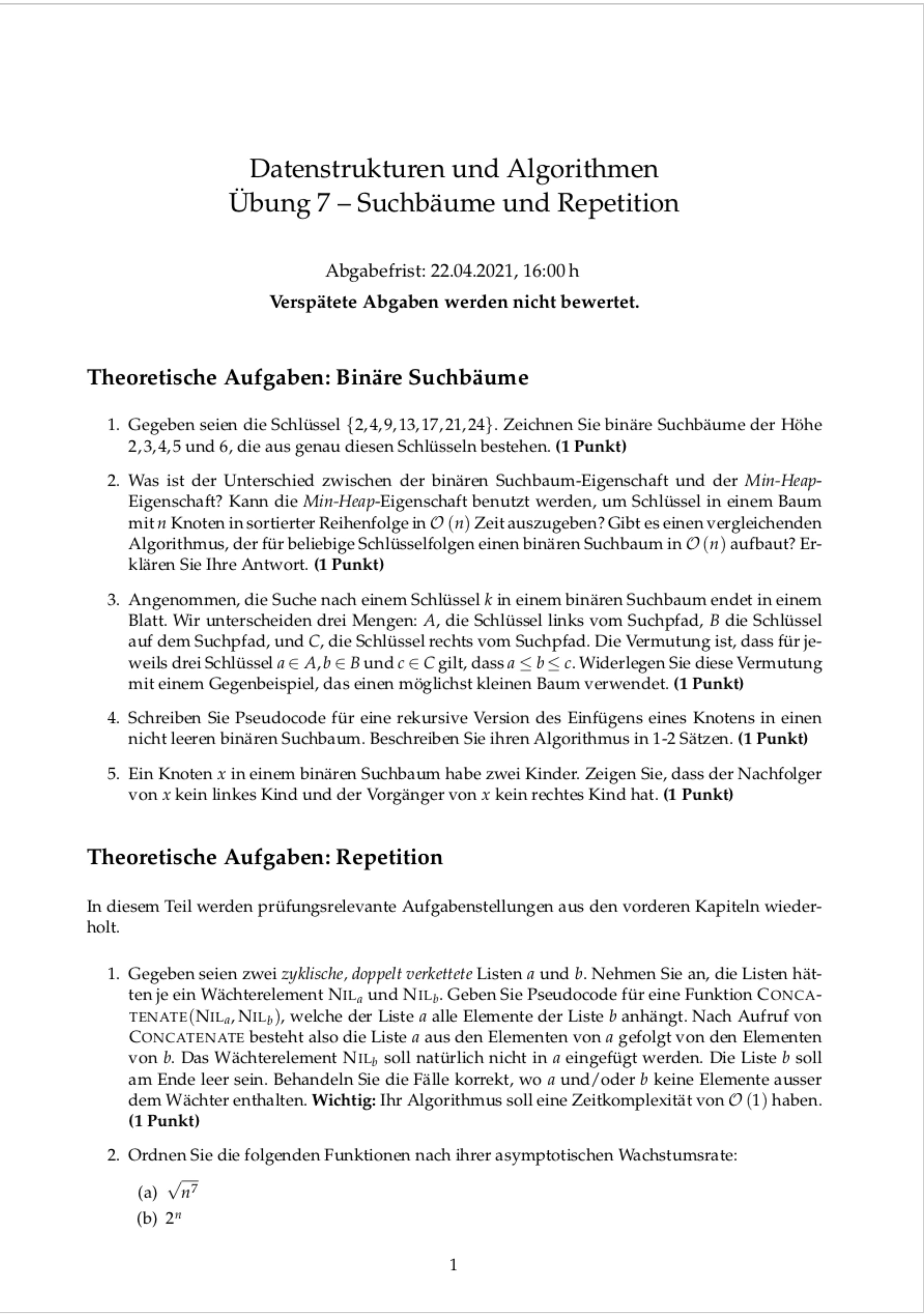
DA-7

Lorenz Pfäffli 18-925-230
Lukas Ingold 20-123-998



Datenstrukturen und Algorithmen
Übung 7 – Suchbäume und Repetition

Abgabefrist: 22.04.2021, 16:00 h

Verspätete Abgaben werden nicht bewertet.

