TU Wien
Institut für Logic and Computation
Forschungsbereich Algorithms and Complexity

186.866 Algorithmen und Datenstrukturen VU

Übungsblatt 4

PDF erstellt am: 4. Mai 2023

Deadline für dieses Übungsblatt ist **Montag, 08.05.2023, 20:00 Uhr**. Um Aufgaben für diese Übung anerkannt zu bekommen, gehen Sie folgendermaßen vor:

1. Öffnen Sie den TUWEL-Kurs der Lehrveranstaltung *186.866 Algorithmen und Datenstrukturen (VU 5.5)* und navigieren Sie zum Abschnitt *Übungsblätter*.
2. Teilen Sie uns mit, welche Aufgaben Sie gelöst haben **und** welche gelösten Aufgaben Sie gegebenenfalls in der Übungseinheit präsentieren können. Gehen Sie dabei folgendermaßen vor:
   - Laden Sie Ihre Lösungen in einem einzigen PDF-Dokument in TUWEL hoch.
     Link *Hochladen Lösungen Übungsblatt 4*
     Button *Abgabe hinzufügen*
     PDF-Datei mit Lösungen hochladen und *Änderungen sichern*.
   - Kreuzen Sie an, welche Aufgaben Sie gegebenenfalls in der Übung präsentieren können. Die Lösungen der angekreuzten Aufgaben müssen im hochgeladenen PDF enthalten sein.
     Link *Ankreuzen Übungsblatt 4*
     Button *Abgabe bearbeiten*
     Bearbeitete Aufgaben anhaken und *Änderungen speichern*.

Bitte beachten Sie:

- Bis zur Deadline können Sie sowohl Ihr hochgeladenes PDF, als auch Ihre angekreuzten Aufgaben beliebig oft verändern. Nach der Deadline ist keine Veränderung mehr möglich. Es werden ausnahmslos keine Nachabgabeversuche (z.B. per E-Mail) akzeptiert.
- Sie können Ihre Lösungen entweder direkt in einem Textverarbeitungsprogramm erstellen, oder aber auch gut leserliche Scans bzw. Fotos von handschriftlichen Ausarbeitungen hochladen (beachten Sie die maximale Dateigröße).
- Beachten Sie die Richtlinien für das An- und Aberkennen von Aufgaben (Details finden Sie in den Folien der Vorbesprechung).

Aufgabe 1. Lösen Sie die folgenden Unteraufgaben zu **B-Bäumen**.

(a) Fügen Sie die Elemente der Folge

$$[11, 14, 16, 33, 42, 44, 71, 89, 10]$$

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.

(b) Geben Sie den B-Baum an, der durch Löschen der Schlüssel 16, 44 und 71 (in dieser Reihenfolge) aus dem in Unteraufgabe (a) erhaltenen Baum entsteht. Stellen Sie den B-Baum nach jedem Zwischenschritt dar und führen Sie gegebenenfalls geeignete Reorganisationsmaßnahmen durch.

Anmerkung: Die Angabe von leeren Blättern ist nicht notwendig.

Aufgabe 2. Gegeben ist eine leere Hashtabelle der Größe $m = 11$. Führen Sie folgende Schritte in der vorgegebenen Reihenfolge aus:

1. Füge nacheinander ein: 3, 17, 11, 25, 2, 14
2. Lösche 25
3. Suche 14
4. Füge 36 ein

Verwenden Sie dazu jeweils eine der folgende Varianten zur Kollisionsbehandlung, und beginnen Sie jeweils mit einer leeren Hashtabelle. Stellen Sie die einzelnen Schritte und die finale Belegung dar.

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

(b) Quadratisches Sondieren mit $h(k, i) = (h'(k) + \frac{1}{2}i + \frac{1}{2}i^2) \bmod m$ und $h'(k) = k \bmod m$.

(c) Double-Hashing mit $h(k, i) = (h_1(k) + ih_2(k)) \bmod m$, $h_1(k) = k \bmod m$ und $h_2(k) = (k \bmod 5) + 1$.

Hinweis: Für das Sondieren (quadratisch und Double-Hashing) gilt $i = 0, 1, \ldots, m - 1$.

Aufgabe 3. Gegeben ist folgende Hashtabelle.

0	1	2	3	4	5	6	7	8	9	10
	127	35		70	27	81		19	4	75

Wir verwenden Double Hashing mit $h_1(k) = k \bmod 11$ und $h_2(k) = (k \bmod 5) + 1$.

(a) Fügen Sie 512 in die obige Tabelle ein. Verwenden Sie dazu das verbesserte Einfügen nach Brent. Geben Sie alle Zwischenschritte an.

