# 3. Tutorium - Algorithmen I

Nina Zimbel

6.05.2015

- Anmerkungen zu den Übungsblättern
- Datenstrukturen
- Amortisierte Analyse
- Hashing

# Zu den Übungsblättern

- Nr. 2d) : Nicht in der Form des Mastertheorems $\rightarrow$ streng genommen nicht anwendbar.
- Nicht aus Musterlösungen abschreiben! Vor allem nicht die Fehler mit abschreiben...

# Aufgabe

Entwickeln Sie eine Datenstruktur, die folgendes kann:

- *pushBack* und *popBack* in O(1) im Worst-Case.
- Zugriff auf das k-te Element in O(log n) im Worst-Case.

Wobei eine Speicherallokation beliebiger Größe nur O(1) kosten soll.

# Es geht auch umgekehrt

Gesucht: eine Datenstruktur mit

- *pushBack* und *popBack* in O(log n) im Worst-Case.
- Zugriff auf das k-te Element in O(1) im Worst-Case.

# Amortisierte Analyse - Methoden

- **Bankkontomethode / Account-Methode**: Operationen werden Kosten zugewiesen, die auf ein Konto eingezahlt bzw. abgebucht werden. Kontostand darf nie negativ werden.
- **Aggregationsmethode**: Gesamtkosten aller Operationen wird ermittelt und durch Anzahl aller Operationen geteilt $\rightarrow$ amortisierte Kosten

# Hashing

- Größe einer Hashtabelle?

# Hashing

- Größe einer Hashtabelle?
- $\rightarrow \Theta(k)$ mit: k = Größe der Eingabe (siehe Theorem 1 auf den VL-Folien)

# Hashing

- Größe einer Hashtabelle?
- $\rightarrow \Theta(k)$ mit: k = Größe der Eingabe (siehe Theorem 1 auf den VL-Folien)
- Wahl der Hashfunktion?

# Hashing

- Größe einer Hashtabelle?
- $\rightarrow \Theta(k)$ mit: k = Größe der Eingabe (siehe Theorem 1 auf den VL-Folien)
- Wahl der Hashfunktion?
- $\rightarrow$ Zufällig aus universeller Familie von Hashfunktionen
- Wahrscheinlichkeit einer Kollision?

# Hashing

- Größe einer Hashtabelle?
- $\rightarrow \Theta(k)$ mit: k = Größe der Eingabe (siehe Theorem 1 auf den VL-Folien)
- Wahl der Hashfunktion?
- $\rightarrow$ Zufällig aus universeller Familie von Hashfunktionen
- Wahrscheinlichkeit einer Kollision?
- $\rightarrow$ 1/m (m: Größe der Tabelle)

# Hashing - Aufgabe

Gegeben sei ein Array $A = A[1], \ldots, A[n]$ mit n Zahlen in beliebiger Reihenfolge. Für eine gegebenen Zahl x soll ein Paar $(A[i], A[j])$, $1 \leq i, j \leq n$ gefunden werden, für das gilt: $A[i] + A[j] = x$.

1. Geben Sie eine Lösung für $x = 33$ und $A = (7, 15, 21, 14, 18, 3, 9)$ an.
2. Geben Sie einen effizienten Algorithmus an, der das Problem in erwarteter Zeit $O(n)$ löst, und bei Erfolg ein Paar $(A[i], A[j])$ ausgibt, ansonsten Nil.

# Klausuraufgabe SS13

Wir betrachten im Foldenden Hashtabellen mit $n$ Buckets und zugehörigen Hashfunktionen,

$h_n(\text{x}) = (\text{x DIV n}) \text{ MOD n}.$

Beispielsweise ist $h_{11}(74) = 6$. Zur Kollisionsauflösung werden einfach verkettete Listen verwendet. Folgende Hashtabelle hat beispielsweise Größe $\text{n} = 11$ und Hashfunktion $h_{11}$:
(siehe Tafel)

**a.** Sei nun eine leere Hashtabelle mit n = 11 und Hashfunktion $h_{11}$ gegeben.
Geben Sie eine Folge von *insert*-Operationen an, so dass die Tabelle nach Ausführen dieser Operationsfolge den obigen Zustand hat.

**b.** Geben Sie für eine leere Hashtabelle der Größe n mit Hashfunktion $h_n$ eine Folge von n **verschiedenen** *insert*-Operationen und n **verschiedenen** *find*-Operationen an, so dass die erwartete Laufzeit für die Operationsfolge nicht gilt.
Begründen Sie kurz, warum Ihre Folge das gewünschte Verhalten liefert.

**c.** Nennen Sie zwei Vorteile von Hashing mit verketteten Listen gegenüber Hashing mit linearer Suche.