

1. Klausur zur Vorlesung  
Algorithmentechnik  
Wintersemester 2008/2009

# Lösung!

**Beachten Sie:**

- Bringen Sie den Aufkleber mit Ihrem Namen und Matrikelnummer auf diesem Deckblatt an und beschriften Sie jedes Aufgabenblatt mit Ihren 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	$\Sigma$	a	b	c	d	$\Sigma$
1	4	-	-	-	4		-	-	-	
2	2	1	4	1	8					
3	2	2	2	-	6				-	
4	3	1	3	-	7				-	
5	2	3	2	-	7				-	
6	1	2	2	-	5				-	
7	1	4	-	-	5			-	-	
8	2	1	3	-	6				-	
9	12x1				12					
$\Sigma$					60					

**Problem 1: Ternärzähler**

4 Punkte

Ein  $k$ -Ternärzähler ist ein Array  $A[0 \dots k-1]$  mit  $A[i] \in \{0, 1, 2\}$  für  $i = 0, \dots, k-1$ . Zu Beginn sei  $A[i] = 0$  für  $i = 0, \dots, k-1$ . Wir interpretieren den Inhalt von  $A[]$  als eine im Dreiersystem dargestellte Zahl  $z$ , d.h.

$$z := \sum_{i=0}^{k-1} A[i] \cdot 3^i$$

Die einzig erlaubte Operation für den Ternärzähler  $A[]$  ist die Operation ERHÖHE, die die dargestellte Zahl um 1 erhöht. Jede Änderung eines Eintrages in  $A[]$  verursacht Kosten von 1. Zeigen oder widerlegen Sie: Die amortisierten Kosten für  $n$  ERHÖHE-Operationen liegen in  $O(n)$ .

*Lösung:* Die Kosten für  $n$  ERHÖHE-Operationen sind

$$\sum_{i=0}^{k-1} \lfloor \frac{n}{3^i} \rfloor \leq n \cdot \sum_{i=0}^{\infty} \left(\frac{1}{3}\right)^i \in O(n)$$

**Problem 2: Union-Find / Baumpartition**

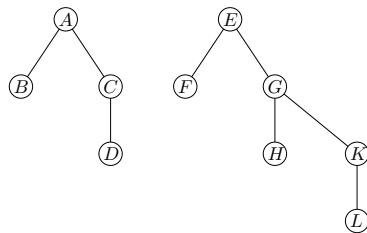
2 + 1 + 4 + 1 = 8 Punkte

(a) Betrachten Sie die unten abgebildete Union-Find-Datenstruktur. Es sollen nun nacheinander die Operationen

- (i) UNION(FIND( $B$ ), FIND( $G$ )) und
- (ii) FIND( $D$ )

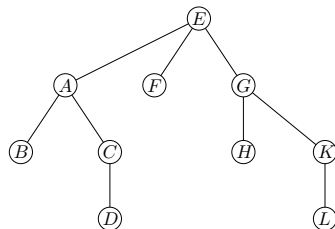
ausgeführt werden.

Zeichnen Sie den Zustand der Datenstruktur jeweils nach Ausführung der Operation (i) und (ii). Die Datenstruktur sei mit Weighted-Union und Pfadkompression implementiert.

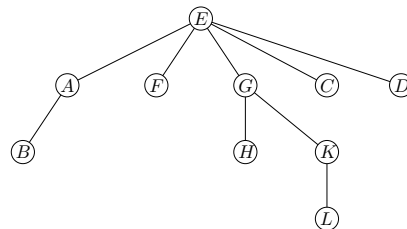


*Lösung:*

nach (i)



nach (ii)



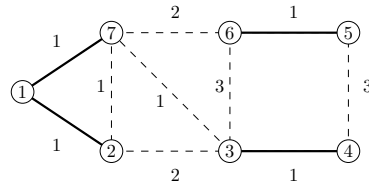
Eine *Baum-Partition* eines gewichteten Graphen  $G = (V, E)$  mit Gewichtsfunktion  $w : E \rightarrow \mathbb{R}^+$  und Parameter  $\gamma \in \mathbb{R}^+$  ist eine Menge von Kanten  $F \subseteq E$  mit folgenden Eigenschaften:

- (i)  $F$  induziert einen Wald in  $G$ .
- (ii) Für jede Menge  $F' \subseteq F$ , die einen Baum induziert, gilt

