Hochschule
Albstadt-Sigmaringen
Albstadt-Sigmaringen University

Technische Informatik
Prof. Dr. Tobias Häberlein

Algorithmik
SS 12

Aufgabenblatt 4

(Bearbeitungszeit: 2-3 Praktikumstermine)

Heaps

Aufgabe 3.0. **10 Punkte**

Gegeben sei die folgende Liste von Werten:

[61, 60, 48, 67, 80, 16, 52, 55, 19, 81, 9, 18, 8, 13, 97, 59, 47, 11, 73, 93]

(a) Simulieren Sie „von Hand“ die Funktion *buildHeap* und erstellen Sie so einen binären (Min-)Heap, die die in obiger Liste enthaltenen Werte enthält.

(b) Zeichnen Sie einen Binomial-Heap (unter vielen möglichen), die die in obiger Liste enthaltenen Werte enthält.

(c) Zeichnen Sie einen binären Suchbaum (unter vielen möglichen), die die in obiger Liste enthaltenen Werte enthält.

(d) Zeichnen Sie einen AVL-Baum (unter vielen möglichen), die die in obiger Liste enthaltenen Werte enthält.

Aufgabe 3.1. **20 Punkte**

Implementieren Sie die folgenden Funktionen für binäre Heaps:

(a) Die Funktion *insert*(*heap*,*x*), die ein Element x in den Heap *heap* einfügt.

(b) Die Funktion *extractMin*(*heap*), die die Minimumsextraktion implementiert.

(c) Die Funktion *delete*(*heap*, *i*), die das Element mit Index i im Heap *heap* löscht.

Aufgabe 3.2. **20 Punkte**

Implementieren Sie die folgenden Funktionen für Binomial-Heaps

(a) Die Funktion *mergeBH*(*heap1*,*heap2*), die zwei Heaps *heap1* und *heap2* verschmelzen.

(b) Die Funktion *insertBH*(*heap*,*x*), die ein Element x in den Heap *heap* einfügt.

(c) Die Funktion *extractMinBH*(*heap*), die die Minimumsextraktion implementiert.

Aufgabe 3.3. **10 (+10) Punkte**

Implementieren Sie die in der Vorlesung vorgestellte *Trie*-Klasse und alle wichtigen Methoden, nämlich Einfügen, Suchen und Löschen (+10 Punkte).

Aufgabe 3.4. **15 Punkte**

(a) Implementieren Sie eine Methode *keys*(), die eine Liste aller in einem Trie befindlichen Schlüsselwerte zurückliefert.

(b) Implementieren Sie eine Methode *vals*(), die eine Liste aller in einem Trie befindlichen Schlüsselwerte zurückliefert.

Aufgabe 3.5. **10+15 Punkte**

(a) Implementieren Sie die in der Vorlesung vorgestellte Klasse *Graph* inklusive der vorgestellten Methoden.

(b) Implementieren Sie die in der Vorlesung vorgestellte topologische Sortierung.in einer Funktion *topSort*(*graph*).

(c) Implementieren Sie eine Funktion *components*(*graph*), die die Komponenten eines Graphen zurückliefert. Eine Komponente soll hierbei als Liste der in der Komponente enthaltenen Knoten repräsentiert sein. Beispielanwendung auf einen Graphen mit 3 Komponenten könnte etwa wie folgt aussehen:

```
>>> components(graph)
  [[1,5],[2,3,8,9,10],[4,6,7,11]]
```