Technische Universität Wien
Institut für Computergraphik und Algorithmen
Arbeitsbereich für Algorithmen und Datenstrukturen

186.813 Algorithmen und Datenstrukturen 1 VU 6.0

Nachtragstest SS 2012

12. Oktober 2012

Machen Sie die folgenden Angaben bitte in deutlicher Blockschrift:

Nachname:		Vorname:	
Matrikelnummer:		Studienkennzahl:	
Anzahl abgegebener Zusatzblätter:		Unterschrift:	

Legen Sie während der Prüfung Ihren Ausweis für Studierende vor sich auf das Pult.

Sie können die Lösungen entweder direkt auf die Angabeblätter oder auf Zusatzblätter schreiben, die Sie von der Aufsicht erhalten. Es ist nicht zulässig, eventuell mitgebrachtes eigenes Papier zu verwenden. Benutzen Sie bitte dokumentenechte Schreibgeräte (keine Bleistifte)!

Die Verwendung von Taschenrechner, Mobiltelefonen, PDAs, Digitalkameras, Skripten, Büchern, Mitschriften, Ausarbeitungen oder vergleichbaren Hilfsmitteln ist unzulässig.

	A1:	A2:	A3:	Summe:
Erreichbare Punkte:	16	16	18	50
Erreichte Punkte:				

Viel Erfolg!

Aufgabe 1.A: Suchverfahren **(16 Punkte)**

Gegeben ist die Zahlenfolge

$$\langle 32, 5, 16, 11, 13, 15, 14, 12 \rangle.$$

a) Fügen Sie alle Schlüssel der Folge nacheinander in einen anfangs leeren AVL-Baum ein. Zeichnen Sie den resultierenden AVL-Baum und alle Zwischenschritte, bei denen eine Reorganisation des Baumes notwendig ist. Geben Sie auch jeweils die Reorganisationsmaßnahme an.

b) Geben Sie den AVL-Baum an, der durch Löschen des Schlüssels 13 aus dem Baum von Punkt (a) entsteht.

Aufgabe 2.A: Optimierung **(16 Punkte)**

a) (16 Punkte) Gegeben sei folgende Instanz des 0/1-Rucksackproblems mit Kapazität $K = 6$ und folgenden Gegenständen:

Gegenstand i	Gewicht w_i	Wert c_i
1	2	2
2	3	2
3	1	1
4	3	4
5	2	3
6	3	5
7	1	2

$i \backslash j$	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1							
2							
3							
4							
5							
6							
7							

Diese Instanz soll mittels *Dynamischer Programmierung* über die möglichen Gesamtgewichte gelöst werden. Dazu wird die oben stehende 8×7-Matrix $\mathbf{m}$ verwendet, wobei der Eintrag im Feld $m_{i,j}$ angibt, welcher Wert mit den ersten i Gegenständen erreicht werden kann, wenn das Gesamtgewicht der gewählten Gegenstände kleiner oder gleich j ist.

Die Felder der Matrix können wie folgt berechnet werden:

$$m_{0,j} = 0 \qquad \text{für } j = 0, \ldots, 6$$

$$m_{i,j} = \begin{cases} m_{i-1,j} & \text{falls } w_i > j, \\ \max\{m_{i-1,j-w_i} + c_i,\ m_{i-1,j}\} & \text{sonst.} \end{cases} \qquad \text{für } \begin{cases} i = 1, \ldots, 7 \\ j = 0, \ldots, 6 \end{cases}$$

- (8 Punkte) Lösen Sie die gegebene Instanz, indem Sie die obenstehende Matrix vervollständigen.
- (3 Punkte) Geben Sie die optimale Lösung der Probleminstanz an und markieren Sie in der Matrix jene Zellen, die beim Rekonstruieren der Lösung betrachtet werden.
- (3 Punkte) Ist es möglich, dass mehrere optimale Lösungen existieren, die denselben Gesamtwert und dasselbe Gesamtgewicht haben? Begründen Sie Ihre Antwort.
- (2 Punkte) Geben Sie die Laufzeit für das Befüllen der Matrix im Worst- und im Best-Case in Θ-Notation, in Abhängigkeit der Anzahl der Elemente n und der Kapazität K, an.

Aufgabe 3.A: Hashverfahren mit Double Hashing **(18 Punkte)**

Gegeben ist eine Hashtabelle in Form eines Feldes *feld* mit der festgelegten Größe m. Jedes Element $feld[j]$, $j = 0, \ldots, m-1$, besteht aus folgenden Komponenten:

- $feld[j].key$ enthält den Schlüssel des Datensatzes;
- $feld[j].zustand$ enthält einen der folgenden Werte:
  - *besetzt*: $feld[j]$ enthält einen gültigen Schlüssel;
  - *frei*: $feld[j]$ ist frei und war nie besetzt;
  - *wiederfrei*: $feld[j]$ war schon besetzt, ist aber wieder frei.

Schreiben Sie eine Funktion $Suche(key)$ in Pseudocode, die den Schlüssel key in der Hashtabelle sucht. Falls der gesuchte Schlüssel in der Tabelle vorhanden ist, soll die Funktion seinen Index zurückliefern.

Folgende Punkte sind zu beachten:

- Zur Behandlung von Kollisionen wird Double Hashing ohne Verbesserung nach Brent verwendet.
- Die Hashfunktionen $h_1(k)$ und $h_2(k)$ zur Verwendung in Double Hashing können Sie als gegeben annehmen und im Code verwenden.
- Auf die Tabellengröße m und das Feld *feld* kann global zugegriffen werden.

