Technische Universität Berlin **SS 2013**
Fakultät II, Institut für Mathematik
Sekretariat MA 5–2, Frau Kiefer

Prof. Dr. Martin Skutella
Torsten Gellert
Martin Groß
Dr. Max Klimm

Aylin Acikel, Katharina Bütow, Christian Döblin,
Alexander Hopp, Daniel Kuske, Olivia Röhrig, Robert Rudow,
Daniel Schmand, Hendrik Schrezenmaier, Mona Setje-Eilers,
Judith Simon, Sebastian Spies, Fabian Wegscheider, Jan Zur

Probeklausur – Computerorientierte Mathematik II

Juli 2013

Name	
Vorname	
Matrikelnummer	
Studiengang	

Die hier vorliegende Zusammenstellung von Aufgaben soll euch einen Hinweis auf den Stil der Fragestellungen und die auftretenden Aufgabentypen in der Modulklausur geben. Das hier vorliegenden Schwierigkeitsniveau kann von der Modulklausur abweichen. Die angegebenen Punkte zeigen euch, worauf wir Wert legen.
Wir stellen euch **keine** Musterlösung zur Verfügung.

Wir wünschen dir viel Erfolg!

1. Aufgabe (7 Punkte)

(a) (1 Punkt) Definiere für eine Funktion $f\colon \mathbb{N} \to \mathbb{R}$ die Aussage: $f \in o(g)$

(b) (6 Punkte) Betrachte die Funktion $f\colon \mathbb{N} \to \mathbb{R}$ mit $f(n) \mapsto \frac{n}{\log(n)} - \log(n)/2$.
In welchen Mengen befindet sich die Funktion f. Es können jeweils mehrere Antworten richtig sein.

Nr.	Frage	$\mathcal{O}(g(n))$	$o(g(n))$	$\Omega(g(n))$	$\Theta(g(n))$
1	$g(n) := \log n$	○	○	○	○
2	$g(n) := n$	○	○	○	○
3	$g(n) := 1$	○	○	○	○
4	$g(n) := n\log(n)$	○	○	○	○
5	$g(n) := n^2$	○	○	○	○
6	$g(n) := n/\log n$	○	○	○	○

2. Aufgabe (3 Punkte)

Verwende den allgemeinen ABS zur asymptotischen Lösung der Rekursionsgleichung $T(2n) = 16 \cdot T(n) + 5 \cdot n^4 + \log(n)$ mit $T(1) = 1$.

Formuliere hierzu den anzuwendenden Fall und weise gegebenenfalls nach, dass die Voraussetzungen zur Anwendung des benutzten Falls erfüllt sind. Wir nehmen jeweils an, dass n eine passende Größe hat, so dass wir uns nicht um Abrunden und Aufrunden kümmern müssen.

3. Aufgabe (10 Punkte)

Für diese Aufgaben nehmen wir einen gerichteten Graphen $G = (V, E)$ mit einer Kantenbewertungsmatrix A als gegeben an und wollen kürzeste Wege berechnen.

(a) (5 Punkte) Berechne für die Kantenbewertungsmatrix A und die Matritzen $U^{(2)}$ und $U^{(3)}$ mit Hilfe der speziellen Matrixmultiplikation des Bellman-Ford-Algorithmus. Hierbei sollen auch die Tree-Matritzen $T^{(2)}$ und $T^{(3)}$ berechnet werden. Wie lautet ein kürzester Weg vom Knoten 2 zum Knoten 4?

$A =$

	1	2	3	4
1	0	∞	∞	4
2	-2	0	∞	3
3	1	1	0	∞
4	∞	∞	-2	0

$U^{(2)} =$

	1	2	3	4
1				
2				
3				
4				

$T^{(2)} =$

	1	2	3	4
1				
2				
3				
4				

$U^{(3)} =$

	1	2	3	4
1				
2				
3				
4				

$T^{(3)} =$

	1	2	3	4
1				
2				
3				
4				

(e) (5 Punkte) Formuliere äquivalente Kriterien für die Existenz eines negativen Zykels und definiere und verwende hierzu den Begriff eines elementaren negativen Zykels. Beweise anschließend die Äquivalenz der Kriterien.

