2. Klausur zur Vorlesung
Algorithmentechnik
Wintersemester 2006/2007
12. April 2007

Lösung!

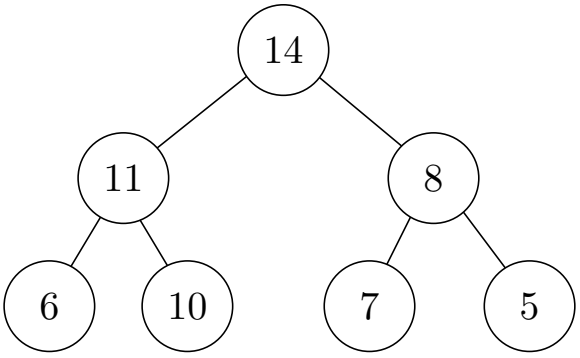
Aufgabe	1	2	3	4	5	6	7	8	9
a	2	2	3	–	3	4	2	–	–
b	2	2	3	–	1	4	1	–	–
c	–	3	3	–	1	–	3	–	–
d	–	–	–	–	2	–	–	–	–
$\sum$	4	7	9	3	7	8	6	6	10
$\sum$	60								
a				–				–	–
b				–				–	–
c	–			–		–		–	–
d	–	–	–	–		–	–	–	–
$\sum$									
$\sum$									

Problem 1: HEAP $2 + 2 = 4$ Punkte

(a) Gegeben sei ein leerer HEAP (genauer: ein MAX-HEAP) H und die folgenden HEAP-Operationen:

1. Insert $(H, 10)$
2. Insert $(H, 11)$
3. Insert $(H, 7)$
4. Insert $(H, 6)$
5. Insert $(H, 14)$
6. Insert $(H, 8)$
7. Insert $(H, 5)$

Geben Sie den Zustand des HEAPS nach Ausführung dieser Operationen als Baum und als Array an.
Baum:



Array:

1	2	3	4	5	6	7
14	11	8	6	10	7	5

(Wie in der Vorlesung beginnen Arrays mit dem Index 1)

(b) Anschließend wird die HEAP-Operation DELETE$(H, 1)$ durchgeführt, indem der untenstehende, aus dem Skript bekannte Algorithmus 1 aufgerufen wird. Geben Sie den Zustand des HEAPS nach dieser Löschoperation als Baum und als Array an.

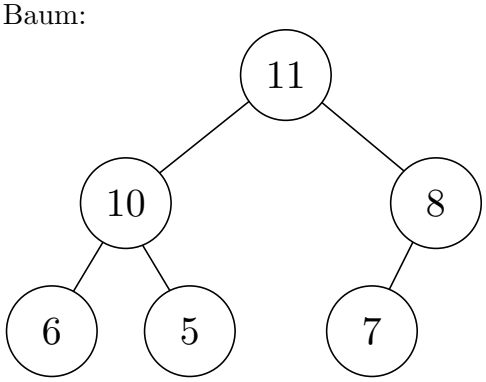
Algorithmus 1 : DELETE (H, i)

Eingabe : HEAP H der Größe n, Index i des zu löschenden Elements
Ausgabe : Elemente des Arrays H ohne $H[i]$, als Heap

```
1 H[i] ← H[n]
2 n ← n − 1
3 Wenn H[i] ≤ H[⌊i/2⌋]
4 │ HEAPIFY (H, i)
5 sonst
6 │ SIFT-UP (H, i)
7 Gib H aus
```

Hinweis: Aus Gründen der Konsistenz gelte in Zeile 3 stets $H[0] = \infty$.

Baum:



Array:

1	2	3	4	5	6
11	10	8	6	5	7

Problem 2: Amortisierte Analyse $2 + 2 + 3 = 7$ Punkte

Eine LISTE sei im Folgenden eine nicht-aufsteigend sortierte, doppelt verkettete Liste, welche als letztes Element eine 0 und ansonsten nur positive Zahlen enthält. Zudem sei folgender Algorithmus X gegeben, der als Eingabe zwei LISTEn jeweils der Länge $n \in \mathbb{N}$ erhält.

Algorithmus 2 : X

Eingabe : Zwei LISTEn der Länge n: LISTE1, LISTE2
Ausgabe : LISTE1 (verändert)

```
1 i ← Erstes Element aus LISTE1
2 j ← Erstes Element aus LISTE2
3 solange j ≠ 0 tue
4 |   solange j < i tue
5 |   |   i ← Nachfolger von i
6 |   Füge j vor i in LISTE1 ein, passe Verkettung an
7 |   j ← Nachfolger von j in LISTE2
```

(a) Gegeben seien die folgenden beiden Listen: LISTE1: $[6, 4, 2, 0]$
LISTE2: $[5, 3, 2, 0]$

Führen Sie Algorithmus X mit den Eingabeparametern LISTE1, LISTE2 durch; brechen Sie jedoch ab, unmittelbar *nachdem* Schritt 7 zum zweiten Mal ausgeführt wurde. Geben Sie LISTE1, i und j zu diesem Zeitpunkt an.

