Hashing

- Wörterbücher
- Hashing mit Verkettung
- Lineares Sondieren
- Hashfunktionen

Holger Dell
Folien adaptiert von Philip Bille

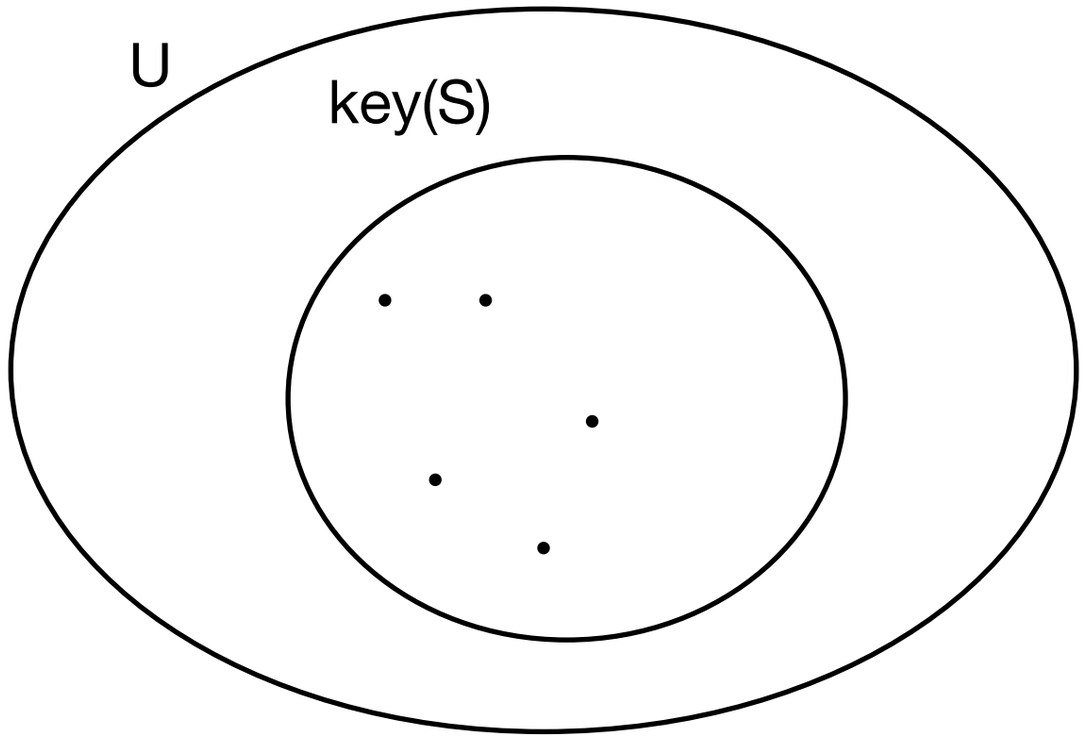
Hashing

- Wörterbücher
- Hashing mit Verkettung
- Lineares Sondieren
- Hashfunktionen

Wörterbücher

- Wörterbücher.
  - Verwaltet eine dynamische Menge S von n Elementen.
  - Jedes Element x hat einen eindeutigen Schlüssel x.key aus einem Universum U und dazugehörige Daten x.data.
  - Unterstützt folgende Operationen:
    - SEARCH(k): stelle fest, ob ein Element mit Schlüssel k existiert. Liefere es.
    - INSERT(x): füge x zu S hinzu (wir nehmen hierbei an, dass x noch nicht in S ist).
    - DELETE(x): entferne x aus S.

- U = {0, ..., 99}
- key(S) = {1, 13, 16, 41, 54, 66, 96}



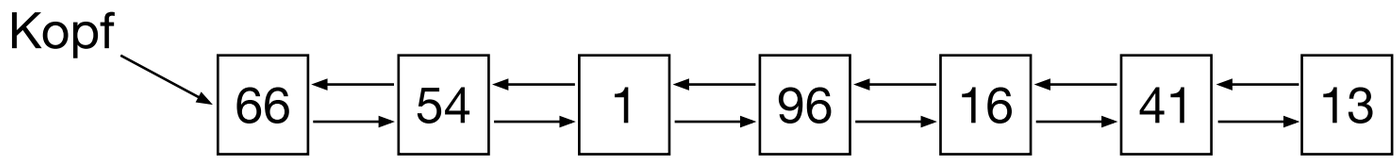
Wörterbücher

- Anwendungen.
  - Abstrakte Datenstruktur, um Mengen darzustellen.
  - Wird in zahlreichen Algorithmen, Datenstrukturen, Datenbanken, etc. verwendet.

- Frage.
  Wie können wir Wörterbücher mit uns bereits bekannten Techniken darstellen?

Wörterbücher

- Lösung 1: Verwalte S als doppelt verkettete Liste.



- SEARCH(k): lineare Suche nach dem Schlüssel k.
- INSERT(x): füge x vorne in die Liste ein.
- DELETE(x): entferne x aus der Liste.

