Klausur Algorithmen und Datenstrukturen

Erstprüfer: Prof. Dr.-Ing. habil. Heiko Vogler
Zweitprüfer: Prof. Dr.-Ing. Franz Baader

10. Februar 2020, 14:50–16:20 Uhr

Ihre Daten – unverzüglich in Blockschrift ausfüllen –

Vorname: .....

Nachname: .....

Matrikelnummer: ..... Fachsemester: .....

Studiengang: ☐ B.Sc. Inf. ☐ B.Sc. Med.inf. ☐ Dipl.-Inf.

☐ Dipl.-IST ☐ Lehramt Inf. ☐ anderer: .....

Hinweise zur Bearbeitung

- Es sind keine Hilfsmittel zugelassen.
- Tragen Sie alle Lösungen in die dafür vorgesehenen Bereiche ein; im Ausnahmefall können Sie die leere(n) Reserveseite(n) am Ende nutzen.
- Sie können die karierten Blätter für Notizen oder Nebenrechnungen verwenden; diese werden *nicht* zur Korrektur herangezogen!
- Geben Sie am Ende *nur die Aufgabenblätter* ab.

Bewertung – von den Korrekteuren auszufüllen –

1	Programmieren in C	/ 17
2	Pulsierender Speicher	/ 17
3	Heapsort	/ 10
4	EBNF & Syntaxdiagramme	/ 16
5	Levenshtein-Distanz	/ 12
6	Algebraisches Pfadproblem	/ 15
7	EM-Algorithmus	/ 13
$\sum$	Summe	/100

....

Erstprüfer Zweitprüfer

1 Programmieren in C

Punkte	
(a)	/ 9
(b)	/ 8
$\sum$	/17

(a) Wir programmieren in C ein Spiel für einen Spieler, der ein Wort (z.B. „`geheimnis`“) erraten muss. Dazu gibt er nacheinander Buchstaben ein. Nach jedem Buchstaben enthüllt das Programm, an welchen Positionen des Wortes dieser vorkommt. Falls der Buchstabe nicht im Wort vorkommt, protokolliert das Programm einen Fehlversuch. Der Spieler gewinnt, wenn er das Wort mit weniger als 5 Fehlversuchen errät; andernfalls hat er das Spiel verloren.

Ein unvollständiger Ausschnitt des Programms ist unten gegeben. Vervollständigen Sie ihn unter Beachtung der Kommentare, sodass folgendes gilt:

- Das zu erratende Wort ist im Array `wort` und der Ratefortschritt des Spielers im Array `fortschritt` gespeichert, wobei ein '`-`' für einen noch nicht erratenen Buchstaben steht. Die Variablen `laenge` und `fehlversuche` speichern die Anzahl der Positionen von `wort` bzw. die Anzahl der bisherigen Fehlversuche.
- Vor jedem Versuch gibt das Programm `fortschritt` und `fehlversuche` aus. Anschließend gibt der Spieler ein Zeichen `eingabe` ein.
- Das Programm trägt an den Positionen von `fortschritt`, die den Vorkommnissen von `eingabe` in `wort` entsprechen, das Zeichen `eingabe` ein und erhöht ggf. `fehlversuche`.
- Nach Spielende teilt das Programm dem Spieler mit, ob er gewonnen oder verloren hat.

Sie dürfen eine Hilfsfunktion `int fertig(char fortschritt[], unsigned int laenge)` verwenden, die 1 zurückgibt, wenn das Wort komplett erraten wurde, und andernfalls 0.

```
char wort[] = "geheimnis";
char fortschritt[] = "---------";
unsigned int fehlversuche = 0;
unsigned int laenge = 9;
while (                                                    ) {
    char eingabe;
    // Bekanntgabe von `fortschritt` und `fehlversuche`




    printf("Nächster Versuch: ");
    // Eingabe einlesen


    // Eingabe prüfen; `fortschritt` bzw. `fehlversuche` anpassen
```

```
}
// Bekanntgabe des Gewinners
```

(b) Gegeben sei der folgende Datentyp zur Darstellung von Listen mit Werten vom Typ `int`.

```
typedef struct elem *list;
struct elem {
    int key;
    list next;
};
```

Geben Sie eine Funktion `void delete(list *lp, unsigned int index)` an, die das Element von `*lp`, das sich an der Position `index` befindet (beginnend bei 0), löscht und seinen Speicherbereich freigibt. Falls `index` größer oder gleich der Länge der Liste ist, soll `*lp` unverändert bleiben. *Hinweis:* Sie können davon ausgehen, dass `lp != NULL` gilt.

