

Tutorium 4: Hashing

Holger Ebhart | 13. Mai 2015

TUTORIUM ZUR VORLESUNG ALGORITHMEN I IM SS15							
000011100	01 0011100100	1011110110	0100000100	0100000010	1110001001		
000110111	1 1001001011	0001000100	0001111001	0110111010	J10J10J1J1		
1001111001	1010001100	1001111101	0110000100	10000000000	0J00J0J0JJ		
1100010001	0110110110	0000010011	1110111011	1110101101	00100111110		
0111110011	1000011111	0111111000	0110001100	0001000111	0001111011		
1000011101	1011101010	1000101001	0110011101	010001011	7 7770770070		
1100001111	1101100110	1110101101	1001001000	11000100	<i>J0 00J100J0J0</i>		
1001000100	0010011010	0001111110	1011111100	1000001	001 0111000011		
0011100101	1111001101	0001110111	0101110100	011001	0110 1101100111		

Gliederung



- 3. Übungsblatt
- Wahrscheinlichkeitstheorie
 - Permutationen
 - Zufallsvariablen
- Hashing
 - Hashing mit linearer Suche
 - Vergleich
 - Kreativaufgabe
- Nächstes Übungsblatt



13. Mai 2015

3. Übungsblatt





Holger Ebhart - Hashing

Hashing

Grundlagen



Ziehen von k Kugeln aus einer Urne mit n Kugeln

- Kugeln unterscheidbar (Beachtung der Reihenfolge)
 - mit Zurücklegen: n^k
 - ohne Zurücklegen: $\frac{n!}{(n-k)!} = n \cdot (n-1) \cdot ... \cdot (n-(k-1))$
- Kugeln nicht unterscheidbar (Reihenfolge egal)
 - mit Zurücklegen: (^{n+k-1})
 - ohne Zurücklegen: (ⁿ_ν)

13. Mai 2015



Berechne Folgendes:

- Wie viele verschiedene Wörter w lassen sich aus $\{A, B, M, N, S\}$ mit |w| = 3 bilden?
- Wie viele, wenn kein Buchstabe doppelt vorkommen darf?
- Wie viele neue Wörter können aus "ANANAS" gebildet werden?



Ein Parkplatz habe 18 Plätze, welche folgendermaßen belegt sind: 6 Opel, 2 Fiat, 4 Volvo, 5 Skoda und 1 Smart. Die Fahrzeuge sind durch ihre Nummernschilder unterscheidbar.

Wie viele Möglichkeiten gibt es wenn

- alle Skodas nebeneinander stehen?
- alle Fahrzeuge der gleichen Marke nebeneinander stehen?



Ein Parkplatz habe 18 Plätze, welche folgendermaßen belegt sind: 6 Opel, 2 Fiat, 4 Volvo, 5 Skoda und 1 Smart. Die Fahrzeuge sind durch ihre Nummernschilder unterscheidbar.

Wie viele Möglichkeiten gibt es wenn

- alle Skodas nebeneinander stehen?
- alle Fahrzeuge der gleichen Marke nebeneinander stehen?





Ein Parkplatz habe 18 Plätze, welche folgendermaßen belegt sind: 6 Opel, 2 Fiat, 4 Volvo, 5 Skoda und 1 Smart. Die Fahrzeuge sind durch ihre Nummernschilder unterscheidbar.

Wie viele Möglichkeiten gibt es wenn

- alle Skodas nebeneinander stehen?
- alle Fahrzeuge der gleichen Marke nebeneinander stehen?



Universelles Hashing



- $\mathcal{H} \subseteq \{0 \dots m-1\}^{Key}$
- $h \in \mathcal{H}$ zufällig
- $\forall x, y \in Key : x \neq y$
- $\mathbb{P}[h(x) = h(y)] = \frac{1}{m}$

■
$$h: \{2^n | n \in \mathbb{N}\} \to \{0, \dots, 9\}$$

 $x \to x \mod 10$

13. Mai 2015

Universelles Hashing



- $\mathcal{H} \subseteq \{0 \dots m-1\}^{Key}$
- $h \in \mathcal{H}$ zufällig
- $\forall x, y \in \textit{Key} : x \neq y$
- $\mathbb{P}[h(x) = h(y)] = \frac{1}{m}$

Ist die folgende Hashfunktion h universell?

$$h: \{2^n | n \in \mathbb{N}\} \to \{0, \dots, 9\}$$
$$x \to x \mod 10$$





- verwende Array mit mind. n + 1 Slots
- insert(e) → h(key(e)) frei?
 ia: Finfügen von e

nein: suche nächsten freien Platz und füge e dort ein

- Ind(k) Suche ab h(k) nach einem Element x mit key(x) = k. Ende spätestens bei einer Lücke.
- remove(k) → find(k) + löscher



13. Mai 2015



- verwende Array mit mind. n + 1 Slots
- insert(e) \rightarrow h(key(e)) frei?

ja: Einfügen von e

nein: suche nächsten freien Platz und füge e dort ein

- find(k) Suche ab h(k) nach einem Element x mit key(x) = k. Ende spätestens bei einer Lücke.
- remove(k) → find(k) + löschen



