Prüfungsteilnehmer	Prüfungstermin	Einzelprüfungsnummer

Kennzahl: _____

Kennwort: _____

Arbeitsplatz-Nr.: _____

Herbst 2021

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 (Aufwandsanalyse) [20 PUNKTE]

a) Gegeben seien die beiden Funktionen f und g, deren asymptotische Laufzeiten bereits bestimmt wurden:

- `f(int n)` mit Laufzeit $f \in \Theta(\log_2 n)$
- `g(int n)` mit Laufzeit $g \in \Theta(n)$.

Bestimmen Sie jeweils für die folgenden drei Programmfragmente die asymptotische Laufzeit in der Θ-Notation. Begründen Sie Ihre Antwort.

(i) `for i = 1 to n do begin g(n); end`

(ii) `i = 1; while (i<n) do begin f(n); g(n); i = i*2; end`

(iii) `i = n; while (i>0) do begin f(n); i = i/2; end`

b) Nehmen Sie an, die Funktion `foo(int n)` habe eine Laufzeit von $\Theta(n)$. Welche asymptotische Laufzeit hat dann die Funktion `bar` in Abhängigkeit von n? Begründen Sie Ihre Antwort.

```
1   procedure bar(int n) : integer
2   begin
3     integer i,j,k;
4     for i := 1 to n do
5     begin
6       j := n;
7       while (j >= 1) do
8       begin
9         for k := 1 to n do
10        begin
11          foo(n);
12        end
13        j := j/2;
14      end
15    end
16  end
```

Fortsetzung nächste Seite!

c) Sortieren Sie die unten angegebenen Funktionen der O-Klassen $O(a(n))$, $O(b(n))$, $O(c(n))$ und $O(d(n))$ bezüglich ihrer Teilmengenbeziehungen. Nutzen Sie ausschließlich die echte Teilmenge $\subset$ sowie die Gleichheit $=$ für die Beziehung zwischen den Mengen. Folgendes Beispiel illustriert diese Schreibweise für einige Funktionen f_1 bis f_4 (diese haben nichts mit den unten angegebenen Funktionen zu tun):

$$O(f_4(n)) \subset O(f_3(n)) \subset O(f_1(n)) = O(f_2(n))$$

Die Beziehungen müssen weder bewiesen noch begründet werden.

- $a(n) = n^2 \cdot \log_2(n)$
- $b(n) = 2^n + n^4 + 42$
- $c(n) = 2^{n+4}$
- $d(n) = 3 \cdot \sqrt{n^5}$

d) Beweisen Sie die folgende Aussage formal durch Rückführung auf die Definition der O-Notation oder widerlegen Sie sie:

$$4n^2 + 12n + 2 \in O(n^2)$$

Aufgabe 2 (Sortierverfahren) [30 PUNKTE]

In der folgenden Aufgabe soll ein Feld `A` von ganzen Zahlen aufsteigend sortiert werden. Das Feld habe `n` Elemente `A[1]` bis `A[n]`. Der folgende Algorithmus sei gegeben:

```
1  var A : array[1..n] of integer;
2
3  procedure bubblysort
4  var i, j, tmp : integer;
5      magic : boolean;
6  begin
7    magic := true;
8    j := n-1;
9    while (j >= 1 and magic) do
10   begin
11     magic := true;
12     for i := 1 to j do
13     begin
14       if A[i] > A[i+1] then
15       begin
16         tmp := A[i];
17         A[i] := A[i+1];
18         A[i+1] := tmp;
19         magic := true;
20       end
21     end
22     j := j-1;
23   end
24 end
```

Fortsetzung nächste Seite!

a) Sortieren Sie das folgende Feld `A` mittels des Algorithmus. Geben Sie die Belegung des Feldes nach jedem Durchlauf der inneren Schleife in einer neuen Zeile auf Ihrem Bearbeitungsbogen an.

Index	1	2	3	4	5
Wert	9	5	7	3	1

Die folgenden Zeilenangaben beziehen sich auf den gegebenen Algorithmus.

b) Geben Sie die *genaue* Anzahl der in Zeile 14 durchgeführten *Vergleichsoperationen* zwischen Feldelementen als Funktion f der Eingabegröße n an. Begründen Sie Ihre Antwort. (Beachten Sie, dass hier die *genaue* Anzahl gefragt ist und *nicht* die asymptotische Komplexität.)

c) Wie muss das Feld beschaffen sein, damit möglichst wenige Vertauschungsoperationen (Zeilen 16–18) durchgeführt werden? Begründen Sie Ihre Antwort.

