Übungsblatt 03 – Datenstrukturen und Algorithmen

Name: Taha Darende

Matrikelnummer: 3724493

Name: Öznur Gencoglu

Matrikelnummer: 3682221

Name: Jeannine Renner

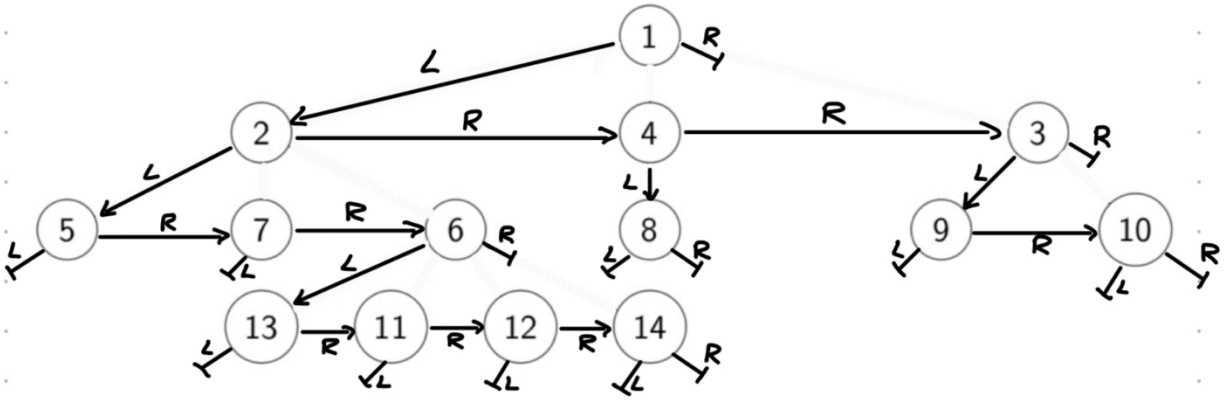
Matrikelnummer: 3724419

Aufgabe 1:

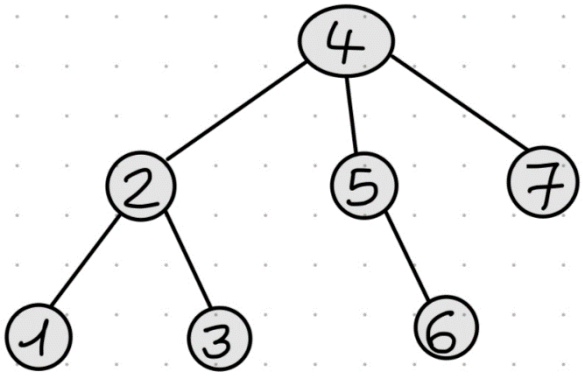
a) Der Baum B_0 hat die Wurzel 1.
Er hat 14 Knoten, 13 Kanten und 5 Elternknoten. Diese sind 1, 2, 3, 4 und 6.
Alle Knoten außer die Wurzel sind Kindknoten.
Alle Elternknoten sind innere Knoten, also 1, 2, 3, 4 und 6.
Als Blatt zählen alle Knoten ohne Kinder: 5, 7, 8, 9, 10, 11, 12, 13 und 14.
Der Baum B_0 besitzt 9 Pfade.
Das Niveau vom Baum beträgt: Niveau 3.
Die Höhe beträgt 4. (Niveau + 1)
Der Baum ist ein 4-är Baum, das bedeutet, dass der Baum höchstens 4 Kinder hat.
Der Baum ist nicht geordnet.

b) Bei B_0 handelt es sich nicht um einen Suchbaum.
Damit ein Baum als Suchbaum gilt, müssen alle Schlüsselwerte im linken Teilbaum kleiner als der Schlüssel und alle Schlüsselwerte im rechten Teilbaum größer sein.
Da dies nicht auf B_0 zutrifft, ist B_0 kein Suchbaum.

c)