2 Pulsierender Speicher

Punkte	
(a)	/ 5
(b)	/12
$\sum$	/17

Gegeben sei das folgende C-Programm:

```
1  #include <stdio.h>
2  int a;
3
4  void g(int x, int *y);
5
6  void f(int *i, int j) {
7    /*label1*/
8    if (*i + j < a) {
9      *i = *i + 1;
10     f(i, j);            /* $1 */
11   }
12   /*label2*/
13 }
14
15 void g(int x, int *y) {
16   int i = 2;
17   /*label3*/
18   while (x != 1) {
19     f(&i, x);           /* $2 */
20     x = x / 2;
21     *y = *y + 1;
22     /*label4*/
23   }
24 }
25
26 int main() {
27   int x = 0;
28   scanf("%i", &a);
29   /*label5*/
30   g(a, &x);             /* $3 */
31   /*label6*/
32   return 0;
33 }
```

Objektname	Gültigkeitsbereich

(a) Tragen Sie den Gültigkeitsbereich jedes Objektes in die Tabelle ein. Nutzen Sie dazu die Zeilennummern.

(b) Führen Sie jedes der drei folgenden Speicherbelegungsprotokolle um jeweils vier Schritte weiter. Dokumentieren Sie die aktuelle Situation beim Passieren der Marken *label1* bis *label6*. Geben Sie jeweils den Rücksprungmarkenkeller und die **sichtbaren** Variablen mit ihrer Wertebelegung an. Die Inhalte von Speicherzellen nicht sichtbarer Variablen müssen Sie nur bei Änderungen eintragen. Die bereits festgelegten Rücksprungmarken sind *$1* bis *$3*.

HP	RM	Umgebung						
		1	2	3	4	5	6	7
label5	—	a 7	x 0					

HP	RM	Umgebung										
		1	2	3	4	5	6	7	8	9	10	11
label1	2:3	a 8	2	2	2	4	i 5	j 2				

HP	RM	Umgebung										
		1	2	3	4	5	6	7	8	9	10	11
label2	1:1:2:3	a 9	2	2	2	7	5	2	5	2	i 5	j 2

