

14. Hashing

Hash Tabellen, Geburtstagsparadoxon, Hashfunktionen, Perfektes und universelles Hashing, Kollisionsauflösung durch Verketteten, offenes Hashing, Sondieren

[Ottman/Widmayer, Kap. 4.1-4.3.2, 4.3.4, Cormen et al, Kap. 11-11.4]

Motivation

Ziel: Tabelle aller n Studenten dieser Vorlesung

Anforderung: Schneller Zugriff per Name

Naive Ideen

Zuordnung Name $s = s_1 s_2 \dots s_{l_s}$ zu Schlüssel

$$k(s) = \sum_{i=1}^{l_s} s_i \cdot b^i$$

b gross genug, so dass verschiedene Namen verschiedene Schlüssel erhalten.

Naive Ideen

Zuordnung Name $s = s_1 s_2 \dots s_{l_s}$ zu Schlüssel

$$k(s) = \sum_{i=1}^{l_s} s_i \cdot b^i$$

b gross genug, so dass verschiedene Namen verschiedene Schlüssel erhalten.

Speichere jeden Datensatz an seinem Index in einem grossen Array.

Naive Ideen

Zuordnung Name $s = s_1 s_2 \dots s_{l_s}$ zu Schlüssel

$$k(s) = \sum_{i=1}^{l_s} s_i \cdot b^i$$

b gross genug, so dass verschiedene Namen verschiedene Schlüssel erhalten.

Speichere jeden Datensatz an seinem Index in einem grossen Array.

Beispiel, mit $b = 100$. Ascii-Werte s_i .

Anna \mapsto 71111065

Jacqueline \mapsto 102110609021813999774

Naive Ideen

Zuordnung Name $s = s_1 s_2 \dots s_{l_s}$ zu Schlüssel

$$k(s) = \sum_{i=1}^{l_s} s_i \cdot b^i$$

b gross genug, so dass verschiedene Namen verschiedene Schlüssel erhalten.

Speichere jeden Datensatz an seinem Index in einem grossen Array.

Beispiel, mit $b = 100$. Ascii-Werte s_i .

Anna \mapsto 71111065

Jacqueline \mapsto 102110609021813999774

Unrealistisch: erfordert zu grosse Arrays.

Bessere Idee?

Allokation eines Arrays der Länge m ($m > n$).

Bessere Idee?

Allokation eines Arrays der Länge m ($m > n$).

Zuordnung Name s zu

$$k_m(s) = \left(\sum_{i=1}^{l_s} s_i \cdot b^i \right) \bmod m.$$

Bessere Idee?

Allokation eines Arrays der Länge m ($m > n$).

Zuordnung Name s zu

$$k_m(s) = \left(\sum_{i=1}^{l_s} s_i \cdot b^i \right) \bmod m.$$

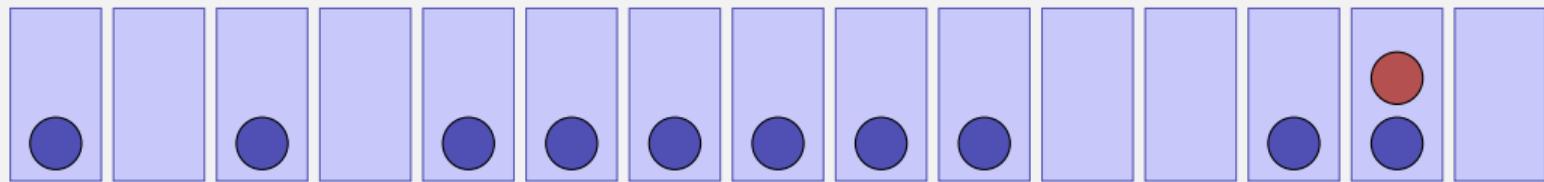
Verschiedene Namen können nun denselben Schlüssel erhalten (“Kollision”). Und dann?

Abschätzung

Vielleicht passieren Kollisionen ja fast nie. Wir schätzen ab ...

Abschätzung

Annahme: m Urnen, n Kugeln (oBdA $n \leq m$).
 n Kugeln werden gleichverteilt in Urnen gelegt.



Wie gross ist die Kollisionswahrscheinlichkeit?

Abschätzung

Annahme: m Urnen, n Kugeln (oBdA $n \leq m$).

n Kugeln werden gleichverteilt in Urnen gelegt.



Wie gross ist die Kollisionswahrscheinlichkeit?

Sehr verwandte Frage: Bei wie vielen Personen (n) ist die Wahrscheinlichkeit, dass zwei am selben Tag ($m = 365$) Geburtstag haben grösser als 50%?

Abschätzung

$$\mathbb{P}(\text{keine Kollision}) = \frac{m}{m} \cdot \frac{m-1}{m} \cdot \dots \cdot \frac{m-n+1}{m} = \frac{m!}{(m-n)! \cdot m^n}.$$

Abschätzung

$$\mathbb{P}(\text{keine Kollision}) = \frac{m}{m} \cdot \frac{m-1}{m} \cdot \dots \cdot \frac{m-n+1}{m} = \frac{m!}{(m-n)! \cdot m^n}.$$

Sei $a \ll m$. Mit $e^x = 1 + x + \frac{x^2}{2!} + \dots$ approximiere $1 - \frac{a}{m} \approx e^{-\frac{a}{m}}$.

Damit:

$$1 \cdot \left(1 - \frac{1}{m}\right) \cdot \left(1 - \frac{2}{m}\right) \cdot \dots \cdot \left(1 - \frac{n-1}{m}\right) \approx e^{-\frac{1+\dots+n-1}{m}} = e^{-\frac{n(n-1)}{2m}}.$$

Abschätzung

$$\mathbb{P}(\text{keine Kollision}) = \frac{m}{m} \cdot \frac{m-1}{m} \cdot \dots \cdot \frac{m-n+1}{m} = \frac{m!}{(m-n)! \cdot m^n}.$$

Sei $a \ll m$. Mit $e^x = 1 + x + \frac{x^2}{2!} + \dots$ approximiere $1 - \frac{a}{m} \approx e^{-\frac{a}{m}}$.

