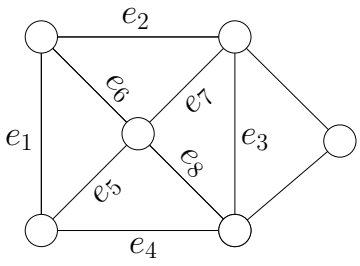
Lösung: In G gibt es genau 4 Kreise der Länge 3 (alle anderen Kreise sind größer). Mit (a) hat jede Basis genau 4 Elemente, d.h. die in (a) angegebene Basis B ist die einzige minimale Basis zu G. Im zugehörigen Spannbaum $\hat{T} := \{e_5, e_6, e_7, e_8\}$ ist jede Baumkante in mindestens zwei Fundamentalkreisen entalten.

Sei nun T' ein weiterer Spannbaum, der B induziert. Eine Baumkante bzgl. $\hat{T}$, die in mehr als einem Fundamentalkreis in B vorkommt, muss auch in T' vorkommen, denn ein Baum, bezüglich dessen eine solche Kante eine Nichtbaumkante ist (und damit in nur genau einem Fundamentalkreis vorkommt!) induziert niemals B.

Damit gilt $\hat{T} \subseteq T'$ und mit $|\hat{T}| = |T'|$ folgt $\hat{T} = T'$.

(c) Erweitern Sie G durch das Einfügen von Kanten (und eventuell Knoten) so zu G', dass gilt:
Es gibt eine Fundamentalbasis zu G', die von mehreren verschiedenen aufspannenden Bäumen induziert wird.
Zeichnen Sie den erweiterten Graphen G' unter diesen Aufgabenteil.

Lösung:



Problem 3: Matroide $1 + 3 + 4 = 8$ Punkte

Gegeben sei ein Matroid $(M,\mathcal{U})$ mit dem zugehörigen Basissystem $\mathcal{B}$.
Weiterhin sei $w : M \to \mathbb{R}^+$ eine Gewichtsfunktion, die jedem Element eine *echt positive* Zahl zuordnet.

Das Problem, eine Basis $I^* \in \mathcal{B}$ zu finden, so dass $w(I^*)$ maximal ist unter allen Elementen aus $\mathcal{B}$, heißt *Maximierungsproblem*.

(a) Geben Sie die Greedy-Methode für das Maximierungsproblem an.

Lösung: Greedy-Methode für das Maximierungsproblem:

Algorithmus 1 : Greedy-Methode für das Maximierungsproblem

```
1 Sortiere M nicht-aufsteigend. Die Sortierung sei ℓ_1, ℓ_2, ..., ℓ_n.
2 I* ← ∅
3 Für i ← 1, ..., n
4 |   Wenn I* ∪ {ℓ_i} ∈ U
5 |   |   I* ← I* ∪ {ℓ_i}
6 return I*
```

(b) Seien nun die Gewichte aller Elemente aus M paarweise verschieden.

Zeigen Sie: Die Optimallösung I^* für das Maximierungsproblem ist eindeutig.

Lösung: Seien I und I' zwei optimale Lösungen mit $I \neq I'$. Dann gibt es in $(I \cup I') \setminus (I \cap I')$ ein kleinstes Element x. O. B. d. A. sei $x \in I$. Außerdem gibt es ein (von x verschiedenes) Element $y \in I' \setminus I$, so dass die Menge $\hat{I} := (I \setminus \{x\}) \cup \{y\}$ wieder eine Basis ist (Austauscheigenschaft und gleiche Elementanzahl aller Basen). Da die Gewichte paarweise verschieden sind folgt: $w(x) < w(y)$ und damit $w(I) < w(\hat{I})$, was einen Widerspruch zur Optimalität von I darstellt.

(c) Gegeben sei ein ungerichteter, einfacher, vollständiger Graph $G = (V, E)$ mit $|V| \geq 3$.
Das heißt, für alle $u, v \in V$ ist $\{u, v\} \in E$ genau dann, wenn $u \neq v$.

