Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst 2023

66115

Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen

— Prüfungsaufgaben —

Fach: **Informatik (vertieft studiert)**

Einzelprüfung: **Theoretische Informatik, Algorithmen**

Anzahl der gestellten Themen (Aufgaben): 2

Anzahl der Druckseiten dieser Vorlage: 18

Bitte wenden!

Thema Nr. 1
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Algorithmen

Aufgabe 1 (Sortierverfahren) [45 PUNKTE]

In der folgenden Aufgabe soll ein Feld A von ganzen Zahlen aufsteigend sortiert werden. Das Feld habe n Elemente A[0] bis A[n−1].

Gegeben seien die beiden folgenden Algorithmen, die auf dem Feld A operieren. Der Algorithmus mysterysort verwendet einen nicht näher bestimmten Parameter h. Der Algorithmus masterysort verwendet den Algorithmus mysterysort und verwendet selbst als Parameter ein Feld d von Zahlen.

```
1   void mysterysort(int h) {
2     for (int i = h; i < n; i++) {
3       int t = A[i];
4       int j = i;
5       while (j >= h && A[j-h] > t) {
6         A[j] = A[j-h];
7         j = j - h;
8       }
9       A[j] = t;
10    }
11  }
12
13  void masterysort(int[] d) {
14    for (int k=0; k < d.length; k++) {
15      int h = d[k];
16      mysterysort(h);
17    }
18  }
```

a) Wenden Sie den Algorithmus mysterysort auf das folgende Feld A für den Wert h = 3 an. Geben Sie die Belegung des Feldes nach jedem Durchlauf der **for**-Schleife in einer neuen Zeile an.

Index	0	1	2	3	4	5	6	7	8
Wert	5	9	3	7	4	1	6	8	2

b) Eine Sequenz von Zahlen heißt k-sortiert, wenn alle Untersequenzen von Zahlen mit Abstand k sortiert sind. Der Abstand ist dabei definiert als die Differenz der Indizes der Zahlen in der Sequenz. Die Sequenz 3, 2, 1, 5, 4 zum Beispiel ist 3-sortiert aber nicht 2-sortiert. Erklären Sie (möglichst in einem Satz), warum nach dem Aufruf von mysteryshort(h) das resultierende Feld h-sortiert ist.

c) Wenden Sie mysterysort auf das folgende Feld A an. Verwenden Sie allerdings nun den Parameter h = 2. Geben Sie die Belegung des Feldes nach jedem Durchlauf der äußeren Schleife in einer neuen Zeile an.

Index	0	1	2	3	4	5	6	7	8
Wert	5	4	1	6	8	2	7	9	3

d) Beweisen oder widerlegen Sie die folgende Aussage: Wenn ein Feld $a \cdot b$ - sortiert ist, dann ist es auch a - sortiert und b - sortiert.

e) Der Algorithmus masterysort bearbeitet das Feld A und ruft dabei mehrfach den Algorithmus mysterysort mit verschiedenen Werten h auf, die durch ein Eingabefeld d übergeben werden.

i. Das Feld d enthalte die Folge d = [3, 2]. Ist das Feld nach dem Aufruf von masterysort(d) garantiert sortiert? Begründen Sie Ihre Antwort.

ii. Das Feld d enthalte die Folge d = [3, 1]. Ist das Feld nach dem Aufruf von masterysort(d) garantiert sortiert? Begründen Sie Ihre Antwort.

iii. Das Feld d enthalte die Folge d = [1, 3]. Ist das Feld nach dem Aufruf von masterysort(d) garantiert sortiert? Begründen Sie Ihre Antwort.

iv. Wie muss die Folge d beschaffen sein, damit masterysort(d) das Feld A garantiert sortiert. Begründen Sie Ihre Antwort.

f) Ein Sortierverfahren heißt *stabil*, wenn es die Reihenfolge der Datensätze bewahrt, bei denen der Sortierschlüssel gleich ist.

i. Ist mysterysort für beliebige Werte von h stabil? Begründen Sie Ihre Antwort.

ii. Ist masterysort für beliebige Werte von d stabil? Begründen Sie Ihre Antwort.

