

46115 Frühjahr 2021

Theoretische Informatik / Algorithmen / Datenstrukturen (nicht vertieft)

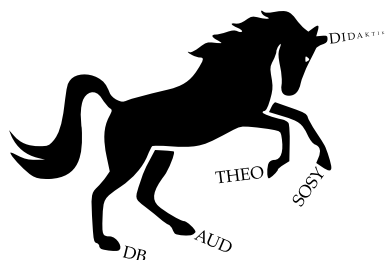
Aufgabenstellungen mit Lösungsvorschlägen



Die Bschlangaul-Sammlung
Hermine Bschlangaul and Friends

Aufgabenübersicht

Thema Nr. 1	4
Teilaufgabe Nr. 1	4
Aufgabe 2 [Nonterminale SABCD, Terminale ab]	4
Aufgabe 3 [Minimierung von Endlichen Automaten]	4
Teilaufgabe Nr. 2	5
Aufgabe [Pseudo-Code Insertionsort, Bubblesort, Quicksort]	5
Aufgabe 2 [Minimum und Maximum]	6
Aufgabe 3 [Schwach zusammenhängend gerichteter Graph]	10
Aufgabe 5 [Sondierfolgen für Hashing mit offener Adressierung]	11
Thema Nr. 2	13
Teilaufgabe Nr. 1	13
Aufgabe 1 [Alphabet abc]	13
Aufgabe 2 [NEA ab]	14
Aufgabe 3 [L1, L2, L3 regulär oder kontextfrei]	15
Teilaufgabe Nr. 2	16
Aufgabe 1 [O-Notation $a()$, $b()$, $c()$, $d()$, $e(n)$]	16
Aufgabe 2 [Java-Klasse Stack]	17
Aufgabe 3 [Lineare und Binäre Suchverfahren]	19
Aufgabe 4 [Graph a-i]	20



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

Teilaufgabe Nr. 1

Aufgabe 2 [Nonterminale SABCD, Terminale ab]

- (a) Verwenden Sie den Algorithmus von Cocke, Younger und Kasami (CYK-Algorithmus), um für die folgende kontextfreie Grammatik $G = (V, D, P, S)$ mit Variablen $V = \{s, A, B, C, D\}$, Terminalzeichen $\Sigma = \{a, b\}$, Produktionen

$P = \{S \rightarrow SB \mid AC \mid a, A \rightarrow a, B \rightarrow b, C \rightarrow DD \mid AB, D \rightarrow AB \mid DC \mid CD\}$

und Startsymbol S zu prüfen, ob das Wort $aabababb$ in der durch G erzeugten Sprache liegt. Erläutern Sie dabei Ihr Vorgehen und den Ablauf des CYK-Algorithmus.

- (b) Mit a^i , wobei $i \in \mathbb{N}_0 = \{0, 1, 2, \dots\}$, wird das Wort bezeichnet, das aus der i -fachen Wiederholung des Zeichens a besteht (d. h. $a^0 = \epsilon$ und $a^i = aa^{i-1}$!, wobei ϵ das leere Wort ist).

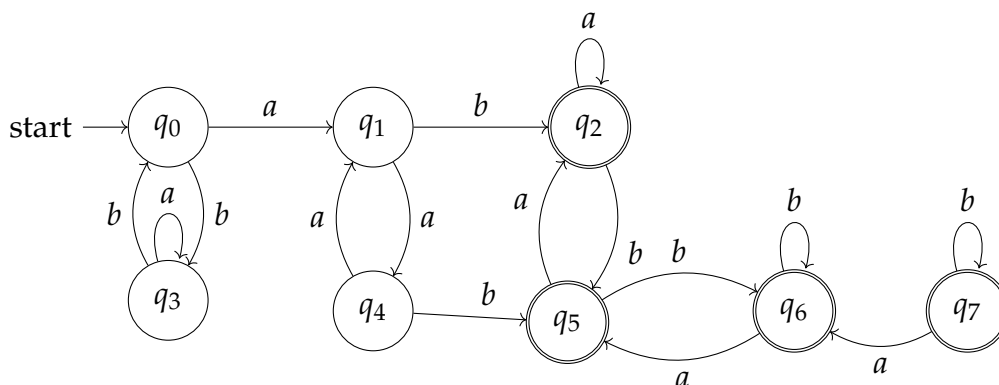
Sei die Sprache L definiert über dem Alphabet $\{a, b\}$ als

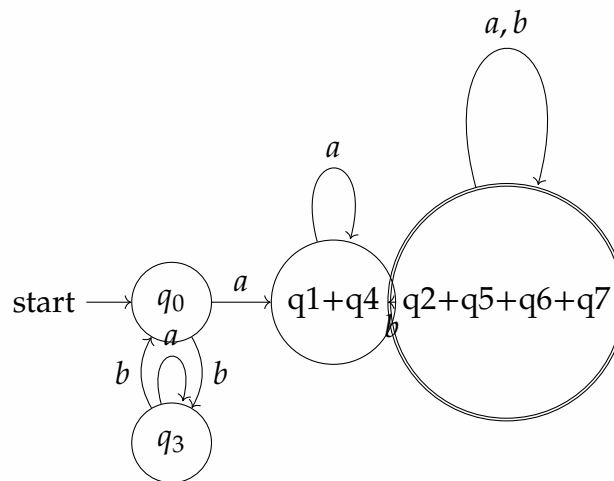
$$L = \{a^i b^j \mid i \in \mathbb{N}_0, j > 1\}.$$

Zeigen Sie, dass die Sprache L nicht vom Typ 3 der Chomsky-Hierarchie (d. h. nicht regulär) ist.

