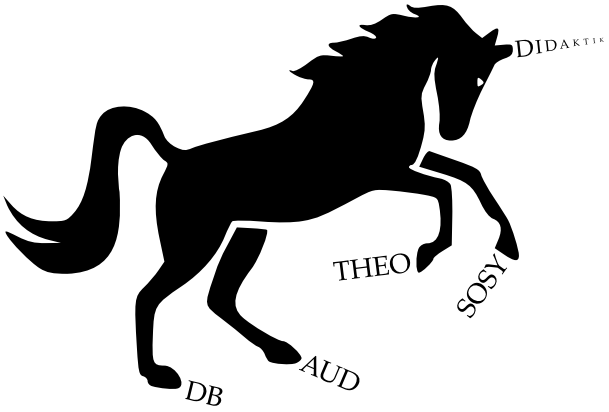
66115
Frühjahr
2019

Theoretische Informatik / Algorithmen (vertieft)

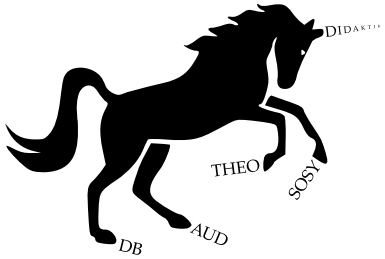
Aufgabenstellungen mit Lösungsvorschlägen



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 [Wissensfragen wahr / falsch]

Antworten Sie auf die folgenden Behauptungen mit Wahr/Falsch und geben Sie eine kurze Begründung an.

(a) Wenn L_2 regulär ist und $L_1 \subseteq L_2$ gilt, dann ist L_1 auch regulär.

Lösungsvorschlag

Falsch

Nein. Wähle $L = \Sigma^*$. Wähle eine beliebige nicht-reguläre Sprache L'. L' ist Teilmenge von L. [a]

$L_2 = \{a^*b^*\}$ ...Regulär

$L_1 = \{a^n b^n\}$ ...nicht regulär,

da man sich das n merken muss, das bedeutet, dass man beispielsweise einen Automaten bräuchte, der unendlich viele Zustände besitzt. Das wiederum ist aber bei DEA's nicht möglich. (Da sie nur endlich viele Zustände haben können.) Für beide Sprachen gilt somit: $L_1 \subseteq L_2$, aber nicht beide sind regulär. [b]

[a] https://www.c-plusplus.net/forum/topic/287036/theoretische-informatik-teilmenge-einer-regulä

[b] https://vowi.fsinf.at/images/3/3a/TU_Wien-Theoretische_Informatik_und_Logik_VU_(Fermüller,_Freund)-Übungen_SS13_-_Uebungsblatt_1.pdf

(b) $L = \{ a^q \mid \exists i \in \mathbb{N}. q = i^2 \}$ ist bekanntlich nicht regulär. Behauptung: Q^* ist ebenfalls nicht regulär.

Lösungsvorschlag

Wahr.

Siehe Abschlusseigenschaften der Formalen Sprachen unter dem Kleene-Stern.

(c) Wenn $L \subseteq \Sigma^*$ entscheidbar ist, dann ist auch das Komplement $\bar{L} = \Sigma^* \setminus L$ entscheidbar.

Lösungsvorschlag

Wahr

Zu jeder entscheidbaren Menge ist auch ihr Komplement entscheidbar.

L entscheidbar, dann auch $\overline{L}$ entscheidbar. Wir benutzen eine DTM τ', die die DTM τ simuliert. Diese vertauscht die beiden Ausgaben „JA“ und „NEIN“. [a]

[a] http://www.informatikseite.de/theorie/node14.php#SECTION00044100000000000000

(d) Jedes $\mathcal{NP}$-vollständige Problem ist entscheidbar.

