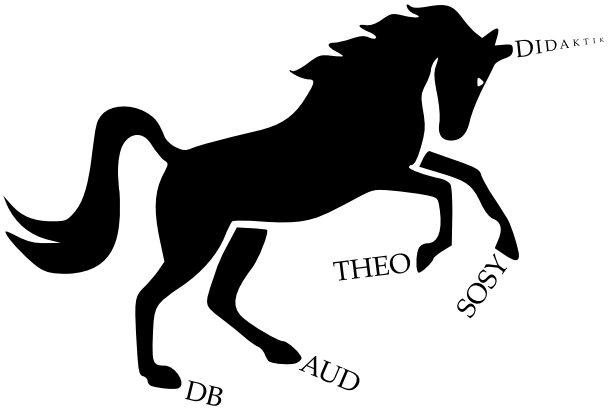
66115
Frühjahr
2018

Theoretische Informatik / Algorithmen (vertieft)

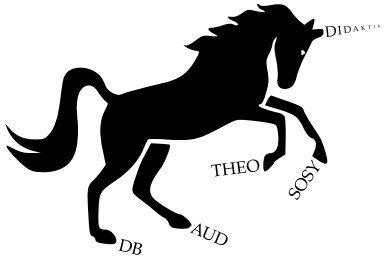
Aufgabenstellungen mit Lösungsvorschlägen



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.

Diese Materialsammlung unterliegt den Bestimmungen der Creative Commons Namensnennung-Nicht kommerziell-Share Alike 4.0 International-Lizenz.

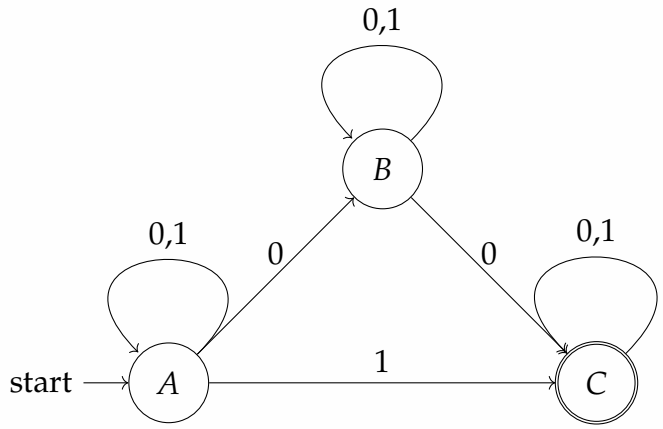
Thema Nr. 1

Aufgabe 1 [NEA]

Gegeben ist der nichtdeterministische endliche Automat (NEA) $(\{0,1\}, Q, \delta, q_0, F)$, wobei $Q = \{A, B, C\}$, $q_0 = A$, $F = \{C\}$ und

δ	0	1
A	$\{A, B\}$	$\{A, C\}$
B	$\{B, C\}$	$\{B\}$
C	$\{C\}$	$\{C\}$

Lösungsvorschlag



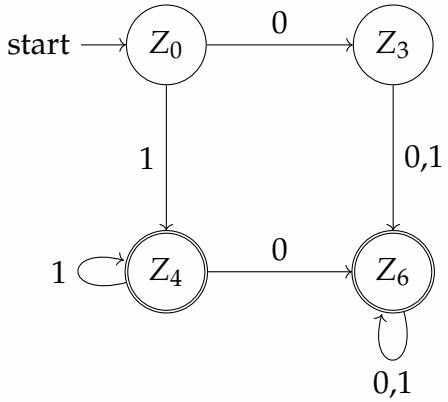
Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aiq3xxgi9

(a) Führen Sie für diesen NEA die Potenzmengenkonstruktion durch; geben Sie alle acht entstehenden Zustände mit ihren Transitionen an, nicht nur die erreichbaren.

