Thema Nr. 1

Teilaufgabe Nr. 1

Staatsexamen 46115 / 2021 / Frühjahr / Thema Nr. 1 / Teilaufgabe Nr. 1 / Aufgabe 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.

(a) Geben Sie zwei Worte aus 20,50* an, die der Automat akzeptiert.

(b) 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).

(c) Geben Sie einen regulären Ausdruck an, der die akzeptierte Sprache des Automaten erzeugt.

Staatsexamen 46115 / 2021 / Frühjahr / Thema Nr. 1 / Teilaufgabe Nr. 1 / Aufgabe Nr. 2

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, 5) mit Variablen V=s,A,B,C,D, Terminalzeichen 3 = a,b, Produktionen

P= S -> SB | AC |a, A -> a, B -> b, C -> DD | AB, D -> AB | DC | CD

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

(b) Mit a', wobei i € No = 0,1,2,..., wird das Wort bezeichnet, das aus der i-fachen Wie- derholung des Zeichens a besteht (d. h. a' = € und a? = aa' !, wobei e das leere Wort ist).

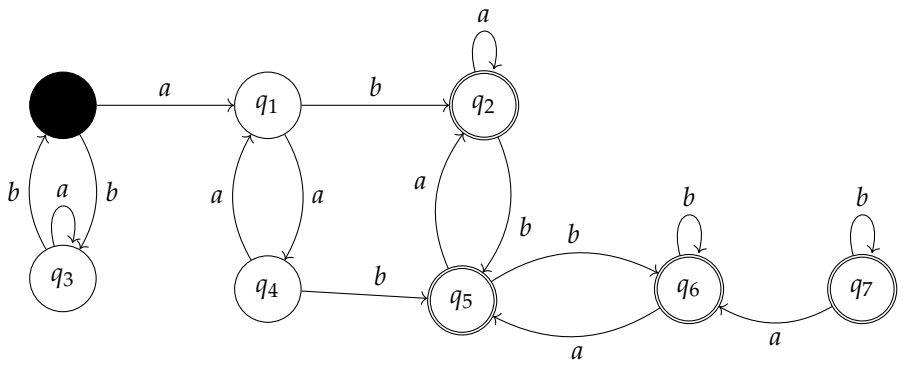
Sei die Sprache L definiert über dem Alphabet a,b als

L = a"b™ | iE No, i > 1.

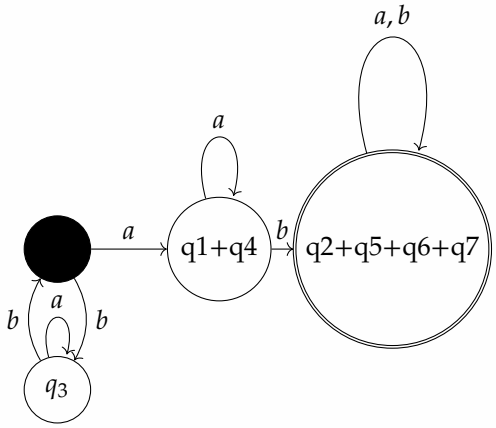
Zeigen Sie, dass die Sprache Z nicht vom Typ 3 der Chomsky-Hierarchie (d. h. nicht regulär) ist.

Aufgabe 3

Betrachten Sie den unten gezeigten deterministischen endlichen Automaten, der Worte über dem Alphabet X = a,b verarbeitet. Bestimmen Sie den dazugehörigen Minimalautomaten, d. h. einen deterministischen endlichen Automaten, der die gleiche Sprache akzeptiert und eine mini- male 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/Ah5v10or9



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

Staatsexamen 46115 / 2021 / Frühjahr / Thema Nr. 1 / Teilaufgabe Nr. 1 / Aufgabe Nr. 4

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

Staatsexamen 46115 / 2021 / Frühjahr / Thema Nr. 1 / Teilaufgabe Nr. 2 / Aufgabe Nr. 1

Aufgabe

Aufgabe 1 (Sortieren) [26 PUNKTE]
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 Feldindi- zes hier bei 1 beginnen. Die im Pseudocode verwendete Unterroutine Swap(A, :, j) vertauscht im Feld A die Elemente mit den Indizes i und j miteinander.

i) Insertionsort ii) Bubblesort iii) Quicksort

Insertionsort(int[ ] A) for 7 = 2 to A.length do key = Alj] i=j-1 while i>0 and Ali] > key do Afi + 1] = Ali] t=t-1 Ali + 1] = key

Bubblesort(int|] A) n := length(A) repeat swapped = false fori=lton—1do if Ali — 1] > Alc] then Swap(A,i — 1,7)

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 = Alr]

i=

for j =¢€tor—1do

if A[j] < pivot then

Swap(A, i, 5) w=i+l

Swap(A,i,r)

return i

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

Staatsexamen 46115 / 2021 / Frühjahr / Thema Nr. 1 / Teilaufgabe Nr. 2 / Aufgabe Nr. 2

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.

```java
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/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.

