

46115 Frühjahr 2013

Theoretische Informatik / Algorithmen / Datenstrukturen (nicht vertieft)

Aufgabenstellungen mit Lösungsvorschlägen

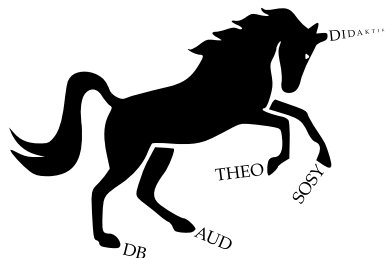


Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Aufgabenübersicht

Thema Nr. 1	3
Aufgabe 4 [Turingmaschinen]	3
 Aufgabe 4	 3
 Thema Nr. 2	 4
„Streuspeicherung“ [Hashing mit Modulo 11]	4
 „Streuspeicherung“	 4
 Aufgabe 10: Graphen I	 6
Aufgabe 6 [Schreibtischlauf Haldensortierung]	8
 Aufgabe 6	 8



Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.



Diese Materialsammlung unterliegt den Bestimmungen der Creative Commons Namensnennung-Nicht kommerziell-Share Alike 4.0 International-Lizenz.

Thema Nr. 1

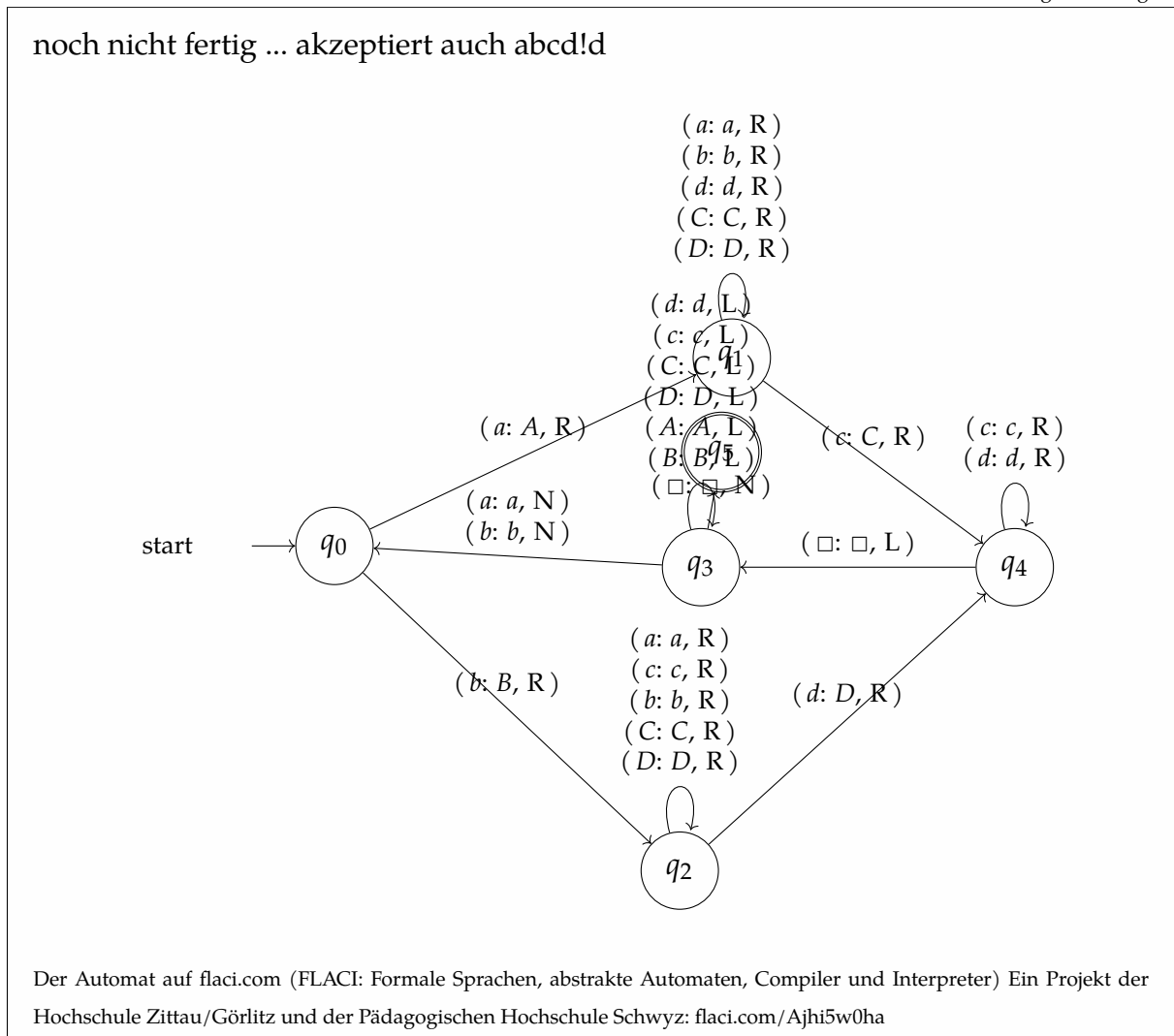
Aufgabe 4 [Turingmaschinen]