Damit:

$$1 \cdot \left(1 - \frac{1}{m}\right) \cdot \left(1 - \frac{2}{m}\right) \cdot \dots \cdot \left(1 - \frac{n-1}{m}\right) \approx e^{-\frac{1+\dots+n-1}{m}} = e^{-\frac{n(n-1)}{2m}}.$$

Es ergibt sich

$$\mathbb{P}(\text{Kollision}) = 1 - e^{-\frac{n(n-1)}{2m}}.$$

Abschätzung

$$\mathbb{P}(\text{keine Kollision}) = \frac{m}{m} \cdot \frac{m-1}{m} \cdot \dots \cdot \frac{m-n+1}{m} = \frac{m!}{(m-n)! \cdot m^n}.$$

Sei $a \ll m$. Mit $e^x = 1 + x + \frac{x^2}{2!} + \dots$ approximiere $1 - \frac{a}{m} \approx e^{-\frac{a}{m}}$.

Damit:

$$1 \cdot \left(1 - \frac{1}{m}\right) \cdot \left(1 - \frac{2}{m}\right) \cdot \dots \cdot \left(1 - \frac{n-1}{m}\right) \approx e^{-\frac{1+\dots+n-1}{m}} = e^{-\frac{n(n-1)}{2m}}.$$

Es ergibt sich

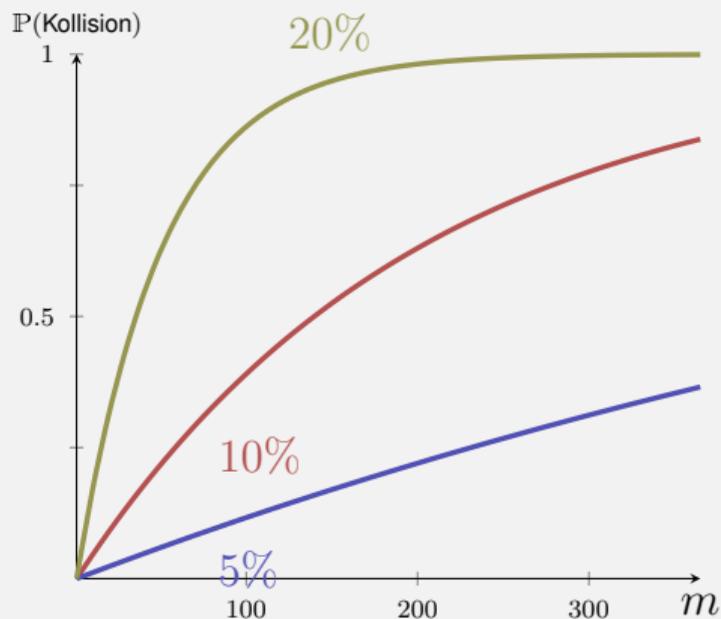
$$\mathbb{P}(\text{Kollision}) = 1 - e^{-\frac{n(n-1)}{2m}}.$$

Auflösung zum Geburtstagsparadoxon: Bei 23 Leuten ist die Wahrscheinlichkeit für Geburtstagskollision 50.7%. Zahl stammt von der leicht besseren Approximation via Stirling Formel.

Mit Füllgrad

Mit Füllgrad $\alpha := n/m$
ergibt sich (weiter vereinfacht)

$$\mathbb{P}(\text{Kollision}) \approx 1 - e^{-\alpha^2 \cdot \frac{m}{2}}.$$



Der maximale Füllgrad sollte sich an n^2/m orientieren.

Nomenklatur

Hashfunktion h : Abbildung aus der Menge der Schlüssel \mathcal{K} auf die Indexmenge $\{0, 1, \dots, m - 1\}$ eines Arrays (*Hashtabelle*).

$$h : \mathcal{K} \rightarrow \{0, 1, \dots, m - 1\}.$$

Meist $|\mathcal{K}| \gg m$. Es gibt also $k_1, k_2 \in \mathcal{K}$ mit $h(k_1) = h(k_2)$ (*Kollision*).

Eine Hashfunktion sollte die Menge der Schlüssel möglichst gleichmässig auf die Positionen der Hashtabelle verteilen.

Beispiele guter Hashfunktionen

- $h(k) = k \bmod m$, m Primzahl
- $h(k) = \lfloor m(k \cdot r - \lfloor k \cdot r \rfloor) \rfloor$, r irrational, besonders gut: $r = \frac{\sqrt{5}-1}{2}$.

Perfektes Hashing

Ist im Vorhinein die Menge der verwendeten Schlüssel bekannt?
Dann kann die Hashfunktion perfekt gewählt werden. Die praktische Konstruktion ist nicht-trivial.

Beispiel: Tabelle der Schlüsselwörter in einem Compiler.

Universelles Hashing

- $|\mathcal{K}| > m \Rightarrow$ Menge “ähnlicher Schlüssel” kann immer so gewählt sein, so dass überdurchschnittlich viele Kollisionen entstehen.
- Unmöglich, einzelne für alle Fälle “beste” Hashfunktion auszuwählen.
- Jedoch möglich¹⁵: randomisieren!

Universelle Hashklasse $\mathcal{H} \subseteq \{h : \mathcal{K} \rightarrow \{0, 1, \dots, m - 1\}\}$ ist eine Familie von Hashfunktionen, so dass

$$\forall k_1 \neq k_2 \in \mathcal{K} : |\{h \in \mathcal{H} | h(k_1) = h(k_2)\}| \leq \frac{1}{m} |\mathcal{H}|.$$

¹⁵Ähnlich wie beim Quicksort

Universelles Hashing

Theorem

Eine aus einer universellen Klasse \mathcal{H} von Hashfunktionen zufällig gewählte Funktion $h \in \mathcal{H}$ verteilt im Erwartungswert eine beliebige Folge von Schlüsseln aus \mathcal{K} so gleichmässig wie nur möglich auf die verfügbaren Plätze.

Universelles Hashing

Vorbemerkung zum Beweis des Theorems.

Definiere mit $x, y \in \mathcal{K}$, $h \in \mathcal{H}$, $Y \subseteq \mathcal{K}$:

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

$$\delta(x, Y, h) = \sum_{y \in Y} \delta(x, y, h),$$

$$\delta(x, y, \mathcal{H}) = \sum_{h \in \mathcal{H}} \delta(x, y, h).$$

\mathcal{H} ist universell, wenn für alle $x, y \in \mathcal{K}$, $x \neq y$: $\delta(x, y, \mathcal{H}) \leq |\mathcal{H}|/m$.

