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

186.813 Algorithmen und Datenstrukturen 1 VU 6.0

Übungsblatt 4

für die Übung am Montag den 8. bzw. Dienstag den 9. Mai 2017.

Geben Sie bis **spätestens Sonntag, 07.05.2017, 12:00 Uhr** über TUWEL an, welche Beispiele Sie bearbeitet und gelöst haben. Gehen Sie dabei folgendermaßen vor:

- TUWEL (`https://tuwel.tuwien.ac.at`)
  Kurs *186.813 Algorithmen und Datenstrukturen 1 (VU 6.0)*
- Übungsblätter
- Bearbeitete Beispiele ankreuzen **und** abgeben
  - Link *Übungsblatt 4 - Details & Bewertung*
    Button *Abgabe bearbeiten*
    Bearbeitete Beispiele anhaken und *Änderungen speichern.*
  - Link *Hochladen Lösungen Übungsblatt 4*
    Button *Abgabe hinzufügen*
    PDF-Datei mit Lösungen hochladen und *Änderungen sichern.*

Bitte beachten Sie:

- Sie können **vor** der Deadline beliebig oft ihre Auswahl an Beispielen und das zugehörige Lösungs-PDF verändern, aber **nach** der Deadline gibt es **keine** Veränderung ihrer angekreuzten Beispiele **und** der PDF-Datei!
- Sie können Ihre Lösungen entweder direkt in einem Textverarbeitungsprogramm erstellen, oder aber auch gut leserliche Scans bzw. Fotos von handschriftlichen Ausarbeitungen einreichen.
- Bitte geben Sie Ihren Namen, Matrikelnummer und E-Mail-Adresse in den Ausarbeitungen an.
- Wenn Sie zur Präsentation Ihrer Lösung eines von Ihnen angekreuzten Beispiels ausgewählt werden und dieses aber nicht bearbeitet haben oder dieses Beispiel nicht in der PDF-Datei vorhanden ist, verlieren Sie **alle** Punkte dieser Übungseinheit!

  (Zusätzlich werden stichprobenartig die abgegebenen PDF-Dateien auf Übereinstimmung mit der entsprechenden Kreuzerlliste überprüft.)

Aufgabe 25 Betrachten Sie eine Variante von Quicksort welche immer den Median des ersten, mittleren und letzten Elementes des aktuellen Bereiches als Pivotelement auswählt. Geben Sie eine Familie von Beispielsequenzen an, die zeigt, dass die Worst-Case-Laufzeit von dieser Variante von Quicksort auch nicht besser als quadratisch sein kann. Die Beispielsequenzen sollten dabei keine Duplikate enthalten.

Aufgabe 26 Gegeben ist eine Hashtabelle mit Tabellengröße $m = 13$ und die Hashfunktion $h_1(k) = k \bmod 13$. In die Tabelle wurden die Werte $\langle 93, 97, 109, 111 \rangle$ bereits eingefügt. Gehen Sie davon aus, dass zur Kollisionsbehandlung

(a) Lineares Sondieren,

(b) Quadratisches Sondieren mit den Konstanten $c_1 = 1$ und $c_2 = 2$,

(c) Double Hashing mit der zusätzlichen Hashfunktion $h_2(k) = k \bmod 7 + 1$

verwendet wurde und fügen Sie jeweils mit derselben Methode die Werte $\langle 123, 124, 136 \rangle$ in dieser Reihenfolge ein.

	0	1	2	3	4	5	6	7	8	9	10	11	12
Lineares Sondieren			93			109	97	111					
Quadratisches Sondieren			93			109	97	111					
Double Hashing			93			109	97	111					

Aufgabe 27 Gegeben sei die Zahlenfolge

$$\langle 27, 81, 19, 70, 35, 4, 75, 127 \rangle$$

und die beiden Hashfunktionen

$$\begin{aligned} h_1(k) &= k \bmod 11 \\ h_2(k) &= (k \bmod 5) + 1 \end{aligned}$$

Fügen Sie die Elemente der Folge mit Hilfe von *Double Hashing mit der Verbesserung nach Brent* in eine anfangs leere Hashtabelle der Größe $m = 11$ ein. Es muss für jede Zahl erkenntlich sein, wie sie zu ihrem endgültigen Platz in der Hashtabelle gekommen ist.

Aufgabe 28 Eine Hashtabelle soll bis zu 3500 ganze Zahlen aus dem Bereich von -4000 bis 4000 aufnehmen. Der Hashwert wird mit der Divisions-Rest-Methode ermittelt und die Kollisionsbehandlung erfolgt mittels Double Hashing.

