Lehrstuhl für Informatik I
Algorithmen und Komplexität
Universität Würzburg

Würzburg, den 18. November 2021

Prof. Dr. Alexander Wolff
Felix Klesen, M. Sc.

1. Kurztest zur Vorlesung Algorithmen und Datenstrukturen (Wintersemester 2021/2022)

Bitte beachten Sie die folgenden Hinweise!

1. **Personalien:** Bitte tragen Sie Ihre Daten gut lesbar ein.

Name	
Vorname	
Matrikelnummer	
Übungsgruppe	

2. **Papier:** Tragen Sie Ihre Lösungen direkt unterhalb der Aufgaben ein. Verwenden Sie kein eigenes Papier.
3. Sie dürfen ein einseitig handgeschriebenes DIN-A4-Blatt mit Notizen verwenden. **Weitere Hilfsmittel sind nicht zugelassen.**
4. Schreiben Sie **nicht** mit Bleistift oder einem roten Stift.
5. Die Bearbeitungszeit beträgt 60 Minuten.

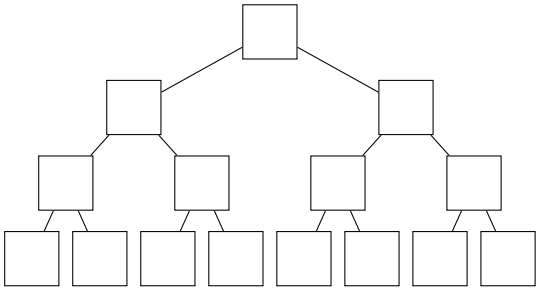
Wir wünschen Ihnen viel Erfolg!

Aufgabe	1	2	3	4	5	6	7	**Gesamt**
Punkte	12	5	11	8	5	4	15	60
Ergebnis								

Aufgabe 1 — Heaps und Prioritätsschlangen

In dieser Aufgabe geht es um Prioritätsschlangen, die mit Hilfe von MaxHeaps implementiert sind. Gehen Sie bei der Lösung der folgenden Aufgaben davon aus, dass die Operationen der Prioritätsschlangen und Heaps so implementiert sind, wie sie in der Vorlesung vorgestellt wurden.

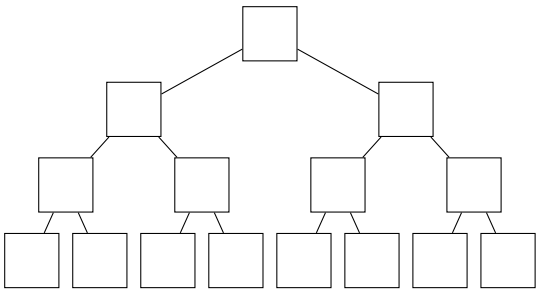
(a) In der Vorlesung haben Sie gesehen, wie sich ein Feld als Baum darstellen lässt. Stellen Sie das Feld $\langle 2, 4, 8, 11, 14, 6, 3, 7, 17, 19 \rangle$ entsprechend dar. Es handelt sich dabei noch nicht um einen Heap. / 1 P.



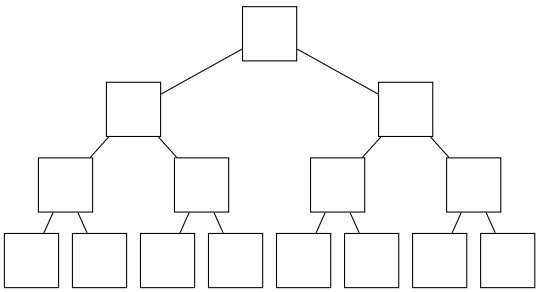
(b) Nutzen Sie nun die aus der Vorlesung bekannte Funktion `BuildMaxHeap` um das Feld aus Teilaufgabe (a) in einen MaxHeap umzuwandeln. In `BuildMaxHeap` wird die Funktion `MaxHeapify` insgesamt 5 mal aufgerufen. / 5 P.

Tragen Sie das Ergebnis nach jedem dieser Aufrufe in die vorgegebenen Bäume ein. Geben Sie jeweils an, für welchen Knoten des Baumes (gegeben durch seinen Index im Feld) `MaxHeapify` aufgerufen wird.