Lösungsvorschlag

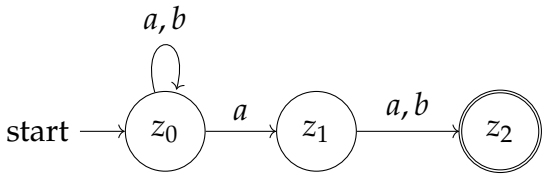
Wahr

Definition von $\mathcal{NP}$-Vollständigkeit:

vollständig für die Klasse der Probleme, die sich nichtdeterministisch in Polynomialzeit lösen lassen, wenn es zu den schwierigsten Problemen in der Klasse NP gehört, also sowohl in NP liegt als auch NP-schwer ist.

Aufgabe 2 [NEA nach DEA]

(a) Gegeben sei der nichtdeterministische endliche Automat A über dem Alphabet $\Sigma = \{a, b\}$ wie folgt:



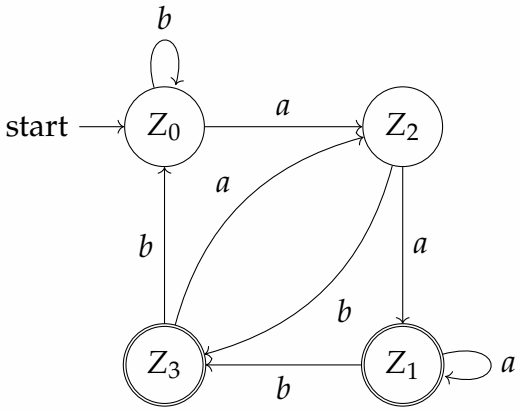
Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Arozq4rm2

Konstruieren Sie einen deterministischen endlichen Automaten, der das Komplement $\bar{L}(A) = \{ w \in \Sigma^* \mid w \notin L(A) \}$ der von A akzeptierten Sprache $L(A)$ akzeptiert.

Lösungsvorschlag

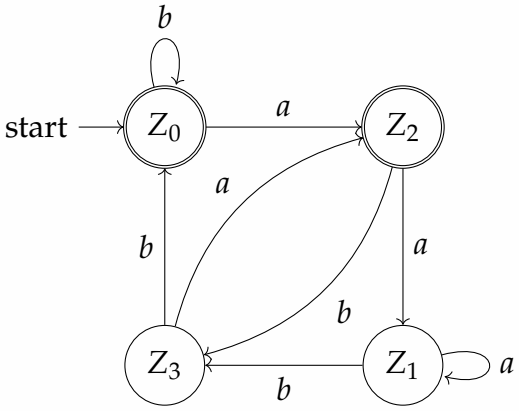
Wir konvertieren zuerst den nichtdeterministischen endlichen Automaten in einen deterministischen endlichen Automaten mit Hilfe des Potenzmengenalgorithmus.

Zustandsmenge	Eingabe a	Eingabe b
Z_0 $\{z_0\}$	Z_1 $\{z_0, z_1\}$	Z_0 $\{z_0\}$
Z_1 $\{z_0, z_1\}$	Z_2 $\{z_0, z_1, z_2\}$	Z_3 $\{z_0, z_2\}$
Z_2 $\{z_0, z_1, z_2\}$	Z_2 $\{z_0, z_1, z_2\}$	Z_3 $\{z_0, z_2\}$
Z_3 $\{z_0, z_2\}$	Z_1 $\{z_0, z_1\}$	Z_0 $\{z_0\}$



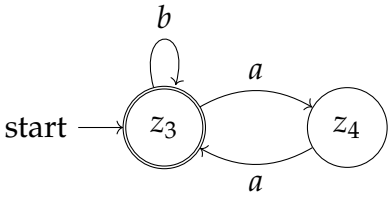
Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Arxujcbdg

Wir vertauschen die End- und Nicht-End-Zustände, um das Komplement zu erhalten:



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/A5zqsonq2

(b) Gegeben sei zudem der nichtdeterministische Automat B über dem Alphabet $\Sigma = \{a, b\}$:

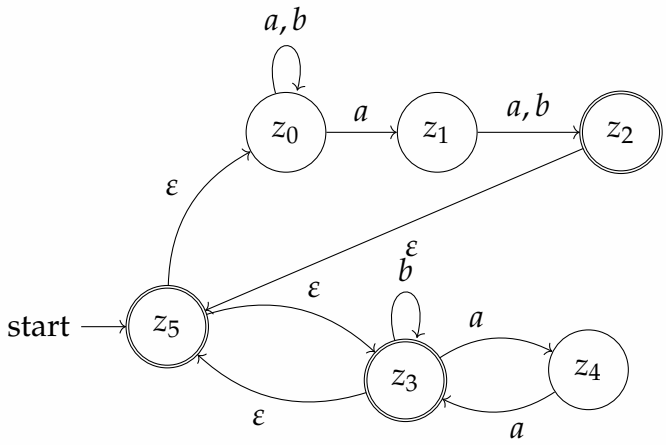


Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Arafgk0h2

Konstruieren Sie einen endlichen Automaten (möglicherweise mit ε-Übergängen), der die Sprache $(L(A)L(B))^* \subseteq \Sigma^*$ akzeptiert (A aus der vorigen Aufgabe). Erläutern Sie auch Ihre Konstruktionsidee.

Lösungsvorschlag

$L(A)L(B))^*$ ist die beliebige Konkatenation (Verknüpfung/Verkettung) der Sprachen $L(A)$ und $L(B)$ mit dem leeren Wort. Wir führen einen neuen Startzustand (z_5) ein, der zugleich Endzustand ist. Dadurch wird das leere Wort akzeptiert. Dieser neue Startzustand führt über ε-Übergängen zu den ehemaligen Startzuständen der Automaten A und B. Die Endzustände der Automaten A und B führen über ε-Übergängen zu z_5. Dadurch sind beliebige Konkatenationen möglich.



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/Aro3uhzjz

Aufgabe 3 [Nonterminale: STU, Terminale: ab]

Gegeben sei die kontextfreie Grammatik $G = (V, \Sigma, P, S)$ mit Sprache $L(G)$, wobei $V = \{S, T, U\}$ und $\Sigma = \{a, b\}$. P bestehe aus den folgenden Produktionen:

$$\mathrm{P} = \left\{ \begin{array}{c} S \to TUUT \\ T \to aT \mid \varepsilon \\ U \to bUb \mid a \end{array} \right\}$$

[1]

(a) Geben Sie fünf verschiedene Wörter $w \in \Sigma^*$ mit $w \in L(G)$ an.

Lösungsvorschlag

- aa
- aaaa
- ababbaba
- aababbabaa
- abbabbbbabba

(b) Geben Sie eine explizite Beschreibung der Sprache $L(G)$ an.

[1] `https://flaci.com/Gjpsin26a`

Lösungsvorschlag

$$L = \{ a^* b^n a b^{2n} a b^n a^* \mid n \in \mathbb{N}_0 \}$$

(c) Bringen Sie G in Chomsky-Normalform und erklären Sie Ihre Vorgehensweise.

Lösungsvorschlag

(i) **Elimination der ε-Regeln**

— *Alle Regeln der Form $A \to \varepsilon$ werden eliminiert. Die Ersetzung von A wird durch ε in allen anderen Regeln vorweggenommen.* ——

$$\mathrm{P} = \Big\{ \begin{aligned} S &\to TUUT \mid TUU \mid UUT \mid UU \\ T &\to aT \mid a \\ U &\to bUb \mid a \end{aligned} \Big\}$$

(ii) **Elimination von Kettenregeln**

— *Jede Produktion der Form $A \to B$ mit $A, B \in S$ wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren.* ——

$\varnothing$ Nichts zu tun

(iii) **Separation von Terminalzeichen**

— *Jedes Terminalzeichen σ, das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal S_σ ersetzt und die Menge der Produktionen durch die Regel $S_\sigma \to \sigma$ ergänzt.* —

$$\mathrm{P} = \Big\{ \begin{aligned} S &\to TUUT \mid TUU \mid UUT \mid UU \\ T &\to AT \mid A \\ U &\to BUB \mid A \\ A &\to a \\ B &\to b \end{aligned} \Big\}$$