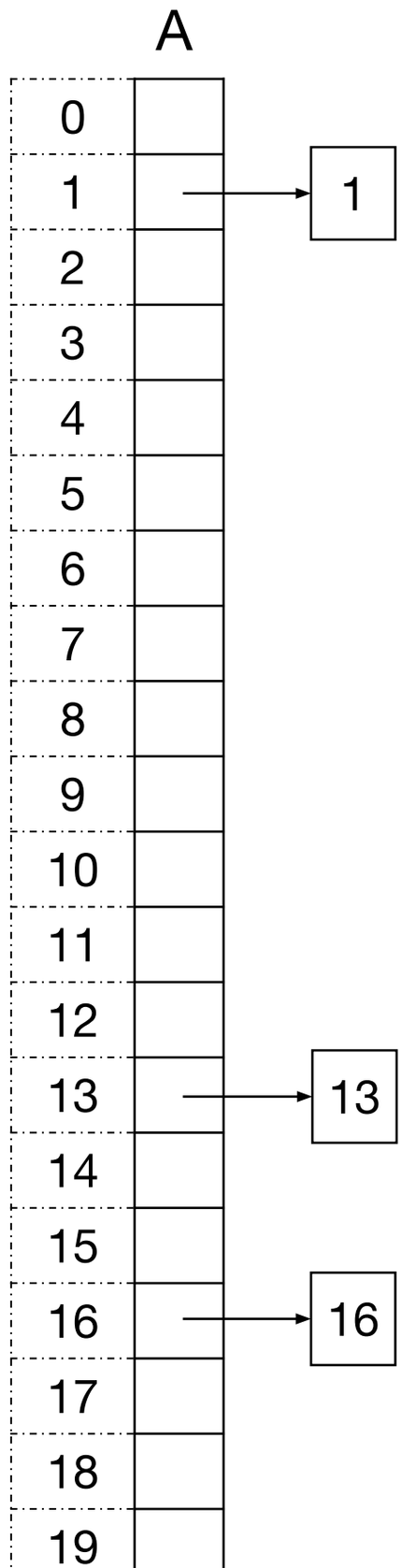
- Zeit.
  - SEARCH in Zeit O(n).
  - INSERT und DELETE in Zeit O(1).
- Platz.
  - O(n).

Wörterbücher

- Lösung 2: Direkter Zugriff.
  - Verwalte S in Feld A von Größe |U|.
  - Speichere Element x in A[x.key].

- SEARCH(k): liefere A[k] zurück.
- INSERT(x): Setze A[x.key] = x.
- DELETE(x): Setze A[x.key] = null.

- Time.
  - SEARCH, INSERT and DELETE in O(1) time.
- Space.
  - O(|U|)



Wörterbücher

Datenstruktur	SEARCH	INSERT	DELETE	space
verkettete Liste	O(n)	O(1)	O(1)	O(n)
direkter Zugriff	O(1)	O(1)	O(1)	O(|U|)

- Frage. Können wir Wörterbücher signifikant besser implementieren?

