Marburg University

Department of Computer Science
Prof. Dr. Sebastian Wild

Date: 2026-05-29
Version: 2026-05-30 16:58

# Übungsblatt 7 zur Vorlesung Algorithmen und Datenstrukturen (Sommer 2026)

***Abgabe:*** *Bis 2026-06-06 18:00, on ILIAS.*

## 1. Aufgabe (3sum)

40 Punkte

Lösen Sie `marburg.kilonova.ro/problems/6` (3sum).

**ILIAS-Abgabe:** Beschreiben Sie in Ihrer ILIAS-Abgabe Ihren verwendeten Algorithmus entsprechend des Templates: *Idee*, *Pseudocode*, *Korrektheit*, *Analyse*. Der Pseudocode-Teil kann hier eine informelle Zusammenfassung Ihres eingereichten Codes auf Kilonova sein.

**Tipp:** Sie werden einen effizienteren Algorithmus als den Brute-Force-Ansatz aus der Vorlesung entwickeln müssen.

Rufen Sie sich dazu auch die Bibliotheksmethoden `java.util.Arrays.sort(int[])` bzw. `edu.princeton.cs.algs4.Quick.sort(Comparable[])` in Erinnerung.

## 2. Aufgabe

30 Punkte

Betrachten Sie Insertion Sort mit binärer Suche, d.h. wir iterieren durch die Elemente der Eingabe einzeln und halten die bereits bearbeiteten Elemente in sortierter Reihenfolge in einem Präfix des Arrays vor. Wenn wir das nächste Element betrachten, verwenden wir *binäre Suche*, um die Stelle zu finden, an der es in der sortierten Reihenfolge erscheinen sollte, und platzieren es dort.

Analysieren Sie die Anzahl der Vergleiche, die von diesem Algorithmus verwendet werden. Wie verhält sie sich im Vergleich zur Gesamtlaufzeit des Algorithmus?

Für volle Punkte muss Ihre Analyse die korrekte $\sim$-Asymptotik angeben.

```
1  static void binaryInsertionsort(Comparable[] a) {
2      int n = a.length;
3      for (int i = 1; i < n; i++) {
4          // binary search to determine index j at which to insert a[i]
5          Comparable v = a[i];
6          int lo = 0, hi = i;
7          while (lo < hi) {
8              int mid = lo + (hi - lo) / 2;
9              if (less(v, a[mid])) hi = mid;
10             else                 lo = mid + 1;
11         }
12         // insert a[i] at index j and shift a[j], ..., a[i−1] to right
13         for (int j = i; j > lo; --j)
14             a[j] = a[j-1];
15         a[lo] = v;
16     }
17 }
```

## 3. Aufgabe

30 Punkte

Schreiben Sie eine Implementierung von Quicksort und Mergesort und vergleichen Sie deren Laufzeiten beim Sortieren von $n$ zufälligen ganzen Zahlen für verschiedene Werte von $n$ von 1 bis $10^6$.

Dokumentieren Sie Ihre Beobachtungen in angemessener Form (z.B. als Plots) und vergleichen Sie diese mit den Angaben aus den Vorlesungsfolien.