Aufgabe 2 (Rekursionsgleichungen) [15 PUNKTE]

Gegeben sei folgende Rekursionsgleichung:

$$T(n) = \begin{cases} n + 2 \cdot T(n/5) & \text{falls } n > 1 \\ 1 & \text{falls } n = 1 \end{cases}$$

a) Die Funktion $T(n)$ beschreibe die Aufwandsfunktion eines Algorithmus für die Problemgröße n.
Bestimmen Sie möglichst genau und unter Angabe Ihres Lösungswegs den asymptotischen Laufzeitaufwand des Algorithmus unter Anwendung des Master-Theorems.

b) Sie können annehmen, dass n eine Potenz von 5 ist. Benutzen Sie also $n = 5^k$ oder $k = \log_5(n)$.
Zeigen Sie unter dieser Annahme mit vollständiger Induktion, dass die geschlossene Form der Rekursionsgleichung wie folgt lautet:

$$T(n) = T(5^k) = \frac{5^{k+1} - 2^{k+1}}{3}$$

Aufgabe 3 (Laufzeitanalyse) [12 PUNKTE]

Geben Sie für die folgenden Codefragmente die (möglichst genaue) asymptotische Laufzeit in Abhängigkeit von n an. Die Laufzeit der Funktion f liegt in $\Theta(1)$. Begründen Sie Ihre Antwort.

a)

```
1   for (int i = n; i ≥ 1; i = i/2 ) {
2     for(int j = 1; j ≤ i; j = 2 · j ) {
3       f();
4     }
5   }
```

b)

```
1  if (n<10) {
2     f();
3   } else {
4     for (int i=1; i≤ n; i++) {
5       f();
6     }
7   }
```

c)

```
1 int i = 0, a = 0, b = 0;
2 while (i < n) {
3   a = b + i ;
4   b = a ;
5   i = i + 1 ;
6 }
```

d)

```
1 for (int i = 1; i*i < n; i++) {
2   for (int j = n; j > 0; j--) {
3     f();
4   }
5 }
```

Aufgabe 4 (Streuspeicherung) [24 PUNKTE]

Gegeben seien die folgenden Schlüssel k zusammen mit ihren Streuwerten $h(k)$:

k	A	B	C	D	E	F	G	H
$h(k)$	1	3	7	5	6	1	5	1

a) Fügen Sie die Schlüssel in der angegebenen Reihenfolge (von links nach rechts) in eine Streutabelle mit 8 Fächern ein und lösen Sie Kollisionen durch verkettete Listen auf.

Stellen Sie die Streutabelle in folgender Art und Weise dar:

Fach	Schlüssel k (verkettete Liste, zuletzt eingetragener Schlüssel rechts)
0	
1	
2	
3	
4	
5	
6	
7	

b) Fügen Sie die gleichen Schlüssel in der gleichen Reihenfolge und mit der gleichen Streufunktion in eine neue Streutabelle mit 8 Fächern ein. Lösen Sie Kollisionen diesmal aber durch lineares Sondieren mit Schrittweite -1 auf.

Geben Sie zu jedem Schlüssel an, welche Fächer Sie in welcher Reihenfolge sondiert haben und wo der Schlüssel schlussendlich gespeichert wird.

c) Bei der doppelten Streuadressierung verwendet man eine Menge h_i von Streufunktionen. Anfangs wird Funktion h_0 verwendet. Im Falle einer Kollision wird die jeweils nächste Streufunktion verwendet. Allgemein wird nach der i-ten Kollision die Streufunktion h_i verwendet.

Die Streufunktionen h_i sind wie folgt gegeben:

k	A	B	C	D	E	F	G	H
$h_0(k)$	1	3	7	5	6	1	5	1
$h_1(k)$	3	6	4	4	7	2	6	0
$h_2(k)$	5	1	1	3	3	3	7	3
$h_3(k)$	7	4	6	2	7	4	0	6
$h_4(k)$	1	7	3	1	3	5	1	1
$h_5(k)$	3	2	0	0	7	6	2	4

Fügen Sie die Schlüssel in der angegebenen Reihenfolge (von links nach rechts) in eine Streutabelle der Größe 8 ein. Geben Sie jeweils an, welche Streufunktion Sie verwendet haben.

Aufgabe 5 (Bäume) [24 PUNKTE]

Gegeben sei die folgende Realisierung von binären Bäumen (in einer an Java angelehnten Notation):

```
1 class Node {
2   Node left, right;
3   int value;
4 }
```

a) Beschreiben Sie in möglichst wenigen Worten, was die folgende Methode foo auf einem nicht-leeren binären Baum berechnet.

```
1  int foo(Node node){
2    int b = node.value;
3    if (node.left != null) {
4      b = b + foo(node.left);
5    }
6    if (node.right != null) {
7      b = b + foo(node.right);
8    }
9    return b;
10 }
```

b) Die Laufzeit der obigen Methode foo(tree) ist linear in n, der Anzahl von Knoten im übergebenen Baum tree. Begründen Sie kurz, warum foo(tree) eine lineare Laufzeit hat.

