

Kapitel 4

Streuen

Wir behandeln nun Implementationen *ungeordneter* Wörterbücher, in denen die Schlüssel ohne Beachtung ihrer Sortierreihenfolge gespeichert werden dürfen, verlangen aber, dass es sich bei den Schlüsseln um Zahlen handelt. Dies ist keine starke Voraussetzung, weil sich die Elemente anderer Schlüsseluniversen als Zahlen codieren lassen.

Bezeichnungen:

(Schlüssel-) Universum	$\mathcal{U} \subseteq \mathbb{N}_0$
Schlüsselmenge	$K \subseteq \mathcal{U}, \quad K = n$
Hashtabelle	$H[0, \dots, m - 1]$ (Array von Zeigern auf Datenelemente)

typisch
in JAVA:
Speicher-
adresse
`Object.hashCode`

Durch eine Hashfunktion $h : \mathcal{U} \rightarrow \{0, \dots, m - 1\}$ wird jedem zulässigen Schlüssel $k \in \mathcal{U}$ seine Speicherstelle $H[h(k)]$ in der Hashtabelle H zugewiesen.

hash (engl.):
streuen

Im Idealfall gilt für die Menge $K \subseteq \mathcal{U}$ der vorkommenden Schlüssel

$$k_1 \neq k_2 \in K \implies h(k_1) \neq h(k_2) \quad (h|_K \text{ injektiv})$$

weil Einfügen, Suchen und Löschen dann jeweils in $\Theta(1)$ realisiert werden können.

4.1 Beispiel (Hashfunktionen)

1. $h(k) = k \bmod m$ für Primzahl m
(Divisions- oder Kongruenzmethode)

2. $h(k) = \lfloor m \cdot (k\alpha - \lfloor k\alpha \rfloor) \rfloor$ z.B. für $\alpha = \phi^{-1} = \frac{\sqrt{5}-1}{2} \approx 0,61803$
(Multiplikationsmethode)

goldener
Schnitt

Die $n+1$ Intervalle nach Einfügen von $1, \dots, n$ haben nur 3 verschiedene Größen; bei $\alpha = \phi^{-1}$ am ähnlichsten.

4.1 Kollisionen

4.1

Im Allgemeinen ist allerdings $h(k_1) = h(k_2)$ für einige $h_1, h_2 \in K$. Dies bezeichnen wir mit Kollisionen, k_1, k_2 heißen Synonyme.

$p_{m,n} := P(\text{keine Kollisionen}) = 1 \cdot \frac{m-1}{m} \cdot \frac{m-2}{m} \cdot \dots \cdot \frac{m-n+1}{m}$, falls Schlüssel durch h gleichmäßig gestreut werden, d.h. alle Positionen gleichwahrscheinlich sind.

4.2 Beispiel (Geburtstagsparadoxon)

$$p_{365,22} > 0.5 > p_{365,23}$$

Anschauliche Deutung: bei 23 oder mehr Personen ist die Wahrscheinlichkeit, dass zwei am gleichen Tag Geburtstag haben, größer als die Wahrscheinlichkeit, dass alle an verschiedenen Tagen Geburtstag haben.

Herleitung:

$$\begin{aligned} p_{m,n} &= \prod_{i=0}^{n-1} \frac{m-i}{m} = \prod_{i=1}^{n-1} \left(1 - \frac{i}{m}\right) \\ &\approx \prod_{i=1}^{n-1} e^{-i/m} = e^{-\frac{1}{m} \sum_{i=1}^{n-1} i} = e^{-\binom{n}{2}/m} \end{aligned}$$

Für welches n ist $p_{m,n} \approx 1/2$?

$$\begin{aligned} e^{-\binom{n}{2}/m} &= \frac{1}{2} \\ \Leftrightarrow -\binom{n}{2}/m &= \ln \frac{1}{2} \\ \Leftrightarrow \binom{n}{2} &= m \ln 2 \\ &\parallel \\ \frac{n(n-1)}{2} & \\ \Rightarrow \frac{n}{n} &\approx \sqrt{2(\ln 2) \cdot m} \quad (\approx 22.49 \text{ für } m = 365) \end{aligned}$$