13. Mai 2015



- verwende Array mit mind. n + 1 Slots
- insert(e) $\rightarrow h(key(e))$ frei?

ja: Einfügen von e

nein: suche nächsten freien Platz und füge e dort ein

- find(k) Suche ab h(k) nach einem Element x mit key(x) = k. Ende spätestens bei einer Lücke.
- lacksquare remove(k) ightarrow find(k) + löschen



13. Mai 2015



- verwende Array mit mind. n + 1 Slots
- insert(e) \rightarrow h(key(e)) frei?

ja: Einfügen von e

nein: suche nächsten freien Platz und füge e dort ein

- find(k) Suche ab h(k) nach einem Element x mit key(x) = k. Ende spätestens bei einer Lücke.
- remove(k) → find(k) + löschen



Abschluss

Beispiel



- 4 Slots
- $h(k) = (k+1) \mod 5$
- key(e) = e
- Führe folgende Operationen aus:
 - 1 Einfügen von 7, 8, 13, 2
 - 2 Suche nach 13 2
 - 3 Entfernen von 13. 7

13. Mai 2015

Beispiel



- 4 Slots
- $h(k) = (k+1) \mod 5$
- key(e) = e
- Führe folgende Operationen aus:
 - Einfügen von 7, 8, 13, 2
 - 2 Suche nach 13, 2
 - Entfernen von 13, 7



13. Mai 2015

Beispiel



- 4 Slots
- $h(k) = (k+1) \mod 5$
- key(e) = e
- Führe folgende Operationen aus:
 - Einfügen von 7, 8, 13, 2
 - 2 Suche nach 13, 2
 - 3 Entfernen von 13, 7

13. Mai 2015

Vergleich



Operation	Array	verkettete Listen	lineare Suche
create	O(1) / O(n)	O(1) / O(n)	O(1) / O(n)
insert	O(1) (*)	O(1)	amortisiert $O(1)$
find	O(1) (*)	erwartet $O(1)$	(amortisiert) $O(1)$
remove	O(1) (*)	erwartet $O(1)$	(amortisiert) $O(1)$

^{*:} Es existiert nur ein Slot pro Wert den die Hashfunktion annehmen kann



13. Mai 2015

 $[\]rightarrow$ Kollision



- a) Entwerfen Sie ein SparseArray ("spärlich besetztes Array"). Dabei handelt es sich um eine Datenstruktur mit den Eigenschaften eines beschränkten Arrays, die zusätzlich schnelle Erzeugung und schnellen Reset ermöglicht. Nehmen Sie dabei an, dass allocate beliebig viel uninitialisierten Speicher in konstanter Zeit liefert. Das SparseArray habe folgende Eigenschaften:

 - **Das** SparseArray unterstützt die Operation get(i) und set(i, x). Dabei



- a) Entwerfen Sie ein SparseArray ("spärlich besetztes Array"). Dabei handelt es sich um eine Datenstruktur mit den Eigenschaften eines beschränkten Arrays, die zusätzlich schnelle Erzeugung und schnellen Reset ermöglicht. Nehmen Sie dabei an, dass allocate beliebig viel uninitialisierten Speicher in konstanter Zeit liefert. Das SparseArray habe folgende Eigenschaften:
 - Ein *SparseArray* mit *n* Slots braucht O(n) Speicher.

 - **Das** SparseArray unterstützt die Operation get(i) und set(i, x). Dabei

11/13



- a) Entwerfen Sie ein SparseArray ("spärlich besetztes Array"). Dabei handelt es sich um eine Datenstruktur mit den Eigenschaften eines beschränkten Arrays, die zusätzlich schnelle Erzeugung und schnellen Reset ermöglicht. Nehmen Sie dabei an, dass allocate beliebig viel uninitialisierten Speicher in konstanter Zeit liefert. Das SparseArray habe folgende Eigenschaften:
 - Ein *SparseArray* mit n Slots braucht O(n) Speicher.
 - Erzeugen eines leeren *SparseArray* mit *n* Slots braucht O(1) Zeit.
 - Das SparseArray unterstützt eine Operation reset, die es in O(1) Zei in leeren Zustand versetzt.
 - Das SparseArray unterstützt die Operation get(i) und set(i, x). Dabei liefert A.get(i) den Wert, der sich im i-ten Slot des SparseArray A befindet; A.set(i, x) setzt das Element im i-ten Slot auf den Wert x. Wurde der i-te Slot seit der Erzeugung bzw. dem letzten reset noch nicht mit auf einen bestimmten Wert gesetzt, so liefert A.get(i) einen speziellen Wert ⊥. Die Operationen get und set dürfen zudem beide nicht mehr als O(1) Zeit verbrauchen (random access).



