Datenstrukturen

Übungen

Version vom 19. Oktober 2019

Aufgabe 1.1

Was ist ein Datentyp?

Aufgabe 1.2

Zähle drei einfache (primitive) Datentypen der Programmiersprache Python auf.

Aufgabe 1.3

Welche Laufzeitkomplexität haben die folgenden Operationen für den Python-Datentyp `list`? Gehe davon aus, dass die gegebene Liste `L` die Länge n hat. Der Index `i` ist jeweils ein gültiger Index ($0 \leq i < n$).

(a) `L[i]`

(b) `L.pop(i)`

(c) `L.pop()`

(d) `L.append(77)`

(e) `L.insert(i, 77)`

(f) `for x in L: ...`

Aufgabe 1.4

Welche Laufzeitkomplexität haben die folgenden Operationen für den Python-Datentyp `dict`? Gehe davon aus, dass das gegebene Dictionary `D` die Länge n hat. Die Variable `k` bezeichnet jeweils einen gültigen Schlüssel.

(a) `D[k]`

(b) `D.del(k)`

(c)
```
for k in D:
    print(D[k])
```

Aufgabe 1.5

Welche Laufzeitkomplexität haben die folgenden Python-Anweisungen, wenn die Menge `A` aus n und die Menge `B` aus m ganzen Zahlen besteht.

(a) `A.add(24)`

(b) `27 in A`

(c) `A.union(B)`

Aufgabe 2.1

Nenne vier Anwendungsgebiete für Stacks in der Informatik.

Aufgabe 2.2

Es sei `s` eine Instanz der Klasse `Stack`. Welches ist der Unterschied zwischen den Ausdrücken `s.peek()` und `s.pop()`?

Aufgabe 2.3

Eine Python-Implementation des Datentyps `Stack` hat folgenden Konstruktor

```
class Stack:

    def __init__():
        self.items = []
```

Implementiere die folgenden Methoden

(a) `push(item)`

(b) `pop()`

(c) `isEmpty()`

für Objekte des Datentyps `Stack`.

Aufgabe 2.4

Stelle die folgenden Infix-Ausdrücke jeweils als gleichwertige, klammerlose Postfix-Terme dar. *Hinweise:* `a`, `b`, `c`, `d` stehen für Zahlen, `**` ist der Potenzoperator.

(a) `a * b - (c + d)`

(b) `c / b ** (d - a)`

Aufgabe 2.5

Gib die Ausgabe des folgenden Programmfragments als Liste an, wenn der Stack von links nach rechts wächst.

```
s = Stack()
s.push('m')
s.push('a')
s.push('z')
s.pop()
s.push('k')
s.push('c')
s.pop()

print(s)
```

Aufgabe 2.6

Beschreibe drei verschiedene Anwendungen für die Datenstruktur *Queue*.

Aufgabe 2.7

Notiere die Kurzform für die Art und Weise, wie Daten zu Queues hinzugefügt bzw. aus ihnen entfernt werden.

Aufgabe 2.8

Eine Python-Implementation des Datentyps Queue hat folgenden Konstruktor

```
class Queue:

    def __init__():
        self.items = []
```

Implementiere die folgenden Methoden für Objekte dieses Datentyps:

(a) `enqueue(item)`

(b) `size()`

Aufgabe 2.9

Gib die Ausgabe des folgenden Programmfragments als Liste an, wenn die Elemente von links in die der Queue zugrunde liegende Liste eingefügt werden.

```
from myqueue import Queue

q = Queue()
q.enqueue('m')
q.enqueue('z')
q.dequeue()
q.enqueue('a')
q.enqueue('f')
q.enqueue(q.dequeue())
q.enqueue('s')
print(q)
```

Aufgabe 2.10

Schreibe auf der Basis der Klasse `Deque` mit der Schnittstelle

Deque
data: <list>
addFront(item: <any>): None addRear(item: <any>): None removeFront(): <any> removeRear(): <any> size(): <int>

eine Funktion `palindrome(string)`, die prüft, ob es sich bei der Zeichenkette `string` um ein Palindrom handelt und entsprechend den Wahrheitswert `True` oder `False` zurückgibt.