Erwartete Anzahl Kollisionen Sei

$$X_{ij} = \begin{cases} 1 & \text{falls } h(k_i) = h(k_j) \quad (\text{Kollision von } k_i, k_j) \\ 0 & \text{sonst} \end{cases}$$

Bei gleichmäßiger Streuung ist $E(X_{ij}) = 1/m$ für $i \neq j$.

$$E(X) = E\left(\sum_{i < j} X_{ij}\right) = \sum_{i < j} E(X_{ij}) = \sum_{i < j} 1/m = \binom{n}{2} \cdot 1/m$$

Im Beispiel war $m = 365$, somit gilt hier

$$E(X) = \begin{cases} < 1 & \text{für } n < 27 \\ \geq 1 & \text{für } n \geq 27 \end{cases}$$

Im Allgemeinen gilt

$$E(X) \geq 1 \Leftrightarrow n(n-1) \geq 2m,$$

also etwa bei $n \geq \sqrt{2m}$.

4.3 Definition (Belegungsfaktor)

Wir definieren den Belegungsfaktor (load factor) als $\beta = \frac{n}{m}$.

→ entscheidend für Nutzen

?

Erwartete Anzahl leerer Felder? Element $k \in K$ steht nicht an Position $i \in \{0, \dots, m-1\}$ mit Wahrscheinlichkeit $\frac{m-1}{m} = 1 - \frac{1}{m}$.

Die Wahrscheinlichkeit, dass Position i nicht belegt ist, ist damit

$$\begin{aligned} P(i \notin \{h(k_1), \dots, h(k_n)\}) &= \left(1 - \frac{1}{m}\right)^n = \left[\underbrace{\left(1 - \frac{1}{m}\right)^m}_{\approx e^{-1/m}}\right]^\beta \approx e^{-\beta} \\ &= E(X_i) \quad \text{für } X_i = \begin{cases} 0 & H[i] \text{ belegt} \\ 1 & H[i] \text{ frei} \end{cases} \end{aligned}$$

$$E(X) = E(\sum X_i) = \sum E(X_i) = \sum e^{-\beta} = m \cdot e^{-\beta}.$$

↪ Die erwartete Anzahl belegter Einträge ist $m - m \cdot e^{-\beta} = m(1 - e^{-\beta})$.

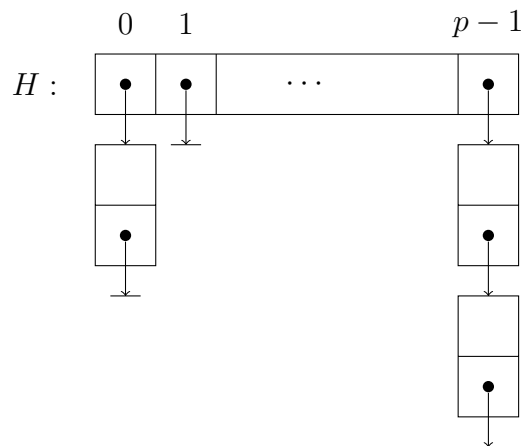
$$\begin{array}{lll} \beta = 1/2 : & 1 - e^{-1/2} \approx 0.39 & \text{Belegung } 39\% \\ \beta = 1 : & 1 - e^{-1} \approx 0.63 & \text{Belegung } 63\% \end{array}$$

4.2 Kollisionsbehandlung

4.2.1 Verkettung

Idee Die Hashtabelle wird als Array von Zeigern auf Listen implementiert, wobei die Listen alle Schlüssel mit gleichem Hashfunktionswert enthalten.

4.4 Beispiel ($h(k) = k \bmod p$)



4.5 Definition (Sondieren)

Wenn die Hashtabelle über Verkettung realisiert wird, finden bei `find` eventuell mehrere Zugriffe auf Zeiger statt (Sondieren). Um die Anzahl der Sondierschritte zu analysieren, definieren wir

$A(m, n)$:= mittlere Anzahl bei erfolgreicher Suche.

