4.2.2 Hashing mit offener Adressierung

Beispiele:

- Lineares Sondieren (linear probing)
- Quadratisches Sondieren
- Double Hashing
- Robin-Hood-Hashing
- ...

Bei dieser Methode werden die Elemente nicht in der Liste, sondern direkt in der Hash-Tabelle gespeichert. Wird bei *Insert* oder *IsElement*, angewendet auf Schlüssel k, ein Element mit Schlüssel $\neq k$ an der Adresse $h(k)$ gefunden, so wird auf deterministische Weise eine alternative Adresse berechnet. Für jeden Schlüssel $k \in U$ wird somit eine Reihenfolge (Sondierungsfolge) von Positionen in $T[\,]$ betrachtet, um k zu speichern bzw. zu finden.

Sondieren:

Sei $s(j,k) : [0..n-1] \times U \to [0..n-1]$;
Definiere $h(j,k) = (h(k) - s(j,k)) \bmod n$; $\quad (0 \leq j \leq n-1)$
Starte mit $h(0,k)$, dann, falls $h(0,k)$ belegt, $h(1,k)$, ...

Grundsätzliches Problem:
Sei $h(k) = h(k')$ für zwei Schlüssel $k, k' \in S$. Werde zunächst k eingefügt, dann k', dann k gelöscht. Wie findet man k'?
(Beachte: k' steht nicht unmittelbar an $h(k')$.)

Lösungsvorschlag: Markiere k als gelöscht, entferne es aber nicht! Wenn Speicher gebraucht wird, k überschreiben.

Beispiele für Sondierungen

Lineares Sondieren:
Setze $s(j,k) = j$ d.h. sondiere gemäß
$h(k), h(k)-1, \ldots, 0, n-1, .., h(k)+1$.
Es wird für *IsElement* solange rückwärts gesucht, bis entweder das Element mit Schlüssel k oder eine freie Position gefunden ist. Im letzteren Fall ist das gesuchte Element nicht in der Hash-Tabelle enthalten.

Problem: Es entstehen primäre Häufungen (primary clustering) um diejenigen Schlüssel herum, die beim Einfügen eine Kollision hervorgerufen haben.

Satz 26

Die durchschnittliche Anzahl der Schritte beim linearen Sondieren ist

$$\mathbb{E}[\#\ \textit{Sondierungsschritte}] = \begin{cases} \frac{1}{2}\left(1+\frac{1}{(1-\alpha)}\right) & \textit{erfolgreich} \\ \frac{1}{2}\left(1+\frac{1}{(1-\alpha)^2}\right) & \textit{erfolglos} \end{cases}$$

Einige Werte:

α	erfolgreich	erfolglos
0.5	1.5	2.5
0.9	5.5	50.5
0.95	10.5	200.5

Beispiele für Sondierungen

Quadratisches Sondieren:
Setze $s(j,k) = (-1)^j \lceil \frac{j}{2} \rceil^2$, d.h. sondiere nach $h(k), h(k)+1, h(k)-1, h(k)+4, h(k)-4, \ldots$.

Frage: Ist das überhaupt eine Permutation von $[0..n-1]$? Ist $s(j,k)$ geeignet, alle Positionen zu erreichen?

Man kann zeigen, dass für Primzahlen n von der Form $4i+3$ die Sondierungsgröße $(h(k) - s(j,k)) \bmod n$ eine Permutation von $[0..n-1]$ liefert.

Problem: Sekundäres Clustering

Satz 27

Die durchschnittliche Anzahl der Schritte bei quadratischem Sondieren ist

$$\mathbb{E}[\#\ \textit{Sondierungsschritte}] = \begin{cases} 1 + \ln\left(\frac{1}{1-\alpha}\right) - \frac{\alpha}{2} & \textit{erfolgreich} \\ \frac{1}{1-\alpha} - \alpha + \ln\left(\frac{1}{1-\alpha}\right) & \textit{erfolglos} \end{cases}$$

Einige Werte:

α	erfolgreich	erfolglos
0.5	1.44	2.19
0.9	2.85	11.4
0.95	3.52	22.05

Beispiele für Sondierungen

Double Hashing:
Setze $s(j,k) = jh'(k)$, wobei h' eine zweite Hashfunktion ist. $h'(k)$ muss relativ prim zu n gewählt werden, damit $(h(k) - s(j,k)) \bmod n$ eine Permutation der Hashadressen wird.

