Technische Universität Berlin **SS 2012**
Fakultät II, Institut für Mathematik
Sekretariat MA 5–1, Frau Klink

Prof. Dr. Rolf Möhring
Torsten Gellert
Jan-Philipp Kappmeier
Jens Schulz

Catharina Broermann, Christian Döblin, Pierluigi Ferrari,
Alexander Müller, Benjamin Müller, Yannick Reichelt,
Robert Rudow, Christopher Ryll, Daniel Schmand,
Judith Simon, Fabian Wegscheider, Jan Zur

Probeklausur – Computerorientierte Mathematik II

September 2012

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!

$\mathcal{O}$-Notation

1. Aufgabe

(5.5 Punkte)

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

(b) [4,5 Punkte] Es sei $f : \mathbb{N} \to \mathbb{R}$ mit $f(n) := \frac{n}{\log(n)} - 0,5 \cdot \log(n)$. Welche der folgenden Aussagen sind korrekt? Es gibt je $\frac{1}{2}$ Punkt pro richtiger Antwort und $-\frac{1}{2}$ bei falscher Antwort.

Nr.	Frage	Ja	Nein
1	$f(n) \in \Omega(\log(n))$	( )	( )
2	$f(n) \in \Omega(n)$	( )	( )
3	$f(n) \in O(1)$	( )	( )
4	$f(n) \in O(n)$	( )	( )
5	$f(n) \in \Theta(n)$	( )	( )
6	$f(n) \in \Theta(n \log(n))$	( )	( )
7	$f(n) \in o(\log(n))$	( )	( )
8	$f(n) \in o(n)$	( )	( )
9	$f(n) \in o(n^3)$	( )	( )

Aufteilungs- und Beschleunigungssatz

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.

Kürzeste Wege

3. Aufgabe (11 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) [3 Punkte] Berechne für die Kantenbewertungsmatrix A und die Matrix $U^{(2)}$ die leeren Stellen in der Matrix $U^{(3)}$ mit Hilfe der speziellen Matrixmultiplikation des Bellman-Ford-Algorithmus. Hierbei soll die Tree-Matrix $T^{(3)}$ gleichzeitig aktualisiert 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	0	∞	+2	4
2	-2	0	1	2
3	-1	1	0	4
4	-1	-1	-2	0

$T^{(2)} =$

	1	2	3	4
1	-1	-1	4	1
2	2	-1	4	1
3	2	3	-1	2
4	3	3	4	-1

$U^{(3)} =$

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

$T^{(3)} =$

	1	2	3	4
1	-1	4	1	-1
2	2			
3				
4				

(b) [5 Punkt] Gib den Bellman-Ford Algorithmus in Pseudocode an. Als Eingabe sind ist ein Graph $G = (V, E)$ mit einer Kantenbewertungsmatrix A gegeben. Berechnet werden soll die Matrix der kürzesten Distanzen $U^{(m)}$ und die Matrix der Vorgängerknoten.

(c) [3 Punkt] Welche Laufzeit hat der Bellman-Ford Algorithmus? Beweise deine Aussage.

Quicksort

4. Aufgabe (9 Punkte)

a) [1,5 Punkte] Welche der drei Eigenschaften *adaptiv*, *stabil* und *in-place* hat Quicksort? Mache deutlich, wenn eine Eigenschaft von der Wahl des Pivotelements abhängt. Zur Erinnerung:

- **Adaptiv.** Ein Sortieralgorithmus ist adaptiv, wenn ein sortierter Array eine Best-Case-Eingabe für ihn ist und er auf dieser Eingabe asymptotisch schneller als im Worst-Case ist.
- **In-Place.** Ein Sortieralgorithmus arbeitet in-place, wenn er für jede Eingabe nur $\mathcal{O}(1)$ zusätzlichen Speicher benutzt.
- **Stabil.** Ein Sortieralgorithmus ist stabil, wenn für jede Eingabe zwei (bezüglich der Sortierordnung) gleiche Elemente in der Ausgabe in derselben Reihenfolge stehen wie in der Eingabe.

b) [1,5 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.