Universelles Hashing

Beweis des Theorems

$S \subseteq \mathcal{K}$: bereits gespeicherte Schlüssel. x wird hinzugefügt:

$$\begin{aligned}\mathbb{E}_{\mathcal{H}}(\delta(x, S, h)) &= \sum_{h \in \mathcal{H}} \delta(x, S, h) / |\mathcal{H}| \\ &= \frac{1}{|\mathcal{H}|} \sum_{h \in \mathcal{H}} \sum_{y \in S} \delta(x, y, h) = \frac{1}{|\mathcal{H}|} \sum_{y \in S} \sum_{h \in \mathcal{H}} \delta(x, y, h) \\ &= \frac{1}{|\mathcal{H}|} \sum_{y \in S} \delta(x, y, \mathcal{H}) \\ &\leq \frac{1}{|\mathcal{H}|} \sum_{y \in S} |\mathcal{H}| / m = \frac{|S|}{m}.\end{aligned}$$

Universelles Hashing ist relevant!

Sei p Primzahl und $\mathcal{K} = \{0, \dots, p-1\}$. Mit $a \in \mathcal{K} \setminus \{0\}$, $b \in \mathcal{K}$ definiere

$$h_{ab} : \mathcal{K} \rightarrow \{0, \dots, m-1\}, h_{ab}(x) = ((ax + b) \bmod p) \bmod m.$$

Dann gilt

Theorem

Die Klasse $\mathcal{H} = \{h_{ab} \mid a, b \in \mathcal{K}, a \neq 0\}$ ist eine universelle Klasse von Hashfunktionen.

Behandlung von Kollisionen

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \bmod m$. Schlüssel
12, 53, 15, 2, 19, 43

Behandlung von Kollisionen

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \bmod m$.

Schlüssel 12, 53

Verkettung der Überläufer

	0	1	2	3	4	5	6
Hashtabelle						12	

Überläufer

Behandlung von Kollisionen

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \bmod m$.

Schlüssel 12, 53, 5

Verkettung der Überläufer

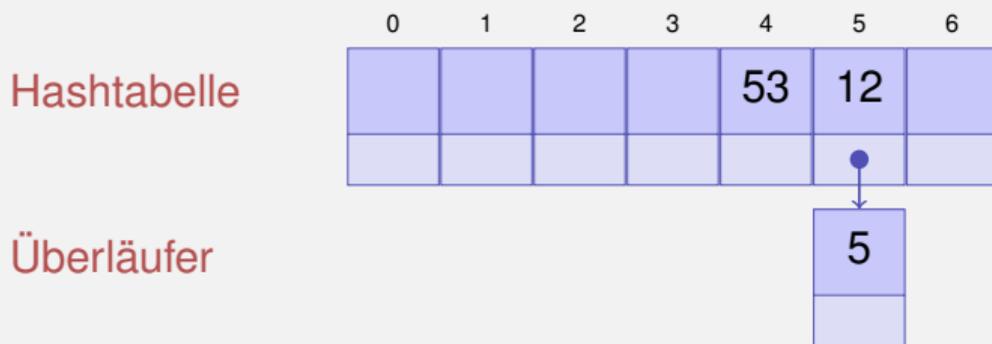
	0	1	2	3	4	5	6
Hashtabelle					53	12	
Überläufer							

Behandlung von Kollisionen

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \bmod m$.

Schlüssel 12, 53, 5, 15

Verkettung der Überläufer

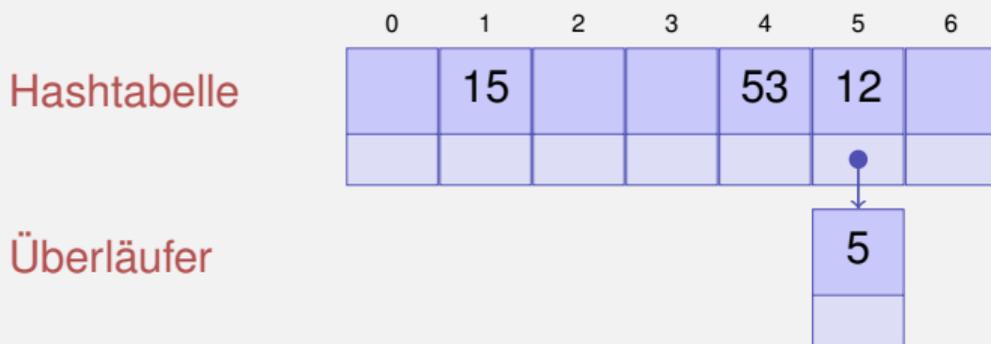


Behandlung von Kollisionen

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \bmod m$.

Schlüssel 12, 53, 5, 15, 2

Verkettung der Überläufer

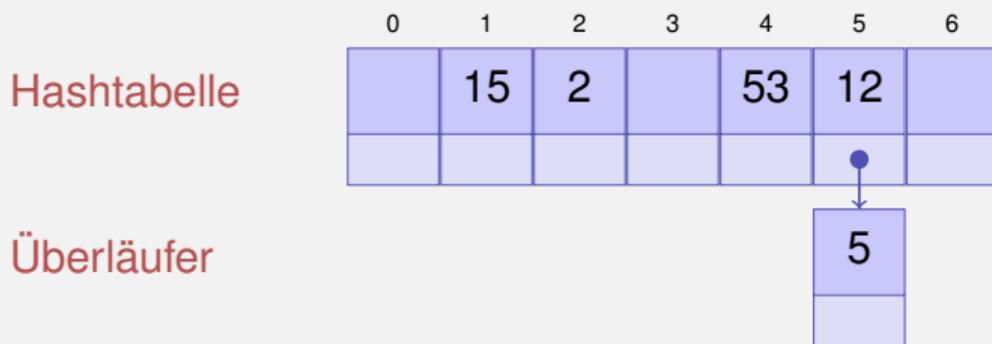


Behandlung von Kollisionen

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \bmod m$.