Wir definieren das Mengensystem $(E,\mathcal{U})$ mit

$$\mathcal{U} := \{I \subseteq E \mid I \text{ ist Teilmenge eines Hamiltonkreises in } G\}.$$

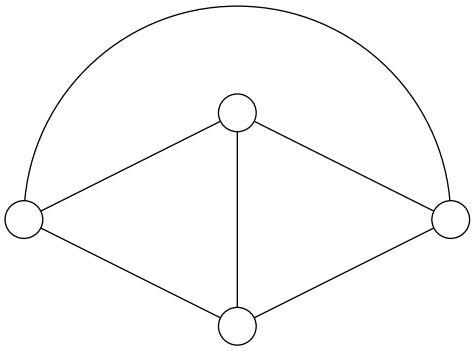
Ein Hamiltonkreis ist dabei ein einfacher Kreis in G, der jeden Knoten genau einmal enthält.

Zeigen Sie: $(E,\mathcal{U})$ ist ein Unabhängkeitssystem.

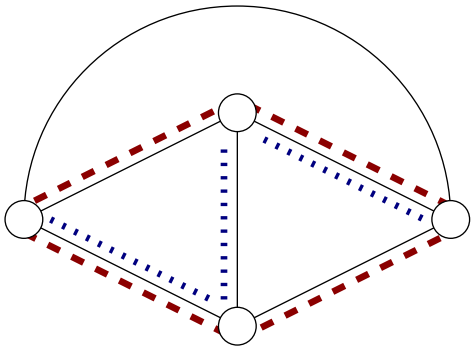
Lösung: Wir überprüfen die Axiome für ein Unabhängigkeitssystem:

- $\emptyset \in \mathcal{U}$:
  Mit $|V| \geq 3$ gibt es mindestens einen (nicht leeren) Hamiltonkreis in G. Damit ist $\emptyset$ Teilmenge eines Hamiltonkreises in G.
- $\forall I \in \mathcal{U}$ und $I' \subseteq I$ folgt $I' \in \mathcal{U}$:
  Sei I Teilmenge eines Hamiltonkreises (Kanten müssen nicht zusammenhängen) und $I' \subseteq I$. So ist I' Teilmenge desselben Hamiltonkreises.

Zeigen Sie an folgendem Gegenbeispiel, dass $(E,\mathcal{U})$ im Allgemeinen kein Matroid ist.
Begründen Sie Ihre Antwort.



Lösung:

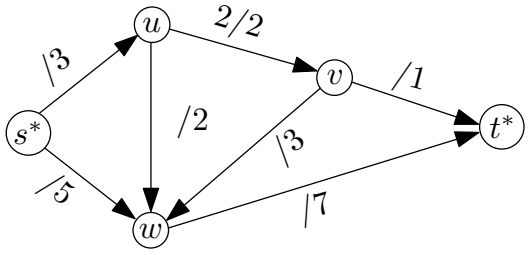


Begründung:

Beide eingezeichneten Kantenmengen sind Teilmengen von Hamiltonkreisen. Jedoch kann die gepunktete Kantenmenge (mit 3 Kanten) nicht mit Kanten aus der gestrichelten Kantenmenge (4 Kanten) erweitert werden, so dass das Resultat eine Teilmenge eines Kreises ergibt, da wir in jedem Falle Grad-3-Knoten erhielten.

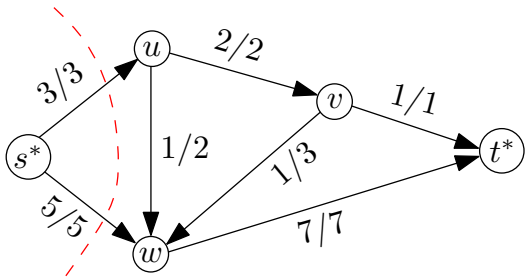
Problem 4: Flussnetzwerke $2 + 1 + 4 = 7$ Punkte

(a) Gegeben sei das untenstehende Netzwerk $(D^*; s^*, t^*; c^*)$, die Kantenkapazitäten $c^*(e)$ sind an den Kanten notiert.

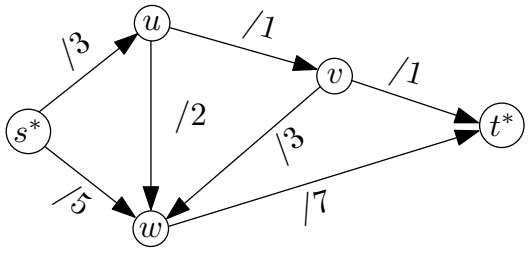


Zeichnen Sie in obiger Abbildung einen maximalen s^*-t^*-Fluss ein. Ergänzen Sie hierzu die Kantenbeschriftung im Format $f(e)/c^*(e)$. Die Kapazität der Kante (u, v) soll dabei **voll ausgenutzt** werden. Beweisen Sie die Maximalität Ihres Flusses, indem Sie einen minimalen Schnitt einzeichnen.

