

Einzelprüfung „Theoretische Informatik / Algorithmen (vertieft)“

Einzelprüfungsnummer 66115 / 2015 / Herbst

Thema 2 / Aufgabe 2

(Haldensortierung)

Stichwörter: Heapsort

Gegeben sei folgende Klasse:

```
class W {  
    int t;  
    String f;  
    // ...  
}
```

Dazu gibt es verschiedene Comparatoren, zum Beispiel:

```
// ascending order for field W.t  
class ComparatorAscByFieldT implements Comparator<W> {  
    // Returns a negative integer, zero, or a positive integer as the  
    // first argument is less than, equal to, or greater than the second.  
    @Override  
    public int compare(W o1, W o2) { // ...  
    }
```

Außerdem steht Ihnen die vorgegebene Methode swap zur Verfügung:

```
void swap(W[] w, int a, int b) { // ...  
}
```

- (a) Phase 1: Die Haldensortierung beginnt mit der Herstellung der Max-Heap-Eigenschaft von rechts nach links. Diese ist für alle Feldelemente im dunklen Bereich bereits erfüllt. Geben Sie die Positionen (IDs) derjenigen Elemente des Feldes an, die das Verfahren im „Versickerschritt“ für das nächste Element mit Hilfe des `ComparatorAscByFieldT` miteinander vergleicht:

IDsangeben> 0 1 2 3 4 5 6 <iDs angeben

Nach dem Vergleichen werden gegebenenfalls Werte mit swap vertauscht. Geben Sie das Resultat (in obiger Array-Darstellung) nach diesem Schritt an.

- (b) Phase 2: Das folgende Feld enthält den bereits vollständig aufgebauten MaxHeap:

Qo 1 2 3 4 5 6

71/6)/5,,3 7) 14,04 2

Die Haldensortierung verschiebt das maximale Element in den sortierten (dunklen) Bereich:

Q 1 2 3 4

2|6|/5/3/1/o

Geben Sie das Ergebnis des nachfolgenden „Versickerns“ (erneut in derselben Array-Darstellung) an, bei dem die Heap-Eigenschaft wiederhergestellt wird.

- (c) Ergänzen Sie die rekursive Methode `reheap`, die die Max-Heap-Eigenschaft im Feld `w` zwischen den Indizes `i` und `k` (jeweils einschließlich) in $O(\log(k-i))$ gemäß `Comparator<W> c` wiederherstellt, indem sie das Element `w[i]` „versickert“. `k` bezeichnet das Ende des unsortierten Bereichs.

```
// restores the max-heap property in w[i to k] using c
void reheap(W[] w, Comparator<W> c, int i, int k) {
    int leftId = 2 * i + 1;
    int rightId = leftId + 1;
    int kidId;
    // ToDo: Code hier ergaenzen
}
```

- (d) Implementieren Sie nun die eigentliche Haldensortierung. Sie dürfen hier die Methode `reheap` verwenden.

```
// sorts w in-situ according to the order imposed by c
void heapSort(W[] w, Comparator<W> c) {
    int n = w.length;

    // Phase 1: Max-Heap-Eigenschaft herstellen
    // (siehe Teilaufgabe a)
    // ToDo: Code hier ergaenzen

    // Phase 2: jeweils Maximum entnehmen und sortierte Liste am Ende aufbauen
    // (siehe Teilaufgabe b)
    // ToDo: Code hier ergaenzen
}
```

Additum: Die komplette Java-Klasse

```
import java.util.Comparator;

public class Heapsort {

    void swap(W[] w, int a, int b) { // ...

    }

    // restores the max-heap property in w[i to k] using c
    void reheap(W[] w, Comparator<W> c, int i, int k) {
        int leftId = 2 * i + 1;
        int rightId = leftId + 1;
        int kidId;
        // ToDo: Code hier ergaenzen
    }

    // sorts w in-situ according to the order imposed by c
    void heapSort(W[] w, Comparator<W> c) {
        int n = w.length;
    }
}
```

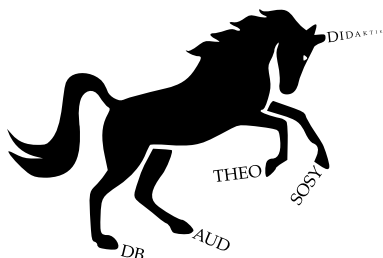
```
// Phase 1: Max-Heap-Eigenschaft herstellen
// (siehe Teilaufgabe a)
// ToDo: Code hier ergaenzen

// Phase 2: jeweils Maximum entnehmen und sortierte Liste am Ende aufbauen
// (siehe Teilaufgabe b)
// ToDo: Code hier ergaenzen
}
}

// ascending order for field W.t
class ComparatorAscByFieldT implements Comparator<W> {
    // Returns a negative integer, zero, or a positive integer as the
    // first argument is less than, equal to, or greater than the second.
    @Override
    public int compare(W o1, W o2) { // ...
        return 0;
    }
}

class W {
    int t;
    String f;
    // ...
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66115/jahr_2015/herbst/Heapsort.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2015/herbst/Heapsort.java)



Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.



Diese Materialsammlung unterliegt den Bestimmungen der Creative Commons Namensnennung-Nicht kommerziell-Share Alike 4.0 International-Lizenz.

Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net. Der \LaTeX -Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden: <https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Examen/66115/2015/09/Thema-2/Aufgabe-2.tex>