LISTE1: $6, 5, 4, 3, 2, 0$
i: 2
j: 2

(b) Beschreiben Sie in Worten, was Algorithmus 2 macht.

Lösung: Es werden zwei sortierte Listen zu einer sortierten Liste zusammengefügt.

(Der Algorithmus durchläuft die zweite Eingabeliste linear und fügt jedes Element an der entsprechenden Stelle in die erste Eingabeliste ein. Da auch die erste Eingabeliste sortiert vorliegt muss auch diese insgesamt nur ein einziges mal linear durchlaufen werden, um alle Einfügepunkte zu finden). □

(c) Zeigen Sie mit Hilfe einer amortisierten Analyse, dass die asymptotische Worst-Case-Laufzeit von Algorithmus 2 in $O(n)$ ist. Zählen Sie dabei nur die Anzahl der Vergleiche.
Hinweis: Wenn Sie die Potentialmethode nutzen wollen, dann verwenden Sie die Potentialfunktion $\mathbb{C}(D_\ell) = 2 \cdot \ell -$(Position von i in aktueller LISTE1), wobei D_ℓ die Datenstruktur nach dem ℓ-ten Durchlauf der äußeren Schleife ist.

Lösung:

Potentialmethode
Für die Potentialfunktion $\mathbb{C}(D_\ell) = 2 \cdot \ell -$(Position von i in aktueller LISTE1) gilt

$$\begin{aligned} &\mathbb{C}(D_0) = 2 \cdot 0 - 1 = -1 && \text{sowie} \\ &\mathbb{C}(D_n) \geq 2 \cdot n - 2 \cdot n && \text{also gilt somit auch} \\ &\mathbb{C}(D_n) \geq \mathbb{C}(D_0) \ . \end{aligned}$$

Nun müssen noch die amortisierten Kosten pro Operation sinnvoll abgeschätzt werden, dabei gilt. Je mehr Vergleich innerhalb eines Durchlaufs der äußeren Schleife stattfinden, desto weiter wandert der Zeiger auf i nach hinten in LISTE1.

$$\begin{aligned} \widehat{c_\ell} = \quad & c_\ell && + \mathbb{C}(D_\ell) - \mathbb{C}(D_{\ell-1}) \\ = \quad & 2 + \ \# \text{ Durchläufe innere Schleife} && + 2 - 1 - \ \# \text{ Durchläufe innere Schleife} \\ = \quad & 3 \end{aligned}$$

Somit gilt insgesamt $\sum_{\ell=1}^{n} c_\ell \leq \sum_{\ell=1}^{n} \widehat{c_\ell} = 3 \cdot \ell$, also ist die Laufzeit von X in $O(n)$.

Ganzheitsmethode

Der Vergleich in Zeile 3 findet genau n mal statt, da j in jedem Durchlauf der äußeren Schleife um eine Position in LISTE2 weiter wandert, bis n. Der Vergleich in Zeile 4 findet maximal $2n$ mal statt: Die Summe der Positionen von i und j in den ursprünglichen LISTEN steigt nach nach jedem Vergleich in Zeile 4 um mindestens 1 an, auf maximal $2n$. Die Anzahl der Vergleiche ist somit maximal $3n \in O(n)$. □

Problem 3: Minimaler Spannbaum MST $3 + 3 + 3 = 9$ Punkte

Gegeben sei ein einfacher, ungerichteter, zusammenhängender, gewichteter Graph $G = (V, E)$ mit injektiver Kantengewichtsfunktion $\omega : E \to \mathbb{N}$ und ein MST T von G. Nun ändert sich das Gewicht einer einzelnen Kante e. Der MST wird nun mit Algorithmus 3 aktualisiert, welcher nach einer Fallunterscheidung eine von vier weiteren Prozeduren aufruft (siehe Zeilen 2,4,6 und 8), die noch nicht näher bekannt seien.
Hinweis: Die Variablen G, E, V, ω, ω' (die neue Gewichtsfunktion) stehen als globale Variablen zur Verfügung und zählen daher nicht zu den Eingabeparametern.