c) Betrachten Sie den folgenden Algorithmus für nicht-leere, binäre Bäume. Beschreiben Sie in möglichst wenigen Worten die Wirkung der Methode magic(tree). Welche Rolle spielt dabei die Methode min?

```
1  void magic(Node node){
2    Node m = min (node);
3    if (m.value < node.value) {
4      // Werte von m und node vertauschen
5      int tmp = m.value;
6      m.value = node.value;
7      node.value = tmp;
8    }
9    if (node.left != null)
10     magic(node.left);
11   if (node.right != null)
12     magic(node.right);
13 }
14 Node min(Node node){
15   Node min = node ;
16   if (node.left != null){
17     Node tmp = min(node.left);
18     if (tmp.value < min.value) min = tmp ;
19   }
20   if (node.right != null){
21     Node tmp = min(node.right);
22     if (tmp.value < min.value) min = tmp ;
23   }
24   return min ;
25 }
```

d) Angenommen, der Baum ist zu einer linearen Liste degeneriert. Welche asymptotische Laufzeit hat die obige Methode magic(tree)? Die Problemgröße n sei die Anzahl der Knoten im Baum. Gezählt werden sollen lediglich Vergleiche zwischen Knotenwerten (in den Zeilen 3, 18 und 22). Begründen Sie Ihre Antwort.

e) Betrachten Sie den folgenden Algorithmus für nicht-leere, binäre Bäume. Beschreiben Sie in möglichst wenigen Worten, was ein Aufruf der Methode bar(tree) berechnet.

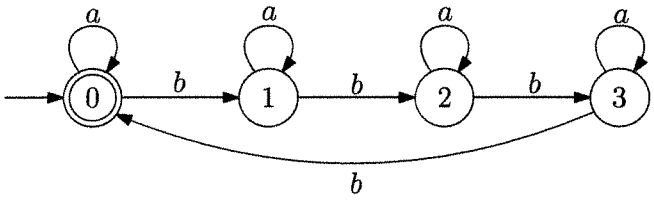
```
1  int bar(Node node){
2    int a = 0;
3    int b = 0;
4    if (node.left != null) {
5      a = bar(node.left);
6    }
7    if (node.right != null) {
8      b = bar(node.right);
9    }
10   return 1 + max(a, b) ;
11 }
```

Teilaufgabe II: Theoretische Informatik

Aufgabe 1 (Reguläre Sprachen) [25 PUNKTE]

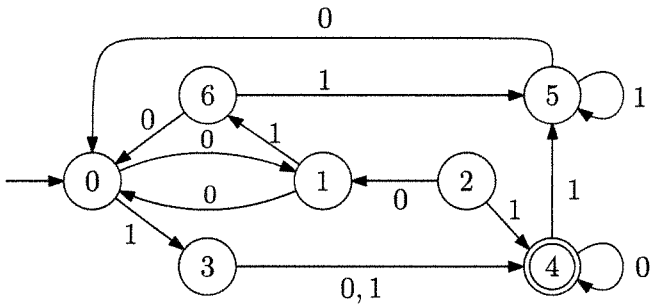
a) Geben Sie einen regulären Ausdruck an, der die Sprache aller Wörter über dem Alphabet $\{a, b, c\}$ beschreibt, die das Teilwort ab nicht enthalten.

b) Gegeben sei folgender deterministischer endlicher Automat M.



Welche Sprache akzeptiert M? Begründen Sie Ihre Antwort.

c) Konstruieren Sie für folgenden deterministischen endlichen Automaten, der Wörter über dem Alphabet $\{0, 1\}$ verarbeitet, den Minimalautomaten, das heißt einen deterministischen endlichen Automaten, der die gleiche Sprache akzeptiert und eine minimale Anzahl an Zuständen benutzt. Erläutern Sie Ihre Vorgehensweise, indem Sie zum Beispiel eine Minimierungstabelle angeben, und geben Sie das Zustandsdiagramm des Minimalautomaten an.



