Algorithmentheorie

05 - Hashing

Prof. Dr. S. Albers

Übersicht

- Einführung
- Universelles Hashing
- Perfektes Hashing

Das Wörterbuch-Problem

Gegeben: Universum $U = [0 \ldots N\text{-}1]$, wobei N eine natürliche Zahl ist.

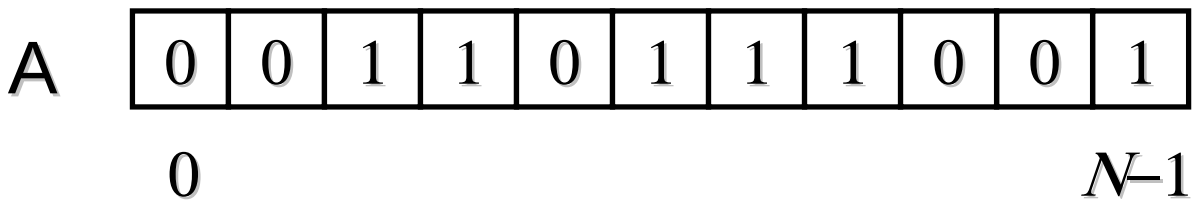
Ziel: Verwalte Menge $S \subseteq U$ mit folgenden Operationen.

- Suche(x,S): Ist $x \in S$?
- Einfüge(x,S): Füge x zu S hinzu, sofern noch nicht vorhanden.
- Entferne(x,S): Entferne x aus S.

Triviale Implementierung

Array A[0…N-1] wobei $A[i] = 1 \Leftrightarrow i \in S$

Jede Operation hat Laufzeit O(1), aber der Platzbedarf ist $\Theta(N)$.



Ziel: Platzbedarf O(|S|) und erwartete Laufzeit O(1).

Idee des Hashings

Verwende ein Array der Länge $O(|S|)$.

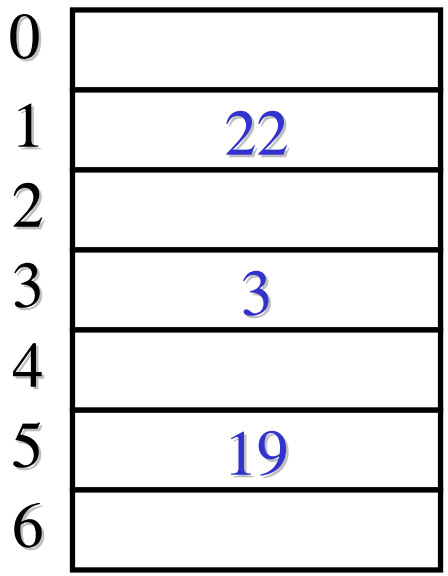
Berechne die Position, an der ein Element abgespeichert wird mit Hilfe einer Funktion aus dem Schlüssel.

Universum	$U = [0 \ldots N\text{-}1]$
Hash-Tafel	Array $T[0 \ldots m\text{-}1]$
Hash-Funktion	$h: U \rightarrow [0 \ldots m\text{-}1]$

Ein Element $x \in S$ wird in $T[h(x)]$ gespeichert.

Beispiel

$N = 100$; $U = [0…99]$; $m = 7$; $h(x) = x \bmod 7$; $S = \{3, 19, 22\}$



Soll als nächstes 17 eingefügt werden, so tritt eine Kollision auf, denn $h(17) = 3$.

Lösungsmöglichkeiten für Kollisionen

- Hashing mit Verkettung: $T[i]$ enthält eine Liste von Elementen.
- Hashing mit offener Adressierung: Statt einer Adresse für ein Element gibt es m viele, die der Reihe nach ausprobiert werden.
- Universelles Hashing: Wähle eine Hash-Funktion, so dass wenige Kollisionen entstehen. Kollisionen werden durch Verkettung aufgelöst.
- Perfektes Hashing: Wähle eine Hash-Funktion, so dass keine Kollisionen entstehen.