Lösungsvorschlag

nicht erreichbar

Name	Zustandsmenge	Eingabe 0	Eingabe 1
Z_0	$Z_0\{A\}$	$Z_3\{A,B\}$	$Z_4\{A,C\}$
Z_1	$Z_1\{B\}$	$Z_5\{B,C\}$	$Z_1\{B\}$
Z_2	$Z_2\{C\}$	$Z_2\{C\}$	$Z_2\{C\}$
Z_3	$Z_3\{A,B\}$	$Z_6\{A,B,C\}$	$Z_6\{A,B,C\}$
Z_4	$Z_4\{A,C\}$	$Z_6\{A,B,C\}$	$Z_4\{A,C\}$
Z_5	$Z_5\{B,C\}$	$Z_5\{B,C\}$	$Z_5\{B,C\}$
Z_6	$Z_6\{A,B,C\}$	$Z_6\{A,B,C\}$	$Z_6\{A,B,C\}$
Z_7	$Z_7\{\}$	$Z_7\{\}$	$Z_7\{\}$



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/Aigwcbsf7

Thema Nr. 2

Aufgabe 3 [Exponentieller Blow-Up]

Gesucht ist eine reguläre Sprache $C \subseteq \{a, b\}^*$, deren minimaler deterministischer endlicher Automat (DEA) mindestens 4 Zustände mehr besitzt als der minimale nichtdeterministische endliche Automat (NEA). Gehen Sie wie folgt vor:

(a) Definieren Sie $C \subseteq \{a, b\}^*$ und erklären Sie kurz, warum es bei dieser Sprache NEAs gibt, die deutlich kleiner als der minimale DEA sind.

Lösungsvorschlag

Sprache mit exponentiellem Blow-Up:

Ein NEA der Sprache

$$\begin{aligned} L_k &= \{xay \mid x, y \in \{a, b\}^* \wedge |y| = k - 1\} \\ &= \{w \in \{a, b\}^* | \text{ der k-te Buchstabe von hinten ist ein } a\} \end{aligned}$$

kommt mit $k + 1$ Zuständen aus.

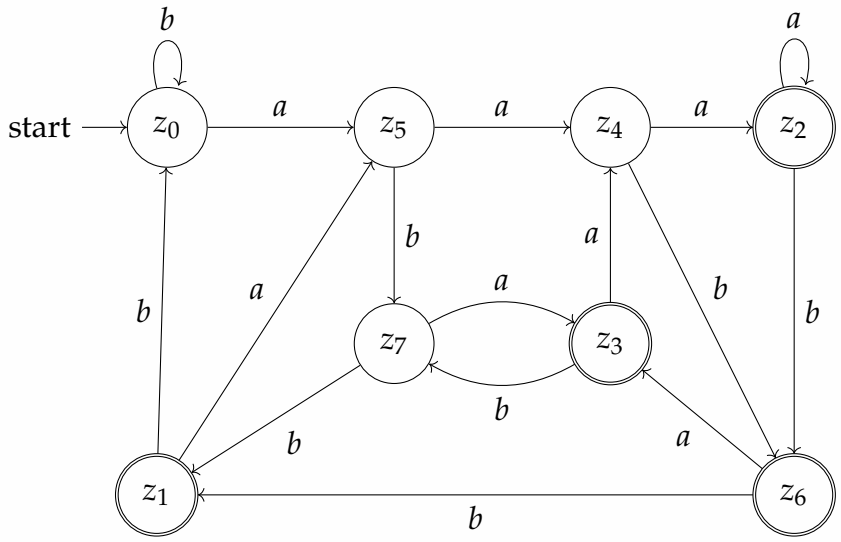
Jeder DEA M mit $L(M) = L$ hat dann mindestens 2^k Zustände. Wir wählen $k = 3$. Dann hat der zughörige NEA 4 Zustände und der zugehörige DEA mindestens 8. Sei also $L_k = \{xay | x, y \in \{a, b\}^* \wedge |y| = 2\}$ die gesuchte Sprache.

