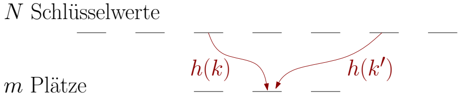
Hashing

- $U = \{0, 1, \dots, N-1\}$ mit $N \in \mathcal{N}$ ist o.B.d.A. das Universum der möglichen Schlüsselwerte
- Speichern des Elements a in Feld der Größe N an Stelle Schlüssel(a)
    - ist effizient (Einfügen, Löschen, Suchen in $O(1)$)
    - benötigt aber oft zu viel Speicherplatz
- Idee von Hashing
    - Definiere *Hashfunktion* $h : U \to \{0, 1, \dots, t-1\}$ mit $t < N \in \mathcal{N}$
    - Annahme: Hashfunktion kann in $O(1)$ evaluiert werden
    - *Hashtabelle* = Feld T der Größe t
    - Speichere Element a an Stelle h(Schlüssel(a))

Schubfachprinzip

- Problem: wir möchten $t < N$ verwenden



- Schubfachprinzip
  - Es gibt Schlüssel $k \neq k'$, so dass $h(k) = h(k')$: *Hashkonflikt*
  - Stärker: es gibt einen Platz $i \in \{0, 1, \ldots, t-1\}$, so dass $h(k) = i$ für mindestens $\lceil \frac{N}{t} \rceil$ Schlüssel k

“Facts of life in probability land”

n Bälle werden in t Eimer geworfen, unabhängig, jeder gleichverteilt auf die Eimer
(oder die “zufällige” Hashfuntion bildet n Schlüssel in t Tabellenplätze ab)

Fakten

- Im Erwartungswert enthält jeder Eimer n/t Bälle.
- Wenn $n \geq \Omega(\sqrt{t})$ dann ist eine Kollision recht wahrscheinlich, d.h. es gibt einen Eimer mit mehr als einem Ball.
- Wenn $n = t$ dann sind im Erwartungswert $n/e \approx n/2.71$ Eimer leer.
- Erst wenn $n = \Omega(t \log t)$ kann man damit rechnen, dass es keinen leeren Eimer gibt.
- Wenn $n = t$ dann ist es sehr wahrscheinlich, dass ein Eimer mindesten $\log n / \log \log n$ Bälle enthält.

Geschlossenes Hashing

- Löse Hashkonflikte mit alternativen Hashfunktionen
- Verwende Folge $h_0, h_1, h_2, \dots$ von Hashfunktionen und speichere Schlüssel a an der Stelle $T[h_i(a)]$ für das kleinste i, für das $T[h_i(a)]$ noch unbesetzt ist
- Achtung: Beim Löschen müssen Elemente nur als gelöscht markiert werden, damit Suchketten nicht unterbrochen werden.

Typische Folgen von Hashfunktionen:

Lineares Sondieren: $h_i(a) = h(a) + i \bmod t$

Quadratisches Sondieren: $h_i(a) = h(a) + i^2 \bmod t$

Hashing mit Verkettung

- Mögliche Art, Hashkonflikte zu lösen
- Statt Feld von Elementen wird Feld von Listen von Elementen verwendet
- Worst-Case Analyse: T enthält n Elemente
  - Suche: Suche Schlüssel k in Liste $T[h(k)]$: $O(n)$
  - Einfügen: Füge Element x am Anfang von $T[h(\text{Schlüssel}(k))]$ ein: $O(1)$
  - Löschen: Lösche Element x aus Liste $T[h(k)]$:
    Mit doppelt verlinkten Listen: Laufzeit Suche + $O(1)$

Worst-Case Analyse

Im schlechtesten Fall sind alle n Elemente in einer Liste gespeichert, und die Laufzeit der Suche ist $\Theta(n)$ – nicht besser als bei einer Liste.

Average-Case Analyse

Laufzeit hängt davon ab, wie gut die Hashfunktion die Schlüssel verteilt.

Einfaches gleichverteiltes Hashing

Annahme Wahrscheinlichkeit, dass ein beliebiges Element in einen der t Plätze hasht, ist gleichverteilt

- $n_j = |T[j]|$ für $j \in \{0, 1, \ldots, t-1\}$
- Erwartungswert $E\left[n_j\right] = \frac{n}{t} := \alpha$
- α nennen wir *Belegungsfaktor*

Satz In einer Hash-Tabelle, in der Konflikte durch Verkettung gelöst werden, und unter Verwendung des einfachen gleichverteilten Hashings, benötigt eine Suche durchschnittlich Zeit $\Theta(1 + \alpha)$.

Beweis In Vorlesung besprochen.