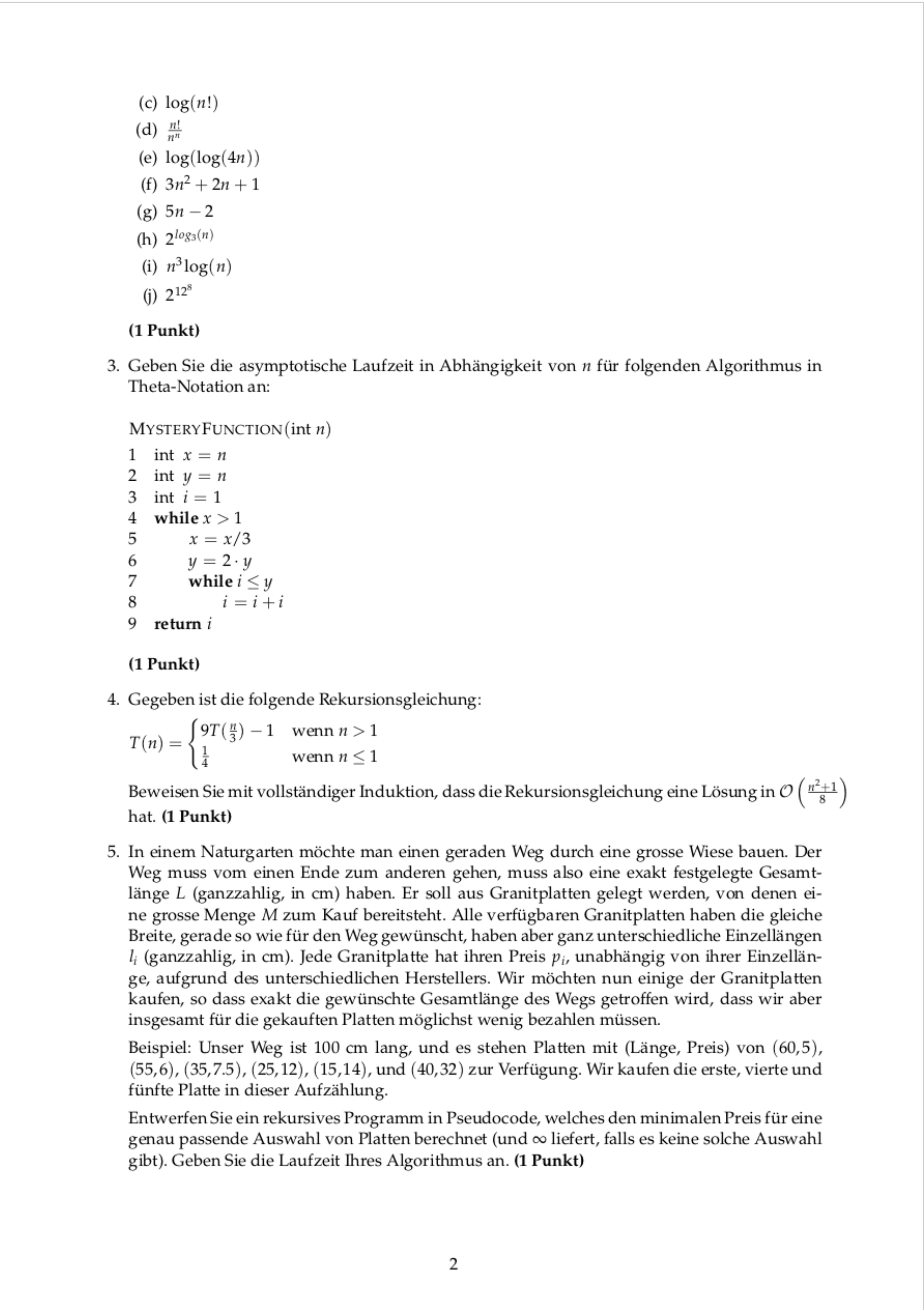
Theoretische Aufgaben: Binäre Suchbäume

1. Gegeben seien die Schlüssel $\{2,4,9,13,17,21,24\}$. Zeichnen Sie binäre Suchbäume der Höhe 2,3,4,5 und 6, die aus genau diesen Schlüsseln bestehen. **(1 Punkt)**
2. Was ist der Unterschied zwischen der binären Suchbaum-Eigenschaft und der *Min-Heap*-Eigenschaft? Kann die *Min-Heap*-Eigenschaft benutzt werden, um Schlüssel in einem Baum mit n Knoten in sortierter Reihenfolge in $\mathcal{O}(n)$ Zeit auszugeben? Gibt es einen vergleichenden Algorithmus, der für beliebige Schlüsselfolgen einen binären Suchbaum in $\mathcal{O}(n)$ aufbaut? Erklären Sie Ihre Antwort. **(1 Punkt)**
3. Angenommen, die Suche nach einem Schlüssel k in einem binären Suchbaum endet in einem Blatt. Wir unterscheiden drei Mengen: A, die Schlüssel links vom Suchpfad, B die Schlüssel auf dem Suchpfad, und C, die Schlüssel rechts vom Suchpfad. Die Vermutung ist, dass für jeweils drei Schlüssel $a \in A, b \in B$ und $c \in C$ gilt, dass $a \leq b \leq c$. Widerlegen Sie diese Vermutung mit einem Gegenbeispiel, das einen möglichst kleinen Baum verwendet. **(1 Punkt)**
4. Schreiben Sie Pseudocode für eine rekursive Version des Einfügens eines Knotens in einen nicht leeren binären Suchbaum. Beschreiben Sie ihren Algorithmus in 1-2 Sätzen. **(1 Punkt)**
5. Ein Knoten x in einem binären Suchbaum habe zwei Kinder. Zeigen Sie, dass der Nachfolger von x kein linkes Kind und der Vorgänger von x kein rechtes Kind hat. **(1 Punkt)**

Theoretische Aufgaben: Repetition