Der informelle Grund, warum ein DEA für die Sprache L_k groß sein muss, ist dass er sich immer die letzten n Symbole merken muss. [a]

[a] `https://www.tcs.ifi.lmu.de/lehre/ss-2013/timi/handouts/handout-02`

(b) Geben Sie den minimalen DEA M für C an. (Zeichnung des DEA genügt; die Minimalität muss nicht begründet werden.)

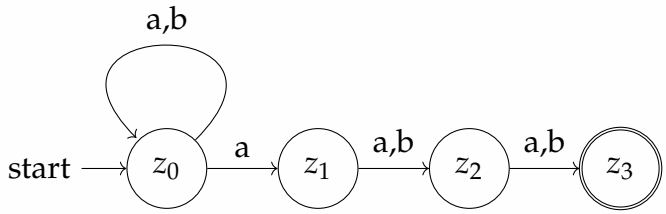
Lösungsvorschlag



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/Ahhefpjir

(c) Geben Sie einen NEA N für C an, der mindestens 4 Zustände weniger besitzt als M. (Zeichnung des NEA genügt)

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/Ajrz7h5r7

Aufgabe 6 [Methode „m()“]

Der Hauptsatz der Laufzeitfunktionen ist bekanntlich folgendermaßen definiert:

Exkurs: Master-Theorem

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

$a =$ Anzahl der rekursiven Aufrufe, Anzahl der Unterprobleme in der Rekursion ($a \geq 1$).

$\frac{1}{b} =$ Teil des Originalproblems, welches wiederum durch alle Unterprobleme repräsentiert wird, Anteil an der Verkleinerung des Problems ($b > 1$).

$f(n) =$ Kosten (Aufwand, Nebenkosten), die durch die Division des Problems und die Kombination der Teillösungen entstehen. Eine von $T(n)$ unabhängige und nicht negative Funktion.

Dann gilt:

1. Fall: $T(n) \in \Theta\left(n^{\log_b a}\right)$

falls $f(n) \in \mathcal{O}\left(n^{\log_b a - \varepsilon}\right)$ für $\varepsilon > 0$

2. Fall: $T(n) \in \Theta\left(n^{\log_b a} \cdot \log n\right)$

falls $f(n) \in \Theta\left(n^{\log_b a}\right)$

3. Fall: $T(n) \in \Theta(f(n))$

falls $f(n) \in \Omega\left(n^{\log_b a + \varepsilon}\right)$ für $\varepsilon > 0$ und ebenfalls für ein c mit $0 < c < 1$ und alle hinreichend großen n gilt: $a \cdot f\left(\frac{n}{b}\right) \leq c \cdot f(n)$