4. Aufgabe (18 Punkte)

(a) (3 Punkte)

(i) Beschreibe die **drei** wesentlichen Schritte von Quicksort.
(Technische Details wie `hiSwap`, `loSwap`, Zeiger, etc. sind nicht erwünscht.)

(ii) Gib die im Best-, Average und Worst-Case benötigten Vergleiche von Quicksort in Θ-Notation an, wenn n Elemente sortiert werden sollen und immer das erste Element als Pivotelement gewählt wird.

(b) (2 Punkte) Es sollen sieben-elementige Arrays, die jede Zahl von 1 bis 7 einmal enthalten, sortiert werden. Gib jeweils ein Array an, das einen Best- bzw. Worst-Case für die Rekursionstiefe von Quicksort darstellt. Als Pivotelement wird immer das erste Element gewählt.

(c) (2 Punkte) Du hast ein Paper von Henry Hieronymus Hatter vorliegen, in dem der neue Sortieralgorithmus QUICKERSORT vorgestellt wird. Der neue Algorithmus unterscheidet sich von Quicksort allerdings nur darin, dass der Array statt in zwei Teile in vier Teile geteilt wird – mittels dreier Pivotelemente und entsprechendem Mehraufwand. Ist dieser Algorithmus asymptotisch schneller als QuickSort? Gehe davon aus, dass beide Algorithmen vergleichbare Pivotelemente wählen und begründe deine Antwort in einem oder zwei Sätzen – hier ist kein formaler Beweis erforderlich.

(d) (2 Punkte) Die Firma QS-Systems hat dich beauftragt, mit ihrer Quicksort-Bibliothek ein "fast vollständig sortiertes Array" zu Ende zu sortieren. Welches Pivotelement solltest du die Bibliothek wählen lassen, und wieso? Ein Satz Begründung genügt.

(e) (4 Punkte) Das Münchhausen Institut für Technologien hat einen $\mathcal{O}(1)$ Algorithmus zur Berechnung des Medians eines beliebigen Arrays vorgestellt. Der Median eines Arrays mit den Werten $a_1 < a_2 < \cdots < a_n$ ist hier definiert als $a_{\lfloor \frac{n}{2} \rfloor}$. Kannst du mit diesem Algorithmus die von Quicksort im Worst-Case benötigten Vergleiche reduzieren? Begründe deine Behauptung kurz. Falls du die benötigten Vergleiche reduzieren kannst, gib die neue Worst-Case-Anzahl an (in Θ-Notation) und beweise sie. Du darfst Gaußklammern ($\lfloor \cdot \rfloor$, $\lceil \cdot \rceil$) in deinem Beweis vernachlässigen.

(f) (5 Punkte) Wir betrachten nun Quicksort mit der Pivotregel, dass jeweils ein zufälliges Element als Pivotelement gewählt wird.

Gib eine Rekursionsformel für die erwarteten benötigten Vergleiche in Abhängigkeit von der Anzahl der Elemente n an. Beweise mit Hilfe der Rekursionsformel die im *erwarteten* Worst-Case benötigten Vergleiche des Algorithmus.

Hinweis: In der Average-Case-Analyse des herkömmlichen QuickSort findet sich eine nützliche Technik.

5. Aufgabe (2 Punkte)

Betrachte InsertionSort für das Array $A = [a_1, a_2, a_3, a_4, a_5]$.