$A'(m, n)$:= mittlere Anzahl, falls Schlüssel nicht in der Hashtabelle ist.

Wir gehen wieder davon aus, dass für das verwendete Modell

$$P(h(k) = i) = \frac{1}{m} \quad \forall k \in \mathcal{U}, i \in \{0, \dots, m-1\}$$

gilt.

Die mittlere Länge der Listen ist gerade der Belegungsfaktor $\beta = \frac{n}{m} \in [0, \infty)$. Also ist $A'(m, n) = 1 + \beta = 1 + \frac{n}{m}$.

Zu $A(m, n)$: Die Suche nach k_i ergibt genau die Sondierungen wie beim Einfügen von k_i , wenn vorher k_1, \dots, k_{i-1} eingefügt wurden. Im Mittel ist damit

$$\begin{aligned} A(m, n) &= \frac{1}{n} \sum_{i=0}^{n-1} A'(m, i) = \frac{1}{n} \sum_{i=0}^{n-1} \left(1 + \frac{i}{m}\right) = 1 + \frac{1}{nm} \cdot \sum_{i=0}^{n-1} i \\ &= 1 + \frac{(n-1) \cdot n}{2nm} = 1 + \frac{\beta}{2} - \frac{1}{2m} \leq 1 + \frac{\beta}{2} \end{aligned}$$

Statistische Eigenschaften Sei $\beta = 1$ ($n = m$). Dann sind 63% der Hashtabelle T besetzt, 37% sind frei (s.o.). Wie groß muss n im Mittel sein, damit alle Plätze in T belegt sind?

Sei $(k_i)_{i=1,2,3,\dots}, k_i \in \mathcal{U}$ die Folge in T einzufügender Schlüssel. Sei $m(i)$ die Anzahl belegter Plätze nach Einfügen von k_1, \dots, k_i und sei $i_k := \min\{i | m(i) = k\}$, d.h. Einfügen von k_{i_k} belegt erstmals den k -ten Platz in T .

Wir betrachten eine Zufallsvariable X mit

$$\begin{aligned} X_k &= i_k - i_{k-1} \\ X &= \sum_{k=1}^m X_k \end{aligned}$$

Dann ist die gesuchte Zahl für n gerade $E[X]$.

$i :$	1	2	3	4	5			\dots	k'	k''	
	k_1	k_2	k_3	k_4	k_5	\dots			k'	k''	
$m(i)$	1	2	3	4				$m-1$		m	
	i_1	i_2	i_3	i_4		i_5		\dots		i_{m-1}	i_m
	X_1	X_2	X_3	X_4		X_5		\dots		X_{m-1}	X_m

Sei $i_{k-1} \leq i < i_k$. Dann ist

$$\begin{aligned} p_k &:= P(i+1 = i_k) = \frac{m-k+1}{m} \\ E[X_k] &= \frac{1}{p_k} \\ E[X] &= E\left[\sum_{k=1}^m X_k\right] = \sum_{k=1}^m E[X_k] = \sum_{k=1}^m \frac{m}{m-k+1} \\ &= m \cdot \sum_{k=1}^m \frac{1}{k} = m \cdot H_m \approx m \cdot \ln m \end{aligned}$$

Coupon
Collector
Problem

Index k
etwas un-
glücklich
bei k für
key

Im Mittel sind also $n = m \ln m$ Einfügeoperationen notwendig, um alle m Positionen in der Hashtabelle zu füllen. Dann ist $\beta = \frac{m \ln m}{m} = \ln m > 1$.

4.2.2 Open Hashing

Idee In jedem Feld der Hashtabelle H wird nur ein Element gespeichert (daher $\beta \leq 1$).

Füge Schlüssel k_1 ein: $h(k_1) = i, H[i] = k_1$.

Füge Schlüssel k_2 ein (Kollision): $h(k_2) = i, H[i]$ besetzt. \curvearrowright Finde freien Platz in H .

