1. Klausur zur Vorlesung Algorithmentechnik Wintersemester 2005/2006

Hier Aufkleber mit Name und Matrikelnummer anbringen

Vorname: _____

Nachname: _____

Matrikelnummer: _____

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	e	Σ	a	b	c	d	e	Σ
1	2	-	-	-	-	2		-	-	-	-	
2	3	3	2	-	-	8				-	-	
3	4	2	-	-	-	6			-	-	-	
4	3	3	2	-	-	8				-	-	
5	1	2	3	-	-	6				-	-	
6	1	1	1	3	3	9						
7	1	3	-	-	-	4			-	-	-	
8	2	3	-	-	-	5			-	-	-	
9	12x1					12						
Σ						60						

Problem 1: 2 Punkte

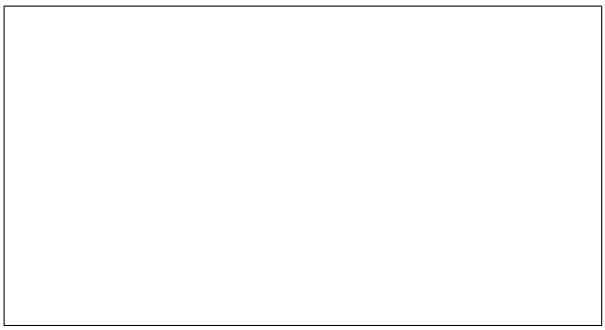
Gegeben seien die folgenden HEAP-Operationen. Geben Sie nach Schritt 5 und nach Schritt 8 jeweils den aktuellen Zustand des HEAPs als Baum und als Array A an.

1. Einfügen von 8
2. Einfügen von 3
3. Einfügen von 10
4. Einfügen von 2
5. Einfügen von 1

Array:

1	2	3	4	5	6	7

Baum:

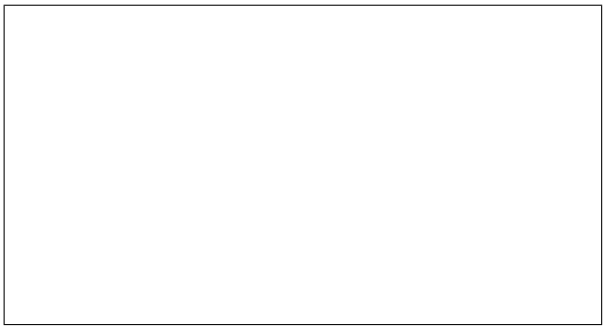


6. Einfügen von 7
7. Löschen von $A[4]$ (mit Wiederherstellung der HEAP-Eigenschaft)
8. Löschen des Maximums (mit Wiederherstellung der HEAP-Eigenschaft)

Array:

1	2	3	4	5	6	7

Baum:



Problem 2: 3+3+2=8 Punkte

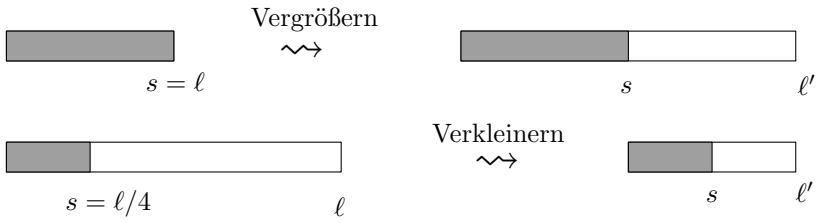
a) Gegeben seien die folgenden Rekursionsformeln. Ordnen Sie diese nach ihrem asymptotischen Verhalten in aufsteigender Reihenfolge (die "schnellste" zuerst) und begründen Sie Ihre Ordnung mit Hilfe geeigneter Schranken. Es gelte jeweils $T_i(1) = c$, für $c > 0$, $i = 1, \ldots, 3$.

(i) $T_1(n) = 4T_1(n/4) + n$

(ii) $T_2(n) = T_2(n-1) + 2n$

(iii) $T_3(n) = 2T_3(n-1) + n$

b) Ein STACK soll mit Hilfe eines Arrays implementiert werden. Er soll theoretisch beliebig groß werden können und trotzdem nicht zu viel Speicher verschwenden. s bezeichne die Anzahl der momentan im STACK enthaltenen Elemente, ℓ die Länge des Arrays. Zu Beginn sind $s = 0$ und $\ell = 1$. (Das als letztes abgelegte Element steht an Stelle s im Array.)