In diesem Teil werden prüfungsrelevante Aufgabenstellungen aus den vorderen Kapiteln wiederholt.

1. Gegeben seien zwei *zyklische, doppelt verkettete* Listen a und b. Nehmen Sie an, die Listen hätten je ein Wächterelement NIL_a und NIL_b. Geben Sie Pseudocode für eine Funktion CONCATENATE($\text{NIL}_a, \text{NIL}_b$), welche der Liste a alle Elemente der Liste b anhängt. Nach Aufruf von CONCATENATE besteht also die Liste a aus den Elementen von a gefolgt von den Elementen von b. Das Wächterelement NIL_b soll natürlich nicht in a eingefügt werden. Die Liste b soll am Ende leer sein. Behandeln Sie die Fälle korrekt, wo a und/oder b keine Elemente ausser dem Wächter enthalten. **Wichtig:** Ihr Algorithmus soll eine Zeitkomplexität von $\mathcal{O}(1)$ haben. **(1 Punkt)**
2. Ordnen Sie die folgenden Funktionen nach ihrer asymptotischen Wachstumsrate:
   (a) $\sqrt{n^7}$
   (b) 2^n



   (c) $\log(n!)$
   (d) $\frac{n!}{n^n}$
   (e) $\log(\log(4n))$
   (f) $3n^2 + 2n + 1$
   (g) $5n - 2$
   (h) $2^{log_3(n)}$
   (i) $n^3 \log(n)$
   (j) 2^{12^8}

   (1 Punkt)
3. Geben Sie die asymptotische Laufzeit in Abhängigkeit von n für folgenden Algorithmus in Theta-Notation an:

```
MYSTERYFUNCTION(int $n$)
int $x = n$
int $y = n$
int $i = 1$
while $x > 1$
    $x = x/3$
    $y = 2 \cdot y$
    while $i \leq y$
        $i = i + i$
return $i$
```

   (1 Punkt)
4. Gegeben ist die folgende Rekursionsgleichung:

$$T(n) = \begin{cases} 9T(\frac{n}{3}) - 1 & \text{wenn } n > 1 \\ \frac{1}{4} & \text{wenn } n \leq 1 \end{cases}$$