Hinweise zu den Aufgaben 3.1–3.3 (→ 2.11–2.13)

- Der Datenspeicher ist jeweils linear aufgebaut. Aus Platzgründen erfolgt die Darstellung mehrzeilig. Die Adressierung erfolgt durch ein Byte, wobei das erste Halbbyte die Zeile und das zweite Halbbyte die Spalte angibt.
- Ein `Node` besteht aus zwei benachbarten Speicherzellen. Die erste Speicherzelle enthält einen hexadezimalen Datenwert, die zweite eine Adresse.
- `00` steht für den Nullzeiger oder ein leeres Datenfeld.

Aufgabe 3.1 (→ 2.11)

Stelle die im Speicherabbild enthaltene Daten einer einfach verkettete Liste mit der Adresse `0x2A` sequentiell dar.

Hinweis: Der Speicher kann auch noch andere oder verwaiste Daten enthalten.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		00	0B	41	00	00	00	00	00	00	00	00	00	00	00	22
1	37	00	00	00	3A	0F	00	32	41	00	00	00	00	00	00	00
2	00	00	00	00	00	00	00	00	00	0C	14	17	00	00	46	20
3	00	00	0A	22	00	00	00	32	00	00	00	00	00	24	46	00
4	00	00	00	00	00	00	00	02	04	00	00	00	00	00	00	00

Aufgabe 3.2 (→ 2.12)

Stelle im leeren Raster das Speicherabbild der einfach verketteten Liste mit der Adresse `0x16` unmittelbar nach dem Einfügen des Datenwerts `0x25` an der Adresse `0x0D` dar.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				07	19											
1							03			10	00					

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																

Aufgabe 3.3 (→ 2.13)

Stelle im leeren Raster das Speicherabbild der einfach verketteten Liste mit der Adresse `0x0C` unmittelbar nach dem Löschen des Datenwerts `0x1C` dar.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		15	00			17	18						05	41	01	
1			1C	0D					1A	12						

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																

Aufgabe 3.4 (→ 2.14)

Gib die Laufzeitkomplexität für die folgenden Operationen auf dem Datentyp einer einfach verketteten Liste mit n Elementen an.

(a) Einfügen eines Elements

(b) Suchen eines Elements

(c) Löschen eines Elements

Aufgabe 3.5 (→ 2.15)

Wie lautet der Fachausdruck für das automatische Freigeben von nicht mehr benötigtem Arbeitsspeicher?

Aufgabe 3.6 (→ 2.16)

Implementiere eine Klasse `Node` mit folgenden Attributen und Methoden:

`Node`
`data: <any>` `next: Node`

`Node(data: <any>)`
`getData(): <any>`
`getNext(): Node`
`setData(data: <any>)`
`getNext(next: Node)`

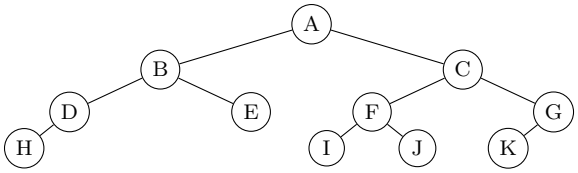
Im Konstruktor erhält das Attribut `next` standardmässig den Wert `None`.

Aufgabe 3.7 (→ 3.1)

Nenne drei unterschiedliche Informatik-Anwendungen, in denen Bäume als Datenstruktur vorkommen.

Aufgabe 3.8 (→ 3.2)

Beantworte die Fragen zum folgenden Baum:



(a) Welchen Schlüssel hat die Wurzel?

(b) Welche Kinder hat der Knoten mit dem Schlüssel C?

(c) Welche Geschwister hat der Knoten mit dem Schlüssel H?