Schlüssel 12, 53, 5, 15, 2, 19

Verkettung der Überläufer

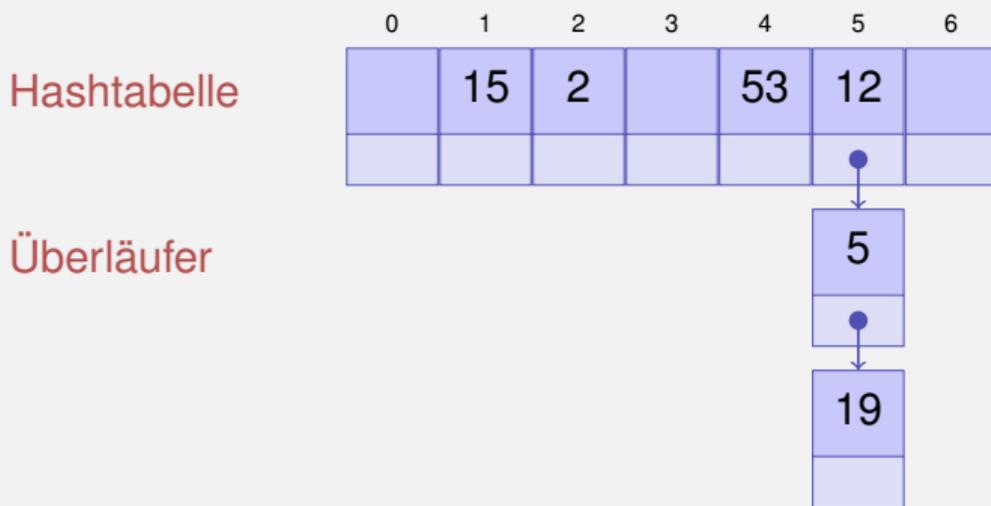


Behandlung von Kollisionen

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \bmod m$.

Schlüssel 12, 53, 5, 15, 2, 19, 43

Verkettung der Überläufer

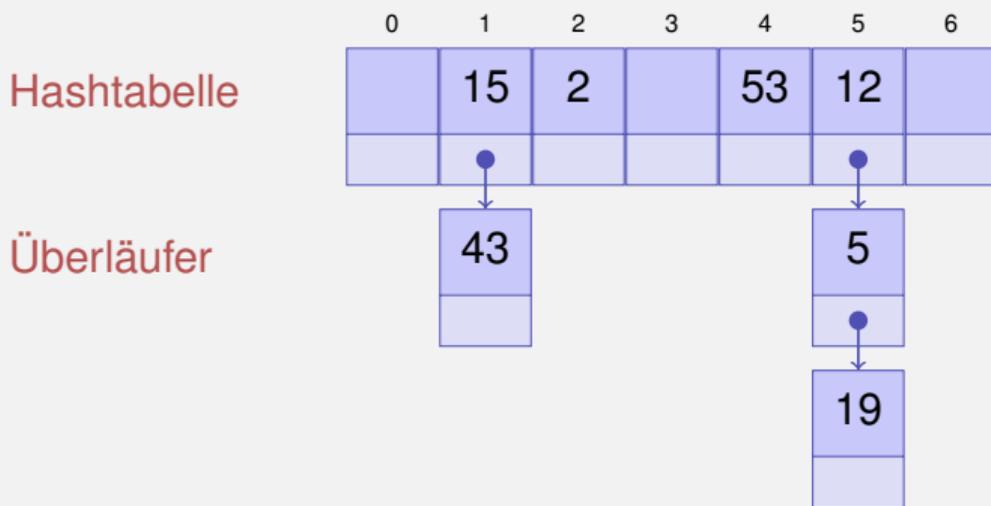


Behandlung von Kollisionen

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \bmod m$.

Schlüssel 12, 53, 5, 15, 2, 19, 43

Verkettung der Überläufer

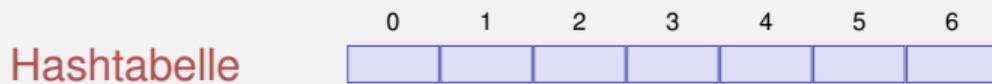


Behandlung von Kollisionen

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \bmod m$.

Schlüssel 12

Direkte Verkettung der Überläufer



Überläufer

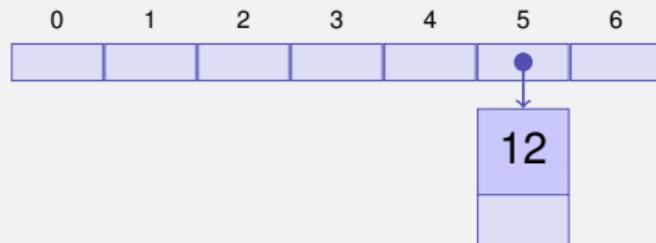
Behandlung von Kollisionen

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \bmod m$.

Schlüssel 12, 53

Direkte Verkettung der Überläufer

Hashtabelle



Überläufer

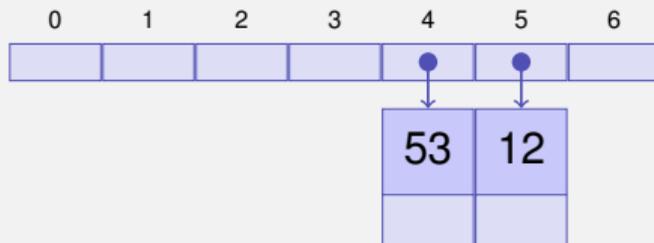
Behandlung von Kollisionen

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \bmod m$.

Schlüssel 12, 53, 5

Direkte Verkettung der Überläufer

Hashtabelle



Überläufer

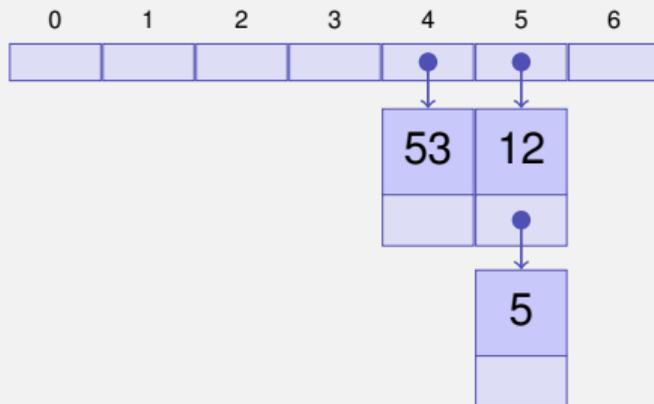
Behandlung von Kollisionen

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \bmod m$.