Algorithmus 3 : UPDATE-MST

Eingabe : Alter MST T, geänderte Kante e
Ausgabe : Neuer MST T' entsprechend neuer Gewichtsfunktion ω'

```
1 Wenn e ist Nichtbaumkante und ω'(e) > ω(e)  //Gewicht von Nichtbaumkante erhöht
2     NichtbaumKantenGewichtHoch(T, e)
3 Wenn e ist Nichtbaumkante und ω'(e) < ω(e)  //Gewicht von Nichtbaumkante gesenkt
4     NichtbaumKantenGewichtRunter(T, e)
5 Wenn e ist Baumkante und ω'(e) > ω(e)  //Gewicht von Baumkante erhöht
6     BaumKantenGewichtHoch(T, e)
7 Wenn e ist Baumkante und ω'(e) < ω(e)  //Gewicht von Baumkante gesenkt
8     BaumKantenGewichtRunter(T, e)
```

Im Folgenden sollen alle vier Unterprozeduren so beschrieben werden, dass Algorithmus UPDATE-MST korrekt funktioniert.

(a) Begründen Sie zunächst, warum es in den Prozeduren NICHTBAUMKANTENGEWICHTHOCH und BAUMKANTENGEWICHTRUNTER (Zeilen 2 und 8) ausreicht, T auszugeben.

Lösung: Betrachte zunächst die ProzedurNICHTBAUMKANTENGEWICHTHOCH im alten Graphen: Wegen der Färbungsinvariante muss es eine Folge von Rot- und Grünfärbungen geben, welche mit e beginnt. Im neuen Graphen steigt das Gewicht von e, und somit ist die bisherige Folge von Färbungen noch immer korrekt, da die initiale Rotfärbung von e korrekt ist, und das veränderte Gewicht von e auf den weiteren Verlauf der Färbungen keinen Einfluss hat.

Analoges gilt für die Prozedur BAUMKANTENGEWICHTRUNTER, wenn man eine Folge von Färbungen betrachtet, welche e zu Beginn grün färbt.

Somit gilt: $T' = T$ □

(b) Schreiben Sie in Pseudocode eine Prozedur NICHTBAUMKANTENGEWICHTRUNTER, welche in asymptotischer Worst-Case-Laufzeit $O(|E|)$ läuft, und welche bei einem Aufruf durch Algorithmus UPDATE-MST die korrekte Ausgabe liefert.
Hinweis: T sei als Kantenmarkierung gegeben.

Algorithmus 4 : NICHTBAUMKANTENGEWICHTRUNTER

Eingabe : Alter MST T, geänderte Kante $e = \{u, v\}$
Ausgabe : Neuer MST T'

```
1 B ← Baum, den eine Breitensuche in T, startend bei v aufbaut
2 C ← (Pfad von u nach v in B) ∪ {e}
3 e_max ← e
4 Für e' ∈ C
5 |   Wenn ω'(e') > ω'(e_max)
6 |   |   e_max ← e'
7 T' ← (T ∪ {e}) \ {e_max}
8 Gib T' aus.
```

(c) Schreiben Sie in Pseudocode eine Prozedur BAUMKANTENGEWICHTHOCH, welche in asymptotischer Worst-Case-Laufzeit $O(|E|)$ läuft, und welche bei einem Aufruf durch Algorithmus UPDATE-MST die korrekte Ausgabe liefert.
Hinweis: T sei als Kantenmarkierung gegeben.

Algorithmus 5 : BAUMKANTENGEWICHTHOCH

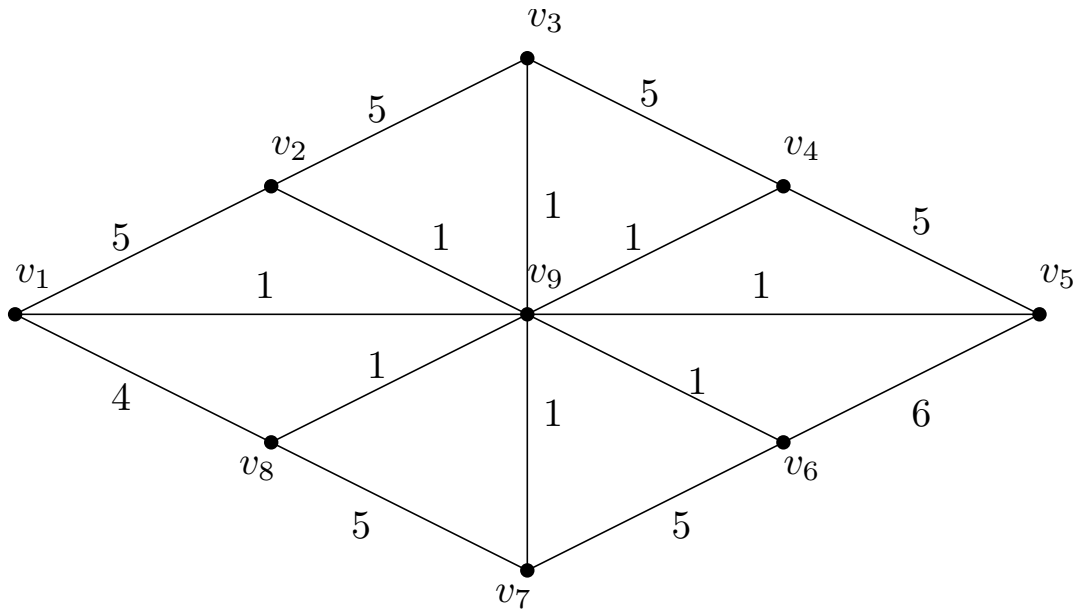
Eingabe : Alter MST T, geänderte Kante $e = \{v, w\}$
Ausgabe : Neuer MST T'

```
1 Führe von v aus Breitensuche in T \ {e} durch und markiere erreichte Knoten
2 e_min ← e
3 Für e' ∈ E
4 |   Wenn genau ein Endknoten von e' markiert
5 |   |   Wenn ω'(e') < ω'(e_min)
6 |   |   |   e_min ← e'
7 T' ← (T ∪ {e'}) \ {e}
8 Gib T' aus.
```

