Prof. Dr. Ir. Joost-Pieter Katoen

Sebastian Junges, Benjamin Kaminski, David Korzeniewski, Tim Quatmann

Übung 8

Hinweise:

- Die Lösungen müssen bis **Donnerstag, den 21. Juni um 16:00 Uhr** in den entsprechenden Übungskasten eingeworfen werden. Sie finden die Kästen am Eingang Halifaxstr. des Informatikzentrums (Ahornstr. 55).
- Die Übungsblätter **müssen** in Gruppen von je 3 Studierenden aus der gleichen Kleingruppenübung abgegeben werden.
- Drucken Sie ggf. digital angefertigte Lösungen aus. Abgaben z.B. per Email sind nicht zulässig.
- Namen und Matrikelnummer sowie die **Nummer der Übungsgruppe** sind auf jedes Blatt der Abgabe zu schreiben. Abgaben, die aus mehreren Blättern bestehen **müssen geheftet bzw. getackert** werden! Die **Gruppennummer muss sich auf der ersten Seite oben links** befinden.
- **Bei Nichtbeachten der obigen Hinweise müssen Sie mit erheblichen Punktabzügen rechnen!**

Aufgabe 1 (Countingsort): (15 Punkte)

Sortieren Sie das Array $3, 6, 2, 4, 5, 5, 0, 2$ mit Countingsort ($k = 6$):
Geben Sie dazu das Histogramm, das Positionsarray an, sowie das Ergebnisarray nach jedem Einfügen.

Aufgabe 2 (Hashing II): (25 Punkte)

Wir betrachten vollständig randomisiertes "Hashing" mit offener Adressierung, welches wie folgt funktionieren soll:
Sei m die Größe der Hashtabelle.

Einfügen eines Schlüssels k:

1) Ermittle eine *zufällige* Slotnummer s zwischen 0 und $m-1$, jeweils mit Wahrscheinlichkeit $\frac{1}{m}$.

2a) Falls der Slot s belegt ist, gehe zu 1)

2b) Falls der Slot s frei ist, speichere Schlüssel k in Slot s.

Suchen eines Schlüssels k:

1) Ermittle eine *zufällige* Slotnummer s zwischen 0 und $m-1$, jeweils mit Wahrscheinlichkeit $\frac{1}{m}$.

2a) Falls der Slot s den Schlüssel k nicht enthält, gehe zu 1)

2b) Falls der Slot s den Schlüssel k enthält, gebe s zurück.

a) Leiten Sie in Abhängigkeit vom Füllgrad α die Average-Case-Komplexität für das Einfügen eines Schlüssels k in die Hashtabelle her.

Hinweise:
Für $0 \leq q < 1$ gilt $\sum_{i=0}^{\infty} q^k \cdot i = \frac{q}{(q-1)^2}$ und $\sum_{i=0}^{\infty} q^k = \frac{1}{1-q}$.
Außerdem gilt $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \in \mathcal{O}\big(\ln(n)\big)$.

b) Leiten Sie in Abhängigkeit von der Anzahl $\#(k)$ an Vorkommnissen eines Schlüssels k in der Hashtabelle die Average-Case-Komplexität für die erfolgreiche Suche nach einem Schlüssel k her.

c) Leiten Sie die Average-Case-Komplexität für die erfolglose Suche nach einem Schlüssel k her.

d) Beurteilen sie die Güte von vollständig randomisiertem Hashing auf Grundlage der in der Vorlesung aufgeführten Gütekriterien von Hashfunktionen. Fällt ihnen ein zusätzliches, in der Vorlesung nicht aufgeführtes, Gütekriterium ein, dass im Zusammenhang mit randomisiertem Hashing relevant ist?

Aufgabe 3 (Güte von Hashingfunktionen): (10 Punkte)

Beurteilen Sie anhand der in der Vorlesung aufgeführten Gütekriterien von Hashfunktionen, d.h.

- einfache Berechenbarkeit
- Surjektivität
- Gleichverteilung auf alle Indizes
- Möglichst breite Verteilung ähnlicher Schlüssel auf die Hashtabelle,

jeweils die folgenden Hashfunktionen:

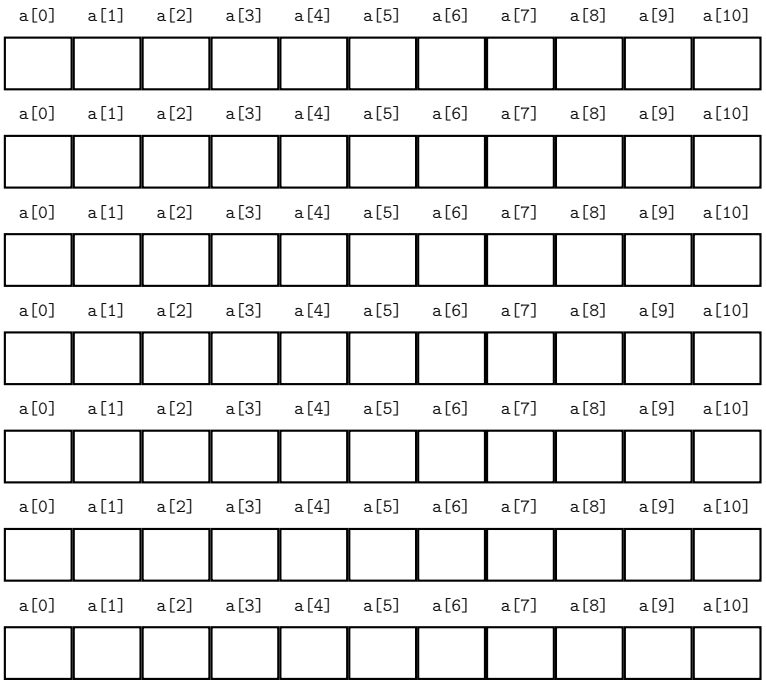
- $f\colon \{1, 2, 3, \ldots, 100\} \to \{0, 1, 2, \ldots, 10\},\ x \mapsto \left\lfloor \frac{10}{x} \right\rfloor$
- $g\colon \mathbb{N} \to \mathbb{Z}/101\mathbb{Z},\ x \mapsto 2^x \bmod 101$
- $h\colon \{0, 1, 2, \ldots, 100\} \to \{0, 1, 2, \ldots, 10\},\ x \mapsto x \bmod 11$
- $i\colon \mathbb{N}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

Begründen Sie ihre Antworten.

Aufgabe 4 (Hashing mit linearer Sondierung): (15 Punkte)

Fügen Sie die folgenden Werte in der angegebenen Reihenfolge in das Array `a` der Länge 11 unter Verwendung der *Multiplikationsmethode* ($c = 0.01$) mit *linear Sondierung* ein:

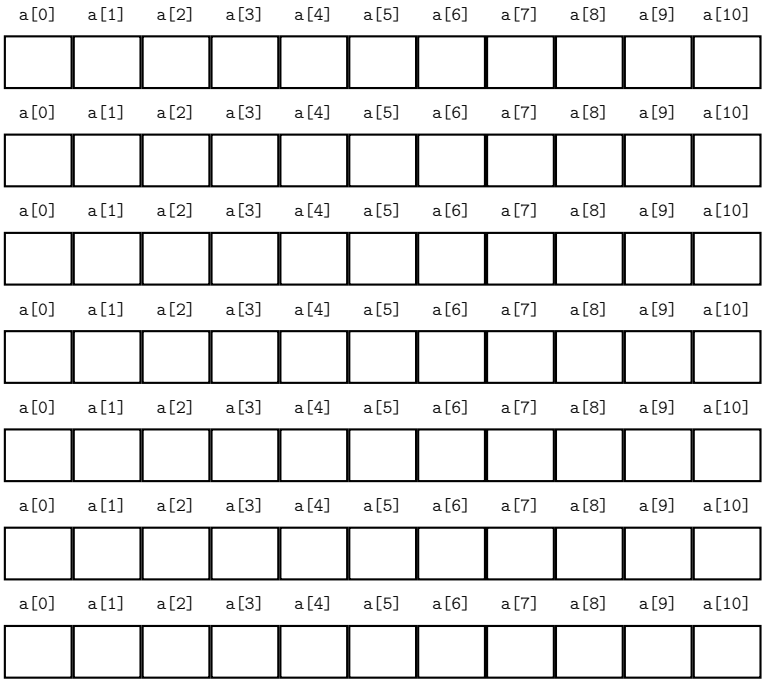
5, 21, 23, 17, 11, 7, 1.



Aufgabe 5 (Hashing mit quadratischer Sondierung): (20 Punkte)

Fügen Sie die folgenden Wert in der angegebenen Reihenfolge in das Array a der Länge 11 unter Verwendung der *Multiplikationsmethode* ($c = 0.01$) mit *quadratischer Sondierung* ($c_1 = 2.0$, $c_2 = 1.0$) ein:

5, 21, 23, 17, 11, 7, 1.



Aufgabe 6 (Laufzeit): (15 Punkte)

Für die nächste Präsenzübung überlegen wir uns, nochmal Laufzeiten abzufragen. Wir haben uns sogar schon die Antworten überlegt, aber uns fehlen noch die Algorithmen. Bitte, helfen Sie uns!
Schreiben Sie, in Pseudocode, **kurze** Algorithmen die einen `print()`-Aufruf so oft, wie angegeben, ausführen. Beachten Sie folgende Regeln:

- Der Algorithmus soll **keine** Rekursion benutzen. Schleifen und If-Abfragen sind natürlich erlaubt.
- Der Pseudo-code soll ausschliesslich folgende Operationen ausführen: $+$, $\cdot$, $/$, mod
- Integer können mit den üblichen Relationen $=, <, \leq$ verglichen werden.

a) A(n): Integer -> void
mit $\frac{2n}{3} + 1$ `print()`-Aufrufe.

b) B(n): Integer -> void
mit $n^3 + n^2$ `print()`-Aufrufe.

c) C(n): Integer -> void
mit $3^n + n^n$ `print()`-Aufrufe.

d) E(n): Integer -> void
mit $\Theta(\log(\log(n)))$ `print()`-Aufrufe.

e) F(n): Integer -> void
mit $\Theta(\sqrt[3]{\log n})$ `print()`-Aufrufe.