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

Hier Aufkleber mit Name und Matrikelnr. 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	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

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

[bitte wenden]

(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

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:
i:
j:

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

[bitte wenden]

(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.

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.

(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
Ausgabe : Neuer MST T'

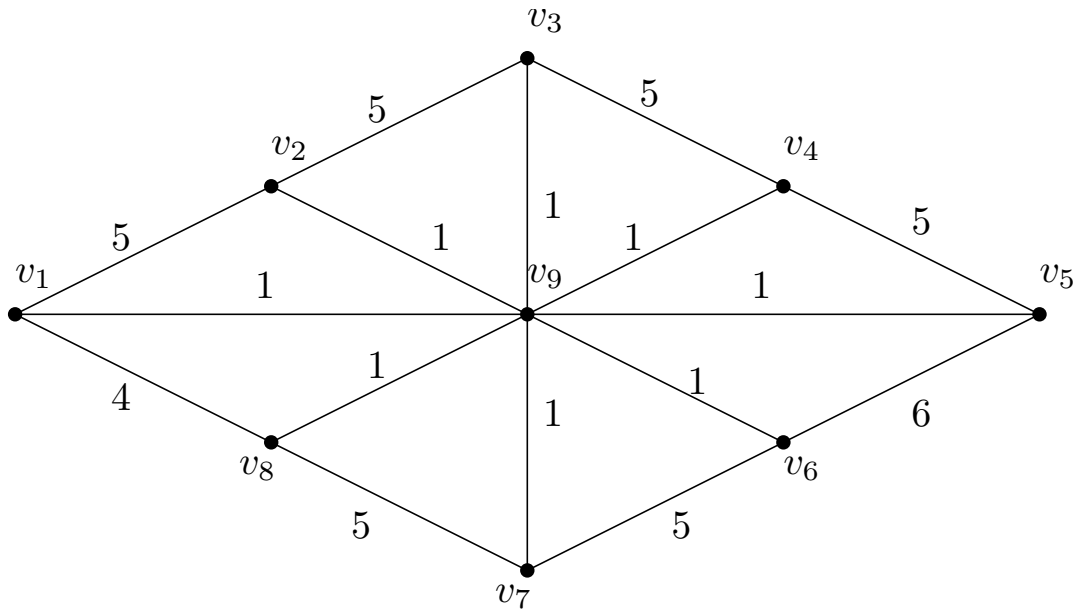
(c) Schreiben Sie in Pseudocode eine Prozedur BAUMKANTENGEWICHTHOCH, welche in asymptotischer Worst-Case-Laufzeit $O(|E|)$ läuft, und welche bei einem Aufruf durch Algorithmus UPDATE-MST die korrekte Ausgabe liefert.
Hinweis: T sei als Kantenmarkierung gegeben.

Algorithmus 5 : BAUMKANTENGEWICHTHOCH

Eingabe : Alter MST T, geänderte Kante e
Ausgabe : Neuer MST T'

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

$s =$ $, t =$

SCHNITT-DER-PHASE: ({ }, { })

Wert des Schnittes aus Phase 1:

Knoten, die in Phase 1 verschmolzen werden:

Phase 2

$s =$ $, t =$

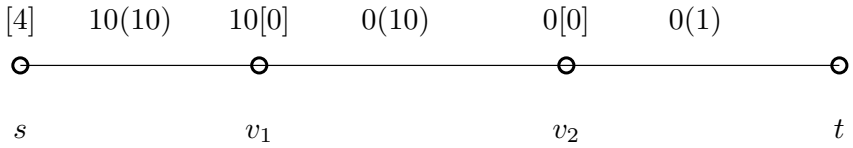
SCHNITT-DER-PHASE: ({ }, { })

Wert des Schnittes aus Phase 2:

Knoten, die in Phase 2 verschmolzen werden:

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

(a) Betrachten Sie folgenden Graphen:



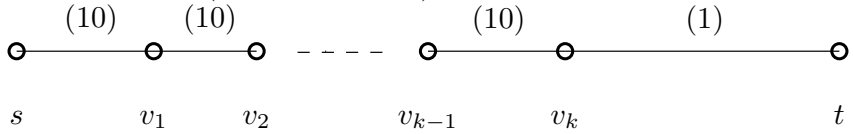
Führen Sie den Algorithmus von Goldberg und Tarjan auf diesem Graphen aus, um einen maximalen s-t-Fluss zu berechnen. Die Initialisierung haben wir schon für Sie durchgeführt. Die Zahlen an den Kanten (ohne Klammern) bezeichnen den Wert des Präflusses, und die Zahlen in Klammern die Kapazität. Die Werte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

1. RELABEL ( ), dist = , 2. PUSH (,), Δ =
3. RELABEL ( ), dist = , 4. PUSH (,), Δ =
5. RELABEL ( ), dist = , 6. PUSH (,), Δ =
7. RELABEL ( ), dist = , 8. PUSH (,), Δ =
9. RELABEL ( ), dist = , 10. PUSH (,), Δ =
11. RELABEL ( ), dist = , 12. PUSH (,), Δ =

Wie hoch ist der Wert des maximalen Flusses?