3 Heapsort

Punkte
$\sum$ /10

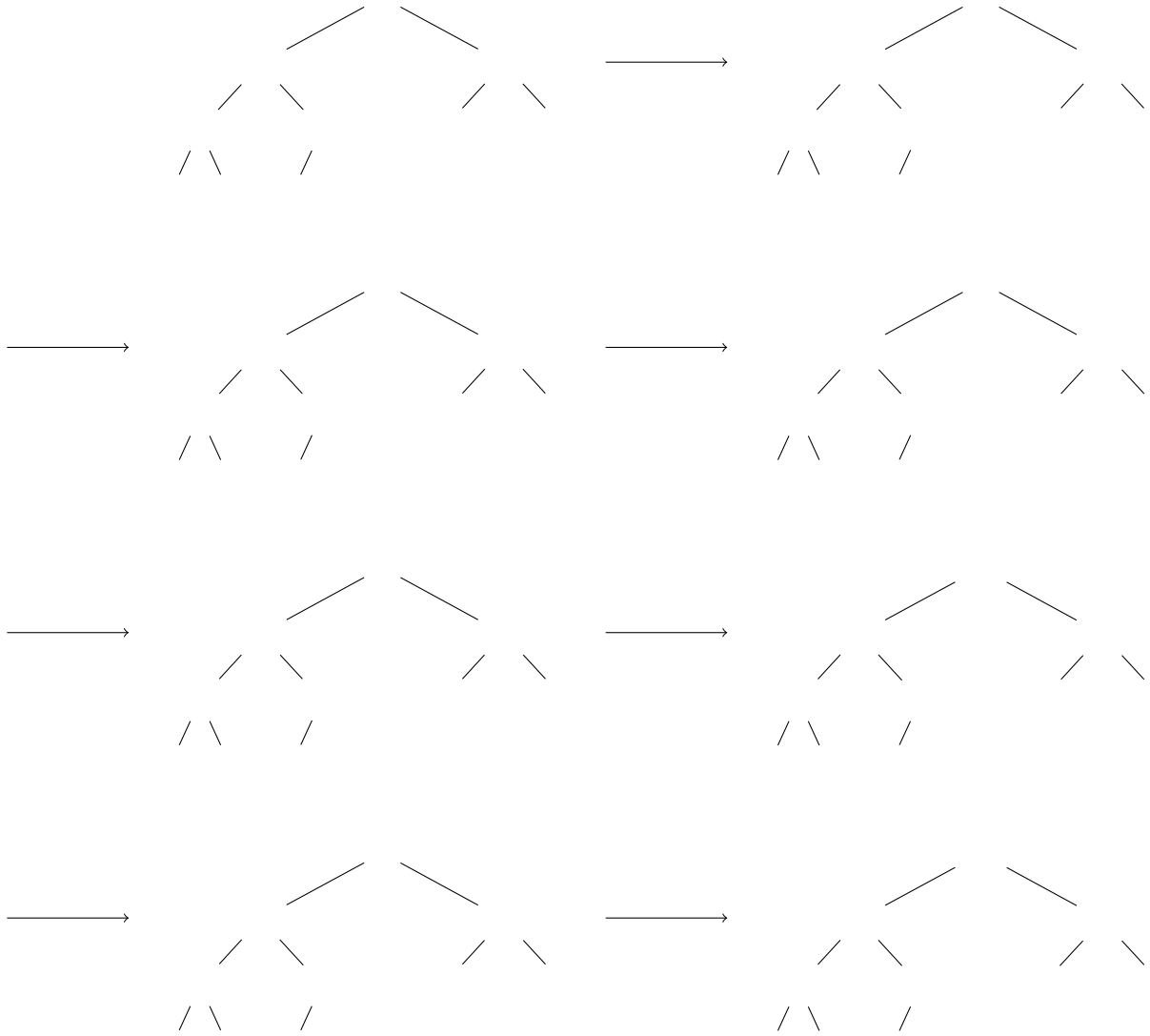
Gegeben sei die Folge 4, 10, 11, 3, 8, 13, 28, 29, 19, 21.

Wenden Sie auf diese Folge den Heapsort-Algorithmus an.