Aufgabe 2 (Kontextfreie Sprachen) [21 PUNKTE]

a) Sei $G = (V, \Sigma, P, S)$ mit $V = \{S, A, B\}$, $\Sigma = \{a, b\}$ und folgender Produktionsmenge P:

$$S \to AB$$
$$A \to BB \mid a$$
$$B \to AA \mid BA \mid b$$

Prüfen Sie mithilfe des CYK-Algorithmus, ob das Wort $w = abba$ in $L(G)$ liegt oder nicht. Falls ja, geben Sie zudem einen Ableitungsbaum für w an. Falls nein, geben Sie alle Teilwörter von w an, die in $L(G)$ liegen.

b) i. Sei K eine kontextfreie Sprache und sei R eine reguläre Sprache. Zeigen oder widerlegen Sie, dass $K \cdot R$ im allgemeinen kontextfrei ist.

ii. Sei K kontextfrei und sei L eine Sprache, sodass $K \cdot L$ kontextfrei ist. Zeigen oder widerlegen Sie, dass L im Allgemeinen kontextfrei ist.

Hinweis: Sie können ohne Beweis verwenden, dass die Sprache $L' = \{a^n b^m c^m \mid n \leq m\}$ nicht kontextfrei ist.

Aufgabe 3 (Chomsky-Hierarchie) [24 PUNKTE]

Sei $\Sigma = \{(,\ ),\ [,\ ]\ \}$ ein Alphabet und sei L die Sprache der korrekten Klammerausdrücke über Σ, die von der Grammatik $G = (V, \Sigma, P, S)$ mit $V = \{S\}$ und folgender Produktionsmenge P erzeugt wird:

$$S \to (S) \mid [S] \mid SS \mid () \mid []$$

Bestimmen Sie für die folgenden Teilsprachen von L jeweils, ob sie regulär sind und ob sie kontextfrei sind. Beweisen Sie Ihre Antworten. Für jede zutreffende Eigenschaft genügt es, eine geeignete Beschreibung (Grammatik/regulärer Ausdruck) oder die Arbeitsweise eines geeigneten Akzeptors (Automat/Maschine) ohne Korrektheitsbeweis zu anzugeben.

(a) $L_1 = \{w \in L$: vor jedem (in w stehen mehr [als]$\}$

(b) $L_2 = \{w \in L$: auf jede öffnende Klammer folgt direkt eine schließende Klammer$\}$

Aufgabe 4 (Entscheidbarkeit) [26 PUNKTE]

Gegeben sei das Alphabet $\Sigma = \{0, 1\}$ und eine Turingmaschine N mit $L(N) = \Sigma^*$. Bestimmen Sie für die folgenden Sprachen, ob sie entscheidbar sind und beweisen Sie jeweils Ihre Antwort. Dabei bezeichnet $\langle M \rangle$ die Gödelnummer der Turingmaschine M.

(a) $L_1 = \{\langle M \rangle \mid M$ hält auf Eingabe $\langle N \rangle$, benötigt dafür aber mindestens $2^{|\langle N \rangle|}$ Schritte$\}$

(b) $L_2 = \{\langle M \rangle \mid M$ hält auf Eingabe $\langle N \rangle\}$

(c) $L_3 = \{\langle M \rangle \mid M$ hält auf mindestens einem Wort, auf welchem N nicht hält$\}$

Aufgabe 5 (Komplexitätstheorie) [24 PUNKTE]

Sei $G = (V, E)$ ein Graph. Eine *Beinahe-3-Färbung* von G weist jedem Knoten von G eine der drei Farben $1, 2, 3$ zu, sodass für *höchstens* eine Kante von G beide Endpunkte dieselbe Farbe haben.

(a) Geben Sie einen Graphen an, der eine Beinahe-3-Färbung besitzt, aber keine 3-Färbung. Begründen Sie Ihre Antwort.

(b) Zeigen Sie, dass es NP-vollständig ist, zu entscheiden, ob ein gegebener Graph eine Beinahe-3-Färbung besitzt. Dabei dürfen Sie verwenden, dass 3-Färbung NP-vollständig ist.

Thema Nr. 2
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Algorithmen

Aufgabe 1 (O-Notation) [28 PUNKTE]

(a) Übertragen Sie die folgende Definition von $\Theta(g(n))$ auf Ihren Bearbeitungsbogen und vervollständigen Sie diese.

$\Theta(g(n)) = \{$ | es existieren Konstanten $c_1, c_2 > 0$ und $n_0 \in \mathbb{N}$,
so dass für gilt: $\}$

(b) Beweisen Sie die folgenden Aussagen mithilfe der Definition von $\Theta(g(n))$ aus a) und Angabe geeigneter Konstanten c_1, c_2, n_0.