Falls nach der Ausführung von PUSH kein Platz mehr im Array frei ist ($s = \ell$), so wird ein neues Array der Länge $\ell' = 2\ell$ angelegt, die Elemente des alten Arrays dorthin kopiert und der Platz für das alte Array freigegeben. Falls $s < \ell/4$ nach Ausführen von POP gilt, wird ein neues Array der Größe $\ell' = \ell/2$ reserviert, die Elemente des alten Arrays dorthin kopiert und das alte Array freigegeben.

Analysieren Sie die amortisierte Worst-case-Laufzeit einer Folge von n PUSH- und POP-Operationen. [*Hinweis*: Sie können annehmen, dass das Anlegen und Freigeben des Arrays keine Kosten hat, aber das Kopieren eines Elements Kosten 1 verursacht.]

[bitte wenden]

c) Begründen Sie kurz, warum die Laufzeit in $\Omega(n^2)$ ist, wenn das Array in der vorherigen Teilaufgabe schon bei $s < \ell/2 - 10$ statt bei $s < \ell/4$ auf die Größe $\ell/2$ verkleinert wird.

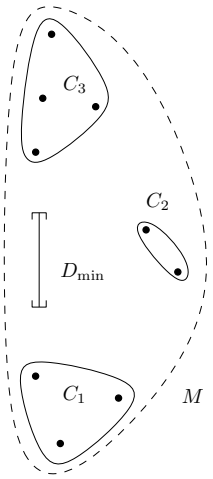
Problem 3: Bottom-Up Clustern von Dokumenten

4+2=6 Punkte

Gegeben sei eine Menge M von n verschiedenen Dokumenten und dazu eine symmetrische *Ähnlichkeitsfunktion* $d : M \times M \to \mathbb{R}$. Die Menge M soll nun wie folgt in "ähnliche" Teilmengen $C_1, \ldots, C_k \subset M$ partitioniert werden.

Zu Beginn bildet jedes Element von M eine eigene, einelementige Teilmenge $C_i, i = 1, \ldots, n$. Dann werden iterativ immer die beiden ähnlichsten Teilmengen verschmolzen. Die Ähnlichkeit zweier Teilmengen C_p, C_q ist $\max_{v \in C_p, w \in C_q} d(v, w)$. Der Algorithmus soll abbrechen, falls nur noch eine Teilmenge vorhanden ist oder kein Paar von Teilmengen mehr mindestens Ähnlichkeit $D_{\min}$ hat.

Die Ähnlichkeiten aller Dokumentenpaare seien vorberechnet und in einem Array $D = D[1], \ldots, D[n^2]$ in nicht-aufsteigender Reihenfolge vorsortiert. Die beiden zu $D[\ell]$ korrespondierenden Dokumente seien in $V[\ell]$ und $W[\ell]$ gespeichert, es gilt also $D[\ell] = d(V[\ell], W[\ell])$ für alle $\ell \in \{1, \ldots, n^2\}$.



(a) Schreiben Sie unter Verwendung einer geeigneten Datenstruktur aus der Vorlesung einen Algorithmus für das beschriebene Verfahren in Pseudocode. Nehmen Sie dazu an, die zur Datenstruktur gehörenden Operationen stünden zur Verfügung.

Algorithmus 1 : BOTTOM-UP-CLUSTERN

Eingabe : Menge M, Array D, Wert $D_{\min}$, Array V, Array W
Ausgabe : Keine
Seiteneffekte : Datenstruktur, welche die Partition verwaltet

[bitte wenden]

(b) Analysieren Sie die asymptotische Laufzeit Ihres Algorithmus und begründen Sie Ihre Antwort. (Sie brauchen nicht die Laufzeiten der Operationen der verwendeten Datenstruktur zu begründen.)

Problem 4: 3+3+2=8 Punkte

Es sei folgender Algorithmus gegeben, von dem nicht bekannt ist, was er berechnet:

Algorithmus 2 :

Eingabe : Ungerichteter, zusammenhängender Graph $G = (V, E)$, Kantengewichte $c\colon E \to \mathbb{R}^+$
Ausgabe : ???

