2. Klausur zur Vorlesung
Algorithmentechnik
Wintersemester 2005/2006
18. April 2006

Lösung!

Aufgabe	1	2	3	4	5	6	7	8	9	10
a	3	4	1	1	5	1	4	2	3	–
b	5	–	1	3	–	3	3	3	–	–
c	–	–	1	2	–	3	–	–	–	–
$\sum$	8	4	3	6	5	7	7	5	3	12
$\sum$	60									
a										–
b		–			–				–	–
c	–	–			–		–	–	–	–
$\sum$										
$\sum$										

Aufgabe 1: $3 + 5 = 8$ Punkte

(a) Für zwei Algorithmen $\mathcal{A}$ und $\mathcal{B}$ gelten die Rekursionsabschätzungen

$$T_{\mathcal{A}}(n) = T_{\mathcal{A}}(n/3) + T_{\mathcal{A}}(n/4) + n^2$$

bzw.

$$T_{\mathcal{B}}(n) = 4 \cdot T_{\mathcal{B}}(n/4) + n^{\alpha} \ .$$

Für welche Werte von α ist $\mathcal{A}$ asymptotisch schneller als $\mathcal{B}$? Begründen Sie Ihre Aussage.

Lösung: Es gilt $T_{\mathcal{A}}(n) \leq 2 \cdot T_{\mathcal{A}}(n/3) + n^2$. Mit dem Mastertheorem folgt $T(A) \in O(n^2)$, also $T(A) \in \Theta(n^2)$. Es gilt auch $T_{\mathcal{A}}(n) \geq 2 \cdot T_{\mathcal{A}}(n/4) + n^2$. Mit dem Mastertheorem folgt $T_{\mathcal{A}}(n) \in \Omega(n^2)$. Für $1 < \alpha$ gilt $T_{\mathcal{B}}(n) = \Theta(n^{\alpha})$. Für $\alpha = 1$ gilt $T_{\mathcal{B}}(n) \in \Theta(n \log n)$. Für $\alpha < 1$ gilt $T_{\mathcal{B}}(n) \in \Theta(n)$. Also ist $\mathcal{A}$ schneller für $\alpha > 2$. □

(b) Eine Warteschlange (FIFO) soll mit Hilfe von zwei STACKs (LIFO) S_{in} und S_{out} implementiert werden. Dabei darf kein zusätzlicher Speicher verwendet werden. Auf den STACKs seien jeweils nur die Operationen S.PUSH(x), $x \leftarrow S$.POP() und S.ISEMPTY () erlaubt. Die POP-Operation soll dabei das oberste Element x vom Stack nehmen und ausgeben.

- Geben Sie eine möglichst effiziente Implementation der Warteschlangenoperationen ENQUEUE (x) (Einstellen des Elements x in die Warteschlange) und DEQUEUE (Ausgabe und Entfernen des Elements, dass am längsten in der Warteschlange ist) an.
- Analysieren Sie die amortisierte Anzahl von PUSH- und POP-Operationen, die im schlechtesten Fall für eine beliebige Folge von n DEQUEUE- oder ENQUEUE-Operationen benötigt werden.

Lösung:

Algorithmus 1 : ENQUEUE (x)

Eingabe : Element x
Datenstrukturen : Stacks S_{in} und S_{out}

1 S_{in}.PUSH(x)

Algorithmus 2 : DEQUEUE

Datenstrukturen : Stacks S_{in} und S_{out}
Ausgabe : Element x, dass am längsten in der Warteschlange war

1 **Wenn** S_{out}.ISEMPTY *()*
2 | **solange** *nicht* S_{in}.ISEMPTY *()* **tue**
3 | | S_{out}.PUSH (S_{in}.POP ())
4 Gib S_{out}.POP () zurück

Jede ENQUEUE-Operation benötigt eine PUSH-Operation, verursacht also Kosten 1. Wir

ordnen dieser Operation jedoch Kosten 3 zu. Jedes Element, dass von DEQUEUE zurückgegeben wird, wird genau einmal vom Vorwärtsstapel zum Rückwärtsstapel transportiert. Dies wurde jedoch bereits von der ENQUEUE-Operation bezahlt. Insgesamt werden durch n Operationen also Kosten von höchstens $4n$ verursacht. □

Aufgabe 2: 4 Punkte

In der Datenstruktur UNION-FIND (mit den aus der Vorlesung bekannten Modifikationen zur Beschleunigung) wird bei einer FIND-Operation eine Pfadkompression durchgeführt.