(i) $4n(2n^2 + n) \in \Theta(n^3)$

(ii) $\log_{10}(n^3) \in \Theta(\log_2 n)$

(c) Ordnen Sie die folgenden Funktionen nach ihrem asymptotischen Wachstum, so dass $f_i \in \mathcal{O}(f_{i+1})$ gilt.

$$(n^2 + 1)(n^2 - 1), \qquad \log^8 n, \qquad n^{\log n}, \qquad 2^n, \qquad 32n^2 \log n$$

Geben Sie kurze Begründungen an.

(d) Lösen Sie folgende Rekursionsgleichung:

$$T(n) = \begin{cases} 3 \cdot T(n/3) + n^3 & \text{falls } n > 1 \\ 1 & \text{sonst.} \end{cases}$$

Aufgabe 2 (Analyse von Algorithmen) [20 PUNKTE]

Die folgende Prozedur REPLICATE erhält eine Liste S von Nullen und Einsen als Eingabe und gibt eine möglicherweise längere Liste aus. Die Prozedur S.**append**(c) hängt das Zeichen c hinten an die Liste S an.

Algorithmus 1: REPLICATE(S)

```
1  S_next := ∅
2  for i := 1 to S.length do
3  |  if S[i] = 0 then
4  |  |  S_next.append(1)
5  |  else
6  |  |  S_next.append(0)
7  |  |  S_next.append(1)
8  |  end
9  end
10 return S_next
```

Angefangen mit der Liste $S_0 = [0]$, die nur eine Null enthält, erzeugen wir Listen $S_1, S_2, \ldots$ durch wiederholtes Aufrufen der Prozedur auf der vorigen Liste, d. h. für $n \geq 1$ ist

$$S_n := \text{REPLICATE}(S_{n-1}).$$

(a) Übertragen Sie die unten angegebene Tabelle und vervollständigen Sie die Einträge für Listen S_0, S_1, S_2, S_3, S_4 und S_5 sowie deren jeweilige Anzahlen an Nullen und Einsen und die Gesamtlänge der Listen. Schreiben Sie Listen als Aufzählung von Nullen und Einsen durch Kommas getrennt, z. B. $[1, 0, 1, 0]$.

n	S_n	Anzahl Nullen in S_n	Anzahl Einsen in S_n	Länge von S_n
0	[0]	1	0	1
1	[1]	0	1	1
...	...	...	...	...

(b) Sei S_n^0 die Anzahl der Nullen in S_n und S_n^1 die Anzahl der Einsen in S_n. Drücken Sie die Werte S_n^0 und S_n^1 für $n \geq 1$ als Funktionen von S_{n-1}^0 und S_{n-1}^1 aus.

(c) Verwenden Sie Ihre Antwort zu b), um S_n^1 für $n \geq 2$ als Funktion von S_{n-1}^1 und S_{n-2}^1 auszudrücken. Drücken Sie S_n^0 für $n \geq 2$ als Funktion von S_{n-1}^0 und S_{n-2}^0 aus.

(d) Geben Sie einen Algorithmus an, der die Länge von S_n berechnet, ohne die Prozedur REPLICATE aufzurufen. Der Algorithmus darf höchstens $O(n)$ Additionen durchführen.

Aufgabe 3 (Young-Tableau als Datenstrukturen) [35 PUNKTE]

Ein $(m \times n)$-**Young-Tableau** ist eine $(m \times n)$-Matrix, deren Einträge in jeder Zeile von links nach rechts und in jeder Spalte von oben nach unten aufsteigend sortiert sind. Einige dieser Einträge können ∞ sein. Dabei interpretieren wir einen Eintrag als ∞, falls an dieser Stelle keine Zahl gespeichert ist. Dabei soll ∞ größer als jeder Eintrag sein, der nicht ∞ ist. Ein Young-Tableau kann also verwendet werden, um $r \leq mn$ Zahlen zu speichern. Ein (2×2)-Young-Tableau bzw. (2×3)-Young-Tableau, das die Zahlen $1, 2, 3, 4$ enthält, wären zum Beispiel:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \text{ bzw. } \begin{pmatrix} 1 & 2 & 3 \\ 4 & \infty & \infty \end{pmatrix}$$

(a) Geben Sie ein (4×4)-Young-Tableau an, das die Zahlen $\{9, 16, 3, 2, 4, 8, 5, 14, 12\}$ enthält.

(b) Erläutern Sie, warum ein $(m \times n)$-Young-Tableau Y leer ist, wenn $Y[1,1] = \infty$ gilt und warum Y vollständig gefüllt ist, falls $Y[m,n] < \infty$ gilt.

