Prof. Dr. Ir. Joost-Pieter Katoen

Christian Dehnert, Friedrich Gretz, Benjamin Kaminski, Thomas Ströder

Allgemeine Hinweise:

- Die **Hausaufgaben** sollen in Gruppen von je **2-3 Studierenden** aus der **gleichen Kleingruppenübung (Tutorium)** bearbeitet werden. **Namen und Matrikelnummern** der Studierenden sind auf jedes Blatt der Abgabe zu schreiben. **Heften bzw. tackern Sie die Blätter!**
- Die **Nummer der Übungsgruppe** muss **links oben** auf das **erste Blatt** der Abgabe geschrieben werden. Notieren Sie die Gruppennummer gut sichtbar, damit wir besser sortieren können.
- Die Lösungen müssen **bis Mittwoch, den 17.6.2015 um 12:15 Uhr** in den entsprechenden Übungskasten eingeworfen werden. Sie finden die Kästen am Eingang Halifaxstr. des Informatikzentrums (Ahornstr. 55). Alternativ können Sie die Lösungen auch vor der Abgabefrist direkt bei Ihrer Tutorin/Ihrem Tutor abgeben.
- In Aufgaben, bei denen Sie Algorithmen implementieren sollen, dürfen Sie Ihre Lösung als Pseudo-Code abgeben. Abgaben in verbreiteten imperativen Sprachen wie Java oder C++ sind ebenfalls erlaubt.

Tutoraufgabe 1 (Güte von Hashfunktionen):

Untersuchen Sie, inwiefern sich die folgenden Funktionen für die Verwendung als Hashfunktion eignen. Begründen Sie Ihre Antwort.

- $f : \{0, 1, 2, \ldots, 100\} \to \{0, 1, 2, \ldots, 10\},\ x \mapsto \left\lfloor \frac{x}{10} \right\rfloor$
- $g : \mathbb{N} \to \mathbb{Z}/100\mathbb{Z},\ x \mapsto 2^x \bmod 100$
- $h : \{0, 1, 2, \ldots, 100\} \to \{0, 1, 2, \ldots, 10\},\ x \mapsto x \bmod 10$
- $i : \mathbb{N} \to \{0, 1, 2, \ldots, 50\},\ x \mapsto \bmod 51$

Tutoraufgabe 2 (Countingsort):

a) Das folgendes Array ist mit Countingsort zu sortieren. Geben Sie das Histogramm- und das Positionsarray vor dem ersten Einfügen ins Ausgabearray an, sowie das Positions- und Ausgabearray nach jedem Einfügeschritt.

4	3	0	1	4	2	3	7	3

b) Der in der Vorlesung vorgestellte Algorithmus Countingsort fügt die Elemente des Eingabearrays von hinten nach vorne in das Ausgabearray ein. Welche Nachteile ergäben sich, wenn man das Eingabearray stattdessen von vorne nach hinten durchlaufen würde?

Tutoraufgabe 3 (Hashing):

a) Gegeben sei eine Hash-Table der Größe m und eine beliebige Hash-Funktion $h : U \to \{0, \ldots, m-1\}$. Die Menge U habe nun mindestens $n \cdot m$ Elemente, also $|U| \geq n \cdot m$. Zeigen Sie, dass U eine Teilmenge U_0 der Größe n besitzt ($|U_0| = n$), so dass

$$h(x_1) = h(x_2) \quad \text{für alle} \quad x_1, x_2 \in U_0$$

Was haben Sie damit für die Worst-Case-Laufzeit der Suche mittels Hashing mit Verkettung bewiesen?

b) Gegeben sei nun eine Hash-Table mit einer initialen Größe von 1000 und eine Hash-Funktion, die ein gleichverteiltes Hashing gewährleistet. Nach wie vielen Einfügungen müssen sie a-priori mit einer Kollisionswahrscheinlichkeit von mehr als 80% rechnen?

Um die Anzahl von Kollisionen beim Hashing gering zu halten, kann man die Größe der Hash-Table nach einer gewissen Anzahl von Einfügungen erhöhen. Nach welcher Anzahl k von Einfügeoperationen muss die Tabelle das erste Mal vergrößert werden, wenn bei den vorhergehenden $k-1$ Einfügeoperationen keine Kollision auftrat und die Wahrscheinlichkeit für eine Kollision bei der k-ten Einfügung weniger als 20% betragen soll? Begründen Sie ihre Antwort.

