Prof. Dr. Ir. Joost-Pieter Katoen

Christian Dehnert, Jonathan Heinen, Thomas Ströder, Sabrina von Styp

Hinweise:

- Die Übungsblätter sind in Gruppen von je 3 Studierenden aus der gleichen Kleingruppenübung zu bearbeiten.
- Die Lösungen müssen bis Montag, den 18. Juni um 11:00 Uhr in den entsprechenden Übungskasten eingeworfen werden. Sie finden die Kästen am Eingang Halifaxstr. des Informatikzentrums (Ahornstr. 55).
- Namen und Matrikelnummern der Studenten sowie die Nummer der Übungsgruppe sind auf jedes Blatt der Abgabe zu schreiben. Heften bzw. tackern Sie die Blätter!

Aufgabe 1 (Duplikaterkennung): (4 + 4 Punkte)

Da Ihr Chef kürzlich einen Artikel über die Vorzüge von Hashing gelesen hat, wettet er mit seinem alten Schulfreund, dass er Folgen von Zahlen, welche der Schulfreund nennt, schneller als dieser auf Duplikate testen kann. Er bittet Sie, die Umsetzung per Hashing zu übernehmen. Sie implementieren (natürlich unter Protest!) ein Verfahren, welches Hashing mit Verkettung als Kollisionsauflösung benutzt, wie folgt.

- Ihre Methode übernimmt ein Array der Länge n von natürlichen Zahlen als Parameter.
- Sie fügen die Elemente nacheinander in die Hashtabelle ein.
- Gibt es eine Kollision der Hashwerte (sind also schon Elemente in der Liste der entsprechenden Zelle), so prüfen sie nacheinander alle Elemente der Liste, ob der neu eingefügte Wert sich bereits darunter befindet.
- Finden Sie das entsprechende Element dabei, so geben Sie "ja" zurück, um anzuzeigen, dass ein Duplikat gefunden wurde.
- Finden Sie das Element jedoch nicht, so fügen Sie das neue Element wie bisher am Anfang der Liste an und fahren mit dem nächsten einzufügenden Element fort.

Die Hashfunktion, die sie dabei benutzen sollen, hat ihr Chef in einer Übungsaufgabe zur Vorlesung "Datenstrukturen und Algorithmen" gefunden. Sie lautet:

$$h(k) = (k + (k \mod 23)^2 + z) \mod 3001$$

wobei 3001 die Größe der Hashtabelle und z eine Zufallszahl ist, die zur Laufzeit einmal gewählt wird und dann beibehalten wird. Sie testen Ihre Implementierung mit ein paar zufälligen Testeingaben und ermitteln eine Laufzeit von weniger als einer halben Sekunde für Listen von 500000 Elementen und sind damit einigermaßen zufrieden.

a) Stolz zeigt ihr Chef seinem Freund Ihren Algorithmus. Dieser schaut eine Zeit lang darauf, verschwindet für eine Minute und gibt ihm daraufhin eine Eingabesequenz der Länge 500000. Verdutzt stellt ihr Chef fest, dass die Duplikatenbestimmung erstaunlich lange läuft, bevor sie terminiert (nämlich über 10 Minuten). Aufgrund dieser Schmach fragt Sie ihr Chef später wie "in Gottes Namen das passieren konnte!" Erklären Sie ihm das. Dazu gehört eine Worst-Case-Analyse und eine Hypothese, wie der Freund es schaffen konnte, so schnell mit so einer Zahlenfolge aufzuwarten.

b) Zähneknirschend nimmt ihr Chef Ihre Erklärungen zur Kenntnis, bittet Sie seine Ehre zu retten und stellt Ihnen einen großen Bonus in Aussicht. Schlagen Sie ihm ein Verfahren vor, welches eine asymptotisch bessere Worst-Case-Komplexität als die aus **a)** besitzt und sich auf Mittel stützt, die sie bereits in der Vorlesung gelernt haben. Dabei reicht eine Beschreibung des Verfahrens in Worten. Vergessen Sie aber nicht, zu erklären, wieso die Worst-Case-Komplexität Ihres Verfahrens besser als diejenige aus **a)** ist.

Aufgabe 2 (Hashing): (8 + 2 + 2 + 2 Punkte)

In einem kleinen Studentenkino mit $m = 23$ Plätzen wird ein Film gezeigt. Um dem Ansturm Herr zu werden, sollen die Plätze mit einem offenen Hashverfahren (Hashing mit Sondieren als Kollisionsauflösung) auf die Wartenden verteilt werden. Als Schlüssel werden dabei nur die beiden letzten Ziffern der Matrikelnummer verwendet. Betrachten Sie die folgenden beiden Hashfunktionen:

- $h_1(x)$:= Quersumme von x
- $h_2(x)$:= x mod 23 (Division-Rest-Methode)

a) Welche Platznummern erhalten die Besucher, wenn sie in der gegebenen Reihenfolge das Kino betreten?

$$6,\ 16,\ 61,\ 87,\ 69,\ 90,\ 4,\ 43,\ 57,\ 4,\ 12,\ 80,\ 46$$

Verwenden Sie jeweils folgende Hashfunktionen:

(i) lineares Sondieren mit h_1 und mit h_2 als Hashfunktion

(ii) quadratisches Sondieren mit h_1 und mit h_2 als Hashfunktion und mit $c_1 = 2$ und $c_2 = 3$ als Konstanten,

(iii) Doppelhashing mit h_1 als erster und h_2 als zweiter Hashfunktion. Inwieweit muss h_2 geändert werden, um Probleme zu vermeiden?

b) Warum ist ein geschlossenes Hashing (Hashing mit Verkettung als Kollisionsauflösung), in dem geschildertem Szenario, nicht praktikabel bzw. sinnvoll?

c) Angenommen, wir haben eine perfekte Hashfunktion und fügen die obigen Elemente in einen Hash der Größe 23 ein. Wie viele Sondierungen sind dann durchschnittlich bei erfolgreicher Suche nötig?

d) Angenommen, wir haben eine perfekte Hashfunktion und fügen die obigen Elemente in einen Hash der Größe 23 ein. Wie viele Sondierungen sind dann durchschnittlich bei nicht erfolgreicher Suche nötig?

Aufgabe 3 (Sondieren): (6 + 2 Punkte)

a) Es sollen zwei gegebene Hashfunktionen h_1 und h_2 für doppeltes Hashing im Rahmen des Hashing mit offener Adressierung benutzt werden. Das heißt, es wird die Hashfunktion

$$h(k, i) = (h_1(k) + i \cdot h_2(k)) \mod m$$

benutzt, wobei m die Größe der Hashtabelle ist. Es sei nun für einen Schlüssel k bekannt, dass der größte gemeinsame Teiler von $h_2(k)$ und m ein Wert $d \geq 1$ ist. Beweisen Sie, dass die Suche nach einem freien Einfügeplatz für k genau $(1/d)$-tel der Tabellenplätze überprüft, bevor sie schließlich wieder zum Ausgangspunkt (also $h_1(k)$) zurückkehrt. Gehen Sie dabei davon aus, dass alle überprüften Plätze belegt sind und daher die jeweils nächste Sondierung notwendig ist.

b) Was bedeutet das Ergebnis von **a)**, falls $d = 1$ ist? Welche Folgerung ergibt sich damit für die kluge Wahl von m?