1 Berechne einen beliebigen aufspannenden Baum T von G
2 $S \leftarrow E \setminus T$ (Menge der Nichtbaumkanten)
3 $R \leftarrow \emptyset$
4 **Solange** $S \neq \emptyset$ **wiederhole**
5 Wähle $e \in S$ und entferne e aus S
6 $e' \leftarrow$ Kante mit maximalem Gewicht auf C_e
7 $R \leftarrow R \cup e'$
8 **Wenn** $e \neq e'$
9 $T \leftarrow (T \setminus e') \cup e$
10 Gib $E \setminus R$ zurück

C_e sei dabei der durch T und e definierte Fundamentalkreis in G.

a) Zeigen Sie, dass die Laufzeit von Algorithmus 2 in $O(nm)$ ist. Geben Sie dazu für jeden Schritt eine Schranke für die Laufzeit an. Geben Sie außerdem kurz an, wie die Schritte 1 und 6 implementiert werden können.

Schritt 1:	Schritt 5:
Schritt 2:	Schritt 6:
Schritt 3:	Schritt 7:
Schritt 4:	Schritt 8:

Schritt 1:

Schritt 6:

b) Was berechnet Algorithmus 2? Begründen Sie.

c) Gegeben sei nun ein Graph $G = (V, E)$ mit Kantengewichten $c\colon E \to \mathbb{R}^+$ und MST T.
Geben Sie einen Linearzeit-Algorithmus an, der zu einer nicht in G enthaltenen Kante e mit Kantengewicht $c(e) \in \mathbb{R}^+$ einen MST in $G' := (V, E')$ mit $E' = E \cup \{e\}$ zurückgibt und begründen sie dessen Korrektheit.

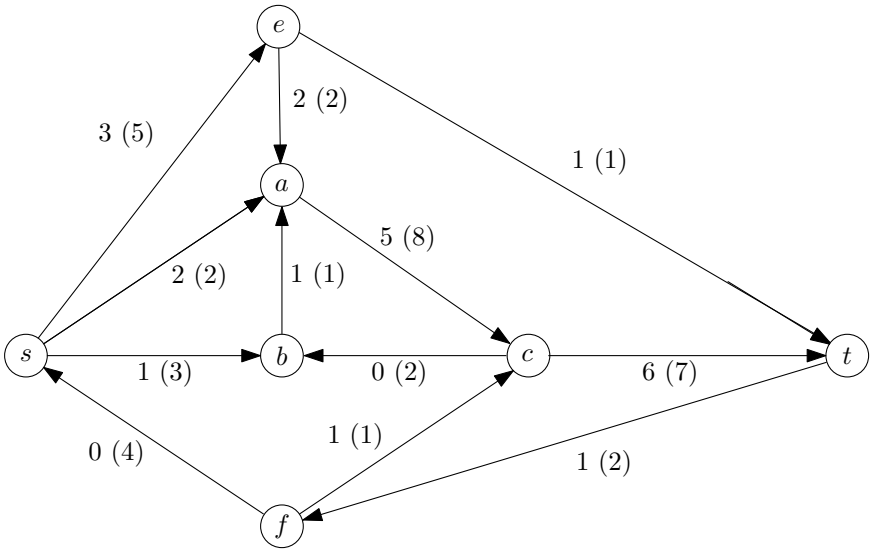
Algorithmus 3 : Algorithmus $\mathcal{B}$

Eingabe : Ungerichteter Graph $G = (V, E)$, Kantengewichte $c\colon E' \to \mathbb{R}^+$, minimaler aufspannender Baum T von G, Kante $e \notin E$
Ausgabe : Minimaler aufspannender Baum T' in G'

Gib T' zurück

Problem 5: 1+2+3=6 Punkte

(a) Zeichnen Sie in den untenstehenden Graphen einen minimalen s-t-Schnitt ein (die Zahlen in Klammern geben die Kantenkapazitäten an, die Zahlen vor den Klammern einen s-t-Fluss). Argumentieren Sie, warum Ihr Schnitt minimal ist.



(b) Beschreiben Sie die Arbeitsweise eines effizienten Algorithmus, der aus einem gegebenen Netzwerk $(D; s; t; c)$ mit maximalem s-t-Fluss f einen minimalen s-t-Schnitt $(S, V \setminus S)$ berechnet. (Eine exakte Formulierung in Pseudocode ist nicht erforderlich.)