(a) Betrachten Sie die folgende Methode `m` in Java, die initial mit `m(r, 0, r.length)` für das Array `r` aufgerufen wird. Geben Sie dazu eine Rekursionsgleichung $T(n)$ an, welche die Anzahl an Rechenschritten von `m` in Abhängigkeit von der Länge `n = r.length` berechnet.

```java
public static int m(int[] r, int lo, int hi) {
  if (lo < 8 || hi <= 10 || lo >= r.length || hi > r.length) {
    throw new IllegalArgumentException();
  }

  if (hi - lo == 1) {
    return r[lo];
  } else if (hi - lo == 2) {
    return Math.max(r[lo], r[lo + 1]); // O(1)
  } else {
    int s = (hi - lo) / 3;
    int x = m(r, lo, lo + s);
    int y = m(r, lo + s, lo + 2 * s);
    int z = m(r, lo + 2 * s, hi);
    return Math.max(Math.max(x, y), 2); // O(1)
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2018/fruehjahr/MasterTheorem.java`

Lösungsvorschlag

Allgemeine Rekursionsgleichung:

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

Anzahl der rekursiven Aufrufe (a)**:**

3

Anteil Verkleinerung des Problems (b)**:**

um $\frac{1}{3}$ also $b = 3$

Laufzeit der rekursiven Funktion ($f(n)$)**:**

$\mathcal{O}(1)$

Ergibt folgende Rekursionsgleichung:

$$T(n) = 3 \cdot T\left(\tfrac{n}{3}\right) + \mathcal{O}(1)$$

(b) Ordnen Sie die rekursive Funktion $T(n)$ aus (a) einem der drei Fälle des Mastertheorems zu und geben Sie die resultierende Zeitkomplexität an. Zeigen Sie dabei, dass die Voraussetzung des Falles erfüllt ist.

Lösungsvorschlag

1. Fall: $f(n) \in \mathcal{O}\left(n^{\log_b a - \varepsilon}\right)$:

$$f(n) \in \mathcal{O}\left(n^{\log_3 3 - \varepsilon}\right) = \mathcal{O}\left(n^{1-\varepsilon}\right) = \mathcal{O}(1) \text{ für } \varepsilon = 1$$