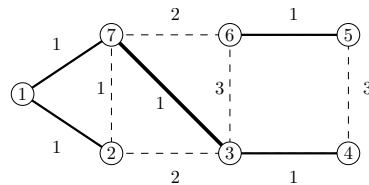
$$w(F') := \sum_{e \in F'} w(e) \leq \gamma.$$

Eine Baum-Partition mit Parameter  $\gamma$  heißt *nicht-erweiterbar*, wenn gilt

- Für alle  $e \in E \setminus F$  gilt:  $F \cup \{e\}$  ist keine Baumpartition mit Parameter  $\gamma$
- (b) Zeigen oder widerlegen Sie, dass die Menge der fett markierten Kanten im folgenden Graphen eine nicht-erweiterbare Baum-Partition des Graphen mit  $\gamma = 4$  ist. Die Zahlen, mit denen die Kanten annotiert sind, entsprechen den Gewichten der Kanten.



*Lösung:* Die Menge der fett markierten Kanten ist keine nicht-erweiterbare Baum-Partition, da die Kante  $\{3, 7\}$  zur Baum-Partition hinzugenommen werden kann, ohne dass der entstehende Baum das maximale Gewicht 4 überschreitet.



- (c) Formulieren Sie mithilfe der Union-Find-Datenstruktur aus der Vorlesung einen Algorithmus, der eine nicht-erweiterbare Baum-Partition zu einem gegebenen gewichteten Graphen  $G = (V, E)$  mit Gewichtsfunktion  $w : E \rightarrow \mathbb{R}^+$  und einem gegebenen Parameter  $\gamma \in \mathbb{R}^+$  berechnet.

Die Ausgabe des Algorithmus sei die Union-Find-Datenstruktur, so dass der Ausdruck  $\text{FIND}(u) = \text{FIND}(v)$  genau dann wahr ist, wenn die Knoten  $u$  und  $v$  im gleichen Baum der Baum-Partition liegen.

*Lösung:*

**Algorithmus 1** : Tree Partition (Baum-Partition)**Eingabe** : Graph  $G = (V, E)$ , Gewichtsfunktion  $w : E \rightarrow \mathbb{R}$ , Schwellwert  $\gamma$ **Ausgabe** : Union-Find Datenstruktur die nicht-erweiterbare Baum-Partition für  $G$  darstellt**Daten** : Array  $A$ 

```

1 Für  $v \in V$ 
2   MAKESET( $v$ )
3    $A[\text{FIND}(v)] \leftarrow 0$ 
4 Für  $\{u, v\} \in E$ 
5    $x \leftarrow \text{FIND}(u)$ 
6    $y \leftarrow \text{FIND}(v)$ 
7   Wenn  $x \neq y$  und  $A[x] + A[y] + w(u, v) \leq \gamma$ 
8     UNION( $u, v$ )
9      $A[\text{FIND}(u)] \leftarrow A[x] + A[y] + w(u, v)$ 

```

- (d) Geben Sie eine möglichst gute obere Schranke für die asymptotische Laufzeit Ihres Algorithmus an. Begründen Sie Ihre Antwort.

*Lösung:* Die mit Union-Find zu verwaltende Menge hat  $n$  Elemente. In der ersten Schleife werden  $n$  MAKESET- und FIND-Operationen mit jeweils konstanter Laufzeit ausgeführt. In der zweiten Schleife werden für jede Kante zwei FIND-Operationen durchgeführt. Der Test  $x \neq y$  in Zeile 8 fällt höchstens  $\min\{m, n-1\} \in \mathcal{O}(m)$  mal positiv aus, so dass maximal weitere  $\min\{m, n-1\}$  UNION- und FIND-Operationen hinzukommen. In der zweiten Schleife werden also maximal  $\mathcal{O}(m)$  UNION- und FIND-Operationen auf  $n$  Elementen ausgeführt. Somit ergibt sich eine Gesamtlaufzeit in  $\mathcal{O}(m\alpha(m, n) + n)$ . Etwas schwächer analysiert ergibt sich auch eine Laufzeit in  $\mathcal{O}((n+m)G(n))$ .