(iv) **Elimination von mehrelementigen Nonterminalketten**

— *Alle Produktionen der Form $A \to B_1 B_2 \dots B_n$ werden in die Produktionen $A \to A_{n-1} B_n, A_{n-1} \to A_{n-2} B_{n-1}, \dots, A_2 \to B_1 B_2$ zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht.* ——

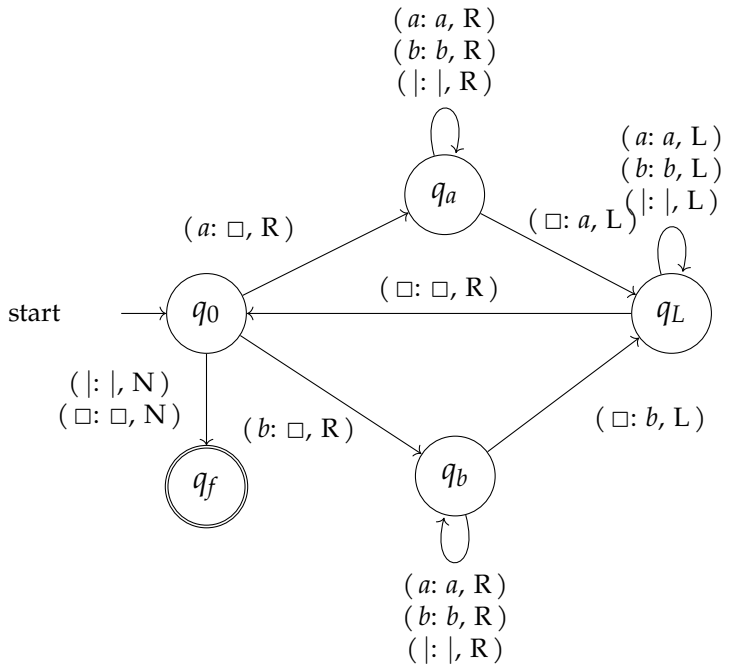
$$\mathrm{P} = \Big\{$$

$$
\begin{aligned}
S &\to TS_1 \mid TS_3 \mid US_2 \mid UU \\
S_1 &\to US_2 \\
S_2 &\to UT \\
S_3 &\to UU \\
T &\to AT \mid a \\
U &\to BU_1 \mid a \\
U_1 &\to UB \\
A &\to a \\
B &\to b
\end{aligned}
$$

}

Aufgabe 4 [Turingmaschine Konfigurationsfolge]

Wir betrachten die Turingmaschine $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$. Hierbei ist die Zustandsmenge $Q = \{q_0, q_a, q_b, q_L, q_f\}$ mit Startzustand q_0 und akzeptierenden Zuständen $F = \{q_f\}$. Das Eingabealphabet ist $\Sigma = \{a, b, |\}$[2] das Bandalphabet ist $\Gamma = \Sigma \cup \{\square\}$ mit Blank-Zeichen $\square$ für leeres Feld. Die Übergangsfunktion $\delta : Z \times \Gamma \to Z \times \Gamma \times \{L, R, N\}$, wobei der Schreib-Lese-Kopf mit L nach links, mit N nicht und mit R nach rechts bewegt wird, ist durch folgende Tabelle gegeben (bspw. ist $\delta(q_0, a) = (q_a, \square, R)$):



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/Aj54q4rd9

[2] In der Angabe ist das Trennzeichen ein „$“. Wir verwenden stattdessen ein „|“, denn „$“ ist eine Tex-Sonderzeichen und müsste deshalb ständig besonders behandelt werden.

(a) Die Notation (v, q, aw) beschreibt eine Konfiguration der Turingmaschine: der interne Zustand ist q, der Schreib-Lesekopf steht auf einem Feld mit $a \in \Gamma$, rechts vom Schreib-Lesekopf steht $w \in \Gamma^*$, links vom Schreib-Lesekopf steht $v \in \Gamma^*$.

Vervollständigen Sie die Folge von Konfigurationen, die die Turingmaschine bei Eingabe $ab|$ bis zum Erreichen des Zustands q_f durchläuft. Sie können auch Ihre eigene Notation zur Darstellung von Konfigurationen verwenden.

