# 2. Vorlesungsprüfung

## 703010 VO Algorithmen und Daten Strukturen 2018

28. September 2018

Vorname: _____ Nachname: _____

Matr. Nr.: _____ Note: _____

Diese Klausur besteht aus **??** Aufgaben und Sie können maximal 65 Punkte erreichen. Es sind keine elektronischen Geräte oder andere Hilfsmittel zulässig. Sie haben 90 Minuten Zeit und müssen *alle* Zettel nach Abschluss der Klausur abgeben. Wenn nötig, benennen Sie Ihre Annahmen. Geben Sie präzise und knappe Antworten.

This exam consists of **??** questions with a total of 65 points. The use of electronic devices is not allowed. The exam is closed book and closed notes. The total duration of this exam is 90 minutes. Participants have to return this copy and *all* additional sheets at the end of the exam. Specify your own assumptions if needed, and provide precise and concise answers.

# 1 Bäume (5 Punkte)

Seien $T$ und $U$ zwei (2,4)-Bäume, die $n$ bzw. $m$ Elemente enthalten, wobei die Schlüssel aller Einträge in $T$ kleiner sind als die Schlüssel aller Einträge in $U$. Beschreiben Sie eine Methode mit Laufzeit $O(\log n + \log m)$, die die Bäume $T$ und $U$ zu einem einzigen (2,4)-Baum vereint, der alle Einträge aus $T$ und $U$ enthält.

Let $T$ and $U$ be (2,4) trees storing $n$ and $m$ entries, respectively, such that all the entries in $T$ have keys less than the keys of all the entries in $U$. Describe an $O(\log n + \log m)$-time method for joining $T$ and $U$ into a single tree that stores all the entries from $T$ and $U$.

# 2 Sorted Map (20 Punkte)

1. Zählen Sie mindestens vier Methoden auf, die den abstrakten Datentyp der *Abbildung* charakterisieren, und beschreiben Sie sie kurz. **[4 Punkte]**

2. Zählen Sie mindestens drei weitere Methoden auf, die den abstrakten Datentyp der *Sortierten Abbildung* charakterisieren, und beschreiben Sie sie kurz. *Zählen Sie hier keine Methoden auf, die bereits in der* Abbildung *enthalten sind.* **[3 Punkte]**

3. Schreiben Sie Pseudo-Code, der eine Lösung des folgenden Problems berechnet:

   Seien $A$ and $B$ ungeordnete, iterierbare Kollektionen positiver ganzer Zahlen, und sei $c$ eine ganze Zahl. Finde ein geordnetes Paar $a \in A, b \in B$ sodass $a - b \leq c$ und $a + b \geq c$.

   *Für volle Punktzahl muss Ihr Algorithmus Laufzeit-effizienter sein als eine* brute-force-*Lösung.* **[10 Punkte]**

4. Charakterisieren Sie die asymptotische Laufzeit-Komplexität Ihres Algorithmus in Groß-O-Notation. **[3 Punkte]**

1. List and describe at least four methods that *characterize* the *Map* ADT.

2. List and describe at least three additional methods that *characterize* the *Sorted Map* ADT. *Do not list methods here that are already part of the Map ADT.*

3. Write pseudo-code to compute a solution for the following problem:

   Let $A$ and $B$ be unsorted iterable collections of positive integer values and $c$ a single integer value. Find a pair $a \in A, b \in B$ such that $a - b \leq c$ and $a + b \geq c$.

   *For full points you have to state a better solution than a brute force approach.*

4. Characterize the asymptotic running-time complexity of your algorithm in big-Oh notation.

## 3 Huffman-Coding (15 Punkte)

1. Gegeben sei die Zeichenkette

   ```
   an algorithm must be seen to be believed
   ```

   Erstellen Sie einen Huffman Coding-Baum (unter Berücksichtugung von Leerzeichen), welcher diese Zeichenkette mit minimaler Anzahl an Bits kodiert. [**10 Punkte**]

2. Was ist die Ersparnis an Bits im Vergleich zu einem 7-bit ASCII Code? [**5 Punkte**]

1. Given the string

   ```
   an algorithm must be seen to be believed
   ```

   create a Huffman coding tree (by considering white-space characters) that encodes this string in a minimal number of bits.

