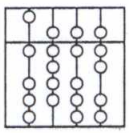
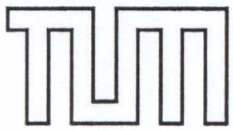


TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK



Lehrstuhl für Effiziente Algorithmen
Grundlagen: Algorithmen und Datenstrukturen
Prof. Dr. Christian Scheideler, Dr. Stefan Schmid

SS 2008
Nachhol-Klausur
9.10.08

Nachhol-Klausur zu Grundlagen: Algorithmen und Datenstrukturen

Name	Vorname	Studiengang	Matrikelnummer
Schmid	Stefan		
Hörsaal	Reihe	Sitzplatz	Unterschrift

Allgemeine Hinweise:

- Bitte füllen Sie die oben angegeben Felder vollständig aus und unterschreiben Sie!
- Schreiben Sie nicht mit Bleistift oder in roter/grüner Farbe!
- Lösen Sie die Aufgaben auf dem freien Platz unter der Angabe (bzw. auf der Rückseite). Fordern Sie weiteres Papier von der Klausuraufsicht, falls Ihnen der Platz nicht ausreicht.
- Als Hilfsmittel ist ausschließlich ein handbeschriebenes DinA4-Blatt zugelassen. Bei Missachtung wird die gesamte Klausur mit null Punkten bewertet.
- Die Arbeitszeit beträgt 120 Minuten.
- Prüfen Sie, ob Sie alle 10 Seiten erhalten haben.

Aufgabe 1 [5 Punkte] Multiple Choice Aufgabe

Kreuzen Sie pro Teilaufgabe höchstens ein Kästchen an. Für ein falsches Kreuz gibt es einen halben Minuspunkt, für ein richtiges einen halben Pluspunkt. Wenn Sie kein Kreuz setzen, bekommen Sie auch keine Punkte. Eine negative Gesamtpunktzahl dieser Aufgabe wird zu 0 aufgerundet. Maximieren Sie Ihre Punktzahl!

a) $o(n \log n) \subset O(n)$. ☐ Richtig ☒ Falsch

b) $f(n) \in O(g(n)) \Leftrightarrow g(n) \in \Omega(f(n))$. ☒ Richtig ☐ Falsch

c) $f'(n) \in O(g'(n)) \Leftrightarrow f(n) \in O(g(n))$ ☐ Richtig ☒ Falsch

d) Falls $T(1) = 1$ und $T(n) = 4 \cdot T(n/2) + 2n$ für alle $n > 1$, dann ist $T(n) \in O(n^2)$.
☒ Richtig ☐ Falsch

e) Jeder zusammenhängende ungerichtete Graph kann durch eine Tiefensuche vollständig exploriert werden.
☒ Richtig ☐ Falsch

f) Bei der amortisierten Analyse wird die durchschnittliche Laufzeit einer Operation im Worst-Case berechnet.
☒ Richtig ☐ Falsch

g) Insertion Sort hat eine Laufzeit von $O(n \log n)$.
☐ Richtig ☒ Falsch

h) Bei dem Feld $[1, 5, 3, 4, 7, 6, 9]$ handelt es sich um einen binären Min-Heap.
☐ Richtig ☒ Falsch

i) Radixsort auf einer Folge von n vielen Zahlen aus der Menge $\{i : i \leq n^c\}$, für eine Kontante c, hat eine lineare Laufzeit in n. ☒ Richtig ☐ Falsch

j) Ein Radixheap kann für beliebige Eingabesequenzen eine Worst Case Laufzeit der Insert-Operation von $O(1)$ sicherstellen.
☐ Richtig ☒ Falsch

Aufgabe 2 [8 Punkte] Rekursion

In dieser Aufgabe wollen wir die Laufzeit zweier rekursiver Algorithmen bestimmen.

a) [3 Punkte] Betrachten Sie ein sortiertes Feld $F =< e_1, ..., e_n >$ mit n unterschiedlichen Elementen e_i, es gilt also $e_i < e_{i+1}$. Gegeben sei der folgende Algorithmus $find(i, j, x)$ der das Element x im Intervall $[i, j]$ in F sucht.

```
1: Function find(x,l,r)
2:   if (l < r) then
3:     m := (l + r) div 2;
4:     if (F[m] = x) then return m;
5:     if (F[m] < x) then return find(x, m + 1, r);
6:     if (F[m] > x) then return find(x, l, m − 1);
```

Abbildung 1: Such-Algorithmus.