Schlüssel 12, 53, 5, 15

Direkte Verkettung der Überläufer

Hashtabelle



Überläufer

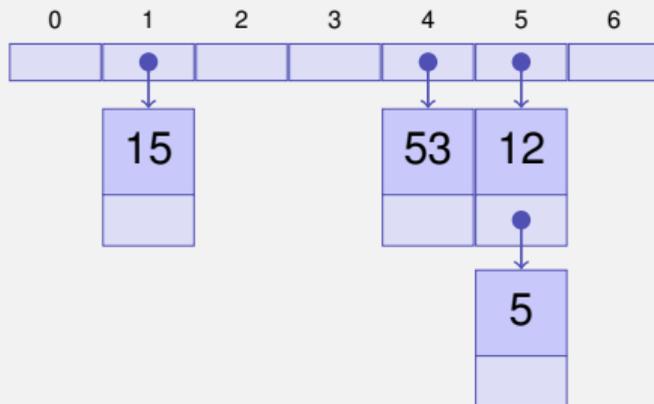
Behandlung von Kollisionen

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \bmod m$.

Schlüssel 12, 53, 5, 15, 2

Direkte Verkettung der Überläufer

Hashtabelle



Überläufer

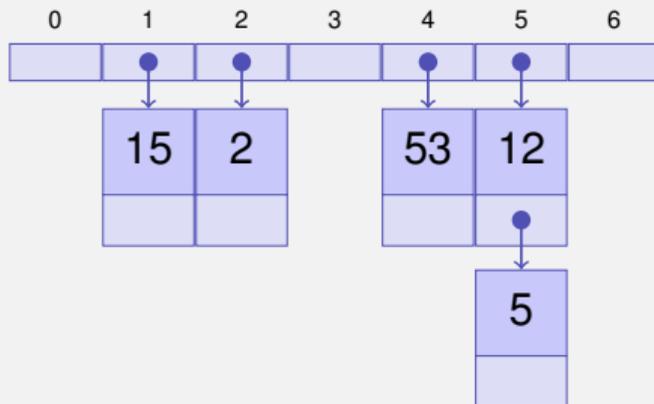
Behandlung von Kollisionen

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \bmod m$.

Schlüssel 12, 53, 5, 15, 2, 19

Direkte Verkettung der Überläufer

Hashtabelle



Überläufer

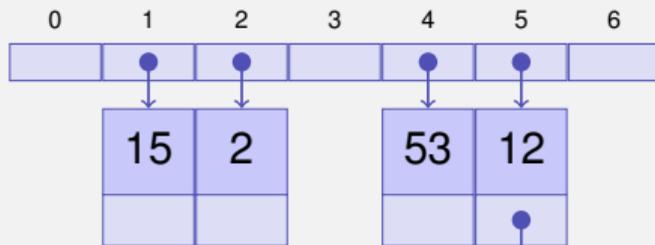
Behandlung von Kollisionen

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \bmod m$.

Schlüssel 12, 53, 5, 15, 2, 19, 43

Direkte Verkettung der Überläufer

Hashtabelle



Überläufer



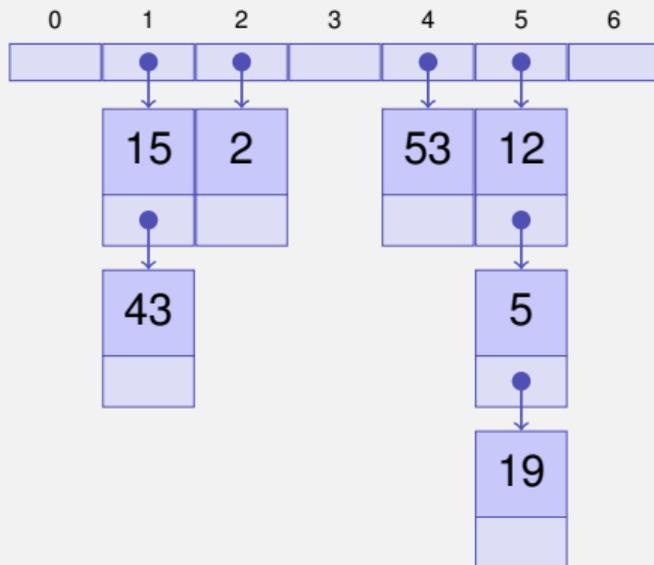
Behandlung von Kollisionen

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \bmod m$.

Schlüssel 12, 53, 5, 15, 2, 19, 43

Direkte Verkettung der Überläufer

Hashtabelle



Überläufer

Algorithmen zum Hashing mit Verkettung

- **search**(k) Durchsuche Liste an Position $h(k)$ nach k . Gib wahr zurück, wenn gefunden, sonst falsch.
- **insert**(k) Prüfe ob k in Liste an Position $h(k)$. Falls nein, füge k am Ende der Liste ein.
- **delete**(k) Durchsuche die Liste an Position $h(k)$ nach k . Wenn Suche erfolgreich, entferne das entsprechende Listenelement.

Analyse (direkt verkettete Liste)

- 1 Erfolglose Suche.

Analyse (direkt verkettete Liste)

- 1 Erfolgreiche Suche. Durchschnittliche Listenlänge ist $\alpha = \frac{n}{m}$. Liste muss ganz durchlaufen werden.

Analyse (direkt verkettete Liste)

- 1 Erfolgreiche Suche. Durchschnittliche Listenlänge ist $\alpha = \frac{n}{m}$. Liste muss ganz durchlaufen werden.

⇒ Durchschnittliche Anzahl betrachteter Einträge

$$C'_n = \alpha.$$

Analyse (direkt verkettete Liste)

- 1 Erfolgreiche Suche. Durchschnittliche Listenlänge ist $\alpha = \frac{n}{m}$. Liste muss ganz durchlaufen werden.