d) Welche Auswirkung auf das Ergebnis des Algorithmus hätte es, wenn die Bedingung in Zeile 14 wie folgt lauten würde? Begründen Sie Ihre Antwort.
```
if A[i] < A[i+1] then
```

e) Welche Auswirkung auf das Verhalten des Algorithmus hätte es, wenn die Bedingung in Zeile 14 wie folgt lauten würde? Erläutern Sie Ihre Antwort.
```
if A[i] >= A[i+1] then
```

f) Welche Auswirkung auf das Ergebnis des Algorithmus hätte es, wenn die Bedingung der `while`-Schleife in Zeile 9 wie folgt lauten würde? Begründen Sie Ihre Antwort.
```
while (j>1 and magic) do
```

g) Welche Auswirkung auf das Ergebnis des Algorithmus hätte es, wenn die Variable `magic` in Zeile 7 auf den Wert `false` initialisiert werden würde? Begründen Sie Ihre Antwort.

h) Welche Auswirkung auf das Ergebnis des Algorithmus hätte es, wenn die Zuweisung in Zeile 19 des Algorithmus `magic := false` lauten würde? Begründen Sie Ihre Antwort.

i) Welche Auswirkung auf das Ergebnis des Algorithmus hätte es, wenn die Zuweisung in Zeile 11 des Algorithmus `magic := false` lauten würde? Begründen Sie Ihre Antwort.

Fortsetzung nächste Seite!

Aufgabe 3 (Streuspeicherung) [25 PUNKTE]

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

k	?	!	X	E	T	S	O	D
$h(k)$	7	3	2	2	0	4	2	0

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

Stellen Sie die Streutabelle auf Ihrem Bearbeitungsbogen 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 ein. Lösen Sie Kollisionen diesmal durch lineares Sondieren mit Schrittweite -1 auf.

Geben Sie für jeden Schlüssel jeweils an, welche Fächer Sie in welcher Reihenfolge sondiert haben und in welchem Fach der Schlüssel schlussendlich gespeichert wird.

Fortsetzung nächste Seite!

c) Bei der doppelten Streuadressierung verwendet man eine primäre Streufunktion h und eine sekundäre Streufunktion h', deren Werte wie folgt gegeben sind:

k	?	!	X	E	T	S	O	D
$h(k)$	7	3	2	2	0	4	2	0
$h'(k)$	5	4	2	2	7	3	1	1

Die vollständige Hash-Funktion lautet dann:

$h_i(k) = (h(k) + i \cdot h'(k)) \bmod 8$.

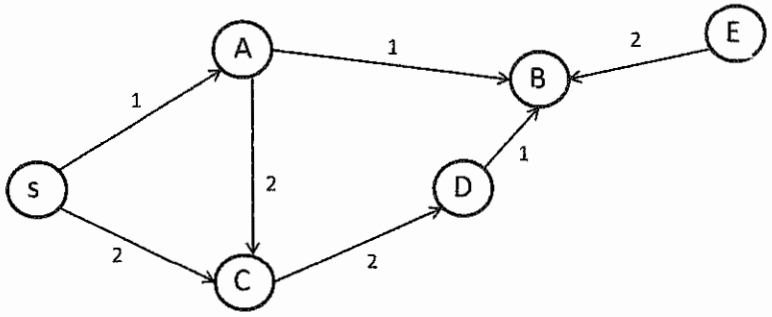
Zu Beginn ist der Index $i = 0$ und wird im Falle einer Kollision jeweils um 1 erhöht.

Fügen Sie die Schlüssel in der angegebenen Reihenfolge (von links nach rechts) in eine Streutabelle der Größe 8 ein.

Fortsetzung nächste Seite!

Aufgabe 4 (Kürzeste Wege) [10 PUNKTE]

Gegeben ist der folgende, gerichtete Graph mit Kanten- bzw. Entfernungsgewichten.



Bestimmen Sie die kürzesten Wege des Graphen vom Startknoten s aus zu allen übrigen Knoten mit Hilfe des Algorithmus von Dijkstra. Verwenden Sie dazu auf Ihrem Bearbeitungsbogen eine Tabelle der folgenden Art und markieren Sie in jeder Zeile den jeweils als nächstes zu betrachtenden Knoten. Tragen Sie das Endergebnis für alle Knoten in der untersten Zeile ein.

	s	A	B	C	D	E
Initialisierung	0	∞	∞	∞	∞	∞
1. Iteration						
Endergebnis						

Fortsetzung nächste Seite!

