

# **Erste Staatsprüfung für ein Lehramt an öffentlichen Schulen**

**46115  
2021  
03**

# Inhaltsverzeichnis

Thema Nr. 1	3
Teilaufgabe Nr. 1	3
Aufgabe 1 [Kaugummi-Automat]	3
Aufgabe 2 [Nonterminale SABCD, Terminale ab]	3
Aufgabe 3 [Minimierung von Endlichen Automaten]	3
Aufgabe 4 [Turingmaschine M]	4
Teilaufgabe Nr. 2	4
Aufgabe [Pseudo-Code Insertionsort, Bubblesort, Quicksort]	4
Aufgabe 2 [Minimum und Maximum]	5
Aufgabe 3 [Schwach zusammenhängend gerichteter Graph]	7
Aufgabe 5 [Sondierfolgen für Hashing mit offener Adressierung]	8
Thema Nr. 2	9
Teilaufgabe Nr. 1	9
Aufgabe 1 [Alphabet abc]	9
Aufgabe 2 [NEA ab]	10
Aufgabe 3 [L1, L2, L3 reguläre oder kontextfrei]	11
Teilaufgabe Nr. 2	11
Aufgabe 1 [O-Notation a(), b(), c(), d(), e(n)]	11
Aufgabe 2 [Java-Klasse Stack]	12
Aufgabe 3 [Lineare und Binäre Suchverfahren]	15
Aufgabe 4 [Graph a-i]	15

# Thema Nr. 1

## Teilaufgabe Nr. 1

### Aufgabe 1 [Kaugummi-Automat]

Ein moderner Kaugummi-Automat erhalte 20- und 50-Cent-Münzen. Eine Packung Kaugummi kostet 120 Cent. Die interne Steuerung des Kaugummi-Automaten verwendet einen deterministischen endlichen Automaten, der die Eingabe als Folge von 20- und 50-Cent-Münzen (d. h. als Wort über dem Alphabet 20,50) erhält, und genau die Folgen akzeptiert, die in der Summe 120 Cent ergeben.

- Geben Sie zwei Worte aus  $20,50^*$  an, die der Automat akzeptiert.
- Zeichnen Sie einen deterministischen endlichen Automaten als Zustandsgraph, der für die interne Steuerung des Kaugummi-Automaten verwendet werden kann (d. h. der Automat akzeptiert genau alle Folgen von 20- und 50-Cent-Münzen, deren Summe 120 ergibt).
- Geben Sie einen regulären Ausdruck an, der die akzeptierte Sprache des Automaten erzeugt.

### Aufgabe 2 [Nonterminale SABCD, Terminale ab]

- 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.

- 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  $\epsilon$  das leere Wort ist).

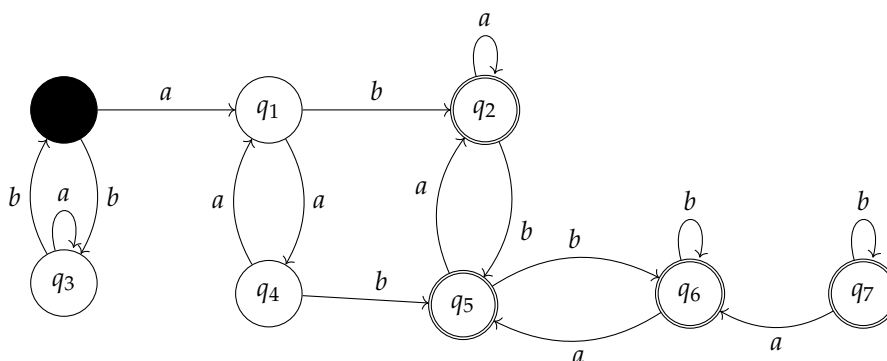
Sei die Sprache  $L$  definiert über dem Alphabet  $\{a, b\}$  als

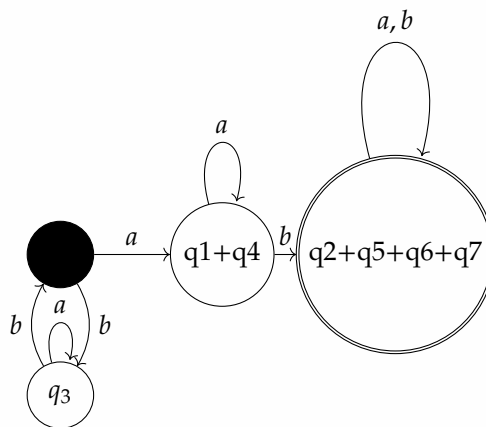
$$L = \{a^i b^j \mid i \in \mathbb{N}_0, j > 1\}.$$

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

#### Aufgabe 4 [Turingmaschine M]

Beweisen Sie, dass das folgende Entscheidungsproblem semi-entscheidbar ist.