(d) Welchen Elternknoten hat der Knoten mit dem Schlüssel E?

(e) Welches sind die Blätter des Baums?

(f) Welches sind die inneren Knoten des Baums?

(g) Welche Tiefe hat der Knoten mit dem Schlüssel D?

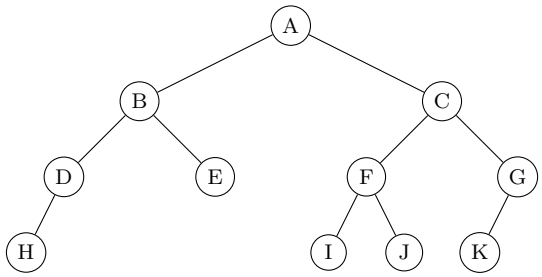
(h) Welche Höhe hat der Baum?

Aufgabe 3.9 (→ 3.3)

Definiere die Struktur eines Baumes auf rekursive Weise.

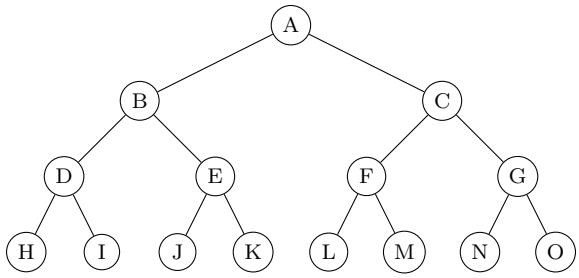
Aufgabe 3.10 (→ 3.4)

Beschreibe den Typ des Baums mit dem richtigen Fachausdruck.



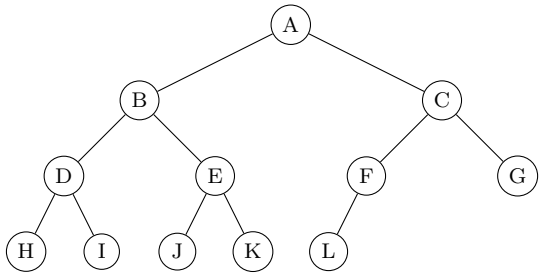
Aufgabe 3.11 (→ 3.5)

Beschreibe den Typ des Baums mit dem richtigen Fachausdruck.



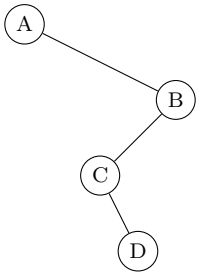
Aufgabe 3.12 (→ 3.6)

Beschreibe den Typ des Baums mit dem richtigen Fachausdruck.



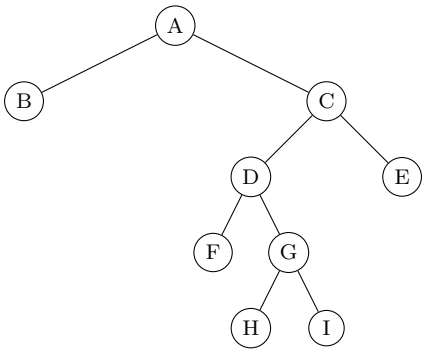
Aufgabe 3.13 (→ 3.7)

Beschreibe den Typ des Baums mit dem richtigen Fachausdruck.



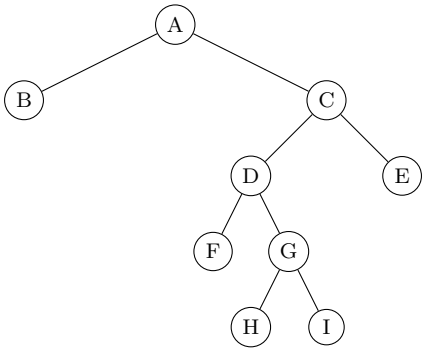
Aufgabe 3.14 (→ 3.8)

Beschreibe den Typ des Baums mit dem richtigen Fachausdruck.