Aufgabe 5 (Algorithmenentwurf) [35 PUNKTE]

Gesucht ist ein Algorithmus zur Bestimmung der kleinsten Anzahl von Münzen, die nötig sind, um einen bestimmten Geldbetrag zu zahlen. Hierzu sind verschiedene ganzzahlige Münzwerte $m_1, m_2, \ldots, m_k$ mit $k > 0$, $k \in \mathbb{N}$, und $m_i \geq 1$ für $i = 1, \ldots, k$ sowie der zu zahlende ganzzahlige Betrag `b` gegeben.

Beispiel: Seien die Münzwerte $m_1 = 1$, $m_2 = 2$ und $m_3 = 5$ sowie der Betrag `b` $= 9$ gegeben, so ist die kleinste Anzahl an Münzen drei, nämlich zwei Münzen mit dem Wert 2 und einer mit dem Wert 5.

Gegeben ist der folgende Algorithmus, bei dem der auszuzahlende Betrag `b` und die Anzahl der Münzwerte `k` als ganze Zahlen übergeben werden. Die eigentlichen Münzwerte $m_1, m_2, \ldots, m_k$ werden in einem Feld `m[0]` bis `m[k-1]` übergeben. Es soll entweder die minimale Anzahl an Münzen zur Bezahlung des Betrags `b` ausgegeben werden oder -1, falls der Betrag `b` nicht mit den gegebenen Münzen auszahlbar ist.

```
1   int minimum(int m[], int k, int b) {
2     sort(m); // sortiert die Werte absteigend
3     int n = 0, c = 0;
4     while (b > 0) {
5       if (b < m[n]) {
6         n = n + 1;
7         if (n >= k) {
8           return -1;
9         }
10      } else {
11        b = b - m[n];
12        c = c + 1;
13      }
14    }
15    return c;
16  }
```

a) Führen Sie den Algorithmus mit den folgenden Eingabewerten aus: `m=[5,2,1]`, `k=3`, `b=13`. Geben Sie dazu die Werte der Variablen `b`, `n` und `c` zu Beginn eines jeden neuen Durchlaufs der `while`-Schleife sowie ganz am Ende des Algorithmendurchlaufs an.

b) Terminiert der Algorithmus für beliebige Werte von `k` und `b`? Begründen Sie Ihre Antwort. (Erinnerung: Es gilt weiterhin $m_i \geq 1$.)

c) Der Algorithmus gehört zur Klasse der "gierigen" Algorithmen (greedy). Es ist bekannt, dass diese Algorithmen nicht immer optimale Lösungen finden. Geben Sie eine Beispieleingabe für den Algorithmus an (Werte für `m`, `k` und `b`), für den der Algorithmus keine Lösung findet, obwohl eine existiert.

d) Kann es sein, dass der Algorithmus -1 ausgibt, falls einer der Münzwerte $m_i = 1$ ist? Begründen Sie Ihre Antwort.

e) Welche Auswirkung auf das Verhalten des Algorithmus hätte es, wenn die Sortierung der Münzwerte in Zeile 2 fehlen würde? Begründen Sie Ihre Antwort.

Fortsetzung nächste Seite!

f) Für das Problem soll ein rekursiver Algorithmus entwickelt werden. Als Grundlage dafür soll eine Rekursionsgleichung für $C(i, b)$ aufgestellt werden, wobei $C(i, b)$ die minimale Anzahl an Münzen ist, die benötigt wird, um den Betrag b mit den Münzwerten m_1 bis m_i zu bezahlen.

Beachten Sie, dass nicht unbedingt alle Beträge zahlbar sind. Diese Fälle sollen durch den Wert ∞ repräsentiert werden.

i) Welchen Wert hat die Rekursionsgleichung für den ersten Basisfall, bei dem der Betrag $b = 0$ ist, also $C(i, 0)$?

ii) Welchen Wert hat die Rekursionsgleichung für den zweiten Basisfall, bei dem zwar noch ein positiver Betrag aber keine Münzwerte mehr zur Verfügung stehen ($b > 0$ und $i = 0$), also $C(0, b)$?

iii) Betrachten wir nun den ersten rekursiven Fall, für den $m_i > b$ gilt und wir die Münze m_i nicht nutzen können. Geben Sie also $C(i, b)$ an für $i, b > 0$ und $m_i > b$.

iv) Für den verbleibenden Fall $m_i < b$ muss entschieden werden, ob die Münze m_i genutzt werden soll oder nicht. Beide Alternativen ergeben ggf. unterschiedliche Werte, von denen das Minimum gewählt wird. Vervollständigen Sie die folgende Rekursionsgleichung für den allgemeinen Fall:

$$C(i, b) = \min(\langle \text{Münze } m_i \text{ wird nicht gewählt} \rangle, \langle \text{Münze } m_i \text{ wird gewählt} \rangle)$$