Eingabe: Eine Turingmaschine M Aufgabe: Entscheiden Sie, ob ein Eingabewort existiert, auf das M hält.

#### 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,...,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 := length(A) repeat swapped = false for  $i = 1$  to n - 1 do if A[i] > A[i + 1] then Swap(A, i, i + 1)

swapped := true

until not swapped

Quicksort(int[] A, l, r) if  $l < r$  then m = Partition(A, l, r) | Quicksort(A, l, m - 1) Quicksort(A, m + 1, r)

int Partition (int[] A, int l, int r)

pivot = A[r]

```

i=
for j = 0 to r-1 do
  if A[j] < pivot then
    Swap(A, j, r)
    w=i+1
    Swap(A, j, 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.

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.

```

5  public static int bestimmeMaximum(int[] a) {
6      int max = a[0];
7      for (int i = 1; i < a.length; i++) {
8          if (a[i] > max) {
9              max = a[i];
10         }
11     }
12     return max;
13 }

```

Code-Beispiel auf Github ansehen: [src/main/java/org/beschlangaul/examen/examen\\_66115/jahr\\_2021/fruehjahr/MinimumMaximum.java](https://github.com/beschlangaul/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.

```

15  /**
16   * Diese Methode ist nicht optimiert. Es werden  $2n - 2$  Vergleiche benötigt.
17   *
18   * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum gesucht
19   *         werden soll.
20   *
21   * @return Ein Feld mit zwei Einträgen. Der erste Eintrag enthält das Minimum,
22   *         der zweite Eintrag das Maximum.
23   */
24  public static int[] minMaxNaiv(int[] a) {
25      int max = a[0];
26      int min = a[0];
27      for (int i = 1; i < a.length; i++) {
28          if (a[i] > max) {
29              max = a[i];
30          }
31          if (a[i] < min) {
32              min = a[i];
33          }
34      }
35      return new int[] { min, max };
36  }
37

```

```

38  /**
39   * Diese Methode ist optimiert. Es werden immer zwei Zahlen paarweise
40   * betrachtet. Die Anzahl der Vergleiche reduziert sich auf  $3n/2 + 2$  bzw.
41   *  $3(n-1)/2 + 4$  bei einer ungeraden Anzahl an Zahlen im Feld.
42   *
43   * nach <a href=
44   * "https://www.techiedelight.com/find-minimum-maximum-element-array-using-minimum-
→ comparisons/">techiedelight.com</a>
45   *
46   * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum gesucht
47   *         werden soll.
48   *
49   * @return Ein Feld mit zwei Einträgen. Der erste Eintrag enthält das Minimum,
50   *         der zweite Eintrag das Maximum.
51   */
52  public static int[] minMaxIterativPaarweise(int[] a) {
53      int max = Integer.MIN_VALUE, min = Integer.MAX_VALUE;
54      int n = a.length;
55
56      boolean istUngerade = (n & 1) == 1;
57      if (istUngerade) {
58          n--;
59      }
60
61      for (int i = 0; i < n; i = i + 2) {
62          int maximum, minimum;
63
64          if (a[i] > a[i + 1]) {
65              minimum = a[i + 1];
66              maximum = a[i];
67          } else {
68              minimum = a[i];
69              maximum = a[i + 1];
70          }
71
72          if (maximum > max) {
73              max = maximum;
74          }
75
76          if (minimum < min) {
77              min = minimum;
78          }
79      }
80
81      if (istUngerade) {
82          if (a[n] > max) {
83              max = a[n];
84          }
85
86          if (a[n] < min) {
87              min = a[n];
88          }
89      }
90      return new int[] { min, max };
91  }
92
93  /**
94   * Diese Methode ist nach dem Teile-und-Herrsche-Prinzip optimiert. Er
95   * funktioniert so ähnlich wie der Mergesort.
96   *
97   * nach <a href=
98   * "https://www.enjoyalgorithms.com/blog/find-the-minimum-and-maximum-value-in-an-
→ array">enjoyalgorithms.com</a>
99   *
100   * @param a Ein Feld mit Zahlen, in dem nach dem Minimum und dem Maximum
101   *          gesucht werden soll.
102   * @param l Die linke Grenze.
103   * @param r Die rechts Grenze.
104   *

```