Hashing

- Wörterbücher
- Hashing mit Verkettung
- Lineares Sondieren
- Hashfunktionen

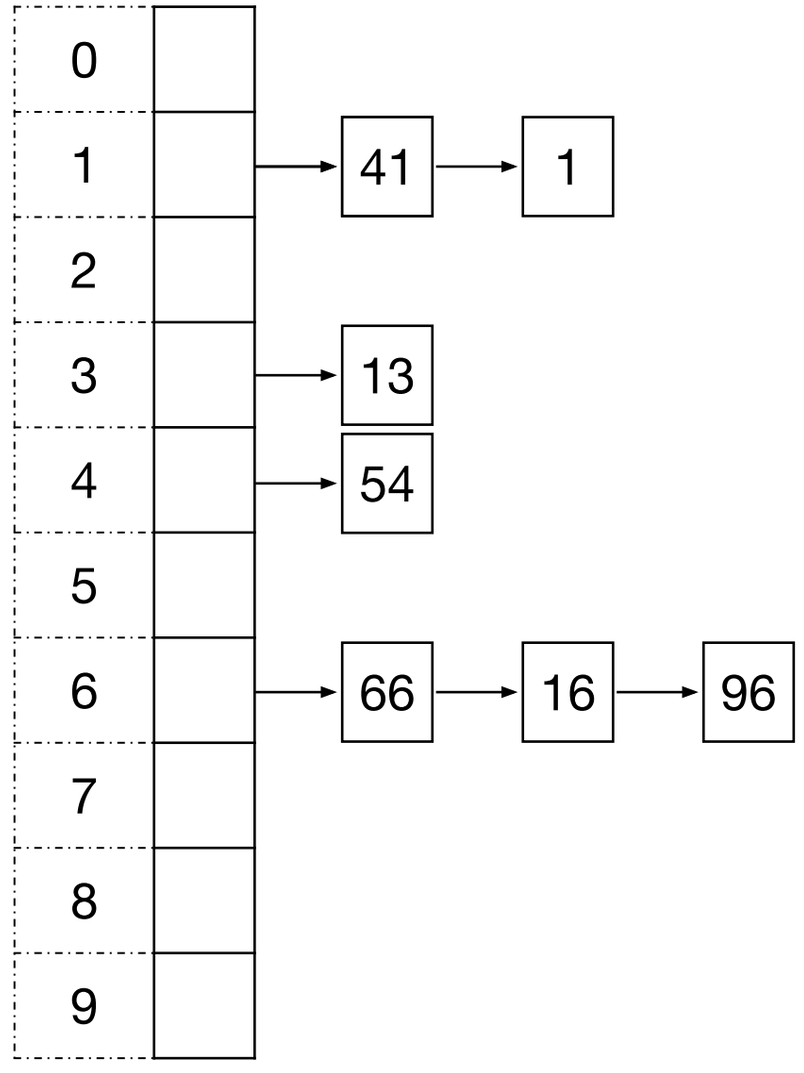
Hashing mit Verkettung

- Idee. Finde eine Hashfunktion $h : U \to \{0, ..., m-1\}$ mit $m = \Theta(n)$.
  Hashfunktionen sollen Schlüssel aus S gleichmäßig über $\{0, ..., m-1\}$ verteilen.

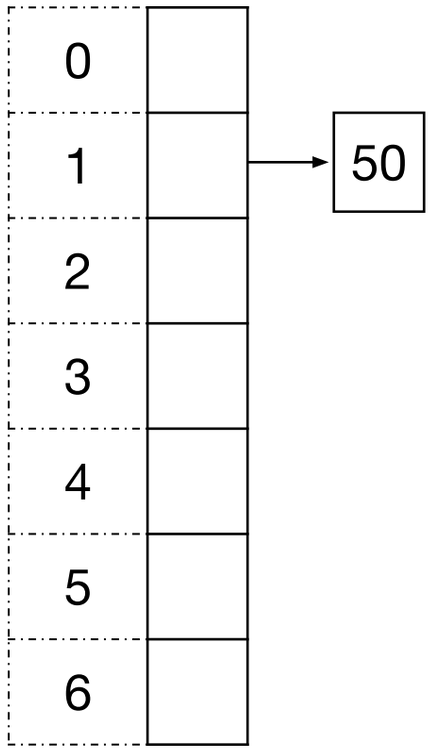
- Hashing mit Verkettung.
  - Verwalte ein Feld A[0..m-1]
    von doppelt verketteten Listen.
  - Speichere Element x in Liste A[h(x.key)].

- Kollision.
  - x und y kollidieren wenn h(x.key) = h(y.key).

- SEARCH(k): lineare Suche in A[h(k)] nach Schlüssel k.
- INSERT(x): füge x vorne in Liste A[h(x.key)] ein.
- DELETE(x): lösche x aus der Liste A[h(x.key)].