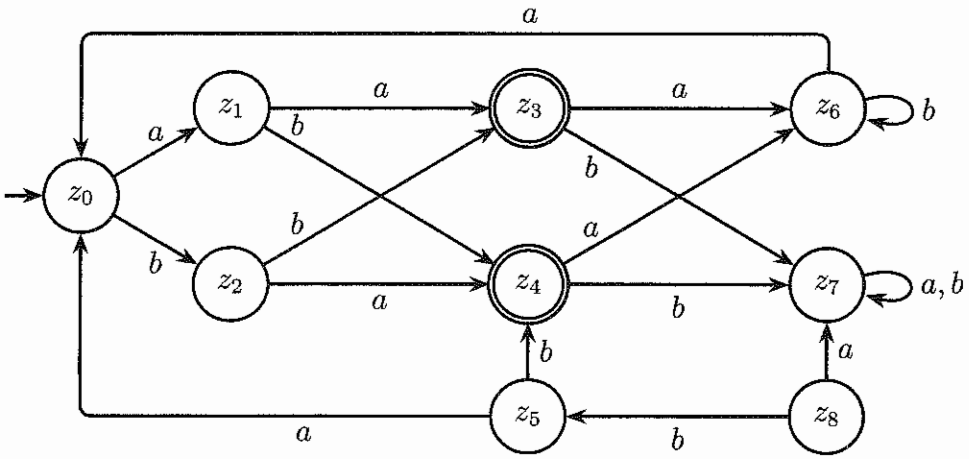
Fortsetzung nächste Seite!

Teilaufgabe II: Theoretische Informatik

Aufgabe 1 (Reguläre Sprachen und endliche Automaten) [30 PUNKTE]

a) Sei L_0 die Sprache aller Wörter $w \in \{a, b\}^*$, sodass w höchstens zwei Vorkommen von a und mindestens zwei Vorkommen von b enthält. Geben Sie ein Zustandsdiagramm eines **deterministischen endlichen Automaten** an, der L_0 erkennt.

b) Gegeben ist das Zustandsdiagramm eines deterministischen endlichen Automaten, der Worte über dem Alphabet $\Sigma = \{a, b\}$ verarbeitet. Bestimmen Sie den **Minimalautomaten**, d. h. einen deterministischen endlichen Automaten, der die gleiche Sprache akzeptiert und eine minimale Anzahl an Zuständen benutzt. Erläutern Sie Ihre Vorgehensweise, indem Sie z. B. eine Minimierungstabelle angeben, und geben Sie das Zustandsdiagramm des Minimalautomaten an.



Aufgabe 2 (Chomsky-Hierarchie) [45 PUNKTE]

Für reguläre Ausdrücke α sei $\mathfrak{L}(\alpha)$ die durch α erzeugte Sprache. Für ein Wort w und ein Symbol a bezeichne $|w|_a$ die Anzahl an Vorkommen von a im Wort w. Das leere Wort wird durch ε symbolisiert und a^i bezeichne die i-fache Wiederholung des Symbols a, d.h. $a^0 = \varepsilon$ und für $i > 0$: $a^i = aa^{i-1}$.

a) Für einen regulären Ausdruck α über einem Alphabet Σ sei die durch $\mathsf{vertausche}(\mathfrak{L}(\alpha))$ erzeugte Sprache $\mathsf{vertausche}(\mathfrak{L}(\alpha))$ definiert als

$$\mathsf{vertausche}(\mathfrak{L}(\alpha)) = \{w \mid \exists v \in \mathfrak{L}(\alpha) : \forall a \in \Sigma : |w|_a = |v|_a\}$$

D. h. der vertausche-Operator erlaubt es, die Symbolreihenfolge der in $\mathfrak{L}(\alpha)$ liegenden Worte beliebig zu ändern.

Beweisen oder widerlegen Sie für jede der folgenden Sprachen, dass diese *regulär* sind.

i) $L_1 = \mathsf{vertausche}(\mathfrak{L}(aab \mid abb))$

ii) $L_2 = \mathsf{vertausche}(\mathfrak{L}(a^*bb))$

iii) $L_3 = \mathsf{vertausche}(\mathfrak{L}((ab)^*))$

Fortsetzung nächste Seite!

b) Sei L_4 die Sprache

$$L_4 = \{a^{3i}b^{2i}c^i \mid i \in \mathbb{N}, i > 0\}$$

i) Beweisen Sie, dass die Sprache L_4 **nicht kontextfrei** ist.

ii) Zeigen Sie, dass die Sprache L_4 **vom Typ 1** ist, indem Sie eine Typ 1-Grammatik für L_4 angeben. Erläutern Sie für Ihre Grammatik insbesondere, wie Worte aus L_4 damit hergeleitet werden, und warum Worte, die nicht in L_4 liegen, nicht hergeleitet werden können.

