

Algorithmen und Datenstrukturen (ADS VO)	schriftliche Einzelprüfung	06.12.2013		1
--	----------------------------	------------	--	---

	<b>24</b>	<b>23</b>	<b>11</b>	<b>34</b>	<b>13</b>	<b>32</b>	<b>25</b>
	+ <input type="text"/>	+ <input type="text"/>	+ <input type="text"/>	+ <input type="text"/>	+ <input type="text"/>	+ <input type="text"/>	+ <input type="text"/>
<b>21</b>							
<b>z<sub>1</sub></b>	<b>z<sub>2</sub></b>	<b>z<sub>3</sub></b>	<b>z<sub>4</sub></b>	<b>z<sub>5</sub></b>	<b>z<sub>6</sub></b>	<b>z<sub>7</sub></b>	<b>z<sub>8</sub></b>

### 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<sub>2</sub>** bis **z<sub>8</sub>**. (**z<sub>1</sub>** ist bereits mit dem fixen Wert 21 belegt.)

### Aufgabe 2 [18]

Schreiben Sie eine rekursive Funktion in C++ mit einem Parameter n (vom Typ int), deren Laufzeitkomplexität **gleichzeitig** die Ordnungen  $O(n^3)$ ,  $\Omega(n)$  und  $\Theta(n^2)$  hat.

Zeigen Sie mithilfe des Mastertheorems, dass die Laufzeitkomplexität Ihrer Funktion die gewünschten Ordnungen hat.

### Aufgabe 3 [20]

Die Werte **z<sub>1</sub>** bis **z<sub>8</sub>**. (aus Aufgabe 1) seien in dieser Reihenfolge von links nach rechts in einem Array gespeichert. Sortieren Sie die Werte aufsteigend mit

- [4] Selection Sort
- [8] Bucketsort
- [8] Quicksort

### Aufgabe 4 [20]

- [9] Fügen Sie die Werte **z<sub>2</sub>** bis **z<sub>8</sub>** 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. (Anmerkung: Werte können mehrfach in der Tabelle gespeichert werden. Die Tabelle ist statisch, wird also nicht vergrößert!)
- [1] Löschen Sie den Wert **z<sub>3</sub>** aus der Tabelle und skizzieren Sie den Zustand der Hashtabelle.
- [5] Geben Sie den Kollisionspfad (besuchte Indexpositionen) bei einer Suche nach dem Wert **z<sub>8</sub>** an.
- [5] Geben Sie den Kollisionspfad (besuchte Indexpositionen) bei einer Suche nach dem Wert 50 an.

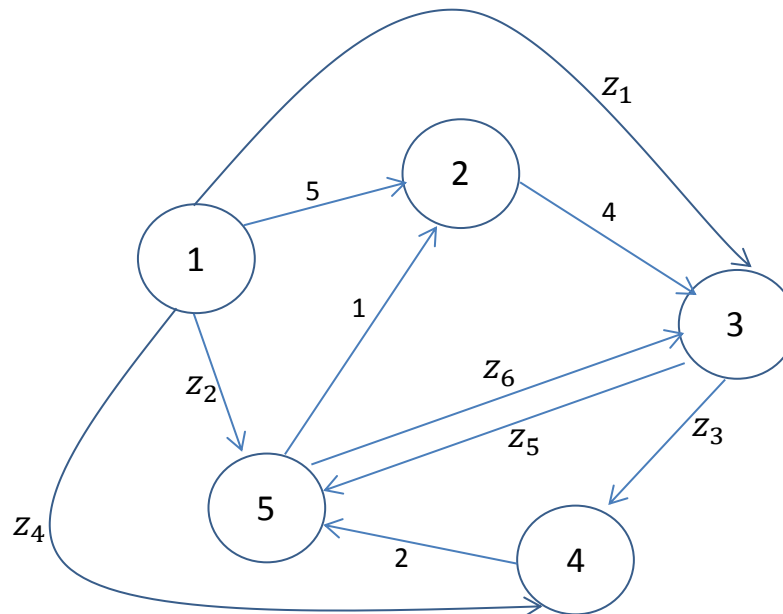
### Aufgabe 5 [20]

- [8] Fügen Sie die Werte **z<sub>2</sub>** bis **z<sub>8</sub>** aus Aufgabe 1 (in dieser Reihenfolge) in einen zu Beginn leeren Heap ein. Skizzieren Sie den Zustand des Baums nach jedem Einfügeschritt. (Anmerkung: Werte können mehrfach im Baum gespeichert werden.)
- [4] Geben Sie in C++ ähnlicher Notation die Definition einer möglichst effizienten Datenstruktur für einen Heap an.
- [4] Geben Sie in C++ ähnlicher Notation eine Definition einer Funktion an, die den Heap breadth first traversiert und alle gespeicherten Werte ausgibt.
- [4] Bestimmen Sie die Laufzeitkomplexität Ihrer Traversierungsfunktion abhängig von der Anzahl n der im Suchbaum gespeicherten Werte in  $\Theta$ -Notation.

## Aufgabe 6 [20]

Gegeben ist der folgende gerichtete Graph

(die Werte  $z_1$  bis  $z_6$  sind aus Aufgabe 1 zu übernehmen):



- [3] Geben Sie die Adjazenzmatrix des Graphen an.
- [3] Skizzieren Sie die Adjazenzliste des Graphen.
- [10] Bestimmen Sie mit dem Algorithmus von Dijkstra die jeweils kürzesten Wege vom Knoten 1 zu allen anderen Knoten des Graphen.
- [4] Ist für den Dijkstra-Algorithmus eher die Verwendung einer Adjazenzmatrix oder einer Adjazenzliste vorteilhaft? Begründen Sie Ihre Aussage.