```java
  /**
   * 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 Einträg 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) {
        max = 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 Einträg 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) {
```

```
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 Einträg 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/bschlangaul/examen/examen_66115/jahr_2021/fruehjahr/MinimumMaximum.java`

Staatsexamen 46115 / 2021 / Frühjahr / Thema Nr. 1 / Teilaufgabe Nr. 2 / Aufgabe Nr. 3

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]

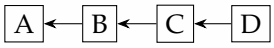
Ein gerichteter Graph heißt (schwach) zusammenhängend, falls der zugehörige ungerichtete Graph (also der Graph, der entsteht, wenn man jede gerichtete Kante durch eine ungerichtete Kante ersetzt) zusammenhängend ist. [b]

[a] `https://studyflix.de/informatik/grundbegriffe-der-graphentheorie-1285`
[b] `https://de.wikipedia.org/wiki/Zusammenhang_(Graphentheorie)`

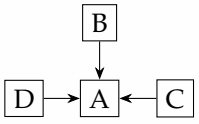
(a) Beschreiben Sie für ein allgemeines $n \in \mathbb{N}$ mit $n \geq 2$ den Aufbau eines schwach zusammenhängenden Graphen G_n, mit n Knoten, bei dem die Breitensuche $\Theta(n)$ mal gestartet werden muss, bis alle Knoten markiert sind.

?

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.

Staatsexamen 46115 / 2021 / Frühjahr / Thema Nr. 1 / Teilaufgabe Nr. 2 / Aufgabe Nr. 4

Aufgabe 5 [Sondierfolgen für Hashing mit offener Adressierung]

Eine Sondierfolge s(k,i) liefert für einen Schlüssel k aus einem Universum U und Versuchsnum- merni = 0,1,...,m-leine Folge von Indizes für eine Hashtabelle $T[0 \ldots m-1]$. Mithilfe einer Son- dierfolge 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,...,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 $s(k,i) = (h(k) + 2i) \mod m$, 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 $s(k,i) = (h(k) + i(i+1)) \mod m$, 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 $s(k,i) = (h(k) + i \cdot h'(k)) \mod m$, 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 $h(k) = k \mod 6$ und $h(k) = k^2 \mod 6$

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

Thema Nr. 2

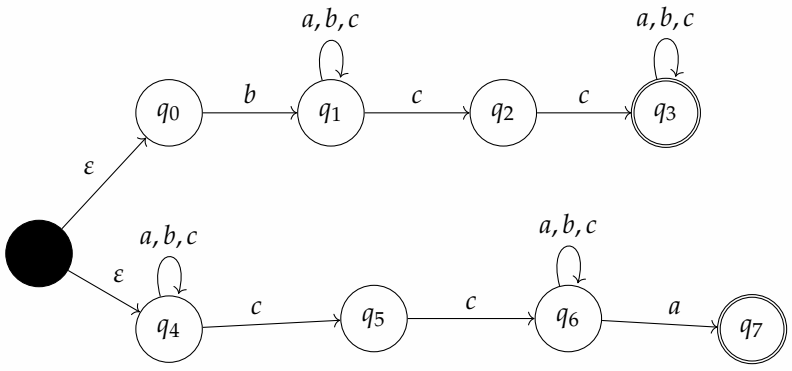
Teilaufgabe Nr. 1

Staatsexamen 46115 / 2021 / Frühjahr / Thema Nr. 2 / Teilaufgabe Nr. 1 / Aufgabe Nr. 1

Aufgabe 1 [Alphabet abc]