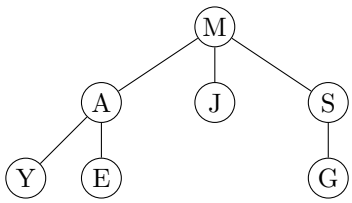
Aufgabe 3.15 (→ 3.9)

Stelle die folgende verschachtelte Liste als Baum dar.



Aufgabe 3.16 (→ 3.10)

Stelle den folgenden ternären Baum als verschachtelte Liste dar.



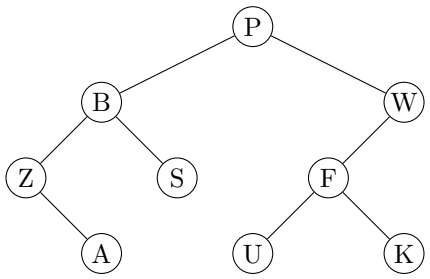
Aufgabe 3.17 (→ 3.11)

Das folgende Speicherabbild enthält einen Baum, dessen Wurzel sich an der Adresse `0x32` befindet. Jeder Knoten besteht aus drei benachbarten Speicherzellen, wobei der erste Knoten den Schlüssel, der zweite Knoten die Referenz auf einen allfälligen linken Teilbaum und der dritte Knoten die Referenz auf einen allfälligen rechten Teilbaum enthält. `0x00` bezeichnet den NULL-Zeiger. Skizziere diesen Baum mit seinen Schlüsseln.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		00	0B	41	1E	2F	10	00	00	00	00	00	00	00	00	22
1	37	00	00	00	3A	0F	00	32	41	00	00	00	00	00	00	00
2	00	00	00	00	00	00	00	00	00	0C	14	17	00	00	46	19
3	00	00	22	04	3A	00	00	32	00	00	30	00	4C	24	46	00
4	00	00	00	00	00	00	00	02	04	00	00	00	2E	00	00	00

Aufgabe 3.18 (→ 3.12)

Gib die Schlüsselreihenfolge bei den folgenden Traversierungen an.



(a) Preorder

(b) Inorder

(c) Postorder

Aufgabe 3.19 (→ 3.13)

Transformiere die Elemente der Liste

```
[5, 28, 13, 7, 9, 2, 16]
```

unter Beibehaltung der Reihenfolge in einen vollständigen binären Baum.

Aufgabe 3.20 (→ 3.15)

Welchen Index hat der Elternknoten des Kindknotens mit dem Index ...

(a) 76 (b) 89

in der Listendarstellung eines vollständigen binären Baums?

Aufgabe 3.21 (→ 3.16)

Ein Knoten K eines vollständigen Binärbaums hat in der Listendarstellung den Index 5.

(a) Wie gross muss diese Liste mindestens sein, damit K Kindknoten hat?

(b) Welche Indizes haben diese Kindknoten, falls sie exisiteren?

Aufgabe 3.22 (→ 3.17)

Wie viele Blätter hat ein vollständiger Baum mit 20 Knoten?

Aufgabe 3.23 (→ 3.18)

Gegeben ist ein Heap mit der Listendarstellung

[3, 5, 7, 16, 13, 8].

Zeige vollständig und schrittweise, wie das Element 4 in den Heap eingefügt und die Heap-Bedingung wiederhergestellt wird, indem du jeden Zustand als Baum darstellst.

Aufgabe 3.24 (→ 3.19)

Gegeben ist ein Heap mit der Listendarstellung

[2, 4, 10, 18, 33, 21]

Zeige vollständig und schrittweise, wie das kleinste Element aus der Heap-Liste entfernt und die Heap-Struktur wiederhergestellt wird, indem du jeden Zustand als Baum darstellst.

Aufgabe 3.25 (→ 3.20)

Gegeben ist die Liste [16, 12, 9, 4, 7, 3].

Zeige vollständig und schrittweise, wie diese Liste effizient in einen Min-Heap transformiert wird, indem du jeden Zustand als Baum darstellst.