ADS theoretische Klausur 29.06.2021

20	**25**	**28**	**19**	**27**	**34**	**29**	**26**
+	+	+	+	+	+	+	+
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]

a. [6] Die Laufzeitanalyse einer Funktion hat die Rekurrenzgleichung G ergeben. Ermitteln Sie mittels des Master-Theorems eine möglichst kleine O-Schranke. Ersetzen Sie die Werte $z_n$duch die Ergebnisse aus Aufgabe 1.

$$G(n) = z_3 * G\left(\left\lfloor\frac{n}{3}\right\rfloor\right) + 4n^2 + z_7 n + 42 \text{ für } n \in \mathbb{N}$$

b. [6] Kann jede beliebige Rekurrenzgleichung mithilfe des vereinfachten Master-Theorems gelöst werden? Begründen Sie Ihre Antwort kurz.

c. [6] Analysieren Sie die Laufzeitkomplexität des folgenden Codes:

```
int xpowy(int x, int n)
{
    if (n==0)
        return 1;
    if (n==1)
        return x;
    if ((n % 2) == 0)
        return xpowy(x*x, n/2);
    else
        return xpowy(x*x, n/2) * x;
}
```

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. [8] Quicksort
b. [4] Counting Sort (**Achtung:** verwenden Sie z_n **% 10** als zu sortierende Werte!)
c. [8] Heap Sort

Aufgabe 4 [20]

a. [9] Fügen Sie die Werte z_2 bis z_8 aus Aufgabe 1 (in dieser Reihenfolge) in eine zu Beginn leere Hashtabelle der Länge 7 ein. Verwenden Sie als Hashfunktion $h(k) = k\%7$ und Double Hashing zur Kollisionsbehandlung. Die zweite Hashfunktion ist $g(k) = k\%5 + 1$.
Skizzieren Sie den Zustand der Hashtabelle nach jedem Einfügeschritt.

b. [4] Geben Sie den Kollisionspfad (besuchte Indexpositionen) bei einer Suche nach dem Wert z_4 in der Tabelle aus (a) an.

c. [4] Geben Sie den Kollisionspfad (besuchte Indexpositionen) bei einer Suche nach dem Wert 5 in der Tabelle aus (a) an.

d. [1] Löschen Sie den Wert z_5 aus der Tabelle und skizzieren Sie den Zustand der Hashtabelle.

e. [2] Wozu wird die beim Double Hashing die Markierung „wiederfrei“ verwendet?

Aufgabe 5 [20]

a. [4] Fügen Sie die Werte z_2 bis z_8 aus Aufgabe 1 (in dieser Reihenfolge) in einen zu Beginn leeren binären Suchbaum ein. Skizzieren Sie den Zustand des Baums nach jedem Einfügeschritt.
(Anmerkung: Werte können mehrfach im Baum gespeichert werden.)

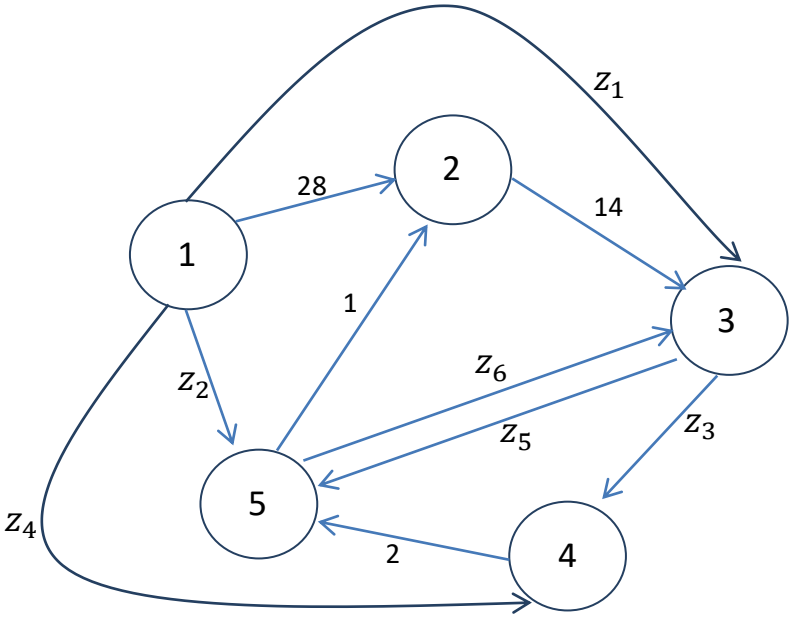
b. [4] Geben Sie in C++ ähnlicher Notation die Definition einer möglichen Datenstruktur für einen binären Suchbaum an.

c. [8] Geben Sie in C++ ähnlicher Notation eine Definition einer Funktion oder Methode an, die das Minimum der im binären Suchbaum gespeicherten Werte ermittelt.

d. [4] Bestimmen Sie die Laufzeitkomplexität Ihrer Funktion abhängig von der Anzahl *n* der im Suchbaum gespeicherten Werte in O-Notation. Begründen Sie Ihr Ergebnis kurz.

Aufgabe 6 [20]

Gegeben ist der folgende gerichtete Graph
(die Werte z_1 bis z_6. sind aus Aufgabe 1 zu übernehmen):



a. [3] Geben Sie die Adjazenzmatrix des Graphen an.
b. [3] Skizzieren Sie die Adjazenzliste des Graphen.
c. [10] Bestimmen Sie mit dem Algorithmus von Dijkstra die jeweils kürzesten Wege vom Knoten 1 (erste Zeile, erste Spalte der Matrix) zu allen anderen Knoten des Graphen. Notieren Sie Ihr Vorgehen so, dass jeder Schritt nachvollzogen werden kann.
d. [2] Ist der oben dargestellte Graph topologisch sortierbar? Falls ja, geben Sie eine topologische Sortierung an, andernfalls begründen Sie, warum eine solche nicht gefunden werden kann.
e. [2] Welche der folgenden Voraussetzungen ist hinreichend, damit der Dijkstra-Algorithmus das korrekte Resultat liefert? Zutreffendes bitte ankreuzen.

(1) Alle Kantengewichte des Eingabegraphen sind nicht-negativ.

(2) Der Eingabegraph ist ein DAG.

(3) Der Eingabegraph enthält keinen negativen Kreis.

(4) Der Eingabegraph enthält einen negativen Kreis.