2. What is the actual number of bits saved compared to a 7-bit ASCII code?

## 4 Palindrom (10 Punkte)

1. Beschreiben Sie einen effizienten Algorithmus zur einmaligen Suche von Palindromen in einer Zeichenkette (nur Kleinbuchstaben, keine Leerzeichen, keine Interpunktion). Betrachten Sie hierzu folgendes Beispiel:
   - Eingabe: `racecarenterelephantmalayalam`
   - Ausgabe: `aceca, layal, cec, aya, racecar, ele, ere, alayala, malayalam, ala`

   [**8 Punkte**]
2. Charakterisieren Sie die asymptotische Laufzeit-Komplexität Ihres Algorithmus in Groß-O-Notation. [**2 Punkte**]

1. Describe an efficient algorithm for identifying unique palindromes in a string of text (only lowercase letters, no white space, no punctuation). At this, consider the following example:
   - Input: `racecarenterelephantmalayalam`
   - Output: `aceca, layal, cec, aya, racecar, ele, ere, alayala, malayalam, ala`
2. Characterize the asymptotic running-time complexity of your algorithm in big-Oh notation.

## 5 Rekursion (15 Punkte)

1. Zählen Sie vier Formen von Rekursion auf und beschreiben Sie sie. [**4 Punkte**]

   List and describe four forms of recursion.
2. Gegeben sind die folgenden drei Code-Snippets. Versuchen Sie deren Funktionsweise zu verstehen, und geben Sie den Output jedes dieser drei Aufrufe an, indem Sie die zugehörige römische Zahl neben dem Aufruf eintragen: [**6 Punkte**]

   Consider the three code snippets below. Try to understand how they work, and indicate the output of each of these three calls by entering the corresponding Roman numeral in the space next to the call:

   rec(0) _____
   rec2(6) _____
   rec3(2018) _____

   i 1 1 1 1 1 1 1 0 1 0 0
   ii 1 1 1 1 1 1 0 0 0 1 0
   iii 1 1 1 1 1 1 1 1 0 1 0
   iv 1 1 1 1 1 1 1 1 0 1 0
   v 1 1 1 1 1 0 0 1 1 0 0
   vi 1 3 5 7 9 0 2 4 6 8 10
   vii 0 1 2 3 4 5 6 7 8 9 10
   viii 1 3 5 7 9 10 8 6 4 2 0

   ix 0 2 4 6 8 10 9 7 5 3 1
   x 0 10 8 6 4 2 1 3 5 7 9
   xi 2 2 3 5 2 3 2 2 3 5 8 13
   xii 2 3 2 2 3 5 8 2 2 3 5 2 3 2 2 3 5 8 13 21
   xiii 2 2 3 5
   xiv 2 3 2 2 3 5 8 2 3 2 2 3 5 8
   xv 2 3 2 2 3 2 2 3 5 8 13
3. Zeichnen Sie den resultierenden Aufrufbaum von *rec3(42)*. [**5 Punkte**]

Listing 1: Function 1

```
int rec(int a) {
        if (a % 2 || a > 8) {
                printf("%d␣", a);
                return a;
        }
        int sum = rec(a + 1) + rec(a + 2);
        printf("%d␣", a);
        return sum;
}
```

Listing 2: Function 2

```
int rec2(int a) {
        if (a <= 1)
                return 1;
        int sum = rec2(a - 2) + rec2(a - 1);
        printf("%d␣", sum);
        return sum;
}
```

Listing 3: Function 3

```
void rec3(int a) {
        if (a) {
                rec3(a/2);
                printf("%d␣", a%2);
        }
}
```