universität wien

# Algorithmen und Datenstrukturen 1

## Theoretische Prüfung

## 2. Nachtermin

## 26.09.2022

| | |
|---|---|
| Name: | |
| Matrikelnummer: | |

Die Angaben sind beidseitig bedruckt!

| 1 | 17 | 29 | 10 | 31 | 19 | 37 | 22 |
|---|---|---|---|---|---|---|---|
| + | + | + | + | + | + | + | + |
| | | | | | | | |
| $z_1$ | $z_2$ | $z_3$ | $z_4$ | $z_5$ | $z_6$ | $z_7$ | $z_8$ |

## Aufgabe 1 [2]

Fügen Sie in obiger Tabelle in den leeren Kästchen, vor denen das Pluszeichen steht, die Ziffern Ihrer Matrikelnummer ein. Führen Sie die Additionen durch und ermitteln Sie die Zahlen $z_1$bis $z_8$.

## Aufgabe 2 [18]

Gegeben seien folgende Funktionen

```
void done(int n) {
  for (int i=3; i<n; ++i)
    for (int j=0; j<i; ++j)
      for (int k=0; k<n*n; ++k);
}

void todo(int n) {
  for (int i=1; i<n; i=i*5)
    done(n);
}

void doing(int n, int digit) {
  if (n==0) return;
  done(n);
  for (int i=0; i<digit; ++i) {
    doing(n/2, digit);
    done(n);
    doing(n/2, digit)

  }
}
```

Finden Sie Laufzeitabschätzungen in Theta-Notation (abhängig von n). Für `digit` setzen Sie ($z_7$%5)+2 ein.

a. [4] die Funktion done,
b. [4] die Funktion todo,
c. [10] und die Funktion doing

## Aufgabe 3 [20]

Die Werte $z_1$ bis $z_8$. (aus Aufgabe 1) seien in dieser Reihenfolge von links nach rechts in einem Array gespeichert. Sortieren Sie die Werte aufsteigend mit

a) [10] Quicksort
b) [6] Counting Sort – jede Zahl **jeweils Modulo 10** (zB $z_1$%10, $z_2$%10, usf.)
c) [4] Mergesort

Geben Sie alle notwendigen Schritte so genau an, dass die Arbeitsweise des Algorithmus klar ersichtlich wird.

## Aufgabe 4 [20]

a) [8] Fügen Sie die Werte $z_1$ bis $z_8$ aus Aufgabe 1 (in dieser Reihenfolge) in einen zu Beginn leeren Max-Heap ein.(Werte können im Heap eventuell mehrfach gespeichert sein.) Skizzieren Sie den Zustand des Heaps nach jedem Einfügeschritt.

b) [3] Geben Sie in C++ ähnlicher Notation die Definition einer möglichst effizienten Datenstruktur für einen Heap an.

c) [6] Geben Sie in C++ ähnlicher Notation eine Definition einer möglichst effizienten Funktion an, die die Tiefe des Heaps in der Baumdarstellung ermittelt und retourniert. Bestimmen Sie die Laufzeitkomplexität ihrer Funktion in Abhängigkeit von der Anzahl (n) der im Suchbaum gespeicherten Werte. Verwenden Sie dazu die Big-O-Notation.

d) [3] Geben Sie in C++ ähnlicher Notation eine Definition einer möglichst effizienten Funktion an, die ermittelt, ob es sich um einen Min-Heap oder einen Max-Heap handelt. Die Funktion soll *true* retournieren, wenn es ein Max-Heap ist, *false* sonst.

## Aufgabe 5 [20]

a) [10] Fügen Sie die Zahlen 7, 0, 4, 21, 14 und $\boldsymbol{z_8}$ (aus Beispiel 1) in dieser Reihenfolge in eine ursprünglich leere Hashtabelle der Größe 7 ein.
Verwenden Sie die Hashfunktion $\boldsymbol{h(x) = x\%7}$ und zur Kollisionsbehandlung double Hashing mit $\boldsymbol{g(x) = x\%3 + 1}$
Geben Sie den Zustand der Hashtabelle nach jeder Einfügeoperation an.

b) [2] Löschen Sie die Zahl 21 aus der Hashtabelle (geben Sie an, welche Positionen der Hashtabelle Sie besuchen müssen, um die zu löschende Zahl zu finden, sowie den Zustand der Hashtabelle nach dem Löschen).

c) [6] Suchen Sie (nach dem Löschen) die Zahlen 7, 14 und 21 in der Hashtabelle (geben Sie für jede Zahl an, welche Positionen der Hashtabelle Sie besuchen müssen, und ob die Suche erfolgreich ist, oder nicht).

d) [2] Was ist der grundlegende Unterschied zwischen statischen und dynamischen Hashverfahren?

## Aufgabe 6 [20]

Gegeben ist die folgende Adjazenzmatrix, die die Kosten der Verbindungen zwischen den Knoten eines gerichteten Graphen beschreibt (verwenden Sie die Zahlen $z_1$ bis $z_8$ aus Aufgabe 1):

$$\begin{pmatrix} 0 & z_4 & 0 & z_3 & z_2 \\ 0 & 0 & z_5 & z_8 & 0 \\ 1 & 0 & 5 & 0 & 1 \\ 0 & 0 & z_6 & 0 & z_7 \\ 41 & 0 & 2 & 0 & 9 \end{pmatrix}$$

a) [2] Skizzieren Sie den Graphen, der durch diese Adjazenzmatrix beschrieben wird.
b) [10] Bestimmen Sie mit dem Algorithmus von Dijkstra die kürzesten Wege vom Knoten 1 zu allen anderen Knoten des Graphen (Dabei entspricht Knoten 1 dem Knoten der ersten Zeile/Spalte in der Adjazenzmatrix).
c) [8] Entfernen Sie eine möglichst kleine Anzahl von Kanten, sodass der Graph topologisch sortierbar wird (führen Sie genau an, welche Kanten entfernt werden müssen) und führen Sie eine topologische Sortierung durch (es reicht, die Abfolge der Knoten anzugeben, Sie müssen nicht den Graphen für das Resultat zeichnen).