Aufgabe 3 (NP-Vollständigkeit) [25 PUNKTE]

Das CLIQUE-Problem kann in der gegeben/gefragt-Notation (als Entscheidungsproblem) definiert werden durch:

CLIQUE

gegeben: Ein ungerichteter Graph $G = (V, E)$ und eine natürliche Zahl $k \geq 1$.

gefragt: Hat G eine k-Clique? D.h.: Gibt es eine Knotenmenge $K \subseteq V$ mit $|K| \geq k$, sodass alle Knoten in K paarweise verbunden sind (für alle $u, v \in K$ gilt: Wenn $u \neq v$, dann $\{u, v\} \in E$)?

Das CLIQUE-Problem ist **NP**-vollständig.

Ein **blau-weiß-gefärbter Graph** ist ein Tupel (V, E, col), wobei (V, E) ein ungerichteter Graph ist und col eine Knotenfärbung mit den Farben blau und weiß ist, d.h. für alle Knoten $v \in V$ gilt $col(v) \in \{\text{blau}, \text{weiß}\}$. Das BLAUE-UND-WEISSE-CLIQUE-Problem ist definiert als:

BLAUE-UND-WEISSE-CLIQUE

gegeben: Ein blau-weiß-gefärbter Graph $G = (V, E, col)$ und eine natürliche Zahl $k \geq 1$.

gefragt: Hat G **sowohl** eine blaue **als auch** eine weiße k-Clique? D.h.: Gibt es Knotenmengen $K_1, K_2 \subseteq V$ mit $|K_1| \geq k$ und $|K_2| \geq k$, sodass alle Knoten in K_1 blau sind ($\forall v \in K_1 : col(v) = \text{blau}$), alle Knoten in K_2 weiß sind ($\forall v \in K_2 : col(v) = \text{weiß}$) und für $i = 1, 2$: alle Knoten in K_i paarweise verbunden sind (für alle $u, v \in K_i$ gilt: Wenn $u \neq v$, dann $\{u, v\} \in E$)?

a) Zeigen Sie, dass BLAUE-UND-WEISSE-CLIQUE $\in$ **NP** gilt.

b) Zeigen Sie, dass das BLAUE-UND-WEISSE-CLIQUE-Problem **NP-schwer** ist, indem Sie eine **Polynomialzeitreduktion** von CLIQUE auf BLAUE-UND-WEISSE-CLIQUE durchführen.
Hinweis: Der Begriff NP-schwer ist gleichbedeutend mit dem Begriff NP-hart.

Fortsetzung nächste Seite!

Aufgabe 4 (Berechenbarkeit) [20 PUNKTE]

Beweisen oder widerlegen Sie für die folgenden Sprachen, dass diese **entscheidbar** sind. Die Gödel-Nummer der Turingmaschine M wird dabei mit $\langle M \rangle$ bezeichnet.

a) $L_5 = \{\langle M \rangle \mid M$ macht für jede Eingabe $w \in \{a, b\}^*$ mindestens 42 Schritte$\}$

b) $L_6 = \{\langle M \rangle \mid M$ hält nicht bei Eingabe ba und M macht mehr als 20 Schritte bei Eingabe $aa\}$

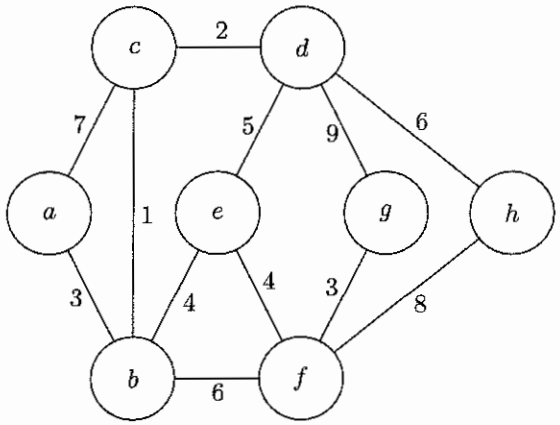
Thema Nr. 2
(Aufgabengruppe)

Es sind alle Aufgaben dieser Aufgabengruppe zu bearbeiten!

Teilaufgabe I: Algorithmen

Aufgabe 1 (Minimale Spannbäume) [32 PUNKTE]

Gegeben sei folgender ungerichteter Graph $G = (V, E)$ mit Kantengewichten $\omega\colon E \to \mathbb{N}$.



a) Berechnen Sie mithilfe des Jarník-Prim-Algorithmus einen minimalen Spannbaum T von G beginnend von Knoten a. Bearbeiten Sie Knoten bei Wahlfreiheit in alphabetischer Reihenfolge.