InsertionSort

Input: Array A mit n Komponenten $[a_1, \ldots, a_n]$

Output: A in aufsteigender Reihenfolge sortiert

s:=2

FOR i:=2 **TO** n **DO**

  `cur` := $A[i]$

  finde Einfügeposition j für `cur` in $A[1] \ldots A[s]$ mit linearer Suche von vorne

  verschiebe Elemente von $A[j] \ldots A[i-1]$ um eine Position nach rechts

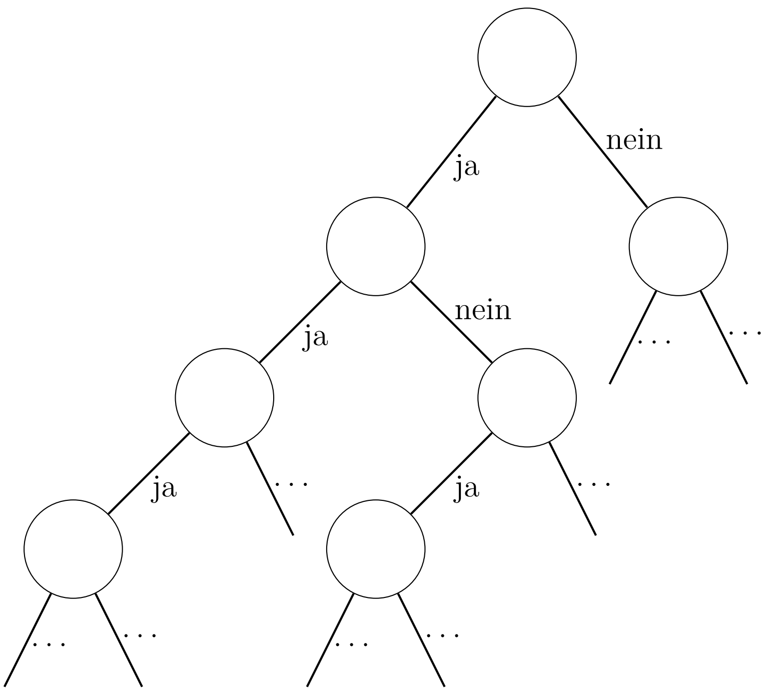
  $A[j]$:= `cur`

  $s := s + 1$

ENDFOR

RETURN A

Trage die entsprechenden Vergleiche in der Form $x < y$ in die vorgegebenen 8 Knoten des Entscheidungsbaums ein. Wie drückt sich Worst-Case-Aufwand in einem Entscheidungsbaum aus?



6. Aufgabe (4 Punkte)

(a) (1 Punkt) Wann ist ein Code *eindeutig dekodierbar*?

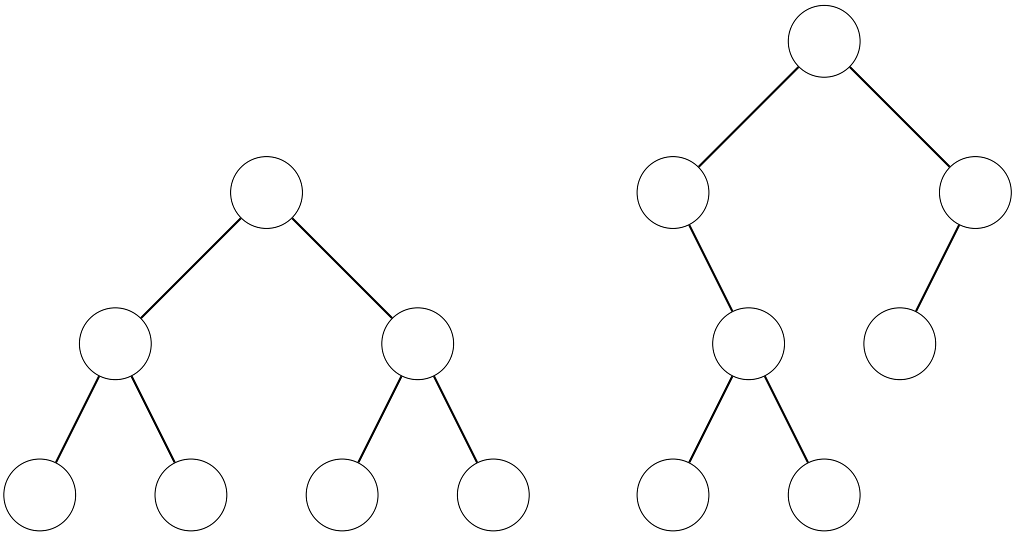
(b) (1 Punkt) Wie ist ein *Präfixcode* definiert?