**Problem 3:** Matroide

2 + 2 + 2 = 6 Punkte

- (a) Sei  $M$  eine Menge mit  $m$  Elementen. Zu  $r \in \mathbb{N}$  mit  $r \leq m$  sei  $\mathcal{U}_r := \{S \subseteq M : |S| \leq r\}$ . Zeigen Sie, dass für alle  $r \leq m$  das Tupel  $(M, \mathcal{U}_r)$  ein Matroid ist.

*Lösung:*

- (i)  $\emptyset \in \mathcal{U}_r$ , da für alle  $r \in \mathbb{N}$  gilt:  $0 = |\emptyset| \leq r$
- (ii) Sei  $B \in \mathcal{U}$  und  $A \subseteq B$ , dann gilt  $|A| \leq |B| \leq r$  und somit  $A \in \mathcal{U}$ .
- (iii) Seien  $A, B \in \mathcal{U}$ ,  $|B| > |A|$  und sei  $x \in B \setminus A$ . Dann ist  $|A \cup \{x\}| = |A| + 1 \leq |B| = |B| \leq r$  und somit  $A \cup \{x\} \in \mathcal{U}$ .
- (b) Sei  $\mathcal{M} = (S, \mathcal{U})$  ein Matroid und sei  $\mathcal{B} \subseteq \mathcal{U}$  die Menge der Basen von  $\mathcal{M}$ , d.h. die Menge der maximalen unabhängigen Mengen in  $\mathcal{U}$ . Seien  $B_1, B_2 \in \mathcal{B}$  mit  $B_1 \neq B_2$  und sei  $x \in B_1 \setminus B_2$ . Zeigen Sie, dass es ein  $y \in B_2 \setminus B_1$  gibt, so dass  $(B_1 \setminus \{x\}) \cup \{y\} \in \mathcal{B}$  gilt.

*Lösung:* Wende die Austauscheigenschaft auf  $B_1 \setminus \{x\}$  und  $B_2$  an: Die Austauscheigenschaft besagt, dass es ein  $y \in B_2 \setminus (B_1 \setminus \{x\})$  gibt, so dass  $(B_1 \setminus \{x\}) \cup \{y\} \in \mathcal{U}$  gilt. Wegen  $|(B_1 \setminus \{x\}) \cup \{y\}| = |B_1|$  ist diese Menge wieder in  $\mathcal{B}$ .

- (c) Sei  $G = (V, E)$  ein zusammenhängender Graph und seien  $T_1, T_2$  zwei aufspannende Bäume in  $G$ . Zu  $T_1, T_2$  bezeichne  $E(T_1)$  und  $E(T_2)$  jeweils die Menge der Kanten. Sei  $e_1$  eine beliebige Kante in  $E(T_1)$ .

Zeigen Sie: Dann gibt es eine Kante  $e_2$  in  $E(T_2)$ , so dass  $(E(T_1) \setminus \{e_1\}) \cup \{e_2\}$  einen aufspannenden Baum in  $G$  induziert.

Anmerkung: Aus der Vorlesung ist bekannt, dass das System aller Teilmengen  $E' \subseteq E$ , die einen Wald in  $G$  induzieren, ein Matroid ist.

*Lösung:* Aus der Vorlesung ist bekannt, dass die Kantenmengen, die Wälder in  $G$  induzieren ein Matroid bilden, dessen Basen gerade die aufspannenden Bäume von  $G$  sind. Wir können also Aufgabenteil (b) anwenden, um die Aussage zu erhalten.

#### Problem 4: Zulässige Gradsequenz

3 + 1 + 3 = 7 Punkte

- (a) Sei  $\mathcal{N}$  ein Flussnetzwerk bestehend aus einem einfachen, gerichteten Graphen  $D = (V, E)$  mit ausgezeichneten Knoten  $s$  (Quelle) und  $t$  (Senke) sowie ganzzahligen Kantenkapazitäten  $c: E \rightarrow \mathbb{N}$ .

Zeigen Sie: Der Algorithmus von Goldberg-Tarjan berechnet einen ganzzahligen Maximalfluss.

*Lösung:* In diesem Fall sind alle Zahlen, die der Algorithmus als Eingabe bekommt ganzzahlig. Auf diese Zahlen werden nur die Operationen  $+$ ,  $-$ ,  $\min$  angewandt. Diese sind abgeschlossen bezüglich der Menge der ganzen Zahlen. Damit sind alle Zahlenwerte die während der Ausführung des Algorithmus vorkommen ganzzahlig, insbesondere auch die Ausgabe.