```

105  * @return Ein Feld mit zwei Einträgen. Der erste Eintrag enthält das Minimum,
106  *         der zweite Eintrag das Maximum.
107  */
108  int[] minMaxRekursiv(int[] a, int l, int r) {
109      int max, min;
110      if (l == r) {
111          max = a[l];
112          min = a[l];
113      } else if (l + 1 == r) {
114          if (a[l] < a[r]) {
115              max = a[r];
116              min = a[l];
117          } else {
118              max = a[l];
119              min = a[r];
120          }
121      } else {
122          int mid = l + (r - l) / 2;
123          int[] lErgebnis = minMaxRekursiv(a, l, mid);
124          int[] rErgebnis = minMaxRekursiv(a, mid + 1, r);
125          if (lErgebnis[0] > rErgebnis[0]) {
126              max = lErgebnis[0];
127          } else {
128              max = rErgebnis[0];
129          }
130          if (lErgebnis[1] < rErgebnis[1]) {
131              min = lErgebnis[1];
132          } else {
133              min = rErgebnis[1];
134          }
135      }
136      int[] ergebnis = { max, min };
137      return ergebnis;
138  }
139
140  }

```

Code-Beispiel auf Github ansehen: [src/main/java/org/beschlangaul/examen/examen\\_66115/jahr\\_2021/fruehjahr/MinimumMaximum.java](https://github.com/src/main/java/org/beschlangaul/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.<sup>a</sup>

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.<sup>b</sup>

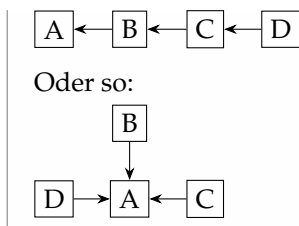
<sup>a</sup><https://studyflix.de/informatik/grundbegriffe-der-graphentheorie-1285>

<sup>b</sup>[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.



- (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  $\Theta$ -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 benutzten Tabelleneintrag gesucht. Seien  $h$  und  $h'$  zwei verschiedene Hashfunktionen, 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?

**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?

**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.



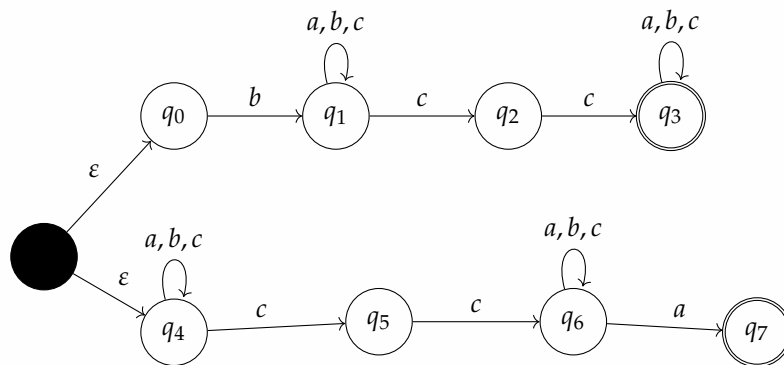
# Thema Nr. 2

## Teilaufgabe Nr. 1

### Aufgabe 1 [Alphabet $abc$ ]

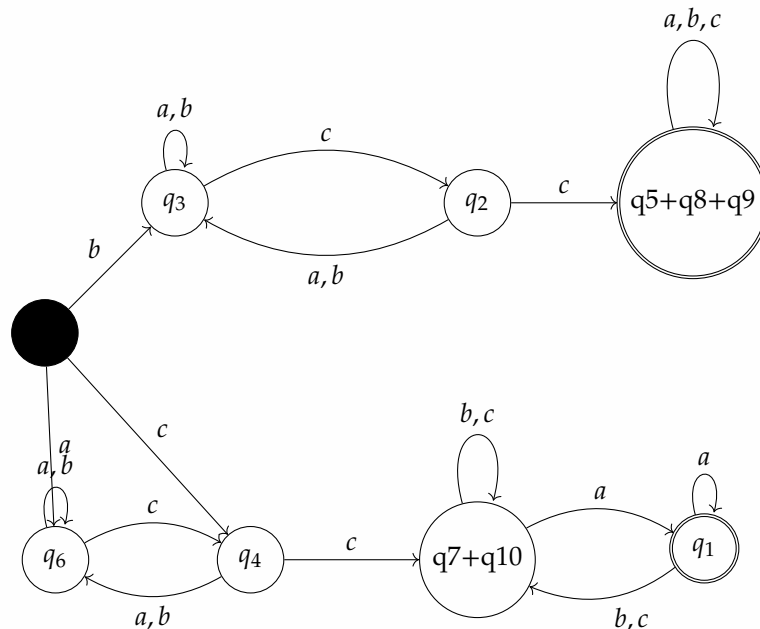