(a) Geben Sie für jede der folgenden Zahlen an, ob sie eine gute Wahl für die Tabellengröße m wäre und begründen Sie Ihre Entscheidung:

$$26209,\ 4096,\ 3517,\ 4127,\ 3499$$

(b) Definieren Sie für eine geeignete Wahl der Tabellengröße aus (a) entsprechende Funktionen $h_1(k)$ und $h_2(k)$.

(c) Auf was muss bei der Wahl der Funktionen $h_1(k)$ und $h_2(k)$ besonders geachtet werden?

Aufgabe 29 Beschreiben Sie eine einfache Modifikation eines binären Suchbaums, die es Ihnen erlaubt das k-te Element mit einer durchschnittlichen Laufzeit von $\Theta(\log n)$ zu finden. Das k-te Element ist das Element an der Stelle k $(0 \leq k < n)$ in der Sortierreihenfolge eines binären Suchbaums mit n Elementen. Sie dürfen keine zusätzliche Datenstruktur benutzen, sondern nur zusätzliche Informationen in den Knoten des Baums speichern. Entwickeln Sie eine Lösung in detailliertem Pseudocode, die das k-te Element in Ihrem modifizierten binären Suchbaum zurückgibt und argumentieren Sie, dass der Algorithmus korrekt ist und die geforderte Laufzeitschranke gilt.

Aufgabe 30

(a) Bei einer In-Order-Durchmusterung eines gegebenen binären Baumes ergibt sich folgende Knotenliste:

$$1, 2, 4, 18, 8, 7, 9, 19, 12, 14, 6, 11.$$

Bei einer Pre-Order-Durchmusterung des gleichen Baumes ergibt sich:

$$9, 18, 2, 1, 4, 8, 7, 12, 19, 6, 14, 11.$$

Zeichnen Sie den zugrundeliegenden Baum und tragen Sie die entsprechenden Knotenwerte ein.

(b) Entwickeln Sie einen Algorithmus in detailiertem Pseudocode, welcher zu einer gegebenen In-Order- sowie Pre-Order-Durchmusterung den eindeutigen binären Baum, der beiden Durchmusterungen entspricht, ausgibt und argumentieren Sie, dass der Algorithmus korrekt ist. Geben Sie weiterhin die Laufzeit des Algorithmus in Θ-Notation an.

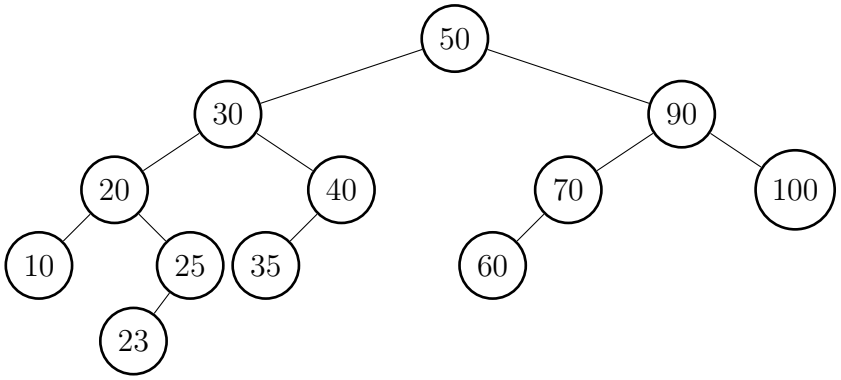
Aufgabe 31

(a) Fügen Sie die Elemente der Folge

$$\langle 30, 80, 90, 20, 60, 50, 85 \rangle$$

in dieser Reihenfolge in einen anfangs leeren AVL-Baum ein. Zeichnen Sie den AVL-Baum jeweils vor und nach jeder Reorganisationsmaßnahme und geben Sie den endgültigen AVL-Baum an.

(b) Gegeben sei folgender AVL-Baum:



Löschen Sie nun den Knoten mit Schlüssel 100 und führen Sie alle notwendigen Operationen durch, um nach der Entfernung dieses Knotens wieder einen gültigen AVL-Baum zu erhalten.

Aufgabe 32

(a) Fügen Sie die Elemente der Folge

$$\langle 10, 30, 60, 80, 110, 140, 200, 400 \rangle$$

in dieser Reihenfolge in einen anfangs leeren B-Baum der Ordnung 3 ein. Zeichnen Sie den B-Baum jeweils vor und nach jeder Reorganisationsmaßnahme und geben Sie den endgültigen B-Baum an. Fügen Sie die Folge auch in einen natürlichen binären Suchbaum ein und vergleichen Sie beide Resultate.

(b) Geben Sie den B-Baum an, der durch Löschen der Schlüssel 110 und 80 (in dieser Reihenfolge) aus dem B-Baum von Punkt (a) entsteht.