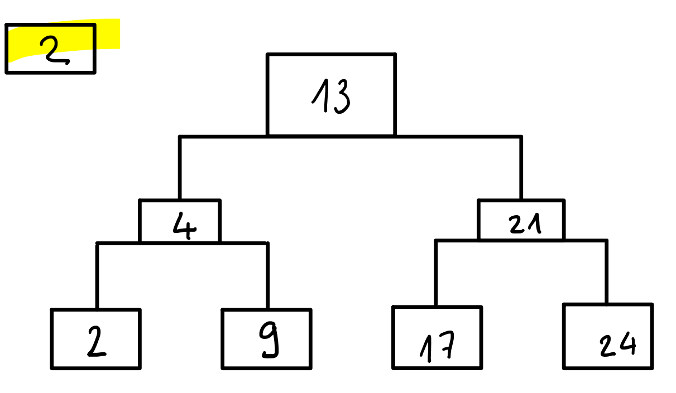
   Beweisen Sie mit vollständiger Induktion, dass die Rekursionsgleichung eine Lösung in $\mathcal{O}\left(\frac{n^2+1}{8}\right)$ hat. **(1 Punkt)**
5. In einem Naturgarten möchte man einen geraden Weg durch eine grosse Wiese bauen. Der Weg muss vom einen Ende zum anderen gehen, muss also eine exakt festgelegte Gesamtlänge L (ganzzahlig, in cm) haben. Er soll aus Granitplatten gelegt werden, von denen eine grosse Menge M zum Kauf bereitsteht. Alle verfügbaren Granitplatten haben die gleiche Breite, gerade so wie für den Weg gewünscht, haben aber ganz unterschiedliche Einzellängen l_i (ganzzahlig, in cm). Jede Granitplatte hat ihren Preis p_i, unabhängig von ihrer Einzellänge, aufgrund des unterschiedlichen Herstellers. Wir möchten nun einige der Granitplatten kaufen, so dass exakt die gewünschte Gesamtlänge des Wegs getroffen wird, dass wir aber insgesamt für die gekauften Platten möglichst wenig bezahlen müssen.

   Beispiel: Unser Weg ist 100 cm lang, und es stehen Platten mit (Länge, Preis) von (60,5), (55,6), (35,7.5), (25,12), (15,14), und (40,32) zur Verfügung. Wir kaufen die erste, vierte und fünfte Platte in dieser Aufzählung.

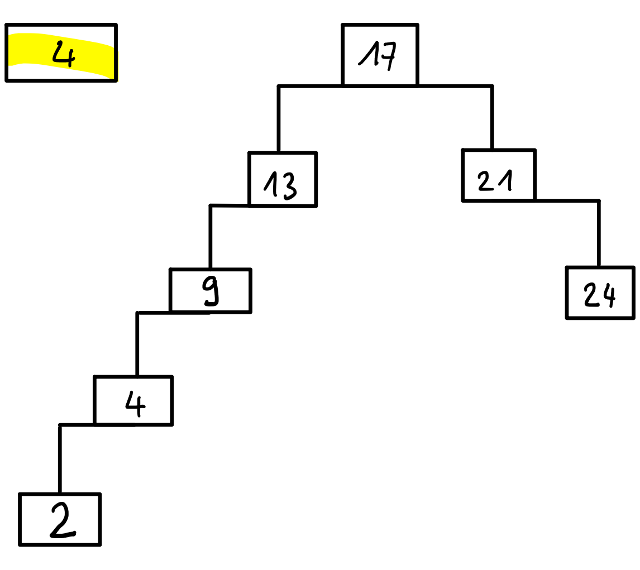
   Entwerfen Sie ein rekursives Programm in Pseudocode, welches den minimalen Preis für eine genau passende Auswahl von Platten berechnet (und ∞ liefert, falls es keine solche Auswahl gibt). Geben Sie die Laufzeit Ihres Algorithmus an. **(1 Punkt)**