(a) Schreiben Sie in Pseudocode eine möglichst effiziente alternative Prozedur FIND, deren Pfadkompression folgende **Seiteneffekte** garantiert:

- Der Suchpfad von x nach r wird bei der Operation FIND(x) nur einmal durchlaufen (nicht zweimal, wie in der Vorlesung).
- Für jeden Knoten w auf dem Suchpfad von x zur Wurzel r ist der Abstand von r zu w nach der Operation FIND(x) höchstens 2.
- Es wird nur konstant viel zusätzlicher Speicher verbraucht.

Lösung:

(a) Das Element x wird an die Wurzel gehängt. Alle Elemente auf dem Weg zur Wurzel werden an x gehängt:

Algorithmus 3 : Alternatives FIND(x)

Eingabe : UNION-FIND Datenstruktur, Element x
Ausgabe : Menge, in der x enthalten ist
Seiteneffekte : [siehe oben]

```
1  Wenn VOR[x] < 0
2      Ausgabe: x
3  sonst
4      aktuell ← VOR[x]
5      solange VOR[aktuell] > 0 tue
6          nächster ← VOR[aktuell]
7          VOR[aktuell] ←x
8          aktuell ← nächster
9      VOR[x] ← aktuell
10     Ausgabe: aktuell
```

□

Aufgabe 3: $1 + 1 + 1 = 3$ Punkte

(a) Geben Sie die HEAP-Eigenschaft an:

Lösung: Für alle i gilt $A[\text{Vorgänger}(i)] \geq A[i]$. □

(b) Folgern Sie daraus, dass in jedem HEAP $A[1]$ das maximale Element ist.

Lösung: Für ein beliebiges Element $A[i]$ gilt $A[i] \leq A[\text{Vorgänger}(i)] \leq A[\text{Vorgänger}(\text{Vorgänger}(i))] \leq \cdots \leq A[1]$. Also ist $A[1]$ das größte Element. □

(c) Zeigen oder widerlegen: Die drei größten Elemente eines HEAPs befinden sich in $A[1]$, $A[2]$ und $A[3]$.

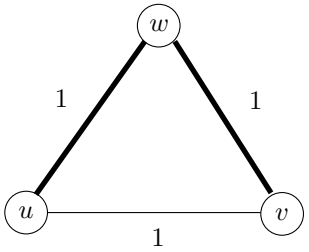
Lösung: [4,3,1,2] erfüllt die HEAP-Eigenschaft und ist somit ein Gegenbeispiel. □

Aufgabe 4: $1 + 3 + 2 = 6$ Punkte

Sei $G = (V, E)$ ein zusammenhängender Graph, c eine positive Kantengewichtsfunktion, T ein aufspannender Baum minimalen Gewichts für G, und u und v zwei beliebige Knoten in G.
Die *Länge* eines Weges $u = v_0, \ldots, v_k = v$ in G zwischen u und v sei $\sum_{i=1}^{k} c(\{v_{i-1}, v_i\})$.
Die *Schrittweite* eines Weges $u = v_0, \ldots, v_k = v$ in G zwischen u und v sei $\max_{i=1\ldots k} c(\{v_{i-1}, v_i\})$.

(a) Widerlegen Sie: Der Weg P von u nach v in T ist ein (in G) kürzester Weg von u nach v (das heißt, seine *Länge* ist minimal unter allen Wegen in G zwischen u und v).

Lösung: In dem Dreiecksgraph ist der kürzeste Weg von u nach v nicht in dem fett eingezeichneten, minimalen aufspannenden Baum enthalten.



