Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Institut für Technische und Betriebliche Informationssysteme
Prof. Dr. A. Nürnberger

Magdeburg, 29.07.08

Prüfung

Algorithmen und Datenstrukturen II

Name:	Matrikelnummer:
Vorname:	Studiengang:
Blattanzahl:	
Unterschrift Student/in:	Unterschrift Aufsicht:

Aufg. 1	Aufg. 2	Aufg. 3	Aufg. 4	Aufg. 5	Summe
(von 10)	(von 10)	(von 11)	(von 3)	(von 6)	(von 40)

Allgemeine Hinweise

- Schreiben Sie auf jedes Blatt Ihren Namen, Ihre Matrikelnummer und die Seitennummer.
- Die Programme sind gut zu kommentieren! Programmstrukturen sind je nach Aufgabenstellung als Java-ähnlicher Pseudocode oder in Java anzugeben.
- Beginn und Ende einer Aufgabenlösung sind durch einen waagerechten Strich deutlich zu kennzeichnen. Ungültige Lösungen sind durchzustreichen.
- Zugelassene Hilfsmittel: ausschließlich Schreibmaterialien
- Die Klausur besteht aus 5 Aufgaben, die Bearbeitungszeit beträgt 120 Minuten.
- Bitte **nicht** mit grünem oder rotem Stift schreiben.

Aufgabe 1 (2+3+5 = 10 Punkte)

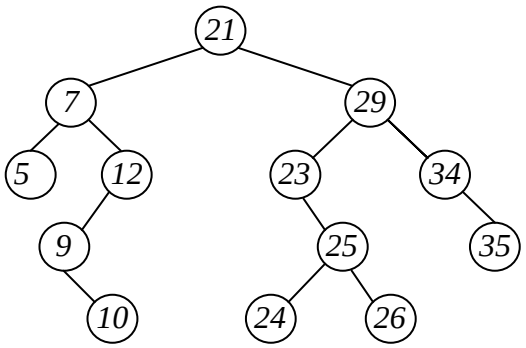
Die Klassen *SearchTree<T extends Comparable<T>>* und *TreeNode<T extends Comparable<T>>* stehen für a) und c) mit den folgenden Methoden zur Verfügung:

```
TreeNode<T extends Comparable<T>>
    public TreeNode();
    public TreeNode(T x);
    public TreeNode(TreeNode<T> r);
    public T getKey();
    public TreeNode<T> getLeft();
    public TreeNode<T> getRight();
    public void setKey(T d);
    public void setLeft(TreeNode<T> l);
    public void setRight(TreeNode<T> r);

SearchTree<T extends Comparable<T>>
    public SearchTree();
    private TreeNode<T> getRoot();
    private void setRoot(TreeNode<T> r);
```

a) Um in einem binären Suchbaum einen **inneren** Knoten *p* (mit zwei Nachfolgern) zu löschen, sucht man zuerst seinen symmetrischen Nachfolger (das sei hier der Knoten mit dem kleinsten Schlüssel im rechten Teilbaum). Schreiben Sie eine Java-Methode *symSucc*, die den symmetrischen Nachfolger des Knotens *p* liefert.

b) Gegeben sei folgender binärer Suchbaum:



- Löschen Sie aus diesem Baum den Knoten mit dem Schlüsselwert 21. In welchen Schritten gehen Sie vor? Wie sieht der Baum nach dem Löschen aus?
- Wie viele Knoten kann ein Suchbaum mit der Höhe *h=5* maximal aufnehmen?
- Wie lang ist der Suchweg von der Wurzel zu einem beliebigen Knoten in einem binären Suchbaum mit *n* Knoten im besten, im mittleren und im schlechtesten Fall in O-Notation?

c) Schreiben Sie eine Java-Methode *negate* innerhalb der Klasse *SearchTree*, die einen binären Suchbaum mit **ganzen** Zahlen (Datentyp muss nicht überprüft werden) als Knoteninhalt so verändert, dass alle Knoteninhalte negiert werden (also ein Knoteninhalt *x* durch *–x* ersetzt wird) und sich dennoch wieder ein Suchbaum ergibt. Das Vergleichskriterium für die Knoteninhalte soll dabei unverändert bleiben.
Skizzieren Sie den Suchbaum für das obige Beispiel nach Anwendung der Methode *negate*.

Aufgabe 2 (2+2+2+2+2 = 10 Punkte)

a) Durch welche Eigenschaften ist ein **AVL-Baum** gekennzeichnet? Welchen Vorteil hat ein AVL-Baum gegenüber einem einfachen binären Suchbaum?
Wenn Sie Werte in sortierter Reihenfolge in einen AVL-Baum einfügen, wird der Aufwand dann größer oder kleiner als bei einem Einfügen in zufälliger Reihenfolge? Begründen Sie kurz ihre Antwort.

b) Fügen Sie die Werte: 7, 4, 2, 5, 8, 6 in dieser Reihenfolge in einen AVL-Baum ein, Skizzieren Sie den Baum nach jeder Einfügeoperation.

c) Was ist ein **2-3-4 Baum**? Nennen Sie die Eigenschaften. Wie wird in einem 2-3-4 Baum garantiert, dass alle Wege von der Wurzel zu einem Blatt stets gleich lang sind?

d) Fügen Sie in der angegebenen Reihenfolge folgende Schlüssel in einen 2-3-4 Baum ein: 4, 6, 2, 5, 3, 7, 8, 1.
Skizzieren Sie den Baum nach jeder Einfügeoperation.

e) Nennen Sie die Eigenschaften eines Rot-Schwarz Baumes. Zeichnen sie zum obigen 2-3-4 Baum den äquivalenten Rot-Schwarz Baum.