Nehmen Sie an, dass die einzelnen Operationen konstante Laufzeit haben. Sei $T(n)$ die totale Anzahl Operationen, die der Such-Algorithmus ausführt beim Aufruf *find(x, 1, n)*, für ein beliebiges Element x. Stellen Sie eine rekursive Gleichung auf für $T(n)$ und leiten Sie dann ausführlich (mit allen Zwischenschritten und ohne Mastertheorem) eine geschlossene Form für $T(n)$ her. Was ist die asymptotische Laufzeit der Find-Operation?

$$T(n) = T\left(\frac{n}{2}\right) + c$$
$$= T\left(\frac{n}{4}\right) + c + c$$
$$= \ldots$$
$$= \sum_{i=1}^{\log n} c = c \log n$$
$$\in O(\log n)$$

b) [5 Punkte] Betrachten Sie den folgenden rekursiven Algorithmus *select*(S, k) zur Bestimmung des k kleinsten Elements einer Folge von n unsortierten und paarweise verschiedenen Zahlen S (wir vernachlässigen Rundungsfehler in dieser Teilaufgabe!):

$|S| = i$

1: **Function** select(S, k)
2: teile S in $|S|/5$ Blöcke auf; (* wir vernachlässigen Rundungsfehler *) $\Rightarrow O(i)$
3: sortiere jeden dieser Blöcke;
4: sei S' die Menge der Mediane dieser Blöcke;
5: setzte $m := select(S', |S'|/2)$; $T(\frac{i}{5})$
6: partitioniere S in die Menge S_1 der Elemente $< m$ und die Menge S_2 der Elemente $> m$; (* als Laufzeit kann hier $|S|$ angenommen werden *)
7: falls $k = |S_1| + 1$ ist, dann gib m zurück;
8: falls $k < |S_1| + 1$ ist, dann gib $select(S_1, k)$ zurück;
9: falls $k > |S_1| + 1$ ist, dann gib $select(S_2, k - |S_1| - 1)$ zurück; $\left\{ T(\frac{3i}{4}) \right.$

Abbildung 2: Selektion-Algorithmus.

Stellen Sie eine worst-case Rekursionsgleichung für die Laufzeit $T(n)$ des Algorithmus auf unter der Annahme, dass $|S_1|$ und $|S_2|$ immer höchstens $3|S|/4$ sind (was man auch formal beweisen kann) und alle Elementaroperationen in einer Zeiteinheit durchführbar sind. Zeigen Sie (mit allen Zwischenschritten und ohne Verwendung des Mastertheorems), dass die Laufzeit $T(n)$ von *select*(S, k) für $|S| = n$ und alle k in $O(n)$ ist. (Dazu können Sie z.B. vollständige Induktion verwenden).

$T(n) = T(\frac{n}{5}) + T(\frac{3n}{4}) + cn$ für eine Konstante c

Sei $p = \frac{1}{5}$ und $q = \frac{3}{4}$, dann:

$T(n) = T(pn) + T(qn) + cn$

$= T(p^2n) + T(pqn) + cpn + T(pqn) + T(q^2n) + cqn + cn$

$= 2T(pqn) + T(p^2n) + T(q^2n) + \underbrace{(p+q)}_{<1}cn + cn$

$= \dots$

$\leq 2^i \cdot 3 \cdot T(q^i n) + \sum_{j=0}^{i} (p+q)^j cn$

$\leq \dots$

$\leq 2^{\log n} \underbrace{T(1)}_{O(1)} + O(n) \in \underline{\underline{O(n)}}$

Aufgabe 3 [8 Punkte] While Schleifen

Wir betrachten zwei verschiedene Algorithmen zur Berechnung des grössten gemeinsamen Teilers (ggT) zweier Zahlen x und y. Beide Algorithmen beruhen auf While Schleifen. Sie dürfen im folgenden annehmen, dass eine einzelne While Schleife genau eine Zeiteinheit benötigt; für die Berechnung der Laufzeit der Algorithmen sind wir nur daran interessiert, *wie viele Male* die While Schleife durchlaufen wird.