- (b) Gegeben sei eine Menge von Knoten  $V = \{v_1, \dots, v_n\}$  sowie Eingangsgrade  $e_1, \dots, e_n \in \{0, \dots, n-1\}$  und Ausgangsgrade  $a_1, \dots, a_n \in \{0, \dots, n-1\}$ . Das Problem ZULÄSSIGE GRADSEQUENZ besteht nun darin zu entscheiden, ob es einen gerichteten Graphen  $G = (V, E)$  ohne Schleifen und Mehrfachkanten gibt, so dass Knoten  $v_j$  Eingangsgrad  $e_j$  und Ausgangsgrad  $a_j$  für alle  $j = 1, \dots, n$  hat.

Gegeben seien die Eingangsgrade  $e_1 = 1, e_2 = 1, e_3 = 1$  und Ausgangsgrade  $a_1 = 0, a_2 = 2, a_3 = 0$ . Zeigen oder widerlegen Sie, dass es einen gerichteten Graphen ohne Schleifen und Mehrfachkanten mit den gewünschten Graden gibt.

*Lösung:* Notwendige Bedingung:  $\sum_{i=1}^n e_i = \sum_{i=1}^n a_i$ . Diese Bedingung ist im Beispiel verletzt.

- (c) Modellieren Sie das Problem ZULÄSSIGE GRADSEQUENZ als Flussproblem. Die Menge der Knoten des Flussnetzwerkes sei  $W := \{s\} \cup \{\vec{v}_i, \overleftarrow{v}_i \mid 1 \leq i \leq n\} \cup \{t\}$ . Geben Sie das Flussnetzwerk formal an. Welche Bedingung muss ein maximaler Fluss  $f$  in Ihrem Netzwerk erfüllen, damit ein Graph mit den gewünschten Eigenschaften existiert?

*Lösung:* Das Flussnetzwerk besteht aus dem Graphen  $D = (W, A)$  mit

$$A := \{(s, \vec{v}_i) \mid 1 \leq i \leq n\} \cup \{(\overleftarrow{v}_i, t) \mid 1 \leq i \leq n\} \cup \{(\vec{v}_i, \overleftarrow{v}_j) \mid i \neq j\}$$

Für die Kapazitäten gilt

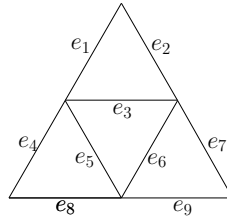
$$c(v, w) := \begin{cases} 1 & \text{falls } \{v, w\} \cap \{s, t\} = \emptyset \\ a_i & \text{falls } v = s \text{ und } w = \vec{v}_i \\ e_i & \text{falls } v = \overleftarrow{v}_i \text{ und } w = t \end{cases} \quad (1)$$

Es gibt genau dann einen einfachen, gerichteten Graphen  $G$  mit den gewünschten Graden, wenn der Maximalfluss in dem konstruierten Netzwerk den Wert  $\sum_{i=1}^n e_i = \sum_{i=1}^n a_i$  hat.

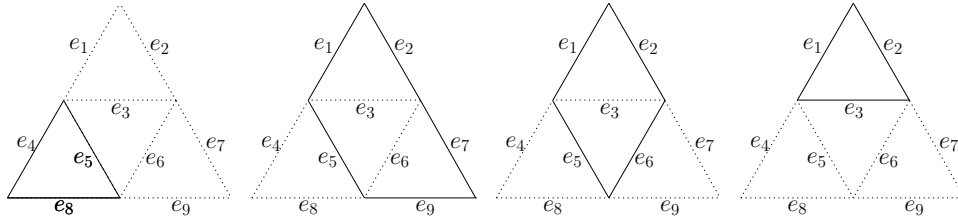
#### Problem 5: Kreisbasen

2 + 3 + 2 = 7 Punkte

Gegeben ist der folgende Graph  $G = (V, E)$ :



- (a) Außerdem ist folgende Fundamentalkreisbasis  $\{e_4, e_5, e_8\}, \{e_1, e_2, e_5, e_7, e_9\}, \{e_1, e_2, e_5, e_6\}, \{e_1, e_2, e_3\}$  von  $G$  gegeben:



Geben Sie alle Spannbäume von  $G$  an, die diese Fundamentalkreisbasis induzieren und argumentieren Sie, dass es keine weiteren gibt.