Aufgabe 3 (2+1+2+5+1 = 11 Punkte)

a) Was ist ein Graph? Geben Sie für einen gerichteten Graphen die formale Definition und ein Beispiel mit 5 Knoten (Bild und formal) an.

In Folgenden wird ein ungerichteter und ungewichteter Graph, der durch eine Adjazenzmatrix repräsentiert wird, betrachtet.

b) Schreiben Sie eine Klasse *Graph* mit den benötigten Membervariablen (Attributen) und einem Konstruktor für einen leeren Graphen.

c) Beschreiben Sie kurz, wie das Einfügen eines neuen Knotens in ihrer Klasse *Graph* durchgeführt werden müsste. Welche Komplexität hat diese Methode? Begründen Sie kurz.

d) Schreiben Sie eine Methode *int shortestDistance(int node1, int node2),* die die kürzeste Entfernung zwischen 2 Knoten zurück gibt.
Nennen Sie einen weiteren Algorithmus, mit dem man dieses Problem hätte lösen können. (Dieser Algorithmus muss nicht im Detail diskutiert werden. Es reicht, wenn Sie einen Algorithmus aus der Vorlesung benennen, der zur Lösung dieses Problems, eventuell auch in angepasster Form, herangezogen werden kann.)

e) Die Klasse *Graph* soll nun so geändert werden, dass die Knoten im Graphen auch beliebige Objekte repräsentieren können (generische Klasse). Wie müsste man Ihre ursprüngliche Klasse erweitern, damit dies möglich wird?

Aufgabe 4 (2+1 = 3 Punkte)

a) Die Hashfunktion *h(e)* sei gegeben durch
h(e) = (Position des 2. Buchstabens von e im Alphabet) mod Tabellenlänge,
also z. B. *h("e**r**klärung") = 18 mod Tabellenlänge*, da *r* an Position 18 steht.

Fügen Sie die Worte {*Uni, Klausur, Student, Ferien, Datenstrukturen, Aufgabe*} in eine Hashtabelle der Größe 7 ein und benutzen Sie zur Kollisionsbehandlung lineares Sondieren. Geben Sie die Anzahl der auftretenden Kollisionen an.

a=1	b=2	c=3	d=4	e=5	f=6	g=7	h=8	i=9
j=10	k=11	l=12	m=13	n=14	o=15	p=16	q=17	r=18
s=19	t=20	u=21	v=22	w=23	x=24	y=25	z=26	

b) Geben Sie eine beliebige Hashfunktion an, die die Wörter {*Computer, Magdeburg, Java, Segdewick*} in eine Tabelle der Größe 4 einfügt, ohne dass es zu Kollisionen kommt. Geben Sie die dazugehörige Hashtabelle an.

Aufgabe 5 (3+3 = 6 Punkte)

a) Geben Sie in Stichpunkten die Ideen und Eigenschaften der Dynamischen Programmierung, des Backtracking und von Greedy Algorithmen an.

b) Nachdem John seine letzte Klausur geschafft und seine Eltern besucht hat, ist er Montagabend um 21.30 Uhr zu Hause und beginnt seine freie Zeit ab Montag 22:00 zu verplanen. Am darauf folgenden Sonntag möchte er mit der Vorbereitung auf das nächste Semester beginnen.

Diese Woche kann er mit den folgenden Aktivitäten verbringen:

Montag	22:00 - Dienstag	11:00	→ ausschlafen
Dienstag	06:00 - Samstag	22:00	→ Kurztrip nach Amsterdam
Dienstag	11:00 - Dienstag	21:00	→ Tennisturnier
Dienstag	19:00 - Dienstag	22:00	→ Rock Konzert
Mittwoch	15:00 - Donnerstag	15:00	→ LAN Party
Donnerstag	10:00 - Donnerstag	16:00	→ Schwimmbad
Freitag	11:00 - Freitag	21:00	→ Freizeitpark
Samstag	12:00 - Samstag	14:00	→ Shopping Tour
Samstag	20:30 - Samstag	20:45	→ Duschen
Samstag	21:00 - Sonntag	06:00	→ Videospiele Abend
Samstag	21:01 - Samstag	23:59	→ All you can eat Wettbewerb in der Disco

John möchte an möglichst vielen Aktivitäten teilnehmen. Geben Sie eine Strategie an, die nach dem Greedy Prinzip möglichst viele der Aktivitäten für die Woche auswählt. Welche Aktivitäten erledigt John nach dieser Strategie?

Hinweis: Alle nicht aufgeführten Aktivitäten, die John erledigen muss (wie z.B. Zähne putzen) brauchen nicht berücksichtigt zu werden.

Wir wünschen Ihnen viel Erfolg!