(c) In dieser Teilaufgabe sollen Sie eine Prozedur EXTRACTMIN entwerfen, die das minimale Element aus einem Young-Tableau löscht und zurückgibt, und dabei sicherstellt, dass die Matrix am Ende der Prozedur wieder ein Young-Tableau ist.

i. Löschen Sie das minimale Element in folgendem Young-Tableau und füllen Sie die entstandene Lücke durch Verschieben geeigneter Elemente auf, bis das Ergebnis wieder ein Young-Tableau ist. Geben Sie das entstandene Young-Tableau an.

$$Y = \begin{pmatrix} 1 & 2 & 6 & 8 \\ 3 & 4 & 9 & 12 \\ 8 & 10 & 11 & 13 \\ 9 & 12 & 14 & \infty \end{pmatrix}$$

Hinweis: Denken Sie an Verfahren zur Wiederherstellung der Heap-Eigenschaft.

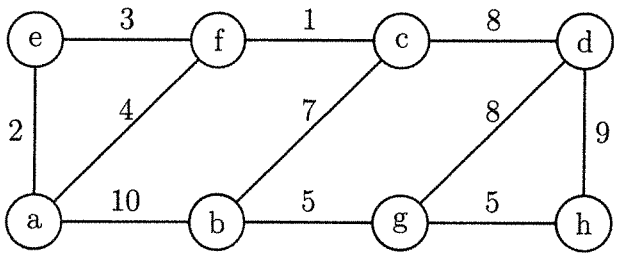
ii. Geben Sie einen Algorithmus für die Prozedur EXTRACTMIN auf einem nichtleeren $(m \times n)$-Young-Tableau in Pseudocode an. Dieser soll Zeit $O(m+n)$ benötigen. Begründen Sie, dass Ihr Algorithmus korrekt arbeitet und tatsächlich in Zeit $O(m+n)$ läuft. Geben Sie den Platzbedarf Ihres Algorithmus in O-Notation an.

(d) Erläutern Sie, wie in der Zeit $O(m+n)$ feststellt werden kann, ob eine gegebene Zahl in einem gegebenen $(m \times n)$-Young-Tableau vorkommt.

Hinweis: Überlegen Sie sich hierzu, welchen Eintrag Sie betrachten können, um direkt eine Zeile oder eine Spalte auszuschließen.

Aufgabe 4 (Spannbäume) [37 PUNKTE]

Bob ist IT-Dienstleister eines kleinen Unternehmens und seine Firma zieht in ein neues Gebäude um. Daher muss er ein kabelgebundenes Netzwerk entwerfen, das die neuen Büros verbindet. Seine Aufgabe besteht darin, die bereits vorhandenen Ethernet-Kabel in diesem Gebäude wiederzuverwenden, wie unten im verbundenen, gewichteten, ungerichtetem Graphen G^* dargestellt. Die Knoten sind die Büros und die Kanten sind die Kabelverbindungen, die Kantengewichte sind die Latenzen der Verbindungen. Bobs Ziel ist es zu entscheiden, welche Kabel verwendet werden, damit alle Büros miteinander verbunden sind, sodass die Gesamtlatenz (die Summe der Latenzen aller verwendeten Kabel) minimiert wird.



(a) Bobs Problem kann mit dem Algorithmus von Kruskal gelöst werden. Führen Sie den Algorithmus auf G^* aus, indem Sie für jeden Schritt tabellarisch mit dem unten angegebenen Schema angeben, welche Kante betrachtet wird, ob diese Kante hinzugefügt wird oder nicht und welche Zusammenhangskomponenten durch die gewählten Kanten induziert werden. Bei Wahlfreiheit wählen Sie die nächste Kante nach alphabetischer Reihenfolge des kleineren Endknotens (z. B. käme $\{a, d\}$ vor $\{b, c\}$).

Zeichnen Sie anschließend den minimalen Spannbaum und geben Sie dessen Gesamtlatenz an.

Kante	hinzugefügt?	Zusammenhangskomponenten
—	—	$\{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}, \{g\}, \{h\}$
...	...	...

(b) Im nächsten Schritt muss Bob ein Rechnernetz im Gebäude installieren. Dazu macht er sich folgende Gedanken. Sei ein beliebiger zusammenhängender gewichteter ungerichteter Graph $G = (V, E, w)$ mit minimalen Spannbaum $T = (V, E_T, w_T)$ gegeben. Dann muss in jedem Büro $u \in V$, welches Grad zwei oder mehr in T hat, ein Router installiert werden.

