Vorlesungsmitschrift

Algorithmen und Berechenbarkeit

Vorlesung 08

Letztes Update: 2017/11/22 - 13:36 Uhr

Weitere Sätze zum Wörterbuchproblem / Hashing

Satz: Sei x ein zufälliges (gleichverteiltes) Element aus S, dann ist die erwartete Suchzeit für x

$$\mathcal{O}\left(1+\frac{1}{n}\sum_{i=0}^{m-1}\frac{l_i\cdot(l_i+1)}{2}\right)$$

Besser wäre eine Schranke unabhängig von l_i, aber $\sum_i l_i = n$ sagt nichts über $\sum_i l_i^2$.

Beweis: Wenn x das j-te Element seiner Liste ist, ist die Suchzeit $\mathcal{O}(1+j)$. Also ist die erwartete Suchzeit

$$\mathcal{O}\left(\sum_{i=0}^{m-1}\sum_{j=1}^{l_i}\right)=\mathcal{O}\left(1+\frac{1}{n}\sum_{i=0}^{m-1}\frac{l_i-(l_i+1)}{2}\right)$$

□

Satz: Sei S eine zufällige (gleichverteilte) Teilmenge aus U der Größe n. Dann ist die erwartete Suchzeit für ein zufälliges $x \in S$

$$\mathcal{O}\left(1+\beta\cdot\frac{3}{2}\cdot l^{\beta}\right)$$

Beweis: Wird nicht bewiesen.

Die beiden letzten Fälle besagen im Prinzip: Falls $S \subseteq U$ zufällig gewählt wurde, ist alles gut. Praktisch ist das aber nicht der Fall.

Perfektes Hashing

Universelles Hashing

Sei $\mathcal{H}$ eine Menge von Hashfunktionen von U nach $\{0,1,\cdots,m-1\}$. Für $c>1$ heißt $\mathcal{H}$ nun c-universell, falls für alle $x,y\in U$ mit $x\neq y$ gilt:

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

Die Gleichung sagt also prinzipiell aus, dass der Anteil der Funktionen, die zwei Inputparameter auf denselben Wert abbildet, nicht *zu* groß ist. Es ist also ein Maß für die Qualität einer Menge von Hashfunktionen.

Satz: Seien $a, b \in \{0, 1, \cdots, N-1\}$ und sei $h_{a,b} \longmapsto ((ax+b) \bmod N) \bmod m$. Dann ist die Klasse $\mathcal{H} = \{h_{a,b} : 0 \leq a \leq N-1 \wedge 0 \leq b \leq N-1\}$ c-universell mit

$$c = \frac{\left\lceil \frac{N}{m} \right\rceil}{\frac{N}{m}} \approx 1$$

Beweis: Wird eine Übungsaufgabe.

Satz: Benutzt man Hashing mit Verkettung und wählt $n \in \mathcal{H}$ zufällig gleichverteilt, wobei $\mathcal{H}$ c-universell ist, dann ist die erwartete Suchzeit

$$\mathcal{O}(1 + c \cdot \beta)$$

für beliebige Mengen $S \subseteq U$ und $|S| = n$.

Beweis: Die Zeit für den Zugriff auf ein x ist $< 1 + \underbrace{\overbrace{\#}^{\text{Anzahl}} (y \in S \text{ mit } h(x) = h(y))}_{\text{Wird jetzt gezeigt}}$ Dazu sei

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

Dann ist

$$\frac{1}{|\mathcal{H}|} \sum_{h \in \mathcal{H}} \sum_{y \in S} \delta_h(x, y) = \frac{1}{|\mathcal{H}|} \sum_{y \in S} \sum_{h \in \mathcal{H}} \delta_h(x, y)$$

wobei

$\frac{1}{|\mathcal{H}|} \sum_{h \in \mathcal{H}}$ $\Rightarrow$ Berechnet den Erwartungswert über alle möglichen Hashfunktionen

$\sum_{h \in \mathcal{H}} \delta_h(x, y)$ $\Rightarrow$ Anzahl der Hashfunktionen, die x und y auf denselben Wert hashen:

