Aufgaben zur Veranstaltung
Algorithmen und Datenstrukturen, SS 2020

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

Hausaufgaben 5 27/28.04.2020

Abgabe der Lösung am 03.05.2020

Aufgabe 1

Fügen Sie in die Klasse `BinarySearchTree` (ILIAS) folgende rekursive Methoden ein:

- `public String toString()`: Gibt die Werte des Suchbaums in In-Order-Reihenfolge (bzw. aufsteigender Reihenfolge) zurück. Siehe Vorlesungsvideo. Verbessern Sie im Vergleich zum Vorlesungsvideo die Behandlung der Kommas.
- `public int getElementCount()`: Siehe Vorlesungsvideo
- `public int getSum()`: Siehe Vorlesungvideo
- `public int getHeight()`: Gibt die Höhe des Baums zurück.
- `public int getLeafCount()`: Gibt die Anzahl der Blätter des Baums zurück.
- `public boolean hasNodesWithOneChild()`: Gibt zurück, ob der Baum Knoten mit einem einzigen Kind besitzt.

Aufgabe 2

Die Funktion `getLargestKElements(int[] list, int k)` aus der Klasse `HeapUtils` (ILIAS) gibt ein Feld mit den `k` größten Elementen aus dem Feld `list` zurück. Der Code ist zwar kurz, aber trotzdem anspruchsvoll und natürlich vollkommen unkommentiert.

- Kommentieren Sie alle schwierigen Codestellen (also den gesamten Code) der Funktion `getLargestKElements`. Ihre Kommentare sollten unter anderem folgende Punkte enthalten:
  - Was bedeutet der Konstruktoraufruf von `q`? Wie müsste man ihn ändern, um die `k` kleinsten Elemente zu finden?
  - Was beinhaltet der Heap `q` nach der ersten Schleife?
  - Was beinhaltet der Heap `q` nach jedem Durchlauf der 2. Schleife?
  - Was passiert genau in der `return`-Zeile?
  - Welche O-Klasse hat die Funktion? Begründen Sie Ihre Antwort. Achten Sie auf die O-Klassen der Methoden `add` und `poll`.
- Hinweise:
  - Der Algorithmus wird auf der Seite https://www.geeksforgeeks.org/k-largestor-smallest-elements-in-an-array/ „erklärt“. Es ist Methode 6.
  - Sie können sich den Inhalt des Heaps einfach mit `q.toString()` anzeigen lassen.

Aufgabe 3

Schreiben Sie eine Klasse `Brackets` mit der Funktion

```
public static boolean isValid(String s)
```

Der übergebene String `s` enthält neben anderen (bedeutungslosen) Zeichen öffnende und schließende Klammern, wobei die Klammern rund (), eckig [] oder geschweift {} sein können.

Die Klammern sind valide, falls gilt:

- Jede sich öffnende Klammer wird von einer Klammer gleicher Art wieder geschlossen.
- Jede sich schließende Klammer wurde von einer Klammer gleicher Art geöffnet.
- Es dürfen keine Überschneidungen in der Form [( ]) stattfinden. Um valide zu bleiben, muss die runde Klammer innerhalb der eckigen Klammern wieder geschlossen werden [( )].

Benutzen Sie für diese Aufgabe einen Stack und verwenden Sie dazu die Java-Klasse `ArrayDeque`.

Tipp:

- Legen Sie öffnende Klammern auf den Stack.
- Bei schließenden Klammern holen Sie die oberste Klammer vom Stack, falls das möglich ist. Diese Klammer muss zur schließenden Klammer passen.
- Am Ende des Ausdrucks darf keine Klammer mehr auf dem Stack sein.

Test:

(([[ ]])) → true

([ )] → false

([ ]]) → false

(( ))) → false

(( ) → false

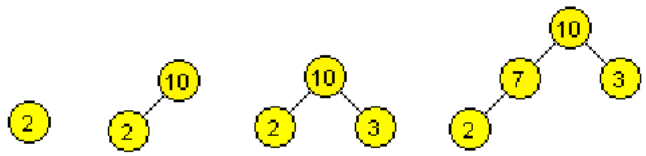
({ [ ]) } → false

Gar keine Klammern → true

Aufgabe 4

a) Fügen Sie die folgenden Werte nacheinander in einen Heap ein. Zeichnen Sie nach jedem Einfügen den Heap neu.
2,10,3,7,5,1,9,4,8

Hinweis: Die ersten Schritte sind:



b) Entfernen Sie vom folgenden Heap nacheinander alle Werte, bis der Heap aufgelöst ist. Zeichnen Sie nach jedem Entfernen den Heap neu.