(*Hinweis*: Grobe Ideen müssen in Pseudocode oder zumindest in algorithmische Konzepte wie *Breitensuche* übersetzt werden!)

Problem 4: Minimaler Schnitt 3 Punkte

Im folgenden Bild ist ein gewichteter Graph $G = (V, E)$ gegeben, die Kantengewichte sind an die Kanten geschrieben. Führen Sie die ersten beiden Durchläufe der Prozedur MINSCHNITTPHASE aus dem Algorithmus von Stoer und Wagner auf G durch. Startknoten sei immer Knoten v_1. Geben Sie dabei die unten geforderten Informationen an.

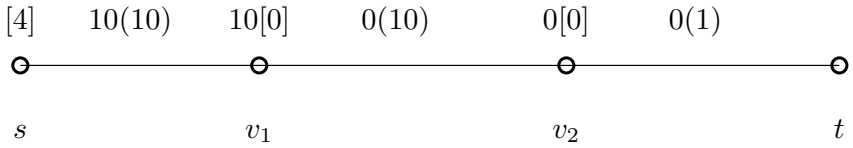


Phase 1
$s = v_7, t = v_8$
SCHNITT-DER-PHASE: $(\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_9\}, \{v_8\})$
Wert des Schnittes aus Phase 1: 10
Knoten, die in Phase 1 verschmolzen werden: v_7, v_8

Phase 2
$s = \{v_7, v_8\}, t = v_9$
SCHNITT-DER-PHASE: $(\{v_1, v_2, v_3, v_4, v_5, v_6, \{v_7, v_8\}\}, \{v_9\})$
Wert des Schnittes aus Phase 2: 8
Knoten, die in Phase 2 verschmolzen werden: $\{v_7, v_8\}, v_9$

Problem 5: Maximaler Fluss $3 + 1 + 1 + 2 = 7$ Punkte

(a) Betrachten Sie folgenden Graphen:



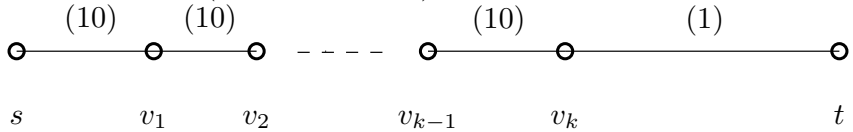
Führen Sie den Algorithmus von Goldberg und Tarjan auf diesem Graphen aus, um einen maximalen s-t-Fluss zu berechnen. Die Initialisierung haben wir schon für Sie durchgeführt. Die Zahlen an den Kanten (ohne Klammern) bezeichnen den Wert des Präflusses, und die Zahlen in Klammern die Kapazität. Die Werte an den Knoten (ohne Klammern) bezeichnen den Flussüberschuss, und die Werte in eckigen Klammern die Distanz. Geben Sie die weiteren Schritte an:

1. RELABEL (v_1), dist $= 1$	,	2. PUSH (v_1, v_2), $\Delta = 10$
3. RELABEL (v_2), dist $= 1$	,	4. PUSH (v_2, t), $\Delta = 1$
5. RELABEL (v_2), dist $= 2$	,	6. PUSH (v_2, v_1), $\Delta = 9$
7. RELABEL (v_1), dist $= 3$	,	8. PUSH (v_1, v_2), $\Delta = 9$
9. RELABEL (v_2), dist $= 4$	,	10. PUSH (v_2, v_1), $\Delta = 9$
11. RELABEL (v_1), dist $= 5$	,	12. PUSH (v_1, s), $\Delta = 9$

Wie hoch ist der Wert des maximalen Flusses?

Wert: 1

(b) Der folgende Graph bestehe aus einer Linie mit k inneren Knoten mit Kantenkapazitäten wie angedeutet (siehe Abbildung).