Satz 28

Die durchschnittliche Anzahl der Sondierungen bei Double Hashing ist

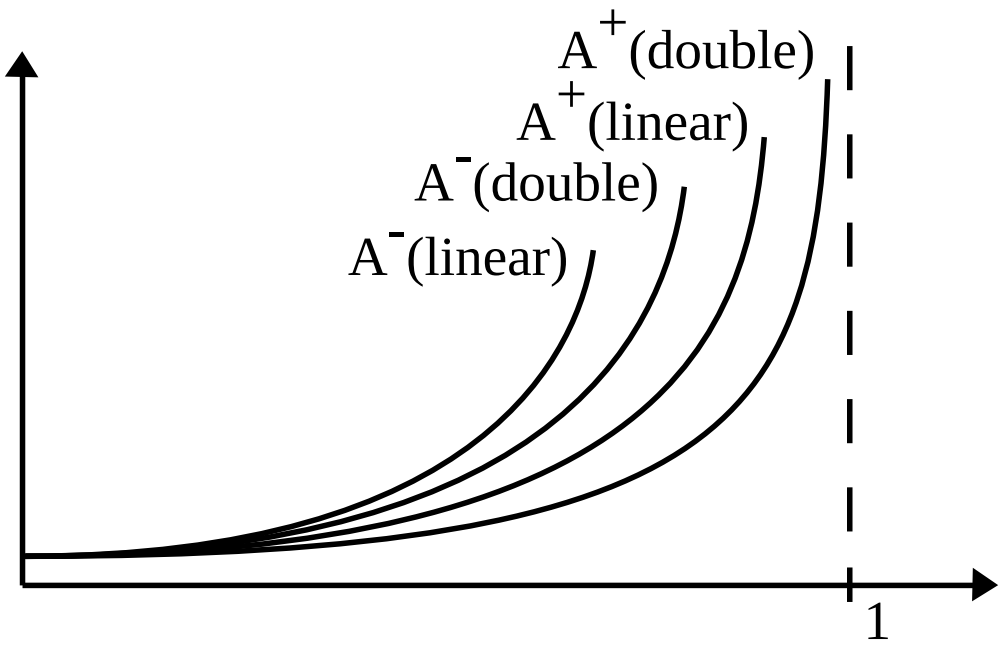
$$\mathbb{E}[\#\ \textit{Sondierungsschritte}] = \begin{cases} \frac{1}{\alpha} \ln \frac{1}{1-\alpha} & \textit{erfolgreich} \\ \frac{1}{1-\alpha} & \textit{erfolglos} \end{cases}$$

Einige Werte:

α	erfolgreich	erfolglos
0.5	1.39	2
0.9	2.55	10
0.95	3.15	20

Zum Beispiel: $h'(k) = 1 + k \bmod (n-2)$ (mit $n > 2$ prim).

Beispiele für Sondierungen



Sondierungskomplexität

4.3 Universelles Hashing

Definition 29

Eine Klasse $\mathcal{H}$ von Hashfunktionen von U nach $[0..n-1]$ heißt universell, falls für alle $x, y \in U$ mit $x \neq y$ gilt

$$\frac{|\{h \in \mathcal{H};\ h(x) = h(y)\}|}{|\mathcal{H}|} \leq \frac{1}{n}\,.$$

Satz 30

Sei $\mathcal{H}$ eine universelle Klasse von Hashfunktionen für eine Hashtabelle der Größe n und sei $h \in \mathcal{H}$ zufällig gleichverteilt gewählt. Für eine Menge S von $m \leq n$ Schlüsseln ist dann die erwartete Anzahl von Kollisionen eines festen Schlüssels $x \in S$ mit anderen Elementen aus S kleiner als 1.

Beweis:

Sei x fest. Setze

$$C_x(y) =_{\text{def}} \begin{cases} 1 & \text{falls } h(x) = h(y)\,; \\ 0 & \text{sonst.} \end{cases}$$

Dann gilt

$$\begin{aligned} \mathbb{E}[C_x(y)] &= 0 \cdot \Pr[h(x) \neq h(y)] + 1 \cdot \Pr[h(x) = h(y)] \\ &= \Pr[h(x) = h(y)] \leq \frac{1}{n}\,. \end{aligned}$$

Für $C_x =_{\text{def}} \sum C_x(y)$ folgt damit

$$\mathbb{E}[C_x] = \sum_{y \in S \setminus \{x\}} \mathbb{E}[C_x(y)] \leq \frac{m-1}{n} < 1\,.$$

□

Sei $U = \{0, 1, \ldots, n-1\}^{r+1}$, für eine Primzahl n. Definiere

$$\mathcal{H} =_{\text{def}} \{h_\alpha;\ \alpha \in U\},$$

wobei

$$h_\alpha : U \ni (x_0, x_1, \ldots, x_r) \mapsto \sum_{i=0}^{r} \alpha_i x_i \bmod n \in \{0, 1, \ldots, n-1\}.$$

Lemma 31

$\mathcal{H}$ ist universell.