Beschreiben Sie kurz einen Algorithmus, der in Zeit $O(|V|)$ läuft (auch wenn z. B. $|E| = \Theta(|V|^2)$) und die Anzahl an Router ausgibt, die für T gebraucht werden. Nehmen Sie an, dass ein minimaler Spannbaum T durch die Adjazenzliste $adj^T[u]$ für $u \in V$ gegeben ist. Wie viele Router braucht Bob im obigen Graphen G^*?

(c) Nachdem Bob die Büros inspiziert hat, erkennt er folgendes Problem: Die veraltete Elektroinstallation mancher Büros verbietet die Installation eines Routers. Somit muss er den Spannbaum T aus a) so ändern, dass bestimmte Büros nur Grad 1 haben. Das Ziel ist es weiterhin, das Gesamtgewicht des Spannbaums zu minimieren.

i. Sei zunächst ein allgemeiner, zusammenhängender gewichteter ungerichteter Graph $G = (V, E, w)$ gegeben. Betrachten Sie $U \subset V$, sodass der Graph $G \setminus U$ zusammenhängend ist. Skizzieren Sie die Schritte zum Finden eines Spannbaums T mit kleinstmöglichem Gesamtgewicht, sodass alle Knoten aus U Grad 1 haben, indem Sie den Algorithmus zum Finden eines minimalen Spannbaums mit einer geeigneten Greedy-Strategie kombinieren. Erläutern Sie die Korrektheit Ihres Algorithmus.

ii. Nehmen Sie an, dass in Büros b und e keine Router installiert werden können. Zeichnen Sie den entsprechenden Spannbaum T. Wie hoch ist dessen Gesamtlatenz?

(d) Kann Bob mit Hilfe des Algorithmus aus c) das Problem lösen, falls die Büros b und c (statt b und e wie zuvor) keine Routerinstallation zulassen? Begründen Sie Ihre Antwort.

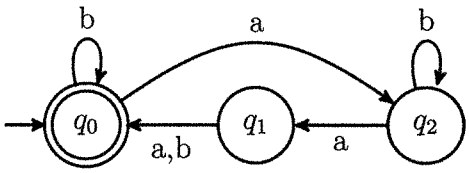
Teilaufgabe II: Theoretische Informatik

Aufgabe 1 (Konstruktion) [22 PUNKTE]

Sei M ein beliebiger nichtdeterministischer endlicher Automat (NFA), sei $v \in \Sigma^*$ und sei L die Sprache $L = \{w \in \Sigma^* \mid w \in L(M) \text{ und } w \text{ enthält } v\}$ (Ein Wort $w \in \Sigma^*$ enthält v, wenn es Wörter $u, u' \in \Sigma^*$ gibt mit $w = uvu'$.).

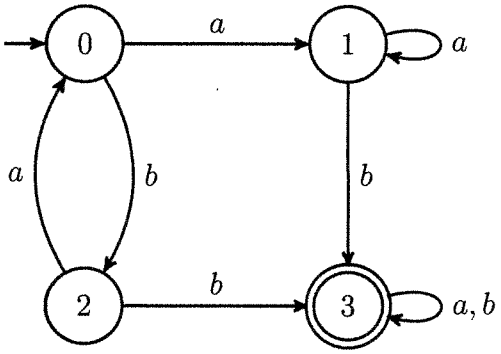
a) Beschreiben Sie ein allgemeines Verfahren, um M in einen NFA M' mit $L(M') = L$ zu konvertieren.

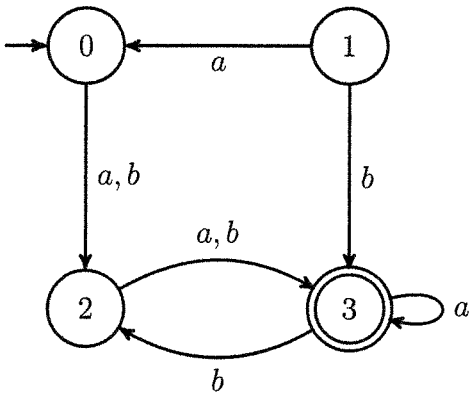
b) Führen Sie Ihr Verfahren aus Teilaufgabe a) auf dem folgenden NFA und dem Wort $v = ab$ durch.

