

Präsenzaufgaben 1

22.03.2022

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

Aufgabe 1

Auf der Seite <https://java2blog.com/find-pair-whose-sum-is-closest-to-zero-in-array/> finden Sie ein Programmierproblem und die zugehörigen Java-Lösungen.

Das Problem heißt: Gegeben ist ein Feld von Integer-Zahlen. Finden Sie aus dem Feld das Zahlenpaar, dessen Summe am nächsten an 0 liegt. Ihre Aufgabe ist es, die gegebenen Lösungen auf ihre Laufzeitkomplexität bezüglich der Feldlänge zu untersuchen.

- Bestimmen Sie die Laufzeitkomplexität (O-Klasse) der Lösung 1.
- Die O-Klasse der Lösung 2 ist unter dem Code angegeben: $O(n \cdot \log n)$. Verantwortlich dafür ist die allererste Zeile, die das Feld sortiert:

```
Arrays.sort(arr);
```

Wenn Sie sich diese Zeile wegdenken, welche Laufzeitkomplexität hätte dann der Rest des Codes? Würde sich die O-Klasse ändern, wenn man voraussetzen würde, dass `arr` bereits sortiert ist?

- Ein weiteres Problem finden Sie auf der Seite <https://www.programcreek.com/2012/12/leetcode-reverse-integer/>. Hier geht es darum, die Ziffern einer Integer-Zahl x herumzudrehen. Untersuchen Sie die gegebenen Lösungen auf die Laufzeitkomplexität bezüglich der Zahl x .

Welche Laufzeitkomplexität haben die Lösungen 2 und 3? Haben beide dieselbe O-Klasse? Tipp: Sehen Sie sich zunächst Lösung 3 an und betrachten Sie genau, was mit der Laufvariablen x passiert.

Aufgabe 2

Einen typischen Code für die binäre Suche in einem Integer-Feld finden Sie auf der Seite <https://www.geeksforgeeks.org/binary-search/>. Der Code ist in zahlreichen Programmiersprachen angegeben. Nehmen Sie für die Aufgabe den Java-Code.

Gegeben sei der folgende Testaufruf:

```
BinarySearch ob = new BinarySearch();  
int arr[] = { 1, 2, 3, 5, 6, 8, 10, 20, 40 };  
int n = arr.length;  
int x = 4;  
int result = ob.binarySearch(arr, 0, n - 1, x);
```

- Notieren Sie (am besten ohne den Code auszuführen) für jeden rekursiven Funktionsaufruf die Parameter l und r . Es beginnt mit:
 - Aufruf: $l=0, r=8$
- Ändern Sie den Code so ab, dass Sie den „Interpolation search“ aus https://en.wikipedia.org/wiki/Binary_search_algorithm#Interpolation_search implementieren.

Aufgabe 3

Sehen Sie sich die Implementierung der add-Methode eines dynamischen Felds („ArrayList“) im YouTube-Video <https://www.youtube.com/watch?v=-PktkiWCqBs> an.

- a) Der Autor sagt selbst zu Beginn, dass seine Methode nicht optimal ist. Was verändert sich in der O-Klasse gegenüber der Implementierung aus der Vorlesung? Warum?
- b) Schreiben Sie eine verbesserte Klasse Storage. Die Vorlage finden Sie im ILIAS-Kurs.
 - Ändern Sie die Attribute und die add-Methode so ab, dass die Laufzeitkomplexität wieder stimmt.
 - Schreiben Sie die Klasse so um, dass sie Generics benutzt und dadurch auch für andere Datentypen als Integer verwendbar ist.
 - Implementieren Sie außer der Methode add(E e) die Methoden get(int index), remove(int index), indexOf(Object o) und toString() entsprechend der API für die Klasse ArrayList. Die Methode remove lässt die Feldgröße unverändert. Andere Methoden können Sie weglassen.

Hinweis: Die Zeilen

```
public class Storage<T> {  
    private T[] speicher = new T[10];  
  
}
```

ergeben einen Fehler. Das ist ein typisches Java-Problem. Man muss statt dessen

```
public class Storage<T> {  
    public T[] speicher = (T[]) new Object[10];  
}
```

schreiben und die Warnung ignorieren.