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

186.866 Algorithmen und Datenstrukturen VU

Übungsblatt 4

für die Übung am Montag den 14. bzw. Dienstag den 15. Mai 2018.

Geben Sie bis **spätestens Sonntag, 13.5.2017, 13:00 Uhr** über TUWEL an, welche Beispiele Sie bearbeitet und gelöst haben. Gehen Sie dabei folgendermaßen vor:

- TUWEL (`https://tuwel.tuwien.ac.at`)
  Kurs *186.866 Algorithmen und Datenstrukturen (VU 5.5)* im Abschnitt *Übungsblätter*
- Bearbeitete Beispiele ankreuzen **und** abgeben
  - Link *Ankreuzen Übungsblatt 4*
    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.*
  - Wenn Sie das Programmierbeispiel gelöst haben:
    Link *Hochladen Source-Code Übungsblatt 4*
    Button *Abgabe hinzufügen*
    Java-Datei (`E4.java`) 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 abgegebenen Dateien!
- 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 das Programmierbeispiel kreuzen, muss sowohl die theoretische Ausarbeitung im PDF, als auch der Source-Code im Tuwel abgegeben werden.
- Beachten Sie die Richtlinien für das An- und Aberkennen von Beispielen in den Vorbesprechungsfolien. Neben der Überprüfung in der Übungseinheit werden danach stichprobenartig weitere Abgaben auf Spekulation und Plagiate überprüft.

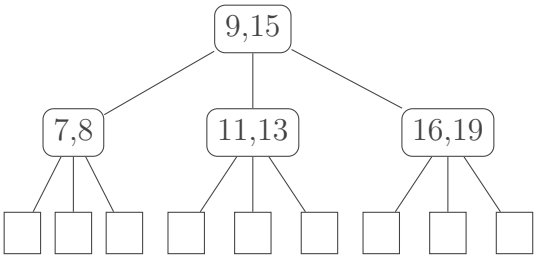
Aufgabe 1

(a) Fügen Sie die Elemente der Folge

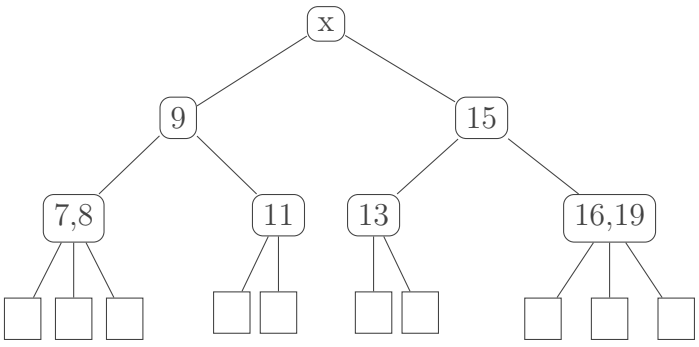
$$\langle 1, 2, 5, 9, 11, 19, 22, 37 \rangle$$

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) Gegeben ist der folgende B-Baum der Ordnung 3:



Durch Einfügen des Elements x entsteht der folgende Baum:



Welchen Wert hat x? Begründen Sie ihre Antwort.

Aufgabe 2 Gegeben seien folgende $n = 7$ geordnete natürliche Zahlen:

$$[14, 5, 16, 19, 20, 1, 9]$$

Fügen Sie diese in der vorgegebenen Ordnung jeweils in folgende Varianten von anfangs leeren Hashtabellen der Größe $m = 9$ ein und stellen Sie die einzelnen Schritte und die finale Belegung dar. Geben Sie weiters jeweils die durchschnittliche Laufzeit einer erfolgreichen Suche eines Elements an.

(a) Verkettung der Überläufer mit $h(k) = k \mod m$.

(b) Quadratisches Sondieren mit $h(k) = (h'(k) + c_1 i + c_2 i^2) \mod m$, $h'(k) = \lfloor m(kA - \lfloor kA \rfloor) \rfloor$, $c_1 = 1/2$, $c_2 = 1/2$ und $A = (\sqrt{5} - 1)/2$.

(c) Double-Hashing mit $h(k) = (h_1(k) + i h_2(k)) \mod m$, $h_1(k) = k \mod m$ und $h_2(k) = 1 + (k \mod 5)$.

Für Sondieren (quadratisch und Double-Hashing) sei $i = 0, 1, \ldots, m - 1$.