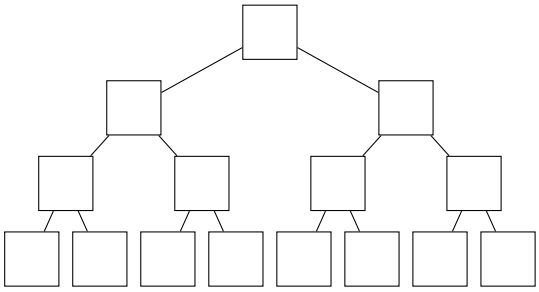
1) `MaxHeapify(____)`



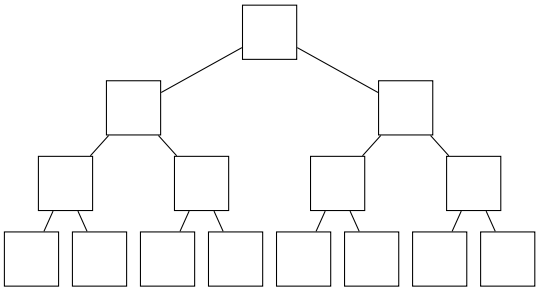
2) `MaxHeapify(____)`



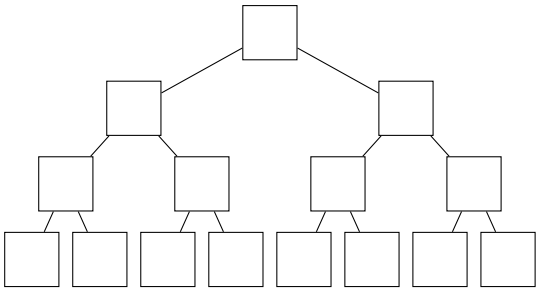
3) `MaxHeapify(____)`



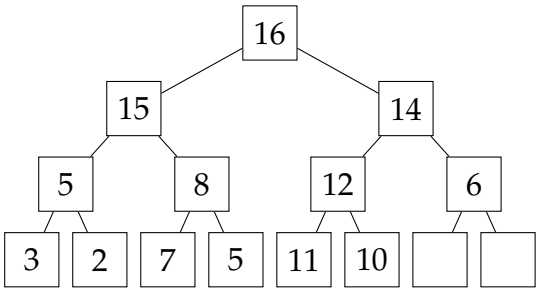
4) `MaxHeapify(____)`

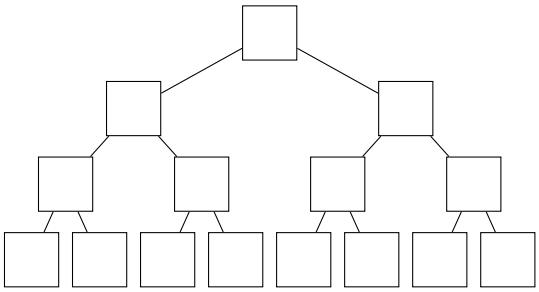


5) `MaxHeapify(____)`

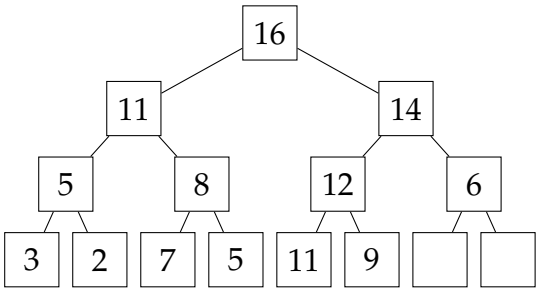


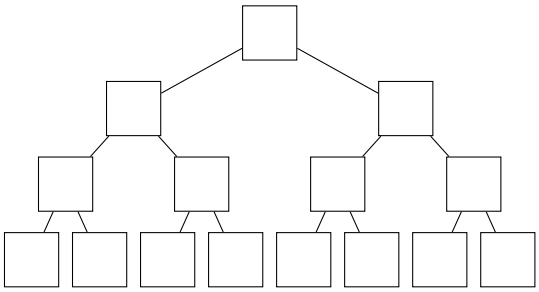
(c) Führen Sie auf dem folgenden Heap `ExtractMax`() aus. Wie sieht der Baum nach diesem Aufruf aus? / 2 P.



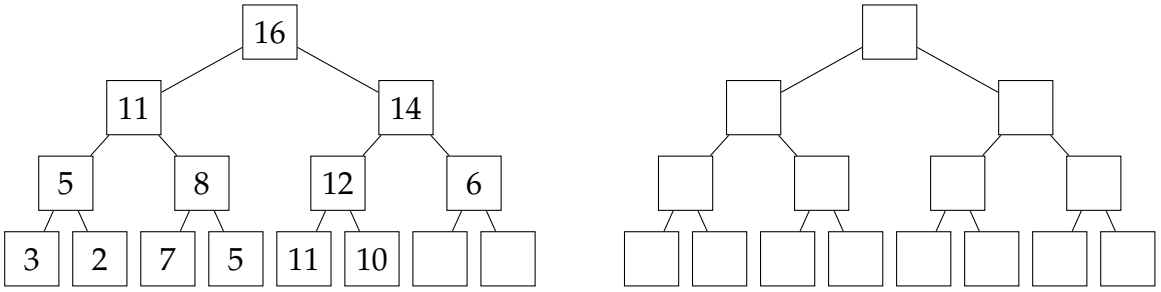


(d) Führen Sie auf dem folgenden Heap `IncreaseKey`(10, 15) aus. Wie sieht der Baum nach diesem Aufruf aus? / 2 P.





(e) Führen Sie auf dem folgenden Heap `Insert`(15) aus. Wie sieht der Baum nach diesem Aufruf aus? / 2 P.



Aufgabe 2 — Pseudocode / 5 P.

Die Methode int `maxRekursiv`(int[] A, int i) soll für $i \in \{1, \ldots, A.length\}$ das größte Element im Teilfeld $A[i..A.length]$ zurückgeben. Die Methode soll *rekursiv* ablaufen und darf weder das Feld A verändern noch ein neues Feld anlegen.