□

(b) Zeigen Sie: Der Weg P von u nach v in T ist ein Weg mit der (in G) kürzesten Schrittweite von u nach v (das heißt, seine *Schrittweite* ist minimal unter allen Wegen in G zwischen u und v).

Lösung: Die Aussage ist wahr. Nehmen wir zum Widerspruch an, es gäbe einen u-v-Weg P' in G mit kleinerer Schrittweite. Sei (x, y) die Kante in P mit maximalem Gewicht. Nach der Annahme hat jede Kante von P' geringeres Gewicht. (x, y) induziert einen Schnitt in G, der von P' gekreuzt wird. Sei (s, t) eine solche kreuzende Kante. $T' := T \setminus (x, y) \cup (s, t)$

ist wieder ein aufspannender Baum und es gilt $c(T') = c(T) - c(x, y) + c(s, t) < c(T)$, ein Widerspruch. □

(c) Zeigen Sie: 2^X (die Menge aller Teilmengen von X), ist ein Matroid über X.

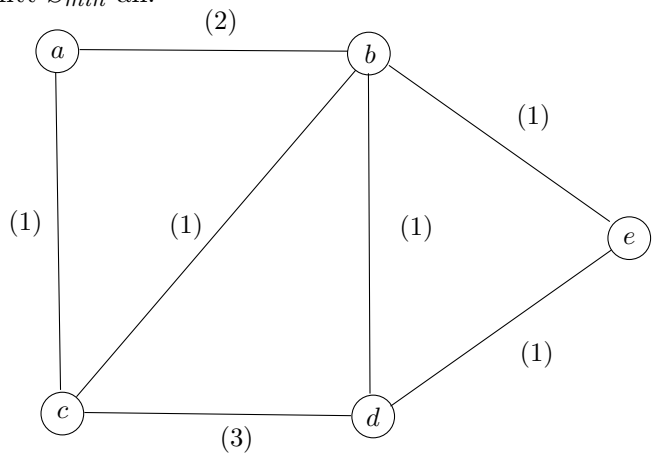
Lösung: $(X, 2^X)$ ist ein Unabhängigkeitssystem, denn es gilt:

- $\emptyset \in 2^X$
- $I_1 \in 2^X$ und $I_2 \subset I_1 \quad \Longrightarrow \quad I_2 \in 2^X$

Auch die Austauscheigenschaft ist erfüllt, denn wenn $I, J \in 2^X$ und $|I| > |J|$, dann gibt es ein $x \in I \setminus J$ und es gilt $J \cup \{x\} \in 2^X$. □

Aufgabe 5: 5 Punkte

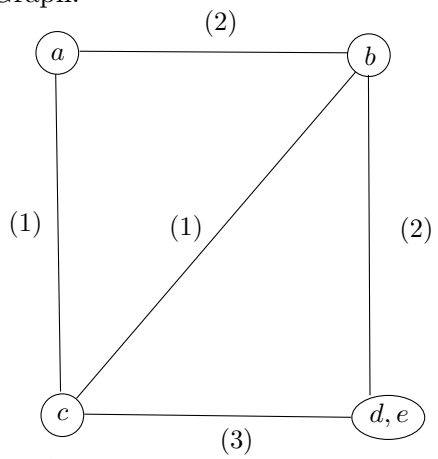
Wenden Sie auf den unten abgebildeten Graphen (Kantengewichte in Klammern) den Algorithmus von Stoer & Wagner an. Geben Sie nach jeder Phase die Knoten s und t, den Schnitt der Phase und dessen Gewicht an und zeichnen Sie den nach dem Verschmelzen resultierenden Graphen (mit Kantengewichten). Der Startknoten sei a. Geben Sie zum Schluss den minimalen Schnitt S_{min} an.



