**Aufgaben zur Veranstaltung**
**Algorithmen und Datenstrukturen, SS 2019**

H. Pflug, J. Dietel FH Aachen, Campus Jülich; IT Center, RWTH Aachen

# Präsenzaufgaben 10 27./28.05.2019

Die Lösung der Aufgaben wird am Ende der Übung von Ihnen vorgestellt.

## Aufgabe 1

Schreiben Sie eine Funktion

```
public static int[] getTwoSum(int[] arr, int sum)
```

Die Funktion soll die Indizes von zwei Zahlen aus dem Feld `arr` zurückgeben, die addiert `sum` ergeben. Finden sich zwei solche Zahlen nicht, wird `null` zurückgegeben. Der Algorithmus soll eine möglichst kleine O-Klasse besitzen.

## Aufgabe 2

Fügen Sie der Klasse `BinTree` (Präsenzblatt 6) die erforderliche Funktionalität hinzu, um sie in einer foreach-Schleife durchlaufen zu können.

- Schreiben Sie eine Klasse `BinTreeIterator`, die das Interface `Iterator<Integer>` implementiert. Die Klasse benötigt:
  - Einen Konstruktor, dem der zu durchlaufende `BinTree` übergeben wird
    `public BinTreeIterator(BinTree tree)`
  - Eine Methode, die das nächste Element des `BinTree` zurückgibt
    `public Integer next()`
  - Eine Methode, die zurückgibt, ob ein weiteres Element im `BinTree` vorhanden ist
    `public boolean hasNext()`
- Die Klasse `BinTree` selbst muss das Interface `Iterable<Integer>` implementieren. Das erfordert eine zusätzliche Methode
  `public Iterator<Integer> iterator()`
  Diese Methode gibt ein neu erzeugtes Objekt der Klasse `BinTreeIterator` zurück.

- Die Reihenfolge, mit der die Elemente des `BinTree` ausgegeben werden, ist **nicht** wichtig. Jedoch soll die Funktionalität möglichst günstig in Bezug auf Zeit und Speicherplatz sein. Angestrebt wird durchschnittlich O(1) Zeit und maximal O(h) Speicherplatz, wobei h die maximale Höhe des Baums ist.
- Testen Sie die Klasse mit dem folgenden Testprogramm

```
public static void main(String[] args) {
  BinTree tree = new BinTree();
  int[] arr = {6,9,2,5,1,12,8,4};
  for (int i: arr) {
    tree.insert(i);
  }
  for (int i: tree) {
    System.out.println(i);
  }
}
```

## Aufgabe 3

Schreiben Sie eine weitere Klasse `BinTreeSortedIterator`, die wie `BinTreeIterator` funktioniert, die Elemente des Baums aber in aufsteigender Reihenfolge (In-Order-Durchlauf im Suchbaum) zurückgibt.

## Aufgabe 4

Schreiben Sie eine Funktion

```
public static int[] getBestTwoSum(int[] arr, int sum)
```

Die Funktion soll die Indizes von zwei Zahlen aus dem Feld `arr` zurückgeben, die addiert `sum` ergeben. Finden sich zwei solche Zahlen nicht, wird im Gegensatz zu Aufgabe 1 nicht `null` zurückgegeben, sondern die Indizes derjenigen Zahlen, die addiert am nächsten an `sum` herankommen. Der Algorithmus soll eine möglichst kleine O-Klasse besitzen.