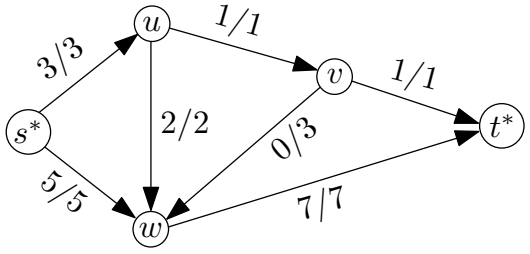
Lösung:



(b) Nun verringere sich die Kapazität der Kante (u, v) um 1. Verändern Sie den nun ungültigen Fluss aus (a), so dass wieder ein gültiger maximaler s^*-t^*-Fluss entsteht. Tragen Sie die neuen Flusswerte in untenstehende Abbildung ein. (Hinweis: Versuchen Sie den in u entstandenen Überschuss umzuleiten).



Lösung:



(c) Gegeben sei nun ein beliebiges Netzwerk $(D; s, t; c)$ mit ganzzahligen Kantenkapazitäten. In $(D; s, t; c)$ existiere eine Kante (s, v) mit ganzzahliger Kapazität $c(s, v) \leq 3$. Weiterhin sei ein maximaler, ganzzahliger s-t-Fluss f gegeben. Nun wird die Kante (s, v) gelöscht.

Beschreiben Sie die Arbeitsweise eines Algorithmus, der nach der Kantenlöschung in $O(|V| + |E|)$-Zeit einen neuen maximalen s-t-Fluss berechnet (kein Code, kein Pseudocode nötig). Begründen Sie die Korrektheit und die Laufzeit.

Lösung:

Algorithmus 2 : FLUSS-UPDATE

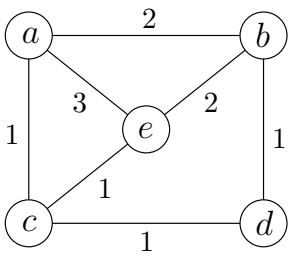
Eingabe : Netzwerk $(D; s; t; c)$, maximaler Fluss f, gelöschte Kante $e = (s, v)$ mit Kapazität $c(e) \leq 3$
Ausgabe : Maximaler Fluss f' im veränderten Netzwerk

```
1 Wenn f(e) > 0
2     solange Defizit in v existiert tue
3         Suche erhöhenden Weg von t nach v
4         Erhöhe Fluss auf Weg um maximal möglichen Wert (≥ 1)
5     solange erhöhender Weg von s nach t existiert tue
6         Erhöhe Fluss auf Weg um maximal möglichen Wert (≥ 1)
```

Falls kein Fluss auf gelöschter Kante ist, bleibt vorheriger Fluss gültig und damit maximal. Sonst entsteht Defizit ≤ 3 in v. Da zuvor Fluss von v nach t floss, gibt es nun erhöhende Wege von t nach v, über die das Defizit abfließen kann. Jeder Weg kann um mindestens 1 erhöht werden (wegen Ganzzahligkeit). Damit genügen maximal 3 Breitensuchen nach Wegen. Nun ist ein neuer gültiger s-t-Fluss entstanden. Dieser wird maximal durch die Suche nach allen erhöhenden Wegen von s nach t. Da der Wert eines maximalen Flusses nach Löschung einer Kante kleiner wird oder maximal gleich bleibt, kann es maximal 3 solcher s-t-Wege geben. Insgesamt ergibt sich ein Aufwand von maximal 6 Breitensuchen. Die Laufzeit der Breitensuche ist in $O(|V| + |E|)$.