*Lösung:* Sei  $B$  die Fundamentalkreisbasis. Sei  $T$  ein Spannbaum der  $B$  erzeugt. Jede Kante, die in mindestens zwei Kreisen vorkommt muss in  $T$  enthalten sein, also  $e_1, e_2$  und  $e_5$ . Die Kanten  $e_3$  und  $e_6$  können nicht in  $T$  enthalten sein, sonst wäre  $T$  kein Baum. Übrig bleiben die Kanten  $e_4, e_7, e_8$  und  $e_9$  von denen genau eine Kante aus  $\{e_4, e_8\}$  und genau eine Kante aus  $\{e_7, e_9\}$  in  $T$  enthalten sein muss damit  $T$  ein Spannbaum ist. Jede dieser Lösungen induziert  $B$ . Lösung ist also  $\{e_1, e_2, e_5, e_4, e_7\}, \{e_1, e_2, e_5, e_4, e_9\}, \{e_1, e_2, e_5, e_8, e_7\}$  und  $\{e_1, e_2, e_5, e_8, e_9\}$ .

- (b) Alle Kanten in  $G$  haben einheitliches Gewicht. Geben Sie eine minimale Kreisbasis  $B$  von  $G$  an. Zeigen Sie zusätzlich, dass  $B$  eine minimale Kreisbasis ist (zu zeigen sind Minimalität UND die Kreisbasen-Eigenschaft).

*Lösung:*

Wir zeigen:  $B = \{\{e_1, e_2, e_3\}, \{e_4, e_5, e_8\}, \{e_3, e_5, e_6\}, \{e_6, e_7, e_9\}\}$  ist MCB. Jede Kreisbasis von  $G$  hat  $|E| - |V| + 1 = 4$  Elemente. Es ist  $|B| = 4$  also ist  $B$  genau dann eine Basis, wenn die Einzelvektoren linear unabhängig sind. Offensichtlich hat

$$\alpha \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} + \gamma \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \delta \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} = 0$$

nur  $\alpha = \beta = \gamma = \delta = 0$  als Lösung. Damit sind die Vektoren in  $B$  linear unabhängig.

Jeder Kreis in  $G$  hat Gewicht mindestens 3. Eine Kreisbasis (mit 4 Elementen) hat also Gewicht mindestens 12. Die Kreisbasis  $B$  hat Gewicht 12 und ist damit minimal.

- (c) Gegeben seien  $t$  ungerichtete, zusammenhängende Graphen  $G_1 = (V_1, E_1), \dots, G_t = (V_t, E_t)$  mit Kreisbasen  $B_1, \dots, B_t$ . Diese Graphen werden zu einem Graphen  $G = (V_1 \cup \dots \cup V_t, E_1 \cup \dots \cup E_t \cup \tilde{E})$  vereinigt. Die zusätzlichen Kanten  $\tilde{E}$  werden so eingefügt, dass  $G$  zusammenhängend ist. Beachten Sie, dass dabei neue Kreise entstehen können.

Geben Sie eine Formel für die Kardinalität einer Kreisbasis von  $G$  in Abhängigkeit der gegebenen Kreisbasen  $B_1, \dots, B_t$ , der Kantenmenge  $\tilde{E}$  und des Parameters  $t$  an.

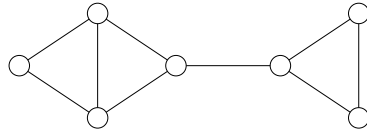
*Lösung:* Die Kardinalität  $K$  einer Kreisbasis von  $G = (V, E)$  ist  $|E| - |V| + 1 = \sum_{i=1}^t |E_i| + |\tilde{E}| - \sum_{i=1}^t |V_i| + 1$ . Wegen  $|B_i| = |E_i| - |V_i| + 1$  für  $1 \leq i \leq t$  ist  $\sum_{i=1}^t |B_i| = \sum_{i=1}^t |E_i| - \sum_{i=1}^t |V_i| + t$  und damit ist  $K = \sum_{i=1}^t |B_i| - t + 1 + |\tilde{E}|$ .

