Übungsblatt 2

Beispiel 2.1

Erklären Sie die Unterschiede zwischen Array und einer einfach verketteten Liste. Vergleichen Sie dabei auch den Aufwand nach O-Notation für die Grundoperationen Einfügen, Suchen und Löschen eines Elements.

Beispiel 2.2

Geben sie einen einfachen Algorithmus zum Einfügen eines Elements am Ende einer Liste in Pseudo-Code an. Bedenken Sie, dass die Liste zu Beginn auch leer sein kann.

Beispiel 2.3

Ein Stack oder auch Kellerspeicher ist eine Datenstruktur, in die nur an einem Ende (top) Objekte eingefügt und wieder entfernt werden können, d.h. das zuletzt eingefügte Datenobjekt wird als erstes wieder entfernt. LIFO -> Last In – First Out. Diese Datenstruktur stellt somit eine besondere Art einer Liste dar und kann einfach mit einer einfach-verketteten Liste realisiert werden. Implementieren Sie die folgenden Stack-Operationen der Klasse Stack in Pseudo-Code, mit Hilfe der angegebenen **StackNode** Typ-Deklaration:

- Die Operation **Push** „legt“ ein Datenobjekt (z.B. eine Integer Zahl) auf den Stack.
- Die Operation **Pop** „holt“ das oberste Datenobjekt vom Stack.
- Mit der Operation **IsEmpty** kann geprüft werden, ob der Stack leer ist.

```
class Stack {
    StackNode Top;
}

class StackNode {
    StackNode NextNode;
    int Data;
}
```

Beispiel 2.4

Vervollständigen Sie den folgenden Lückentext über binäre Bäume und begründen Sie die Eigenschaften:

Ein binärer Baum mit **n** Knoten hat:

- höchstens _____ Blattknoten
- mindestens _____ innere Knoten
- mindestens Tiefe _____

Beispiel 2.5

Bestimmen Sie im folgenden Baum die Typen der Knoten. Füllen Sie dazu die vorgegebene Tabelle aus. Zeichnen Sie im Baum die Niveaus und die Höhe ein. Überlegen Sie sich welche Ordnung dieser Baum haben könnte, und begründen Sie ihre Entscheidung.