- $\rightarrow$ Für $x = y$: $\quad |\mathcal{H}|$
- $\rightarrow$ Für $x \neq y$: $\quad \frac{c}{m} \cdot |\mathcal{H}|$, da $\mathcal{H}$ c-universell ist

Also

$$\begin{aligned} \frac{1}{|\mathcal{H}|} \sum_{h \in \mathcal{H}} \sum_{y \in S} \delta_h(x, y) &= \frac{1}{|\mathcal{H}|} \sum_{y \in S} \sum_{h \in \mathcal{H}} \delta_h(x, y) \\ &\leq \sum_{y \in S} \left[\texttt{if } (x = y) \texttt{ 1 else } \frac{c}{m} \right] \\ &\leq \begin{cases} 1 + \frac{c \cdot (n-1)}{m} & x \in S \\ \frac{c}{m} \cdot n & x \notin \end{cases} \\ &\leq 1 + c \cdot \beta \end{aligned}$$

(Besser wäre nicht nur eine erwartete sondern auch eine Worst-Case-Zugriffszeit von $\mathcal{O}(1)$, aber das kommt später). $\square$

Perfektes Hashing

Man möchte eine Hashfunktion h mit $h(x) \neq h(y)$ für alle $x \neq y \in S$ finden. Außerdem soll h eine Hashtafel der Größe $m = \mathcal{O}(|S|) = \mathcal{O}(n)$ bilden. Zusammengefasst: Man möchte eine Hashfunktion finden, die zwei unterschiedliche Inputparamter auch auf zwei Unterschiedliche Werte mappt. Außerdem soll nur linear viel Platz verbraucht werden.

Ansatz 1: Einstufiges perfektes Hashing

Die Anzahl der Kollisionen die eine Hashfunktion h für eine Schlüsselmenge S erzeugt, ist durch die folgende Funktion beschrieben:

$$c_S(h) = \left| \{x, y\} \in \binom{S}{2} : h(x) = h(y) \right|$$

Das Ziel sind 0 Kollisionen also

$$c_S(h) = 0 \quad \Leftrightarrow \quad h \underbrace{|_s}_{\substack{\text{Einge-}\\ \text{schränkt}}} \text{ injektiv}$$

Satz: Für ein zufälliges $h \in \mathcal{H}$, wobei $\mathcal{H}$ c-universell ist, gilt:

$$E(c_S(h)) \leq \binom{n}{2} \cdot \frac{c}{m}$$

Beweis: Sei

$$c_S(h) = \sum_{\{x,y\} \in \binom{S}{2}} \delta_h(x, y)$$

und weiter

$$\begin{aligned} E(c_S(h)) &= \sum_{\{x,y\} \in \binom{S}{2}} E(\delta_h(x, y)) = \sum_{\{x,y\} \in \binom{S}{2}} \Pr(\overbrace{\delta_h(x, y) = 1}^{\leq \frac{c}{m}}) \\ &\leq \binom{|S|}{2} \cdot \frac{c}{m} = \binom{n}{2} \cdot \frac{c}{m} \end{aligned}$$

□

Kollorar: Für $m > c \cdot \binom{n}{s}$ existiert ein $h \in \mathcal{H}$ mit $h|_s$ injektiv.

(Probabilistischer) Beweis: Durch Einsetzen erhält man

$$E(c_S(h)) \leq \binom{n}{2} \cdot \frac{c}{m} < 1$$

Das bedeutet, **es gibt mindestens eine Hashfunktion h mit keinen Kollisionen**. □

Probleme mit einstufigem perfekten Hashing

1. Der obige Satz ist nicht konstruktiv: Ein zufällig gewähltes $\mathcal{H}$ könnte nur mit extrem geringer Wahrscheinlichkeit injektiv sein. Das würde bedeuten, man bräuchte sehr viele Versuche, bis man ein injektives h erwischt (*Da lässt sich was machen.*).

2. Ein Platzbedarf von n^2 ist schlecht (*Dieser Nachteil bleibt bestehen*).

Kollorar zu Problem 1: Falls $m > 2 \cdot c \cdot \binom{n}{2}$, kann in erwarteter $\mathcal{O}(n+m)$ Zeit eine injektive Hashfunktion gefunden werden.