   Erstellen Sie dazu eine Tabelle mit zwei Spalten und stellen Sie jeden einzelnen Schritt des Verfahrens in einer eigenen Zeile dar. Geben Sie in der ersten Spalte denjenigen Knoten v an, der vom Algorithmus als nächstes in T aufgenommen wird (der neue "schwarze" Knoten). Führen Sie in der zweiten Spalte alle anderen vom aktuellen Baum T direkt erreichbaren Knoten v (die "grauen" Knoten) auf, die noch nicht im aktuellen Baum enthalten sind. Notieren Sie Knoten immer zusammen mit deren Vorgänger und der Distanz zum Vorgänger als Tripel: (Knoten, Vorgänger, Distanz).

b) Zeichnen Sie den minimalen Spannbaum aus Teilaufgabe a) und geben Sie sein Gewicht an.

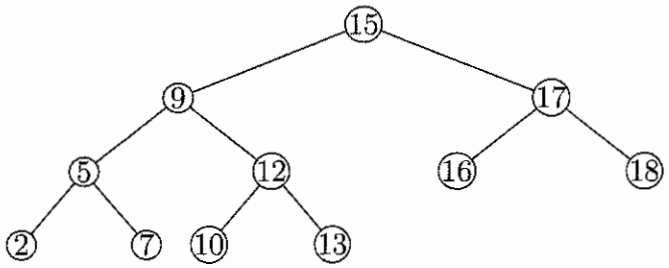
c) Begründen Sie, dass G genau einen minimalen Spannbaum hat.

d) Wählen Sie ein neues Gewicht für eine Kante, so dass es mindestens drei minimale Spannbäume gibt. Begründen Sie Ihre Antwort oder geben Sie drei minimale Spannbäume explizit an.

Fortsetzung nächste Seite!

Aufgabe 2 (Binäre Suchbäume) [17 PUNKTE]

Gegeben sei folgender binärer Suchbaum:



a) Fügen Sie den Wert 6 in den binären Suchbaum ein und stellen Sie den resultierenden binären Suchbaum graphisch dar.

b) Entfernen Sie den Wert 15 aus dem Suchbaum, sodass die binäre Suchbaumeigenschaft erhalten bleibt. Stellen Sie Ihr Ergebnis graphisch dar. Gehen Sie dabei vom ursprünglichen Suchbaum aus und *nicht* von Ihrem Ergebnis aus Teilaufgabe a).

c) Gegeben sei folgender Algorithmus, der über den Aufruf IsBinarySearchTree(T.root) prüfen soll, ob ein gegebener Binärbaum T mit paarweise verschiedenen Werten die Suchbaum-Eigenschaft erfüllt.

```
1 IsBinarySearchTree(node)
2 if node == null then
3     return true
4 else if node.left != null and node.left.key ≥ node.key then
5     return false
6 else if node.right != null and node.right.key ≤ node.key then
7     return false
8 else
9     return IsBinarySearchTree(node.left) and IsBinarySearchTree(node.right)
```

Geben Sie ein Beispiel an, für das der Algorithmus ein falsches Ergebnis liefert und erklären Sie, wo der Fehler liegt.

d) Beschreiben Sie einen alternativen Weg, wie tatsächlich in linearer Laufzeit überprüft werden kann, ob ein gegebener Binärbaum T mit paarweise verschiedenen Werten die Suchbaum-Eigenschaft erfüllt.

Aufgabe 3 ($\mathcal{O}$ -Notation) [25 PUNKTE]

a) Gilt zwischen den folgenden Mengen eine Teilmengenbeziehung? Ist diese echt? Geben Sie einen formalen Beweis an.

$$\mathcal{O}(n^2 \log(n)) \qquad \mathcal{O}(\sqrt{n^5})$$

Fortsetzung nächste Seite!

b) Ordnen Sie folgende Funktionen paarweise (soweit möglich) bezüglich der $\mathcal{O}$-Notation. Geben Sie jeweils eine kurze Begründung an. Schreiben Sie $f \preceq g$ gdw. $f \in \mathcal{O}(g)$. Die Transitivität von $\preceq$ muss nicht gezeigt werden. Geben Sie auch an, welche Funktionen hinsichtlich $\preceq$ unvergleichbar oder äquivalent sind.