⇒ Durchschnittliche Anzahl betrachteter Einträge

$$C'_n = \alpha.$$

- 2 Erfolgreiche Suche. Betrachten die Einfügeschicht: Schlüssel j sieht durchschnittliche Listenlänge $(j - 1)/m$.

Analyse (direkt verkettete Liste)

- 1 Erfolgreiche Suche. Durchschnittliche Listenlänge ist $\alpha = \frac{n}{m}$. Liste muss ganz durchlaufen werden.

⇒ Durchschnittliche Anzahl betrachteter Einträge

$$C'_n = \alpha.$$

- 2 Erfolgreiche Suche. Betrachten die Einfügeschicht: Schlüssel j sieht durchschnittliche Listenlänge $(j - 1)/m$.

⇒ Durchschnittliche Anzahl betrachteter Einträge

$$C_n = \frac{1}{n} \sum_{j=1}^n (1 + (j - 1)/m)$$

Analyse (direkt verkettete Liste)

- 1 Erfolgreiche Suche. Durchschnittliche Listenlänge ist $\alpha = \frac{n}{m}$. Liste muss ganz durchlaufen werden.

⇒ Durchschnittliche Anzahl betrachteter Einträge

$$C'_n = \alpha.$$

- 2 Erfolgreiche Suche. Betrachten die Einfügeschicht: Schlüssel j sieht durchschnittliche Listenlänge $(j - 1)/m$.

⇒ Durchschnittliche Anzahl betrachteter Einträge

$$C_n = \frac{1}{n} \sum_{j=1}^n (1 + (j - 1)/m) = 1 + \frac{1}{n} \frac{n(n - 1)}{2m} .$$

Analyse (direkt verkettete Liste)

- 1 Erfolgreiche Suche. Durchschnittliche Listenlänge ist $\alpha = \frac{n}{m}$. Liste muss ganz durchlaufen werden.

⇒ Durchschnittliche Anzahl betrachteter Einträge

$$C'_n = \alpha.$$

- 2 Erfolgreiche Suche. Betrachten die Einfügeschicht: Schlüssel j sieht durchschnittliche Listenlänge $(j - 1)/m$.

⇒ Durchschnittliche Anzahl betrachteter Einträge

$$C_n = \frac{1}{n} \sum_{j=1}^n (1 + (j - 1)/m) = 1 + \frac{1}{n} \frac{n(n - 1)}{2m} \approx 1 + \frac{\alpha}{2}.$$

Vor und Nachteile

Vorteile der Strategie:

- Belegungsfaktoren $\alpha > 1$ möglich
- Entfernen von Schlüsseln einfach

Nachteile

- Speicherverbrauch der Verkettung

Offene Hashverfahren

Speichere die Überläufer direkt in der Hashtabelle mit einer *Sondierfunktion* $s(j, k)$ ($0 \leq j < m, k \in \mathcal{K}$)

Tabellenposition des Schlüssels entlang der *Sondierungsfolge*

$$S(k) := (h(k) - s(0, k) \bmod m, \dots, (h(k) - (m - 1, k)) \bmod m)$$

Algorithmen zum Open Addressing

- **search**(k) Durchlaufe Tabelleneinträge gemäss $S(k)$. Wird k gefunden, gib **true** zurück. Ist die Sondierungsfolge zu Ende oder eine leere Position erreicht, gib **false** zurück.
- **insert**(k) Suche k in der Tabelle gemäss $S(k)$. Ist k nicht vorhanden, füge k an die erste freie Position in der Sondierungsfolge ein.¹⁶
- **delete**(k) Suche k in der Tabelle gemäss $S(k)$. Wenn k gefunden, markiere Position von k mit einem **deleted**-flag.

¹⁶Als frei gilt auch eine nicht leere Position mit **deleted** flag.

Lineares Sondieren

$$s(j, k) = j \Rightarrow$$

$$S(k) = (h(k) \bmod m, (h(k) - 1) \bmod m, \dots, (h(k) + 1) \bmod m)$$

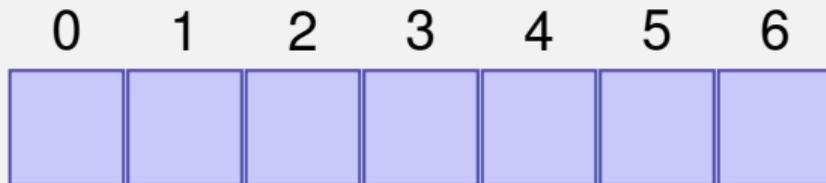
Lineares Sondieren

$$s(j, k) = j \Rightarrow$$

$$S(k) = (h(k) \bmod m, (h(k) - 1) \bmod m, \dots, (h(k) + 1) \bmod m)$$

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \bmod m$.

Schlüssel 12



Lineares Sondieren

$$s(j, k) = j \Rightarrow$$

$$S(k) = (h(k) \bmod m, (h(k) - 1) \bmod m, \dots, (h(k) + 1) \bmod m)$$

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \bmod m$.

Schlüssel 12, 53

0	1	2	3	4	5	6
					12	

Lineares Sondieren

$$s(j, k) = j \Rightarrow$$

$$S(k) = (h(k) \bmod m, (h(k) - 1) \bmod m, \dots, (h(k) + 1) \bmod m)$$

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \bmod m$.

Schlüssel 12, 53, 5

0	1	2	3	4	5	6
				53	12	

Lineares Sondieren

$$s(j, k) = j \Rightarrow$$

$$S(k) = (h(k) \bmod m, (h(k) - 1) \bmod m, \dots, (h(k) + 1) \bmod m)$$

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \bmod m$.

Schlüssel 12, 53, 5, 15

0	1	2	3	4	5	6
			5	53	12	

Lineares Sondieren

$$s(j, k) = j \Rightarrow$$

$$S(k) = (h(k) \bmod m, (h(k) - 1) \bmod m, \dots, (h(k) + 1) \bmod m)$$

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \bmod m$.

Schlüssel 12, 53, 5, 15, 2

0	1	2	3	4	5	6
	15		5	53	12	

Lineares Sondieren

$$s(j, k) = j \Rightarrow$$

$$S(k) = (h(k) \bmod m, (h(k) - 1) \bmod m, \dots, (h(k) + 1) \bmod m)$$

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \bmod m$.