Geht nicht schneller

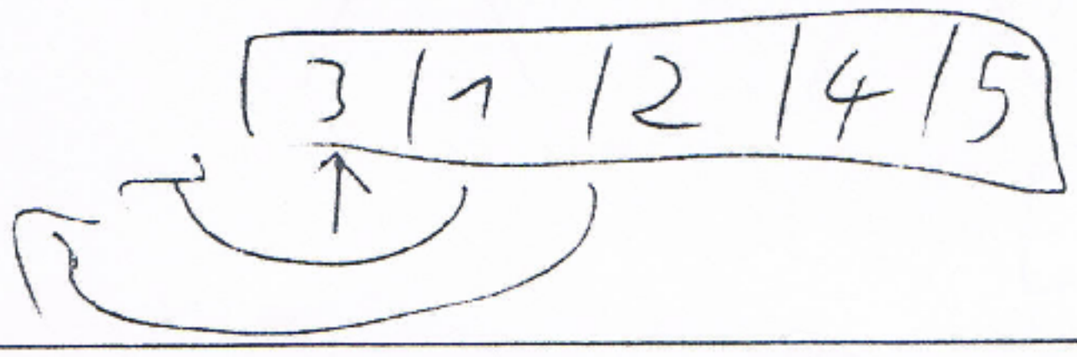
c) [2 Punkte] Die Firma QS-Systems hat dich beauftragt, mit ihrer Quicksort-Bibliothek ein "fast vollständig sortiertes Array" zu Ende zu sortieren. Welches Pivotelement solltest du die Bibliothek wählen lassen, und wieso? Ein Satz Begründung genügt.

Mittleres

d) [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.



Untere Sortierschranke

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

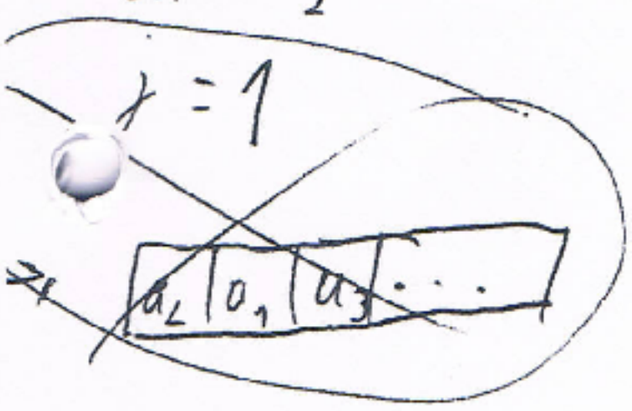
RETURN A

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?