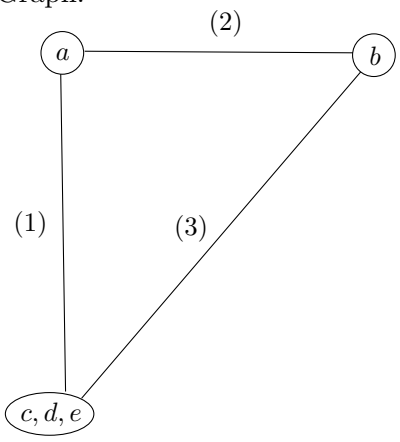
Lösung:

minimaler Schnitt $S_{\min} = S_1$

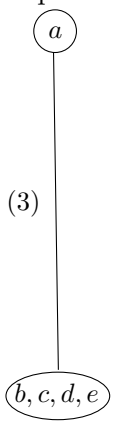
Phase 1
$s = d, t = e$
Schnitt der Phase $S_1 = \{e\}$
$c(S_1, V \setminus S_1) = 2$
Graph:



Phase 2
$s = c, t = \{d, e\}$
Schnitt der Phase $S_2 = \{d, e\}$
$c(S_2, V \setminus S_2) = 5$
Graph:



Phase 3
$s = b, t = \{c, d, e\}$
Schnitt der Phase $S_3 = \{a.b\}$
$c(S_3, V \setminus S_3) = 4$
Graph:



Phase 4
$s = a, t = \{b, c, d, e\}$
Schnitt der Phase $S_4 = \{a\}$
$c(S_4, V \setminus S_4) = 3$

□

Aufgabe 6: $1 + 3 + 3 = 7$ Punkte

Gegeben sei ein Netzwerk D mit ganzzahligen Kantenkapazitäten, auf dem schon ein maximaler, ganzzahliger s-t-Fluss f gegeben ist. Nun wird auf einer Kante e der Kapazitätswert um 1 erhöht.

(a) Zeigen Sie: Der maximale Flusswert erhöht sich dadurch um höchstens 1.

Lösung: Da sich nur eine Kantenkapazität erhöht, kann sich keine Schnittkapazität um mehr als 1 erhöhen, inbesondere auch die Kapazität des minimalen Schnittes. Nach dem Max-Flow-Min-Cut-Theorem erhöht sich auch der maximale Flusswert um höchstens 1. □

(b) Beschreiben Sie die Arbeitsweise eines Algorithmus, der in Linearzeit einen neuen maximalen s-t-Fluss f' berechnet (falls es einen gibt), nachdem auf einer Kante e der Kapazitätswert um 1 erhöht wurde. [Eine exakte Formulierung in Pseudocode ist nicht erforderlich.] Begründen Sie die Korrektheit.

Lösung:

Algorithmus 4 : Fluss-Update

Eingabe : Netzwerk $(D; s; t; c)$, maximaler Fluss f, Kante e mit neuer Kapazität $c'(e)$
Ausgabe : Maximaler Fluss f' im veränderten Netzwerk

```
1 Wenn f(e) = c(e)
2     Führe eine Breitensuche im Residualnetzwerk mit dem veränderten Fluss durch.
      Wenn t erreicht wird
3         Erhöhe den Fluss auf dem Weg von s nach t jeweils um 1.
```

Falls der Fluss erhöht werden kann, muss es einen erhöhenden Weg geben. Da sowohl der (alte) Fluss als auch die Kapazitäten ganzzahlig sind, kann der Fluss entlang dieses Weges um mindestens 1 erhöht werden. Da sich der Fluss nicht um mehr als 1 erhöhen kann, genügt die einmalige Suche nach dem erhöhenden Weg. Die Laufzeit der Breitensuche ist linear. □

(c) Gegeben sei ein gerichteter Graph $D = (V, E)$ ohne Kantengewichte und zwei Knoten $s, t \in V$. Die Kantenzusammenhangszahl $\kappa(s, t)$ von s und t sei die maximale Anzahl kantendisjunkter s-t-Wege in D. (Eine Menge von Wegen ist kantendisjunkt, wenn keine zwei Wege eine Kante gemeinsam haben.)