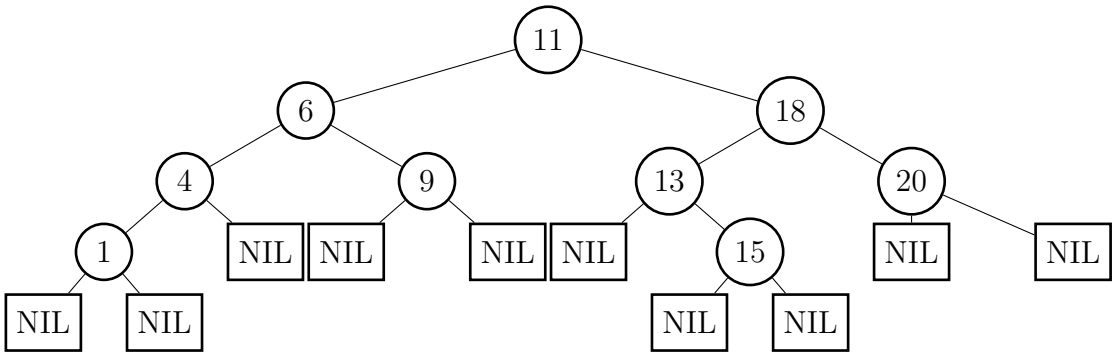
(b) Bestimmen Sie jeweils die durchschnittliche Anzahl an Schritten für eine erfolgreiche Suche in der obigen Tabelle bevor **und** nachdem 512 eingefügt wurde.

Aufgabe 4. Untersuchen Sie das Laufzeitverhalten von Hashing mit Verkettung der Überläufer theoretisch. Setzen Sie dabei die „simple uniform hashing“ Annahme voraus, d.h. jeder Hashwert wird mit der gleichen Wahrscheinlichkeit $1/m$ angenommen. Die Größe einer Hashtabelle sei m, die aktuelle Anzahl der Elemente in dieser n und der Belegungsfaktor $\alpha = n/m$.

Zeigen Sie, dass unter der Annahme von „simple uniform hashing“ die **erwartete** Laufzeit einer erfolgreichen Suche in einer Hashtabelle mit Verkettung der Überläufer in $\Theta(1 + \alpha)$ liegt.

Aufgabe 5. In folgenden Unteraufgaben geht es um **Rot-Schwarz-Bäume**.

(a) Beschreiben Sie sämtliche zulässigen Rot-Schwarz-Färbungen der Knoten des folgenden Suchbaums T, sodass das Ergebnis die Eigenschaften eines Rot-Schwarz-Baums erfüllt:

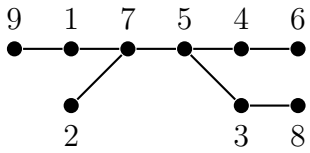


Hinweis: Gehen Sie systematisch vor und überlegen Sie zunächst, welche Knoten stets die Farbe Schwarz tragen müssen.

(b) In einem Wurzelbaum T mit Wurzel $\text{root}(T)$ seien zwei gerichtete Pfade $p_1 : \text{root}(T) \to v_2 \to v_3 \to \ldots \to v_m$ und $p_2 : \text{root}(T) \to w_2 \to w_3 \to \ldots \to w_n$ gegeben, wobei v_m und w_n Blätter des Typs NIL sind und $m \leq n$. Wie groß muss n im Vergleich zu m *mindestens* sein, sodass die Nichtexistenz einer Rot-Schwarz-Färbung von T automatisch gefolgert werden kann? Denken Sie darüber nach, welchen Vorteil in Hinblick auf Suchbäume dieses Phänomen für Rot-Schwarz-Bäume liefert.

Aufgabe 6. Betrachten Sie einen beliebigen freien Baum $T = (V, E)$ (d.h. einen ungerichteten, azyklischen zusammenhängenden Graphen ohne dezidierter Wurzel), wobei $V = \{1, \ldots, n\}$ mit $n \geq 2$. Ein Blatt eines solchen Baums ist ein Knoten, der genau einen Nachbarn besitzt.

(a) Betrachten Sie das folgende iterative Verfahren ENCODE, welches ausgehend von T in ein zunächst leeres Array s der Größe $n-2$ schreibt: *Im Schritt $i \in \{1, \ldots, n-2\}$ wird das kleinste Blatt v aus dem Baum entfernt und der i-te Eintrag von s auf den Wert w gesetzt, wobei w der einzige Nachbar von v ist.* Wenden Sie ENCODE auf folgenden Baum mit $V = \{1, \ldots, 9\}$ an:



(b) Inspizieren Sie Ihr Ergebnis $s = \text{ENCODE}(T)$ aus Unteraufgabe (a): Welche Werte von V treten nicht auf? Warum?

(c) Finden Sie einen freien Baum $T' = (V, E)$ mit $V = \{1, \ldots, 8\}$, für den ENCODE $s = [4, 8, 1, 8, 3, 8]$ liefert.

(d) **Optional, für Interessierte:** Für Folgendes dürfen Sie annehmen, dass es zu ENCODE ein umkehrendes Verfahren DECODE gibt, das aus jedem beliebigen Array $s \in \{1, \ldots, n\}^{n-2}$ einen freien Baum $T = (V, E)$ mit $V = \{1, \ldots, n\}$ generiert, sodass $\text{ENCODE}(T) = s$ gilt. Mit anderen Worten kann jeder freie Baum T durch Ausführung von $\text{DECODE}(\text{ENCODE}(T))$ und jedes Array s durch $\text{ENCODE}(\text{DECODE}(s))$ rückgewonnen werden. Mit wievielen Bits kommen Sie aus, um einen beliebigen freien Baum T bestehend aus 2^k ($k \in \mathbb{N}$) Knoten zu beschreiben?