Universelles Hashing

Idee: Verwende eine Klasse H von Hash-Funktionen. Die tatsächlich verwendete Hash-Funktion $h \in H$ wird zufällig aus H gewählt.

Ziel: Für jedes $S \subseteq U$ soll die erwartete Laufzeit jeder Operation $O(1 + \beta)$ sein, wobei $\beta = |S|/m$ der Lastfaktor der Tafel ist.

Eigenschaft von H: Für zwei beliebige Elemente $x,y \in U$ führen nur wenige $h \in H$ zu einer Kollision ($h(x) = h(y)$).

Universelles Hashing

Definition: Seien N und m natürliche Zahlen. Eine Klasse

$H \subseteq \{ h : [0\ldots N\text{-}1] \rightarrow [0\ldots m\text{-}1] \}$ heißt universell, wenn für alle $x,y \in U = [0\ldots N\text{-}1]$, $x \neq y$, gilt:

$$\frac{|\{h \in H : h(x) = h(y)\}|}{|H|} \leq \frac{1}{m}$$

Intuitiv: Ein zufällig gewähltes h ist genau so gut, wie wenn die Tafelpositionen der Elemente zufällig gewählt würden.

Eine universelle Klasse von Funktionen

Seien N, m natürliche Zahlen, wobei N prim ist.

Für Zahlen $a \in \{1, \ldots, N\text{-}1\}$ und $b \in \{0, \ldots, N\text{-}1\}$ sei

$h_{a,b} : U = [0 \ldots N\text{-}1] \rightarrow \{0, \ldots, m\text{-}1\}$ definiert durch:

$$h_{a,b}(x) = ((ax + b) \bmod N) \bmod m$$

Satz: $H = \{h_{a,b}(x) \mid 1 \leq a < N \text{ und } 0 \leq b < N\}$ ist eine universelle Klasse von Hash-Funktionen.

Beweis

Betrachte festes Paar x,y mit $x \neq y$.

$h_{a,b}(x) = ((ax+b) \bmod N) \bmod m \quad h_{a,b}(y) = ((ay+b) \bmod N) \bmod m$

1. Paare (q,r) mit $q = (ax+b) \bmod N$ und $r = (ay+b) \bmod N$ durchlaufen für variables a,b den gesamten Bereich $0 \leq q,r < N$ mit $q \neq r$

-- $q \neq r$: $q = r$ impliziert $a(x-y) = cN$

-- Verschiedene Paare a,b ergeben verschiedene Paare (q,r).

$(ax+b) \bmod N = q \qquad (ay+b) \bmod N = r$

$(a'x+b') \bmod N = q \qquad (a'y+b') \bmod N = r$

implizieren $(a-a')(x-y) = cN$

Beweis

Festes Paar x,y mit $x \neq y$.

$h_{a,b}(x) = ((ax+b) \bmod N) \bmod m \quad h_{a,b}(y) = ((ay+b) \bmod N) \bmod m$

2. Wieviele Paare (q,r) mit $q = (ax+b) \bmod N$ und $r = (ay+b) \bmod N$ werden auf die gleiche Restklasse mod m abgebildet?

   Für festes q gibt es nur $(N-1)/m$ Zahlen r, mit
   $q \bmod m = r \bmod m$ und $q \neq r$.

$$|\{h \in H : h(x) = h(y)\}| \leq N(N-1)/m = |H|/m$$

Analyse der Operationen

Annahmen: 1. *h* wird zufällig (gemäß Gleichverteilung) aus einer universellen Klasse *H* gewählt.
2. Kollisionen werden durch Verkettung gelöst.

Für $h \in H$ und $x,y \in U$ sei

$$\delta_h(x,y) = \begin{cases} 1 & h(x) = h(y) \text{ und } x \neq y \\ 0 & \text{sonst} \end{cases}$$

$\delta_h(x,S) = \sum_{y \in S} \delta_h(x,y)$ ist die Anzahl der von *x* verschiedenen Elemente in $T[h(x)]$, wenn *S* gespeichert wird.

Analyse der Operationen

Satz: Sei H eine universelle Klasse und $S \subseteq U = [0 \ldots N\text{-}1]$ mit $|S| = n$.