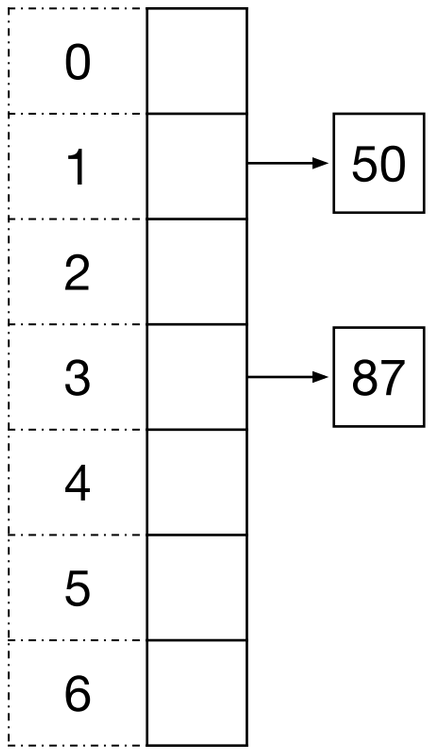


k = 50



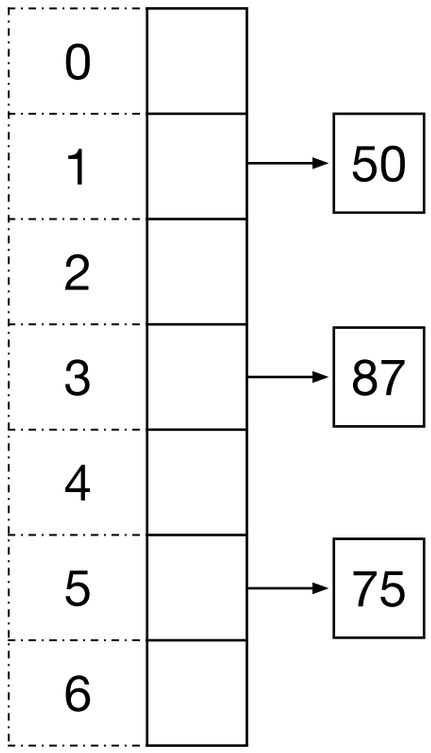
h(k) = k mod 7

k = 87



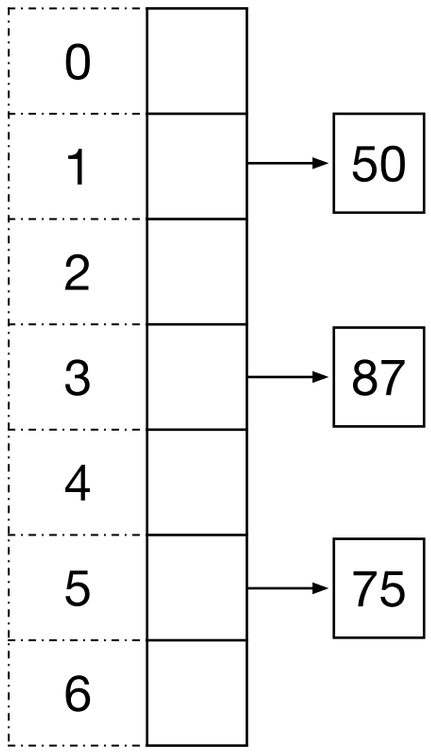
h(k) = k mod 7

k = 75



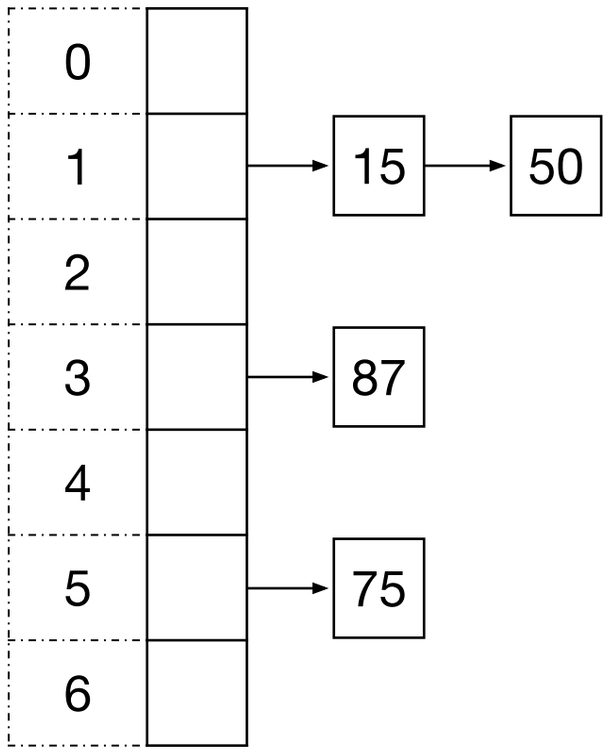
h(k) = k mod 7

k = 15



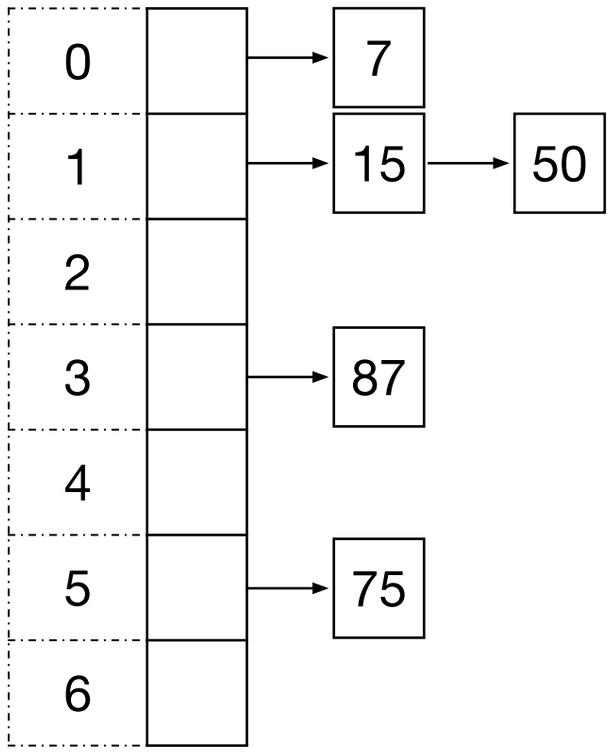
h(k) = k mod 7

k = 15



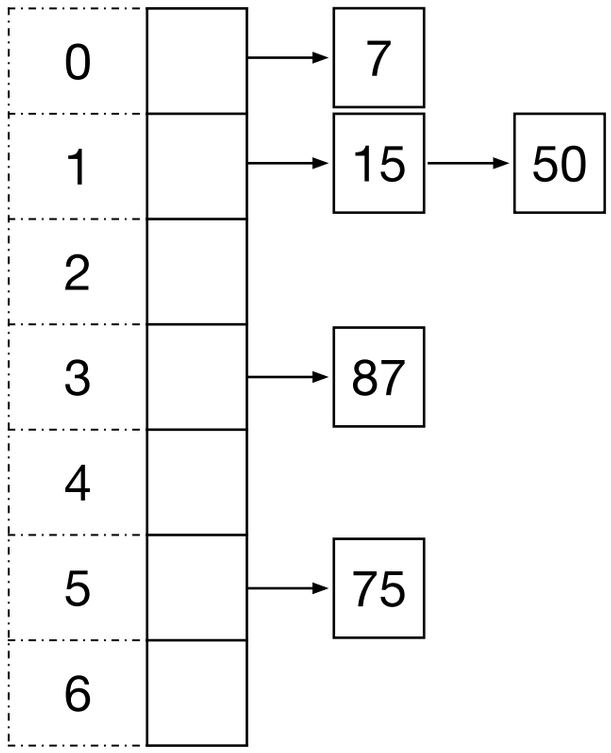
h(k) = k mod 7

k = 7



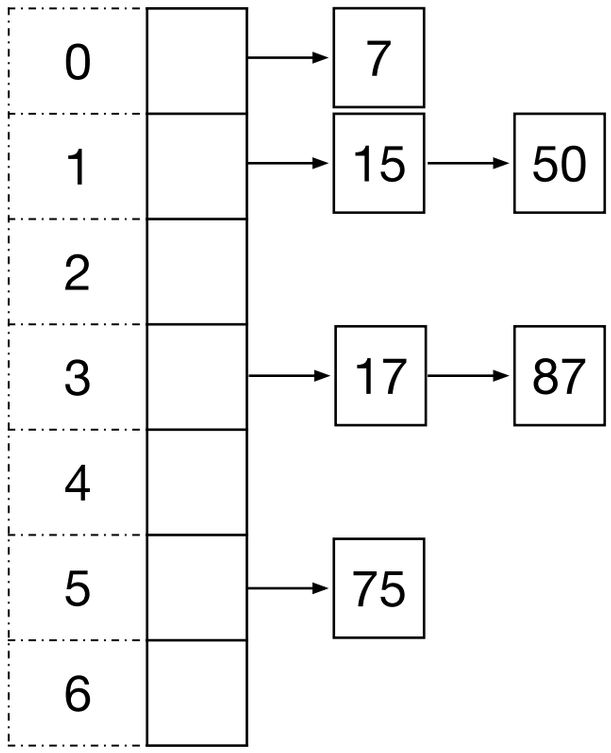
h(k) = k mod 7

k = 17