Aufgabe 3 [Minimierung von Endlichen Automaten]

Betrachten Sie den unten gezeigten deterministischen endlichen Automaten, der Worte über dem Alphabet $\Sigma = \{a, b\}$ verarbeitet. Bestimmen Sie den dazugehörigen Minimalautomaten, d. h. einen deterministischen endlichen Automaten, der die gleiche Sprache akzeptiert und eine minimale Anzahl an Zuständen benutzt. Erläutern Sie Ihre Berechnung, indem Sie z. B. eine Minimierungstabelle angeben.





Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Apkyuoo1g

Teilaufgabe Nr. 2

Aufgabe [Pseudo-Code Insertionsort, Bubblesort, Quicksort]

- (a) Geben Sie für folgende Sortierverfahren jeweils zwei Felder A und B an, so dass das jeweilige Sortierverfahren angewendet auf A seine Best-Case-Laufzeit und angewendet auf B seine Worst-Case-Laufzeit erreicht. (Wir messen die Laufzeit durch die Anzahl der Vergleiche zwischen Elementen der Eingabe.) Dabei soll das Feld A die Zahlen $1, 2, \dots, 7$ genau einmal enthalten; das Feld B ebenso. Sie bestimmen also nur die Reihenfolge der Zahlen.

Wenden Sie als Beleg für Ihre Aussagen das jeweilige Sortierverfahren auf die Felder A und B an und geben Sie nach jedem größeren Schritt des Algorithmus den Inhalt der Felder an.

Geben Sie außerdem für jedes Verfahren asymptotische Best- und Worst-Case-Laufzeit für ein Feld der Länge n an.

Für drei der Sortierverfahren ist der Pseudocode angegeben. Beachten Sie, dass die Feldindizes hier bei 1 beginnen. Die im Pseudocode verwendete Unteroutine $\text{Swap}(A, i, j)$ vertauscht im Feld A die Elemente mit den Indizes i und j miteinander.

- (i) Insertionsort
- (ii) Bubblesort
- (iii) Quicksort

Insertionsort(int[] A) for $i = 2$ to A.length do key = A[i] $i = i - 1$ while $i > 0$ and A[i] > key do A[i + 1] = A[i] $i = i - 1$ A[i + 1] = key

Bubblesort(int[] A) $n := \text{length}(A)$ repeat swapped = false for $i = n$ to 1 do if A[i] > A[i - 1] then Swap(A, i - 1, i)

```

swapped := true
until not swapped
Quicksort(int[] A, @ = 1, r = A.length) if 2 < r then m = Partition(A, 2, r) | Quicksort(A,
2, m - 1) Quicksort(A, m + 1, r)
int Partition (int[] A, int 2, intr)
pivot = A[r]
i =
for j = 2 to r - 1 do
if A[j] < pivot then
Swap(A, i, j) i = i + 1
Swap(A, i, r)
return i

```

- (b) Geben Sie die asymptotische Best- und Worst-Case-Laufzeit von Mergesort an.

Aufgabe 2 [Minimum und Maximum]

- (a) Argumentieren Sie, warum man das Maximum von n Zahlen nicht mit weniger als $n - 1$ Vergleichen bestimmen kann.

Lösungsvorschlag

Wenn die n Zahlen in einem unsortierten Zustand vorliegen, müssen wir alle Zahlen betrachten, um das Maximum zu finden. Wir brauchen dazu $n - 1$ und nicht n Vergleiche, da wir die erste Zahl zu Beginn des Algorithmus als Maximum definieren und anschließend die verbleibenden Zahlen $n - 1$ mit dem aktuellen Maximum vergleichen.

- (b) Geben Sie einen Algorithmus im Pseudocode an, der das Maximum eines Feldes der Länge n mit genau $n - 1$ Vergleichen bestimmt.

Lösungsvorschlag

```

public static int bestimmeMaximum(int[] a) {
    int max = a[0];
    for (int i = 1; i < a.length; i++) {
        if (a[i] > max) {
            max = a[i];
        }
    }
    return max;
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/MinimumMaximum.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/MinimumMaximum.java)

- (c) Wenn man das Minimum und das Maximum von n Zahlen bestimmen will, dann kann das natürlich mit $2n - 2$ Vergleichen erfolgen. Zeigen Sie, dass man bei jedem beliebigen Feld mit deutlich weniger Vergleichen auskommt, wenn man die beiden Werte statt in zwei separaten Durchläufen in einem Durchlauf geschickt bestimmt.