(c) (2 Punkte) Stelle den folgenden Code als Baum dar. Ist der Code ein Präfixcode?

Zeichen	Codewort
a	0
b	1
c	00
d	01
e	10
f	11
g	000
h	001

7. Aufgabe (9 Punkte)

(a) (2 Punkte) Füge die Schlüssel $1, 2, \ldots, 7$ in die jeweiligen Bäume ein, so dass Suchbäume folgender Gestalt entstehen:



(b) (2 Punkte) Lösche aus beiden Bäumen das Wurzelelement und stelle die entstehenden Suchbäume dar. Erkläre dabei, wie die neue Lösung zustande kommt.

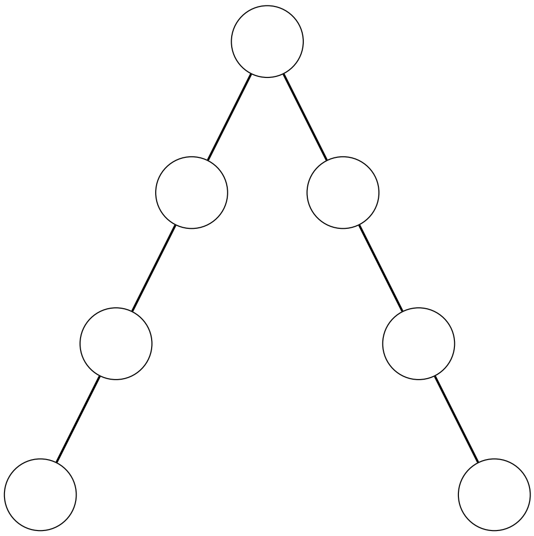
(c) (1 Punkt) Definiere die Höhe $h(T)$ eines binären Suchbaums T rekursiv.

(d) (4 Punkte) Angenommen, die Suche nach einem Schlüssel k in einem Suchbaum T endet in einem Blatt und die Schlüsselmenge von T wird in die drei Mengen A, B, C mit folgender Eigenschaft zerlegt: A enthält die Schlüssel links des Suchpfades, B enthält genau die Schlüssel auf dem Suchpfad und C enthält die Schlüssel rechts des Suchpfades. Finde ein möglichst kleines Gegenbeispiel zu der Behauptung, dass dann gilt: $a \leq b \leq c$ für beliebige $a \in A$, $b \in B$, $c \in C$.

8. Aufgabe (3 Punkte)

(a) (1 Punkt) Wie lauten die Rekursionsgleichungen zur Berechnung eines optimalen statischen Suchbaums?

(b) (2 Punkte) Gib ein Beispiel mit 7 Schlüsseln und zugehörigen strikt positiven Zugriffshäufigkeiten an, so dass der zugehörige optimale statische Suchbaum folgende Struktur hat. Begründe deine Vorgehensweise.



9. Aufgabe (12 Punkte)

(a) (1 Punkt) Was ist die Ausgangssituation beim Hashing? (Fasse dich kurz!) Wie ist eine Kollision definiert?

(b) (2 Punkte) Chaining mit Überlauflisten ist ein Verfahren, welches Kollisionen behandeln kann. Wie funktioniert es? Wie hoch ist der Aufwand im Average-Case für die Operation "Suchen nach einem Element", wenn sich n Schlüssel in der Hashtabelle befinden?

(c) (1 Punkt) Wie ist die Permutationsbedingung definiert und wofür sorgt sie beim Hashing?

(d) (2 Punkte) Für Hashing wurden in der Vorlesung unter anderem die Kollisionsstrategien quadratisches Sondieren und der Doppel-Hash vorgestellt. Die Kollisionsbehandlung bei einer Hashfunktion wird für einen Schlüssel k durch eine Sondierungsfolge $h(k, 0)$, $h(k, 1)$, $h(k, 2)$ usw. festgelegt. Gib für die beiden Strategien die jeweilige Hashfunktion an.

(e) (1 Punkt) Formuliere die Bedingung, die äquivalent zur Permutationsbedingung beim Doppel-Hash ist (ohne Beweis).