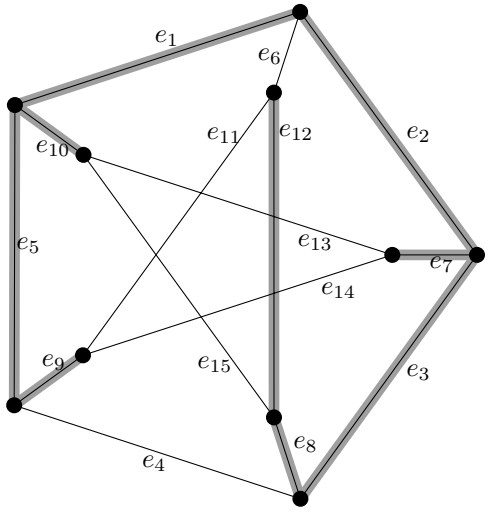
(c) Gegeben sei folgende Erweiterung des Max-Flow-Problems: In dem Netzwerk D gebe es nun k Quellen $s_1, \ldots, s_k$ und ℓ Senken $t_1, \ldots, t_\ell$. Die Flusserhaltungsbedingungen müssen für alle Knoten außer $s_1, \ldots, s_k, t_1, \ldots, t_\ell$ erfüllt sein, ansonsten ist alles wie im gewöhnlichen Flussproblem. Der Wert des Flusses sei

$$w(f) := \sum_{i=1}^{k} \left(\sum_{(s_i,v)\in E} f(s_i, v) - \sum_{(v,s_i)\in E} f(v, s_i) \right) .$$

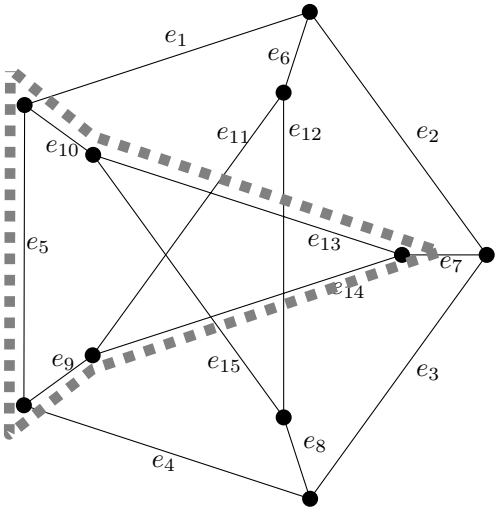
Wie kann dieses Flussproblem mit Hilfe eines gewöhnlichen Flussproblems gelöst werden? Zeigen Sei die Korrektheit Ihres Ansatzes.

Problem 6: Kreisbasen von PETE 1+1+1+3+3=9 Punkte

Betrachten Sie den Petersen-Graph PETE, der im Folgenden zweimal abgebildet ist. Das Gewicht jeder Kante sei 1. (Benutzen Sie zum Benennen von Kanten nur den Index i nicht e_i. Das spart Zeit und ist besser lesbar.)



PETE mit Baum T.



PETE mit Kreis C.

(a) Welche Dimension hat der Kreisraum dieses Graphen?

(b) Erstellen Sie eine Fundamentalbasis $\mathcal{F}$ von PETE zu dem dick grau eingezeichneten aufspannenden Baum T in der linken Abbildung.

(c) Stellen Sie den dick gestrichelt eingezeichneten Kreis C in der rechten Abbildung mit Hilfe Ihrer Basis $\mathcal{F}$ aus Teilaufgabe (b) dar.

[bitte wenden]

(d) Ist Ihre Basis $\mathcal{F}$ eine minimale Kreisbasis? Wenn ja, begründen Sie diese Behauptung. Wenn nicht, geben Sie eine minimale Kreisbasis an und begründen Sie, warum diese eine minimale Kreisbasis ist.

(e) Sei $\mathcal{A}$ ein Algorithmus, der eine minimale Kreisbasis approximiert, indem er die Fundamentalbasis zu einem minimalen aufspannenden Baum berechnet. Zeigen Sie, dass man im Allgemeinen keine relative Gütegarantie angeben kann für das Gewicht einer Kreisbasis, die von Algorithmus $\mathcal{A}$ berechnet wird.