- $f_1(n) = \begin{cases} n^2, & \text{wenn } n \text{ gerade} \\ 2^n, & \text{wenn } n \text{ ungerade} \end{cases}$
- $f_2(n) = n!$
- $f_3(n) = \log(n!)$
- $f_4(n) = n^3$

Aufgabe 4 (Quicksort) [46 PUNKTE]

Gegeben sei folgende Implementierung des QuickSort-Verfahrens für Arrays mit *paarweise verschiedenen* Elementen in Pseudocode:

```
1 QuickSort(A, ℓ = 1, r = A.length)
2 if ℓ < r then
3 |   i =Partition(A, ℓ, r, FindPivot(A, ℓ, r))
4 |   QuickSort(A, ℓ, i − 1)
5 |   QuickSort(A, i + 1, r)
```

Die nicht angegebene Funktion FindPivot wählt hierbei einen beliebigen Index aus dem übergebenen Bereich als Pivot-Element aus.

a) Vervollständigen Sie den Algorithmus, indem Sie eine passende Implementierung von Partition$(A, \ell, r, \textit{pivotIndex})$ als Pseudocode angeben. Für die Laufzeit soll (ohne Beweis) gelten, dass Partition$(A, \ell, r, \textit{pivotIndex}) \in \Theta(r - \ell)$. Beachten Sie auch die Verwendung des Rückgabewerts ab Zeile 3. Es ist keine Fehlerbehandlung für ungültige Eingaben verlangt.

b) Wenden Sie den oben angegebenen QuickSort auf das Array $[3, 7, 1, 8, 5, 2, 6, 4]$ an. Als Pivot-Element soll hierbei die Funktion FindPivot immer das am weitesten rechts stehende Element auswählen, d.h. FindPivot$(A, \ell, r) = r$. Geben Sie nach jedem Partition-Aufruf den betrachteten Arrayabschnitt zusammen mit den Parametern ℓ, r, und A[pivotIndex] an. Für Arrayabschnitte mit zwei Elementen oder weniger genügt es, das Ergebnis direkt anzugeben.

c) Nun soll die QuickSort-Funktion so abgeändert werden, dass sie in möglichst kurzer Zeit das k-kleinste Element des Arrays findet, ohne dass das Array fertig sortiert sein muss. Gesucht ist also das Element, das an der k-ten Position landen würde, wenn man das Array komplett fertig sortieren würde. Geben Sie den entsprechend veränderten Algorithmus an.

Hinweis: Kann nach dem Aufruf von Partition das Pivot-Element nochmal seinen Platz wechseln?

Fortsetzung nächste Seite!

d) Gehen Sie nun davon aus, dass die Funktion FindPivot in linearer Laufzeit den Index des Medians des übergebenen Bereichs zurückgibt. Analysieren Sie die Laufzeit Ihrer veränderten Implementierung in $\mathcal{O}$-Notation.

Hinweis: Für eine endliche Menge von Zahlen mit n Elementen ist der Median wie folgt definiert: Betrachtet man die Folge der der Größe nach aufsteigend sortierten Zahlen $x_1, ..., x_n$, so ist der Median

- $x_{(n+1)/2}$, falls n ungerade ist,
- $x_{n/2}$, falls n gerade ist.

Teilaufgabe II: Theoretische Informatik

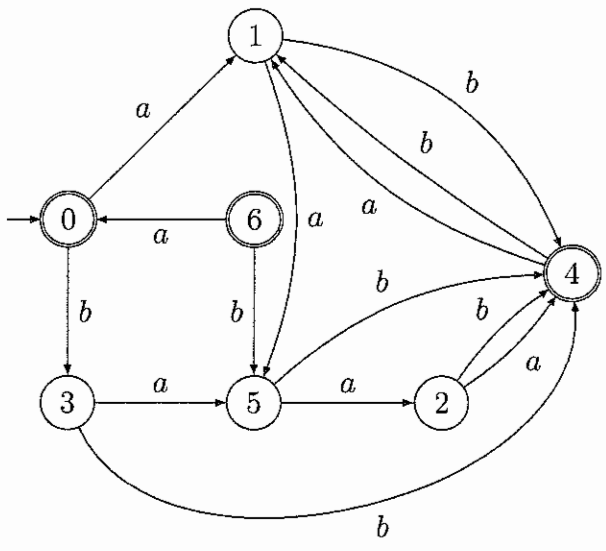
Aufgabe 1 (Reguläre Sprachen) [26 PUNKTE]

a) Geben Sie einen möglichst kurzen regulären Ausdruck an, der genau alle Wörter über dem Alphabet $\{a, b\}$ akzeptiert, die nicht das Teilwort ab beinhalten.

b) Geben Sie einen möglichst kurzen regulären Ausdruck an, der genau alle Wörter über dem Alphabet $\{a, b\}$ akzeptiert, die genau einmal das Teilwort ab beinhalten.

c) Geben Sie einen deterministischen endlichen Automaten mit möglichst wenigen Zuständen an, der genau alle Wörter über dem Alphabet $\{a, b, c\}$ akzeptiert, die genau einmal das Teilwort ab beinhalten.

d) Minimieren Sie den folgenden deterministischen endlichen Automaten.



