Datenstrukturen und effiziente Algorithmen

Markus Vieth, David Klopp, Christian Stricker

4. Dezember 2015

Inhaltsverzeichnis

I. Sortieren **4**

1. Vorlesung 1 **5**
1.1. Bubblesort 5
1.1.1. Pseudocode 5
1.1.2. Laufzeitanalyse 5
1.2. Heapsort 6
1.2.1. Heap-Eigenschaft 6

2. Vorlesung 12 **7**
2.1. (a,b)-Suchbäume 7
2.1.1. Aufspaltung bei Einfügen 7
2.1.2. Verschmelzen von Knoten beim Löschen 7
2.2. Amortisierte Analyse 7
2.2.1. Bankkonto-Methode 7

3. Vorlesung 13 **9**
3.1. Hashing 9
3.1.1. Universelles Hashing 10

Teil I.

Sortieren

1. Vorlesung 1

1.1. Bubblesort

1.1.1. Pseudocode

```
void bubblesort (int[] a) {
  int n = a.length;
  for (int i = 1; i < n; i++) {
    for (int j = 0; j < n-i; j++) {
      if ( a[j] < a[j+1])
        swap (a, j, j+1);
    }
  }
}
```

Schleifen-Invariante: Nach dem Ablauf der i-ten Phase gilt:

Die Feldpositionen n-i,...,n-i enthalten die korrekt sortierten Feldelemente

Beweis durch Induktion nach i $\overset{i=n-1}{\Longrightarrow}$ Sortierung am Ende korrekt.

1.1.2. Laufzeitanalyse

1.	Phase	n-1
2.	Phase	n-1
3.	Phase	n-1
	$\vdots$	
i.	Phase	n-1
	$\vdots$	
(n-1).	Phase	n-1
$1+2+3+\ldots+(n+1)$		

$$T(n) = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} \in O(n^2)$$

n	T_{real}
2^{10}	8ms
2^{11}	11ms
2^{12}	26ms
$\vdots$	
2^{16}	5,819s
2^{17}	23,381s
$\vdots$	
2^{20}	16min
$\vdots$	
2^{26}	52d

$$T_{real}(n) \approx cn^2 \quad c \approx 10^{-6}$$

1.2. Heapsort

z.B. 21 6 4 7 12 5 3 11 14 17 19 8 9 10 42

Skizze

1.2.1. Heap-Eigenschaft

2. Vorlesung 12

2.1. (a,b)-Suchbäume

Blattorientierte Speicherung der Elemente
Innere Knoten haben mindestens a und höchstens b Kinder und tragen entsprechende Schlüsselwerte, um die Suche zu leiten.

Beispiel:

$$h \hat{=} \text{Tiefe} \Rightarrow \quad a^h \leq n \leq b^h \quad \Rightarrow \quad \log_b n \leq h \leq \log_a n$$

2.1.1. Aufspaltung bei Einfügen

2.1.2. Verschmelzen von Knoten beim Löschen

Aufspalte- und Verschmelze-Operationen können sich von der Blattebene bis zur Wurzel kaskadenartig fortpflanzen. Sie bleiben aber auf den Suchpfad begrenzt.
$\Rightarrow$ Umbaukosten sind beschränkt durch die Baumtiefe $= O(\log n)$

2.2. Amortisierte Analyse

Beispiel: Binärzähler

$$\begin{array}{lrl} 000 & & \\ 001 & \text{Kosten}(1) = 1 & \\ 010 & = 2 & \\ 011 & = 1 & \\ 100 & = 3 & \text{Kosten der Inkrement-Operation} \hat{=} \text{Zahl der Bit-Flips} \\ 101 & = 1 & \\ 110 & = 2 & \\ 111 & = \underline{1} & \\ & 11 & \end{array}$$

Naive Analyse $2^k = n$

$$1 \cdot \frac{n}{2} + 2 \cdot \frac{n}{4} + 3 \cdot \frac{n}{8} + \ldots + k \cdot \frac{n}{2^k} = \frac{n}{2} \sum_{i=1}^{k} i(\frac{1}{2})^{i-1} = 2^{k+1} - k - 2 = 2n - k - 2$$