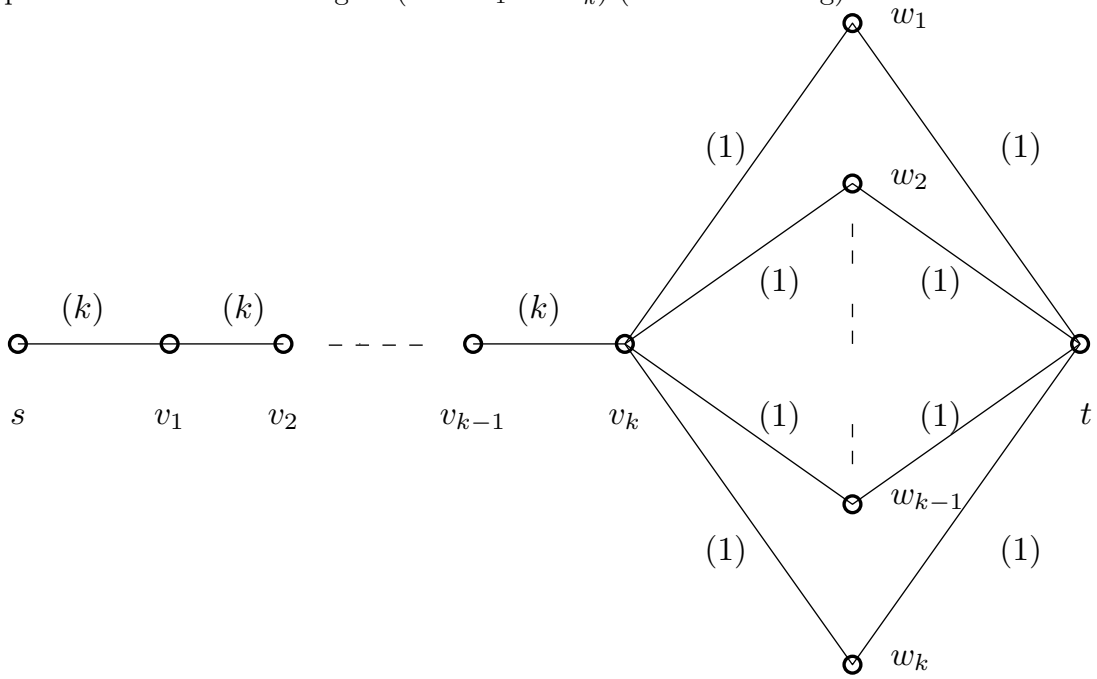
Wie viele PUSH- und wie viele RELABEL-Operationen sind hier insgesamt asymptotisch mindestens erforderlich, in Abhängigkeit von k (möglichst scharf)?

Lösung: $\Omega(k^2)$ □

(c) Angenommen, eine Breitensuche in dem Graphen aus Teil (b) brauche $k + 1$ Schritte. Wie viele Schritte benötigt dann der Algorithmus von Edmonds und Karp asymptotisch mindestens auf diesem Graphen, in Abhängigkeit von k (möglichst scharf)?

Lösung: $\Omega(k)$ □

(d) Der folgende Graph bestehe aus einer Linie mit k inneren Knoten (v_1 bis v_k), gefolgt von k parallelen Pfaden der Länge 2 (über w_1 bis w_k) (siehe Abbildung).



(i) Wie viele PUSHs und RELABELs sind hier insgesamt asymptotisch scharf erforderlich, in Abhängigkeit von k?

Lösung: $\Theta(k)$ □

(ii) Angenommen, eine Breitensuche in diesem Graphen brauche $k + 2$ Schritte. Wie viele Schritte benötigt dann der Algorithmus von Edmonds und Karp asymptotisch mindestens auf diesem Graphen (möglichst scharf)?

Lösung: $\Omega(k^2)$ □

Problem 6: Kreisbasen $4 + 4 = 8$ Punkte

(a) Beweisen Sie, dass jeder Kreis C einer minimalen Kreisbasis $\mathcal{B}$ ein *einfacher* Kreis in dem Sinne ist, dass kein Knoten des zugrundeliegenden Graphen zu mehr als zwei Kanten von C inzident ist.

Lösung: Annahme: Sei $C \in \mathcal{B}$ kein einfacher Kreis. Dann lässt sich C in zwei Kreise $C_1, C_2 \subsetneq C$ zerlegen mit $C = C_1 \oplus C_2$. Da aber $B \setminus \{C\} \cup \{C_1, C_2\}$ den Kreisraum erzeugt, gibt es eine Teilmenge $\mathcal{B}' \subset B \setminus \{C\} \cup \{C_1, C_2\}$, die Basis ist und wegen $w(C_1) + w(C_2) = w(C)$ gilt $w(\mathcal{B}') < w(\mathcal{B})$, ein Widerspruch zur Minimalität von $\mathcal{B}$. □

(b) Im Folgenden ist der aus der Vorlesung bekannte Algorithmus von de Pina (algebraische Form) zur Berechnung einer minimalen Kreisbasis aufgelistet.