$(\square, q_0, ab|) \vdash$
$(\square, q_a, \square b|) \vdash$
$(\square, q_a, b|) \vdash$
...

Lösungsvorschlag

$(\square, q_0, ab|) \vdash$
$(\square, q_a, \square b|) \vdash$
$(\square, q_a, b|) \vdash$
$(b, q_a, |) \vdash$
$(b|, q_L, a) \vdash$
$(b, q_L, |a) \vdash$
$(\square, q_L, \square b|a) \vdash$
$(\square, q_b, \square b|a) \vdash$
$(\square, q_b, \square|a) \vdash$
$(\square, q_b, |a) \vdash$
$(|, q_b, a) \vdash$
$(|a, q_L, b) \vdash$
$(|, q_L, ab) \vdash$
$(\square, q_L, |ab) \vdash$
$(\square, q_0, \square|ab) \vdash$
$(\square, q_f, |ab)$

(b) Sei $w \in \{a, b\}^*$ beliebig. Mit welchem Bandinhalt terminiert die Turingmaschine bei Eingabe von $w|$? Geben Sie auch eine kurze Begründung an.

Lösungsvorschlag

Die Turingmaschine terminiert bei alle möglichen Wörtern $w \in \{a, b\}^*$, auch bei dem leeren Wort vor |. Die Turing-Maschine verschiebt alle a's und b's vor dem Trennzeichen | nach rechts. Ist das Trennzeichen | schließlich das erste Zeichen von links gesehen, dann terminiert die Maschine.

Aufgabe 5 [BEHAELTER GERADEBEHAELTER]

Wir betrachten das Behälterproblem BEHAELTER. Gegeben ist eine Menge von k € N Behältern, die jeweils ein Fassungsvermögen der Größe b € N haben. Gegeben sind weiterhin n Objekte mit jeweiligen Größen a1,...,@,,. Gesucht ist eine Zuordnung der n Objekte auf die k Behälter, sodass keiner der Behälter überläuft.

Formal sind Instanzen des Behalterproblems BEHAELTER durch Tupel (k,, a1,...@n) gegeben, die wie folgt zu interpretieren sind:

- k EN steht für eine Anzahl von Behältern.
- Jeder Behälter hat ein Fassungsvermögen von bEN.
- Die a, stehen für die jeweiligen Größen von n Objekten.

Zuordnungen von Objekten zu Behältern geben wir durch eine Funktion v an, wobei v(j) = i wenn das j-te Objekt (mit Größe a,) dem i-ten Behälter zugeordnet wird.

(k,b,a1,...Q,) ist eine JA-Instanz von BEHAELTER, wenn es eine Zuordnung v von Objekten auf Behälter (v : [1;n] — [1; k]) gibt, die sicherstellt, dass kein Behälter überläuft:

(k,b,a1,...4n) € BEHABLTER <=> (3v: [1;n] > [1;k]. Vik. S> a; <0) i=v(3)

Wir betrachten auch das modifizierte Problem GERADEBEHAELTER. Instanzen von GERADEBEHAELTER tragen die zusätzliche Einschränkung, dass alle a; gerade (durch zwei teilbar) sein müssen.

(a) Warum ist sowohl BEHAELTER € NP als auch GERADEBEHAELTER € NP?

(b) Beweisen Sie, dass das Problem BEHAELTER auf das Problem GERADEBEHAELTER in polynomieller Zeit reduzierbar ist.

(c) BEHAELTER ist NP-vollständig. Begründen Sie, was obige Reduktion für die Komplexität von GERADEBEHAELTER bedeutet. BEHAELTER ist NP-vollständig. Begründen Sie, was obige Reduktion für die Komplexität von GERADEBEHAELTER bedeutet.

Aufgabe 6 [Muffinsorten]

Aus dem Känguru-Wettbewerb 2017 — Klassenstufen 3 und 4.