Von 0 bis n im Binärsystem zu zählen kostet $\leq 2n$ Bit-Flips

Sprechweise: amortisierte Kosten einer Inkrement-Operation sind 2
Folge von n-Ops kostet $2n$

2.2.1. Bankkonto-Methode

$$\text{Konto}(i+1) = \text{Konto}(i) - \text{Kosten}(i) + \text{Einzahlung}(i)$$

$$\sum_{i=1}^{n} \text{Kosten}(i) = \text{tatsächliche Gesamtkosten} = \sum_{i=1}^{n} (\text{Einzahlung}(i) + \text{Konto}(i - \text{Konto}(i+1))$$

$$= \sum_{i=1}^{n} \text{Einzahlung}(i) + \text{Konto}(1) - \text{Konto}(n+1)$$

$$\begin{array}{ll}
000 & \\
001_{€} & \text{Kosten}(1) = 1 \\
01_{€}0 & = 2 \\
01_{€}1_{€} & = 1 \\
1_{€}00 & = 3 \\
1_{€}01_{€} & = 1 \\
1_{€}1_{€}0 & = 2 \\
1_{€}1_{€}1_{€} & = \underline{1} \\
 & 11
\end{array}$$

Kontoführungsschema: für Binärzähler

1 € pro 1 in der Binärdarstellung
Jeder Übergang $1_{€} \to 0$ kann dann mit dem entsprechenden Euro Betrag auf dieser 1 bezahlt werden.
Es gibt pro Inkrement Operation nur einen $0 \to 1$ Übergang
2 € Einzahlung für jede Inc-Operation reichen aus um:

1. diesen $0 \to 1$ Übergang zu bezahlen

2. die neu entstandene $1_{€}$ mit einem Euro zu besparen.

$$\text{GK} = 2(2^k - 1) + 0^{\text{I}} - k^{\text{II}} = 2n - k - 2$$

[I] Zählerstand(000)

[II] Zählerstand($\overbrace{111\ldots1}^{k}$)

3. Vorlesung 13

Satz: Ausgehend von einem leeren 2-5-Baum betrachten wir die Rebalancierungskosten C (Split- und Fusionsoperationen) für eine Folge von m Einfüge- oder Löschoperationen. Dann gilt: $C \in O(m)$
d.h. Amortisierte Kosten der Split- und Fusionsopeartionen sind konstant.
! Dies bezieht sich nicht auf die Suchkosten, die in $O(\log n)$ liegen.

Beweisidee:

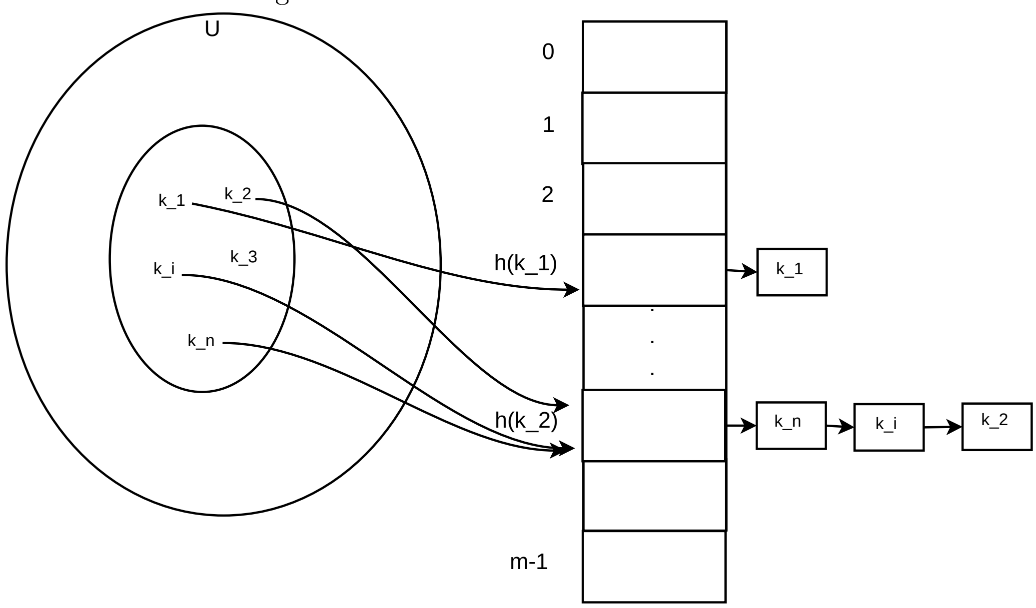
Kontoführung:

1	2	3	4	5	6
2 €	1 €	0 €	0 €	1 €	2 €

regelmäßige Einzahlung: 1 €
Durch eine Einfüge- oder Löschoperation steigt oder fällt der Knotengrad des direkt betroffenen Knotens um höchstens 1. $\Rightarrow$ 1 € Einzahlung reicht zur Aufrechterhaltung dieses Sparplanes.
Jetzt Beseitigung der temporären 1- und 6-Knoten:
Ein 6-Knoten nutzt 1 € um seinen Split zu bezahlen. Die beiden neu entstehenden 3-Knoten benötigen kein Kapital. Der Vaterknoten des gesplitteten 6-Knotens benötigt ggf. den zweiten verfügbaren €. Analoge Betrachtung für Fusion eines temp. 1-Knotens.

3.1. Hashing

Abbildung 3.1.: Universum und Hashtabelle der Größe m



$U \subseteq \mathbb{N}$ z.B. 64-Bit-Integer
$n =$ Zahl dr zu verwaltenden Schlüssel

$$|U| >> n$$

Hashfunktion h:

$$h : U \to [0, \dots, m-1]$$

$$\text{z.B. } k \mapsto k \mod m$$

Einfache Annahme: (einfaches uniformes Hashing)

$$\forall\ k_i, k_j \in U : Pr(h(k_i) = h(k_j)) = \frac{1}{m}$$

Analyse der Laufzeit zum Einfügen eines neuen Elementes k

- $h(k)$ berechnen $\longrightarrow O(1)$
- Einfügen am Listenanfang in Fach $h(k)$. $\longrightarrow O(1)$

Analyse der Suchzeit für einen schlüssel k

- $h(k) \longrightarrow O(1)$
- Listenlänge zum Fach $h(k)$ sei $n_{h(k)}$ also beim Durchlauf der kompletten Liste $\longrightarrow O(n_{h(k)})$

$$E(n_{h(k)}) = \frac{n}{m} = \alpha^{\mathrm{I}}$$

$$\text{Suchzeit(Einfügen)} \in O(1+\alpha)$$

Laufzeit beim Löschen von Schlüssel k

- $h(k) \longrightarrow O(1)$
- Durchlaufen der Liste $\longrightarrow 0(n_{h(k)})$
- Löschen durch „Pointer-Umbiegen" $\longrightarrow O(1)$

3.1.1. Universelles Hashing

Idee Arbeite nicht mit einer festen Hashfunktionm sondern wähle am Anfang eine zufällige Hashfunktion aus einer Klasse von Hashfunktionen aus.

z.B.

$$h_{a,b}(k) = ((a \cdot k + b) mod p) mod m$$

p sei eine hinreichend große Primzahl $\quad 0 < a < p, 0 \leq b < p$

$$\mathcal{H}_{p,m} = \{h_{a,b}(k) | 0 < a < p,\ 0 \leq b < p\}$$

$$|\mathcal{H}_{p,m}| = p(p-1)$$

Definition $\mathcal{H}$ heißt universell $\Leftrightarrow \quad \forall\ k, l \in U :\ Pr(h(k) = h(l)) \leq \frac{1}{m}$

[I] Belegungsfaktor

Suchzeit

$$\mathcal{X}_{k,l} = \begin{cases} 1 & \text{für } h(k) = h(l) \\ 0 & \text{sonst} \end{cases}$$

$$E(n_{h(k)}) = E\left(\sum_{l \in T, l \neq k}\right) = \sum_{l \in T, l \neq k} E(X_{k,l}) = \sum_{l \in T, l \neq k} Pr(h(k) = h(l)) = \sum_{l \in T, l \neq k} \frac{1}{m} = \frac{n-1}{m} = \alpha$$