Algorithmus 6 : Algorithmus von de Pina algebraisch

Eingabe : Graph $G = (V, E)$
Ausgabe : MCB von G
1 **Für** $i = 1$ *bis* N
2 $S_i \leftarrow \{e_i\}$
3 **Für** $k = 1$ *bis* N
4 Finde einen kürzesten Kreis C_k mit $\langle C_k, S_k \rangle = 1$
5 **Für** $i = k + 1$ *bis* N
6 **Wenn** $\langle C_k, S_i \rangle = 1$
7 $S_i \leftarrow S_i \oplus S_k$
8 Ausgabe ist: $\{C_1, \ldots, C_N\}$

(i) Geben Sie knapp an, welchen Zweck Zeile 7 erfüllt.

Lösung: In diese Zeile werden alle zukünftige *Zeugen*, also die Vektoren S_ℓ, vorbereitet. Es wird sichergestellt, dass folgende Invariante gilt:
$\langle C_i, S_{j+1} \rangle = 0$ für $1 \leq i \leq j \leq N$
(In Worten bedeutet dies, dass stets alle zuknftigen Zeugen zu allen bisherigen Kreisen der MCB orthogonal sind.) □

(ii) Wäre Algorithmus 6 auch dann korrekt, wenn Zeile 7 ersetzt würde durch die Zeile

$$S_i \leftarrow S_i \oplus C_k \qquad ?$$

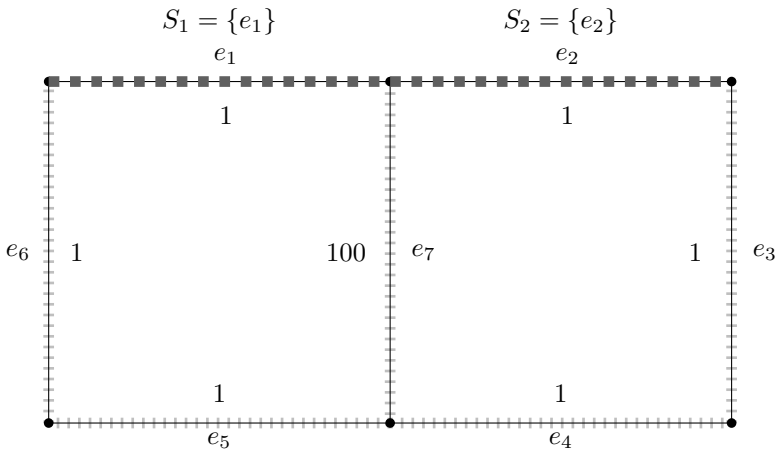
Begründen Sie ihre Antwort.

Lösung: Nein! Intuitiv scheint dies zwar eine Orthogonalisierung zu sein, jedoch funktioniert dies bei Kreisbasen nicht. Liefert Zeile 6 WAHR, so gilt nach Ausführung von Zeile 7 für den geänderten Zeugen $\overline{S_i}$:

$$\langle C_k, \overline{S_i} \rangle = \langle C_k, S_i \rangle + \langle C_k, C_k \rangle = \begin{cases} \langle C_k, S_i \rangle = 1 & \text{, falls } |C_k| \text{ gerade} \\ (\langle C_k, S_i \rangle + 1) \mod 2 = 0 & \text{, falls } |C_k| \text{ ungerade} \end{cases}$$

Die geforderte Invariante aus dem Teil (i) gilt also nicht notwendigerweise, was lineare Abhängigkeit der gefunden Basiskreise bewirken kann.

Gegenbeispiel: Das folgende Beispiel illustriert einen Fall in dem dies zu einem Fehler führt (Gewichte wie angegeben).



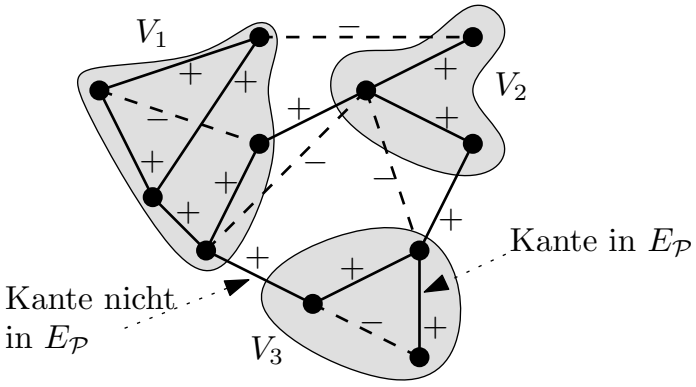
Im geänderten Algorithmus würde zunächst gesetzt $C_1 = \{e_1, \ldots e_6\}$, dann würde S_2 inkorrekt orthogonalisiert zu $S_2 = \{e_1, e_3, \ldots, e_6\}$. Mit Hilfe dieses Zeugen wird dann gesetzt $C_2 = \{e_1, \ldots e_6\} = C_1$, und der Algorithmus terminiert. Offenbar ist $\{C_1, C_2\}$ keine Basis! □

Problem 7: Approximationsalgorithmus für MAXAGREE $2 + 1 + 3 = 6$ Punkte

Gegeben sei ein ungerichteter, einfacher Graph $G = (V, E)$. Jede Kante sei entweder als *Pluskante* oder als *Minuskante* beschriftet, so dass für die Kantenmenge gilt $E = E_+ \uplus E_-$.