Beweis: Man erhält

$$\begin{aligned} E(c_S(h)) &\leq \binom{n}{2} \cdot \frac{c}{m} \quad < \cancel{\binom{n}{2}} \cdot \frac{\cancel{c}}{2 \cdot \cancel{c} \cdot \cancel{\binom{n}{2}}} \\ &= \frac{1}{2} \end{aligned}$$

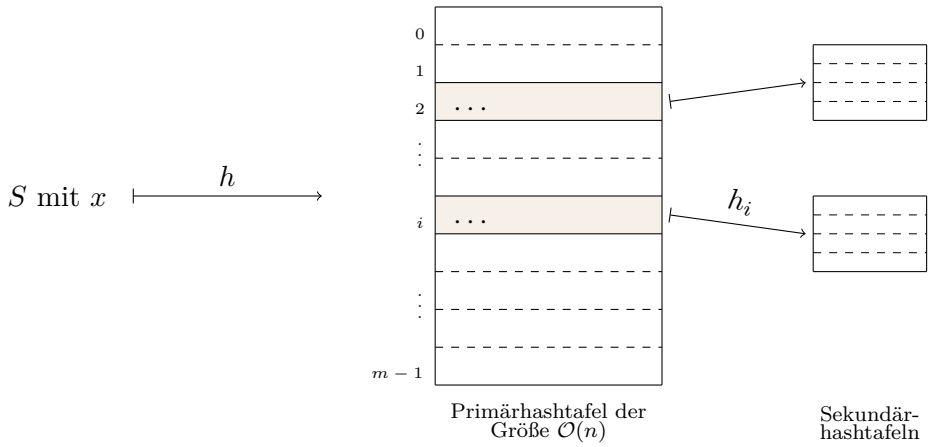
Mit der Markov-Ungleichung erhält man nun

$$\Pr(X \geq c) \leq \frac{E(X)}{c}$$
$$\Pr(c_S(h) \geq 1) \leq \frac{\frac{1}{2}}{1} = \frac{1}{2}$$

Das bedeutet, man erhält mit einer Wahrscheinlichkeit von $\geq \frac{1}{2}$ eine Hashfunktion h, die keine Kollisionen provoziert. □

Ansatz 2: Mehrstufiges perfektes Hashing

Die Idee für ein mehrstufiges perfektes Hashing ist im Folgenden aufgezeichnet:



Man wählt eine Primärhashfunktion, sodass etwa linear viele Kollisionen erzeugt werden, also $c_S(h) = \mathcal{O}(n)$. Zusätzlich erzeugt man für jeden Primärhashtafeleintrag i, für den es mehr als einen Eintrag gibt, eine Sekundärhashfunktion h_i, die injektiv in eine Sekundärhashtafel hasht.

Des Weiteren werden festgelegt:

$m =$ Größe der Primärhashtafel
$m_i =$ Größe der i-ten Sekundärhashtafel
$l_i =$ Anzahl der Elemente, die von Primärhashfunktion auf i gemappt wurden.

Für einen Primärhashtafeleintrag mit l_i Elementen wählt man $m_i > 2c \cdot \binom{l_i}{2}$. Das bedeutet, man kann h_i injektiv in $\mathcal{O}(\binom{l_i}{2})$-Zeit finden.

Der Platzverbrauch aller Sekundärhashtafeln ist

$$\begin{aligned}\sum_{i=0}^{m-1} 2 \cdot c \cdot \binom{l_i}{2} &= 2c \cdot \sum_{i=0}^{m-1} \binom{l_i}{2} \\ &= 2c \cdot \text{ Anzahl der Kollisionen von } h \\ &= 2c \cdot c_S(h)\end{aligned}$$

Außerdem fällt auf

$$\Rightarrow c_S(h) = \sum_{i=0}^{m-1} \binom{l_i}{2} \approx n$$

Für die Wahl $m = c \cdot n$ ergibt sich

$$E(c_S(n)) \leq \binom{n}{2} \cdot \frac{c}{m} = \frac{\cancel{c} \cdot (n-1) \cdot \cancel{c}}{2 \cdot \cancel{c} \cdot \cancel{c}} = \frac{n-1}{2}$$

Das heißt, der Platz aller Sekundärhashtafeln entspricht der Anzahl der Kollisionen in der Primärhashtafel.