```

/**
 * Diese Methode ist nicht optimiert. Es werden  $2n - 2$  Vergleiche benötigt.
 *
 * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum gesucht
 *         werden soll.
 *
 * @return Ein Feld mit zwei Einträgen. Der erste Eintrag enthält das Minimum,
 *         der zweite Eintrag das Maximum.
 */
public static int[] minMaxNaiv(int[] a) {
    int max = a[0];
    int min = a[0];
    for (int i = 1; i < a.length; i++) {
        if (a[i] > max) {
            max = a[i];
        }
        if (a[i] < min) {
            min = a[i];
        }
    }
    return new int[] { min, max };
}

/**
 * Diese Methode ist optimiert. Es werden immer zwei Zahlen paarweise
 * betrachtet. Die Anzahl der Vergleiche reduziert sich auf  $3n/2 + 2$  bzw.
 *  $3(n-1)/2 + 4$  bei einer ungeraden Anzahl an Zahlen im Feld.
 *
 * nach <a href=
 * "https://www.techiedelight.com/find-minimum-maximum-element-array-using-
→ minimum-comparisons/">techiedelight.com</a>
 *
 * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum gesucht
 *         werden soll.
 *
 * @return Ein Feld mit zwei Einträgen. Der erste Eintrag enthält das Minimum,
 *         der zweite Eintrag das Maximum.
 */
public static int[] minMaxIterativPaarweise(int[] a) {
    int max = Integer.MIN_VALUE, min = Integer.MAX_VALUE;
    int n = a.length;

    boolean istUngerade = (n & 1) == 1;
    if (istUngerade) {
        n--;
    }

    for (int i = 0; i < n; i = i + 2) {
        int maximum, minimum;

```



```

        if (a[i] > a[i + 1]) {
            minimum = a[i + 1];
            maximum = a[i];
        } else {
            minimum = a[i];
            maximum = a[i + 1];
        }

        if (maximum > max) {
            max = maximum;
        }

        if (minimum < min) {
            min = minimum;
        }
    }

    if (istUngerade) {
        if (a[n] > max) {
            max = a[n];
        }

        if (a[n] < min) {
            min = a[n];
        }
    }

    return new int[] { min, max };
}

/**
 * Diese Methode ist nach dem Teile-und-Herrsche-Prinzip optimiert. Er
 * funktioniert so ähnlich wie der Mergesort.
 *
 * nach <a href=
 * "https://www.enjoyalgorithms.com/blog/find-the-minimum-and-maximum-value-in-
→ an-array">enjoyalgorithms.com</a>
 *
 * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum
 *          gesucht werden soll.
 * @param l Die linke Grenze.
 * @param r Die rechts Grenze.
 *
 * @return Ein Feld mit zwei Einträgen. Der erste Eintrag enthält das Minimum,
 *         der zweite Eintrag das Maximum.
 */
int[] minMaxRekursiv(int[] a, int l, int r) {
    int max, min;
    if (l == r) {
        max = a[l];
        min = a[l];
    } else if (l + 1 == r) {
        if (a[l] < a[r]) {
            max = a[r];

```

```

        min = a[l];
    } else {
        max = a[l];
        min = a[r];
    }
} else {
    int mid = l + (r - l) / 2;
    int[] lErgebnis = minMaxRekursiv(a, l, mid);
    int[] rErgebnis = minMaxRekursiv(a, mid + 1, r);
    if (lErgebnis[0] > rErgebnis[0]) {
        max = lErgebnis[0];
    } else {
        max = rErgebnis[0];
    }
    if (lErgebnis[1] < rErgebnis[1]) {
        min = lErgebnis[1];
    } else {
        min = rErgebnis[1];
    }
}
int[] ergebnis = { max, min };
return ergebnis;
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/MinimumMaximum.java](https://github.com/orgs/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/MinimumMaximum.java)

Aufgabe 3 [Schwach zusammenhängend gerichteter Graph]

Wir betrachten eine Variante der Breitensuche (BFS), bei der die Knoten markiert werden, wenn sie das erste Mal besucht werden. Außerdem wird die Suche einmal bei jedem unmarkierten Knoten gestartet, bis alle Knoten markiert sind. Wir betrachten gerichtete Graphen. Ein gerichteter Graph G ist *schwach zusammenhängend*, wenn der ungerichtete Graph (der sich daraus ergibt, dass man die Kantenrichtungen von G ignoriert) zusammenhängend ist.