Beschreiben Sie die Arbeitsweise eines Algorithmus, der mit Hilfe eines maximalen Flusses die maximale Anzahl kantendisjunkter s-t-Wege ($\kappa(s, t)$) berechnet (Sie können annehmen, dass ein Flussalgorithmus zur Verfügung steht).
Begründen Sie die Korrektheit Ihres Ansatzes knapp.

Lösung:

Algorithmus 5 : KANTENDISJUNKTE WEGE

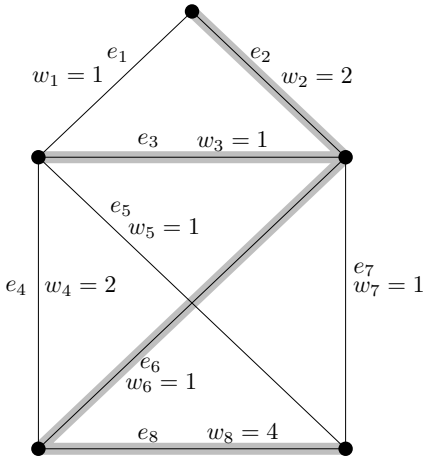
Eingabe : Graph $G = (V, E)$ und zwei Knoten $s, t \in V$
Ausgabe : $\kappa(s, t)$

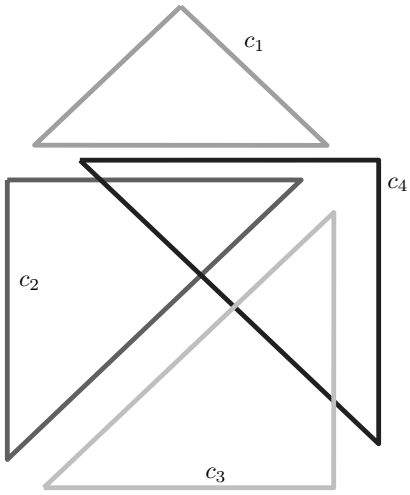
```
1 Definiere Kantenkapazitäten c(e) ≡ 1 für alle Kanten e ∈ E
2 F ← maximaler Fluss auf Netzwerk (D, s, t, c)
3 Ausgabe: Wert von F
```

Wenn es k kantendisjunkte Wege in D gibt, dann ist der Flusswert mindestens k, da auf jedem Weg der Fluss einmal um 1 erhöht werden kann.
Wenn es umgekehrt einen s-t-Fluss mit Wert k in D gibt, gibt es auch einen Fluss mit ganzzahligen Werten. Es gibt dann einen Weg von s nach t auf dessen Kanten der Fluss genau 1 ist. Verringert man auf diesen Kanten den Flusswert um 1, so entsteht ein Fluss mit Wert $k-1$. Durch vollständige Induktion gelangt man so zu k kantendisjunkten Wegen. □

Aufgabe 7: $4 + 3 = 7$ Punkte

(a) Die folgende, linke Abbildung zeigt den Graphen G_{Nikolaus} mit Kantengewichten. Grau unterlegt ist zudem ein aufspannender Baum T (dick, grau) eingezeichnet (die rechte Abbildung illustriert Teilaufgabe (a)(iii)).





(i) Geben Sie die Fundamentalbasis $\mathcal{B} := \{b_1, b_2, b_3, b_4\}$ zu dem Baum T an.
Lösung:

$$\begin{aligned}\mathcal{B} := \{&b_1 = \{e_1, e_2, e_3\},\\ &b_2 = \{e_4, e_3, e_6\},\\ &b_3 = \{e_7, e_6, e_8\},\\ &b_4 = \{e_5, e_3, e_6, e_8\}\}\end{aligned}$$

□

(ii) Geben Sie das Gewicht von $\mathcal{B}$ an:
Lösung: $w(\mathcal{B}) = w(b_1) + w(b_3) + w(b_3) + w(b_4) = 4 + 4 + 6 + 7 = 21$ □