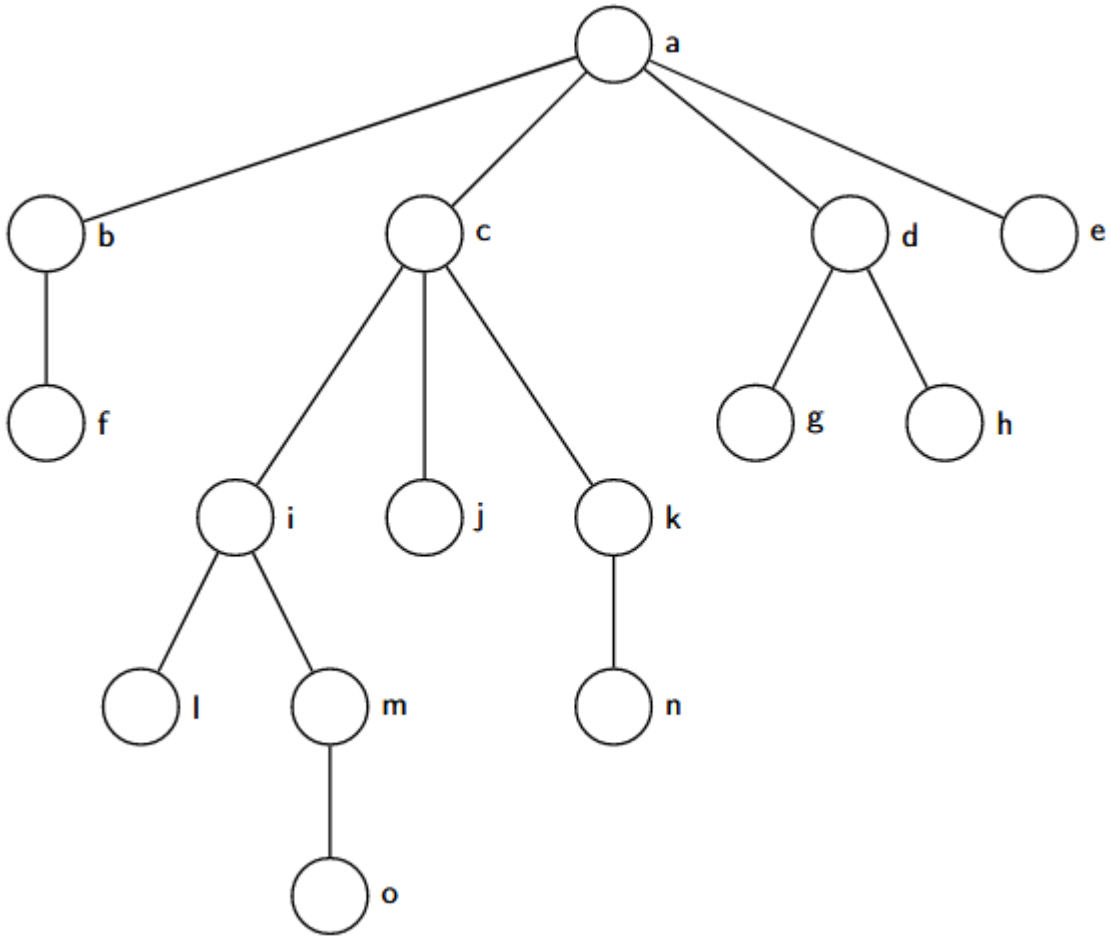


Abbildung 1 - Baum

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
Wurzel															
Blatt															
Innerer_Knoten															

Beispiel 2.6

Eine mögliche Repräsentation für mathematische Ausdrücke ist ein abstrakter Syntaxbaum. Das Erstellen von Syntaxbäumen aus Texten ist ein wesentlicher Bestandteil von Compilern und Interpretern. Wichtig ist das richtige Setzen der Klammern, da die beiden Teilbäume eines Knotens vor diesem Knoten ausgewertet werden müssen. Das Setzen der Klammern kann durch das Umschließen der einzelnen Ausdrücke der Teilbäume erreicht werden. Wird nun der Baum korrekt traversiert, erhält man (in diesem Bsp.) einen mathematischen Ausdruck.

Welche Durchlaufordnung muss hier angewandt werden?
Wie lautet der mathematische Ausdruck und dessen Ergebnis, der sich durch die Traversierung des Syntaxbaumes ergibt?

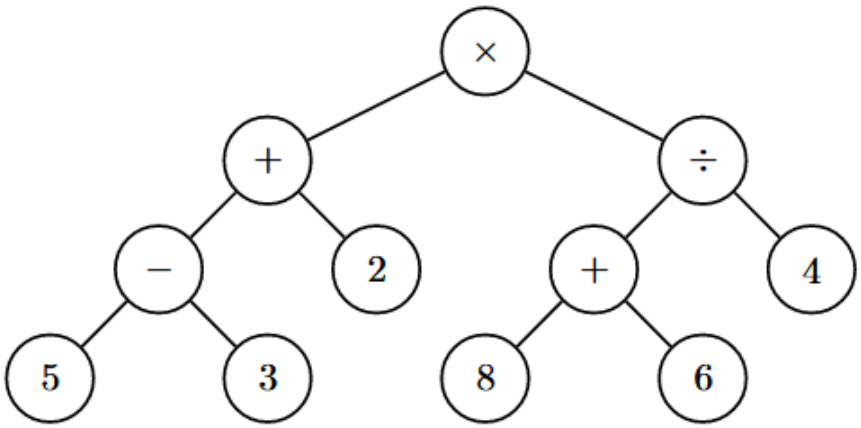


Abbildung 2 - Abstrakter Syntaxbaum

Beispiel 2.7

Traversieren Sie die folgenden Bäume jeweils in Preorder, Inorder und Postorder.

a)

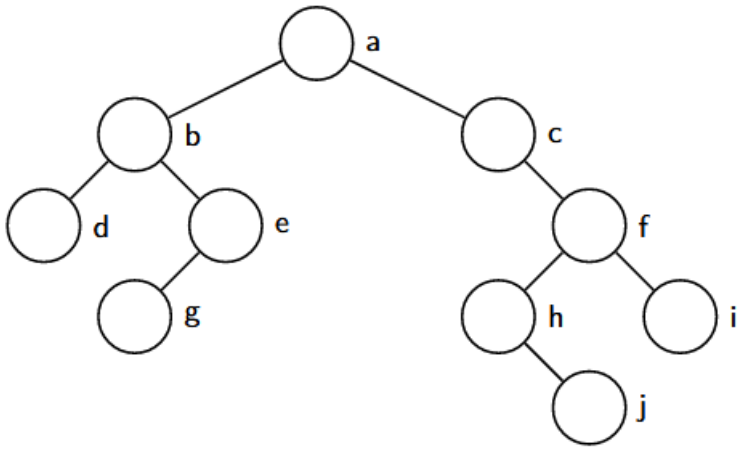


Abbildung 3 - Baum

b)