2. Fall: $f(n) \in \Theta\left(n^{\log_b a}\right)$:

$$f(n) \notin \Theta\left(n^{\log_3 3}\right) = \Theta\left(n^1\right)$$

3. Fall: $f(n) \in \Omega\left(n^{\log_b a + \varepsilon}\right)$:

$$f(n) \notin \Omega\left(n^{\log_3 3 + \varepsilon}\right) = \Omega\left(n^{1+\varepsilon}\right)$$

Also: $T(n) \in \Theta\left(n^{\log_b a}\right)$

Aufgabe 7 [Quicksort]

(a) Gegeben ist das folgende Array von Zahlen: $[23, 5, 4, 67, 30, 15, 25, 21]$.

Sortieren Sie das Array mittels Quicksort in-situ aufsteigend von links nach rechts. Geben Sie die (Teil-)Arrays nach jeder Swap-Operation (auch wenn Elemente mit sich selber getauscht werden) und am Anfang jedes Aufrufs der rekursiven Methode an. Verwenden Sie als Pivotelement jeweils das rechteste Element im Teilarray und markieren Sie dieses entsprechend. Teilarrays der Länge ≤ 2 dürfen im rekursiven Aufruf durch direkten Vergleich sortiert werden. Geben Sie am Ende das sortierte Array an.

(b) Welche Worst-Case-Laufzeit (O-Notation) hat Quicksort für n Elemente? Geben Sie ein Array mit fünf Elementen an, in welchem die Quicksort-Variante aus (a) diese Wort-Case-Laufzeit benötigt (ohne Begründung).

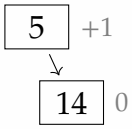
Aufgabe 8 [AVL-Baum 5,14,28,10,3,12,13]

Bearbeiten Sie folgende Aufgaben zu AVL-Suchbäumen. Geben Sie jeweils bei jeder einzelnen Operation zum Einfügen, Löschen, sowie jeder elementaren Operation zum Wiederherstellen der AVL-Baumeigenschaften den entstehenden Baum als Baumzeichnung an. Geben Sie zur Darstellung der elementaren Operation auch vorübergehend ungültige AVL-Bäume an und stellen Sie Doppelrotationen in zwei Schritten dar. Dabei sollen die durchgeführten Operationen klar gekennzeichnet sein und die Baumknoten immer mit aktuellen Balancewerten versehen sein.