s=2

i=2

cur = a_2



j = 1

cur = a_2

s = 3

i = 3

cur = a_3

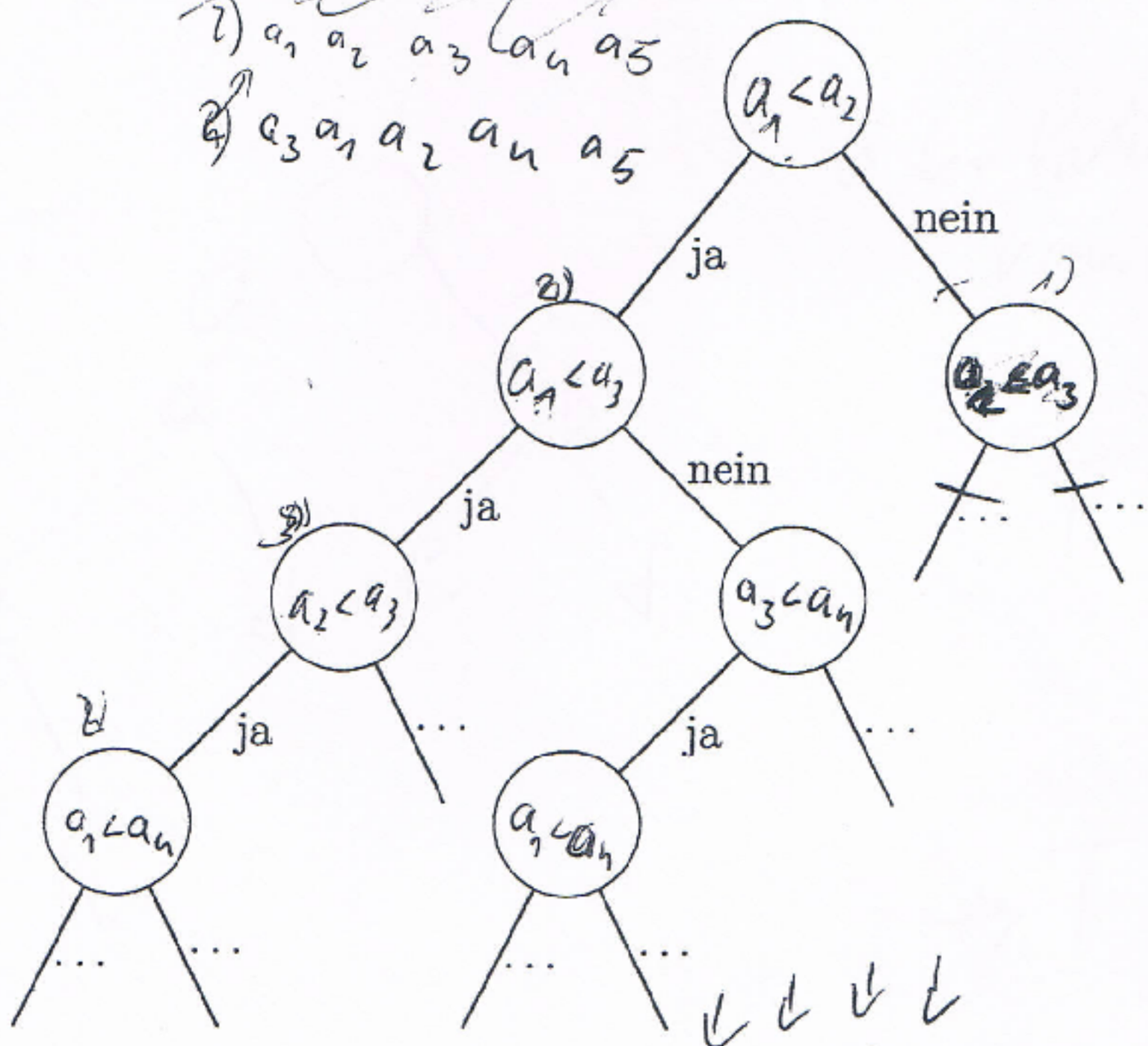
1) a_2 a_1 a_3 a_4 a_5

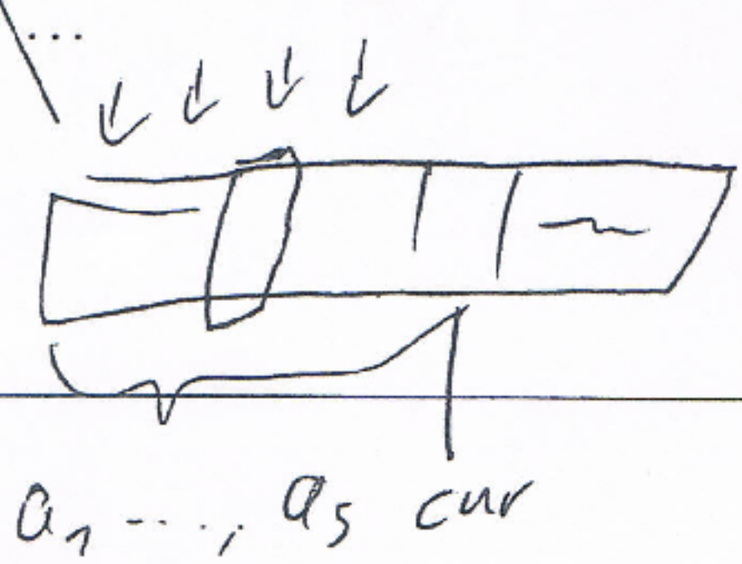
2) a_1 a_2 a_3 a_4 a_5

3) a_3 a_1 a_2 a_4 a_5

Anzahl d. Knoten

Höhe