Aufgabe 4

Sei $L = \{ uv \mid u \in \{a, b\}^*, v \in \{c, d\}^*, \#_a(u) = \#_c(v) \text{ und } \#_b(u) = \#_d(v) \}$ wobei $\#_a(u)$ die Anzahl der in u vorkommenden a 's ist.

- (a) Geben Sie eine Turingmaschine M an, die L erkennt. Beschreiben Sie in Worten, wie Ihre Turingmaschine arbeitet.

Lösungsvorschlag



- (b) Welche Laufzeit (Zeitkomplexität) hat Ihre Turingmaschine (in O-Notation). Begründen Sie Ihre Angabe auf der Grundlage der Beschreibung.

Thema Nr. 2

„Streuspeicherung“ [Hashing mit Modulo 11]

„Streuspeicherung“

Die Werte 7, 0, 9, 11, 18, 4, 5, 3, 13, 24, 2 sollen in eine Hashtabelle der Größe 11 (Fächer 0 bis 10) eingetragen werden. Die zur Hashfunktion $h(x) = (7 \cdot x) \% 11$ gehörenden Schlüssel sind in der folgenden Tabelle bereits ausgerechnet:

x	7	0	9	11	18	4	5	3	13	24	2
$h(x)$	5	0	8	0	5	6	2	10	3	3	3

- (a) Fügen Sie die oben genannten Schlüssel in der vorgegebenen Reihenfolge in einen Streuspeicher ein, welcher zur Kollisionsauflösung verkettete Listen verwendet, und stellen Sie die endgültige Streutabelle dar.

Lösungsvorschlag

Index	0	1	2	3	4	5	6	7	8	9	10
Schlüssel	0		5	13		7	4		8		3
	11			24		18					
				2							

- (b) Fügen Sie die gleichen Schlüssel mit linearem Sondieren bei Schrittweite +1 zur Kollisionsauflösung in eine neue Hash-Tabelle ein. Geben Sie für jeden Schlüssel an, auf welche Felder beim Einfügen zugegriffen wird und ob Kollisionen auftreten. Geben Sie die gefüllte Streutabelle an.

Lösungsvorschlag

Index	0	1	2	3	4	5	6	7	8	9	10
Schlüssel	0	11 ₁	5	13	24 ₁	7	18 ₁	4 ₁	9	2 ₆	3

- (c) Wie hoch ist der „Load“-Faktor (die Belegung) der Hashtabelle aus a) bzw. b) in Prozent? Können Sie weitere Schlüssel einfügen?

Lösungsvorschlag

Teilaufgabe a)

$\frac{11}{11} = 100\%$: Es können allerdings weitere Elemente eingefügt werden. Die Verkettung lässt einen Loadfaktor über 100% zu. Der Suchaufwand wird dann jedoch größer.

Teilaufgabe b)

$\frac{11}{11} = 100\%$: Es können keine weiteren Elemente eingefügt werden, da alle Buckets belegt sind.

- (d) Würden Sie sich bei dieser Zahlensequenz für das Hashing-Verfahren nach a) oder nach b) entscheiden? Begründen Sie kurz Ihre Entscheidung.

Lösungsvorschlag

Das Verfahren a) scheint hier sinnvoller, da noch nicht zu viele Suchoperationen notwendig sind (max. 2), während bei Verfahren b) einmal bereits 6-mal sondiert werden muss.

Aufgabe 10: Graphen I

Gegeben seien folgende ungerichtete Graphen in textueller Notation, wobei die erste Menge die Menge der Knoten und die zweite Menge die Menge der Kanten ist:

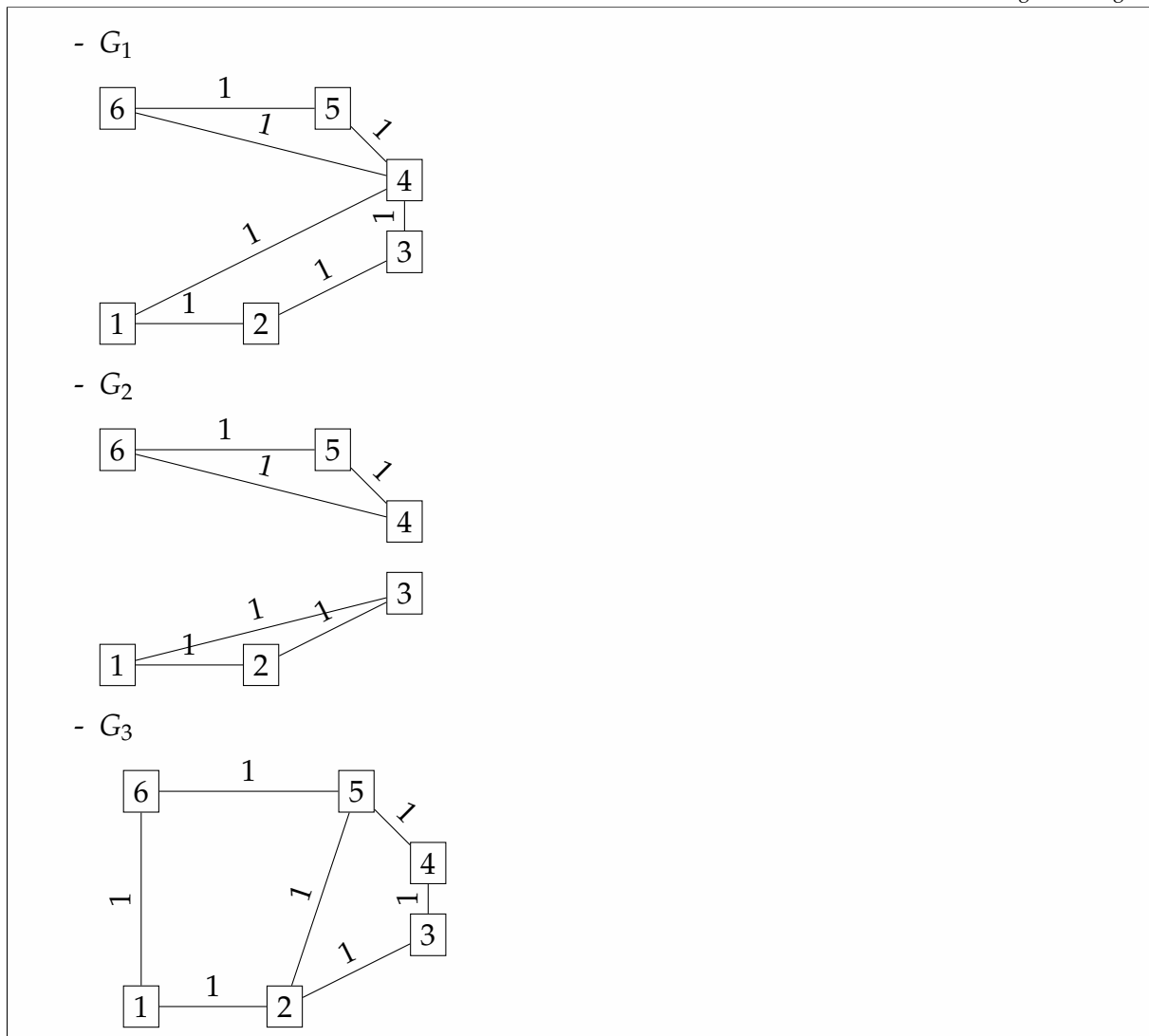
$$G_1 = (\{1, 2, 3, 4, 5, 6\}, \{[1, 2], [1, 4], [2, 3], [3, 4], [4, 5], [4, 6], [5, 6]\})$$

$$G_2 = (\{1, 2, 3, 4, 5, 6\}, \{[1, 2], [1, 3], [2, 3], [4, 5], [4, 6], [5, 6]\})$$

$$G_3 = (\{1, 2, 3, 4, 5, 6\}, \{[1, 2], [1, 6], [2, 3], [2, 5], [3, 4], [4, 5], [5, 6]\})$$

(a) Zeichnen Sie die obigen Graphen.

Lösungsvorschlag



(b) Erstellen Sie zu jedem Graphen die zugehörige Adjazenzmatrix mit X als Symbol für eine eingetragene Kante.

