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 9. bzw. Dienstag den 10. Mai 2016.

Geben Sie bis **spätestens Sonntag, 8.5.2016, 23:59 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 Mergesort die das Eingabearray in jedem Schritt in drei anstelle von zwei Teile teilt (gegeben durch folgenden Pseudocode).

> `Mergesort()`:
> **if** $l < r$ **then**
>   **if** $r - l > 2$ **then**
> $m_1 \leftarrow \lfloor (2l + r)/3 \rfloor$
> $m_2 \leftarrow \lfloor (l + 2r)/3 \rfloor$
>   `Mergesort` (A,l,m_1)
>   `Mergesort` (A,$m_1 + 1$,m_2)
>   `Mergesort` (A,$m_2 + 1$,r)
>   `Merge` (A,l,m_1,m_2,r)
>   **else**
>   `Mergesort` (A,l,l)
>   `Mergesort` (A,r,r)
>   `Merge` (A,l,l,r)

Geben Sie die Laufzeit dieser Variante von Mergesort in O-Notation an und beweisen Sie diese mithilfe der "Beweis durch Rekursionsbaum" Methode aus der Vorlesung. Sie können dabei annehmen, dass die Methode `Merge(A,`l`,`m_1`,`m_2`,`r`)`, welche die drei sortierten Listen, gegeben durch (l bis m_1), ($m_1 + 1$ bis m_2), und ($m_2 + 1$ bis r), zu einer sortierten Gesamtliste verschmilzt, eine Laufzeit von $O(r - l)$ hat.

Aufgabe 26 Betrachten Sie eine Variante von Quicksort welche immer das kleinere der (höchstens zwei) mittleren Elemente des aktuellen Bereiches als Pivotelement auswählt. Ändert sich die Worst-Case Laufzeit für diese Variante von Quicksort? Erklären Sie ihre Antwort.

Aufgabe 27 Gegeben ist eine Hashtabelle mit Tabellengröße $m = 13$ und der Hashfunktion $h_1(k) = k \bmod 13$. In die Tabelle wurden die Werte $\langle 2, 6, 18, 20 \rangle$ 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 $\langle 32, 33, 45 \rangle$ in dieser Reihenfolge ein.

	0	1	2	3	4	5	6	7	8	9	10	11	12
Lineares Sondieren			2			18	6	20					
Quadratisches Sondieren			2			18	6	20					
Double Hashing			2			18	6	20					

Aufgabe 28 Gegeben sei die Zahlenfolge

$$\langle 38, 26, 15, 13, 4, 20, 72 \rangle$$

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 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 29 Schreiben Sie einen rekursiven Algorithmus (in Pseudocode), welcher für einen binären Baum alle inneren Knoten mit ihren Gewichten ausgibt. Das Gewicht eines Knotens v sei hierbei die Anzahl aller Knoten im Unterbaum mit der Wurzel v. Blattknoten sollen nicht ausgegeben werden. Verwenden Sie hierfür diesselben Datenstrukturen für einen binären Baum wie sie aus der Vorlesung bekannt sind.

Aufgabe 30 Gegeben sei ein binärer Suchbaum für Werte zwischen 1 und 1000. In diesem Baum wird die Zahl 364 gesucht.

(a) Überprüfen Sie für die Folgen (1) bis (4), ob die Elemente in der angegebenen Reihenfolge Werte von Knoten repräsentieren können, die bei der Suche im binären Suchbaum durchlaufen wurden.

  (1) $\langle 900, 5, 765, 105, 764, 333, 357, 352 \rangle$
  (2) $\langle 898, 56, 292, 740, 212, 555, 342, 344 \rangle$
  (3) $\langle 999, 106, 897, 212, 295, 898, 605 \rangle$
  (4) $\langle 8, 412, 411, 156, 259, 279, 294, 380, 360 \rangle$

(b) Schreiben Sie einen Algorithmus als Pseudocode, der dieses Problem im Allgemeinen löst, d.h. für beliebige Folgen, Obergrenzen, Untergrenzen und gesuchte Zahlen. Sie können von paarweise unterschiedlichen Werten und korrekten Eingaben ausgehen.

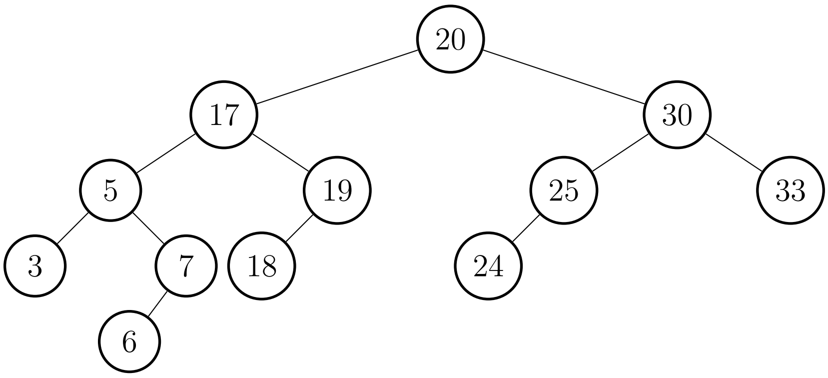
Aufgabe 31

(a) Fügen Sie die Elemente der Folge

$$\langle 20, 50, 70, 10, 40, 30, 60 \rangle$$

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 33 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

$$\langle 1, 3, 6, 8, 11, 14, 20, 40 \rangle$$

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 binären Baum ein und vergleichen Sie beide Resultate.

(b) Geben Sie den B-Baum an, der durch Löschen der Schlüssel 11 und 8 (in dieser Reihenfolge) aus dem Baum von Punkt (a) entsteht.
