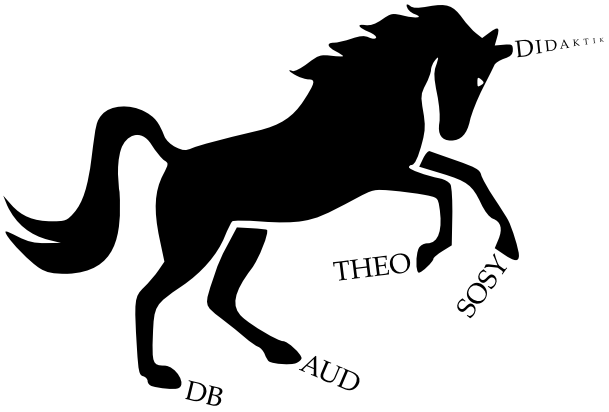
46115
Frühjahr
2020

Theoretische Informatik / Algorithmen / Datenstrukturen (nicht vertieft)

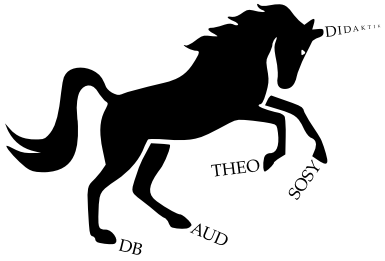
Aufgabenstellungen mit Lösungsvorschlägen



Die Bschlangaul-Sammlung

Hermine Bschlangaul and Friends

Aufgabenübersicht



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.

Thema Nr. 1

Aufgabe 1 [Reguläre Sprache]

(a) Betrachten Sie die formale Sprache $L \subseteq \{0,1\}^*$ aller Wörter, die 01 oder 110 als Teilwort enthalten.

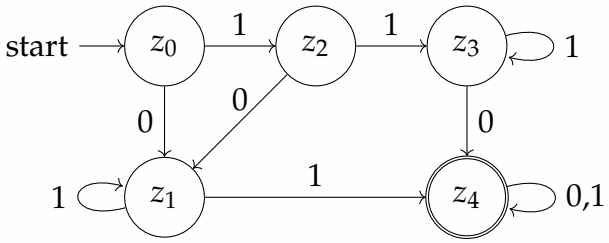
Geben Sie einen regulären Ausdruck für die Sprache L an.

Lösungsvorschlag

(0|1)*(01|110)(0|1)*

(b) Entwerfen Sie einen (vollständigen) deterministischen endlichen Automaten, der die Sprache L aus Teilaufgabe (a) akzeptiert. (Hinweis: es werden nicht mehr als 6 Zustände benötigt.)

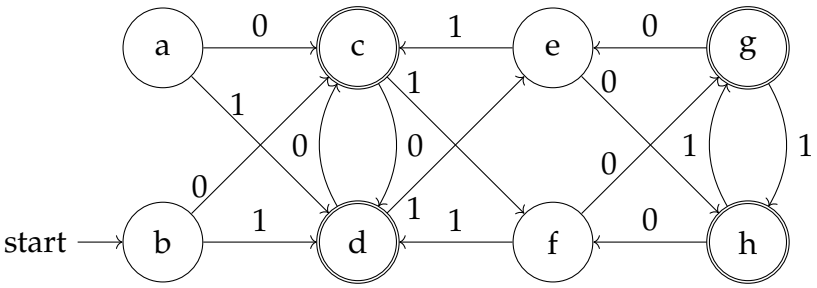
Lösungsvorschlag



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/A54gek0vz

(c) Minimieren Sie den folgenden deterministischen endlichen Automaten:

Machen Sie dabei Ihren Rechenweg deutlich!



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/Ajpw4j73w

Lösungsvorschlag

a	∅	∅	∅	∅	∅	∅	∅	∅
b		∅	∅	∅	∅	∅	∅	∅
c	x_1	x_1	∅	∅	∅	∅	∅	∅
d	x_1	x_1		∅	∅	∅	∅	∅
e	x_2	x_2	x_1	x_1	∅	∅	∅	∅
f	x_3	x_3	x_1	x_1		∅	∅	∅
g	x_1	x_1	x_2	x_2	x_1	x_1	∅	∅
h	x_1	x_1	x_2	x_2	x_1	x_1		∅
	a	b	c	d	e	f	g	h

x_1 Paar aus End-/ Nicht-Endzustand kann nicht äquivalent sein.

x_2 Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.

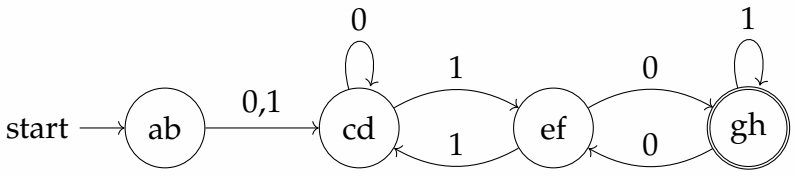
x_3 In weiteren Iterationen markierte Zustände.

x_4 ...

Die Zustandpaare werden aufsteigend sortiert notiert.