- (a) Betrachten Sie die formale Sprache  $L \subseteq a,b,c^*$ : aller Wörter, die entweder mit  $b$  beginnen oder mit  $a$  enden (aber nicht beides gleichzeitig) und das Teilwort  $cc$  enthalten. Entwerfen Sie einen (vollständigen) deterministischen endlichen Automaten, der die Sprache  $L$  akzeptiert.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

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](http://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](http://flaci.com/Ai89m0txw)

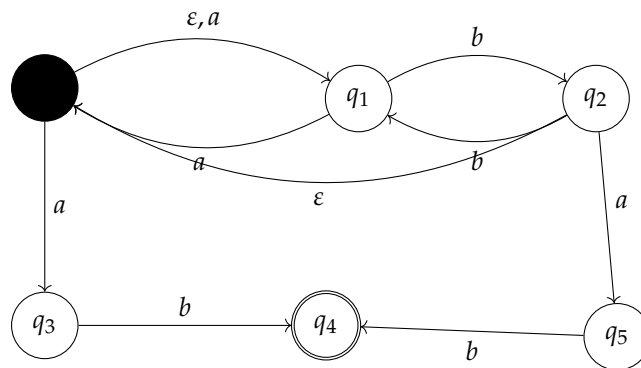
- (b) Ist die folgende Aussage richtig? Begründen Sie Ihre Antwort.

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

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

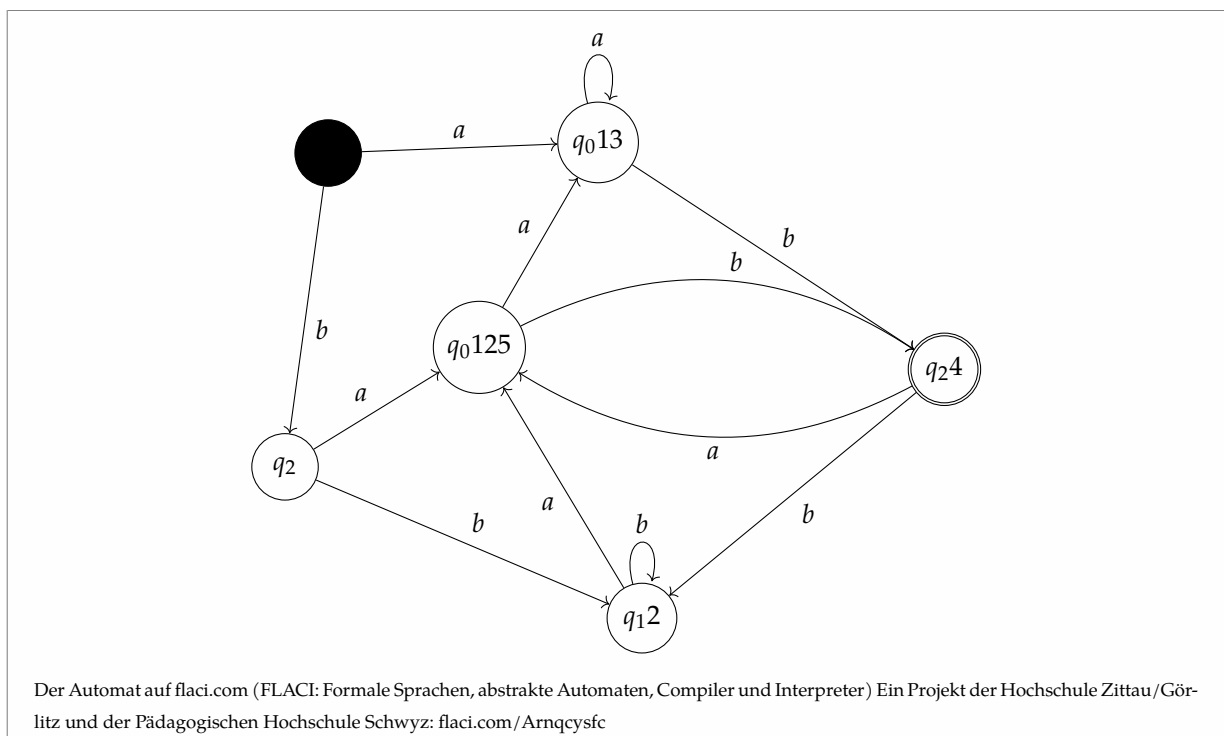
## Aufgabe 2 [NEA ab]

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



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/A54bbh2iz

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



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

Endet auf ab, Präfix beliebig auch leer

### Aufgabe 3 [L1, L2, L3 reguläre oder kontextfrei]

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

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

$$P = \left\{ \begin{array}{l} S \rightarrow a^n b^n \mid n \in \mathbb{N}_0 \\ B \rightarrow \end{array} \right\}$$

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

$$P = \left\{ \begin{array}{l} a^{5n} b^n \\ S \rightarrow a^n b^{2n} \mid n \in \mathbb{N}_0 \end{array} \right\}$$

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

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 nicht 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 Ausdrucks, 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  $O(a)$ ,  $O(b)$ ,  $O(c)$ ,  $O(d)$  und  $O(e)$  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  $f_5$ . (Diese haben nichts mit den unten angegebenen Funktionen zu tun.)

$O(f_1) \subset O(f_2) = O(f_3)$   $O(f_4) = O(f_5)$  Die angegebenen Beziehungen müssen weder bewiesen noch begründet werden.

$$- a(n) = \sqrt{n^5} + 4n - 5$$

$$- b(n) = \log(\log(n))$$

$$- c(n) = 2^n$$

$$- d(n) = n \log(n) + 2n$$

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

## Aufgabe 2 [Java-Klasse Stack]

Gegeben sei die folgende Java-Implementierung eines Stacks.