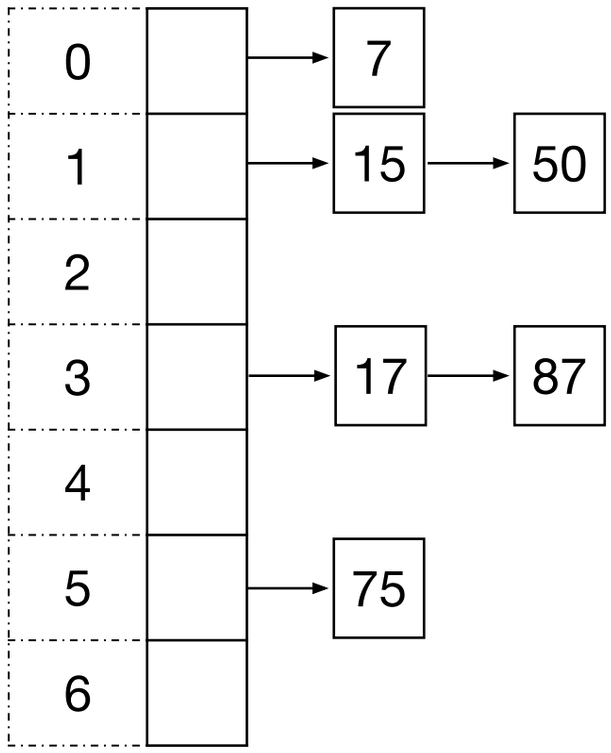
h(k) = k mod 7

k = 17



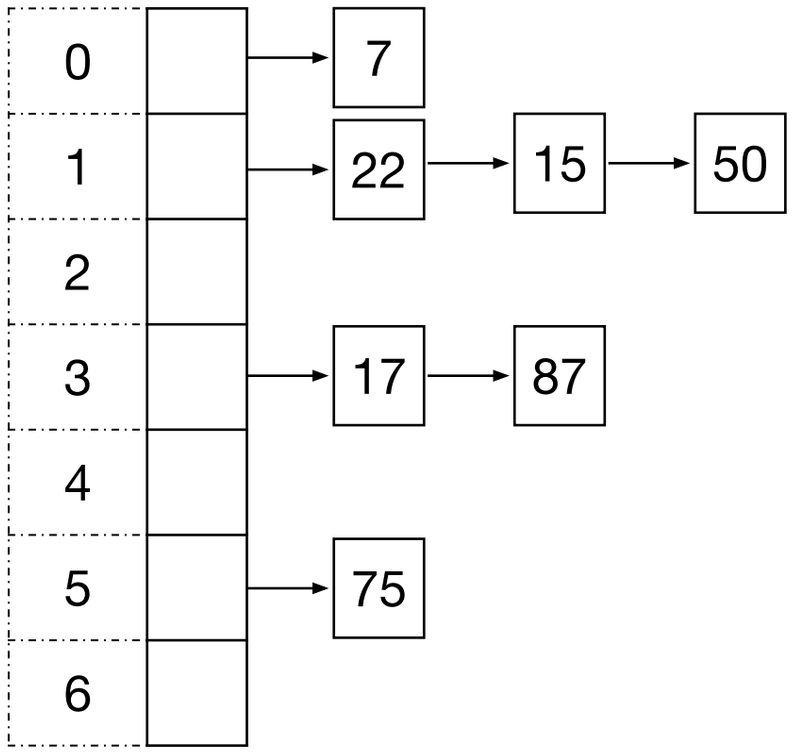
h(k) = k mod 7

k = 22



h(k) = k mod 7

k = 22



h(k) = k mod 7

Hashing mit Verkettung

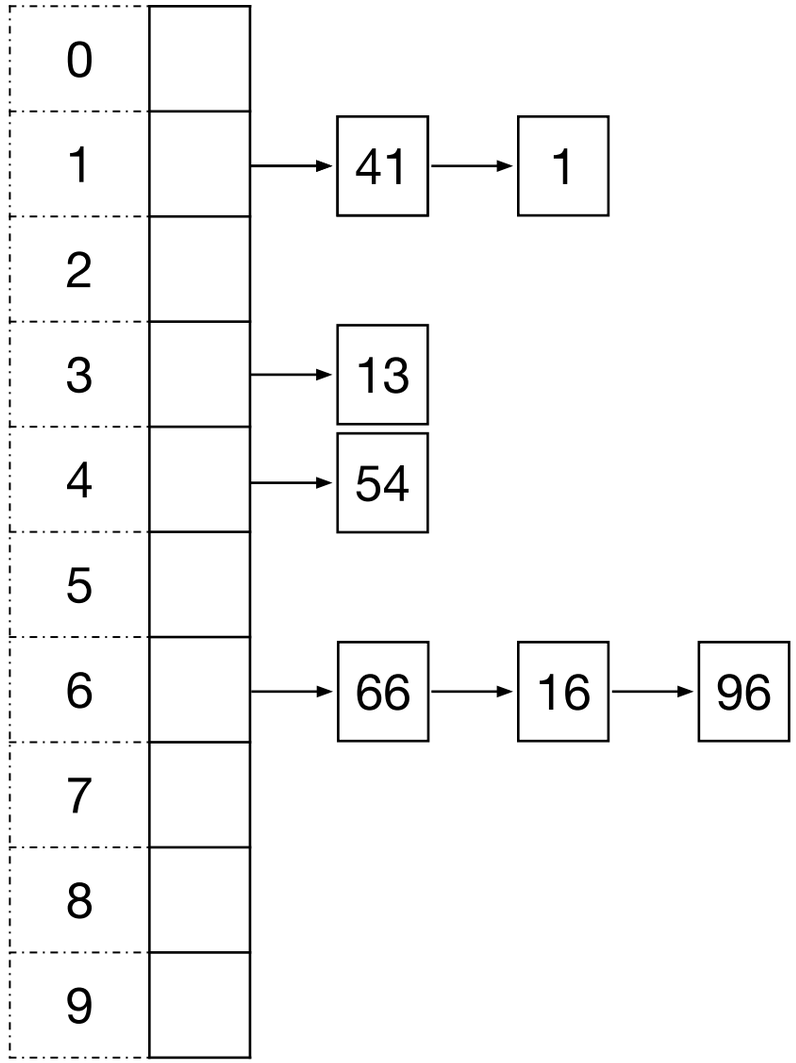
- SEARCH(k): Lineare Suche in A[h(k)] nach Schlüssel k.
- INSERT(x): füge x vorne in Liste A[h(x.key)] ein.
- DELETE(x): lösche x aus der Liste A[h(x.key)].

- Übung. Füge die Sequenz k = 5, 28, 19, 15, 20, 33, 12, 17, 10 von Schlüsseln in eine anfangs leere Hashtabelle der Größe 9 ein, die Hashing mit Verkettung und die Hashfunktion h(k) = k mod 9 benutzt.

Hashing mit Verkettung

- SEARCH(k): Lineare Suche in A[h(k)] nach Schlüssel k.
- INSERT(x): füge x vorne in Liste A[h(x.key)] ein.
- DELETE(x): lösche x aus der Liste A[h(x.key)].

- Zeit.
  - SEARCH in Zeit O(Länge der Liste).
  - INSERT und DELETE in Zeit O(1).
  - Länge der Listen hängt von der Hashfunktion ab.
- Platz.
  - O(m + n) = O(n).