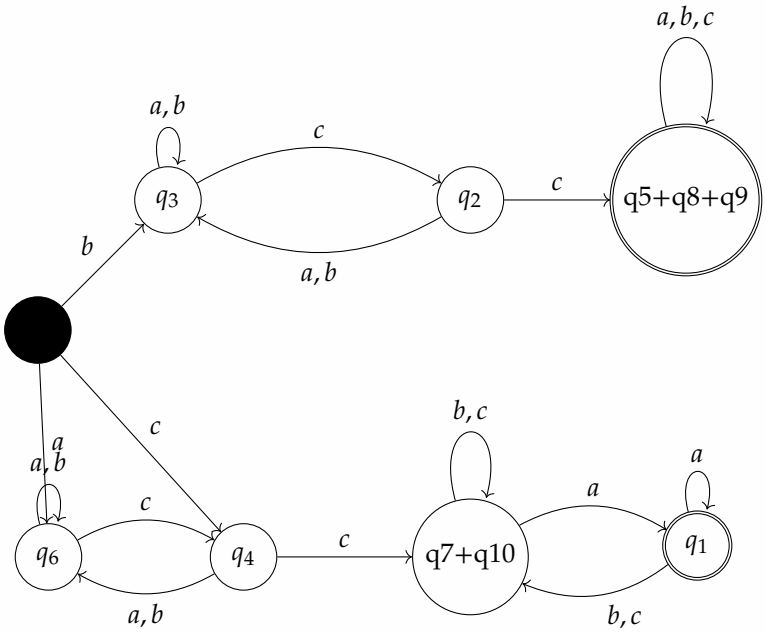
(a) Betrachten Sie die formale Sprache L © a,b,c*: aller Wörter, die entweder mit b beginnen oder mit a enden (aber nicht beides gleichzeitig) und das Teilwort cc enthalten. Entwerfen Sie einen (vollständigen) deterministischen endlichen Automaten, der die Sprache L akzeptiert. (Hinweis: Es werden weniger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

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.

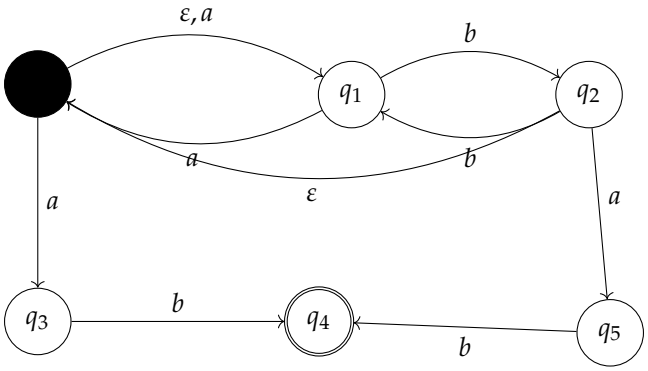
„Jede Teilsprache einer regulären Sprache ist regulär, d. h. für ein Alphabet und formale Sprachen L[' C LC * ist L' regular, falls L regulär ist."

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

Staatsexamen 46115 / 2021 / Frühjahr / Thema Nr. 2 / Teilaufgabe Nr. 1 / Aufgabe Nr. 2

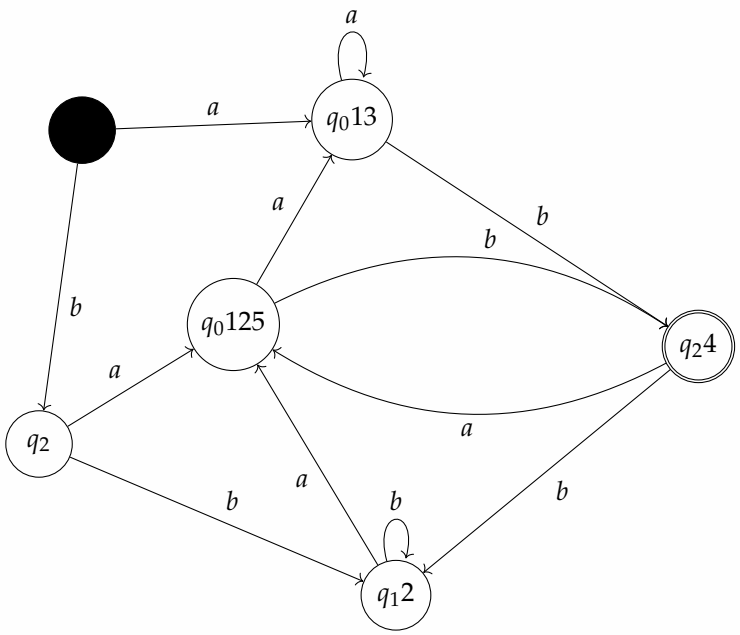
Aufgabe 2 [NEA ab]

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 Potenzmengenkonstruktion an.



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/Arnqcysfc

(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

Staatsexamen 46115 / 2021 / Frühjahr / Thema Nr. 2 / Teilaufgabe Nr. 1 / Aufgabe Nr. 3

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

Sei No = 0,1,2,... die Menge aller natürlichen Zahlen mit 0. Betrachten Sie die folgenden Sprachen.

