Übung Algorithmen und Datenstrukturen

Sommersemester 2016

Patrick Schäfer, Humboldt-Universität zu Berlin

Agenda

- Amortisierte Analyse
- Suche in sortierten Arrays
- Heaps
- Vorstellen des fünften Übungsblatts

Binärzähler

- Geg.: k-Bit Binärzähler
- Array von k Bits $b_0, b_1, \dots, b_{k-1}$
  - Entspricht Binärzahl $b_{k-1} \dots b_1 b_0$ bzw. Dezimalzahl $\sum_i b_i 2^i$
- Operation: Zahl inkrementieren (um 1 erhöhen)
- Kosten: Anzahl der Bitänderungen (jedes Bit kostet 1)

n	b_4	b_3	b_2	b_1	b_0
0	0	0	0	0	0
1	0	0	0	0	1

1 Bitwechsel

n	b_4	b_3	b_2	b_1	b_0
23	1	0	1	1	1
24	1	1	0	0	0

4 Bitwechsel

Binärzähler – Kostenabschätzung

- Gesucht: Kosten für n Inkrement-Operationen, wenn Zähler bei 0 beginnt.
- Best Case: Eine Bitänderung
- Worst Case: Alle k Bits werden verändert: $O(nk)$
- Problem: sehr pessimistische Abschätzung, da Worst Case eher selten

n	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
127	0	1	1	1	1	1	1	1
128	1	0	0	0	0	0	0	0

k Bitwechsel

n	b_3	b_2	b_1	b_0	#BW
0	0	0	0	0	–
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	1
4	0	1	0	0	3
5	0	1	0	1	1
6	0	1	1	0	2
7	0	1	1	1	1
8	1	0	0	0	4
9	1	0	0	1	1
10	1	0	1	0	2

Kosten

$T(n)$	$\sum$ #BW	$\sum_{i=1}^{n} k = \sum_{i=1}^{n} 4$
n=1	1	4
2	3	8
3	4	12
4	7	16
5	8	20
6	10	24
7	11	28
8	15	32
9	16	36
10	18	40

⇒ Häufig geringe Kosten

⇒ Abschätzung zu pessimistisch

Amortisierte Analyse

- Benötigte Kosten werden über alle Operationen gemittelt.
- Beschreibt mittlere Kosten jeder Operation im schlechtesten Fall.

- Grundidee von Guthaben- und Potentialmethode: Anfänglich günstige Operation teurer bewerten, um spätere (teure) Operationen auszugleichen.
  - Überschuss wird als Kredit/Potential gespeichert.
  - Kredit/Potential wird verwendet, um für teurere Operation zu zahlen.
  - Verfahren unterstützten mehr als ein Operationstyp.

- Für das Übungsblatt: Aggregat-Methode.

Aggregat-Methode

- *Obere Schranke für Gesamtkosten:* Zeigen, dass eine Sequenz von n Operationen im schlechtesten Fall insgesamt $\boldsymbol{T(n)}$ Operationen benötigt.

- *Mittlere Kosten:* Die amortisierten Kosten pro Operation betragen dann

$$\boldsymbol{T(n) / n}.$$

- *Unterstützt nur einen Operationstyp:* Kosten gelten für jede Operation, auch wenn es verschiedene Operationstypen gibt.

Aggregat-Methode für k-Bit Binärzähler

- Beobachtung (vgl. Vorlesung, Folie 7): Das k-niedrigste Bit ändert sich bei jeder 2^k-ten Inkrementation

n	b_3	b_2	b_1	b_0	#BW
0	0	0	0	0	–
1	0	0	0	1	**1**
2	0	0	1	0	**2**
3	0	0	1	1	**1**
4	0	1	0	0	**3**
5	0	1	0	1	**1**
6	0	1	1	0	**2**
7	0	1	1	1	**1**
8	1	0	0	0	**4**
	$= \lfloor \frac{n}{8} \rfloor$	$= \lfloor \frac{n}{4} \rfloor$	$= \lfloor \frac{n}{2} \rfloor$	$= \lfloor \frac{n}{1} \rfloor$	