(iii) Betrachten Sie die folgende Kreisbasis $\mathcal{C}$ des Graphen, wie in der rechten, obigen Abbildung illustriert:

$$\begin{aligned}\mathcal{C} := \{&c_1 = \{e_1, e_2, e_3\},\\ &c_2 = \{e_4, e_3, e_6\},\\ &c_3 = \{e_6, e_7, e_8\},\\ &c_4 = \{e_5, e_3, e_7\}\}\end{aligned}$$

Stellen Sie die Elemente ihrer Fundamentalbasis $\mathcal{B}$ als Linearkombinationen von Elementen aus $\mathcal{C}$ dar:
Lösung:

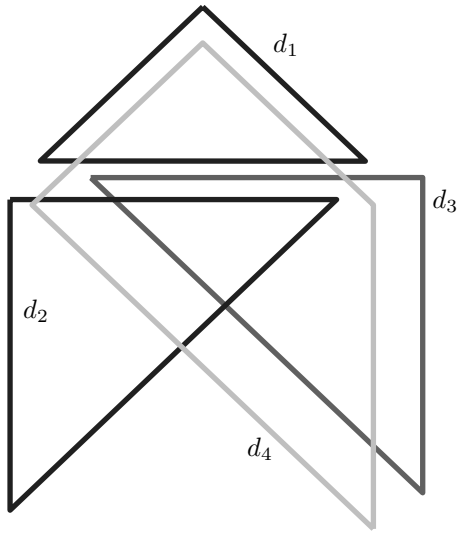
$$\begin{aligned}b_1 &= c_1\\ b_2 &= c_2\\ b_3 &= c_3\\ b_4 &= c_3 \oplus c_4\end{aligned}$$

□

(iv) Sei $\mathcal{D}$ die folgende Menge von Kreisen:

$$\begin{aligned}\mathcal{D} := \{&d_1 = \{e_1, e_2, e_3\},\\ &d_2 = \{e_4, e_3, e_6\},\\ &d_3 = \{e_3, e_5, e_7\},\\ &d_4 = \{e_1, e_2, e_5, e_7\}\}\ .\end{aligned}$$

Es ist $w(\mathcal{D}) = 16$.



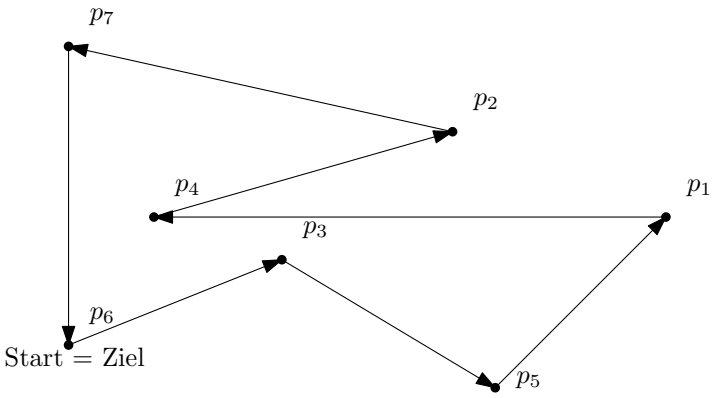
Eine der Mengen $\mathcal{B}$, $\mathcal{C}$ oder $\mathcal{D}$ ist eine minimale Kreisbasis. Welche?
Lösung: $\mathcal{C}$. [$\mathcal{D}$ ist keine Basis. Jede Kante in G_{Nikolaus} ist in einem Kreis enthalten, somit muss jede Kante in mindestens einem Basiskreis enthalten sein. Jede Basis hat genau 4 Kreise und jeder Kreis hat mindestens 3 Kanten. Zu dem Gewicht einer Basis tragen also mindestens 12 Kantengewichte bei, und jedes Kantengewicht aus G_{Nikolaus} mindestens einmal. Das Gewicht jeder Basis ist also mindestens: $9 \cdot 1 + \underbrace{2}_{w_2} + \underbrace{2}_{w_4} + \underbrace{3}_{w_8} = 17$. Wegen $w(C) = 17$ ist $\mathcal{C}$ also minimale Kreisbasis.] □

(b) Sei $\mathcal{M}$ die Vereinigung aller Fundamentalbasen eines gegebenen, zusammenhängenden (ungerichteten) Graphen.
Zeigen oder widerlegen Sie: Für jede minimale Kreisbasis $\mathcal{B}$ gilt $\mathcal{B} \subset \mathcal{M}$.