> Luna hat für den Kuchenbasar Muffins mitgebracht: 10 Apfelmuffins, 18 Nussmuffins, 12 Schokomuffins und 9 Blaubeermuffins. Sie nimmt immer 3 verschiedene Muffins und legt sie auf einen Teller. Welches ist die kleinste Zahl von Muffins, die dabei übrig bleiben können?
>
> A: 1, B: 3, C: 4, D: 7, E: 8

(a) Geben Sie die richtige Antwort auf die im Känguru-Wettbewerb gestellte Frage und begründen Sie sie.

Lösungsvorschlag

4 [a]

[a] https://www.youtube.com/watch?v=ceJW9kAplVY

(b) Lunas Freundin empfiehlt den jeweils nächsten Teller immer aus den drei aktuell häufigsten Muffinsorten zusammenzustellen. Leiten Sie aus dieser Idee einen effizienten GreedyAlgorithmus her, der die Fragestellung für beliebige Anzahlen von Muffins löst (nach wie vor soll es nur vier Sorten und je drei pro Teller geben). Skizzieren Sie in geeigneter Form, wie Ihr Algorithmus die Beispielinstanz von oben richtig löst.

Lösungsvorschlag

```java
public static int berechneRest4Sorten3ProTeller() {
  int[] muffins = new int[] { 10, 18, 12, 9 };
  int n = muffins.length;
  sortiere(muffins);

  // Wir nehmen uns 3 verschiedene Muffins solange, wie die dritthäufigste
  // Muffinsorte noch Muffins hat.
  while (muffins[n - 3] > 0) {
    muffins[n - 1]--;
    muffins[n - 2]--;
    muffins[n - 3]--;
    sortiere(muffins);
  }
  return berechneGesamtzahl(muffins);
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2019/fruehjahr/Muffin.java`

(c) Beschreiben Sie eine mögliche und sinnvolle Verallgemeinerung Ihrer Lösung auf n Muffinsorten und k Muffins pro Teller für $n > 4$ und $k > 3$.

Lösungsvorschlag

```java
public static int berechneRestAllgemein(int[] muffins, int k) {
  int n = muffins.length;
  sortiere(muffins);
  while (muffins[n - k] > 0) {
    for (int i = 1; i <= k; i++) {
      muffins[n - i]--;
    }
    sortiere(muffins);
  }
  return berechneGesamtzahl(muffins);
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2019/fruehjahr/Muffin.java`

(d) Diskutieren Sie, wie man die Korrektheit des Greedy-Algorithmus zeigen könnte, also dass er tatsächlich immer eine optimale Lösung findet. Ein kompletter, rigoroser Beweis ist nicht verlangt.

Thema Nr. 2

Aufgabe 1 [k-kleinste Elemente]

Gegeben sei eine unsortierte Liste von n verschiedenen natürlichen Zahlen. Das k-kleinste Element ist das Element, das größer als genau $k-1$ Elemente der Liste ist.

(a) Geben Sie einen Algorithmus mit Laufzeit $\mathcal{O}(n \cdot \log n)$ an, um das k-kleinste Element zu berechnen.

Lösungsvorschlag

a

[a] https://en.wikipedia.org/wiki/Quickselect

(b) Gegeben sei nun ein Algorithmus A, der den Median einer unsortierten Liste von n Zahlen in $\mathcal{O}(n)$ Schritten berechnet. Nutzen Sie Algorithmus A um einen Algorithmus B anzugeben, welcher das k-kleinste Element in $\mathcal{O}(n)$ Schritten berechnet.

Argumentieren Sie auch, dass der Algorithmus die gewünschte Laufzeit besitzt.

Lösungsvorschlag

a

[a] https://en.wikipedia.org/wiki/Median_of_medians

(c) Geben Sie einen Algorithmus an, der für alle $i = 1 \ldots, \lfloor n/k \rfloor$ das $i \cdot k$-kleinste Element berechnet. Die Laufzeit Ihres Algorithmus sollte $\mathcal{O}(n \cdot log(n/k))$ sein. Sie dürfen weiterhin Algorithmus A, wie in Teilaufgabe (b) beschrieben, nutzen.