Exkurs: Schwach zusammenhängend gerichteter Graph

Beim gerichteten Graphen musst du auf die Kantenrichtung achten. Würde man die Richtungen der Kanten ignorieren wäre aber trotzdem jeder Knoten erreichbar. Einen solchen Graphen nennt man schwach zusammenhängend.^a

Ein gerichteter Graph heißt (schwach) zusammenhängend, falls der zugehörige ungerichtete Graph (also der Graph, der entsteht, wenn man jede gerichtete Kante durch eine ungerichtete Kante ersetzt) zusammenhängend ist.^b

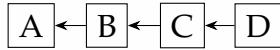
^a<https://studyflix.de/informatik/grundbegriffe-der-graphentheorie-1285>

^b[https://de.wikipedia.org/wiki/Zusammenhang_\(Graphentheorie\)](https://de.wikipedia.org/wiki/Zusammenhang_(Graphentheorie))

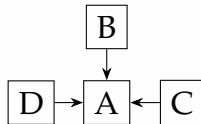
- (a) Beschreiben Sie für ein allgemeines $n \in \mathbb{N}$ mit $n \geq 2$ den Aufbau eines schwach zusammenhängenden Graphen G_n , mit n Knoten, bei dem die Breitensuche $\Theta(n)$ mal gestartet werden muss, bis alle Knoten markiert sind.

?

Die Breitensuche benötigt einen Startknoten. Die unten aufgeführten Graphen finden immer nur einen Knoten nämlich den Startknoten.



Oder so:



- (b) Welche asymptotische Laufzeit in Abhängigkeit von der Anzahl der Knoten (n) und von der Anzahl der Kanten (m) hat die Breitensuche über alle Neustarts zusammen? Beachten Sie, dass die Markierungen nicht gelöscht werden. Geben Sie die Laufzeit in Θ -Notation an. Begründen Sie Ihre Antwort.

Aufgabe 5 [Sondierfolgen für Hashing mit offener Adressierung]

Eine Sondierfolge $s(k, i)$ liefert für einen Schlüssel k aus einem Universum U und Versuchsnummern $i = 0, 1, \dots, m-1$ eine Folge von Indizes für eine Hashtabelle $T[0 \dots m-1]$. Mithilfe einer Sondierfolge wird beim Hashing mit offener Adressierung z. B. beim Einfügen eines neuen Schlüssels k nach einem noch nicht benützten Tabelleneintrag gesucht. Seien h und h' zwei verschiedene Hash-funktionen, die U auf $0, 1, \dots, m-1$ abbilden. Beantworten Sie die folgenden Fragen und geben Sie an, um welche Art von Sondieren es sich jeweils handelt.

- (a) Was ist problematisch an der Sondierfolge $s(k, i) = (h(k) + 2i) \bmod m$, wobei $m = 1023$ die Größe der Hashtabelle ist?

Lösungsvorschlag

Art Es handelt sich um lineares Sondieren.

Problematisch Es wird für einen großen Bereich an Sondierfolgen (512 (0-511, 512-1023)) nur in jeden zweiten Bucket (z. B. geradzahlig) sondiert, erst dann wird in den bisher ausgelassenen Buckets (z. B. ungeradzahlig) sondiert.

- (b) Was ist problematisch an der Sondierfolge $s(k, i) = (h(k) + i(i+1)) \bmod m$, wobei $m = 1024$ die Größe der Hashtabelle ist?

Lösungsvorschlag

Art Es handelt sich um quadratisches Sondieren

Problematisch $i(i+1)$ gibt immer eine gerade Zahl. Eine gerade Zahl Modulo 1024 gibt auch immer eine grade Zahl. Es wird nie in den ungeraden Buckets sondiert.

- (c) Was ist vorteilhaft an der Sondierfolge $s(k, i) = (h(k) + i \cdot h'(k)) \bmod m$, wobei m die Größe der Hashtabelle ist?

Auch die Sondierfolge ist abhängig von dem Schlüsselwert. Die Entstehung von Ballungen ist unwahrscheinlicher bei gut gewählten Hashfunktionen, eine gleichmäßige Verteilung wahrscheinlicher.

- (d) Sei $h(k) = k \bmod 6$ und $h(k) = k^2 \bmod 6$

Fügen Sie die Schlüssel 14, 9, 8, 3, 2 in eine Hashtabelle der Größe 7 ein. Verwenden Sie die Sondierfolge $s(k, i) = (h(k) + i \cdot h(k)) \bmod 7$ und offene Adressierung. Notieren Sie die Indizes der Tabellenfelder und vermerken Sie neben jedem Feld die erfolglosen Sondierungen.