### Problem 6: Unabhängige Menge

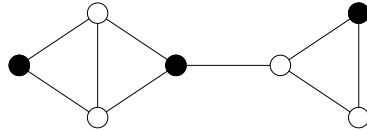
1 + 2 + 2 = 5 Punkte

Gegeben sei ein Graph  $G = (V, E)$ . Eine *unabhängige Menge* von  $G$  ist eine Menge von Knoten  $V' \subseteq V$ , so dass für jede Kante in  $G$  höchstens ein Endknoten in  $V'$  ist. Das Optimierungsproblem UNABHÄNGIGE MENGE besteht darin, eine unabhängige Menge maximaler Kardinalität zu finden.

- (a) Zeichnen Sie eine unabhängige Menge maximaler Kardinalität in den folgenden Graphen ein.



*Lösung:*



- (b) Formulieren Sie das Problem UNABHÄNGIGE MENGE als ILP. Erklären Sie die Bedeutung der Variablen und Nebenbedingungen.

*Lösung:*

$$\max \sum_{v \in V} X_v \quad (2)$$

$$X_v \in \{0, 1\} \quad v \in V \quad (3)$$

$$X_u + X_v \leq 1 \quad \{u, v\} \in E \quad (4)$$

$X_v$  hat den Wert 1 genau dann, wenn  $v$  in der maximalen unabhängigen Menge enthalten ist (ansonsten hat  $X_v$  den Wert 0). Für jede Kante gibt es eine Nebenbedingung, die sicherstellt, dass es keine benachbarten Knoten in der Lösungsmenge gibt. Die Zielfunktion maximiert die Anzahl der Knoten in der gesuchten Menge.

(c) Betrachten Sie das folgende Lineare Programm:

$$\min \sum_{\{u,v\} \in E} X_{\{u,v\}} \quad (5)$$

$$X_{\{u,v\}} \in \{0, 1\} \quad \{u, v\} \in E \quad (6)$$

$$\sum_{u: \{u,v\} \in E} X_{\{u,v\}} \geq 1 \quad v \in V \quad (7)$$

Geben Sie jeweils eine sinnvolle Interpretation der Variablen und Nebenbedingungen an und interpretieren Sie die Zielfunktion des Programms entsprechend.

*Lösung:*

Für eine Kante  $\{u, v\} \in E$  sei  $X_{\{u,v\}} = 1$  genau dann, wenn  $\{u, v\}$  in der gesuchten Menge liegt. Für jeden Knoten  $v$  gibt es eine Nebenbedingung, die sicherstellt, dass mindestens eine zu  $v$  inzidente Kante in der gesuchten Menge liegt. Die Zielfunktion minimiert die Anzahl der Kanten in der gesuchten Menge. Das Programm versucht also, die Knoten des Graphen mit möglichst wenigen Kanten zu überdecken.

**Problem 7:** Minimaler Metrischer Steiner-Baum

1 + 4 = 5 Punkte

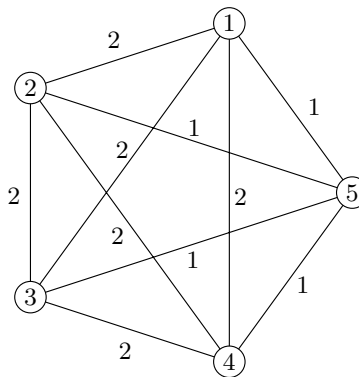
Gegeben sei ein vollständiger Graph  $G = (V, E)$  und eine Knotenmenge  $S \subset V$ , sowie eine Gewichtsfunktion  $w : E \rightarrow \mathbb{N}$ , die die Dreiecksungleichung erfüllt, d.h. für alle Knoten  $u, v, x \in V$  gilt  $w(u, x) \leq w(u, v) + w(v, x)$ .

Gesucht ist ein minimaler Steiner-Baum  $T$ , d.h. ein Baum in  $G$ , der alle Knoten aus  $S$  enthält, so dass das Gewicht

$$w(T) := \sum_{e \in E(T)} w(e)$$

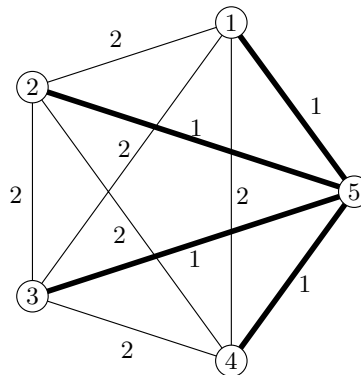
minimal ist. Beachten Sie, dass  $T$  nicht alle Knoten aus  $V$  enthalten muss.