Wähle eine Folge $(d_i)_{i=1,2,\dots}, d_i \in \mathbb{Z}$ und teste

$$H[(h(k) + d_i) \bmod m], \quad i = 1, 2, 3, \dots$$

Für die Folge $(d_i)_i$ gibt es verschiedene Wahlen:

lineares Sondieren: $d_i = i$

quadratisches Sondieren: $d_i = i^2$

double Hashing: $d_i = i \cdot h'(k)$, wobei $h' : \mathcal{U} \rightarrow \{1, \dots, m-1\}$ eine zweite Hashfunktion ist mit $h'(k) \neq 0 \quad \forall k \in \mathcal{U}$. Hier sollte m prim sein! Warum?

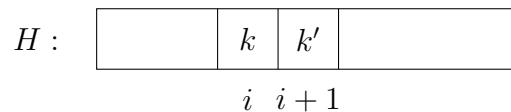
Index bei h' ?

4.6 Problem

Einfügen und Suchen ist damit klar, Löschen ist aber ein Problem.

4.7 Beispiel

Sei $h(k) = h(k') = i$ und füge k, k' ein:



Lösche dann k . \curvearrowright Anschließend versagt die Suche nach k' !

Lösung:

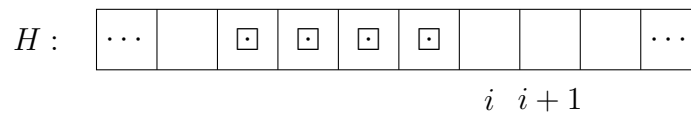
- markiere gelöschte Schlüssel als "gelöscht"

- bei Suche werden sie behandelt wie belegte Felder
- beim Einfügen wie freie Felder
- ungünstig, wenn oft Elemente gelöscht werden, da die Suche verteuert.

4.8 Problem

Clusterbildung

4.9 Beispiel (lineares Sondieren)



Nächster einzufügender Schlüssel: $k \in \mathcal{U}$

$$\left. \begin{aligned} P(h(k) = i + 1) &= \frac{1}{\frac{n}{m}} \\ P(h(k) = i) &= \frac{1}{\frac{n}{m}} \end{aligned} \right\} \Rightarrow \text{Cluster wachsen tendenziell}$$

Analyse

lineares Sondieren	double Hashing
$A \approx \frac{1}{2}(1 + \frac{1}{1-\beta})$	$A \approx \frac{1}{\beta} \ln(\frac{1}{1-\beta})$
$A' \approx \frac{1}{2}(1 + \frac{1}{(1-\beta)^2})$	$A' \approx \frac{1}{1-\beta}$

Beweis. lineares Sondieren: schwer, s. Schönig S.140-141.

Referenz

double Hashing: (idealisierende) Annahme: Sondierungen $h_1(k), h_2(k), \dots$ zur erfolglosen Suche von k mit

$$\begin{aligned} P(H[h_i(k)] \text{ ist frei}) &= 1 - \frac{n}{m} = 1 - \beta \\ E[\text{Anzahl Sondierungen bis freies Feld gefunden}] &= \frac{1}{1-\beta} \end{aligned}$$

Also $A'(n, m) = \frac{1}{1-\beta}$.

$$\begin{aligned} A(n, m) &= \frac{1}{n} \sum_{i=0}^{n-1} A'(m, i) = \frac{1}{n} \sum_{i=0}^{n-1} \frac{1}{1-\frac{i}{m}} = \frac{m}{n} \sum_{i=0}^{n-1} \frac{1}{m-i} \\ &= \frac{1}{\beta} \left(\frac{1}{m-n+1} + \dots + \frac{1}{m} \right) = \frac{1}{\beta} (H_m - H_{m-n}) \\ &\approx \frac{1}{\beta} (\ln m - \ln(m-n)) = \frac{1}{\beta} \ln \frac{m}{m-n} \\ &= \frac{1}{\beta} \ln \frac{1}{1-\beta} \end{aligned}$$

□

4.3 Kollisionsvermeidung

4.3.1 Streufunktionen

Ziel gleichmäßige Streuung durch Hashfunktion, wie in der Berechnung der Kollisionswahrscheinlichkeiten unterstellt.