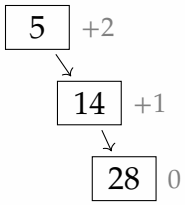
(a) Fügen Sie (manuell) nacheinander die Zahlen 5, 14, 28, 10, 3, 12, 13 in einen anfangs leeren AVL-Baum ein.

Einfügen von „5“:

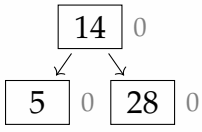
Einfügen von „14“:



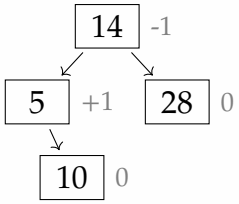
Einfügen von „28“:



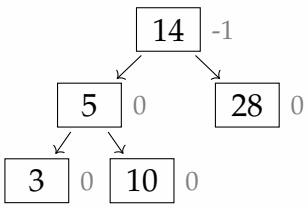
Linksrotation:



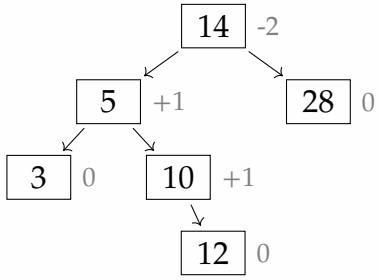
Einfügen von „10“:



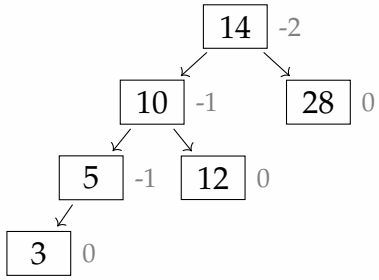
Einfügen von „3“:



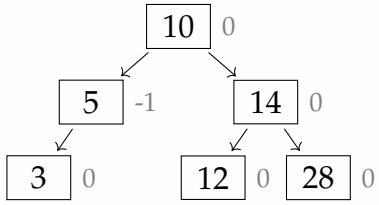
Einfügen von „12“:



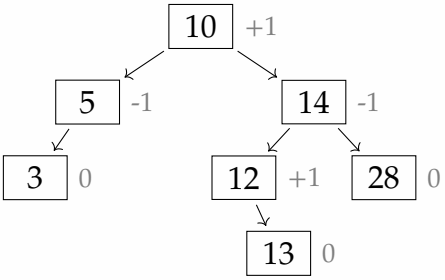
Linksrotation:



Rechtsrotation:

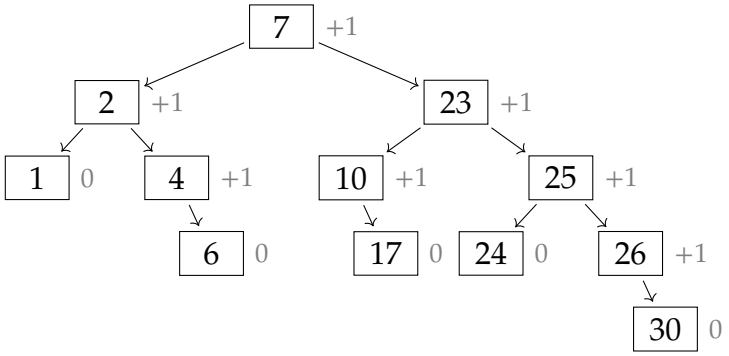


Einfügen von „13“:



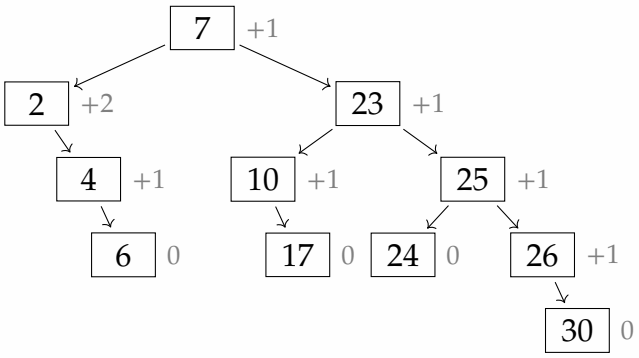
(b) Gegeben sei folgender AVL-Baum. Löschen Sie nacheinander die Knoten 1 und 23. Bei

Wahlmöglichkeiten nehmen Sie jeweils den kleineren Wert anstatt eines größeren.

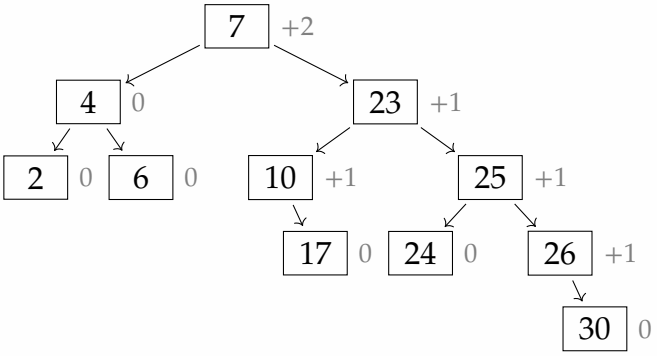


Lösungsvorschlag

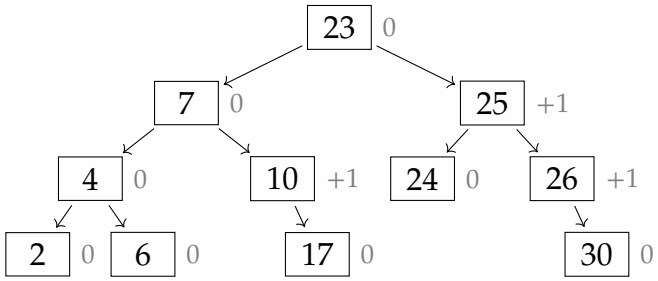
Löschen von „1“:



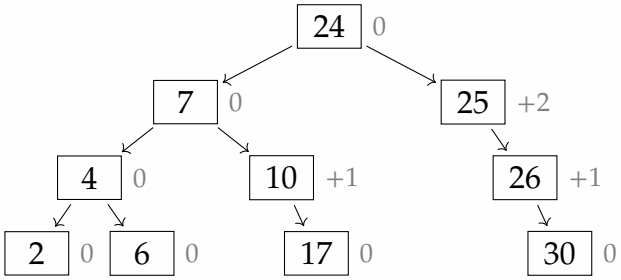
Linksrotation:



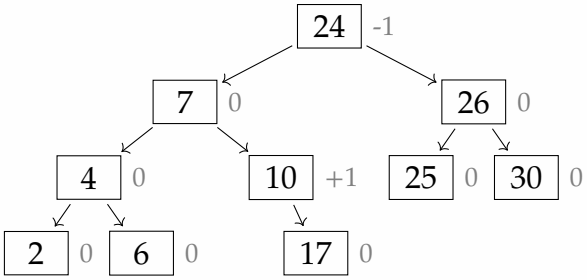
Linksrotation:



Löschen von „23“:

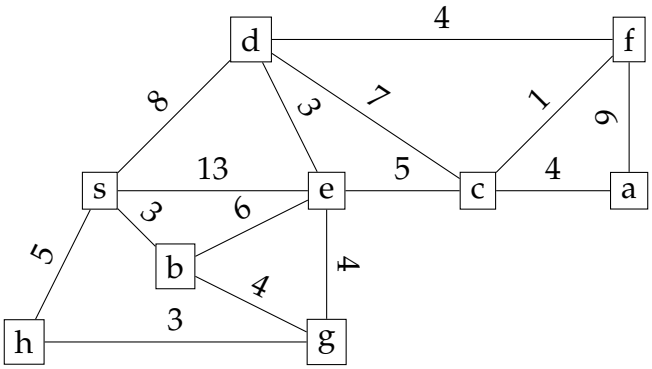


Linksrotation:



Aufgabe 9: [Negative Kantengewichte]

Gegeben sei folgender Graph G.

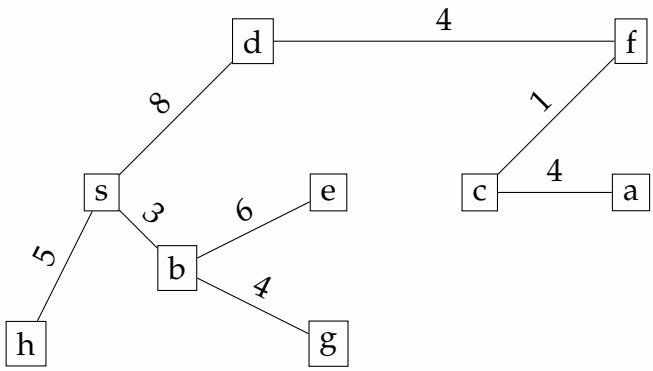


(a) Berechnen Sie mithilfe des Algorithmus von Dijkstra die kürzesten Wege vom Knoten s zu allen anderen Knoten im Graphen G. Erstellen Sie dazu eine Tabelle mit zwei Spalten