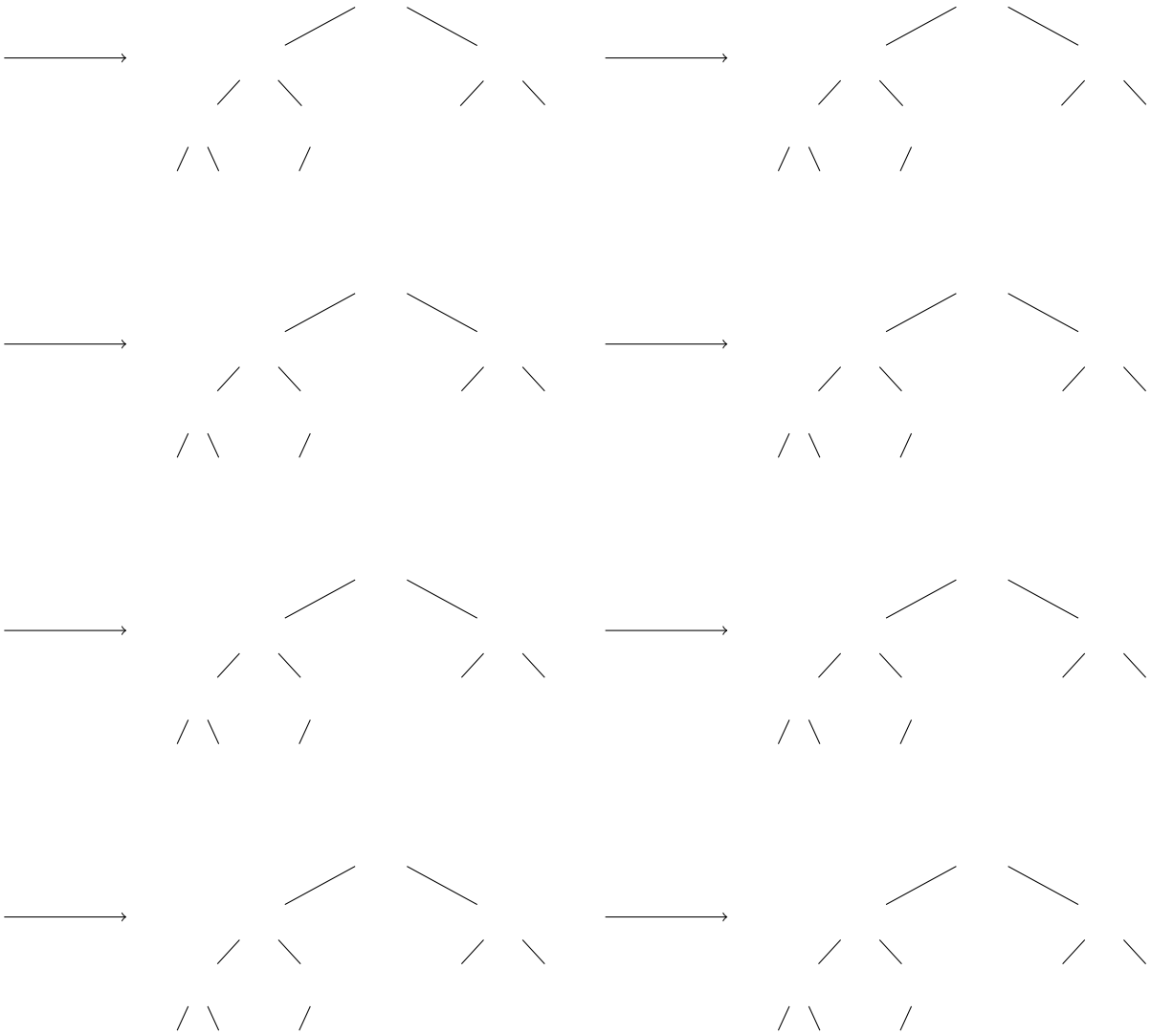
Dokumentieren Sie dazu in der Phase 1 das schrittweise Herstellen der Heap-Eigenschaft und dabei insbesondere die Veränderungen durch die Funktion „sinkenlassen“. Das Sinkenlassen in mehreren *unabhängigen* Teilbäumen darf in einem Schritt protokolliert werden.

In der Phase 2 brauchen Sie nur **zwei** Sortierschritte auszuführen. Ein Sortierschritt besteht aus einem Tausch- und einem Sinkenlassen-Schritt, die jeweils einzeln zu notieren sind.

Verwenden Sie die untenstehenden Vordrucke zur Dokumentation des Ablaufs des Algorithmus. Notieren Sie an jedem Pfeil die ausgeführte Operation. Möglicherweise benötigen Sie nicht alle Vordrucke.



(weitere Vordrucke auf der nächsten Seite)



4 EBNF & Syntaxdiagramme

Punkte	
(a)	/ 6
(b)	/10
$\sum$	/16

(a) Gegeben sei die Sprache

$$L = \{(ab)^n d^{2m} x c^{n+m} \mid n \geq 0, m \geq 1, x \in \{cd, b\}^*\}.$$

Geben Sie ein System von Syntaxdiagrammen $\mathcal{U}$ an, das die Sprache L erzeugt. Notieren Sie den Namen des Startdiagramms von $\mathcal{U}$.