Lösung: Ja.
Sei $\mathcal{B}$ minimale Kreisbasis von G, und sei $B \in \mathcal{B}$. Da G zusammenhängend ist, existiert mindestens ein aufspannender Baum T_B von G, der alle Kanten von B nutzt, bis auf eine. Somit ist B ein Fundamentalkreis zu T_B und somit $B \in \mathcal{M}$ □

Aufgabe 8: $2 + 3 = 5$ Punkte

Das euklidische *Travelling Salesman Problem* (TSP) lautet wie folgt:
Gegeben: Eine Menge $P = \{p_1, \ldots, p_n\}$ von n verschiedenen Punkten in der Ebene.
Gesucht: Eine möglichst kurze Rundtour durch alle Punkte, also eine Folge $T = p_{i_1}, \ldots, p_{i_{n+1}}$ mit $p_{i_1} = p_{i_{n+1}}$, in der jeder Punkt aus P genau einmal vorkommt (bis auf p_1, der genau zweimal vorkommt) und für die gilt: $\sum_{j=1}^{n} d(p_{i_j}, p_{i_{j+1}})$ = minimal. Dabei ist $d(a, b)$ der euklidische Abstand von a und b in der Ebene.



Ein beispielhafte TSP-Tour $p_6, p_3, p_5, p_1, p_4, p_2, p_7, p_6$ durch alle Punkte.

(a) Zeigen Sie: Ein minimaler Spannbaum (MST) ist kürzer als die kürzeste TSP-Tour.

Lösung: Sei T ein MST und die Tour K kürzer als T. Dann ist der Spannbaum T_0, der durch Wegnahme einer Kante aus K ensteht, echt kürzer als der MST T (wegen $p_i \neq p_j$ hat jede Kante positive Länge). Dies ist ein Widerspruch zur Annahme, dass T ein MST ist. □

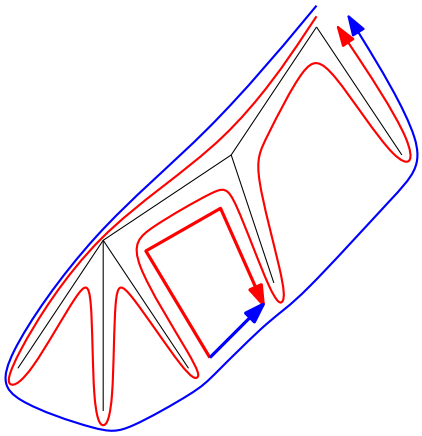
(b) Geben Sie einen Faktor-2-Approximationsalgorithmus für das euklidische TSP an. [Hinweis: Benutzen Sie dazu einen MST (Sie brauchen keinen MST-Algorithmus anzugeben).]

Lösung:

Algorithmus 6 : APPROX-TSP

Eingabe : n verschieden Punkte $P = \{p_1, \ldots, p_n\}$ in der Ebene
Ausgabe : TSP-Tour, die höchstens doppelt so lang ist wie das Optimum

1 Berechne einen MST T von P
2 $K \leftarrow$ Pre-Order-Durchlauf des Baumes T
3 Hänge die Wurzel von T an das Ende von K (Kreis schließen)
4 Ausgabe: K



Läuft man die Knoten des Baumes in (Left-First-)Tiefensuchreihenfolge ab und betrachtet dabei Vorwärts- und Backtrackingkanten, entspricht dies genau zwei mal der Länge des Baumes. Die Pre-Order-Reihenfolge ist (nach der Dreiecksungleichung) höchstens kürzer. Also gilt Länge von $K \leq 2 \cdot c(T) \leq 2 \cdot \text{OPT}$. Folglich ist APPROX-TSP ein Faktor-2-Approximationsalgorithmus.