Problem 5: Stoer-Wagner-Schnitt-Algorithmus 5 Punkte

Wenden Sie auf den nebenstehend abgebildeten Graphen den Algorithmus von Stoer und Wagner an. Die Kantengewichte sind an den Kanten notiert. Geben Sie nach jeder Phase die Knoten s und t, den Schnitt der Phase und dessen Gewicht an. Zeichnen Sie den nach dem Verschmelzen resultierenden Graphen mit Kantengewichten. Der Startknoten in jeder Phase enthalte a. Geben Sie zum Schluss den minimalen Schnitt $S_{\min}$ an.



Phase 1:

$s =$ $t =$ Graph:

Schnitt der Phase $S_1 =$

$c(S_1, V \setminus S_1) =$

Phase 2:

$s =$ $t =$ Graph:

Schnitt der Phase $S_2 =$

$c(S_2, V \setminus S_2) =$

Phase 3:

$s =$ $t =$ Graph:

Schnitt der Phase $S_3 =$

$c(S_3, V \setminus S_3) =$

Phase 4:

$s =$ $t =$ Graph:

Schnitt der Phase $S_4 =$

$c(S_4, V \setminus S_4) =$

Minimaler Schnitt $S_{\min} =$

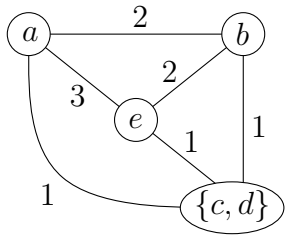
Lösung:

Phase 1:

$s = c \qquad t = d$

Schnitt der Phase $S_1 = (\{d\}, \{a, b, c, e\})$

$c(S_1, V \setminus S_1) = 2$

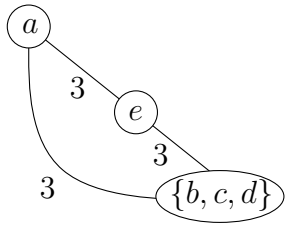


Phase 2:

$s = b \qquad t = \{c, d\}$

Schnitt der Phase $S_2 = (\{c, d\}, \{a, b, e\})$

$c(S_2, V \setminus S_2) = 3$

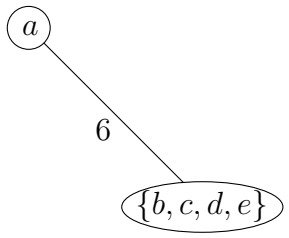


Phase 3:

$s = e \qquad t = \{b, c, d\}$

Schnitt der Phase $S_3 = (\{b, c, d\}, \{a, e\})$

$c(S_3, V \setminus S_3) = 6$

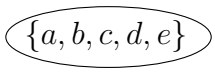


Phase 4:

$s = a \qquad t = \{b, c, d, e\}$

Schnitt der Phase $S_4 = (\{b, c, d, e\}, \{a\})$

$c(S_4, V \setminus S_4) = 6$



Minimaler Schnitt $S_{\min} = S_1$

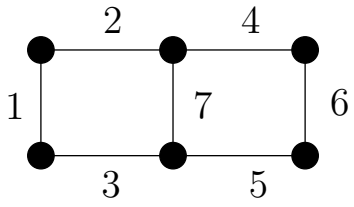
Problem 6: Maximum Spanning Tree $2+2+3+2=9$ Punkte

Gegeben sei ein zusammenhängender, ungerichteter Graph $G = (V, E, c)$ mit paarweise verschiedenen, positiven Kantengewichten $c : E \to \mathbb{R}^+$. Weiter bezeichne $\max(v) \in E$ die maximal gewichtete Kante unter allen zu $v \in V$ inzidenten Kanten. Es sei $S := \{\max(v) \mid v \in V\}$.

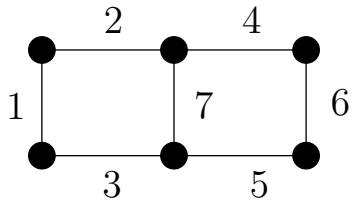
(a) Ein *Spannbaum* in G ist ein Baum $T := (V, E_T)$ mit $E_T \subseteq E$. Ein *maximaler* Spannbaum hat maximales Gewicht unter allen Spannbäumen in G. Zeichen Sie in untenstehenden Beispielgraphen G^* Folgendes ein:

- in (i) die Menge S.
- in (ii) einen maximalen Spannbaum.

Zeichnen Sie hierzu die Kanten jeweils fett ein. Die Kantengewichte sind an den Kanten notiert.

