Albert-Ludwigs-Universität
Institut für Informatik
Prof. Dr. F. Kuhn
M. Fuchs, G. Schmid

Algorithmen und Datenstrukturen
Sommersemester 2026
Musterlösung Übungsblatt 5

Abgabe: Dienstag, 02. Juni, 2026, 10:00 Uhr

Aufgabe 1: Hashing mit offener Adressierung *(5 Punkte)*

Sei $\mathcal{T}$ eine Hashtabelle der Größe $m = 13$ und seinen $h_1, h_2, h_3 : \mathbb{N}_0 \mapsto \{0, ..., m-1\}$ Hashfunktionen definiert wie folgt[1]:

- $h_1(x) := \overline{x} \mod m$
- $h_2(x) := 3 \cdot x \mod m$
- $h_3(x) := x + 1 \mod m$

Fügen Sie die Schlüssel $23, 12, 75, 945, 30, 99, 345$ (in dieser Reihenfolge) in die initial leere Hashtabelle $\mathcal{T}$ ein. Lösen Sie Konflikte wie folgt:

(a) Lineares Sondieren unter der Benutzung von h_1. *(2 Punkte)*

(b) Doppel-Hashing unter Benutzung von h_2 und h_3.[2] *(3 Punkte)*

Geben Sie den Zustand der Hashtabelle in jedem Schritt an!

Musterlösung

(a)

$$h(x, i) := h_1(x) + i \mod m$$

Berechnung der Hashwerte und Kollisionsbehandlung:

- $x = 23$: $h_1(23) = (2 + 3) \mod 13 = 5$ (keine Kollision, Index 5).
- $x = 12$: $h_1(12) = (1 + 2) \mod 13 = 3$ (keine Kollision, Index 3).
- $x = 75$: $h_1(75) = (7 + 5) \mod 13 = 12$ (keine Kollision, Index 12).
- $x = 945$: $h_1(945) = (9 + 4 + 5) \mod 13 = 5$ (Kollision bei 5, weicht aus auf Index 6).
- $x = 30$: $h_1(30) = (3 + 0) \mod 13 = 3$ (Kollision bei 3, weicht aus auf Index 4).
- $x = 99$: $h_1(99) = (9 + 9) \mod 13 = 18 \mod 13 = 5$ (Kollision bei 5, 6 belegt, weicht aus auf Index 7).
- $x = 345$: $h_1(345) = (3 + 4 + 5) \mod 13 = 12$ (Kollision bei 12, weicht aus auf Index 0).

[1] Wir definieren $\overline{x}$ als die Quersumme von x.
[2] Es soll also $(h_2(x) + i \cdot h_3(x)) \mod m$ als Hashfunktion verwendet werden.

0	1	2	3	4	5	6	7	8	9	10	11	12
-	-	-	-	-	23	-	-	-	-	-	-	-

0	1	2	3	4	5	6	7	8	9	10	11	12
-	-	-	12	-	23	-	-	-	-	-	-	-

0	1	2	3	4	5	6	7	8	9	10	11	12
-	-	-	12	-	23	-	-	-	-	-	-	75

0	1	2	3	4	5	6	7	8	9	10	11	12
-	-	-	12	-	23	945	-	-	-	-	-	75

0	1	2	3	4	5	6	7	8	9	10	11	12
-	-	-	12	30	23	945	-	-	-	-	-	75

0	1	2	3	4	5	6	7	8	9	10	11	12
-	-	-	12	30	23	945	99	-	-	-	-	75

0	1	2	3	4	5	6	7	8	9	10	11	12
345	-	-	12	30	23	945	99	-	-	-	-	75

(b)

$$h(x,i) := h_2(x) + i \cdot h_3(x) \mod m$$

Berechnung der Hashwerte und Kollisionsbehandlung:

- $x = 23$: $h_2(23) = 69 \mod 13 = 4$ (keine Kollision, Index 4).
- $x = 12$: $h_2(12) = 36 \mod 13 = 10$ (keine Kollision, Index 10).
- $x = 75$: $h_2(75) = 225 \mod 13 = 4$. Kollision bei Index 4. Mit $h_3(75) = 76 \mod 13 = 11$ prüfen wir als nächstes $(4 + 1 \cdot 11) \mod 13 = 2$ (keine Kollision, Index 2).
- $x = 945$: $h_2(945) = 2835 \mod 13 = 1$ (keine Kollision, Index 1).
- $x = 30$: $h_2(30) = 90 \mod 13 = 12$ (keine Kollision, Index 12).
- $x = 99$: $h_2(99) = 297 \mod 13 = 11$ (keine Kollision, Index 11).
- $x = 345$: $h_2(345) = 1035 \mod 13 = 8$ (keine Kollision, Index 8).