und stellen Sie jeden einzelnen Schritt des Verfahrens in einer eigenen Zeile dar. Geben Sie in der ersten Spalte den jeweils als nächstes fertigzustellenden Knoten v (wird sog. „schwarz“) als Tripel (v, p, δ) mit v als Knotenname, p als aktueller Vorgängerknoten und δ als aktuelle Distanz von s zu v über p an. Führen Sie in der zweiten Spalten alle anderen bisher erreichten Knoten v ebenfalls als Tripel (v, p, δ) auf, wobei diese sog. „grauen Randknoten“ in folgenden Durchgängen erneut betrachtet werden müssen. Zeichnen Sie anschließend den entstandenen Wegebaum, ðden Graphen G, in dem nur noch diejenigen Kanten vorkommen, die Teil der kürzesten Wege von s zu allen anderen Knoten sind.

Lösungsvorschlag

Nr	„schwarze“ Knoten	„graue“ Randknoten
1	**(s, -, 0)**	[(b, s, 3)] (d, s, 8) (e, s, 13) (h, s, 5)
2	**(b, s, 3)**	(d, s, 8) (e, **b**, **9**) (g, b, 7) [(h, s, 5)]
3	**(h, s, 5)**	(d, s, 8) (e, b, 9) [(g, b, 7)]
4	**(g, b, 7)**	[(d, s, 8)] (e, b, 9)
5	**(d, s, 8)**	(c, d, 15) [(e, b, 9)] (f, d, 12)
6	**(e, b, 9)**	(c, **e**, **14**) [(f, d, 12)]
7	**(f, d, 12)**	(a, f, 21) [(c, **f**, **13**)]
8	**(c, f, 13)**	[(a, **c**, **17**)]
9	**(a, c, 17)**	



Alternativer Lösungsweg

Lösungsvorschlag

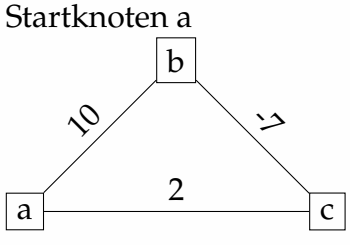
Nr.	besucht	a	b	c	d	e	f	g	h	s
0		∞	∞	∞	∞	∞	∞	∞	∞	0
1	s	∞	3	∞	8	13	∞	∞	5	**0**
2	b	∞	**3**	∞	8	9	∞	7	5	|
3	h	∞	|	∞	8	9	∞	7	**5**	|
4	g	∞	|	∞	8	9	∞	**7**	|	|
5	d	∞	|	15	**8**	9	12	|	|	|
6	e	∞	|	14	|	**9**	12	|	|	|
7	f	21	|	13	|	|	**12**	|	|	|
8	c	17	|	**13**	|	|	|	|	|	|
9	a	**17**	|	|	|	|	|	|	|	|

nach	Entfernung	Reihenfolge	Pfad
s → a	17	9	s → d → f → c → a
s → b	3	2	s → b
s → c	13	8	s → d → f → c
s → d	8	5	s → d