1. Für $x \in U$ gilt:

$$\frac{1}{|H|}\sum_{h \in H}(1+\delta_h(x,S)) \leq \begin{cases} 1+n/m & x \notin S \\ 1+(n-1)/m & x \in S \end{cases}$$

2. Die erwartete Laufzeit einer Suche-, Einfüge bzw. Lösche-Operation ist $O(1+\beta)$, wobei $\beta = n/m$ der Lastfaktor ist.

Beweis

1.
$$\sum_{h\in H}(1+\delta_h(x,S)) = |H| + \sum_{h\in H}\sum_{y\in S}\delta_h(x,y)$$
$$= |H| + \sum_{y\in S}\sum_{h\in H}\delta_h(x,y)$$
$$\leq |H| + \sum_{y\in S\setminus\{x\}}\frac{|H|}{m}$$
$$\leq \begin{cases} |H|(1+n/m) & x\notin S \\ |H|(1+(n-1)/m) & x\in S \end{cases}$$

2. Folgt aus 1.

Perfektes Hashing

Wähle eine Hash-Funktion, die für die abzuspeichernde Menge *S* injektiv ist. *S* sei im Voraus bekannt.

Zweistufiges Hashverfahren

1. Die erste Stufe verteilt *S* auf “kurze Listen”.
   (Hashing mit Verkettung)
2. In der zweiten Stufe wird für jede Liste eine eigene injektive Hash-Funktion benutzt.

Konstruktion von injektiven Hashfunktionen

Sei $U = [0 \ldots N\text{-}1]$

Für $k \in \{1,\ldots,N\text{-}1\}$ sei

$$h_k : U \rightarrow \{0,\ldots,m\text{-}1\}$$
$$x \rightarrow ((kx) \bmod N) \bmod m$$

Sei $S \subseteq U$. Kann k so gewählt werden, dass h_k eingeschränkt auf S injektiv ist?

h_k eingeschränkt auf S ist injektiv, wenn für alle $x,y \in S$, $x \neq y$, gilt

$$h_k(x) \neq h_k(y)$$

Maß für Verletzung der Injektivität

Für $0 \leq i \leq m\text{-}1$ und $1 \leq k \leq N\text{-}1$ sei

$$b_{ik} = |\{ x \in S : h_k(x) = i \}|$$

Dann gilt:

$$|\{ (x,y) \in S^2 : x \neq y \text{ und } h_k(x) = h_k(y) = i \}| = b_{ik}\,(b_{ik} - 1)$$

Definiere

$$B_k = \sum_{i=0}^{m-1} b_{ik}\,(b_{ik} - 1)$$

B_k misst, wie wenig injektiv h_k eingeschränkt auf S ist.

Injektivität

Lemma 1: h_k eingeschränkt auf S ist injektiv $\Leftrightarrow$ $B_k < 2$

Beweis:

$B_k < 2 \Rightarrow B_k \leq 1 \Rightarrow b_{ik}\,(b_{ik} - 1) \in \{0,1\}$ für alle i

$\Rightarrow b_{ik} \in \{0,1\} \Rightarrow h_k$ eingeschränkt auf S ist injektiv

h_k eingeschränkt auf S ist injektiv $\Rightarrow b_{ik} \in \{0,1\}$ für alle i

$\Rightarrow B_k = 0$

Injektivität

Lemma 2: Sei N Primzahl, $S \subseteq U = [0 \ldots N\text{-}1]$ mit $|S| = n$. Dann gilt

$$\sum_{k=1}^{N-1} \underbrace{\sum_{i=0}^{m-1} b_{ik}(b_{ik}-1)}_{B_k} \leq 2\frac{n(n-1)}{m}(N-1)$$

Bemerkung: $\Sigma_i\, b_{ik}\,(b_{ik}\text{-}1)$ ist zweimal die Anzahl der Kollisionen, falls h_k als Hash-Funktion gewählt wird. Die mittlere Anzahl der Kollisionen ist somit $n(n\text{-}1)/m$.