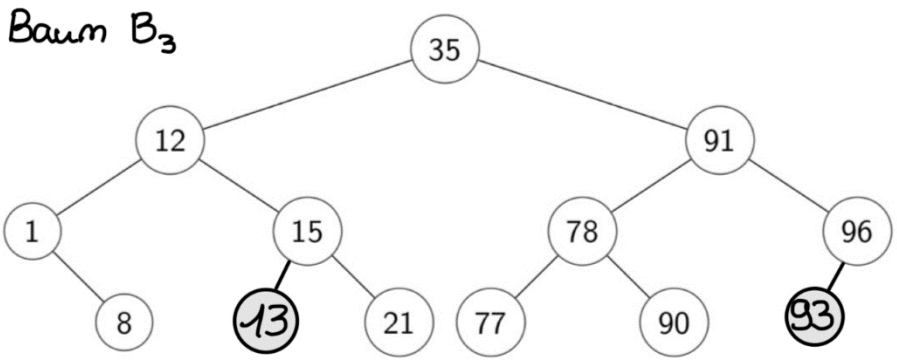


d)

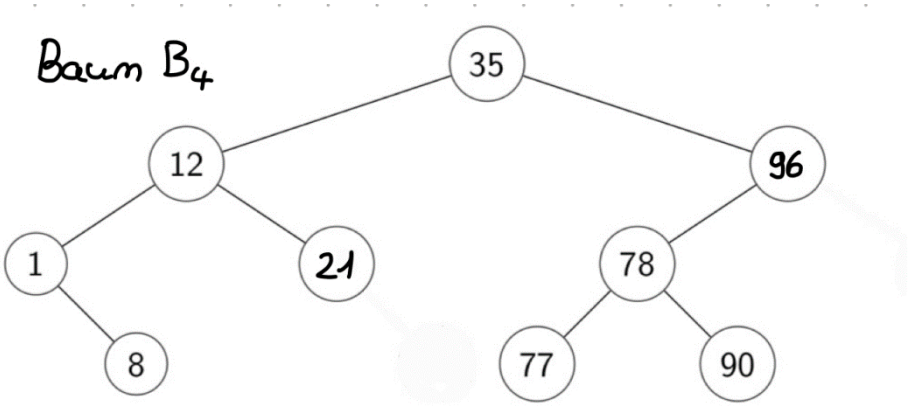


Aufgabe 2:

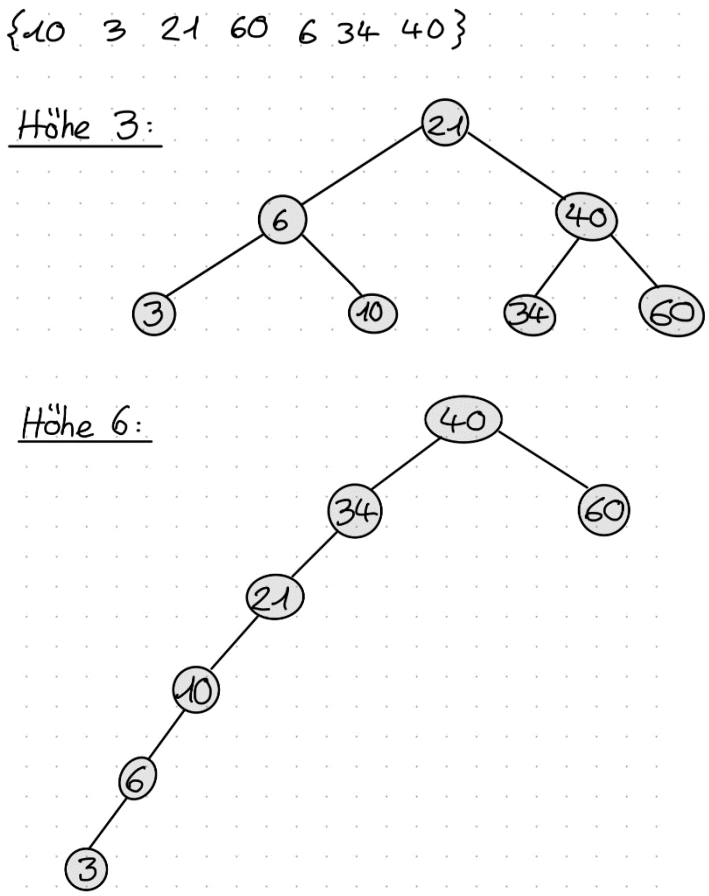
a)