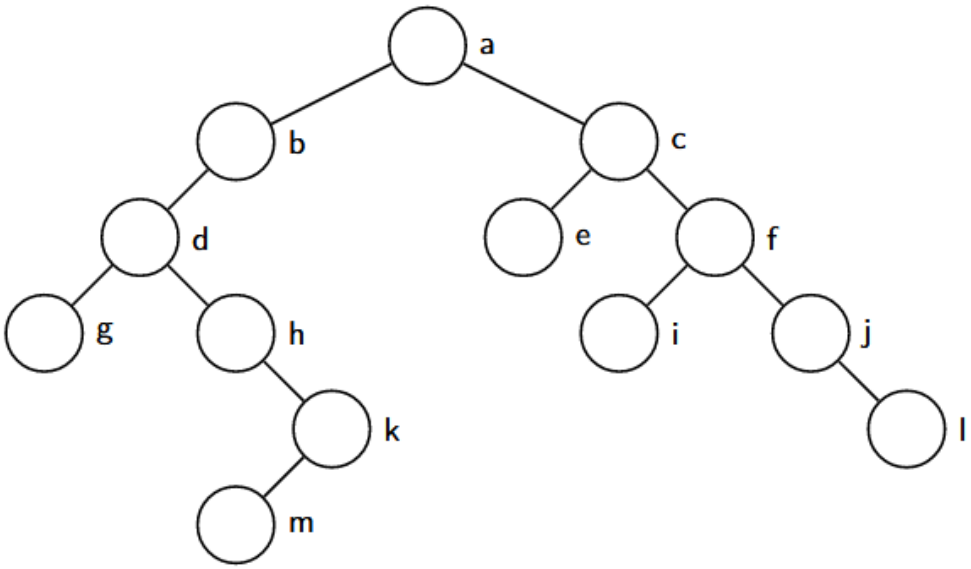


Abbildung 4 - Baum

Beispiel 2.8

Ein Binärer Heap (oft einfach nur als Heap bezeichnet) ist eine Datenstruktur aus der Informatik zum effizienten Sortieren von Elementen. Des Weiteren wird bei Verwendung der Ordnungsrelation < bzw. > ein Heap als Min-Heap bzw. Max-Heap bezeichnet (Quelle: https://de.wikipedia.org/wiki/Bin%C3%A4rer_Heap). Fügen Sie in den folgenden Min-Heap den Schlüssel 10 ein und zeichnen Sie den daraus resultierenden Baum. Geben Sie weiters alle notwendigen Schritte an.

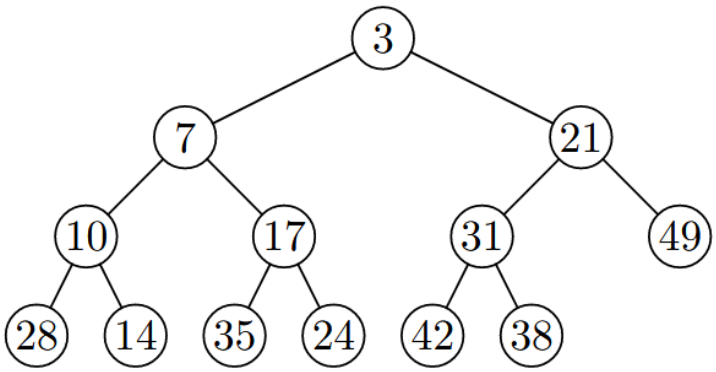


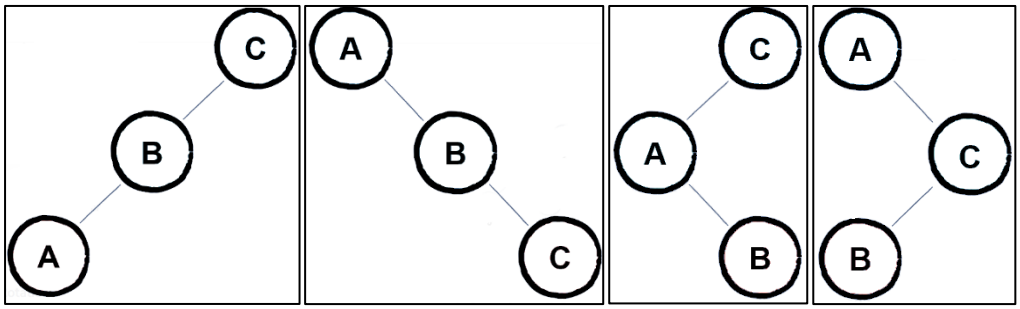
Abbildung 5 - Baum

Beispiel 2.9

Wann gilt ein binärer Baum als ausgeglichen? Geben Sie die Definition an und skizzieren Sie einen normalen binären Baum, sowie einen AVL Baum, um die Definition zu illustrieren.