Lösungsvorschlag



$$\begin{array}{c}
 \begin{array}{cccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 \\
 1 & \left(\begin{array}{cccccc}
 - & X & & X & & \\
 X & - & X & & & \\
 & X & - & X & & \\
 X & & X & - & X & X \\
 & & & X & - & X \\
 & & & X & X & -
 \end{array} \right)
 \end{array}
 \end{array}$$

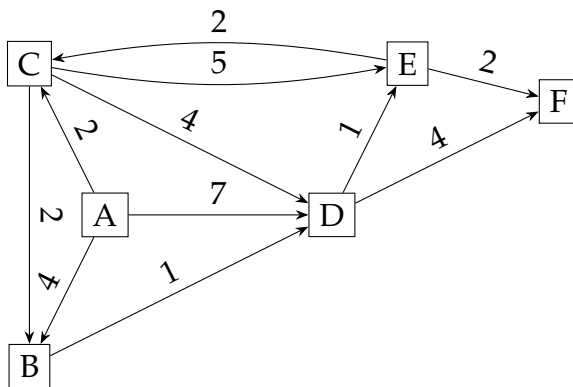
- G_2

$$\begin{array}{c}
 \begin{array}{cccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 \\
 1 & \left(\begin{array}{cccccc}
 - & X & X & & & \\
 X & - & X & & & \\
 X & X & - & & & \\
 & & & - & X & X \\
 & & & X & - & X \\
 & & & X & X & -
 \end{array} \right)
 \end{array}
 \end{array}$$

- G_3

$$\begin{array}{c}
 \begin{array}{cccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 \\
 1 & \left(\begin{array}{cccccc}
 - & X & & & X & X \\
 X & - & X & & & \\
 & X & - & X & & \\
 & & X & - & X & \\
 & & & X & - & X \\
 X & & & & X & -
 \end{array} \right)
 \end{array}
 \end{array}$$

(c) Betrachten Sie nun folgenden gerichteten Graphen G_4 :



Bestimmen Sie die kürzeste Entfernung von Knoten A zu jedem anderen Knoten des Graphen. Verwenden Sie dazu den Algorithmus von Dijkstra und tragen Sie Ihre einzelnen Rechenschritte in eine Tabelle folgender Form ein (schreiben Sie neben jede Zeile die Prioritätswarteschlange der noch zu bearbeitenden Knoten, priorisiert nach ihren Wegkosten):

Hinweis: Mit den „Wegkosten“ eines Knotens ist die gegenwärtige Entfernung dieses Knotens vom Startknoten gemeint.

Lösungsvorschlag

A	B	C	D	E	F	Warteschlange
0	∞	∞	∞	∞	∞	A
	4 (A)	2 (A)	7 (A)	∞	∞	C, B, D
			6 (C)	7 (C)	∞	B, D, E
			5 (B)	7 (C)	∞	D, E
				6 (D)	9 (D)	E, F
					8 (E)	F

Aufgabe 6 [Schreibtischlauf Haldensortierung]

Aufgabe 6

- (a) Vervollständigen Sie die folgende Sortierung mit MergeSort (Sortieren durch Mischen) — beginnen Sie dabei Ihren „rekursiven Abstieg“ immer im linken Teilfeld:

D | 40 5 89 95 85 84 || 14 25 20 52 7 71 |

Notation: Markieren Sie Zeilen mit D(ivide), in denen das Array zerlegt wird, und mit M(erge), in denen Teilarrays zusammengeführt werden. Beispiel:

D | 82 || 89 44 |

D 82 | 89 || 44 |

M 82 | 44 89 |

M | 44 82 89 |

Lösungsvorschlag

D		40		5		89		95		85		84		14		25		20		52		7		71	
D		40		5		89		95		85		84													
D		40		5		89																			
D		40		5																					
M		5		40																					
M		5		40		89																			
D								95		85		84													
D								95		85		84													
D								95		85															
M								85		95															
M								84		85		95													
M		5		40		84		85		89		95													
D														14		25		20		52		7		71	
D														14		25		20							
D														14		25									
M														14		25									
M														14		20		25							
D																				52		7		71	
D																				52		7			
M																				7		52			
M																				7		52		71	
M														7		14		20		25		52		71	
M														7		14		20		25		52		71	
M		5		7		14		20		25		40		52		71		84		85		89		95	

- (b) Sortieren Sie mittels HeapSort (Haldensortierung) die folgende Liste weiter: Notation: Markieren Sie die Zeilen wie folgt:

I: Initiale Heap-Eigenschaft hergestellt (größtes Element am Anfang der Liste).

R: Erstes und letztes Element getauscht und letztes „gedanklich entfernt“.

S: Erstes Element nach unten „versickert“ (Heap-Eigenschaft wiederhergestellt).

Lösungsvorschlag

I		99	63	91	4	36	81	76		
R		76	63	91	4	36	81		99	
S		91	63	81	4	36	76		99	
R		76	63	81	4	36		91	99	
S		81	76	63	4	36		91	99	
R		36	76	63	4		81	91	99	
S		76	36	63	4		81	91	99	
R		4	36	63		76	81	91	99	
S		63	4	36		76	81	91	99	
R		4	36		63	76	81	91	99	
S		36	4		63	76	81	91	99	
R		4		36	63	76	81	91	99	
S		4		36	63	76	81	91	99	
R		4	36	63	76	81	91	99		