Startdiagramm: .....

(b) Sei $\Sigma = \{a, c, d\}$ und $\mathcal{E} = (V, \Sigma, S, R)$ eine EBNF-Definition mit $V = \{S, A\}$ sowie

$$R = \{\ S ::= ddAc\,, \quad A ::= \widehat{[}\widehat{S]}a\ \}.$$

Geben Sie die für die Fixpunktsemantik von $\mathcal{E}$ zu iterierende Funktion f an.

$$f(\rho) = \begin{pmatrix} f(\rho)(S) \\ f(\rho)(A) \end{pmatrix} =$$

Dokumentieren Sie fünf Iterationsschritte der Fixpunktsemantik von $\mathcal{E}$.

$$\begin{pmatrix} \emptyset \\ \emptyset \end{pmatrix} \mapsto$$

Geben Sie die Sprachen $W(\mathcal{E}, S)$ und $W(\mathcal{E}, A)$ an.

$$W(\mathcal{E}, S) = \{$$

$$W(\mathcal{E}, A) = \{$$

5 Levenshtein-Distanz

Punkte	
(a)	/10
(b)	/ 2
$\sum$	/12

Gegeben seien die Wörter $w =$ MALÖR und $v =$ KNALLER.
(In der Tat ist das Wort MALÖR falsch geschrieben.)

(a) Berechnen Sie die Levenshtein-Distanz $d(w, v)$. Vervollständigen Sie dazu die bereits begonnene Berechnungsmatrix; geben Sie dabei alle Zelleneinträge zusammen mit den dazugehörigen Pfeilen an.

Berechnungsmatrix:

$d(j,i)$			K		N		A		L		L		E		R
	0	$\rightarrow$	1												
	$\downarrow$														
M	1														
A															
L															
Ö															
R															

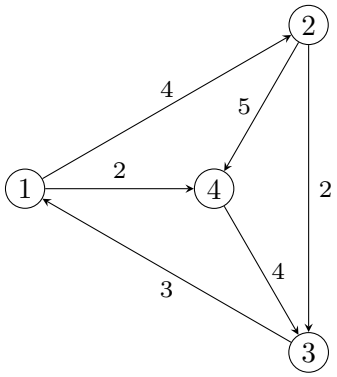
Levenshtein-Distanz:

$d(\text{MALÖR}, \text{KNALLER}) =$

b) Geben Sie zwei Alignments zwischen MALÖR und KNALLER an, die zu den minimalen Kosten führen. Dabei sollen die Alignments die jeweils angewendeten Editieroperation enthalten.

6 Algebraisches Pfadproblem

Punkte	
(a)	/ 2
(b)	/ 4
(c)	/ 8
(d)	/ 1
$\sum$	/15



Der Graph G stellt das Schienennetz einer Stadt dar, wobei jeder Streckenabschnitt nur in eine Richtung befahren werden darf. Aufgrund vieler Brücken dürfen die Züge nicht beliebig hoch sein: die Zahl an jeder Kante steht für die erlaubte Zughöhe auf dem Streckenabschnitt. Für jedes Knotenpaar (u, v) soll die maximale Höhe eines Zuges ermittelt werden, der vom Knoten u zum Knoten v verkehren darf.

(a) Um welches Pfadproblem handelt es sich? Nennen Sie den Namen des Problems und geben Sie den zugehörigen Semiring an.