(a) Gegeben sei der folgende Graph und  $S := \{1, 2, 3, 4\}$ . Die Gewichte sind direkt an den Kanten annotiert. Zeichnen Sie einen minimalen Steiner-Baum ein.



*Lösung:*





(b) Betrachten Sie den folgenden Algorithmus:

---

**Algorithmus 2** : Steiner-Baum

---

**Eingabe** : vollständiger Graph  $G = (V, E)$ ,  $S \subseteq V$ , Gewichtsfunktion  $w : E \rightarrow \mathbb{R}$ , die die Dreiecksungleichung erfüllt

**Ausgabe** : Steiner-Baum  $T$  für  $S$

- 1  $E' \leftarrow \{\{u, v\} \mid \{u, v\} \subseteq S\}$
  - 2  $G' \leftarrow (S, E')$
  - 3  $T \leftarrow$  Spannbaum minimalen Gewichts in  $G'$
- 

Zeigen Sie, dass Algorithmus 2 ein 2-Approximationsalgorithmus ist, d.h. dass

$$\mathcal{A}(I) \leq 2 \cdot \text{OPT}(I)$$

gilt. Hierbei sei  $I$  eine Instanz des Problems,  $\mathcal{A}(I)$  das Gewicht des von Algorithmus 2 zurückgegebenen Baumes sowie  $\text{OPT}(I)$  das Gewicht einer optimalen Lösung.

*Hinweis:* Betrachten Sie die Rundtour, die durch das Traversieren eines optimalen Steiner-Baumes induziert wird.

*Lösung:* Sei  $T^*$  ein optimaler Steinerbaum und sei  $C$  die Rundtour, die durch das Traversieren von  $T^*$  induziert wird. Dann gilt  $w(C) \leq 2w(T^*) = 2 \cdot \text{OPT}(I)$ . Wenn man Knoten aus  $G \setminus S$  aus  $C$  entfernt, erhält man eine Tour  $C'$ , die aufgrund der Dreiecksungleichung geringeres Gewicht als  $C$  hat, d.h.  $w(C') \leq w(C)$ . Entfernt man eine Kante  $e$  aus  $C'$ , so erhält man einen Spannbaum  $T'$  in  $G'$ . Für diesen gilt  $w(T') \geq w(T)$ , da  $T$  ein minimaler Spannbaum in  $G'$  ist. Somit gilt:

$$\mathcal{A}(I) = w(T) \leq w(T') \leq w(C') \leq w(C) \leq 2w(T^*) = 2\text{OPT}(I)$$

**Problem 8:** Randomisierte Algorithmen

2 + 1 + 3 = 6 Punkte

Gegeben sei folgender Las-Vegas Algorithmus UNBEKANNTERALGORITHMUS( $A[]$ ). Die Eingabe des Algorithmus sei ein Array  $A[]$  mit Einträgen  $A[1], \dots, A[n] \in \mathbb{N}$ . Die Einträge von  $A[]$  seien paarweise verschieden.

**Algorithmus 3 :** UNBEKANNTERALGORITHMUS( $A[]$ )**Input :** Ein Array  $A[]$  von  $n$  paarweise verschiedenen Zahlen**Output :** Eine Zahl

```

1  $r \leftarrow \text{Random}(1,n)$ 
2  $m \leftarrow A[r]$ 
3  $\text{found} \leftarrow \text{true}$ 
4 for  $i=1$  to  $n$  do
5   | if  $A[i] > m$  then
6   | |  $\text{found} \leftarrow \text{false}$ 
7 if  $\text{found}=\text{true}$  then
8   | return  $m$ 
9 else
10  | return UNBEKANNTERALGORITHMUS( $A[]$ )