Kompression

6. Aufgabe

(8 Punkte)

a) [1 Punkt] Wann ist ein Code *eindeutig dekodierbar*? Verschiedene Daten führen zu verschiedenen codierten Daten

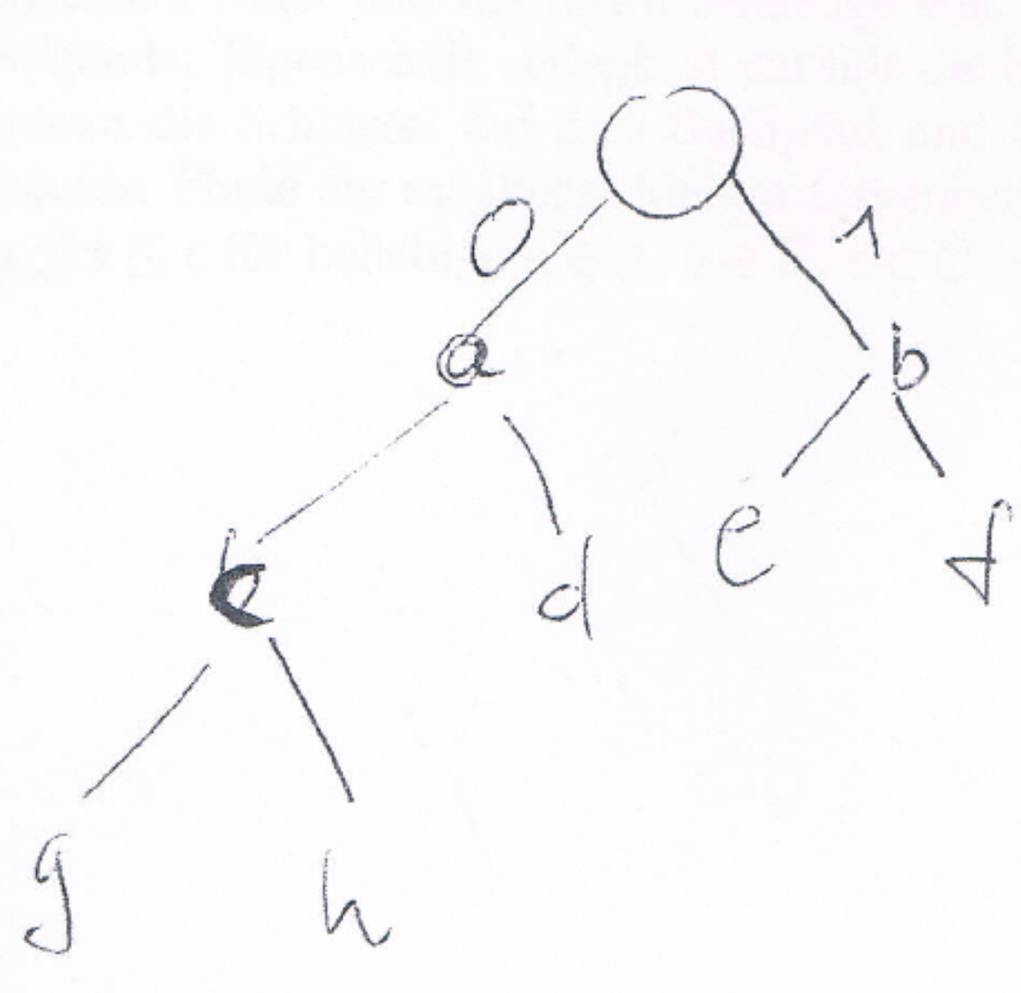
b) [1 Punkt] Wie ist ein *Präfixcode* definiert?