(a) Ly = a"brar Ine No

P = {

$S \to aaaBa \mid \varepsilon$

$B \to$

}

(b) Le = aä"dne No

$a^{5n}b^n$

P = {

$S \to aaaaaSb \mid \varepsilon$

}

(c) L3 = (ab)ä(ba)"b(ab)" | n € No

Geben Sie jeweils an, ob L„ L2 und L3 kontextfrei und ob Ly, Lz und Ls regulär sind. Beweisen Sie Ihre Behauptung und ordnen Sie jede Sprache in die kleinstmögliche Klasse (regulär, kontext-

frei, nicht kontextfrei) ein. Für eine Einordnung in konterxtfrei zeigen Sie also, dass die Sprache kontextfrei und nicht regulär ist.

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

Teilaufgabe Nr. 2

Staatsexamen 46115 / 2021 / Frühjahr / Thema Nr. 2 / Teilaufgabe Nr. 2 / Aufgabe Nr. 1

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 € sowie die Gleichheit = für die Beziehung zwischen den Mengen. Folgendes Beispiel illustriert diese Schreib- weise für einige Funktionen fı bis fs. (Diese haben nichts mit den unten angegebenen Funktionen zu tun.)

O(fa) & O(fs) = O(fs) O(f:) = O(a) Die angegebenen Beziehungen miissen weder bewiesen noch begründet werden.

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

Staatsexamen 46115 / 2021 / Frühjahr / Thema Nr. 2 / Teilaufgabe Nr. 2 / Aufgabe Nr. 2

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`

(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      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

(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

```
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

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;
```

```
  }

  public int min() {
    return min;
  }

  class Item {
    private int val;
    private Item next;

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

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/Stack.java`

```
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertThrows;

import org.junit.Test;

public class StackTest {

  private Stack makeStack() {
    Stack stack = new Stack();
    stack.push(1);
    stack.push(2);
    stack.push(3);
    return stack;
  }

  @Test
  public void methodPop() {
    Stack stack = makeStack();
    assertEquals(3, stack.pop());
    assertEquals(stack.size(), 2);
    assertEquals(2, stack.pop());
    assertEquals(stack.size(), 1);
    assertEquals(1, stack.pop());
    assertEquals(stack.size(), 0);
    assertThrows(IndexOutOfBoundsException.class, () -> {
      stack.pop();
    });
  }

  @Test
  public void methodSize() {
    Stack stack = makeStack();
    assertEquals(stack.size(), 3);
  }

  @Test
  public void methodMin() {
    Stack stack = makeStack();
    assertEquals(stack.min(), 1);
  }
}
```

Code-Beispiel auf Github ansehen: `src/test/java/org/bschlangaul/examen/examen_46115/jahr_2021/fruehjahr/StackTest.java`

Staatsexamen 46115 / 2021 / Frühjahr / Thema Nr. 2 / Teilaufgabe Nr. 2 / Aufgabe Nr. 3

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 = Ali] oder NIL, falls © & A.

ı int BinarySearch(int[ ] A, int r) 2 f=1 3 r= A.length 4 while r > é do 5 m = | 45] 6 if 2 < Alm] then 7 r=om—l 8 else if 2 = Alm] then 9 | return m 10 else 11 | €=m+l1 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 /,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 Algorith- mus 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 auf- einanderfolgende, unterschiedliche Werte vertauscht wurden? Betrachten Sie hierbei die be- troffenen 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: Alk] =1 und Ak—1]=0.

Betrachten Sie die folgende rekursive Variante von BinarySearch.

1 int RekBinarySearch(int[ ] A. int x. int £. int r)
| mi
3 | (rekursive Implementierung)
Der initiale Aufruf der rekursiven Variante lautet: RekBinarySearch (A, z, 1, A.length)

Staatsexamen 46115 / 2021 / Frühjahr / Thema Nr. 2 / Teilaufgabe Nr. 2 / Aufgabe Nr. 4

Aufgabe 4

(a) Berechnen Sie im gegebenen gerichteten und gewichteten Graph G = (V,E,w) mit Kantenlängen w : E — R mittels des Dijkstra-Algorithmus die kürzesten (gerichteten) Pfade ausgehend vom Startknoten a.

Knoten, deren Entfernung von a bereits f+eststeht, seien als schwarz bezeichnet und Knoten, bei denen lediglich eine obere Schranke # oo 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 po- tentiell auch negativen Kantengewichten w : # — 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 niedrigs- ten (also des betragsmäßig größten negativen) Kantengewichts im Graphen zu allen Kanten addiert wird.