

Übungsblatt 8

Abgabe bis Freitag, 20.06.2016, 23:59 Uhr

Hinweis:

Aufgaben immer per E-Mail (eine E-Mail pro Blatt und Gruppe) an den zuständigen Tutor schicken (bei Programmieraufgaben Java Quellcode und evtl. benötigte Datendateien).

Aufgabe 8.1

Betrachten Sie den Auszug der Klasse `MyTuple`, die ein Tupel aus drei Objekten darstellt.

```
public class MyTuple<A,B,C> {  
    public void setA( ... ) { ... }  
    public void setB( ... ) { ... }  
    public void setC( ... ) { ... }  
  
    public ... getA(){ ... }  
    public ... getB(){ ... }  
    public ... getC(){ ... }  
  
    public String toString() {  
        ...  
    }  
  
    A a;  
    B b;  
    C c;  
}
```

1. Vervollständigen Sie die `set` - und `get`-Methoden der Klasse `MyTuple`.
2. Vervollständigen Sie die `toString` Methoden der Klasse `MyTuple`, so dass sie einen String der Form “(... , ... , ...)” ausgibt.
Hinweis: Verwenden Sie die `toString`-Methoden der drei Member-Variablen.
3. Implementieren Sie eine Klasse `MyTupleList<A, B, C>`, die eine Liste aus `MyTuple<A, B, C>`-Elementen enthält. Die Liste soll hierbei mit Hilfe der Klasse `ArrayList` repräsentiert werden.
4. Implementieren Sie für die Klasse `MyTupleList<A, B, C>` eine Methode `addTuple(MyTuple<A, B, C> tuple)`, die ein Tupel in die Liste einfügt.

$$n^2 \leq c \cdot n \Rightarrow n \leq c$$

$$n^2 + 27n \leq n^2 + 27n^2 = 28n^2$$

$$n + 27 \leq 28n$$

$$27 \leq 27n \Rightarrow 1 \leq n$$

$$\left. \begin{array}{l} c = 28 \\ n_0 = 1 \end{array} \right\}$$

Aufgabe 8.2

- Geben Sie die Definition der O-Notation für die asymptotische Komplexität an.

- Beweisen oder widerlegen Sie folgende Aussagen:

(a) $n^2 + 27n \in O(n^2)$ (b) $2^{n+1} \in O(2^n)$

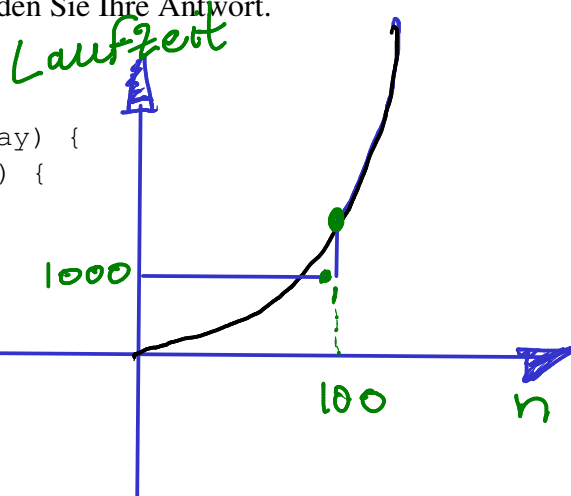
(c) $n^2 \in O(n)$

(d) $f(n) \in O(n^2)$ mit $f(n) = \begin{cases} 1000 & \text{für } n < 100 \\ n^2 & \text{für } n \geq 100 \end{cases}$

- Bestimmen Sie die Komplexität der Methoden alg1 und alg2 in Abhängigkeit der Länge n der übergebenen ArrayList Objekte. Geben Sie dabei die **best-** und **worst case** Laufzeit in der O-Notation an. Gehen Sie davon aus, dass die Methode `System.out.println(i)` in $O(1)$ liegt. Begründen Sie Ihre Antwort.

Hinweis: $\sum_{i=0}^n i^2 = \frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6}$

```
public void alg1(ArrayList<Integer> array) {
    for (int i = 0; i < array.size(); i++) {
        if (array.get(i) > 0) {
            for (int j = 0; j < i; ++j) {
                System.out.println(i);
            }
        } else {
            System.out.println(i);
        }
    }
}
```



```
public void alg2(ArrayList<Integer> array) {
    for (int i = 1; i <= array.size(); ++i) {
        for (int j = 1; j <= i * i; ++j) {
            System.out.println(i);
        }
    }
}
```

```
for(int j = i + 1; j < sortedList.size(); ++j) {
    contains_duplicates = true;
}
```

sortedList.size() - 1

$$1 + 1 + 1 + 1 \dots + 1 = \text{sortedList.size()} \cdot 1 = \sum_{j=1+i} 1$$

Aufgabe 8.3

Folgende Methode erhält eine **sortierte** ArrayList und soll überprüfen, ob die Liste zwei gleiche Zahlen enthält.

- Bestimmen Sie die Komplexität der Methode `containsDuplicates` in Abhängigkeit der Länge n der übergebenen ArrayList. Geben Sie dabei die **best-** und **worst case** Laufzeit in der O -Notation an. Gehen Sie davon aus, dass die Methode `equals` in $O(1)$ liegt.
- Schreiben Sie ein Java-Programm, das die gleiche Aufgabe im best-case in $O(1)$ und im worst-case in $O(n)$ löst.

```
public boolean containsDuplicates(ArrayList<Integer> sortedList) {
    boolean contains_duplicates = false;
    for(int i = 1; i < sortedList.size(); ++i) {
        for(int j = i + 1; j < sortedList.size(); ++j) {
            if(sortedList.get(i).equals(sortedList.get(j))) {
                contains_duplicates = true;
            }
        }
    }
    return contains_duplicates;
}
```