0	1	2	3	4	5	6	7	8	9	10	11	12
-	-	-	-	23	-	-	-	-	-	-	-	-

0	1	2	3	4	5	6	7	8	9	10	11	12
-	-	-	-	23	-	-	-	-	-	12	-	-

0	1	2	3	4	5	6	7	8	9	10	11	12
-	-	75	-	23	-	-	-	-	-	12	-	-

0	1	2	3	4	5	6	7	8	9	10	11	12
-	945	75	-	23	-	-	-	-	-	12	-	-

0	1	2	3	4	5	6	7	8	9	10	11	12
-	945	75	-	23	-	-	-	-	-	12	-	30

0	1	2	3	4	5	6	7	8	9	10	11	12
-	945	75	-	23	-	-	-	-	-	12	99	30

0	1	2	3	4	5	6	7	8	9	10	11	12
-	945	75	-	23	-	-	-	345	-	12	99	30

Aufgabe 2: Anwendung von Hashtabellen *(5 Punkte)*

Gegeben ist folgender Algorithmus:

Algorithm 1 `algorithm` ▷ Input: Array A of length n with integer entries

```
1: for i = 1 to n − 1 do
2:     for j = 0 to i − 1 do
3:         for k = 0 to n − 1 do
4:             if |A[i] − A[j]| = A[k] then
5:                 return true
6: return false
```

(a) Beschreiben Sie, was `algorithm` berechnet und analysieren Sie die asymptotische Laufzeit. *(2 Punkte)*
Hinweis: Die Differenz $|A[i] - A[j]|$ kann beliebig große Werte annehmen.

(b) Beschreiben Sie einen auf *hashing* basierenden alternativen Algorithmus $\mathcal{B}$ für dieses Problem (d.h. $\mathcal{B}(A) =$ `algorithm`(A) für jede Eingabe A) mit einer Laufzeit von $\mathcal{O}(n^2)$ (mit Begründung). *(3 Punkte)*
Hinweis: Sie dürfen annehmen, dass das Einfügen und Finden von Schlüsseln in einer Hashtabelle $\mathcal{O}(1)$ Zeitschritte benötigt, wenn $\alpha = \mathcal{O}(1)$ (α ist der Load der Hashtabelle).

(c) **Bonusaufgabe:** Beschreiben Sie einen weiteren Algorithmus für dieses Problem ohne Verwendung von Hashing mit einer Laufzeit von $\mathcal{O}(n^2 \log n)$ (mit Begründung). *(5 Bonuspunkte)*

Musterlösung

(a) Der Algorithmus testet, ob es zwei Elemente im Array gibt, deren Abstand (Betrag der Differenz) einem Wert im Array entspricht. Falls ja, so gibt der Algorithmus "true" aus, andernfalls "false". In letzterem Fall werden alle Schleifen vollständig durchlaufen. Man betrachtet dabei

$$\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} = \mathcal{O}(n^2)$$

viele Paare (i, j) und für jedes dieser Paare checkt man n mal die Bedingung in Zeile 4. Die Laufzeit ist also $\mathcal{O}(n^3)$

(b) Wir fügen alle n Elemente des Arrays A in eine Hashtabelle ein. Dann iterieren wir über all $O(n^2)$ Index-Paare $\{(i, j) \mid 0 \leq i < j \leq n-1\}$ und berechnen die Differenz $|A[i] - A[j]|$. Wenn die Differenz in unserer Hashtabelle vorkommt, dann existiert auch ein Element $A[k]$ in unserem Array mit $A[k] = |A[i] - A[j]|$, also geben wir true zurück.

Das Einfügen der n Elemente in die Hashtabelle kostet uns $O(n)$ Zeit, da jede insert Operation nur konstante Zeit benötigt. Das Iterieren der $O(n^2)$ Index-Paare braucht $O(n^2)$ Zeit, da jede find Operation wieder nur konstante Zeit benötigt. Die Gesamtkosten betragen also $\mathcal{O}(n^2)$.

(c) Man sortiert A (Kosten $\mathcal{O}(n \log n)$). Nun Iterieren wir über all $O(n^2)$ Index-Paare $\{(i, j) \mid 0 \leq i < j \leq n-1\}$ und berechnen die Differenz $|A[i] - A[j]|$. Nun testet man für jeden Wert $|A[i] - A[j]|$, ob dieser im sortierten A vorkommt, unter Verwendung von binary search. Dies verursacht n^2 mal Kosten $\mathcal{O}(\log n)$. Die Gesamtlaufzeit wird also durch den letzten Schritt dominiert und beträgt $\mathcal{O}(n^2 \log n)$.