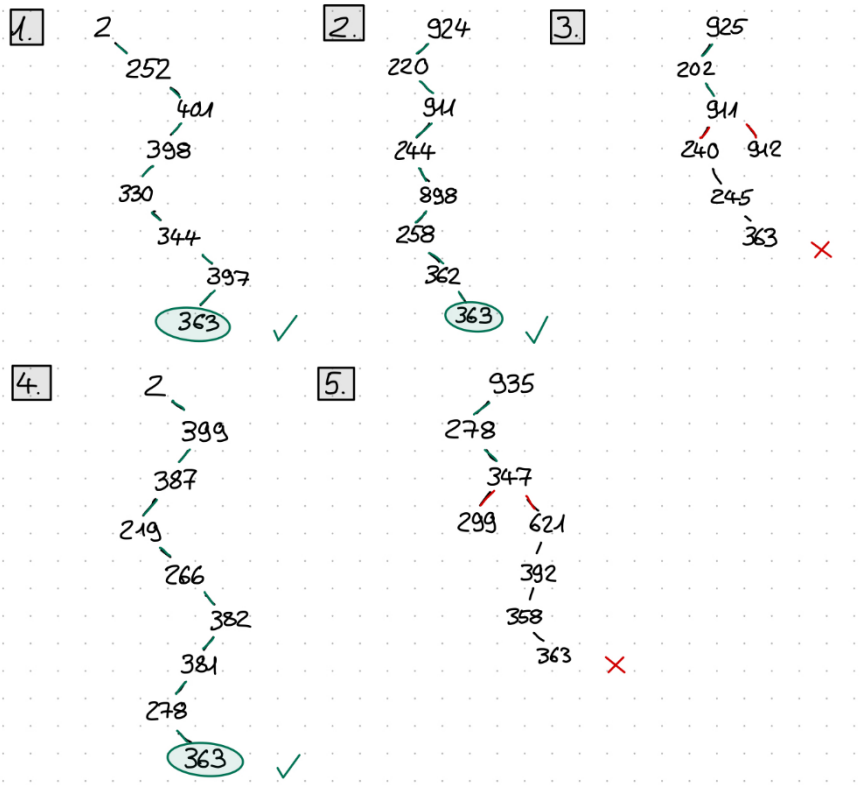
b)



c)



d)



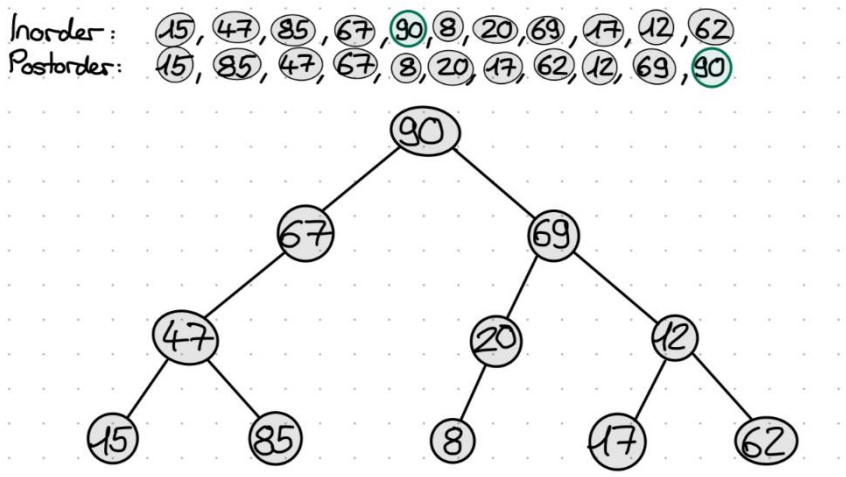
➔ Die Sequenzen 3. Und 5. Können keine Knotenfolgen sein, da es bei beiden Sequenzen keinen eindeutigen Pfad gibt, da man schaut, ob die Zahl kleiner/größer ist.

e) Es würde weniger Sinn machen, anstelle einer Liste einen Suchbaum zu verwenden, da die Liste schon alphabetisch geordnet ist. Dadurch hätte man einen schnellen Zugriff auf die Elemente. Hinzu kommt, dass der Aufwand eines Suchbaums viel höher wäre und der Pfad des Baums viel zu lang wäre, wodurch man mehr Speicherplatz benötigen würde. Hinzu kommt, dass beide eine Zeitkomplexität von O(log(n)) haben, also wäre der Suchbaum nicht schneller und man müsste den Baum erst erstellen, anders als bei der Liste.