Beispiel 2.10

Bestimmen Sie für jeden Knoten der vier nachfolgenden Bäume den Balance Faktor und wenden Sie die passende Rotations-Technik an, um AVL-Bäume zu erzeugen (R-Rotation, L-Rotation, LR-Rotation, RL-Rotation). Skizzieren Sie die dafür notwendigen Schritte.



Beispiel 2.11

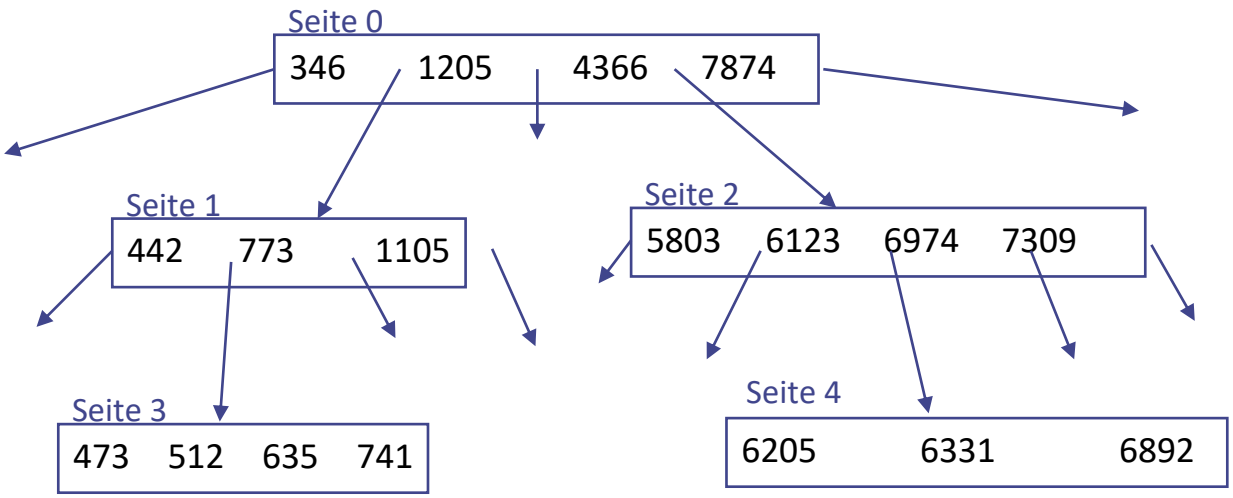
Geben Sie an welche der folgenden Aussagen entweder für B-Baum oder einen B*-Baum oder für beide Baumarten gelten.

Aussage	B-Baume	B*Baum
Schlüssel können sich wiederholen.		
Blattknoten und interne Knoten können Daten speichern.		
Suche ist langsamer.		
Blattknoten können nicht verlinkt werden.		
Löschen von Daten ist einfach		
Löschen von Daten ist komplex und langsam.		

Beispiel 2.12

Fügen Sie in den folgenden B-Baum mit m=2 (also min. 2 und maximal 4 Datensätze pro Seite) folgende Datensätze hintereinander ein:

6501, 450, 6733, 627



Wie viele Seitenzugriffe und wie viele Schlüsselvergleiche sind nun notwendig, um den Datensatz mit Key 6733 zu finden?

Beispiel 2.13

Bilden Sie aus dem folgenden Datensatz einen Quadtree, welcher für die Segmentierung das Alter und ein fiktives Bruttoeinkommen berücksichtigt.

Name	Alter	Bruttoeinkommen
A	25	60
B	50	120
C	25	400
D	45	60
E	70	110
F	45	350
G	50	75
H	85	140
I	50	275
J	50	100
K	30	260

Beispiel 2.14

Zeichnen Sie einen kd-Tree (d.h. den binären Suchbaum & den partitionierten Raum) mit den folgenden Punkten:
(30,40), (5,25), (10,12), (70,70), (50,30), (35,45)

Beispiel 2.15

Optimieren Sie den kd-Tree aus Bsp. 2.14 und geben Sie die optimierten Versionen des binären Suchbaumes & des partitionierten Raumes an.