(f) (5 Punkte) Beim Beweis zur Anzahl der notwendigen Sondierungen beim Doppel-Hash bis ein Einfügeplatz gefunden wurde, kommt das Urnenmodell zum Einsatz. Definiere das Urnenmodell und gib eine Formel für die erwartete Anzahl an Ziehungen einer weißen Kugel (ohne Beweis) an. Nutze nun diese Aussage zum Beweis einer Aussage über die erwartete Anzahl an Sondierungen beim Doppelhash.

10. Aufgabe (4 Punkte)

Es sei eine Hashtabelle mit $m \geq n+1$ Speicherplätzen gegeben, in die gleichverteilte Schlüssel eingefügt werden. Es sind schon n Schlüssel zufällig eingefügt worden seien. Für die Einfügung des $n+1$-ten Schlüssels gilt die Rekursionsformel $f(n,m) = 1 + \frac{n}{m} \cdot f(n-1, m-1)$, welche die erwartete Anzahl an besuchten Speicherzellen bei Einfügung des $n+1$-ten Schlüssels angibt.

(a) (2 Punkte) Erkläre, warum die behauptete Rekursionsformel gilt.

(b) (1 Punkt) Beweise, dass $f(n,m) = \frac{m+1}{m+1-n}$ gilt.

(c) (1 Punkt) Wie ist der Zusammenhang zur Abschätzung der erwarteten Anzahl an Sondierungen beim Doppel-Hash?

11. Aufgabe (6 Punkte)

(a) (3 Punkt) Definiere *rekursiv* und *rekursiv aufzählbar*. Verwende die Begriffe *Alphabet*, *Sprache* und *Turingmaschine*. Was ist der Unterschied zwischen rekursiv und rekursiv auifzählbar?

(b) (1 Punkt) Definiere das Halteproblem.

(c) (2 Punkte) Zeige durch Reduktion auf das Halteproblem, dass folgende Sprache nicht entscheidbar ist:

$$L := \{\langle M\rangle \;\big|\; M \text{ hält für keine Eingabe } \}$$

12. Aufgabe (3 Punkte)

Betrachtet folgende Turingmaschine $M := (Q, \Sigma, \Gamma, q_0, B, \delta, F)$ mit den Zuständen $Q := \{q_0, a_0, a_1, a_2, s_0, s_1, s_2, q_1\}$, dem Alphabet $\Sigma := \{0, 1\}$, dem Bandalphabet $\Gamma := \Sigma \cup \{B\}$, dem Leerzeichen $B \in \Gamma \backslash \Sigma$, den akzeptierenden Endzuständen $F := \{q_1\}$ und der Übergangsfunktion $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, N, R\}$:

δ	0	1	B
q_0	$(q_0, 0, R)$	$(q_0, 1, R)$	(a_0, B, L)
q_1	$-$	$-$	$-$
a_0	(s_0, B, L)	(s_1, B, L)	(q_1, B, N)
a_1	(s_1, B, L)	(s_2, B, L)	$-$
a_2	(s_2, B, L)	(s_0, B, L)	$-$
s_0	(a_0, B, L)	(a_2, B, L)	(q_1, B, N)
s_1	(a_1, B, L)	(a_0, B, L)	$-$
s_2	(a_2, B, L)	(a_1, B, L)	$-$

(a) (1 Punkt) Gib für die Eingabe “110” die Konfigurationsfolge der Turingmaschine M an.

(b) (1 Punkt) Akzeptiert die Turingmaschine für die Eingaben “11” und “101”?

(c) (1 Punkt) Fasse die Eingaben als Binärzahlen auf. Welche mathematische Fragestellung wird von der Turingmaschine beantwortet?

13. Aufgabe (2 Punkte)

Eine Turingmaschine heißt *einseitig beschränkt*, wenn sie niemals eine Position links der Startposition benutzt. Es sei eine Turingmaschine $M = (Q, \Sigma, \Gamma, q_0, B, \delta, F)$ gegeben. Beschreibe informell eine Turingmaschine M', die sich wie M verhält, aber einseitig beschränkt ist.