Zu einer Partition $\mathcal{P} = \{V_1, \ldots, V_k\}$ der Knotenmenge V bezeichne $E_\mathcal{P}$ die Menge all der Kanten, für die beide Endpunkte im selben V_i liegen. Die Funktion $\text{AGREE}(\mathcal{P}) = |E_+ \cap E_\mathcal{P}| + |E_- \setminus E_\mathcal{P}|$ zählt also die *inneren Pluskanten* und *äußeren Minuskanten*. Das Problem MAXAGREE besteht nun darin, diejenige Partition $\mathcal{P}$ der Knotenmenge zu finden, welche $\text{AGREE}(\mathcal{P})$ maximiert.

Das Bild rechts illustriert eine gegebene Instanz. Die durchgezogenen Kanten bilden E_+ und die gestrichelten E_-. Zur der eingezeichneten Partition $\mathcal{P} = \{V_1, V_2, V_3\}$ der Knotenmenge liefert $\text{AGREE}(\mathcal{P})$ den Wert $9 + 3 = 12$.



Der folgende Algorithmus APPROXMAXAGREE ist ein einfacher Approximationsalgorithmus für MAXAGREE:

Algorithmus 7 : APPROXMAXAGREE

Eingabe : Ungerichteter, einfacher Graph $G = (V, E = E_+ \uplus E_-)$
Ausgabe : Partition $\mathcal{P} = \{V_1, \ldots, V_k\}$ von V

1 **Wenn** $|E_+| \geq |E_-|$
2 ⌊ Gib $\mathcal{P}_{\text{Gesamt}} = \{\{V\}\}$ als Liste von Listen aus
3 **Wenn** $|E_-| > |E_+|$
4 ⌊ Gib $\mathcal{P}_{\text{Einzeln}} = \{\{v_1\}, \{v_2\}, \ldots, \{v_n\}\}$ als Liste von Listen aus

(a) Geben Sie die asymptotische Worst-Case-Laufzeit von Algorithmus APPROXMAXAGREE an und begründen Sie diese.

Lösung: Um die Kardinalitäten von E_- und E_+ zu erhalten muss in $O(|E|)$ über die Kanten E iteriert werden und die mit *Plus* bzw. *Minus* beschrifteten Kanten gezählt werden. Die Ausgaben in Zeilen 2 und 4 iterieren insgesamt über die Knotenmenge und sind somit in $O(|V|)$. Die Gesamtlaufzeit ist also in $\Theta(|E| + |V|)$.

(Graph nicht notw. zusammenhängend, betrachte die Fälle $G = K_n$ und $G = (V, \emptyset)$.) □

(b) Zeigen Sie, dass Folgendes gilt:

$$\max_{\mathcal{P} \text{ Partition von } V}(\textsc{Agree}(\mathcal{P})) \leq |E_+| + |E_-|$$

Lösung: Es gilt stets:

$$\begin{array}{rcccl} \textsc{Agree}(\mathcal{P}) & = & |E_+ \cap E_{\mathcal{P}}| & + & |E_- \setminus E_{\mathcal{P}}| \\ & \leq & |E_+ \cap E| & + & |E_- \setminus \emptyset| \\ & = & |E_+| & + & |E_-| \end{array}$$

Damit gilt diese Ungleichung insbesondere auch für das Maximum über alle Partitionen. □

(c) Beweisen Sie, dass Algorithmus ApproxMaxAgree ein 2-Approximationsalgorithmus für das Problem MaxAgree ist.