(b) Geben Sie die modifizierte Adjazenzmatrix von G vollständig an.

$$mA_G = \begin{pmatrix} \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots \\ \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots \\ \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots \\ \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots \end{pmatrix}$$

(c) Die Matrix $D_G^{(2)}$ ist gegeben. Vervollständigen Sie die Einträge der Matrix $D_G^{(3)}$ und $D_G^{(4)}$!

$$D_G^{(2)} = \begin{pmatrix} \infty & 4 & 2 & 4 \\ 0 & \infty & 2 & 5 \\ 3 & 3 & \infty & 3 \\ 0 & 0 & 4 & \infty \end{pmatrix} \quad D_G^{(3)} = \begin{pmatrix} \ldots\ldots & \ldots\ldots & 2 & 4 \\ \ldots\ldots & \ldots\ldots & 2 & 5 \\ \ldots\ldots & \ldots\ldots & \infty & 3 \\ \ldots\ldots & \ldots\ldots & 4 & \infty \end{pmatrix} \quad D_G^{(4)} = \begin{pmatrix} \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots \\ \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots \\ 3 & 3 & \infty & 3 \\ 3 & 3 & 4 & \infty \end{pmatrix}$$

(d) Wie hoch darf ein Zug maximal sein, um zwischen beliebigen Knotenpaaren verkehren zu können?

7 EM-Algorithmus

Punkte	
(a)	/ 2
(b)	/ 6
(c)	/ 5
$\sum$	/13

Bei einem Spiel werden zwei Zahlen x und y zufällig gezogen, wobei x einen Wert der Menge $\{1, 2\}$ und y einen Wert der Menge $\{1, 2, 3\}$ annehmen kann. In jeder Runde werden die Zahlen gezogen; Sie beobachten aber lediglich den Restbetrag der Division $(x \cdot y) \div 4$. Die Menge der möglichen Beobachtungen ist also $Y = \{0, 1, 2, 3\}$.

a) Geben Sie den Analysator A für dieses Szenario an.

$A(0) =$ ..... $A(1) =$ .....

$A(2) =$ ..... $A(3) =$ .....

b) Sie beobachten 100 Runden des Spiels und halten Ihre Beobachtungen im folgenden Korpus h mit unvollständigen Daten fest:

$$h(0) = 20 \qquad h(1) = 27$$
$$h(2) = 25 \qquad h(3) = 28$$

Wir wollen aus diesem Korpus h mit dem EM-Algorithmus die Wahrscheinlichkeitsverteilungen der beiden Zufallszahlen bestimmen. Die initiale Wahrscheinlichkeitsverteilung $q_0 = q_0^1 \times q_0^2$ ist gegeben durch $q_0^1(1) = \frac{3}{5}$, $q_0^2(1) = \frac{3}{10}$ und $q_0^2(2) = \frac{1}{2}$. Dabei ist q_0^1 die Wahrscheinlichkeitsverteilung der 1. Zahl und q_0^2 die Wahrscheinlichkeitsverteilung der 2. Zahl. Geben Sie die Verbundwahrscheinlichkeit q_0 vollständig an!

$q_0(1,1) =$ ..... $q_0(1,2) =$ ..... $q_0(1,3) =$ .....

$q_0(2,1) =$ ..... $q_0(2,2) =$ ..... $q_0(2,3) =$ .....

Führen Sie den E-Schritt aus. Vervollständigen Sie also das Korpus h zum Korpus h_1.

$h_1(1,1) =$ ..... $h_1(1,2) =$ ..... $h_1(1,3) =$ .....

$h_1(2,1) =$ ..... $h_1(2,2) =$ ..... $h_1(2,3) =$ .....

c) Führen Sie nun den M-Schritt aus. Bestimmen Sie dafür zunächst die Teilkorpora h_1^1 und h_1^2 für die erste bzw. zweite Zufallszahl:

$h_1^1(1) =$ ..... $h_1^1(2) =$ .....

$h_1^2(1) =$ ..... $h_1^2(2) =$ ..... $h_1^2(3) =$ .....

Schätzen Sie nun die Wahrscheinlichkeitsverteilungen q_1^1 und q_1^2 der beiden Zufallszahlen, indem Sie die relative Häufigkeit der Teilkorpora bestimmen:

$q_1^1(1) =$ ..... $q_1^1(2) =$ .....

$q_1^2(1) =$ ..... $q_1^2(2) =$ ..... $q_1^2(3) =$ .....