Wert:

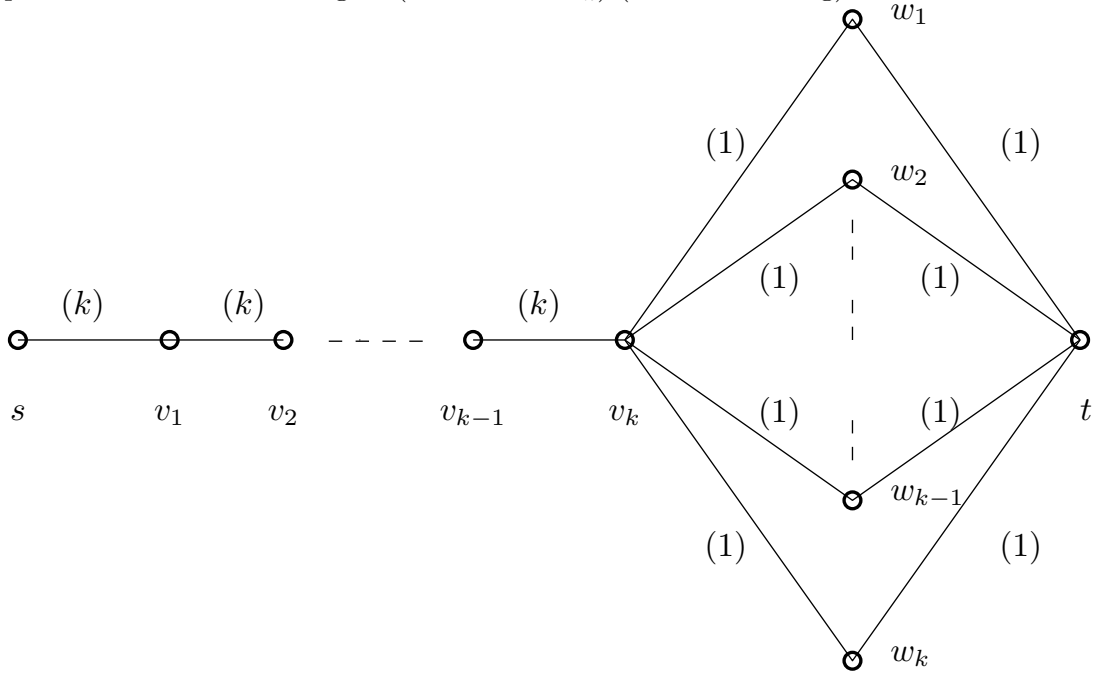
(b) Der folgende Graph bestehe aus einer Linie mit k inneren Knoten mit Kantenkapazitäten wie angedeutet (siehe Abbildung).



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

(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)?

(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?

(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)?

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.

(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 ← {e_i}
3 Für k = 1 bis N
4     Finde einen kürzesten Kreis C_k mit ⟨C_k, S_k⟩ = 1
5     Für i = k + 1 bis N
6         Wenn ⟨C_k, S_i⟩ = 1
7             S_i ← S_i ⊕ S_k
8 Ausgabe ist: {C_1, ..., C_N}
```

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

[bitte wenden]

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

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

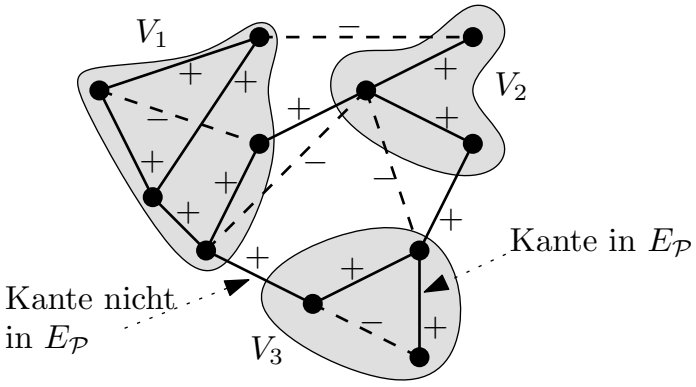
Begründen Sie ihre Antwort.

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.

(b) Zeigen Sie, dass Folgendes gilt:

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

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

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.

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.

Die asymptotische Worst-Case-Laufzeit für *eine* FIND-Operation wird durch die Verwendung von Pfadkompression echt geringer. ☐ Wahr ☐ Falsch

Es gilt : $\omega(n \log^2 n) \cap O(n^2 \log n) = \Theta(n \log^2 n)$ ☐ Wahr ☐ Falsch

Ein Algorithmus mit der rekursiven Laufzeit $T(n) = 4T(\frac{n}{2}) + \Theta(n^3)$ hat eine asymptotische Laufzeit in $o(n^3)$. ☐ Wahr ☐ Falsch

Sei Π ein Entscheidungsproblem, LUEGNER ein randomisierter Algorithmus, so dass für alle Instanzen I von Π gilt:

$$I \in Y_\Pi \longrightarrow \Pr[\text{LUEGNER}(I) \text{ ist „JA“}] < \frac{1}{2}$$
$$I \notin Y_\Pi \longrightarrow \Pr[\text{LUEGNER}(I) \text{ ist „JA“}] > \frac{1}{2}$$

☐ Wahr ☐ Falsch

Dann gilt auch $\Pi \in \mathcal{PP}$.

Der Algorithmus von Ford-Fulkerson terminiert auf jedem beliebigen Flussnetzwerk. ☐ Wahr ☐ Falsch

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. ☐ Wahr ☐ Falsch

In einem gewichteten, ungerichteten, zusammenhängenden, einfachen Graphen ist die schwerste zu einem Knoten inzidente Kante in keinem MST enthalten. ☐ Wahr ☐ Falsch

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 $\mathcal{P}$. ☐ Wahr ☐ Falsch

Die Laufzeit des Algorithmus von Horton ist in $O(m^3 \cdot n)$. ☐ Wahr ☐ Falsch

In einem zusammenhängenden Graphen G gilt: Die Menge aller Kantenmengen, welche Spannbäume in G induzieren, bilden ein Matroid über dem Kantenraum von G. ☐ Wahr ☐ Falsch