Schlüssel 12, 53, 5, 15, 2, 19

0	1	2	3	4	5	6
	15	2	5	53	12	

Lineares Sondieren

$$s(j, k) = j \Rightarrow$$

$$S(k) = (h(k) \bmod m, (h(k) - 1) \bmod m, \dots, (h(k) + 1) \bmod m)$$

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \bmod m$.

Schlüssel 12, 53, 5, 15, 2, 19

0	1	2	3	4	5	6
19	15	2	5	53	12	

Analyse Lineares Sondieren (ohne Herleitung)

Analyse Lineares Sondieren (ohne Herleitung)

- 1 Erfolglose Suche. Durchschnittliche Anzahl betrachteter Einträge

$$C'_n \approx \frac{1}{2} \left(1 + \frac{1}{(1 - \alpha)^2} \right)$$

Analyse Lineares Sondieren (ohne Herleitung)

- 1 Erfolglose Suche. Durchschnittliche Anzahl betrachteter Einträge

$$C'_n \approx \frac{1}{2} \left(1 + \frac{1}{(1 - \alpha)^2} \right)$$

- 2 Erfolgreiche Suche. Durchschnittliche Anzahl betrachteter Einträge

$$C_n \approx \frac{1}{2} \left(1 + \frac{1}{1 - \alpha} \right).$$

Diskussion

Diskussion

Beispiel $\alpha = 0.95$

Erfolglose Suche betrachtet im Durchschnitt 200 Tabelleneinträge!

Diskussion

Beispiel $\alpha = 0.95$

Erfolglose Suche betrachtet im Durchschnitt 200 Tabelleneinträge!

❓ Nachteile des Verfahrens?

Diskussion

Beispiel $\alpha = 0.95$

Erfolglose Suche betrachtet im Durchschnitt 200 Tabelleneinträge!

❓ Nachteile des Verfahrens?

❗ *Primäre Häufung*: Ähnliche Hashadressen haben ähnliche Sondierungsfolgen \Rightarrow lange zusammenhängende belegte Bereiche.

Quadratisches Sondieren

$$s(j, k) = \lceil j/2 \rceil^2 (-1)^j$$

$$S(k) = (h(k) + 1, h(k) - 1, h(k) + 4, h(k) - 4, \dots) \pmod m$$

Quadratisches Sondieren

$$s(j, k) = \lceil j/2 \rceil^2 (-1)^j$$

$$S(k) = (h(k) + 1, h(k) - 1, h(k) + 4, h(k) - 4, \dots) \pmod m$$

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \pmod m$.

Schlüssel 12

0	1	2	3	4	5	6

Quadratisches Sondieren

$$s(j, k) = \lceil j/2 \rceil^2 (-1)^j$$

$$S(k) = (h(k) + 1, h(k) - 1, h(k) + 4, h(k) - 4, \dots) \pmod m$$

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \pmod m$.

Schlüssel 12, 53

0	1	2	3	4	5	6
					12	

Quadratisches Sondieren

$$s(j, k) = \lceil j/2 \rceil^2 (-1)^j$$

$$S(k) = (h(k) + 1, h(k) - 1, h(k) + 4, h(k) - 4, \dots) \pmod m$$

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \pmod m$.

Schlüssel 12, 53, 5

0	1	2	3	4	5	6
				53	12	

Quadratisches Sondieren

$$s(j, k) = \lceil j/2 \rceil^2 (-1)^j$$

$$S(k) = (h(k) + 1, h(k) - 1, h(k) + 4, h(k) - 4, \dots) \pmod m$$

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \pmod m$.

Schlüssel 12, 53, 5, 15

0	1	2	3	4	5	6
				53	12	5

Quadratisches Sondieren

$$s(j, k) = \lceil j/2 \rceil^2 (-1)^j$$

$$S(k) = (h(k) + 1, h(k) - 1, h(k) + 4, h(k) - 4, \dots) \pmod m$$

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \pmod m$.

Schlüssel 12, 53, 5, 15, 2

0	1	2	3	4	5	6
	15			53	12	5

Quadratisches Sondieren

$$s(j, k) = \lceil j/2 \rceil^2 (-1)^j$$

$$S(k) = (h(k) + 1, h(k) - 1, h(k) + 4, h(k) - 4, \dots) \pmod m$$

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \pmod m$.

Schlüssel 12, 53, 5, 15, 2, 19

0	1	2	3	4	5	6
	15	2		53	12	5

Quadratisches Sondieren

$$s(j, k) = \lceil j/2 \rceil^2 (-1)^j$$

$$S(k) = (h(k) + 1, h(k) - 1, h(k) + 4, h(k) - 4, \dots) \pmod m$$

Beispiel $m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \pmod m$.

Schlüssel 12, 53, 5, 15, 2, 19

0	1	2	3	4	5	6
19	15	2		53	12	5
.

Analyse Quadratisches Sondieren (ohne Herleitung)

Analyse Quadratisches Sondieren (ohne Herleitung)

- 1 Erfolglose Suche. Durchschnittliche Anzahl betrachteter Einträge

$$C'_n \approx \frac{1}{1-\alpha} - \alpha + \ln\left(\frac{1}{1-\alpha}\right)$$

Analyse Quadratisches Sondieren (ohne Herleitung)

- 1 Erfolglose Suche. Durchschnittliche Anzahl betrachteter Einträge

$$C'_n \approx \frac{1}{1-\alpha} - \alpha + \ln \left(\frac{1}{1-\alpha} \right)$$

- 2 Erfolgreiche Suche. Durchschnittliche Anzahl betrachteter Einträge

$$C_n \approx 1 + \ln \left(\frac{1}{1-\alpha} \right) - \frac{\alpha}{2}.$$

Diskussion

Beispiel $\alpha = 0.95$

Erfolglose Suche betrachtet im Durchschnitt 22 Tabelleneinträge

Diskussion

Beispiel $\alpha = 0.95$

Erfolglose Suche betrachtet im Durchschnitt 22 Tabelleneinträge

⊗ Nachteile des Verfahrens?