Ist $m > n(n\text{-}1)$, so ist die mittlere Anzahl < 1,

d.h. es existiert ein h_k, das eingeschränkt auf S injektiv ist.

Folgerungen

Korollar 1: Es gibt mindestens $(N\text{-}1)/2$ viele k mit $B_k \leq 4n(n\text{-}1)/m$. Ein solches k an in erwarteter Zeit $O(m+n)$ bestimmt werden.

Beweis: Sei $A = \{ k : B_k > 4n(n\text{-}1)/m \}$

Annahme: $|A| > (N\text{-}1)/2$

$$\Rightarrow \sum_{k=1}^{N-1} B_k \geq \sum_{k \in A} B_k > \frac{N-1}{2} \frac{4n(n-1)}{m} = \frac{N-1}{m} 2n(n-1)$$

Mit WSK $\geq ½$ erfüllt ein zufällig gewähltes k die Bedingung. Die erwartete Anzahl der Versuche ist ≤ 2.

Folgerungen

Korollar 2:

a) Sei $m = 2n(n\text{-}1)+1$. Dann sind mindestens $(N\text{-}1)/2$ der h_k injektiv auf S. Ein solches h_k findet man in erwarteter Zeit $O(m+n)=O(n^2)$.

b) Sei $m = n$. Dann gilt für mindestens $(N\text{-}1)/2$ der h_k, dass $B_k \leq 4(n\text{-}1)$. Ein solches h_k findet man in erwarteter Zeit $O(n)$.

Beweis: Wegen Korollar 1 gibt es $(N\text{-}1)/2$ viele h_k mit

$$B_k \leq \frac{4n(n-1)}{2n(n-1)+1} < 2.$$

Wegen Lemma 1 ist h_k injektiv.

Zweistufiges Schema

$S \subseteq U = [0 \ldots N\text{-}1]$ $\quad |S| = n = m$

Idee: Wende Kor. 2b an und teile S in Teilmengen der Größe $O(\sqrt{n})$.
Auf jede Teilmenge wende Kor. 2a an.

1. Wähle k mit $B_k \leq 4(n\text{-}1) \leq 4n$.
   $h_k : x \rightarrow ((kx) \bmod N) \bmod n$
2. $W_i = \{ x \in S : h_k(x) = i \}$, $\quad b_i = |W_i|$, $\quad m_i = 2b_i(b_i - 1) + 1$ $\quad$ für $1 \leq i \leq n\text{-}1$
   Wähle k_i so, dass