Thema Nr. 2

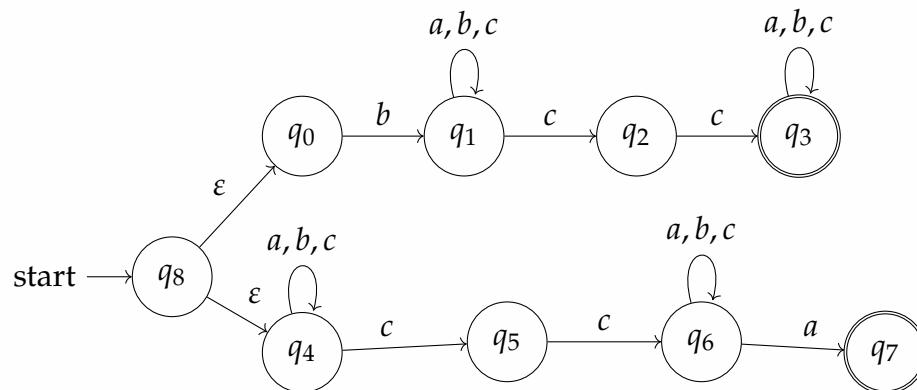
Teilaufgabe Nr. 1

Aufgabe 1 [Alphabet abc]

- (a) Betrachten Sie die formale Sprache $L \subseteq \{a, b, c\}^*$: aller Wörter, die entweder mit b beginnen oder mit a enden (aber nicht beides gleichzeitig) und das Teilwort cc enthalten. Entwerfen Sie einen (vollständigen) deterministischen endlichen Automaten, der die Sprache L akzeptiert. (Hinweis: Es werden weniger als 10 Zustände benötigt.)

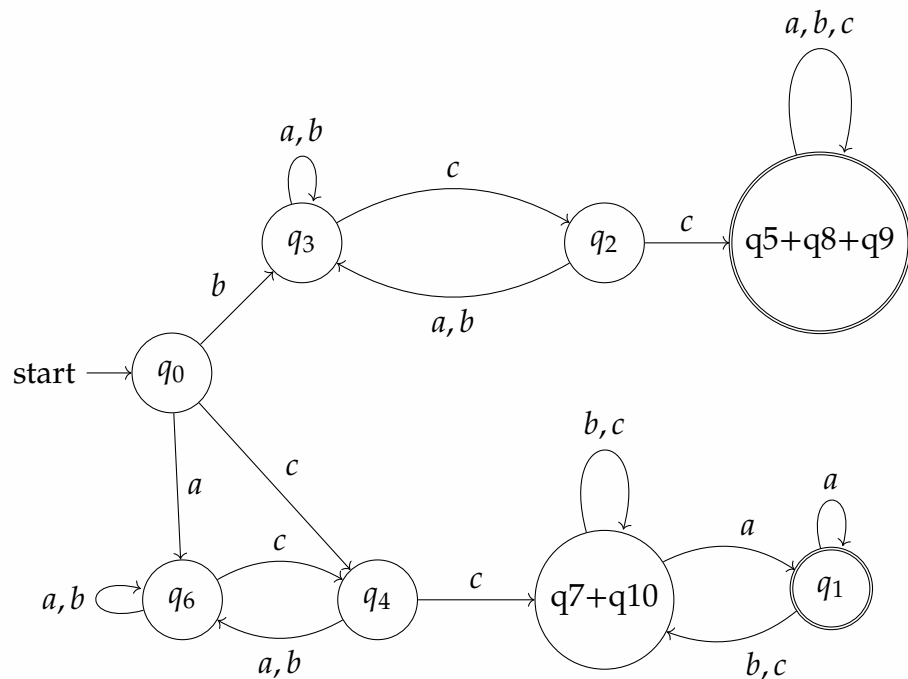
Lösungsvorschlag

NEA:



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ar3pvv7ha

konvertierter DEA:



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ai89m0txw

(b) Ist die folgende Aussage richtig? Begründen Sie Ihre Antwort.

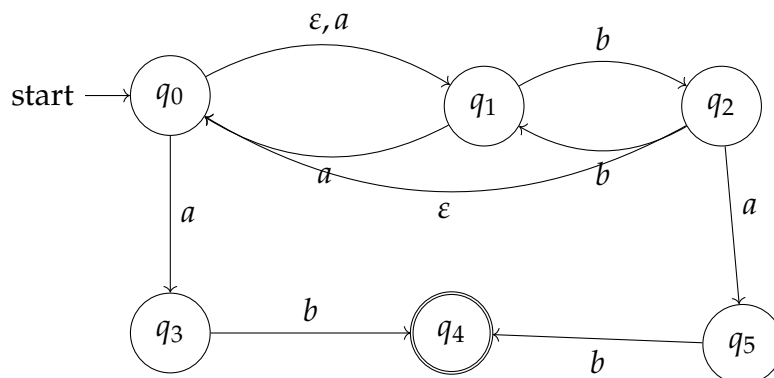
„Jede Teilsprache einer regulären Sprache ist regulär, d. h. für ein Alphabet und formale Sprachen $L' \subseteq L \subseteq \Sigma^*$ ist L' regulär, falls L regulär ist.“

Lösungsvorschlag

Ja. Reguläre Sprachen sind abgeschlossen unter dem Komplement und der Vereinigung.