Implementieren Sie die Methode in Pseudocode:

int `maxRekursiv`(int[] A, int i)

Aufgabe 3 — Sortieren

Gegeben sei folgender iterativer Algorithmus, wobei `swap`(F, i, j) im Feld F die Elemente an den Stellen i und j miteinander vertauscht.

```
BackwardsSort(int[] A)
  n = A.length
  for i = 1 to n do
  |  MergeSort(A, i, n)
  |  swap(A, i, n)
```

(a) Zeigen Sie mittels folgender Schleifeninvariante, dass `BackwardsSort(A)` das Eingabefeld A *absteigend* sortiert: / 5 P.

Bei der i*-ten Ausführung des for-Schleifenkopfes gilt:*

(i) A *enthält die gleichen Zahlen wie zu Beginn und*

(ii) $A[1..i-1]$ *enthält die* $(i-1)$ *größten Zahlen aus* A *absteigend sortiert.*

Initialisierung:

Aufrechterhaltung:

Terminierung:

(b) Die Laufzeit von `BackwardsSort`(int[] A) in Abhängigkeit von $n = A.length$ liegt in Θ(_____), denn

/ 3 P.

(c) Nehmen Sie nun an, dass in `BackwardsSort`(int[] A) `InsertionSort` anstelle von `MergeSort` verwendet wird (so dass das gleiche Teilfeld $A[i..n]$ wie beim Aufruf von MergeSort sortiert wird).

Die Laufzeit in Abhängigkeit von $n = A.length$ liegt nun in Θ(_____), denn

/ 3 P.

Aufgabe 4 — Rekursionsgleichungen

Lösen Sie folgende Rekursionsgleichungen mit der Meistermethode, wenn möglich. Geben Sie dazu den entsprechenden Fall der Meistermethode an. Wenn eine Rekursionsgleichung nicht mit der Meistermethode gelöst werden kann, dann geben Sie den Grund dafür an (Sie müssen sie dann nicht weiter lösen).

(a) Sei / 4 P.

$$T(n) = \begin{cases} 4T(\lceil n/4 \rceil) + 8n + 64, & \text{falls } n > 1 \\ 1 & \text{sonst.} \end{cases}$$

Ist bei der Lösung von T die Meistermethode anwendbar?

☐ Ja, nach Fall _____ der Meistermethode ist die Laufzeit $T \in \Theta($_____$)$, denn
☐ Nein, denn

Berechnung bzw. Begründung falls nicht möglich:

(b) Sei / 4 P.

$$T(n) = \begin{cases} 8T(\lfloor n/2 \rfloor) + n^2 \cdot \log_2 n, & \text{falls } n > 1 \\ 1 & \text{sonst.} \end{cases}$$

Ist bei der Lösung von T die Meistermethode anwendbar?

☐ Ja, nach Fall _____ der Meistermethode ist die Laufzeit $T \in \Theta($_____$)$, denn
☐ Nein, denn

Berechnung bzw. Begründung falls nicht möglich:

Aufgabe 5 — O-Notation

/ 5 P.

Sei $f\colon \mathbb{N} \to \mathbb{R}$ eine Funktion. Beweisen oder widerlegen Sie folgende Behauptung. Nutzen Sie dazu die Definition von Ω aus der Vorlesung.

Für $f\colon n \mapsto 4n^2 - 7n \log n$ gilt $f \in \Omega(n^3)$.

Aufgabe 6 — Logarithmische Laufzeitklassen

/ 4 P.

Man kann $O(\log n)$ statt $O(\log_3 n)$ schreiben. Warum ist das möglich, ohne ungenau zu werden? Begründen Sie Ihre Antwort.

Name: **Matrikelnummer:**

Aufgabe 7 — Algorithmen

Geben Sie zu den folgenden Algorithmen jeweils an, was diese zurückgeben und in welcher Laufzeitklasse sie liegen.

(a) `Algorithmus1`(int n) / 5 P.

```
if n < 2 then
    return −1
a = 2
for i = 3 to n do
    b = 0
    for j = 2 to i − 1 do
        if j teilt i then
            b = b + 1
    if b == 0 then
        a = i
return a
```

`Algorithmus1` gibt folgendes zurück:

_____

_____

Die Laufzeit von `Algorithmus1` liegt in $\Theta($_____$)$ in Abhängigkeit von n.

(b) `Algorithmus2`(int n) / 5 P.

```
sum = 0
i = 1
while i · i · i ≤ n do
    sum = sum + i
    i = i + 1
return sum
```

`Algorithmus2` gibt folgendes zurück:

_____

_____

Die Laufzeit von `Algorithmus2` liegt in $\Theta($_____$)$ in Abhängigkeit von n.

(c) `Algorithmus3`(int[] A, int i = 1) / 5 P.

```
n = A.length
if A[i] == i then
    return true
if i == n then
    return false
return Algorithmus3(A, i + 1)
```

`Algorithmus3` gibt folgendes zurück:

_____

_____

Die Laufzeit von `Algorithmus3` liegt in Θ(_____) in Abhängigkeit von $n = A.length$.