$$h_{k_i} : x \rightarrow (k_i x \bmod N) \bmod m_i$$

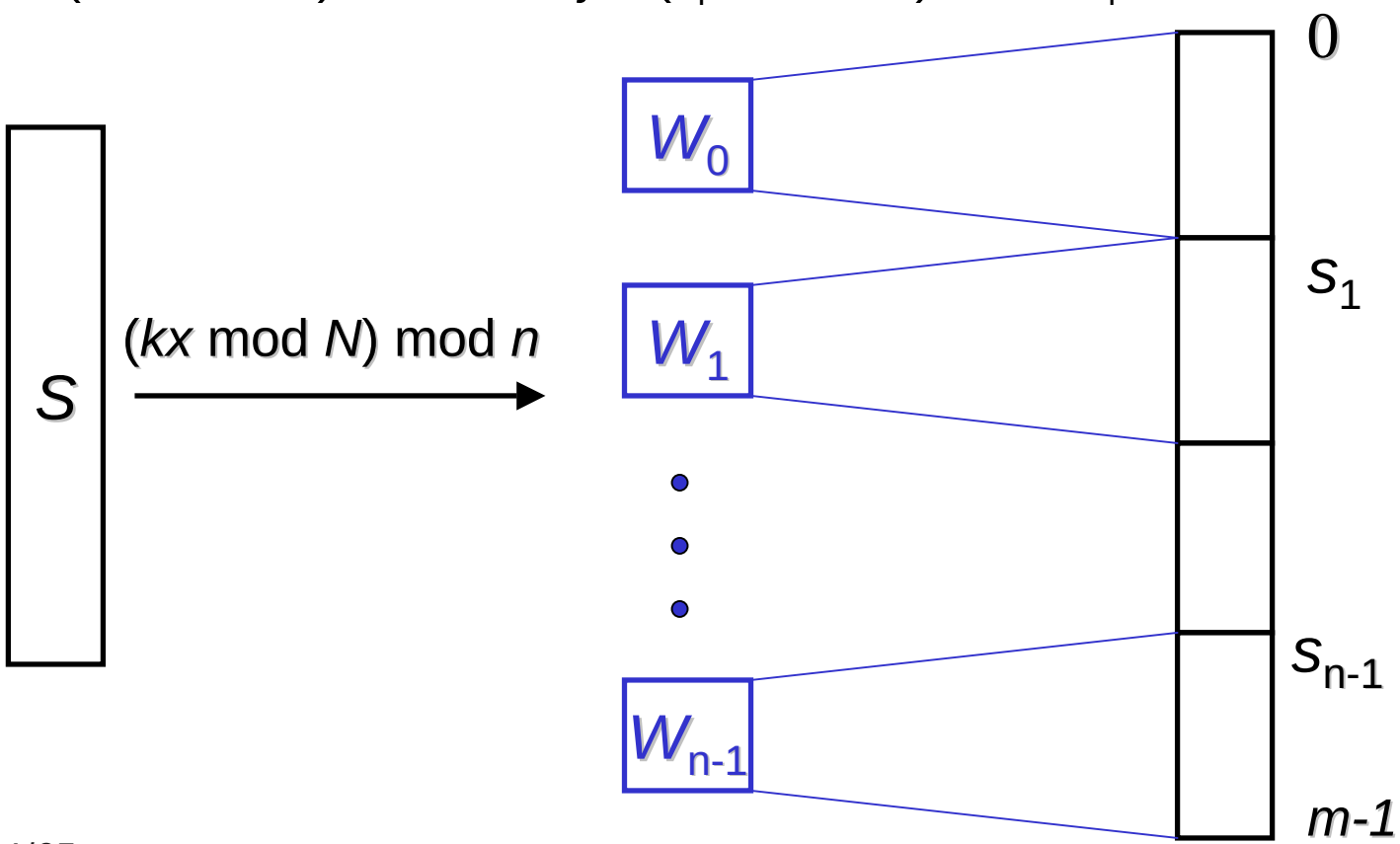
eingeschränkt auf W_i injektiv ist.

Zweistufiges Schema

3. $s_i = \sum_{j<i} m_j$

Speichere $x \in S$ in Tafelposition $T[s_i + j]$ wobei

$i = (k\,x \bmod N) \bmod n$ $\quad$ $j = (k_i\,x \bmod N) \bmod m_i$



Platzbedarf für Hash-Tafel und Hash-Funktion

$$m = \sum_{i=0}^{n-1} m_i = \sum_{i=0}^{n-1} (2b_i(b_i - 1) + 1) = n + 2B_k$$

$$\leq n + 8(n-1) \leq 9n$$

Zusätzlich braucht man Platz für die k_i, m_i und s_i.

Platzbedarf insgesamt O(n).

Zeitbedarf für den Aufbau

- Nach Kor. 2b kann k in erwarteter Zeit $O(n)$ gefunden werden.
- Die W_i, b_i, m_i, s_i können in Zeit $O(n)$ berechnet werden.
- Nach Kor. 2a kann jedes k_i in erwarteter Zeit $O(b_i^2)$ berechnet werden.

Erwartete Gesamtlaufzeit:

$$O\left(n + \sum_{i=0}^{n} b_i^2\right) = O(n + B_k) = O(n)$$

Hauptergebnis

Satz: Sei N eine Primzahl und $S \subseteq U = [0 \ldots N\text{-}1]$ mit $|S| = n$.
Für S kann eine perfekte Hash-Tafel der Größe $O(n)$ und eine Hash-Funktion mit Zugriffszeit $O(1)$ in erwarteter Zeit $O(n)$ aufgebaut werden.