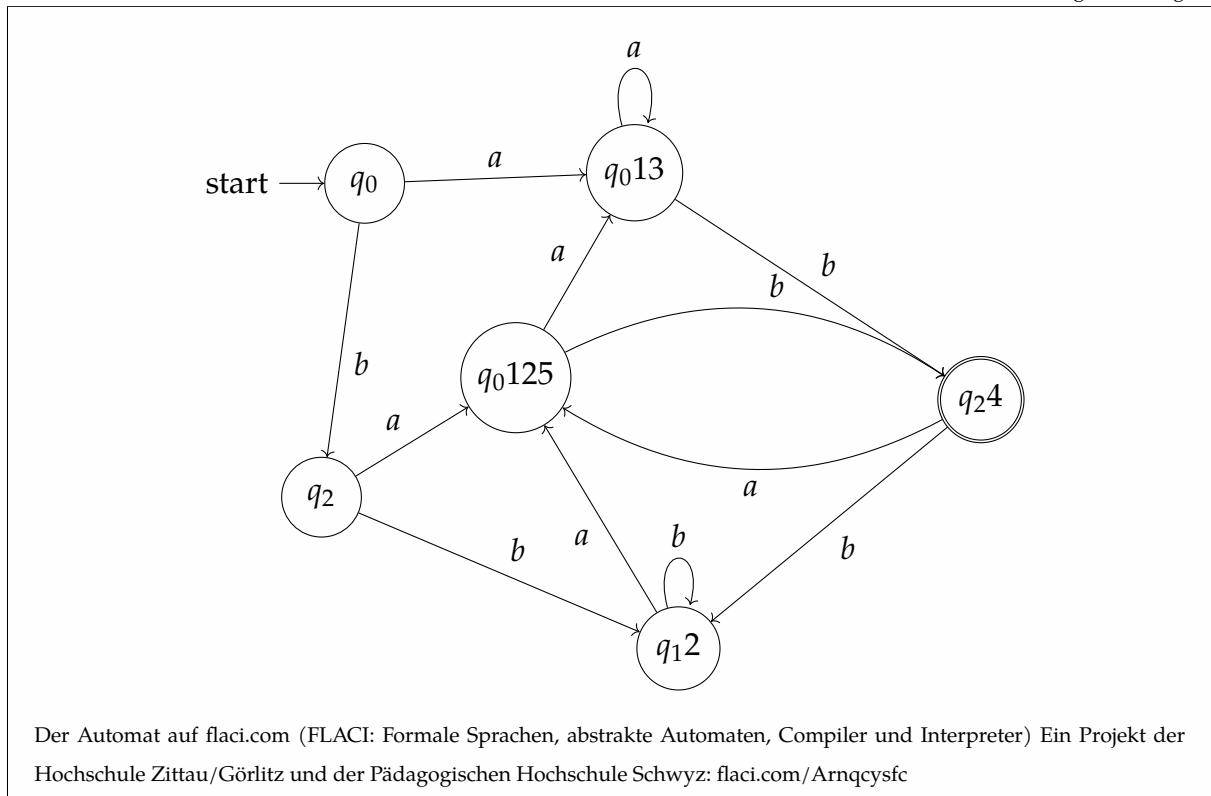
Aufgabe 2 [NEA ab]

Gegeben sei der folgende e-nichtdeterministische endliche Automat A über dem Alphabet



- (a) Konstruieren Sie einen deterministischen endlichen Automaten für A. Wenden Sie dafür die Potenzmengenkonstruktion an.

Lösungsvorschlag



- (b) Beschreiben Sie die von A akzeptierte Sprache $L(A)$ mit eigenen Worten und so einfach wie möglich.

Lösungsvorschlag

Endet auf ab, Präfix beliebig auch leer

Aufgabe 3 [L1, L2, L3 regulär oder kontextfrei]

Sei $\mathbb{N}_0 = \{0, 1, 2, \dots\}$ die Menge aller natürlichen Zahlen mit 0. Betrachten Sie die folgenden Sprachen.

- (a) $L_1 = \{a^{3n}b^{2n}a^n \mid n \in \mathbb{N}_0\}$

Lösungsvorschlag

nicht kontextfrei

- (b) $L_2 = \{a^{3n}a^{2n}b^n \mid n \in \mathbb{N}_0\}$

kontextfrei.

Der Ausdruck lässt umformen in: $L_2 = \{ a^{5^n} b^n \mid n \in \mathbb{N}_0 \}$

$P = \{$

$$S \rightarrow aaaaaSb \mid \varepsilon$$

$\}$

(c) $L_3 = \{ (ab)^n a (ba)^n b (ab)^n \mid n \in \mathbb{N}_0 \}$

nicht kontextfrei

Geben Sie jeweils an, ob L_1 , L_2 und L_3 kontextfrei und ob L_1 , L_2 und L_3 regulär sind. Beweisen Sie Ihre Behauptung und ordnen Sie jede Sprache in die kleinstmögliche Klasse (regulär, kontextfrei, nicht kontextfrei) ein. Für eine Einordnung in kontextfrei zeigen Sie also, dass die Sprache kontextfrei und nicht regulär ist.