Aufgabe 3: (Keine) Familien Universeller Hashfunktionen *(10 Punkte)*

(a) Sei $\mathcal{S} = \{0, \ldots, M-1\}$. Wir definieren $\mathcal{H}_1 := \{h : x \mapsto a \cdot x^2 \bmod m \mid a \in \mathcal{S}\}$. Zeigen Sie, dass $\mathcal{H}_1$ *nicht* c-universell ist, für konst. $c \geq 1$ (d.h., c ist fest und unabhängig von m). *(4 Punkte)*

(b) Sei nun m *prim* und $k = \lfloor \log_m M \rfloor$. Wir betrachten Schlüssel $x \in \mathcal{S}$ in ihrer Darstellung zur Basis m, d.h., $x = \sum_{i=0}^{k} x_i m^i$. Betrachten Sie die in der Vorlesung (Woche 5, Folie 15) vorgestellte Funktionsmenge

$$\mathcal{H}_2 := \Big\{ h : x \mapsto \sum_{i=0}^{\mathbf{k}} a_i x_i \mod m \mid a_i \in \{0, \ldots, m-1\} \Big\}.$$

Zeigen Sie, dass $\mathcal{H}_2$ 1-universell ist. *(6 Punkte)*

Hinweis: Zwei Schlüssel $x \neq y$ müssen sich in mindestens einer Stelle $x_j \neq y_j$ ihrer Basis m-Darstellung unterscheiden.
Nachtrag: Da m Prim ist, existiert zu jedem Wert $r \in \{1, \ldots, m-1\}$ sein Inverses $e \in \{1, \ldots, m-1\}$, sodass $r \cdot e \equiv 1 \mod m$.

Musterlösung

(a) Für ein $x \in \mathcal{S}$ sei $y = x + i \cdot m \in \mathcal{S}$ für ein $i \in \mathbb{Z} \setminus \{0\}$. D.h. y unterscheidet sich nur um ein Vielfaches von m von x. Solch ein y gibt es zu x für $M > 2m$. Sei $h \in \mathcal{H}_1$. Dann ist

$$\begin{aligned} h(y) &= a \cdot y^2 \mod m \\ &\equiv a \cdot (x + im)^2 \mod m \\ &\equiv a \cdot (x^2 + 2xim + (im)^2) \mod m \\ &\equiv a \cdot x^2 \mod m = h(x). \qquad \textit{(die wegfallenden Terme sind Vielfache von } m\textit{)} \end{aligned}$$

Es folgt, dass $|\{h \in \mathcal{H}_1 \mid h(x) = h(y)\}| = |\mathcal{H}_1|$. Für $m > c$ gilt also

$$|\{h \in \mathcal{H}_1 \mid h(x) = h(y)\}| > \frac{c}{m} |\mathcal{H}_1| \ .$$

Somit ist $\mathcal{H}_1$ für konstante c nicht c-universell.

(b) Seien $x, y \in \mathcal{S}$ beliebig mit $x \neq y$. Sei $x_j \neq y_j$ die erste Stelle der Basis m Darstellung in welcher sich x und y unterscheiden. Sei $h \in \mathcal{H}_2$. Angenommen die Schlüssel $h(x) = h(y)$ kollidieren.

$$\begin{aligned} & h(x) = h(y) \\ \iff \quad & \sum_{i=0}^{k} a_i x_i \equiv \sum_{i=0}^{k} a_i y_i \mod m \\ \iff \quad & a_j \underbrace{(x_j - y_j)}_{\neq 0} \equiv \sum_{i \neq j} a_i (y_i - x_i) \mod m \\ \iff \quad & a_j \equiv (x_j - y_j)^{-1} \sum_{i \neq j} a_i (y_i - x_i) \mod m \qquad (x_j - y_j)^{-1} \textit{ existiert da } m \textit{ prim} \end{aligned}$$

Dies bedeutet, dass für eine Funktion aus $\{h \in \mathcal{H}_2 \mid h(x) = h(y)\}$ der Parameter a_j bereits eindeutig durch die Wahl von $a_0, \ldots, a_{j-1}, a_{j+1}, \ldots, a_k$ bestimmt ist. Man hat also m^k Möglichkeiten eine Funktion aus $\{h \in \mathcal{H}_2 \mid h(x) = h(y)\}$ zu wählen. Es folgt

$$\frac{|\{h \in \mathcal{H}_2 \mid h(x) = h(y)\}|}{|\mathcal{H}_2|} = \frac{m^k}{m^{k+1}} = \frac{1}{m} \ .$$