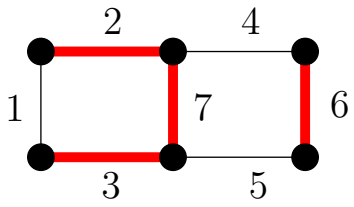


(i) Menge S.

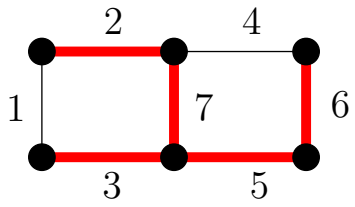


(ii) maximaler Spannbaum.

Lösung:



(i) Menge S.



(ii) maximaler Spannbaum.

(b) Sei $T_{\max} = (V, E_{\max})$ ein maximaler Spannbaum in G.
Zeigen Sie, dass $S \subseteq E_{\max}$ gilt.

Lösung: Annahme: Sei $T_{\max}$ ein maximaler Spannbaum und $e \in S \setminus E_{\max}$. So induziert e einen Fundamentalkreis, dessen (eindeutig) leichteste Kante e' nicht in S liegt, da zu jeder ihrer beiden Knoten eine schwerere Kreiskante inzident ist. Eine Ersetzung von e' durch e im Spannbaum $T_{\max}$ ergibt wieder einen Spannbaum, jedoch mit höherem Gewicht (da $c(\{e'\}) < c(\{e\})$ gilt).

(c) Zeigen Sie, dass $c(E_{\max}) \leq 2\,c(S)$ gilt.

Lösung: Eine Kante in S ist für maximal zwei Knoten die schwerste Inzidenzkante. Damit gilt: $|S| \geq |V|/2$ und mit $S \subseteq E_{\max}$ induziert S maximal $|V|/2$ Zusammenhangskomponenten von $T_{\max}$. Um daraus $T_{\max}$ zu konstruieren braucht man maximal $|S|$ weitere Kanten $\{u, w\}$ mit u, w in verschiedenen Zusammenhangskomponenten. Für $\{u, w\}$ gilt: $c(\{u, w\}) < \min\{\max(u), \max(w)\}$. Da $T_{\max}$ ein Baum ist, dient jedes $\max(v) \in S$ maximal einem Gewicht $c(\{u, w\})$ als obere Schranke. Somit gilt: $c(E_{\max} \setminus S) \leq c(S)$ und $c(E_{\max}) \leq 2\,c(S)$.

(d) Für G bezeichne $\mathrm{OPT}(G)$ das Gewicht eines maximalen Spannbaums in G. Zeigen Sie, dass für untenstehenden Algorithmus $\mathcal{A}$ gilt:

$$\mathrm{OPT}(G) \leq 2 \cdot \mathcal{A}(G),$$

mit $\mathcal{A}(G) := c(E_T)$. (Hinweis: Man kann Teilaufgabe (c) nutzen.)

Algorithmus 3 : APPROX-MAX-SPANNBAUM

Eingabe : Graph $G = (V, E)$
Ausgabe : Baum $T = (V, E_T)$

```
1 S ← ∅
2 für alle v ∈ V tue
3 |  S ← S ∪ {max(v)}
4 E_T ← S
5 solange |E_T| < n − 1 tue
6 |  Wähle e ∈ E \ E_T beliebig, so dass e mit E_T keinen Kreis erzeugt
7 |  E_T ← E_T ∪ {e}
8 return T = (V, E_T)
```

Lösung: Mit (c) und wegen $S \subseteq E_T$ (Zeile 4) gilt: $c(E_{\max}) \leq 2\, c(S) \leq 2\, c(E_T)$.

Problem 7: Randomisierter Schnitt-Algorithmus 3 Punkte

Gegeben sei ein einfacher, ungerichteter Graph $G = (V, E)$.

Folgender randomisierter Algorithmus berechnet einen Schnitt $(S, V \setminus S)$ in G.

Algorithmus 4 : RANDOMCUT

Eingabe : Einfacher, ungerichteter Graph $G = (V, E)$.
Ausgabe : Schnitt $(S, V \setminus S)$ mit $S \subseteq V$.

```
1 S ← ∅
2 für alle v ∈ V tue
3 |   r ← rand({0,1}) ;          // Wähle r zufällig gleichverteilt aus {0, 1}
4 |   Wenn r = 1
5 |   |   S ← S ∪ {v}
6 return (S, V \ S)
```

Berechnen Sie die erwartete Anzahl an Kanten aus E, die den Schnitt $(S, V \setminus S)$ kreuzen.