Einfaches gleichverteiltes Hashing

Problem:

- Wir machen Annahmen über die Eingabe, die eigentlich durch nichts begründet sind.
- Für eine gegebene Hashfunktion können besonders “schwierige” Elementemengen gewählt werden, die zu hohen Laufzeiten führen

Universelles Hashing (Carter and Wegman, 1977)

Idee Hashfunktion wird zufällig und unabhängig von Schlüsselwerten gewählt

- Das heisst, das mehrfache Ausführen mit denselben Elementen führt zu unterschiedlichen Hashtabellen
- Wir nehmen den Zufall nun selbst in die Hand

Def. Sei $\mathcal{H}$ eine endliche Menge an Hashfunktionen von U nach $\{0, 1, \ldots, t-1\}$. $\mathcal{H}$ ist *universell* falls für $a, b \in U$ mit $a \neq b$ die Anzahl der Hashfunktionen $h \in \mathcal{H}$ mit $h(a) = h(b)$ höchstens $\frac{|\mathcal{H}|}{t}$ ist.

Universelles Hashing

Idee Hashfunktion wird zufällig und unabhängig von Schlüsselwerten gewählt

- Das heisst, das mehrfache Ausführen mit denselben Elementen führt zu unterschiedlichen Hashtabellen
- Wir nehmen den Zufall nun selbst in die Hand

Def. Sei $c > 0$ eine Konstante. Sei $\mathcal{H}$ eine endliche Menge an Hashfunktionen von U nach $\{0, 1, \ldots, t-1\}$. $\mathcal{H}$ ist *c-universell* falls für $a, b \in U$ mit $a \neq b$ die Anzahl der Hashfunktionen $h \in \mathcal{H}$ mit $h(a) = h(b)$ höchstens $c\frac{|\mathcal{H}|}{t}$ ist.

Universelles Hashing

Satz Sei h eine zufällig gewählte Hashfunktion aus einer Menge $\mathcal{H}$ an universellen Hashfunktionen. Sei T eine Hashtabelle der Größe t, in der Konflikte mit Verkettung gelöst werden, und in die n Schlüssel mit Hilfe von h eingefügt wurden. Der Erwartungswert der Länge der Liste $T[h(a)]$, die nach Schlüssel a durchsucht wird, ist höchstens $1 + \alpha$.

Kor. Universelles Hashing mit Verkettung zur Lösung von Konflikten erlaubt es, in $O(1 + \alpha)$ erwarteter Zeit in einer Hashtabelle der Größe t, die n Elemente enthält, zu suchen.

$\Rightarrow$ Für $t = \Omega(n)$ brauchen die Operationen Suche, Einfügen und Löschen durchschnittlich Zeit $O(1)$.

Universelles Hashing

Wie wird universelles Hashing implementiert?

- Klasse UniversellesHashing
- Im Konstruktor wird mit Zufallsgenerator eine Hashfunktion $h \in \mathcal{H}$ erzeugt.
- Destuktor löscht h.
- So lange eine Instanz existiert, wird die Hashfunktion h nicht verändert.

Universelles Hashing

Zu zeigen: es gibt Mengen von universellen Hashfunktionen

Bsp.

- t ist Primzahl, z.B. 257
- $a \in U$ kann eindeutig als $d + 1$-Tupel $a = \langle a_0, a_1, \ldots, a_d \rangle$ mit $d > 0$ und $0 \leq a_i < t$ geschrieben werden, z.B. bitweise geschrieben
- Hashfunktion: Für jedes $x = \langle x_0, x_1, \ldots, x_d \rangle \in \{0, 1, \ldots, m - 1\}^{d+1}$ definieren wir die Hashfunktion $h_x : U \to \{0, 1, \ldots, t - 1\}$

$$h_x(a) = \left(\sum_{i=0}^{d} a_i x_i \right) \mod t$$

Universelles Hashing

Bsp. Fortsetzung:

Lemma Die Menge $\mathcal{H} = \left\{ h_x | a \in \{0, \ldots, t-1\}^{d+1} \right\}$ ist universell.

Beweis

- Betrachte $a, b \in U$ mit $a \neq b$ und o.B.d.A. $a_0 \neq b_0$
- Es ist $h_x(a) = h_x(b)$ falls
  $x_0 \underbrace{(b_0 - a_0)}_{\neq 0} = \left(\sum_{i=1}^{d} x_i (a_i - b_i) \right) \mod m$
- Sind $x_1, \ldots, x_d$ fest, so gibt es nur eine Wahl für x_0 da t prim ist
- Es gibt t^d Möglichkeiten, um $x_1, \ldots, x_d$ zu wählen
- Es gibt also $t^d \leq \frac{|\mathcal{H}|}{t} = \frac{t^{d+1}}{t}$ Funktionen in $\mathcal{H}$, so dass $h_x(a) = h_x(b)$

Andere Beispiele für Universelle Hashfunktionen (Dietzfelbinger) nicht in Vorlesung behandelt

- $t = 2^k$ und $w \geq k$ ist Bitlänge des Computerwortes

  $$h_x(a) = (a \cdot x \bmod 2^w) \operatorname{div} 2^{w-k}$$

  mit x ungerade ist 2-universell

  $$h_{x,y}(a) = ((a \cdot x + y) \bmod 2^w) \operatorname{div} 2^{w-k}$$

  mit x ungerade und $0 \leq y < 2^{w-k}$ ist universell