Diskussion

Beispiel $\alpha = 0.95$

Erfolgreiche Suche betrachtet im Durchschnitt 22 Tabelleneinträge

❓ Nachteile des Verfahrens?

❗ **Sekundäre Häufung:** Synonyme k und k' (mit $h(k) = h(k')$) durchlaufen dieselbe Sondierungsfolge.

Double Hashing

Zwei Hashfunktionen $h(k)$ und $h'(k)$. $s(j, k) = j \cdot h'(k)$.

$S(k) = (h(k) - h'(k), h(k) - 2h'(k), \dots, h(k) - (m - 1)h'(k)) \pmod m$

Double Hashing

Zwei Hashfunktionen $h(k)$ und $h'(k)$. $s(j, k) = j \cdot h'(k)$.

$$S(k) = (h(k) - h'(k), h(k) - 2h'(k), \dots, h(k) - (m - 1)h'(k)) \pmod m$$

Beispiel:

$m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \pmod 7$, $h'(k) = 1 + k \pmod 5$.

Schlüssel 12

0	1	2	3	4	5	6

Double Hashing

Zwei Hashfunktionen $h(k)$ und $h'(k)$. $s(j, k) = j \cdot h'(k)$.

$$S(k) = (h(k) - h'(k), h(k) - 2h'(k), \dots, h(k) - (m-1)h'(k)) \pmod{m}$$

Beispiel:

$m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \pmod{7}$, $h'(k) = 1 + k \pmod{5}$.

Schlüssel 12, 53

0	1	2	3	4	5	6
					12	

Double Hashing

Zwei Hashfunktionen $h(k)$ und $h'(k)$. $s(j, k) = j \cdot h'(k)$.

$$S(k) = (h(k) - h'(k), h(k) - 2h'(k), \dots, h(k) - (m-1)h'(k)) \pmod{m}$$

Beispiel:

$m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \pmod{7}$, $h'(k) = 1 + k \pmod{5}$.

Schlüssel 12, 53, 5

0	1	2	3	4	5	6
				53	12	

Double Hashing

Zwei Hashfunktionen $h(k)$ und $h'(k)$. $s(j, k) = j \cdot h'(k)$.

$$S(k) = (h(k) - h'(k), h(k) - 2h'(k), \dots, h(k) - (m-1)h'(k)) \pmod m$$

Beispiel:

$m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \pmod 7$, $h'(k) = 1 + k \pmod 5$.

Schlüssel 12, 53, 5, 15

0	1	2	3	4	5	6
			5	53	12	

Double Hashing

Zwei Hashfunktionen $h(k)$ und $h'(k)$. $s(j, k) = j \cdot h'(k)$.

$$S(k) = (h(k) - h'(k), h(k) - 2h'(k), \dots, h(k) - (m - 1)h'(k)) \pmod{m}$$

Beispiel:

$m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \pmod{7}$, $h'(k) = 1 + k \pmod{5}$.

Schlüssel 12, 53, 5, 15, 2

0	1	2	3	4	5	6
	15		5	53	12	

Double Hashing

Zwei Hashfunktionen $h(k)$ und $h'(k)$. $s(j, k) = j \cdot h'(k)$.

$S(k) = (h(k) - h'(k), h(k) - 2h'(k), \dots, h(k) - (m - 1)h'(k)) \pmod m$

Beispiel:

$m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \pmod 7$, $h'(k) = 1 + k \pmod 5$.

Schlüssel 12, 53, 5, 15, 2, 19

0	1	2	3	4	5	6
	15	2	5	53	12	

Double Hashing

Zwei Hashfunktionen $h(k)$ und $h'(k)$. $s(j, k) = j \cdot h'(k)$.

$$S(k) = (h(k) - h'(k), h(k) - 2h'(k), \dots, h(k) - (m - 1)h'(k)) \pmod{m}$$

Beispiel:

$m = 7$, $\mathcal{K} = \{0, \dots, 500\}$, $h(k) = k \pmod{7}$, $h'(k) = 1 + k \pmod{5}$.

Schlüssel 12, 53, 5, 15, 2, 19

0	1	2	3	4	5	6
19	15	2	5	53	12	

Double Hashing

- Sondierungsreihenfolge muss Permutation aller Hashadressen bilden. Also $h'(k) \neq 0$ und $h'(k)$ darf m nicht teilen, z.B. garantiert mit m prim.
- h' sollte unabhängig von h sein (Vermeidung sekundärer Häufung).

Unabhängigkeit:

$$\mathbb{P}((h(k) = h(k')) \wedge (h'(k) = h'(k'))) = \mathbb{P}(h(k) = h(k')) \cdot \mathbb{P}(h'(k) = h'(k')).$$

Unabhängigkeit erfüllt von $h(k) = k \bmod m$ und $h'(k) = 1 + k \bmod (m - 2)$ (m prim).

Analyse Double Hashing

Sind h und h' unabhängig, dann:

Analyse Double Hashing

Sind h und h' unabhängig, dann:

- 1 Erfolglose Suche. Durchschnittliche Anzahl betrachteter Einträge

$$C'_n \approx \frac{1}{1 - \alpha}$$

Analyse Double Hashing

Sind h und h' unabhängig, dann:

- 1 Erfolglose Suche. Durchschnittliche Anzahl betrachteter Einträge

$$C'_n \approx \frac{1}{1 - \alpha}$$

- 2 Erfolgreiche Suche. Durchschnittliche Anzahl betrachteter Einträge

$$C_n \approx 1 + \frac{\alpha}{2} + \frac{\alpha^3}{4} + \frac{\alpha^4}{15} - \frac{\alpha^5}{18} + \dots < 2.5$$

Übersicht

	$\alpha = 0.50$		$\alpha = 0.90$		$\alpha = 0.95$	
	C_n	C'_n	C_n	C'_n	C_n	C'_n
Separate Verkettung	1.250	1.110	1.450	1.307	1.475	1.337
Direkte Verkettung	1.250	0.500	1.450	0.900	1.475	0.950
Lineares Sondieren	1.500	2.500	5.500	50.500	10.500	200.500
Quadratisches Sondieren	1.440	2.190	2.850	11.400	3.520	22.050
Double Hashing	1.39	2.000	2.560	10.000	3.150	20.000