Lösung: Für jede Kante $e \in E$ sei X_e die Zufallsvariable, die angibt, ob e den Schnitt $(S, V \setminus S)$ kreuzt:

$$\text{Für alle } e \in E: \quad X_e := \begin{cases} 1 & \text{wenn } e \text{ den Schnitt kreuzt,} \\ 0 & \text{sonst.} \end{cases}$$

Die Anzahl Kanten die den Schnitt kreuzen ist dann gerade $X = \sum_{e \in E} X_e$.

Sei $\mathbf{E}[X_e]$ der Erwartungswert, dass $e = \{u, v\}$ den Schnitt kreuzt. Für ihre inzidenten Knoten gibt es folgende vier Fälle bezüglich der Zuweisung zu S bzw. $V \setminus S$.

$$\begin{aligned} u \notin S \text{ und } v \notin S \quad &\Rightarrow \quad e \text{ kreuzt den Schnitt nicht} \\ u \notin S \text{ und } v \in S \quad &\Rightarrow \quad e \text{ kreuzt den Schnitt} \\ u \in S \text{ und } v \notin S \quad &\Rightarrow \quad e \text{ kreuzt den Schnitt} \\ u \in S \text{ und } v \in S \quad &\Rightarrow \quad e \text{ kreuzt den Schnitt nicht} \end{aligned}$$

Da für alle Knoten $v \in V$ gilt, dass $\Pr[v \in S] = 1/2$, sind alle vier oben aufgeführten Fälle gleich wahrscheinlich. Somit ist $\mathbf{E}[X_e] = 1/2$. Insgesamt ergibt sich damit

$$\mathbf{E}[X] = \mathbf{E}[\sum_{e \in E} X_e] = \sum_{e \in E} \mathbf{E}[X_e] = \sum_{e \in E} 1/2 = |E|/2.$$

Problem 8: Kernbildung für MaxSAT — 1 + 3 + 4 = 8 Punkte

Das MaxSAT-Problem aus der Vorlesung ist wie folgt definiert.

Gegeben: Eine Menge von Klauseln C über der Variablenmenge V und ein Parameter k.
Frage: Gibt es eine Wahrheitsbelegung der Variablen aus V, so dass mindestens k Klauseln erfüllt werden?

(a) Betrachten Sie folgende Instanz:

$$V = \{x, y, z\}, \quad C = \{(x \vee \neg z), (\neg x \vee \neg y \vee \neg z), (\neg x \vee \neg y), (x \vee y \vee z)\}, \quad k = 3.$$

Geben Sie eine Wahrheitsbelegung $\beta : V \to \{\texttt{true}, \texttt{false}\}$ an, so dass mindestens 3 Klauseln aus C erfüllt werden.

Lösung:

$$\beta(x) = \texttt{true} \qquad \beta(y) = \texttt{false} \qquad \beta(z) = \texttt{true}$$

(b) Sei k beliebig mit $1 \leq k \leq |C|$.

Sei $C_{\text{large}} \subseteq C$ die Teilmenge der Klauseln aus C, die mindestens k verschiedene Variablen enthalten.

Zeigen Sie: Ist $|C_{\text{large}}| \geq k$, dann gibt es eine Wahrheitsbelegung der Variablen in V, so dass mindestens k Klauseln aus C erfüllt werden.

Lösung: Wir betrachten k beliebige Klauseln $C_1, \ldots C_k$ aus C_{large} und weisen sukzessive für jede Klausel C_i einer unbelegten Variablen aus C_i einen Wert aus $\{\texttt{true}, \texttt{false}\}$ so zu, dass C_i erfüllt wird. Da jede Klausel C_i mindestens k Literale enthält, gibt es immer eine Variable in C_i die noch keinen Wert zugewiesen bekommen hat. Wir erfüllen mit der so gefundenen Variablenbelegung also mindestens k Klauseln aus C_{large}, und somit auch aus C.

(c) Sei $k \leq \lfloor |C|/2 \rfloor$.