```

(a) Was berechnet UNBEKANNTERALGORITHMUS( $A[]$ )?*Lösung:* Der Algorithmus berechnet das Maximum von  $\{A[1], A[2], \dots, A[n]\}$ .(b) Geben Sie die Wahrscheinlichkeit an, dass sich UNBEKANNTERALGORITHMUS( $A[]$ ) genau 10 mal selbst rekursiv aufruft (also Zeile 10 genau 10 mal ausgeführt wird).*Lösung:* Die Wahrscheinlichkeit ist  $\left(\frac{n-1}{n}\right)^{10} \frac{1}{n}$ .(c) Was ist die erwartete Anzahl an rekursiven Selbstaufrufen von UNBEKANNTERALGORITHMUS( $A[]$ )? Begründen Sie Ihr Ergebnis.*Lösung:* Sei  $p := 1/n$  und  $q := 1 - p$ . Die erwartete Anzahl an rekursiven Selbstaufrufen ist dann

$$\sum_{k=0}^{\infty} k \cdot q^k p = pq \sum_{k=0}^{\infty} k \cdot q^{k-1} = pq \left( \sum_{k=0}^{\infty} q^k \right)' = pq \left( \frac{1}{1-q} \right)' = pq \frac{1}{(1-q)^2} = \frac{1-p}{p} = n-1$$

**Problem 9:**

12 × 1 = 12 Punkte

Kreuzen Sie für folgende Aussagen an, ob diese wahr oder falsch sind.

*Hinweis:* 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.

Bei amortisierter Analyse wird die erwartete Laufzeit über alle Eingaben gemittelt.

<input type="checkbox"/>	<input checked="" type="checkbox"/>
Wahr	Falsch

Der parallele Algorithmus  $\mathcal{A}$  benötige zum Sortieren eines Arrays mit  $n$  Elementen und  $m$  Prozessoren die Laufzeit  $O(\frac{n \log n}{\log m})$ . Dies ist kostenoptimal.

<input type="checkbox"/>	<input checked="" type="checkbox"/>
Wahr	Falsch

Sei  $P$  ein ILP für ein ganzzahliges Minimierungsproblem und sei  $R$  die Relaxierung von  $P$ . Dann ist die optimale Lösung von  $R$  eine untere Schranke für die optimale Lösung von  $P$ .

<input checked="" type="checkbox"/>	<input type="checkbox"/>
Wahr	Falsch

Der Algorithmus FIRST FIT (FF) ist ein APAS für Bin Packing.

<input type="checkbox"/>	<input checked="" type="checkbox"/>
Wahr	Falsch

Sei  $G$  ein Graph und  $v$  ein Knoten in  $G$ . Dann ist mindestens entweder  $v$  oder alle Nachbarn von  $v$  in jedem Vertex Cover von  $G$ .

<input checked="" type="checkbox"/>	<input type="checkbox"/>
Wahr	Falsch

Der Algorithmus von De Pina hat eine Gesamtlaufzeit von  $\mathcal{O}(mn \log n)$ .

<input type="checkbox"/>	<input checked="" type="checkbox"/>
Wahr	Falsch

Sei  $G$  ein Graph mit  $n$  Knoten,  $m$  Kanten und  $k$  Zusammenhangskomponenten. Dann hat jede Kreisbasis von  $G$  mindestens  $m - n + k$  Elemente.

<input checked="" type="checkbox"/>	<input type="checkbox"/>
Wahr	Falsch

Sei  $G$  ein Graph und sei  $C$  ein beliebiger Kreis in  $G$ . Dann gilt: Die leichteste Kante von  $C$  ist in jedem minimalen Spannbaum enthalten.

<input type="checkbox"/>	<input checked="" type="checkbox"/>
Wahr	Falsch

Die average-case Laufzeit eines Algorithmus  $\mathcal{A}$  ist eine untere Schranke für die worst-case Laufzeit von  $\mathcal{A}$ .

<input checked="" type="checkbox"/>	<input type="checkbox"/>
Wahr	Falsch

Falls es für jedes Problem in  $\mathcal{FPT}$  einen Lösungsalgorithmus gibt, der polynomiell in der Eingabegröße ist, dann gilt:  $\mathcal{P} = \mathcal{NP}$ .

<input checked="" type="checkbox"/>	<input type="checkbox"/>
Wahr	Falsch

Sei  $P$  ein lineares Programm und seien  $x$  und  $y$  zwei zulässige Lösungen von  $P$ . Dann ist  $\lambda x + (1 - \lambda)y$  für  $\lambda \in [\frac{1}{4}, \frac{3}{4}]$  eine zulässige Lösung von  $P$ .

<input checked="" type="checkbox"/>	<input type="checkbox"/>
Wahr	Falsch

Jeder Präfluss ist ein Fluss, aber nicht jeder Fluss ist auch ein Präfluss.

<input type="checkbox"/>	<input checked="" type="checkbox"/>
Wahr	Falsch