□

Aufgabe 9: 3 Punkte

Sei Π ein Problem aus $\mathcal{RP}$ mit zugehörigem Algorithmus A. Algorithmus A ist also ein randomisierter einseitiger Monte Carlo Algorithmus. Geben Sie einen randomisierten Algorithmus B an mit polynomialer Laufzeit, für den gilt:

$$\begin{cases} I \in Y_\Pi \longrightarrow \Pr(B(I) \text{ ist „JA“}) \geq \frac{3}{4} \\ I \notin Y_\Pi \longrightarrow \Pr(B(I) \text{ ist „JA“}) = 0 \end{cases}$$

Begründen Sie, dass B die gewünschten Eigenschaften hat.

Lösung:

Algorithmus 7 : Algorithmus B

Eingabe : Probleminstanz I, Algorithmus A
Ausgabe : „JA“ oder „NEIN“

```
1 Führe A zweimal auf I aus.
2 Wenn die Antwort mindestens einmal „JA“ ist
3 |  Antworte „JA“
4 sonst
5 |  Antworte „NEIN“
```

B ist asymptotisch gleich schnell wie A. Es gilt zusätzlich:

$$\begin{cases} I \in Y_\Pi \longrightarrow \begin{cases} \Pr(B(I) \text{ ist „JA“}) = 1 - (\Pr(A(I) \text{ ist zweimal „Nein“})) \\ \qquad\qquad\qquad = 1 - \Pr(A(I) \text{ ist „Nein“})^2 \geq 1 - (1/2)^2 = \dfrac{3}{4} \end{cases} \\ I \notin Y_\Pi \longrightarrow \Pr(B(I) \text{ ist „JA“}) = \Pr(A(I) \text{ ist „JA“}) = 0 \end{cases}$$

□

Aufgabe 10: $12 \times 1 = 12$ 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.

	Wahr	Falsch
$\log_e(n^2) \in \Theta(\log_2 n)$.	X	
Heapsort braucht zum Sortieren einer rückwärts sortierten Folge eine asymptotische Laufzeit in $(\Omega(n \log n) \setminus \Theta(n \log n))$.		X
Die Umwandlung einer n-elementigen Eingabefolge in einen Heap ist asymptotisch in Linearzeit möglich.	X	
Jeder ungerichtete kreisfreie Graph mit n Knoten hat genau $n-1$ Kanten.		X
Zwei MST eines Graphen haben das gleiche minimale, das gleiche durchschnittliche und das gleiche maximale Kantengewicht.	X	
Sei T ein MST von G. Wenn jedes Kantengewicht in G mit einer positiven Zahl $\lambda \in \mathbb{R}$ multipliziert wird, dann bleibt T ein MST.	X	
Sei G ein ungerichteter Graph mit positiven Kantengewichten. Jeder MinCut von G wird von mindestens einer Kante minimalen Gewichts (in G) gekreuzt.		X
Ist der zulässige Bereich eines LP beschränkt, dann hat das LP eine eindeutige Lösung.		X
Die Laufzeit von Goldberg-Tarjan ist $O(\|V\|^2\|E\|)$, wenn eine Relabel-Operation in $O(\|V\|)$ und eine PUSH-Operation in $O(1)$ implementiert werden kann.	X	
Das LP zu einem MaxFlow-Problem (auf einem Graphen mit mindestens einer Kante) hat stets einen beschränkten zulässigen Bereich.	X	
Das MinCut-Problem mit negativen Kantengewichten ist NP-schwer, wenn der Eingabegraph ein Baum ist.		X
Im Algorithmus von de Pina (algebraisch) gilt die Invariante: „Für alle zukünftigen Zeugen $S_{k+1,j}$ (mit $j \geq k+1$) gilt $\langle S_{k+1,j}, C_k \rangle = 0$.“	X	