a) [3 Punkte] Eine einfache Methode, den ggT zu bestimmen, ist *Algorithmus 1* (siehe unten). Bestimmen Sie eine geeignete Potenzialfunktion $\Phi(x, y)$ und weisen Sie damit nach, dass die Laufzeit im worst-case linear in x und y ist.

```
1: while (x ≠ y) {
2:     if (x > y) then x := x − y;
3:     else y := y − x;
4: }
```

Abbildung 3: Algorithmus 1.

Potenzialfunktion $\Phi(x,y) := x+y$

Startwert: $\Phi_0(x,y) = x+y$

Abnahme pro Iteration: $\Delta\Phi(x,y) \geq 1$

Endwert: $\Phi_{stop}(x,y) \geq 0$

$\Rightarrow$ Laufzeit $O(x+y)$

b) [5 Punkte] Der ggT kann aber auch effizienter berechnet werden, nämlich mit dem folgenden Algorithmus (siehe *Algorithmus 2*). Zeigen Sie, dass Algorithmus 2 lediglich $O(\log(x+y))$ viele Schritte benötigt. *Tipp: Sei c_i der Wert von c zu Beginn des i-ten Schleifendurchlaufs. Beweisen Sie eine geeignete Beziehung zwischen c_i und c_{i+2}.*

```
1: a := x; b := y;
2: while (b ≠ 0) {
3:     c := a mod b; a := b; b := c;
4: }
5: return a;
```

Abbildung 4: Algorithmus 2.

Wähle Potential $\Phi(a,b,c) = c$

Startwert $\Phi_0 \leq a+b$

Abnahme $\Phi_i \leq \frac{\Phi_{i-2}}{2}$ (*) $\Rightarrow$ Laufzeit $O(\log(x+y))$

Endwert $\Phi_{stop} \geq 1$

Beweis (*): $x_i = q_i y_i + c_i$ für $0 \leq c_i < y_i$

$y_i = q_{i+1} c_i + c_{i+1}$ für $0 \leq c_{i+1} < c_i$

zusammen 2 Schritte:

$y_i \geq c_i + c_{i+1} > 2 c_{i+1}$

und es gilt: $y_i = c_{i-1}$.

Aufgabe 4 [9 Punkte] Datenstrukturen

In der Vorlesung haben Sie verschiedene Datenstrukturen kennengelernt. Im folgenden werden wir zwei davon genauer betrachten: $(2,3)$-Bäume und Radix Heaps.

a) [4 Punkte] Betrachten Sie folgenden $(2,3)$-Baum.

