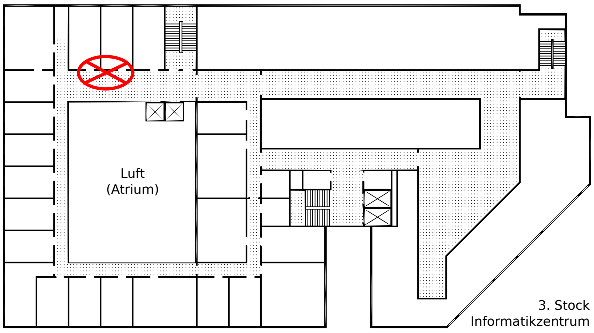
Abteilung Algorithmik **Winter 2016/17**
Institut für Betriebssysteme und Rechnerverbund
TU Braunschweig

Prof. Dr. Sándor P. Fekete
Phillip Keldenich
Arne Schmidt

Algorithmen und Datenstrukturen
Übung 5 vom 9. 1. 2017

Abgabe der Lösungen bis zum Montag, den 23. 1. 2017 um 13:15 im Hausaufgabenrückgabeschrank bei Raum IZ 337.



Bitte die Blätter zusammenheften und vorne deutlich mit eigenem Namen, Matrikel- und Gruppennummer, sowie Studiengang versehen!

Aufgabe 1 (Höhe, Knotenzahl und Balance von AVL-Bäumen): In dieser Aufgabe soll untersucht werden, wie balanciert AVL-Bäume im Bezug auf die Knotenzahl statt der Höhe sind.

a) Wie viele Knoten hat ein AVL-Baum der Höhe h mindestens? Gib einen AVL-Baum der Höhe 5 mit der minimal möglichen Knotenzahl an!

b) Wie viele Knoten hat ein AVL-Baum der Höhe h höchstens? Gib einen AVL-Baum der Höhe 5 mit der maximal möglichen Knotenzahl an!

c) Sei $\mathcal{A}(h)$ die Menge der AVL-Bäume mit Höhe h. Für einen AVL-Baum $A \in \mathcal{A}(h)$ sei $N_1(A)$ die Anzahl der Knoten im linken Unterbaum der Wurzel und $N_2(A)$ die Anzahl der Knoten im rechten Unterbaum der Wurzel. Mit $U(h) := \max\limits_{A \in \mathcal{A}(h)} \frac{N_1(A)}{N_2(A)}$ bezeichnen wir den Quotienten zwischen der Anzahl Knoten im linken und rechten Unterbaum der Wurzel im *worst case*. Zeige: Es gibt ein $c > 1$ mit $U(h) \in \Theta(c^h)$. Hinweis: Für die Fibonacci-Zahlen F_n gilt: $F_n \in \Theta(\phi^n)$, wobei $\phi = \frac{1+\sqrt{5}}{2}$ der *goldene Schnitt* ist.

(5+3+7 Punkte)

Aufgabe 2 (Heaps): Algorithmus 1 zeigt einen Sortieralgorithmus. Wir wollen diesen nun untersuchen.

a) Wende Sort(A) auf das Array $A = [3, 1, 8, 5, 2, 9, 10, 4, 7, 6]$ an. Gib das Array nach jedem nicht-rekursiven Aufruf von Max-Heapify an (dazu zählen auch jene in der Funktion Build-Max-Heap). Gib außerdem die Baumstruktur des Heaps

nach Build-Max-Heap an.
(*Hinweis:* Ein Feld $A[i]$ besitzt als Kinder die Felder $A[2i]$ und $A[2i+1]$.)

b) Zeige, dass Sort(A) ein korrektes Sortierverfahren ist.

c) Zeige, dass Sort(A) eine Laufzeit von $O(n \log n)$ besitzt.

```
function SORT(A)
    BUILD-MAX-HEAP(A)
    for i ←length(A) downto 2 do
        vertausche A[1] und A[i]
        heap-größe[A] ← heap-größe[A] − 1
        MAX-HEAPIFY(A, 1)
    end for
end function
```

Algorithmus 1: Ein Sortieralgorithmus

(15+5+5 Punkte)

Aufgabe 3 (Mergesort): Gegeben sei folgendes Array:

$$A = [2, 15, 4, 1, 3, 16, 7, 8, 5, 12, 10, 14, 6, 11, 9, 13]$$

a) Gib separat und in chronologischer Reihenfolge die Ergebnisse aller Mergeschritte an, indem Du die Zeilen der Tabelle 1 ausfüllst. Die Mergeschritte auf einem Teilarray der Länge 1 müssen nicht angegeben werden. (Hinweis: Ein Mergeschritt im rechten Teilarray wird erst vorgenommen, wenn das linke Teilarray komplett sortiert worden ist.)

b) In Abbildung 1 ist der Rekursionsbaum der Ausführung von Mergesort aus Aufgabenteil a) abgebildet. Jeder Knoten korrespondiert zu genau einem Mergeschritt. Zeichne in jedem Knoten die Zeilennummer des zugehörigen Mergeschrittes ein. (Hinweis: Die Rekursionstiefe entspricht der Tiefe im Baum.)

(15+5 Punkte)

$A =$	2	15	4	1	3	16	7	8	5	12	10	14	6	11	9	13
1. $A =$																
2. $A =$																
3. $A =$																
4. $A =$																
5. $A =$																
6. $A =$																
7. $A =$																
8. $A =$																
9. $A =$																
10. $A =$																
11. $A =$																
12. $A =$																
13. $A =$																
14. $A =$																
15. $A =$																

Tabelle 1: Tabelle für Mergesort

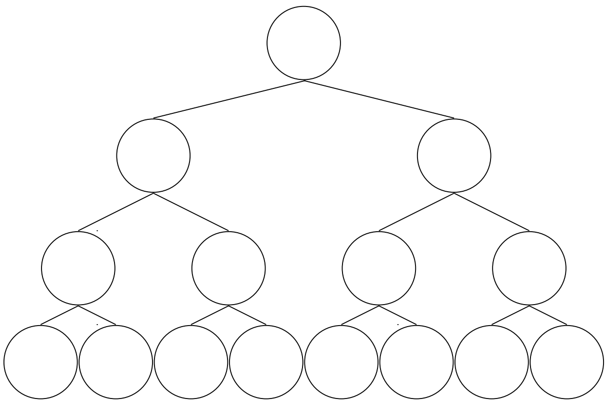


Abbildung 1: Der Rekursionbaum der Ausführung von Mergesort auf A