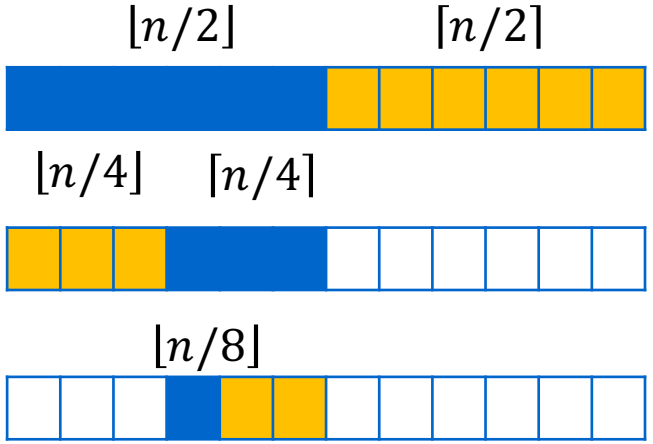
- n Inkrement-Operationen haben die Gesamtkosten:

$$T(n) = \left\lfloor \frac{n}{1} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor + \left\lfloor \frac{n}{4} \right\rfloor + \cdots + \left\lfloor \frac{n}{2^k} \right\rfloor$$
$$\leq n \sum_{i=0}^{k} \frac{1}{2^i}$$
$$\leq n \sum_{i=0}^{\infty} \frac{1}{2^i} = 2n \in O(n)$$

- Amortisierte Kosten pro Operation:

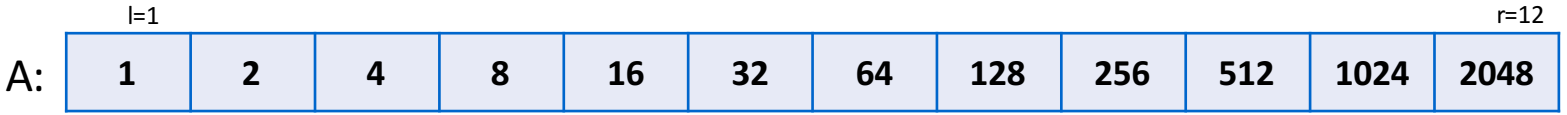
$$\frac{T(n)}{n} \leq 2 \in O(1)$$

Suche in sortierten Arrays



- Suchverfahren aus der Vorlesung:
  - **Binäre Suche**
  - Interpolations-Suche (Übungsblatt)
  - Fibonacci-Suche (Übungsblatt)

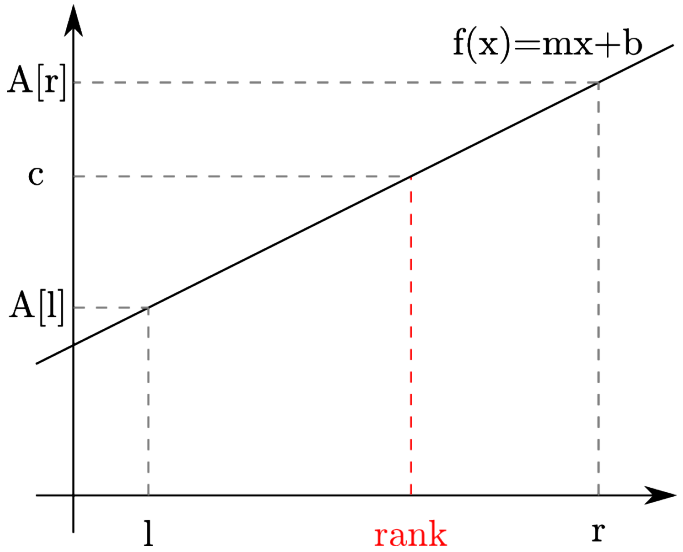
c=8



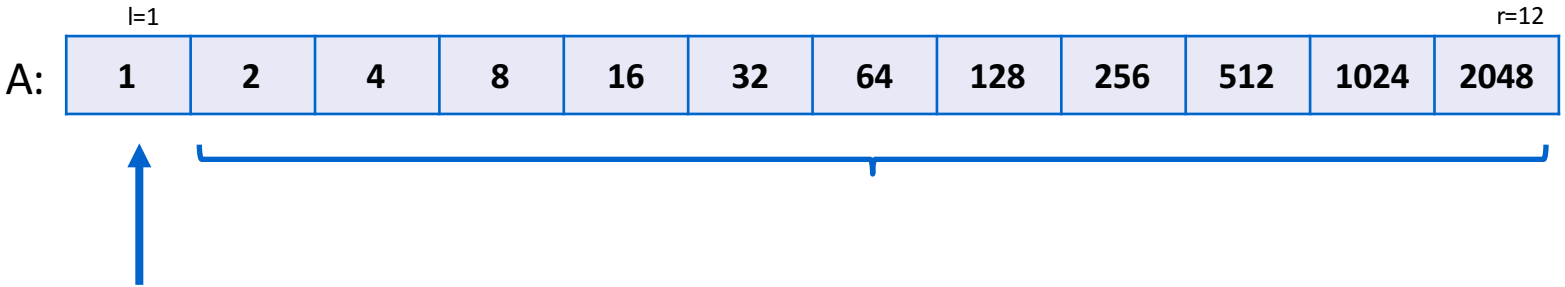
$$m_{bin} = \left\lfloor \frac{(12+1)}{2} \right\rfloor = 6$$