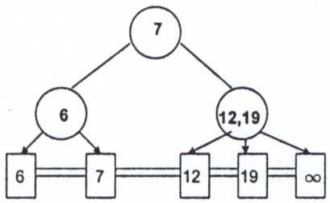
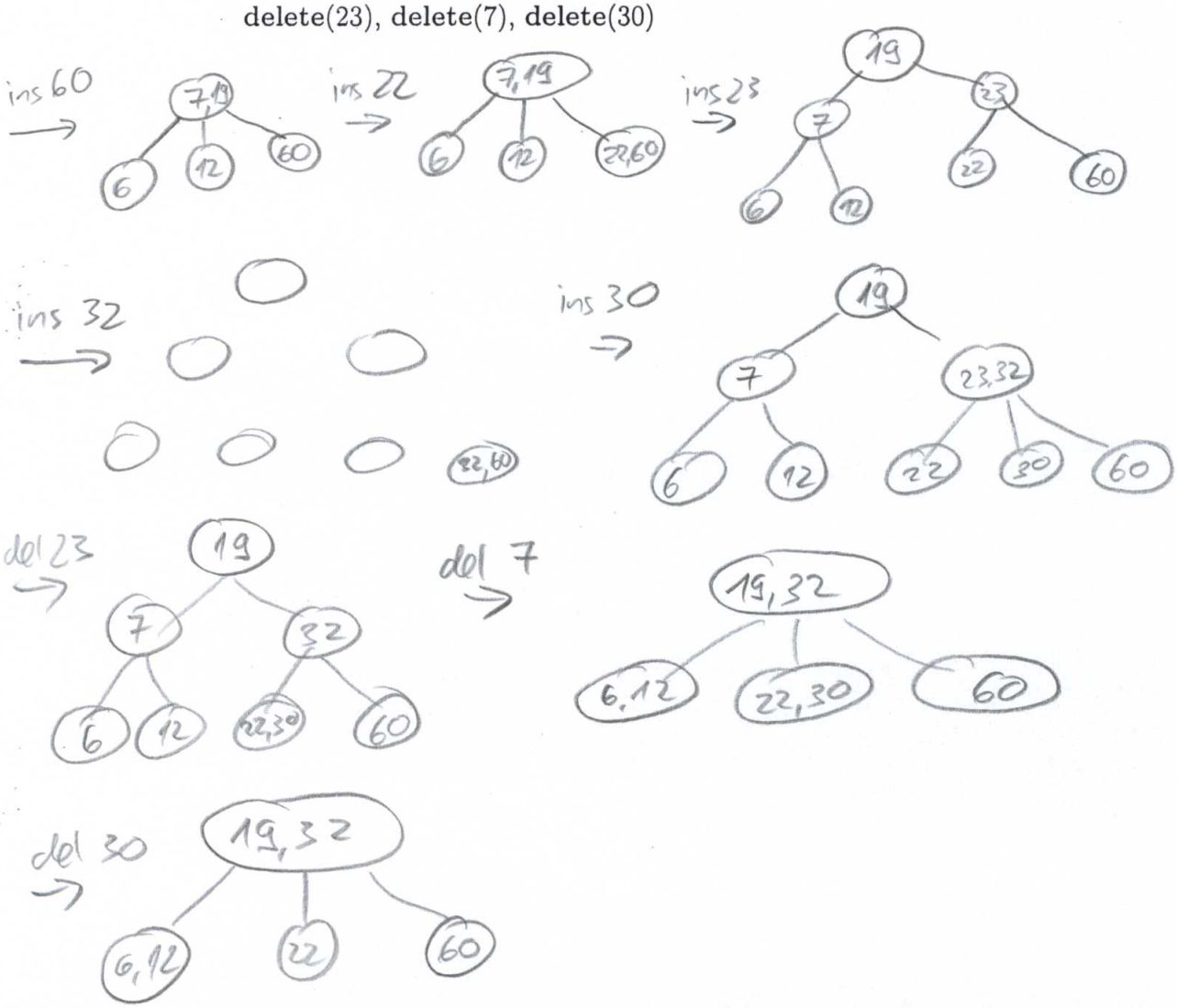


Abbildung 5: $(2,3)$-Baum.

Führen Sie folgende Operationssequenz auf diesem Baum aus, und zeichnen Sie dabei alle Zwischenschritte!

insert(60), **insert**(22), **insert**(23), **insert**(32), **insert**(30),

delete(23), **delete**(7), **delete**(30)

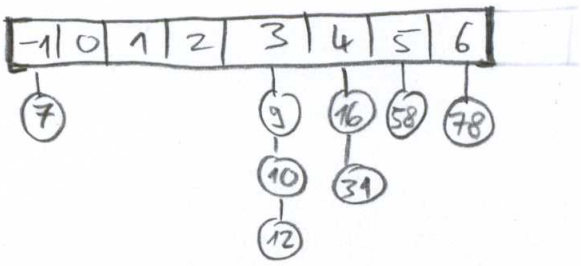


b) [5 Punkte]

i) Bestimmen Sie den Radix (Min-)Heap für folgende Elementmenge:

$$\{31, 10, 7, 78, 58, 12, 9, 16\}.$$

Geben Sie dazu alle Binärdarstellungen der Elemente an und berechnen Sie deren *msd*-Werte. Zeichnen Sie schliesslich den Heap, wie Sie es in der Vorlesung kennengelernt haben.



ii) Betrachten Sie folgenden Radix-Heap:

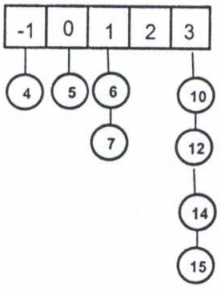
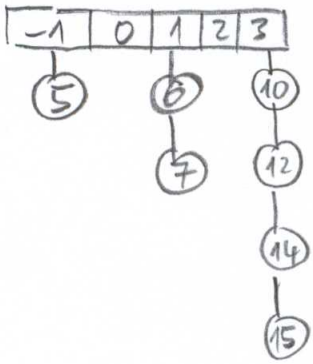
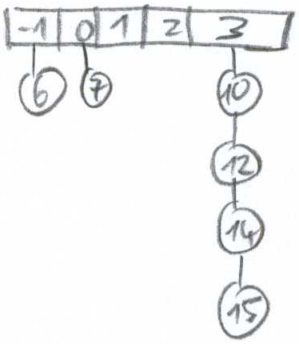
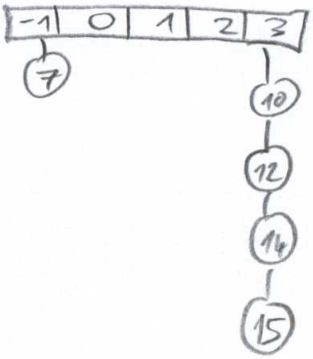


Abbildung 6: Radix-Heap.

Führen Sie auf diesem Heap *drei* DeleteMin Operationen aus, und zeichnen Sie jeweils den resultierenden Radix Heap!







Aufgabe 5 [6 Punkte] Kürzeste Wege

Sie haben gelernt, dass man das All-Pairs-Shortest-Paths (APSP) Problem auf beliebigen Graphen mit n Dijkstra-Anwendungen lösen kann, wenn man zuvor die Kantenkosten in geeignete positive Kosten umwandelt.

Betrachten Sie folgenden Graphen G:

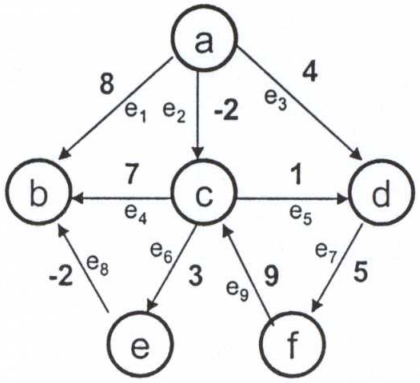


Abbildung 7: Ausgangslage fürs APSP Problem: Graph G.

Im folgenden wollen wir *zwei Schritte* des APSP Algorithmus ausführen.

a) [4 Punkte] Erweitern Sie G um eine künstliche Quelle s, die mittels 0-Kanten mit allen Knoten im Graphen verbunden ist, und berechnen Sie für s die kürzesten Wege! Geben Sie dabei die aktuellen Distanzen von s zu allen Knoten nach jedem Durchlauf des Bellmann-Ford Algorithmus über alle Graphkanten an. Benutzen Sie dazu folgende Tabelle, und arbeiten Sie die Kanten in der Reihenfolge $e_1, e_2, e_3, ..., e_9$ (siehe Abbildung) ab!

	s	a	b	c	d	e	f
Start	0	∞	∞	∞	∞	∞	∞
Runde 1	0	0	0	0	0	0	0
Runde 2	0	0	-2	-2	-1	0	0
Runde 3	0	0	-2	-2	-1	0	0
Runde 4							

done!

Abbildung 8: Lösungstabelle.

$\Phi(u) + c(e) - \Phi(v)$

b) [2 Punkte] Berechnen Sie die reduzierten Kantenkosten ~~$r(u,v) = \mu(s,v) + c(e) +$~~ $\mu(s,w) \geq 0$ für alle Kanten in G mit Hilfe der ermittelten Längen der kürzesten Wege in Teilaufgabe a). Tragen Sie Ihre Lösung in die Tabelle unten ein.

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9
r(e)	10	0	5	7	0	1	4	0	11

Abbildung 9: Lösungstabelle.