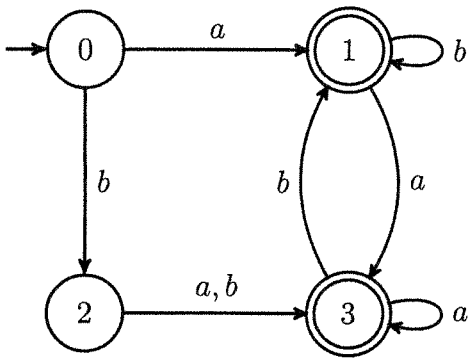


Aufgabe 2 (Minimale DFAs) [24 PUNKTE]

Wir benutzen das Alphabet $\Sigma = \{a, b\}$. Welche der folgenden DFAs sind minimal? Wenn der DFA minimal ist, zeigen Sie es mit Hilfe eines allgemeinen Verfahrens. Wenn der DFA nicht minimal ist, geben Sie einen anderen DFA mit weniger Zuständen an und erklären Sie, warum er äquivalent zum ursprünglichen DFA ist.







Aufgabe 3 (Kontextfreie Grammatiken) [24 PUNKTE]

Eine kontextfreie Grammatik ist in Chomsky-Normalform (CNF), wenn sämtliche Produktionen die Gestalt $X \to YZ$ oder $X \to a$ besitzen, wobei X, Y, Z Variablen (auch Nichtterminale genannt) und a ein Terminalzeichen ist.

a) Gegeben sei die Grammatik $G = (V, \Sigma, P, S)$ mit $V = \{S, X, Y\}$ und $\Sigma = \{a, b\}$ und folgender Produktionsmenge P:

$$S \to aXY$$
$$X \to Y \mid \epsilon$$
$$Y \to aS \mid b$$

Geben Sie eine kontextfreie Grammatik H in CNF an mit $L(H) = L(G) \backslash \{\epsilon\}$. Die Grammatik soll höchstens 5 Variablen haben. Erklären Sie Ihren Lösungsweg.

b) Geben Sie eine kontextfreie Grammatik in CNF für die Sprache $\{a^{12}\}$ an, die höchstens 5 Produktionen verwendet.

Hinweis: Betrachten Sie Produktionen der Form $X \to YY$.

c) Sei $G = (\Sigma, V, S, P)$ eine beliebige Grammatik in CNF. Konstruieren Sie eine kontextfreie Grammatik $H = (\Sigma, V', S', P')$ mit $L(H) = \{w \in L(G) : w \text{ ist gerade}\}$, d.h. $L(H)$ enthält genau die Wörter aus $L(G)$, deren Länge geradzahlig ist.

Nehmen Sie $V' := \{X_g, X_u \mid X \in V\}$ und $S' := S_g$. Sie müssen also nur die Menge P' der Produktionen von H präzise beschreiben. Erklären Sie die Idee hinter Ihrer Konstruktion.

Aufgabe 4 (Berechenbarkeit) [20 PUNKTE]

Geben Sie an, ob die folgenden Aussagen wahr oder falsch sind. Wenn die jeweilige Aussage wahr ist, beweisen Sie die Aussage; wenn sie falsch ist, geben Sie ein Gegenbeispiel an.

a) Seien M_1, M_2 beliebige Turingmaschinen und sei L eine Sprache mit $L(M_1) \subseteq L \subseteq L(M_2)$. Dann ist L semi-entscheidbar.

b) Seien L_1, L_2 semi-entscheidbar. Dann ist $L_1 \setminus L_2$ semi-entscheidbar.

c) Sei L entscheidbar. Dann ist $L' := \{uv \mid u, v \in L\}$ entscheidbar.

d) Sei L unentscheidbar. Dann ist $L' := \{u \mid \exists v\colon uv \in L\}$ unentscheidbar.

Aufgabe 5 (Reduktion) [30 PUNKTE]

Eine aussagenlogische Formel ist in konjunktiver Normalform (KNF), wenn sie eine Konjunktion von Disjunktionen von Literalen ist. Z. B. ist $(x \vee y \vee \neg z) \wedge (\neg x \vee \neg y)$ eine KNF-Formel.

Eine Formel ist *erfüllbar*, wenn sie mindestens eine erfüllende Belegung hat. Zwei Formeln über denselben Variablen sind *äquivalent*, wenn sie dieselben erfüllenden Belegungen haben, d. h. jede Belegung erfüllt beide Formeln oder keine. Wir betrachten folgende Probleme:

Negierte-KNF-SAT

Eingabe: Eine KNF-Formel F.

Ausgabe: Ist $\neg F$ erfüllbar?

KNF-Nicht-Äquivalenz

Eingabe: Zwei KNF-Formeln F und G.

Ausgabe: Sind F und G nicht-äquivalent?

Nehmen Sie P$\neq$ NP an. Welche dieser zwei Probleme sind unter dieser Annahme in P und welche sind NP-vollständig? Begründen Sie Ihre Antwort.