Lehrstuhl für Informatik I
Algorithmen und Komplexität
Universität Würzburg

Würzburg, den 15. Oktober 2026

Prof. Dr. Alexander Wolff
Samuel Wolf, M. Sc.

1. Übungsblatt zur Vorlesung Algorithmen und Datenstrukturen (Winter 2026/27)

PABS **Aufgabe 1 – MaxFinder**

In dieser Aufgabe werden Sie in Java einen Divide&Conquer-Algorithmus implementieren, der in einem Eingabefeld das größte Element nach vorne verschiebt.

Erstellen Sie die Klasse `public class MaxFinder` im Paket `algorithms`.

a) Implementieren Sie zunächst die folgende Methode:

- `private static void conquerMax(int[] a, int ℓ, int m, int r)`.

Werfen Sie eine `IllegalArgumentException`, wenn $0 \leq \ell \leq m < r < a.\texttt{length}$ nicht gilt. Nach Aufruf der Methode soll a die gleichen Zahlen wie vor dem Aufruf enthalten, wobei in $a[\ell]$ das Maximum aus den zwei Elementen $\{a[\ell], a[m+1]\}$ stehen soll. Die Laufzeit soll konstant sein. **2 Punkte**

b) Implementieren Sie weiter folgende Methode:

- `public static void moveMaxToFront(int[] a, int ℓ, int r)`

Werfen Sie bei ungültiger Eingabe eine `IllegalArgumentException`. Nach Aufruf von `moveMaxToFront(a,ℓ,r)` soll a die gleichen Zahlen wie vor dem Aufruf enthalten, wobei die größte Zahl von $a[\ell..r]$ an der Stelle $a[\ell]$ stehen soll.

Ihre Implementierung soll nach dem Divide&Conquer-Prinzip aufgebaut sein: Wenn $r > \ell$, teilen Sie das Problem in zwei Teilprobleme $a[\ell..m]$ und $a[m+1..r]$ für ein geeignetes m und lösen Sie diese rekursiv. Anschließend lösen Sie das Problem für $a[\ell..r]$ mithilfe der Lösungen für die Teilprobleme. Benutzen Sie hierfür Ihre Methode `conquerMax`. **3 Punkte**

Aufgabe 2 – EasySort

Sei A ein Array der Länge n und sei folgender Algorithmus in Pseudocode gegeben.

```
EasySort(int[ ] A)
for i = 1 to A.length − 1 do
    min = A[i]
    indexMin = i
    for j = i + 1 to A.length do
        if A[j] < min then
            min = A[j]
            indexMin = j
    A[indexMin] = A[i]
    A[i] = min
```

a) Beweisen Sie, dass EasySort das Eingabefeld A aufsteigend sortiert. Verwenden Sie dazu folgende Schleifeninvariante: **4 Punkte**

Bei der i-ten Ausführung des äußeren for-Schleifenkopfes gilt, dass

(i) *die $(i-1)$ kleinsten Elemente von A aufsteigend geordnet in den ersten $(i-1)$ Zellen des Feldes stehen und*

(ii) *in den übrigen Zellen die $(n-i+1)$ größten Zahlen von A stehen.*

b) Wie viele Vergleiche benötigt EasySort in Abhängigkeit von n? **2 Punkte**

c) Welchen Sortieralgorithmus würden Sie vorziehen, InsertionSort oder EasySort? Begründen Sie Ihre Antwort. **1 Punkt**

Aufgabe 3 – Vereinigung

Geben Sie in gut kommentiertem Pseudocode einen Algorithmus an, der als Eingabe zwei aufsteigend sortierte Felder A und B erhält. Die Ausgabe soll ein Feld C sein, das jede Zahl aus A und B genau einmal enthält. Da in A und B Zahlen mehrfach vorkommen können, ist die Länge von C anfangs unbekannt. Zur Hilfe dürfen Sie noch genau ein weiteres Feld H der Länge $n = A.length + B.length$ verwenden. Jedes Feld darf höchstens zweimal durchlaufen werden. **4 Punkte**

Aufgabe 4 – Sowjetische Münzen

Die Währung der Sowjetunion war der sowjetische Rubel. Die kleinere Einheit dazu war die Kopeke, wobei ein Rubel 100 Kopeken entsprach. Im Umlauf waren Kopekenmünzen im Wert von 1, 2, 3, 5, 10, 15, 20 und 50 Kopeken. Eine interessante Eigenschaft war dabei, dass jeder ganzzahlige Betrag ab einschließlich 8 Kopeken nur aus Münzen im Wert von 3 und 5 Kopeken bezahlt werden konnte. (So sind z.B. 16 Kopeken = 5 + 5 + 3 + 3 Kopeken.) Beweisen Sie dies mittels vollständiger Induktion. Zeigen Sie also, dass für alle natürlichen $n \geq 8$ gilt: $n = k \cdot 3 + l \cdot 5$, wobei $k, l \in \mathbb{N}_0$.
Tipp: Führen Sie den Induktionsanfang nicht nur für $n = 8$ durch. **4 Punkte**

Bitte geben Sie Ihre Lösungen bis **Donnerstag, 22. Oktober 2026, 14:00 Uhr** einmal pro Gruppe über Wuecampus als pdf-Datei ab. Vermerken Sie dabei stets die Namen und Übungsgruppen aller BearbeiterInnen auf der Abgabe.

Grundsätzlich sind stets alle Ihrer Aussagen zu begründen und Ihr Pseudocode ist stets zu kommentieren.

Die Lösungen zu den mit PABS gekennzeichneten Aufgaben, geben Sie bitte nur über das PABS-System ab. Vermerken Sie auf Ihrem Übungsblatt, in welchem Repository (*sXXXXXX*-Nummer) die Abgabe zu finden ist. Geben Sie Ihre Namen hier als Kommentare in den Quelltextdateien an.