1.) {2, 4, 9, 13, 17, 21, 24}

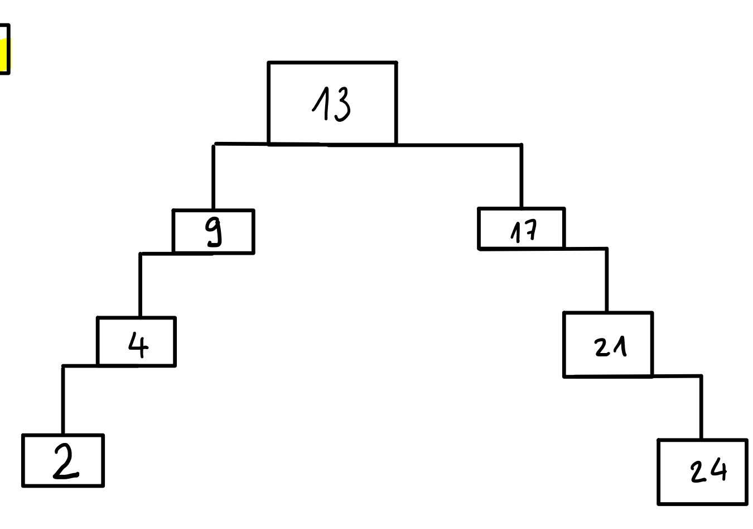
2



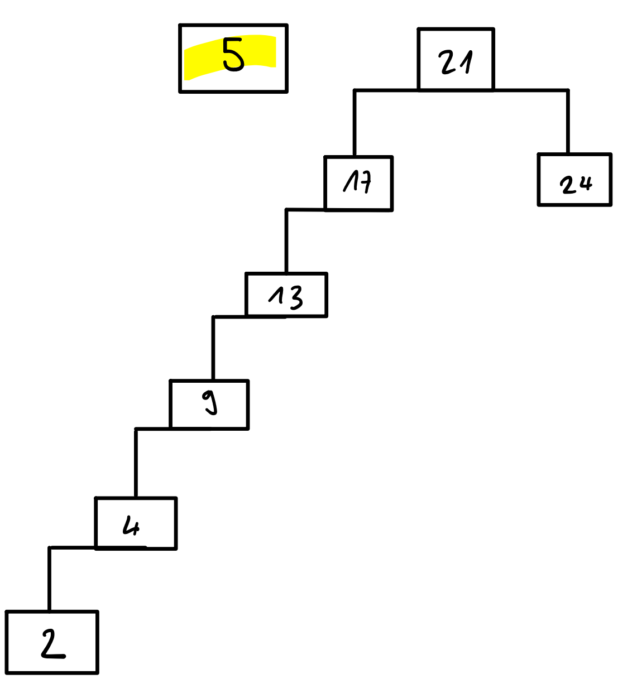
4



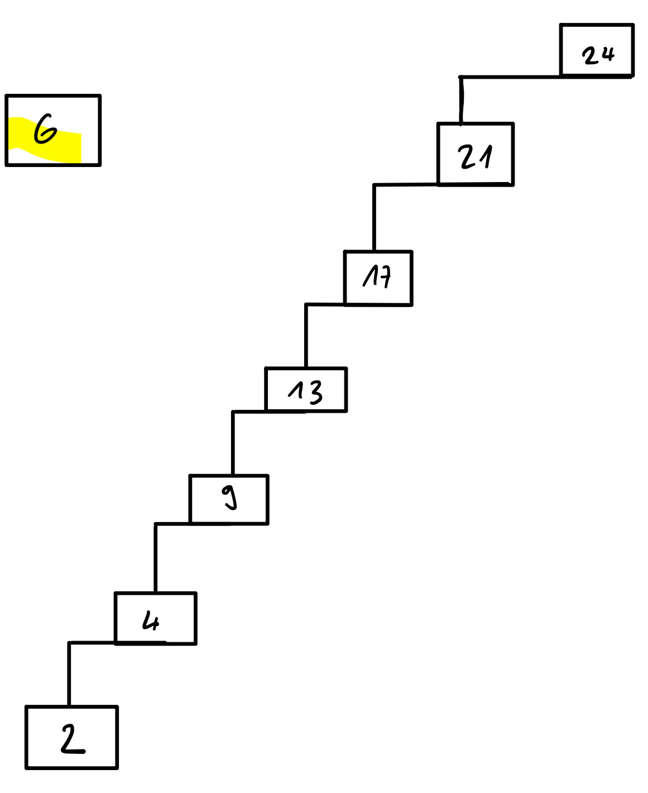
3



5



6

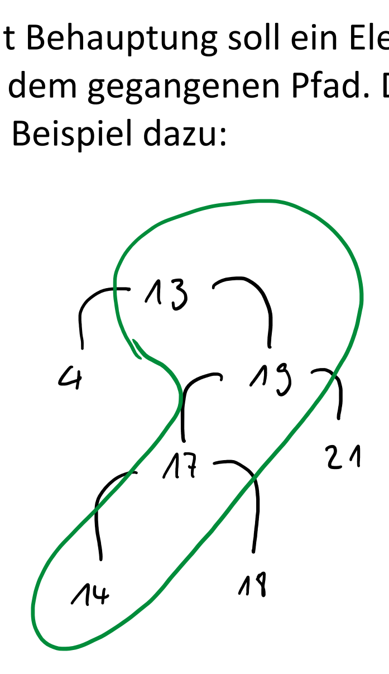


2.) Die Min-Heap-Bedingung bedeutet das die Schlüssel der Kinder eines Knotens sind stets grösser als die Schlüssel der Elternknoten.

Bei dem Binärbaum sind im Unterschied dazu die Kinder auf der linken Seite immer kleiner gleich dem Elternknoten und das rechte Kind grösser gleich dem Elternknoten.

Sortierte Ausgabe mit Min-Heap: Dies ist nicht möglich, da zu bestimmten Knoten nichts über die beiden Teilbäume ausgesagt werden kann (anders als beim binären Suchbaum).

3.) Laut Behauptung soll ein Element mit zB. Schlüssel 18 grösser sein als zB. Ein Element mit Schlüssel 19 auf dem gegangenen Pfad. Dies ist aber nicht der Fall, somit kann die Behauptung nicht stimmen. Hier das Beispiel dazu:



4.)

```
Insert(root, key){
    If (v!= null){
        If(key<root.getKey()){
            Insert(left(root),key) //Knoten wird im linken Teilbaum gespeichert
        }
        Else{