Lösung: Aus Teilaufgabe (b) ist bekannt das stets gilt $\textsc{Agree}(\mathcal{P})) \leq |E_+| + |E_-|$. Für die Ausgabe von Algorithmus 7 gilt somit:

$$\textsc{Agree}(\mathcal{P}_{\text{Approx.}}) = \begin{cases} \textsc{Agree}(P_{\text{Gesamt}}) = |E_+| & \text{, falls } |E_+| \geq |E_-| \\ \textsc{Agree}(P_{\text{Einzeln}}) = |E_-| & \text{, falls } |E_-| > |E_+| \end{cases}$$

In beiden Fällen gilt:

$$\frac{\textsc{Opt}_{\mathcal{P} \text{ Partition von } V}(\textsc{Agree}(\mathcal{P}))}{\textsc{Agree}(\mathcal{P}_{\text{Approx.}}} \leq \frac{|E_+| + |E_-|}{\max\{|E_+|, |E_-|\}} \leq 2$$

Somit gilt $\mathcal{R}_{\textsc{ApproxMaxAgree}}(G) \leq 2$ für jeden Eingabegraphen G. □

Problem 8: ILP für MAXAGREE 6 Punkte

Modellieren Sie das Problem MAXAGREE aus Aufgabe 7 als ILP. Erklären Sie Zielfunktion, Variablen und Nebenbedingungen. Nutzen Sie dazu binäre Variablen $x_{i,j}$, die eine Äquivalenzrelation zwischen Knoten beschreiben.
Hinweis: Es ist keine Standardform erforderlich.

Lösung: Definiere n^2 Variablen $x_{i,j} \in \{0,1\}$, je eine pro Paar von Knoten i, j aus V. Es gilt:
$x_{i,j} = 1 \Rightarrow i$ und j sind in der selben Gruppe
$x_{i,j} = 0 \Rightarrow i$ und j sind nicht der selben Gruppe
Da dies eine Äquivalenzrelation ist, werden folgende Nebenbedingungen gefordert:

$$\begin{aligned}
&\text{Reflexivität} && \forall\, i\colon x_{i,i} = 1 \ , \\
&\text{Symmetrie} && \forall\, i,j\colon x_{i,j} = x_{j,i} \ , \text{ und} \\
&\text{Transitivität} && \forall\, i,j,k\colon \begin{cases} x_{i,j} + x_{i,k} - 1 \ \leq \ x_{j,k} \\ x_{i,j} + x_{j,k} - 1 \ \leq \ x_{i,k} \\ x_{i,k} + x_{j,k} - 1 \ \leq \ x_{i,j} \end{cases} .
\end{aligned}$$

(*Hinweis:* Kehrt man die Variablen x_{ij} um so erhält die Dreiecksungleichung.)

Analog zur Definition von AGREE ist die Zielfunktion dann die Summe der Pluskanten, deren Variablen auf 1 gesetzt sind (innerhalb einer Gruppe), plus die Summe der Minuskanten, deren Variablen auf 0 gesetzt sind (zwischen Gruppen):

$$f(\mathcal{P}) = \sum_{\{u,v\} \in E_+} x_{uv} + \sum_{\{u,v\} \in E_-} (1 - x_{uv})$$

□

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

	Wahr	Falsch
Die asymptotische Worst-Case-Laufzeit für *eine* FIND-Operation wird durch die Verwendung von Pfadkompression echt geringer.	☐	☒
Es gilt : $\omega(n \log^2 n) \cap O(n^2 \log n) = \Theta(n \log^2 n)$	☐	☒
Ein Algorithmus mit der rekursiven Laufzeit $T(n) = 4T(\frac{n}{2}) + \Theta(n^3)$ hat eine asymptotische Laufzeit in $o(n^3)$.	☐	☒
Sei Π ein Entscheidungsproblem, LUEGNER ein randomisierter Algorithmus, so dass für alle Instanzen I von Π gilt: $I \in Y_\Pi \longrightarrow \Pr[\text{LUEGNER}(I) \text{ ist „JA“}] < \frac{1}{2}$ $I \notin Y_\Pi \longrightarrow \Pr[\text{LUEGNER}(I) \text{ ist „JA“}] > \frac{1}{2}$ Dann gilt auch $\Pi \in \mathcal{PP}$.	☒	☐
Der Algorithmus von Ford-Fulkerson terminiert auf jedem beliebigen Flussnetzwerk.	☐	☒
Mit $O(n/((\log n)^2))$ Prozessoren kann man auf einer CREW-PRAM das Maximum aus n Werten in einer asymptotischen Worst-Case-Laufzeit von $O(\log n)$ bestimmen.	☐	☒
In einem gewichteten, ungerichteten, zusammenhängenden, einfachen Graphen ist die schwerste zu einem Knoten inzidente Kante in keinem MST enthalten.	☐	☒
Ein Entscheidungsproblem, das sich mit Hilfe eines ILPs polynomieller Größe lösen lässt, welches nur konstant viele $\{0, 1\}$-Variablen und sonst keine ganzzahligen Variablen enthält, ist in $\mathcal{P}$.	☒	☐
Die Laufzeit des Algorithmus von Horton ist in $O(m^3 \cdot n)$.	☒	☐
In einem zusammenhängenden Graphen G gilt: Die Menge aller Kantenmengen, welche Spannbäume in G induzieren, bilden ein Matroid über dem Kantenraum von G.	☐	☒