Erfolgt ein Beweis durch Angabe eines Automaten, so ist eine klare Beschreibung der Funktionsweise des Automaten und der Bedeutung der Zustände erforderlich. Erfolgt der Beweis durch Angabe eines regulären Ausdruckes, so ist eine intuitive Beschreibung erforderlich. Wird der Beweis durch die Angabe einer Grammatik geführt, so ist die Bedeutung der Variablen zu erläutern.

Teilaufgabe Nr. 2

Aufgabe 1 [O-Notation $a()$, $b()$, $c()$, $d()$, $e(n)$]

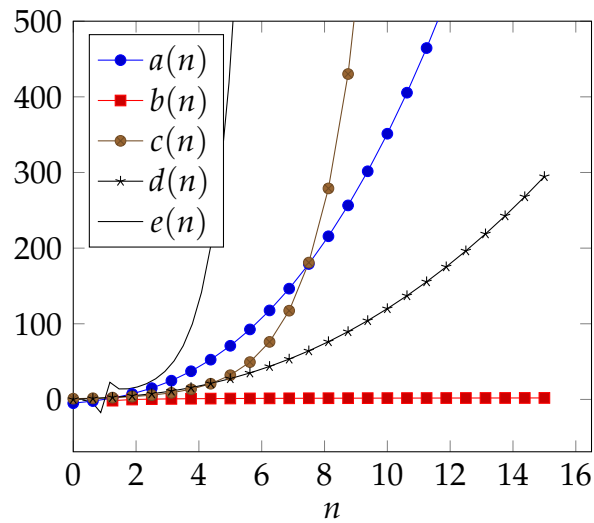
Sortieren Sie die unten angegebenen Funktionen der O-Klassen $\mathcal{O}(a)$, $\mathcal{O}(b)$, $\mathcal{O}(c)$, $\mathcal{O}(d)$ und $\mathcal{O}(e)$ bezüglich ihrer Teilmengenbeziehungen. Nutzen Sie ausschließlich die echte Teilmenge \subsetneq sowie die Gleichheit $=$ für die Beziehung zwischen den Mengen. Folgendes Beispiel illustriert diese Schreibweise für einige Funktionen f_1 bis f_5 . (Diese haben nichts mit den unten angegebenen Funktionen zu tun.)

$$\mathcal{O}(f_4) \subsetneq \mathcal{O}(f_3) = \mathcal{O}(f_5) \subsetneq \mathcal{O}(f_1) = \mathcal{O}(f_2)$$

Die angegebenen Beziehungen müssen weder bewiesen noch begründet werden.

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

$$- e(n) = \frac{4^n}{\log_2 n}$$



Aufgabe 2 [Java-Klasse Stack]

Gegeben sei die folgende Java-Implementierung eines Stacks.

```
class Stack {
    private Item head;

    public Stack() {
        head = null;
    }

    public void push(int val) {
        if (head == null) {
            head = new Item(val, null);
        } else {
            head = new Item(val, head);
        }
    }

    public int pop() {
        // ...
    }

    public int size() {
        // ...
    }

    public int min() {
        // ...
    }
}

class Item {
    private int val;
    private Item next;
}
```

```

public Item(int val, Item next) {
    this.val = val;
    this.next = next;
}
}
}

```

- (a) Implementieren Sie die Methode `pop` in einer objektorientierten Programmiersprache Ihrer Wahl, die das erste Item des Stacks entfernt und seinen Wert zurückgibt. Ist kein Wert im Stack enthalten, so soll dies mit einer `IndexOutOfBoundsException` oder Ähnlichem gemeldet werden.

Beschreiben Sie nun jeweils die notwendigen Änderungen an den bisherigen Implementierungen, die für die Realisierung der folgenden Methoden notwendig sind.

Lösungsvorschlag

```

public int pop() {
    if (head != null) {
        int val = head.val;
        size--;
        head = head.next;
        return val;
    } else {
        throw new IndexOutOfBoundsException("The stack is empty");
    }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/Stack.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/Stack.java)

- (b) `size` gibt in Laufzeit $\mathcal{O}(1)$ die Anzahl der enthaltenen Items zurück.

Lösungsvorschlag

```

public void push(int val) {
    if (head == null) {
        head = new Item(val, null);
    } else {
        head = new Item(val, head);
    }
    if (min > val) {
        min = val;
    }
    size++;
}

public int pop() {
    if (head != null) {
        int val = head.val;
        size--;
        head = head.next;
        return val;
    } else {
        throw new IndexOutOfBoundsException("The stack is empty");
    }
}

```

```
public int size() {
    return size;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/Stack.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/Stack.java)

- (c) `min` gibt (zu jedem Zeitpunkt) in Laufzeit $\mathcal{O}(1)$ den Wert des kleinsten Elements im Stack zurück.