Fortsetzung nächste Seite!

Aufgabe 2 (Regularität und Kontextfreiheit) [34 PUNKTE]

a) Zeigen Sie, dass die Sprache

$$L_1 = \{a^n b^m \mid n, m \in \mathbb{N}, n \leq m\}$$

nicht regulär ist.

b) Ist $L_2 = \{w_1 x w_2 y w_3 \mid w_1, w_2, w_3 \in \{a\}^* \cup \{b\}^*, x, y \in \{a, b\}, |w_1| = |w_2| = |w_3|, x \neq y\}$ eine kontextfreie Sprache? Beweisen Sie Ihre Antwort.

c) Zeigen Sie mit dem CYK-Algorithmus, dass das Wort $w = caaaabc$ in der Sprache der Grammatik $G = (\{A,B,C,S\}, \{a,b,c\}, P, S)$ mit

$$\begin{aligned} P = \{&S \to SA \mid CC, \\ &A \to CC, \\ &B \to SC \mid b, \\ &C \to BB \mid a \mid c\} \end{aligned}$$

enthalten ist. Geben Sie zwei verschiedene Ableitungsbäume von w für die Grammatik G an.

Aufgabe 3 (Entscheidbarkeit) [30 PUNKTE]

Seien L, L_1 und L_2 Sprachen über dem Alphabet Σ. Sei $\overline{L}$ das Komplement von L, das heißt $\overline{L} = \{w \in \Sigma^* \mid w \notin L\}$.

a) Definieren Sie, was man unter einer entscheidbaren Sprache L versteht.

b) Definieren Sie, was man unter einer semi-entscheidbaren Sprache L versteht.

c) Zeigen Sie: Falls eine Sprache L und ihr Komplement $\overline{L}$ semi-entscheidbar sind, dann ist die Sprache L entscheidbar.

d) Zeigen Sie: Falls eine Sprache L semi-entscheidbar ist, sind die Sprachen $L \cap \overline{L}$ und $L \cup \overline{L}$ immer entscheidbar.

e) Sei M_w die Turingmaschine, die durch einen String w kodiert wird. Sei H die Sprache $\{w \mid M_w$ akzeptiert die Eingabe $w\}$. Beweisen Sie, dass H eine unentscheidbare Sprache ist.

Fortsetzung nächste Seite!

Aufgabe 4 (Komplexität) [30 PUNKTE]

Betrachten Sie die folgenden Probleme, die in Abhängigkeit einer Zahl $k \in \{1, 2, 3, \ldots\}$ definiert sind:

EXACT-COVER-BY-k	
Gegeben:	Eine endliche Menge X mit $\lvert X\rvert \bmod k = 0$ und eine Menge $\mathcal{C}$ von k-elementigen Teilmengen von X, d.h. $\forall C \in \mathcal{C}$: $\lvert C\rvert = k$ und $C \subseteq X$.
Frage:	Gibt es eine Teilmenge $\mathcal{S}$ von $\mathcal{C}$, sodass $\mathcal{S}$ eine Partition von X ist?

Hinweis: $\mathcal{S}$ ist eine Partition von X genau dann, wenn $X = \bigcup_{S \in \mathcal{S}} S$ und alle Mengen in $\mathcal{S}$ paarweise disjunkt sind.

Beispiel: Sei $X = \{1, 2, 3, 4, 5, 6\}$. Dann ist $\lvert X\rvert \bmod 3 = 0$ und eine Menge von 3-elementigen Mengen $\mathcal{C}$ beispielsweise $\{\{1, 2, 3\}, \{1, 3, 4\}, \{2, 3, 5\}, \{1, 5, 6\}\}$. Jedoch gilt $(X, \mathcal{C}) \notin$ EXACT-COVER-BY-3, da es keine Menge $\mathcal{S} \subseteq \mathcal{C}$ gibt, in der jedes Element von X exakt einmal vorkommt.

a) Zeigen Sie, dass EXACT-COVER-BY-4 in NP ist.

b) Nehmen Sie an, dass EXACT-COVER-BY-3 (EC3) NP-vollständig ist und zeigen Sie damit, dass EXACT-COVER-BY-4 (EC4) NP-schwer ist.

Hinweis: Der Begriff NP-schwer ist gleichbedeutend mit dem Begriff NP-hart.