Problem Wir wissen nicht, welche Schlüssel $K \subseteq \mathcal{U}$ gespeichert werden. Schlimmstenfalls ist $h(k_1) = H(k_2) \quad \forall k_1, k_2 \in K$

→ Sondieren bei Suche entspricht linearer Suche $\mathcal{O}(n)$.

Schlüsselmengen $K \subseteq \mathcal{U}$ führen zu sehr verschiedenen Laufzeiten. Versuche daher, die Wahrscheinlichkeit für schlechtes Laufzeitverhalten besser auf diese zu verteilen.

Idee Wähle aus einer Menge von Hashfunktionen

$$\mathcal{H} \subseteq \{h : \mathcal{U} \rightarrow \{0, \dots, m-1\}\}$$

zufällig eine aus.

4.3.2 Universelles Streuen

4.10 Definition (universelles Streuen)

Eine Familie von Streufunktionen $\mathcal{H} \subseteq \{h : \mathcal{U} \rightarrow \{0, \dots, m-1\}\}$ heißt universell, falls

$$|\{h \in \mathcal{H} : h(k_1) = h(k_2)\}| \leq \left\lceil \frac{|\mathcal{H}|}{m} \right\rceil \quad \forall k_1, k_2 \in \mathcal{U}.$$

Bei zufälliger Wahl $h \in \mathcal{H}$ sind wir dann berechtigt, $P(h(k_1) = h(k_2)) = \frac{1}{m}$ anzunehmen.

4.11 Satz

Ist \mathcal{H} universell, dann ist für $k \in \{k_1, \dots, k_n\} = K \subseteq \mathcal{U}$ bei zufällig gewähltem $h \in \mathcal{H}$ die erwartete Anzahl Kollisionen gerade $\frac{n}{m} = \beta$.

Beweis. Sei C_{ij} eine Zufallsvariable mit

$$C_{ij} = \begin{cases} 1 & h(k_i) = h(k_j) \\ 0 & \text{sonst} \end{cases}$$

Da \mathcal{H} universell ist, ist $P(C_{ij} = 1) = \frac{1}{m}$, und es folgt für $k = k_i$

$$E\left(\sum_{k_j \in K \setminus \{k_i\}} C_{ij}\right) = \sum_{k_j \in K \setminus \{k_i\}} E(C_{ij}) \leq \frac{|K|}{m} = \beta.$$

□

Die erwartete Anzahl Kollisionen entspricht dann also wie erhofft dem Belegungsfaktor β .

4.12 Satz

Die Familie von Streufunktionen

$$\mathcal{H}_p = \{(ak + b \pmod p) \pmod m : 0 < a < p, 0 \leq b < p\}, p \text{ prim}, p > m$$

ist universell.

Beweis. Sei $h_{a,b} \in \mathcal{H}_p$ mit $h_{a,b}(k) = (ak + b \pmod p) \pmod m$.

Betrachte zunächst für $k, k' \in \{0, \dots, p-1\}$

$$\begin{aligned} r &= (ak + b) \pmod p \\ s &= (ak' + b) \pmod p \end{aligned}$$

Dann ist $r - s \equiv \underbrace{a}_{\neq 0}(k - k') \pmod p$, also $r \neq s$ falls $k \neq k'$, weil p prim ist.

Es gibt also zunächst keine Kollisionen der Zahlen $\{0, \dots, p-1\}$, sondern diese werden nur permutiert.

Außerdem liefert jede der $(p-1) \cdot p$ Wahlen für (a, b) ein anderes Paar $r \neq s$, denn gegebene $r \neq s$ können wir nach

$$\begin{aligned} a &= (r - s) \cdot \underbrace{(k - k')^{-1}}_{\text{Inverses bzgl. } \mathbb{Z}_p} \pmod p \\ b &= (r - ak) \pmod p \end{aligned}$$

auffösen. Weil es aber auch nur $p(p-1)$ Paare (r, s) mit $r \neq s$ gibt, liegt eine Bijektion vor. Werden a, b zufällig gewählt, dann erhalten wir also auch jedes Paar $r \neq s \in \{0, \dots, p-1\}$ mit gleicher Wahrscheinlichkeit.

