Technische Universität Berlin **SoSe 2020**
Fakultät II, Institut für Mathematik
Sekretariat MA 6–2, Antje Schulz
Prof. Dr. Michael Joswig
Dr. Frank Lutz, Sarah Morell, Manuel Radons

1. Programmieraufgabe Computerorientierte Mathematik II

Abgabe: 11.5.2020 (verlängert wegen des Feiertags am 8.5.) über den Comajudge bis 18:00 Uhr

In dieser Programmieraufgabe geht es um eine erste Implementierung binärer Bäume. Schreiben Sie hierfür eine Klasse `Node` mit den Attributen

- `key` (ganze Zahl)
- `leftChild` (Instanz der Klasse `Node`)
- `rightChild` (Instanz der Klasse `Node`)

die auf die ganzzahlige Knotenschlüssel, das linke und das rechte Kind verweisen. Ein **Binärbaum** wird dann durch seinen Wurzelknoten dargestellt.

Konvention: In jedem Baum kommt jeder Schlüssel höchstens einmal vor.

Beispiele

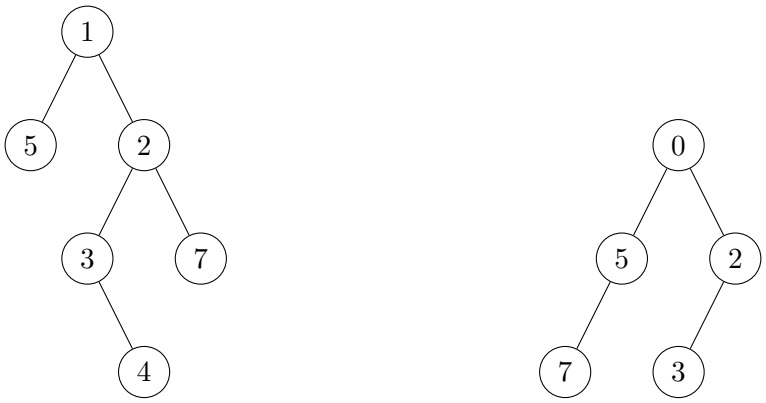

Abbildung 1: Binäre Wurzelbäume `bin1` und `bin2`

```
1 import PA01
2
3 nodeRLR = PA01.Node(4,None,None)
4 nodeRL = PA01.Node(3,None,nodeRLR)
5 nodeRR = PA01.Node(7,None,None)
6 nodeL = PA01.Node(5,None,None)
7 nodeR = PA01.Node(2,nodeRL,nodeRR)
8 bin1 = PA01.Node(1,nodeL,nodeR)
```

```
1 import PA01
2
3 nodeLL = PA01.Node(7,None,None)
4 nodeL = PA01.Node(5,nodeLL,None)
5 nodeRL = PA01.Node(3,None,None)
6 nodeR = PA01.Node(2,nodeRL,None)
7 bin2 = PA01.Node(0,nodeL,nodeR)
```

Aufgabenstellung

Implementieren Sie die folgenden Methoden:

a) Konstruktor `__init__(self, key, leftChild, rightChild)`

b) `keys(self)` gibt die Knotenschlüssel seines Baums in einer Liste zurück.

c) `height(self)` gibt die Höhe des Knotens in seinem Baum zurück.

d) `leaves(self)` gibt die Schlüssel der Blätter seines Baums in einer Liste zurück.

```
>>> bin1.height()
3
>>> nodeRL.height()
1
>>> bin1.keys()
[1, 5, 2, 3, 4, 7]
>>> bin1.leaves()
[5, 4, 7]
>>> nodeR.leaves()
[4, 7]
```

```
>>> bin2.keys()
[0, 5, 7, 2, 3]
>>> nodeLL.height()
0
>>> bin2.height()
2
>>> bin2.leaves()
[7, 3]
>>> nodeLL.leaves()
[7]
```

- Bitte verzichten Sie auf den Import graphentheoretischer Module. Programme, die auf externe Software zurückgreifen, können von uns auch noch nachträglich aberkannt werden.
- Gerne dürfen Sie vom `pickle`-Modul Gebrauch machen, um `python`-Objekte zu speichern und zu laden. Tauschen Sie Tests untereinander aus!

Tests

Anmerkung: Die folgende Anleitung gilt für Konsolenaufrufe unter Linux. Unter Windows kann das Vorgehen abweichend sein. Hierzu empfehle ich die Lektüre des Forumsthreads "Test: Error student.ot existiert nicht".

Zur Unterstützung wird Ihnen das ZIP-Archiv `Tests_Neu.zip` zur Verfügung gestellt. Dieses enthält vier Dateien: `test.py`, `test_students.sh`, `new_classtest.py`, `correct.ot`. Kopieren Sie diese in ein Verzeichnis, in dem Ihre Abgabedatei liegt. Diese muss `PA01solution.py` benannt sein. Anschließend führen Sie in einer Konsole im betreffenden Verzeichnis den Befehl

python3 test.py

aus. Hierauf werden zwei Textdateien erzeugt.

- `student.ot`,
- `comparison`.

Die erste dieser Dateien, `student.ot`, enthält die Testausgaben, die mittels Ihres Programmes von `new_classtest.py` produziert wurden. Die zweite der Dateien ist eine Vergleichsdatei zwischen der Datei `student.ot` der Ausgaben, die mittels Ihres Programmes von `classtest.py` erzeugt wurden und der Datei `correct.ot` der korrekten Ausgaben. Es werden in `comparison` nur voneinander abweichende Zeilen von `student.ot` und `correct.ot` gelistet. Die Ausgaben in `student.ot` sind also genau dann korrekt, wenn `comparison` leer ist.

Sie können diese Aufgabe bis einschließlich Montag, den 18.5.2020, bei Ihrer Tutorin oder Ihrem Tutor vorstellen.