Aufgabe 3 Wir wollen nun das Laufzeitverhalten von Hashing mit einerseits Verkettung der Überläufer und andererseits Sondieren theoretisch analysieren. Dabei setzen wir die "simple uniform hashing" Annahme voraus, d.h. jeder Hashwert wird mit der gleichen Wahrscheinlichkeit $1/m$ angenommen. Die Größe der Hashtabelle sei m, die aktuelle Anzahl der Elemente in dieser n und der Belegungsfaktor $\alpha = n/m$.

(a) Zeigen Sie, dass die erwartete Laufzeit einer erfolgreichen Suche in einer Hashtabelle mit Verkettung der Überläufer mit $\Theta(1 + \frac{\alpha}{2})$ geht, unter der Annahme von "simple uniform hashing".

(b) Was besagt die "uniform hashing" Annahme?

(c) Zeigen Sie, dass die erwartete Laufzeit einer erfolgreichen Suche in einer Hashtabelle mit Sondieren durch $\frac{1}{\alpha} \log(\frac{1}{1-\alpha})$ nach oben beschränkt ist, wenn "uniform hashing" angenommen wird.

Hinweise: Nehmen Sie eine Reihenfolge, in der n Elemente in die Hashtabelle eingetragen wurden, an. Betrachten Sie bei (a) die erwartete Anzahl von Elementen, die vor einem Element in dessen Liste stehen. Bei (c) wissen Sie, dass die erwartete Anzahl der erfolglosen Sondierungen höchstens $1/(1 - \alpha)$ ist.

Aufgabe 4 Sie haben eine Hashfunktion h gegeben, die der "simple uniform hashing" Annahme genügt. Sie implementieren damit eine Hashtabelle ohne Kollisionsbehandlung, stattdessen setzen Sie einfach die Größe der Hashtabelle m sehr hoch und nehmen an, dass das Verhältnis n/m klein ist.

(a) Geben Sie eine Herleitung für eine Näherung der Wahrscheinlichkeit $p(m, n)$, dass eine Kollision auftritt, abhängig von m und der Anzahl n der einzufügenden Elemente. *Hinweis:* $1 + x \approx e^x$ für kleine $|x|$ und generell $1 + x \leq e^x$.

(b) Leiten Sie weiters daraus ab, wie viele n abhängig von m zumindest in die Hashtabelle eingefügt werden müssen, um eine Wahrscheinlichkeit $\geq 1/2$ einer Kollision zu haben.

(c) Basierend auf dem Resultat von Punkt (b), welchen Wert n erhalten Sie für $m = 100000$? Welchen für $m = 365$?

Aufgabe 5

(a) Finden Sie heraus, welcher Sortieralgorithmus für die folgenden Funktionen in den folgenden Sprachen verwendet wird:

- `sort()` in Python.
- `sort()` in PHP.

(b) Die Funktion `sort()` der $C++$ Standard Library wird üblicherweise mit einer Kombination mehrere Sortieralgorithmen namens Introsort implementiert. Finden Sie heraus, welche Algorithmen bei Introsort kombiniert werden. Erklären Sie kurz wie diese Kombination funktioniert und welche Vorteile sie bietet.

(c) Ist die Insertionsort Implementierung aus den Vorlesungfolien stabil? Falls ja, argumentieren Sie kurz warum, falls nein, konstruieren Sie ein Gegenbeispiel.

Aufgabe 6 (Programmieraufgaben)

Ihre Aufgabe ist die **Insert-Operation für AVL-Bäume** zu implementieren.

Die **detaillierte Angabe** zu dieser Programmieraufgabe finden Sie **im TUWEL** (PDF-Dokument „Programmieraufgabe 4“). Dieses Dokument enthält auch eine Beschreibung der zu implementierenden Funktionen und die Theoriefragen dieser Aufgabe.

Sie bekommen das benötigte Framework („AlgoDat_E4.zip“) ebenfalls im TUWEL zur Verfügung gestellt. Implementieren Sie darin in der Datei „exercise/E4.java“ die geforderten Methoden.

Wichtig: Damit das Beispiel als positiv absolviert gilt, müssen **alle** geforderten Methoden korrekt implementiert und die Theoriefragen in der PDF Abgabe ausführlich beantwortet werden!