Tutoraufgabe 4 (Hashing):

Gegeben sei eine Hash-Table der Größe 23 und die folgenden beiden Hash-Funktionen über der Universalmenge $U = \{0, \ldots, 499\}$:

- $h_1(x) =$ Quersumme von x
- $h_2(x) = x \mod 23$

a) Fügen Sie die Werte $12, 99, 111, 76, 23, 30$ sowohl mit h_1 als auch h_2 mittels

(i) Hashing mit Verkettung

(ii) Hashing mit linearem Sondieren

in jeweils eine Tabelle ein (es sind also 4 Tabellen zu erstellen). Geben Sie die nicht-leeren Teile der Tabellen nach jedem Einfügeschritt an.

b) Löschen Sie nacheinander die Werte 111, 12 und 76 aus allen Tabellen aus Aufgabe **a)**. Geben Sie die nicht-leeren Teile der Tabellen nach jedem Löschschritt an. Erläutern Sie, welches Vorgehen dafür jeweils nötig ist.

c) Suchen Sie in den Tabellen, die aus Teilaufgabe **b)** resultierten, nach dem Wert 30. Erläutern Sie, welches Vorgehen dafür jeweils nötig ist.

Aufgabe 5 (Güte von Hashfunktionen): (2 + 2 + 2 + 2 Punkte)

Untersuchen Sie, inwiefern sich die folgenden Funktionen für die Verwendung als Hashfunktion eignen. Begründen Sie Ihre Antwort.

- $f : \{1, 2, 3, \ldots, 100\} \to \{0, 1, 2, \ldots, 10\},\ x \mapsto \left\lfloor \frac{10}{x} \right\rfloor$
- $g : \mathbb{N} \to \mathbb{Z}/101\mathbb{Z},\ x \mapsto 2^x \bmod 101$
- $h : \{0, 1, 2, \ldots, 100\} \to \{0, 1, 2, \ldots, 10\},\ x \mapsto x \bmod 11$
- $i : \mathbb{N} \to \{0, 1, 2, \ldots, 50\},\ x \mapsto \left\lfloor \frac{x}{2} \right\rfloor \bmod 51$

Aufgabe 6 (Multiplikationsmethode mit geschlossener Adressierung): (4 Punkte)

Fügen Sie die folgenden Werte in ein anfangs leeres Array der Länge 8 unter Verwendung der *Multiplikationsmethode* ($c = 0.62$) *ohne Sondierung* (also durch Verkettung) ein:

4, 5, 6, 17, 1, 8, 92, 0.

Aufgabe 7 (Divisionsmethode mit offener Adressierung): (5 Punkte)

Fügen Sie die folgenden Werte in ein anfangs leeres Array der Länge 11 unter Verwendung der *Divisionsmethode* mit *quadratischer Sondierung* ($c_1 = 1.0$, $c_2 = 2.0$) ein:

4, 5, 6, 17, 92, 26.

Aufgabe 8 (Doppeltes Hashing): (1+5 Punkte)

Wir betrachten doppeltes Hashing. Dazu seien

$$h_1(k) = \text{Quersumme}(k) \quad \text{und} \quad h_2(k) = 2k + 1 \ .$$

Die Länge des Arrays m sei 13.

a) Geben Sie die sich daraus ergebende Hashfunktion $h(k, i)$ an.

b) Führen Sie mit Hilfe dieser Hashfunktion folgenden Operationen durch:

- Fügen Sie die folgenden Werte in ein Array ein:

  $$453, 66, 39, 10, 1.$$

- Löschen Sie nun die 10.
- Fügen Sie nun eine 0 zusätzlich ein.

Aufgabe 9 (Rehashing): (1+3 Punkte)

Ein Assistent versucht für ein Array der Länge 13 die ultimative Hashfunktion zu entwerfen und macht nach tagelanger Arbeit folgenden Vorschlag:

Zuerst wird zu dem gegebenen Wert k die Quersumme modulo 13 berechnet. Das Ergebnis sei x. Um das Hashing aber noch besser zu machen wird x mit 7 multipliziert, dann wird dazu 1 addiert und schließlich wieder der Rest der Division durch 13 berechnet. Das Endergebnis ist nun der Hashwert. Falls eine Kollision auftritt, wird diese behoben indem man auf den berechneten Hashwert die Hashfunktion nochmal anwendet.

a) Übersetzen Sie die obige Beschreibung in eine Funktion $h(k)$.

b) Die Kollegen des Assistenten sind eher skeptisch, ob diese Funktion wirklich so gut ist. Ist die Skepsis begründet? *Hinweis: untersuchen Sie insbesondere wie gut $h(k)$ "streut".*