Übergangstabelle

Zustandspaar	**0**	**1**
(a, b)	(c, c)	(d, d)
(a, e)	(c, h) x_2	(c, d)
(a, f)	(c, g) x_3	(d, d)
(b, e)	(c, h) x_2	(c, d)
(b, f)	(c, g) x_3	(d, g)
(c, d)	**(c, d)**	**(e, f)**
(c, g)	(d, e) x_2	(e, f)
(c, h)	(d, f) x_2	(f, f)
(d, g)	(c, e) x_2	(e, e)
(d, h)	(c, f) x_2	(e, f)
(e, f)	**(g, h)**	**(c, d)**
(g, h)	**(e, f)**	**(g, h)**



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/Arzvh5kyz

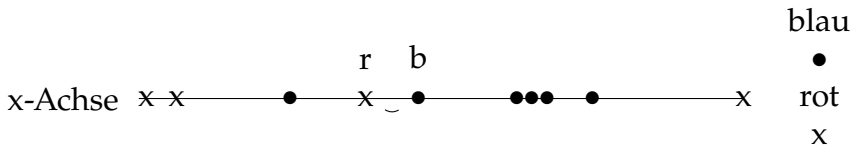
(d) Ist die folgende Aussage richtig oder falsch? Begründen Sie Ihre Antwort!

„Zu jeder regulären Sprache L über dem Alphabet Σ gibt es eine Sprache $L' \subseteq \Sigma^*$, die L enthält ($\eth L \subseteq L$) und nicht regulär ist.“

Thema Nr. 2

Aufgabe 6 [Nächstes rot-blaues Paar auf der x-Achse]

Gegeben seien zwei nichtleere Mengen R und B von roten bzw. blauen Punkten auf der x-Achse. Gesucht ist der minimale euklidische Abstand $d(r,b)$ über alle Punktepaare (r,b) mit $r \in R$ und $b \in B$. Hier ist eine Beispielinstanz:



Die Eingabe wird in einem Feld A übergeben. Jeder Punkt $A[i]$ mit $1 \leq i \leq n$ hat eine x-Koordinate $A[i].x$ und eine Farbe $A[i].color \in \{\text{rot}, \text{blau}\}$. Das Feld A ist nach x-Koordinate sortiert, ðes gilt $A[1].x < A[2].x < \cdots < A[n].x$, wobei $n = |R| + |B|$.

(a) Geben Sie in Worten einen Algorithmus an, der den gesuchten Abstand in $\mathcal{O}(n)$ Zeit berechnet.

Lösungsvorschlag

Pseudo-Code

Algorithmus 1: Minimaler Euklidischer Abstand

$d_{min} := \max$; // Setze d_{min} zuerst auf einen maximalen Wert.
for *i in* $0 \ldots$ *vorletzter Index* **do** ; // Iteriere über die Indizes des Punkte-Arrays P bis zum vorletzten Index $P[n-1]$
  if $P[n].color \neq P[n+1].color$ **then** ; // Berechne den Abstand nur, wenn die Punkte unterschiedliche Farben haben
    $d = P[n+1].x - P[n].x$
    if $d < d_{min}$ **then**
      $d_{min} = d$
    end
  end
end

Java

```java
public double findMinimalDistance() {
  double distanceMin = Double.MAX_VALUE;
  for (int i = 0; i < latestIndex - 1; i++) {
    if (points[i].color != points[i + 1].color) {
      double distance = points[i + 1].x - points[i].x;
      if (distance < distanceMin) {
        distanceMin = distance;
```

```
        }
      }
    }
    return distanceMin;
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/fruehjahr/RedBluePairCollection.java`

(b) Begründen Sie kurz die Laufzeit Ihres Algorithmus.

Lösungsvorschlag

Da das Array der Länge n nur einmal durchlaufen wird, ist die Laufzeit $\mathcal{O}(n)$ sichergestellt.

(c) Begründen Sie die Korrektheit Ihres Algorithmus.

Lösungsvorschlag

In d_{min} steht am Ende der gesuchte Wert (sofern nicht $d_{min} = Integer.MAX\_VALUE$ geblieben ist)

(d) Wir betrachten nun den Spezialfall, dass alle blauen Punkte links von allen roten Punkten liegen. Beschreiben Sie in Worten, wie man in dieser Situation den gesuchten Abstand in $o(n)$ Zeit berechnen kann. (Ihr Algorithmus darf also insbesondere nicht Laufzeit $\Theta(n)$ haben.)

Lösungsvorschlag

Zuerst müssen wir den letzten blauen Punkt finden. Das ist mit einer binären Suche möglich. Wir beginnen mit dem ganzen Feld als Suchbereich und betrachten den mittleren Punkt. Wenn er blau ist, wiederholen wir die Suche in der zweiten Hälfte des Suchbereichs, sonst in der ersten, bis wir einen blauen Punkt gefolgt von einem roten Punkt gefunden haben.

Der gesuchte minimale Abstand ist dann der Abstand zwischen dem gefundenen blauen und dem nachfolgenden roten Punkt. Die Binärsuche hat eine Worst-case-Laufzeit von $\mathcal{O}(\log n)$.