Beweis:

Seien $x, y \in U$ mit $x \neq y$. Wir nehmen o.B.d.A. an, dass $x_0 \neq y_0$. Ist $h_\alpha(x) = h_\alpha(y)$ für ein $\alpha \in U$, so gilt

$$\alpha_0(y_0 - x_0) = \sum_{i=1}^{r} \alpha_i(x_i - y_i) \bmod n\,.$$

Da n prim ist, ist $\mathbb{Z}_n$ ein Körper, und es gibt, bei vorgegebenen x, y und $\alpha_1, \ldots, \alpha_r$, genau ein α, so dass $h_\alpha(x) = h_\alpha(y)$.

Für festes x und y gibt es damit genau n^r Möglichkeiten, α zu wählen, so dass $h_\alpha(x) = h_\alpha(y)$.

Damit:

$$\frac{|\{h_\alpha \in \mathcal{H};\ h_\alpha(x) = h_\alpha(y)\}|}{|\mathcal{H}|} = \frac{n^r}{n^{r+1}} = \frac{1}{n}\,.$$

□

Wie groß müssen universelle Klassen von Hashfunktionen sein?

- Aus dem Beispiel:

$$|\mathcal{H}| = n^{r+1} = |U|\,.$$

- Es gibt Konstruktionen für Klassen der Größe $n^{\log(|U|)}$ bzw. $|U|^{\log n}$.

Satz 32

Sei $\mathcal{H}$ eine universelle Klasse von Hashfunktionen $h : U \to \{0, 1, \ldots, n-1\}$. Dann gilt

$$|\mathcal{H}| \geq n \left\lfloor \frac{\log(|U|) - 1}{\log n} \right\rfloor .$$

Beweis:

Sei $\mathcal{H} = \{h_1, h_2, \ldots, h_t\}$. Betrachte die Folge $U = U_0 \supseteq U_1 \supseteq U_2 \supseteq \cdots \supseteq U_t$, die definiert ist durch

$$U_i =_{\text{def}} U_{i-1} \cap h_i^{-1}(y_i)\,,$$

wobei $y_i \in \{0, 1, \ldots, n-1\}$ so gewählt ist, dass $|U_i|$ maximiert wird. Damit gilt

- h_j ist auf U_i konstant, für $j = 1, \ldots, i$,
- $|U_i| \geq \frac{|U_{i-1}|}{n}$, d.h. $|U_i| \geq \frac{|U|}{n^i}$.

Sei nun $\bar{t} = \left\lfloor \frac{\log(|U|)-1}{\log n} \right\rfloor$. Dann folgt

$$\log|U_{\bar{t}}| \geq \log|U| - \bar{t}\log n \geq \log|U| - \left(\frac{\log(|U|)-1}{\log n}\right) \cdot \log n = 1\,.$$

Beweis:

Sei $\mathcal{H} = \{h_1, h_2, \ldots, h_t\}$. Betrachte die Folge $U = U_0 \supseteq U_1 \supseteq U_2 \supseteq \cdots \supseteq U_t$, die definiert ist durch

$$U_i =_{\text{def}} U_{i-1} \cap h_i^{-1}(y_i)\,,$$

wobei $y_i \in \{0, 1, \ldots, n-1\}$ so gewählt ist, dass $|U_i|$ maximiert wird. Damit gilt

- h_j ist auf U_i konstant, für $j = 1, \ldots, i$,
- $|U_i| \geq \frac{|U_{i-1}|}{n}$, d.h. $|U_i| \geq \frac{|U|}{n^i}$.

Seien $x, y \in U_{\bar{t}}$, $x \neq y$. Dann ist

$$\bar{t} \leq |\{h \in \mathcal{H};\ h(x) = h(y)\}| \leq |\mathcal{H}|/n$$

und damit

$$|\mathcal{H}| \geq n\bar{t} = n \left\lfloor \frac{\log(|U|) - 1}{\log n} \right\rfloor .$$

□