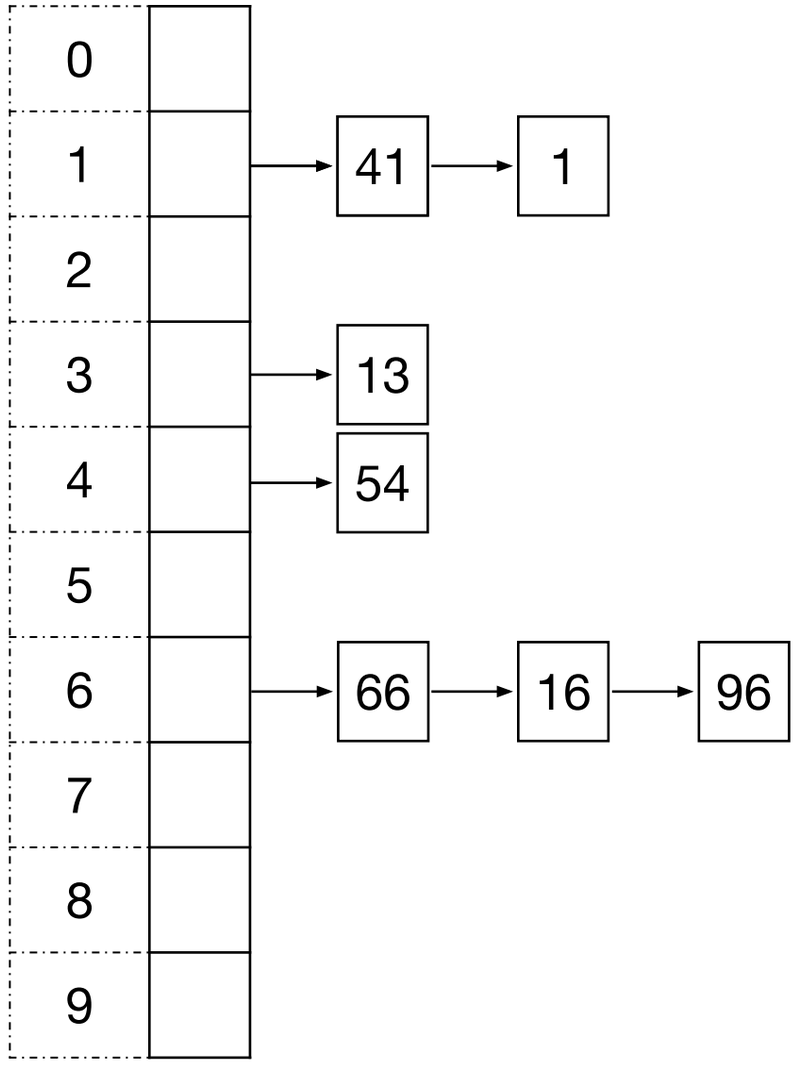


Hashing mit Verkettung

- Definition. Belegungsfaktor α:
  α = durchschnittliche Listenlänge = n/m = O(1)

- Einfaches gleichmäßiges Hashing. Die Annahme, dass alle Schlüssel k unabhängig voneinander einen uniform zufälligen Hashwert h(k) in {0, ..., m-1} erhalten.
  - Erwartete Listenlänge = α.
  - ⇒ erwartete Zeit für SEARCH ist O(1).

- Zeit (unter einfachem gleichmäßigem Hashing).
  - SEARCH in erwarteter Zeit O(1).
  - INSERT und DELETE in Zeit O(1).



Wörterbücher

Datenstruktur	SEARCH	INSERT	DELETE	space
verkettete Liste	O(n)	O(1)	O(1)	O(n)
direkter Zugriff	O(1)	O(1)	O(1)	O(|U|)
Hashing mit Verkettung	O(1)†	O(1)	O(1)	O(n)