Suche in sortierten Arrays

- Suchverfahren aus der Vorlesung:
  - Binäre Suche
  - **Interpolations-Suche (Übungsblatt)**
  - Fibonacci-Suche (Übungsblatt)



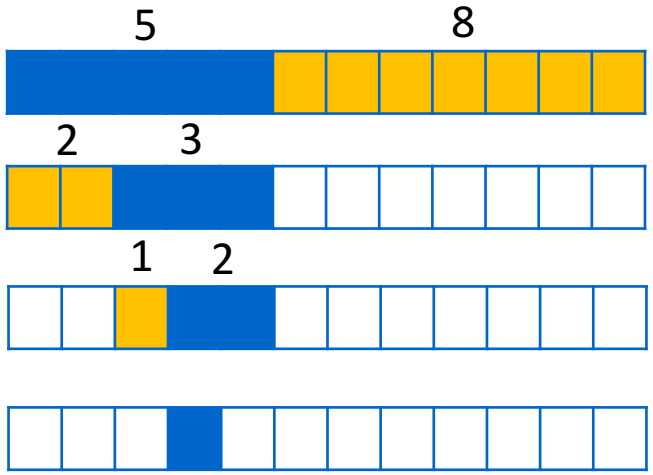
c=8



$$\text{rank} = 1 + \left\lfloor \frac{(12-1)(8-1)}{2048-1} \right\rfloor = 1$$

$$\text{rank} = l + \frac{(r-l)(c-A[l])}{A[r]-A[l]}$$

Suche in sortierten Arrays



- Suchverfahren aus der Vorlesung:
  - Binäre Suche
  - Interpolations-Suche (Übungsblatt)
  - **Fibonacci-Suche (Übungsblatt)**

c=8

l=1 r=12

A:

1	2	4	8	16	32	64	128	256	512	1024	2048

$fib2 \sim \frac{1}{3}(r-l)$

$fib1 \sim \frac{2}{3}(r-l)$

$m_{fib} = \min(fib2, n)$

fib=13, fib1=8, fib2=5

Suche: Schreibtischtest

Führen Sie einen Schreibtischtest für die binäre Suche durch, bei dem das folgende Array A nach dem Wert $c = 69$ durchsucht wird. Geben Sie dazu an, mit welchen Werten die Variablen `l`, `r` und `m` nach jedem Aufruf von Zeile 4 belegt sind.

$A = [5, 12, 15, 17, 22, 29, 45, 47, 60, 61, 68, 74, 77]$

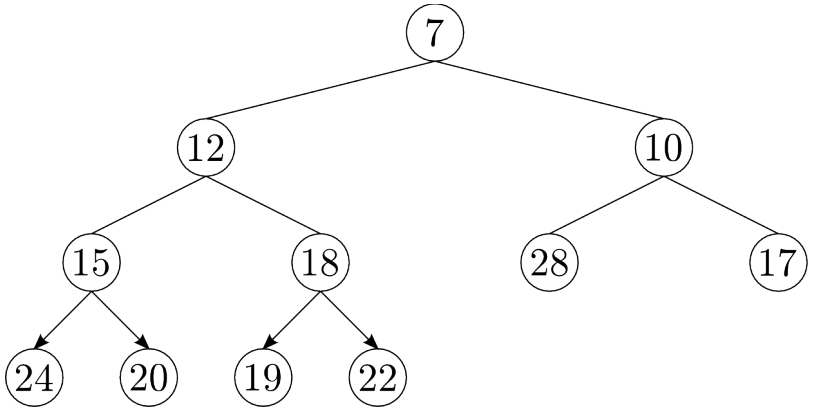
Algorithmus BinarySearch(A, c)

Input: Sortiertes Array A und Integer c
Output: TRUE, falls das Element c in A ist.

```
1:  l := 1;
2:  r := |A|;
3:  while l ≤ r do
4:      m := (l + r) div 2;
5:      if c < A[m] then
6:          r := m − 1;
7:      else if c > A[m] then
8:          l := m + 1;
9:      else
10:         return true;
11:     end if
12: end while
13: return false;
```