            Insert(right(root),key) //Knoten wird im rechten Teilbaum gespeichert
        }
    }
    Else{
        Füge neuen Knoten mit Schlüssel Key ein
    }
}
```

5.) Def.

Nachfolger: Das nächst zu Sortierende Element
Linker Teilbaum: Element <= Eltern Element
Ende: ein Nachfolger von x kann kein linkes Element haben => Der frühere Nachfolger von x wäre sonst kein Nachfolger mehr

Wenn man annimmt es gäbe ein linkes Kind vom Nachfolger von x. Dieses müsste in der Reihenfolge der Sortierung vor dem Nachfolger von x liegen. Da dieses Element aber nicht x ist und weil x als Vorgänger definiert ist, folgt daraus ein Widerspruch. => es kann kein linkes Kind geben => Der Vorgänger von x kann kein rechtes Kind haben.

Repetition:

1.)

```
CONCARTENATE(NIL_a,NIL_b){
    T1 = next(NIL_b)
    T2 = last(NIL_b)

    If(T1 != NIL_b){
        Next(last(NIL_a)) = T1 //Zeigt auf anfang von b
        Last(T1) = last(NIL_a) //verlinkt letztes element von a
        Next(T2) = NIL_a
        Last(NIL_a) = T2
    }
}
```

2.) A.) $\Theta(n^{7/2})$
B.) $\Theta(2^n)$
C.) $\Theta(n*\log(n))$
D.)
E.)
F.)
G.) $\Theta(n)$
H.)
I.)
J.) $\Theta(1)$

3.) $\Theta(n)$