Aufgabe 3:

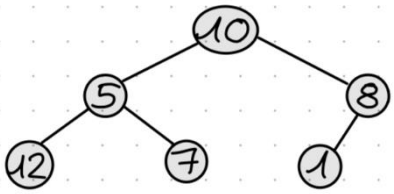
a) Preorder: 35, 12, 1, 8, 15, 21, 91, 78, 77, 90, 96
Inorder: 1, 8, 12, 15, 21, 35, 77, 78, 90, 91, 96

b)



➔ Zuerst muss man die Wurzel finden. Diese findet man ganz einfach bei Pre- oder Postorder. Bei Preorder ist die Wurzel die erste Zahl und bei Postorder die letzte. Bei uns ist es daher die 90. Diese können wir auch bei der Inorder markieren, dadurch kann man sehen, ob die Werte sich im linken oder rechten Teilbaum befinden. Alle Werte links von der 90 befinden sich im linken Teilbaum und alle Werte rechts dementsprechend im rechten Baum.
Im nächsten Schritt geht man die Werte von rechts nach links bei Postorder durch, sucht den Wert bei der Inorder-Traversierung und schaut, ob es sich rechts oder links von der Wurzel befindet.
Hat man einen Wert, der zwischen zwei anderen liegt (z.B. 17, liegt bei Inorder-Traversierung zwischen den Werten 69 und 12), dann muss diese dazwischen liegen. Es gilt trotzdem, dass 17 rechts von 69 und links von 12 liegt.
Man geht so alle Werte von rechts nach links bei der Postorder-Traversierung durch. Dadurch lässt sich der Baum einfach rekonstruieren.

c) Nein, die Aussage stimmt nicht. Bei der Preorder-Traversierung befindet sich die Wurzel ganz links und bei der Postorder-Traversierung ganz rechts. Dadurch kann man die Wurzel schnell finden.
Allerdings kann man nicht eindeutig identifizieren, ob der Wert links oder rechts von der Wurzel liegt. Dafür bräuchte man die Inorder-Traversierung.
Aufgrund dessen lässt sich der Baum mithilfe von Pre- und Postorder-Reihenfolgen nicht eindeutig rekonstruieren.
(siehe Beispiel auf der nächsten Seite)



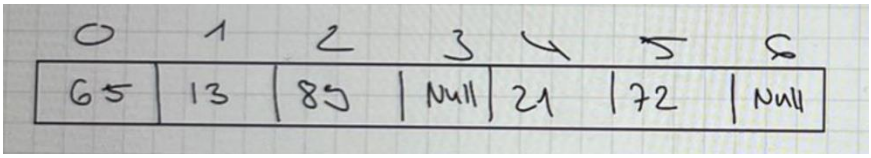
Preorder: Wurzel 10 - 5 - 12 - 7 - 8 - 1

Postorder: 12 - 7 - 5 - 1 - 8 - Wurzel 10

Aufgabe 4:

a) Für B_6 hätten folgende Arrays dienen können:
   1. [65, 13, 89, 21, 72]
   2. [65, 13, 89, 72, 21]

b) Knoten an Array Index b abgelegt



Für linke Kindknoten des Knotens befindet sich am Index 2b + 1 und für rechte Kindknoten des Knotens in 2b + 2 im Array.

z.B. linke Kindknoten von 65 ist 13:

2b + 1 = 2 * 0 + 1 = 1 -> Also hat 13 Index 1.

Für 89:

2b + 2 = 2 * 0 + 2 = 2 -> 89 hat Index 2.

c) $\lceil \log_2 (b+1) \rceil$
Log₂, da sich der Baum jeweils immer verdoppelt und b+1, da b für den Index steht und +1, da das Array mit dem Index 0 startet. Somit kommt man auf die genannte Formel.
Beispiel:

65 ist Wurzelknoten mit Index 0
$\log_2(0+1) = \log_2(1) = 0$

für 13: $\log_2(1+1) = \log_2(2) = 1$