```
1 class Stack {
2     private Item head;
3
4     public Stack() {
5         head = null;
6     }
7
8     public void push(int val) {
9         if (head == null) {
10             head = new Item(val, null);
11         } else {
12             head = new Item(val, head);
13         }
14     }
15
16     public int pop() {
17         // ...
18     }
19
20     public int size() {
21         // ...
22     }
23
24     public int min() {
25         // ...
26     }
27
28     class Item {
29         private int val;
30         private Item next;
31
32         public Item(int val, Item next) {
33             this.val = val;
34             this.next = next;
35         }
36     }
37 }
```

- (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.

```
25     public int pop() {
26         if (head != null) {
27             int val = head.val;
28             size--;
29             head = head.next;
30             return val;
31         } else {
32             throw new IndexOutOfBoundsException("The stack is empty");
33         }
34     }
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46115/jahr\\_2021/fruehjahr/Stack.java](https://github.com/orgs/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/Stack.java)

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

```
13     public void push(int val) {
14         if (head == null) {
15             head = new Item(val, null);
16         } else {
17             head = new Item(val, head);
18         }
19     }
```

```

18     }
19     if (min > val) {
20         min = val;
21     }
22     size++;
23 }
24
25 public int pop() {
26     if (head != null) {
27         int val = head.val;
28         size--;
29         head = head.next;
30         return val;
31     } else {
32         throw new IndexOutOfBoundsException("The stack is empty");
33     }
34 }
35
36 public int size() {
37     return size;
38 }

```

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  $O(1)$  den Wert des kleinsten Elements im Stack zurück.

```

13 public void push(int val) {
14     if (head == null) {
15         head = new Item(val, null);
16     } else {
17         head = new Item(val, head);
18     }
19     if (min > val) {
20         min = val;
21     }
22     size++;
23 }

```

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)

```

40 public int min() {
41     return min;
42 }
43

```

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.