Die Kollisionswahrscheinlichkeit von $k \neq k' \in \mathcal{U}$ entspricht damit der Wahrscheinlichkeit, dass $r \equiv s \pmod m$ für zufällig gewählte $r \neq s \in \{0, \dots, p-1\}$. Für festes r ist die Anzahl der s mit $r \neq s$ und $r \equiv s \pmod m$ höchstens

$$\left\lceil \frac{p}{m} \right\rceil - 1 \leq \frac{p+m-1}{m} - 1 = \frac{p-1}{m}$$

und wegen der zufälligen Wahl aus den $p-1$ möglichen $s \neq r$ ist die Kollisionswahrscheinlichkeit für $k \neq k' \in \mathcal{U}$ höchstens $\frac{1}{m}$. Also ist \mathcal{H}_p universell. \square

Wir wollen nun noch eine andere universelle Familie von Streufunktionen betrachten.

Annahme \mathcal{U} besteht aus Bitstrings fester Länge m , m ist prim.

Zerlege $K \subseteq \mathcal{U}$ in Blöcke der Länge $\lceil \log m \rceil$

$$k : \begin{array}{|c|c|c|c|} \hline & \dots & & \\ \hline k_r & & k_1 & k_0 \\ \hline \end{array} \quad \curvearrowright \quad 0 \leq k_i < m, \quad i = 0, \dots, r$$

Wir definieren die Menge von Hashfunktionen $\mathcal{H}_B = \{h_a : \mathcal{U} \rightarrow \{0, \dots, m-1\}\}$ durch

$$h_a(k) = \left(\sum_{i=0}^r a_i k_i \right) \pmod m$$

wobei $a = (a_r, \dots, a_0) \in \{0, \dots, m-1\}^{r+1}$. Damit gilt $|\mathcal{H}_B| = m^{r+1}$.

4.13 Satz

\mathcal{H}_B ist universell.

Beweis. Seien $k \neq k' \in \mathcal{U}$, dann $k_i \neq k'_i$ für ein $i \in \{0, \dots, r\}$. OBdA. sei $i = 0$. Für ein $h_a \in \mathcal{H}_B$ gilt dann

$$\begin{aligned} h_a(k) = h_a(k') &\Leftrightarrow \sum_{i=0}^r a_i k_i \equiv \sum_{i=0}^r a_i k'_i \pmod{m} \\ &\Leftrightarrow a_0(k_0 - k'_0) \equiv \sum_{i=1}^r a_i(k'_i - k_i) \pmod{m} \\ &\Leftrightarrow a_0 \equiv \underbrace{(k_0 - k'_0)^{-1}}_{\text{mult. Inv. in } (\mathbb{Z}_m, +, \cdot)} \sum_{i=1}^r a_i(k'_i - k_i) \pmod{m} \end{aligned}$$

\curvearrowright Für alle Wahlen von $(a_r, \dots, a_1) \in \{0, \dots, m-1\}^r$ existiert genau ein $a_0 \in \{0, \dots, m-1\}$ mit $h_a(k) = h_a(k')$. Also gilt

$$P(\text{Kollision } k \neq k') = \frac{|\{a : h_a(k) = h_a(k')\}|}{|\mathcal{H}_B|} = \frac{m^r}{m^{r+1}} = \frac{1}{m}.$$

□

4.4 Ausblick

- viele andere Techniken zur Kollisionsbehandlung, z.B. Cuckoo-Hashing.
- viele andere Techniken zur Konstruktion von Hashfunktionen, z.B. perfektes Hashing (für statische Schlüsselmengen).
- viele andere Anwendungen, z.B. Bloom-Filter. .

s. Übung