Lösungsvorschlag

```
public void push(int val) {
    if (head == null) {
        head = new Item(val, null);
    } else {
        head = new Item(val, head);
    }
    if (min > val) {
        min = val;
    }
    size++;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/Stack.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/Stack.java)

```
public int min() {
    return min;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/Stack.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/Stack.java)

Sie dürfen jeweils alle anderen angegebenen Methoden der Klasse verwenden, auch wenn Sie diese nicht implementiert haben. Sie können anstelle von objektorientiertem Quellcode auch eine informelle Beschreibung Ihrer Änderungen angeben.

Aufgabe 3 [Lineare und Binäre Suchverfahren]

Gegeben ist ein aufsteigend sortiertes Array A von n ganzen Zahlen und eine ganze Zahl x . Es wird der Algorithmus `BinarySearch` betrachtet, der A effizient nach dem Wert x absucht. Ergebnis ist der Index i mit $x = A[i]$ oder `NIL`, falls $x \notin A$.

- (a) Durchsuchen Sie das folgende Feld jeweils nach den in (i) bis (iii) angegebenen Werten mittels binärer Suche. Geben Sie für jede Iteration die Werte l, r, m und den betretenen `if`-Zweig an. Geben Sie zudem den Ergebnis-Index bzw. `NIL` an.

Index

`i[s] <= 2] 4] off`

wen `[ilsfol7] io] w]u]al ale!`

- (i) 10

Funktion BinarySearch(int A, int r)

```
l = 1;  
r = A.length;  
while r ≥ l do  
    if x < A[m] then  
        r = m - 1;  
    else if x = A[m] then  
        return m;  
    else  
        l = m + 1;  
    end  
    return NIL;  
end
```

(ii) 13

(iii) 22

- (b) Betrachten Sie auf das Array aus Teilaufgabe a). Für welche Werte durchläuft der Algorithmus nie den letzten else-Teil in Zeile 11? Hinweis: Unterscheiden Sie auch zwischen enthaltenen und nicht-enthaltenen Werten.
- (c) Wie ändert sich das Ergebnis der binären Suche, wenn im sortierten Eingabefeld zwei aufeinanderfolgende, unterschiedliche Werte vertauscht wurden? Betrachten Sie hierbei die betroffenen Werte, die anderen Feldelemente und nicht enthaltene Werte in Abhängigkeit vom Ort der Vertauschung.
- (d) Angenommen, das Eingabearray A für den Algorithmus für die binäre Suche enthält nur die Zahlen 0 und 1, aufsteigend sortiert. Zudem ist jede der beiden Zahlen mindestens ein Mal vorhanden. Ändern Sie den Algorithmus für die binäre Suche so ab, dass er den bzw. einen Index k zurückgibt, für den gilt: $A[k] = 1$ und $A[k-1] = 0$.
- (e) Betrachten Sie die folgende rekursive Variante von BinarySearch.

1 int RekBinarySearch(int[] A, int x, int l, int r)

2 | mi

3 | (rekursive Implementierung)

Der initiale Aufruf der rekursiven Variante lautet: RekBinarySearch (A, x, 1, A.length)

Aufgabe 4 [Graph a-i]

- (a) Berechnen Sie im gegebenen gerichteten und gewichteten Graph $G = (V, E, w)$ mit Kantenlängen $w : E \rightarrow \mathbb{R}_0^+$ mittels des Dijkstra-Algorithmus die kürzesten (gerichteten) Pfade ausgehend vom Startknoten a .

nach	Entfernung	Reihenfolge	Pfad
$a \rightarrow a$	0	1	
$a \rightarrow b$	7	7	$a \rightarrow f \rightarrow b$
$a \rightarrow c$	5	6	$a \rightarrow h \rightarrow c$
$a \rightarrow d$	11	8	$a \rightarrow h \rightarrow c \rightarrow d$
$a \rightarrow e$	3	4	$a \rightarrow f \rightarrow e$
$a \rightarrow f$	1	2	$a \rightarrow f$
$a \rightarrow g$	4	5	$a \rightarrow f \rightarrow g$
$a \rightarrow h$	2	3	$a \rightarrow h$
$a \rightarrow i$	2.147483647E9	9	$a \rightarrow i$

- (ii) Zeichnen Sie zudem den entstandenen Kürzeste-Pfade-Baum.
- (b) Warum berechnet der Dijkstra-Algorithmus auf einem gerichteten Eingabegraphen mit potentiell auch negativen Kantengewichten $w : E \rightarrow \mathbb{R}$ nicht immer einen korrekten Kürzesten-Wege-Baum von einem gewählten Startknoten aus? Geben Sie ein Beispiel an, für das der Algorithmus die falsche Antwort liefert.
- (c) Begründen Sie, warum das Problem nicht gelöst werden kann, indem der Betrag des niedrigsten (also des betragsmäßig größten negativen) Kantengewichts im Graphen zu allen Kanten addiert wird.