Technische Universität Wien
Institut für Computergraphik und Algorithmen
Arbeitsbereich für Algorithmen und Datenstrukturen

186.813 Algorithmen und Datenstrukturen 1 VU 6.0

Nachtragstest SS 2012

12. Oktober 2012

Machen Sie die folgenden Angaben bitte in deutlicher Blockschrift:

Nachname:		Vorname:	
Matrikelnummer:		Studienkennzahl:	
Anzahl abgegebener Zusatzblätter:		Unterschrift:	

Legen Sie während der Prüfung Ihren Ausweis für Studierende vor sich auf das Pult.

Sie können die Lösungen entweder direkt auf die Angabeblätter oder auf Zusatzblätter schreiben, die Sie von der Aufsicht erhalten. Es ist nicht zulässig, eventuell mitgebrachtes eigenes Papier zu verwenden. Benutzen Sie bitte dokumentenechte Schreibgeräte (keine Bleistifte)!

Die Verwendung von Taschenrechner, Mobiltelefonen, PDAs, Digitalkameras, Skripten, Büchern, Mitschriften, Ausarbeitungen oder vergleichbaren Hilfsmitteln ist unzulässig.

	B1:	B2:	B3:	Summe:
Erreichbare Punkte:	16	18	16	50
Erreichte Punkte:				

Viel Glück!

Aufgabe 1.B: Suchverfahren **(16 Punkte)**

Gegeben ist die Zahlenfolge

$$\langle 8, 20, 40, 14, 16, 19, 17, 15 \rangle.$$

a) Fügen Sie alle Schlüssel der Folge nacheinander in einen anfangs leeren AVL-Baum ein. Zeichnen Sie den resultierenden AVL-Baum und alle Zwischenschritte, bei denen eine Reorganisation des Baumes notwendig ist. Geben Sie auch jeweils die Reorganisationsmaßnahme an.

b) Geben Sie den AVL-Baum an, der durch Löschen des Schlüssels 16 aus dem Baum von Punkt (a) entsteht.

Aufgabe 2.B: Hashverfahren mit Double Hashing **(18 Punkte)**

Gegeben ist eine Hashtabelle in Form eines Feldes *feld* mit der festgelegten Größe m. Jedes Element $feld[j]$, $j = 0, \ldots, m-1$, besteht aus folgenden Komponenten:

- $feld[j].key$ enthält den Schlüssel des Datensatzes;
- $feld[j].zustand$ enthält einen der folgenden Werte:
  - *besetzt*: $feld[j]$ enthält einen gültigen Schlüssel;
  - *frei*: $feld[j]$ ist frei und war nie besetzt;
  - *wiederfrei*: $feld[j]$ war schon besetzt, ist aber wieder frei.

Schreiben Sie eine Funktion $Suche(key)$ in Pseudocode, die den Schlüssel key in der Hashtabelle sucht. Falls der gesuchte Schlüssel in der Tabelle vorhanden ist, soll die Funktion seinen Index zurückliefern.

Folgende Punkte sind zu beachten:

- Zur Behandlung von Kollisionen wird Double Hashing ohne Verbesserung nach Brent verwendet.
- Die Hashfunktionen $h_1(k)$ und $h_2(k)$ zur Verwendung in Double Hashing können Sie als gegeben annehmen und im Code verwenden.
- Auf die Tabellengröße m und das Feld *feld* kann global zugegriffen werden.

Aufgabe 3.B: Optimierung **(16 Punkte)**

a) (16 Punkte) Gegeben sei folgende Instanz des 0/1-Rucksackproblems mit Kapazität $K = 7$ und folgenden Gegenständen:

Gegenstand i	Gewicht w_i	Wert c_i
1	2	1
2	1	1
3	3	2
4	3	4
5	2	4
6	1	2

$i \backslash j$	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1								
2								
3								
4								
5								
6								

Diese Instanz soll mittels *Dynamischer Programmierung* über die möglichen Gesamtgewichte gelöst werden. Dazu wird die oben stehende 7×8-Matrix $\mathbf{m}$ verwendet, wobei der Eintrag im Feld $m_{i,j}$ angibt, welcher Wert mit den ersten i Gegenständen erreicht werden kann, wenn das Gesamtgewicht der gewählten Gegenstände kleiner oder gleich j ist.

Die Felder der Matrix können wie folgt berechnet werden:

$$m_{0,j} = 0 \qquad \text{für } j = 0, \ldots, 7$$

$$m_{i,j} = \begin{cases} m_{i-1,j} & \text{falls } w_i > j, \\ \max\{m_{i-1,j-w_i} + c_i,\ m_{i-1,j}\} & \text{sonst.} \end{cases} \qquad \text{für } \begin{cases} i = 1, \ldots, 6 \\ j = 0, \ldots, 7 \end{cases}$$

- (7 Punkte) Lösen Sie die gegebene Instanz, indem Sie die obenstehende Matrix vervollständigen.
- (4 Punkte) Geben Sie die optimale Lösung der Probleminstanz an und markieren Sie in der Matrix jene Zellen, die beim Rekonstruieren der Lösung betrachtet werden.

b) (2 Punkte) Geben Sie die Laufzeit für das Befüllen der Matrix im Worst- und im Best-Case in Θ-Notation, in Abhängigkeit der Anzahl der Elemente n und der Kapazität K, an.

c) (3 Punkte) Ist es möglich, dass mehrere optimale Lösungen existieren, die denselben Gesamtwert und dasselbe Gesamtgewicht haben? Begründen Sie Ihre Antwort.