† = erwartete Zeit unter einfachem gleichmäßigem Hashing

Hashing

- Wörterbücher
- Hashing mit Verkettung
- Lineares Sondieren
- Hashfunktionen

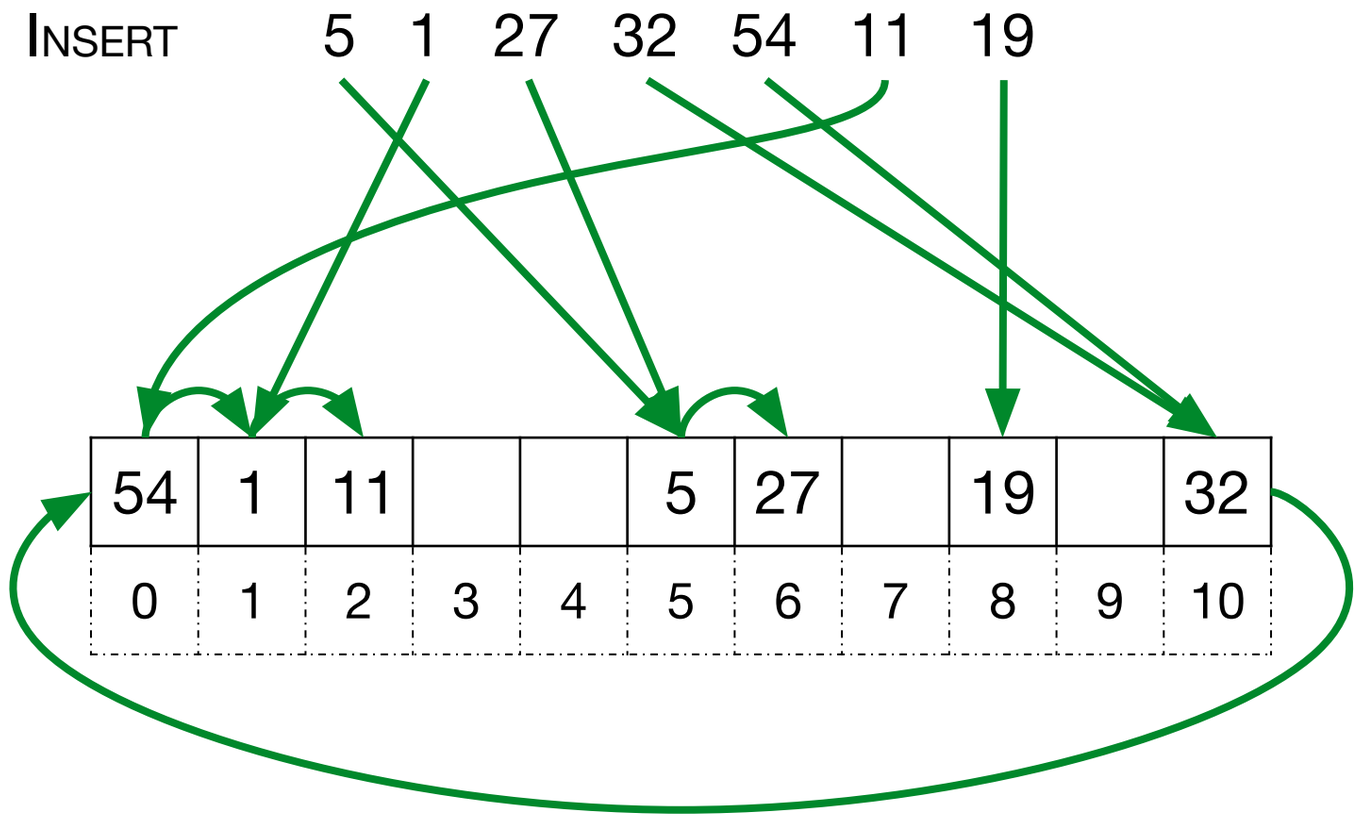
Lineares Sondieren

- Lineares Sondieren.
  - Verwalte S in einem Feld A der Größe m.
  - Element x gespeichert in A[h(x.key)] oder im Cluster rechts von A[h(x.key)].
  - Cluster = (zyklisch) zusammenhängende Sequenz von nichtleeren Einträgen.

	41	1	11	13	54			98	
0	1	2	3	4	5	6	7	8	9

h(k) = k mod 10

- SEARCH(k): starte bei A[h(k)], dann lineare Suche im Cluster rechts davon (zyklisch).
- INSERT(x): versuche x in A[h(x.key)] einzufügen. Wenn der Eintrag nicht leer ist, versuche es im nächsten Eintrag rechts davon, usw. (zyklisch).
- DELETE(x): suche den Eintrag A[i] von x mit SEARCH(x.key), dann lösche x aus A[i] und füge alle Elemente im Cluster rechts von x erneut ein.



$h(k) = k \bmod 11$

Lineares Sondieren

- Satz. Einfaches gleichmäßiges Hashing $\Longrightarrow$ erwartete Zeit O(1) für alle Operationen im linearen Sondieren.
- Caching. Lineares Sondieren ist Cache-effizient.

	41	1	11	13	54			98	
0	1	2	3	4	5	6	7	8	9

h(k) = k mod 10

- Varianten.
  - Quadratisches Sondieren
  - Doppeltes Hashing.

Wörterbücher

Datenstruktur	SEARCH	INSERT	DELETE	space
verkettete Liste	O(n)	O(1)	O(1)	O(n)
direkter Zugriff	O(1)	O(1)	O(1)	O(|U|)
Hashing mit Verkettung	O(1)†	O(1)	O(1)	O(n)
lineares Sondieren	O(1)†	O(1)†	O(1)†	O(n)

† = erwartete Zeit unter einfachem gleichmäßigem Hashing

Hashing

- Wörterbücher
- Hashing mit Verkettung
- Lineares Sondieren
- Hashfunktionen

Hashfunktionen

- Einfache Hashfunktionen.
  - Divisionsmethode. h(k) = k mod m. Typischerweise ist m Primzahl.
  - h(k) =⌊m(kZ -⌊kZ⌋)⌋ für eine Konstante Z mit 0 < Z < 1.

- Universelle Hashfunktionen.
  - Wähle eine zufällige Hashfunktion aus einer Familie von Hashfunktionen.
  - Haben starke Garantien für die Kollisionswahrscheinlichkeiten.
  - ⇒ Wörterbücher mit Operationen erwarteter konstanter Laufzeit.
  - Nur die Hashfunktion wird zufällig ausgewählt. Die Garantien gelten dann unabhängig von der Eingabemenge S.

- Andere Hashfunktionen.
  - Tabulation hashing, MurmurHash, SHA-xxx, FNV, ...

- Anwendungen.
  - Kryptographie, Ähnlichkeit, Programmierung, ...

Hashing

- Wörterbücher
- Hashing mit Verkettung
- Lineares Sondieren
- Hashfunktionen