- a) Entwerfen Sie ein SparseArray ("spärlich besetztes Array"). Dabei handelt es sich um eine Datenstruktur mit den Eigenschaften eines beschränkten Arrays, die zusätzlich schnelle Erzeugung und schnellen Reset ermöglicht. Nehmen Sie dabei an, dass allocate beliebig viel uninitialisierten Speicher in konstanter Zeit liefert. Das SparseArray habe folgende Eigenschaften:
 - Ein *SparseArray* mit *n* Slots braucht O(n) Speicher.
 - Erzeugen eines leeren SparseArray mit n Slots braucht O(1) Zeit.
 - Das SparseArray unterstützt eine Operation reset, die es in O(1) Zeit in leeren Zustand versetzt.
 - Das SparseArray unterstützt die Operation get(i) und set(i, x). Dabei liefert A.get(i) den Wert, der sich im i-ten Slot des SparseArray A befindet; A.set(i, x) setzt das Element im i-ten Slot auf den Wert x. Wurde der i-te Slot seit der Erzeugung bzw. dem letzten reset noch nicht mit auf einen bestimmten Wert gesetzt, so liefert A.get(i) einen speziellen Wert ⊥. Die Operationen get und set dürfen zudem beide nicht mehr als O(1) Zeit verbrauchen (random access).



- a) Entwerfen Sie ein SparseArray ("spärlich besetztes Array"). Dabei handelt es sich um eine Datenstruktur mit den Eigenschaften eines beschränkten Arrays, die zusätzlich schnelle Erzeugung und schnellen Reset ermöglicht. Nehmen Sie dabei an, dass allocate beliebig viel uninitialisierten Speicher in konstanter Zeit liefert. Das SparseArray habe folgende Eigenschaften:
 - Ein *SparseArray* mit *n* Slots braucht O(n) Speicher.
 - Erzeugen eines leeren *SparseArray* mit n Slots braucht O(1) Zeit.
 - Das SparseArray unterstützt eine Operation reset, die es in O(1) Zeit in leeren Zustand versetzt.
 - Das SparseArray unterstützt die Operation get(i) und set(i, x). Dabei liefert A.get(i) den Wert, der sich im i-ten Slot des SparseArray A befindet; A.set(i, x) setzt das Element im *i*-ten Slot auf den Wert x. Wurde der i-te Slot seit der Erzeugung bzw. dem letzten reset noch nicht mit auf einen bestimmten Wert gesetzt, so liefert A.get(i) einen speziellen Wert \perp . Die Operationen get und set dürfen zudem beide nicht mehr als O(1) Zeit verbrauchen (*random access*).



- b) Nehmen Sie an, dass die Datenelemente, die Sie im SparseArray ablegen wollen, recht groß sind (also z.B. Records mit 10, 20 oder mehr Einträgen). Geht Ihre Realisierung unter dieser Annahme sparsam oder verschwenderisch mit dem Speicherplatz um? Wenn Sie Ihre Realisierung für verschwenderisch halten, überlegen Sie ob und wie Sie es besser machen können.
- c) Vergleichen Sie Ihr SparseArray mit Bounded-Arrays. Welche Vorteile und Nachteile sehen Sie im Hinblick auf den Speicherverbrauch und auf das Iterieren über alle Elemente mit set eingefügten Elemente?



12/13



- b) Nehmen Sie an, dass die Datenelemente, die Sie im SparseArray ablegen wollen, recht groß sind (also z.B. Records mit 10, 20 oder mehr Einträgen). Geht Ihre Realisierung unter dieser Annahme sparsam oder verschwenderisch mit dem Speicherplatz um? Wenn Sie Ihre Realisierung für verschwenderisch halten, überlegen Sie ob und wie Sie es besser machen können.
- c) Vergleichen Sie Ihr SparseArray mit Bounded-Arrays. Welche Vorteile und Nachteile sehen Sie im Hinblick auf den Speicherverbrauch und auf das Iterieren über alle Elemente mit set eingefügten Elemente?



Hashtabelle



- Es sei die Hashfunktion $h(key) = (key * 3) \mod 9$ gegeben
- Es sei: $key : Element \rightarrow \{0, 1, \dots, 9\}$
- Füge die Elemente 1,5,9,23,44,65,78,91 in eine Hashtabelle
- Benutze Hashing mit verketteten Listen
- Benutze Hashing mit linearer Suche

Hashtabelle



- Es sei die Hashfunktion $h(key) = (key * 3) \mod 9$ gegeben
- Es sei: $key: Element \rightarrow \{0, 1, \dots, 9\}$
- Füge die Elemente 1,5,9,23,44,65,78,91 in eine Hashtabelle
- Benutze Hashing mit verketteten Listen
- Benutze Hashing mit linearer Suche



13. Mai 2015

Hashtabelle

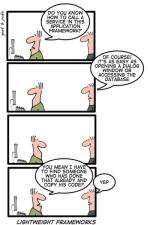


- Es sei die Hashfunktion $h(key) = (key * 3) \mod 9$ gegeben
- Es sei: $key : Element \rightarrow \{0, 1, \dots, 9\}$
- Füge die Elemente 1,5,9,23,44,65,78,91 in eine Hashtabelle
- Benutze Hashing mit verketteten Listen
- Benutze Hashing mit linearer Suche



13. Mai 2015

Vielen Dank für eure Aufmerksamkeit! Bis zum nächsten Mal.



datamation.com



13. Mai 2015