### Additum: Kompletter Java-Code

```

3 class Stack {
4     private Item head;
5
6     private int size;
7     private int min = Integer.MAX_VALUE;
8
9     public Stack() {
10         head = null;
11     }
12
13     public void push(int val) {
14         if (head == null) {

```

```

15     head = new Item(val, null);
16 } else {
17     head = new Item(val, head);
18 }
19 if (min > val) {
20     min = val;
21 }
22 size++;
23 }
24
25 public int pop() {
26     if (head != null) {
27         int val = head.val;
28         size--;
29         head = head.next;
30         return val;
31     } else {
32         throw new IndexOutOfBoundsException("The stack is empty");
33     }
34 }
35
36 public int size() {
37     return size;
38 }
39
40 public int min() {
41     return min;
42 }
43
44 class Item {
45     private int val;
46     private Item next;
47
48     public Item(int val, Item next) {
49         this.val = val;
50         this.next = next;
51     }
52 }
53 }

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46115/jahr\\_2021/fruehjahr/Stack.java](https://github.com/orgs/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/Stack.java)

```

3 import static org.junit.Assert.assertEquals;
4 import static org.junit.Assert.assertThrows;
5
6 import org.junit.Test;
7
8 public class StackTest {
9
10     private Stack makeStack() {
11         Stack stack = new Stack();
12         stack.push(1);
13         stack.push(2);
14         stack.push(3);
15         return stack;
16     }
17
18     @Test
19     public void methodPop() {
20         Stack stack = makeStack();
21         assertEquals(3, stack.pop());
22         assertEquals(stack.size(), 2);
23         assertEquals(2, stack.pop());
24         assertEquals(stack.size(), 1);
25         assertEquals(1, stack.pop());
26         assertEquals(stack.size(), 0);
27         assertThrows(IndexOutOfBoundsException.class, () -> {
28             stack.pop();
29         });

```

```

30 }
31
32 @Test
33 public void methodSize() {
34     Stack stack = makeStack();
35     assertEquals(stack.size(), 3);
36 }
37
38 @Test
39 public void methodMin() {
40     Stack stack = makeStack();
41     assertEquals(stack.min(), 1);
42 }
43 }

```

Code-Beispiel auf Github ansehen: [src/test/java/org/bschlangaul/examen/examen\\_46115/jahr\\_2021/fruehjahr/StackTest.java](https://github.com/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/StackTest.java)

### Aufgabe 3 [Lineare und Binäre Suchverfahren]

Gegeben ist ein aufsteigend sortiertes Array  $A$  von  $n$  ganzen Zahlen und eine ganze Zahl  $r$ . 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  $x \notin A$ .

```

1 int BinarySearch(int[] A, int r)
2 f=1 3 r= A.length 4 while r > f do
5 m = (f+r)/2 6 if 2 < A[m] then
7 r=m 8 else if 2 = A[m] then
9 return m 10 else
11 f=m+1 12 return NIL

```

Durchsuchen Sie das folgende Feld jeweils nach den in (i) bis (iii) angegebenen Werten mittels binärer Suche. Geben Sie für jede Iteration die Werte  $f, r, m$  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!

10

13

22

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.

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.

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  $A[k-1] = 0$ .

Betrachten Sie die folgende rekursive Variante von `BinarySearch`.

```

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

```

```

| 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  $w : E \rightarrow \mathbb{R}$  mittels des Dijkstra-Algorithmus die kürzesten (gerichteten) Pfade ausgehend vom Startknoten  $a$ .

Knoten, deren Entfernung von  $a$  bereits feststeht, seien als schwarz bezeichnet und Knoten, bei denen lediglich eine obere Schranke  $\infty$  für ihre Entfernung von  $a$  bekannt ist, seien als grau bezeichnet.

- (b) Geben Sie als Lösung eine Tabelle an. Fügen Sie jedes mal, wenn der Algorithmus einen Knoten schwarz färbt, eine Zeile zu der Tabelle hinzu. Die Tabelle soll dabei zwei Spalten beinhalten: die linke Spalte zur Angabe des aktuell schwarz gewordenen Knotens und die rechte Spalte mit der bereits aktualisierten Menge grauer Knoten. Jeder Tabelleneintrag soll anstelle des nackten Knotennamens  $v$  ein Tripel  $(v, v.d, v.r)$  sein. Dabei steht  $v.d$  für die aktuell bekannte kürzeste Distanz zwischen  $a$  und  $v$ .  $v.r$  ist der direkte Vorgänger von  $v$  auf dem zugehörigen kürzesten Weg von  $a$ .
- (c) Zeichnen Sie zudem den entstandenen Kürzeste-Pfade-Baum.
- (d) Warum berechnet der Dijkstra-Algorithmus auf einem gerichteten Eingabegraphen mit potentiell auch negativen Kantengewichten  $w : \# \rightarrow \mathbb{R}$  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.
- (e) 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.