Problem 7: LP 1+3=4 Punkte

Sei ein lineares Programm P gegeben, für das eine *ganzzahlige* Lösung gefordert wird und dessen zulässiger Bereich ein Polytop ist.

(a) Angenommen es sei bekannt, dass alle Ecken des zulässigen Bereichs ganzzahlige zulässige Lösungen von P sind. Begründen Sie, warum man P mit dem Simplexverfahren lösen kann.

(b) Angenommen es sei lediglich bekannt, dass es eine ganzzahlige Optimallösung von P gibt. Warum kann man P dann im Allgemeinen nicht mehr mit dem Simplexverfahren lösen? Zeigen Sie dies zusätzlich anhand eines konkreten Beispiels (keine Standardform erforderlich).

Problem 8: 2+3 = 5 Punkte

Der Approximationsalgorithmus FIRST FIT DECREASING (FFD) für das Problem BIN PACKING sortiert zuerst die Elemente einer Instanz nach nicht-aufsteigenden Gewichten und führt dann FIRST FIT aus.

(a) Geben Sie eine Instanz I zu BIN PACKING an, für die gilt

$$\mathrm{FFD}(I) > \mathrm{OPT}(I) \ .$$

(b) Zeigen Sie, dass es zu jedem gegebenen $k \in \mathbb{N}$ eine Instanz I_k gibt, für die $\mathrm{OPT}(I_k) \geq k$ und

$$\mathrm{FFD}(I_k) \geq \frac{7}{6}\mathrm{OPT}(I_k)$$

gilt.

Hinweis: Man kommt mit zwei verschiedenen Gewichtswerten aus.

Problem 9: $12 \times 1 = 12$ 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.

$O(n \cdot (\log\log n) \cdot \sqrt{n}) \cap \Theta(n^{7/4}) = \emptyset$.	☐ Wahr	☐ Falsch
Ein absteigend sortiertes Array repräsentiert einen Heap.	☐ Wahr	☐ Falsch
Die Zahlenfolge $(12, 8, 2, 6, 4, \pi, 0, 3)$ ist ein 3-HEAP.	☐ Wahr	☐ Falsch
Der Algorithmus von Kruskal ist für dichte Graphen besser geeignet als der Algorithmus von Prim.	☐ Wahr	☐ Falsch
Wenn zu jedem Kantengewicht in G die gleiche positive Zahl $\lambda \in \mathbb{R}$ addiert wird, dann bleibt der minimale Schnitt gleich.	☐ Wahr	☐ Falsch
Wenn zu jedem Kantengewicht in G die gleiche positive Zahl $\lambda \in \mathbb{R}$ addiert wird, dann bleibt der minimale Spannbaum gleich.	☐ Wahr	☐ Falsch
Gegeben ist ein Flussnetzwerk mit ganzzahligen Kapazitäten. Dann ist jeder maximale Fluss auf jeder Kante ganzzahlig.	☐ Wahr	☐ Falsch
Für einen Graphen $G = (V, E)$ und eine Kreisbasis $\mathcal{B}$ von G gilt stets: $\forall e \in E : \exists B \in \mathcal{B} : e \in B$.	☐ Wahr	☐ Falsch
Falls $\mathcal{P} \neq \mathcal{NP}$, dann gibt es einen polynomialen Algorithmus zur Lösung eines beliebigen LP.	☐ Wahr	☐ Falsch
Falls $\mathcal{P} \neq \mathcal{NP}$, dann gibt es keinen polynomialen Algorithmus zur Lösung eines beliebigen ganzzahligen LP.	☐ Wahr	☐ Falsch
QUICKSORT mit zufälliger Pivotelementwahl ist ein randomisierter Las-Vegas-Algorithmus.	☐ Wahr	☐ Falsch
Sei Π ein Minimierungsproblem und $\mathcal{A}_\epsilon$ eine Familie von Approximationsalgorithmen für $\epsilon > 0$, die polynomial in $\lvert I\rvert$ und $1/\epsilon$ sind und für die gilt $\exists k > 1 : \forall I : \mathcal{A}_\epsilon(I) \leq k \cdot OPT(I)$. Dann ist $\mathcal{A}_\epsilon$ ein FPAS für Π.	☐ Wahr	☐ Falsch