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

e) (4 Punkte) Beweise folgendes Lemma: sei Σ ein Alphabet, x und y Zeichen mit minimaler Häufigkeit. Sei T ein Präfixcode in dem x und y einen gemeinsamen Elter haben und T' der Baum, der entsteht wenn x und y durch z ersetzt werden, mit der Summe der Häufigkeiten $f'(z) = f(x) + f(y)$ und $f'(a) = f(a)$ sonst. T' ist ein Präfixcode für das Alphabet $\Sigma' := \Sigma \setminus \{x, y\} \cup \{z\}$. Dann gilt:

$$T \text{ optimal für } \Sigma, f \Leftrightarrow T' \text{ ist optimal für } \Sigma', f'$$



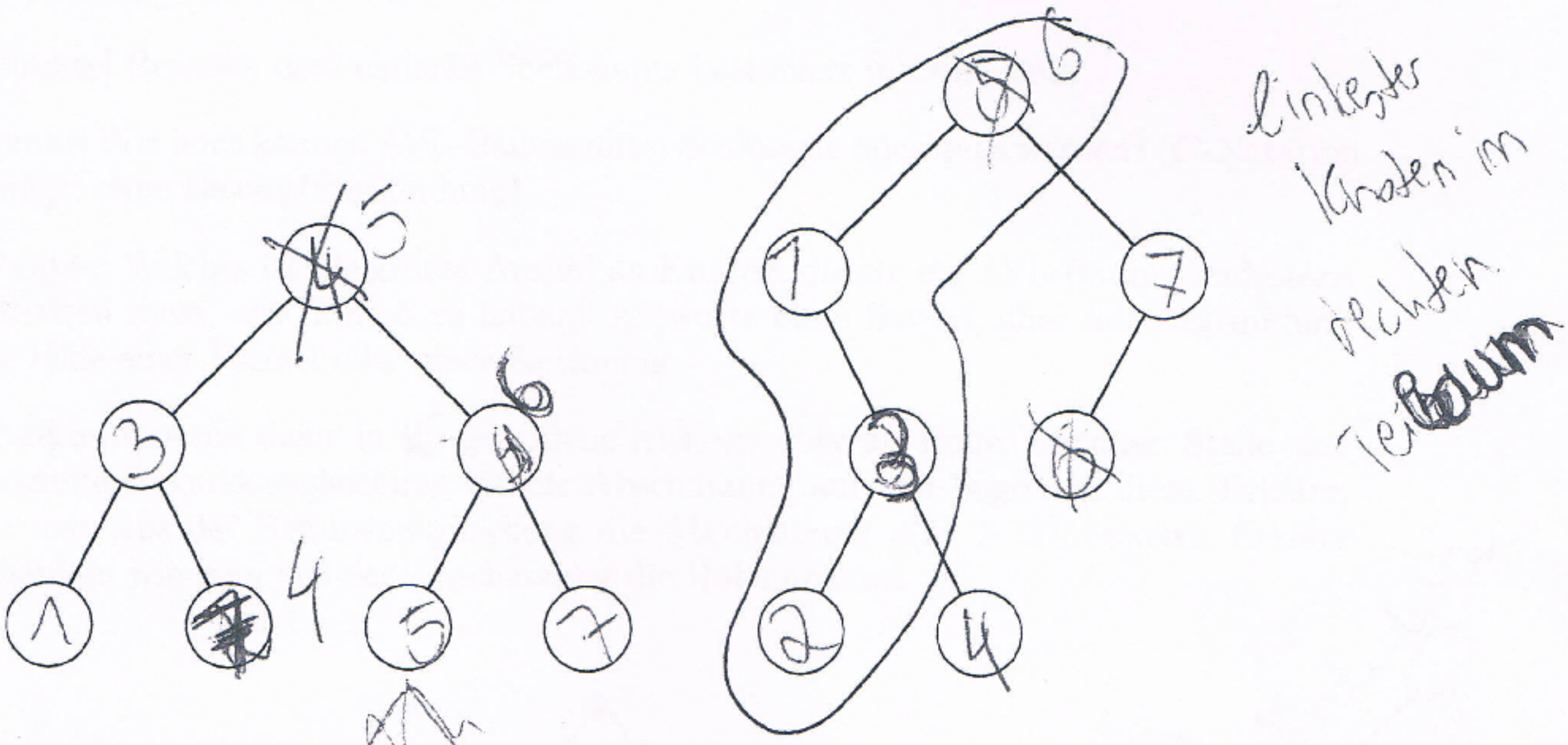
z der Vater von x

Suchbäume

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 $a \in A$, $b \in B$, $c \in C$.

AVL-Bäume

8. Aufgabe

(14 Punkte)

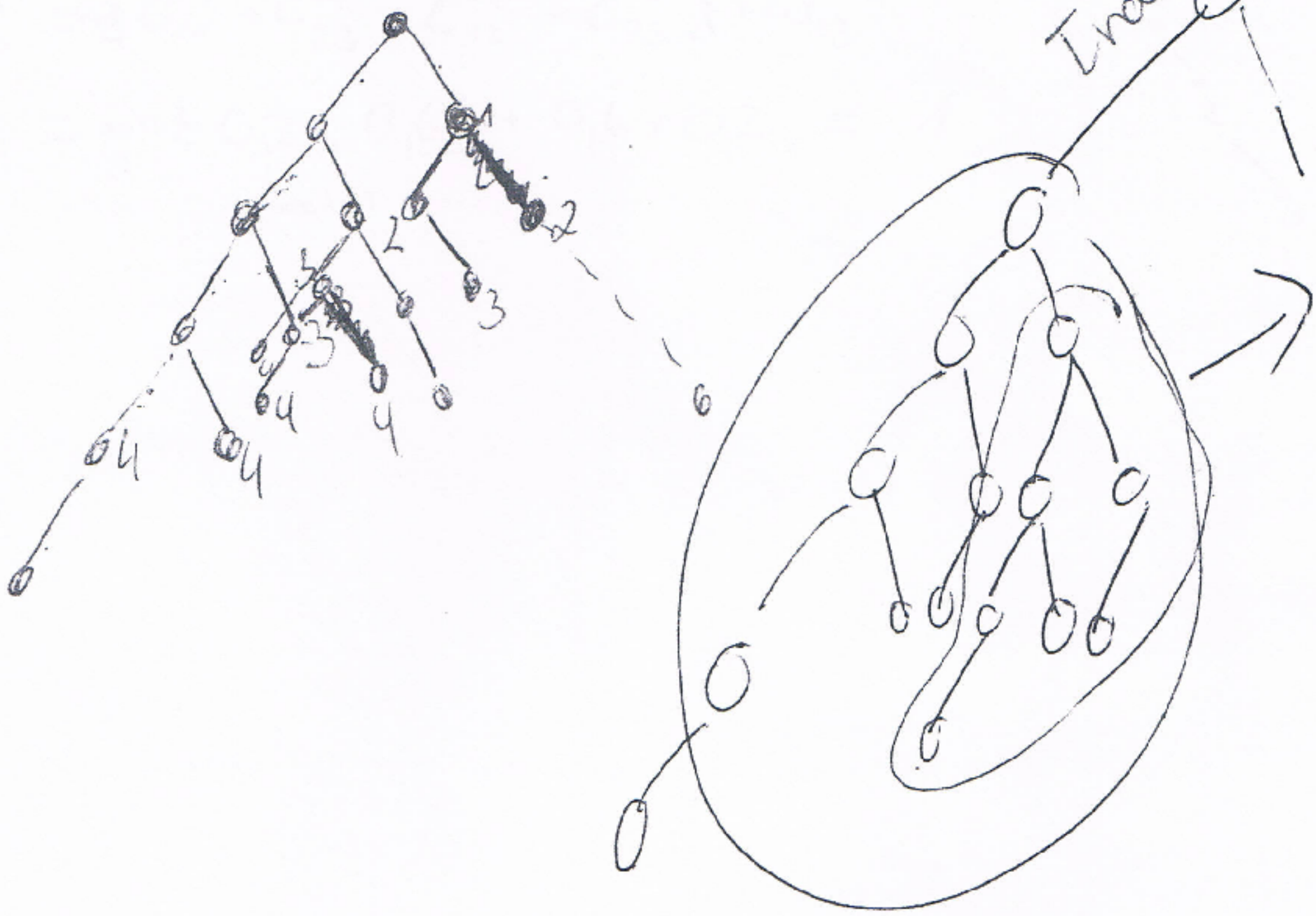
(a) [3 Punkte] Was bedeutet die Aussage, dass eine Klasse von Suchbäumen balanciert in $f(n)$ ist.