Geben Sie einen Algorithmus mit polynomieller Laufzeit in der Anzahl $|V| + |C|$ der Variablen und Klauseln an (kein Pseudocode erforderlich), der eine Wahrheitsbelegung der Variablen findet, so dass mindestens k Klauseln aus C erfüllt sind.

Lösung: Algorithmus 5 findet eine gültige Wahrheitsbelegung.

Algorithmus 5 : Wahrheitsbelegung für $k \leq \lfloor |C|/2 \rfloor$

Eingabe : Klauselmenge C über V, Parameter k
Ausgabe : Wahrheitsbelegung β, die mindestens k Klauseln aus C erfüllt

```
für alle v ∈ V tue
    β(v) ← 0
Wenn β erfüllt weniger als k Klauseln
    für alle v ∈ V tue
        β(v) ← ¬β(v)
return β
```

Wenn β bereits k Klauseln erfüllt, sind wir fertig. Andernfalls, wenn β weniger als $k \leq \lfloor |C|/2 \rfloor$ Klauseln erfüllt, dann erfüllt $\neg\beta$ mindestens $\lfloor |C|/2 \rfloor \geq k$ Klauseln. Fertig. Der Aufwand des Verfahrens ist in $\mathcal{O}(|V| + |C||V|)$, also polynomiell in $|V| + |C|$.

Problem 9: $10 \times 1 = 10$ Punkte

Kreuzen Sie für folgende Aussagen an, ob diese wahr oder falsch sind. Für jede richtige Antwort gibt es einen Punkt, für jede falsche Antwort wird ein Punkt abgezogen. Es wird keine negative Gesamtpunktzahl für diese Aufgabe geben.

	Wahr	Falsch
Sei (PP) ein lineares Programm und (DP) das zu (PP) duale Programm. Hat (PP) eine zulässige Lösung, dann hat auch immer (DP) eine zulässige Lösung.	☐	☒
Gegeben sei das CRCW-PRAM Modell, wobei mehrere Prozessoren genau dann gleichzeitig in eine Speicherstelle schreiben dürfen, wenn sie denselben Wert schreiben wollen. In diesem Modell gibt es einen Algorithmus, der das logische Oder von n booleschen Variablen mit n Prozessoren in Zeit $\mathcal{O}(1)$ berechnen kann.	☒	☐
Sei A die Array-Darstellung eines Heaps. Dann ist A entweder nicht-absteigend oder nicht-aufsteigend sortiert.	☐	☒
Seien B_1 und B_2 verschiedene minimale Spannbäume eines ungerichteten, gewichteten Graphen G und e eine Kante maximalen Gewichts von B_1. Dann gibt es eine Kante $e' \in G \setminus B_1$, so dass $B_2 = B_1 \setminus \{e\} \cup \{e'\}$ gilt.	☐	☒
Sei $(D; s, t; c)$ ein Flussnetzwerk. Bevor der Algorithmus von Goldberg und Tarjan ausgeführt wird, kann für das Label $\text{dist}(s)$ der Quelle s im Allgemeinen keine obere Schranke angegeben werden.	☐	☒
Sei P ein ILP und sei M das Lösungspolyeder zur Relaxierung von P. Falls M nicht leer ist, dann sind alle Ecken von M ganzzahlig.	☐	☒
Sei $G = (V, E)$ ein gewichteter, ungerichteter, einfacher Graph und e eine minimalgewichtete Kante aus E. Dann ist e in jedem MST enthalten.	☐	☒
Es gibt eine (unendliche) Familie $(G_i)_{i \in \mathbb{N}}$ von Graphen, so dass für jeden Graphen G_i die Anzahl der Kreisbasen in G_i exponentiell in der Anzahl der Knoten von G_i ist.	☒	☐
Sei Π ein Problem, k ein Parameter und n die Eingabelänge. Sei $\mathcal{A}$ ein Lösungsalgorithmus für Π, der eine erschöpfende Suche in einem binären Suchbaum der Tiefe 2^k durchführt. Weiterhin sei der Aufwand pro Baumknoten $p(n)$, wobei $p(n)$ ein Polynom ist, das ausschließlich von n abhängt. Dann ist Π fixed-parameter-tractable.	☒	☐
Jeder $\mathcal{RP}$-Algorithmus ist ein $\mathcal{BPP}$-Algorithmus.	☐	☒