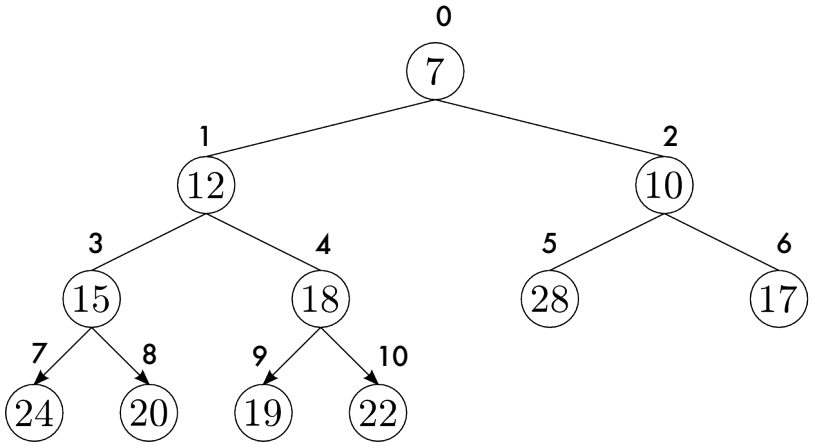
Heaps

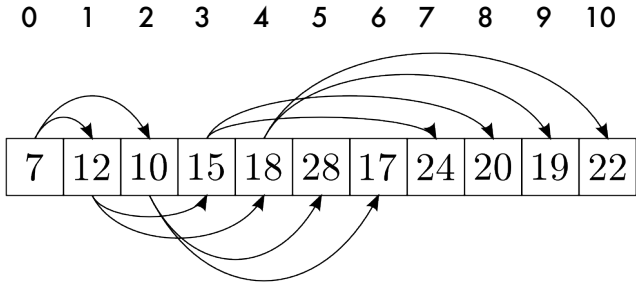
- Ein Heap ist eine auf Bäumen basierende Datenstruktur zum Speichern von Elementen, über deren Schlüssel eine totale Ordnung definiert ist
- Form Constraint: Der Baum ist fast vollständig
  - Alle Ebenen außer der untersten müssen vollständig gefüllt sein
  - In der letzten Ebene werden Elemente von links nach rechts aufgefüllt
- Heap Constraint: Bei einem Min-Heap (Max-Heap) sind die Schlüssel jedes Knotens kleiner (größer) als die Schlüssel seiner Kinder



Headgeordnete Arrays

- Heaps lassen sich als heapgeordnete Arrays repräsentieren.

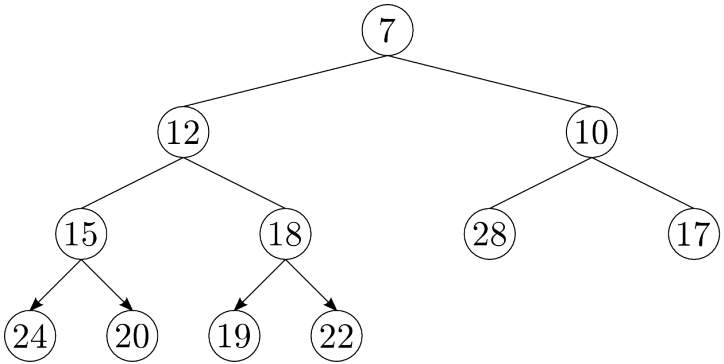


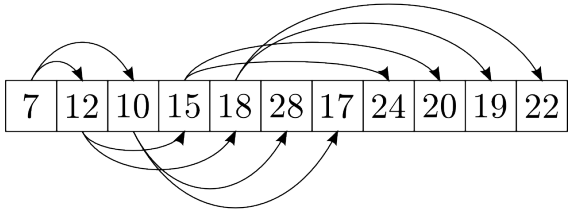


- Die Zahl über dem Knoten entspricht dem Index im Array.
- Der Vater eines Knotens steht an Index $\text{parent(i)} = \left\lfloor \frac{i-1}{2} \right\rfloor$
- Ein Knoten hat die Kinder $left(i) = 2i + 1$ und $right(i) = 2i + 2$

Aufgaben zu Min-Heaps

1. Geben Sie alle möglichen Min-Heaps zu den Zahlen 1, 2, 3, 4 und 5 an.
2. Es sei der folgende Min-Heap als heapgeordnetes Array gegeben:





Wie sehen Heap und Array nach Anwendung der folgenden Operationen (in der gegebenen Reihenfolge) aus? `deleteMin()`, `deleteMin()`, `add(14)`, `add(8)`