(b) [3 Punkte] Beweise, dass einfache Suchbäume balanciert in $\mathcal{O}(n)$ sind.

(c) [1 Punkt] Wie hoch können AVL-Bäume mit n Schlüsseln höchstens werden? ($\mathcal{O}$-Notation genügt, ohne Beweis/Begründung)

(d) [2 Punkte] Welches ist die größte Anzahl an Knoten, die ein ein AVL-Baum mindestens enthalten muss, um Höhe 5 zu haben? Antworte ohne Beweis, aber mit Begründung mit Hilfe einer Formel oder einer Zeichnung.

(e) [5 Punkte] Beweise deine in (b) getroffene Aussage über die Höhe. Genauer: Stelle eine geeignete Rekursionsgleichung für die Abschätzung auf und begründe diese. Erkläre, wie man aus der Rekursionsgleichung die Abschätzung $n(h) \geq 2^{\frac{h}{2}}$ beweist. Erkläre außerdem wie man aus der Abschätzung die Höhe ableitet.



Optimale statische Suchbäume

9. Aufgabe

(4 Punkte)

(a) [1 Punkt] Wie lauten die Rekursionsgleichungen zur Berechnung eines optimalen statischen Suchbaums? $C_{ii}^{opt} = 0$; $C_{ij}^{opt} = \min_{k=i+1,\dots,j} \{ C_{i,k-1}^{opt} + C_{ik}^{opt} \} + \omega_{ij}$

(b) [3 Punkte] Berechne und zeichne für die vier Schlüssel $\{1, 2, 3, 4\}$ und die nachfolgende Wahrscheinlichkeitsverteilung den optimalen statischen Suchbaum. Nutze hierzu den Algorithmus aus der Vorlesung. Für Intervalle der Länge 1 ist keine Begründung anzugeben.

Schlüssel	1	2	3	4
Wahrscheinlichkeit	0.1	0.6	0.2	0.1

IL 0: $C_{ii}^{opt} = 0$

IL 1: $C_{12}^{opt} = 0,6$; $C_{23}^{opt} = 0,2$; $C_{34}^{opt} = 0,1$

IL 2: $C_{13}^{opt} = \min\{ C_{11}^{opt} + C_{23}^{opt} ; C_{12}^{opt} + C_{33}^{opt} \} + \omega_{13}$

$= \min\{ 0,2 ; 0,6 \} + 0,6 + 0,2 = 1$

Wurzel 1

Hashing

10. Aufgabe (11 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) [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.

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

(e) [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.

11. Aufgabe (3,5 Punkte)

Es sei eine Hashtabelle mit $m \geq n+1$ Speicherplätzen gegeben, in die schon n Schlüssel zufällig durch Hashing mit offener Adressierung (Gleichverteilungsannahme / uniform hashing!) 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) [1,5 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?