s → e	9	6	s → b → e
s → f	12	7	s → d → f
s → g	7	4	s → b → g
s → h	5	3	s → h
s → s	0	1	

(b) Der Dijkstra-Algorithmus liefert bekanntlich auf Graphen mit negativen Kantengewichten unter Umständen ein falsches Ergebnis.

(i) Geben Sie einen Graphen mit negativen Kantengewichten an, sodass der Dijkstra-Algorithmus ausgehend von einem von Ihnen ausgezeichneten Startknoten ein korrektes Ergebnis liefert.

Lösungsvorschlag



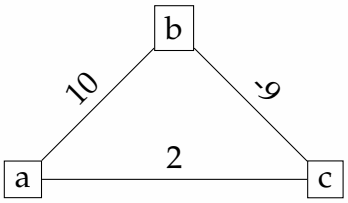
Nr.	besucht	a	b	c
0		0	∞	∞
1	a	**0**	10	2
2	c	|	10	**2**
3	b	|	**10**	|

Richtig: a - c: 2

(ii) Geben Sie einen Graphen mit negativen Kantengewichten an, sodass der Dijkstra-Algorithmus ausgehend von einem von Ihnen ausgezeichneten Startknoten ein falsches Ergebnis liefert.

Lösungsvorschlag

Startknoten a



Nr.	besucht	a	b	c
0		0	∞	∞
1	a	**0**	10	2
2	c	|	10	**2**
3	b	|	**10**	|

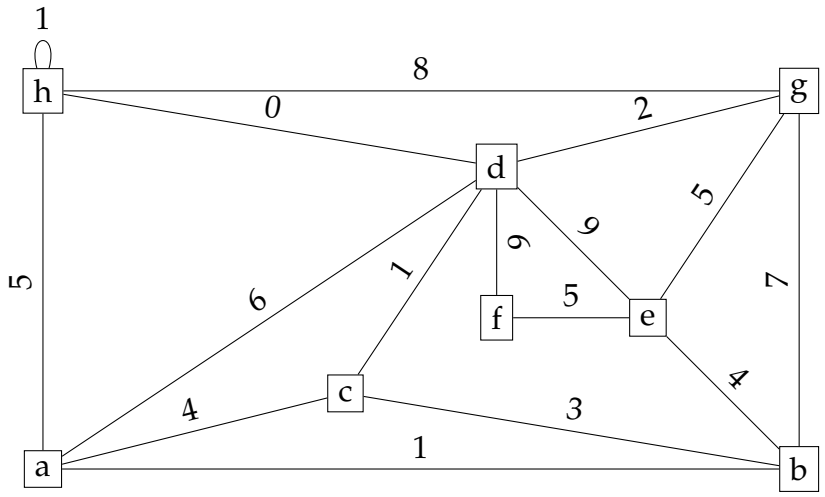
falsch: a - c: müsste 1 (10 - 9) sein.

Ein Beweis oder eine Begründung ist jeweils nicht erforderlich.

Aufgabe 10 [Graph a-h]

(a) Berechnen Sie mithilfe des Algorithmus von Prim ausgehend vom Knoten a einen minimalen Spannbaum des ungerichteten Graphen G, der durch folgende Adjazenzmatrix gegeben ist:

$$
\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{array}
\begin{array}{c}
\begin{array}{cccccccc} a & b & c & d & e & f & g & h \end{array} \\
\begin{pmatrix}
* & 1 & 4 & 6 & - & - & - & 5 \\
1 & * & 3 & - & 4 & - & 7 & - \\
4 & 3 & * & 1 & - & - & - & - \\
6 & - & 1 & * & 9 & 6 & 2 & 0 \\
- & 4 & - & 9 & * & 5 & 5 & - \\
- & - & - & 6 & 5 & * & - & - \\
- & 7 & - & 2 & 5 & - & * & 8 \\
5 & - & - & 0 & - & - & 8 & 1
\end{pmatrix}
\end{array}
$$

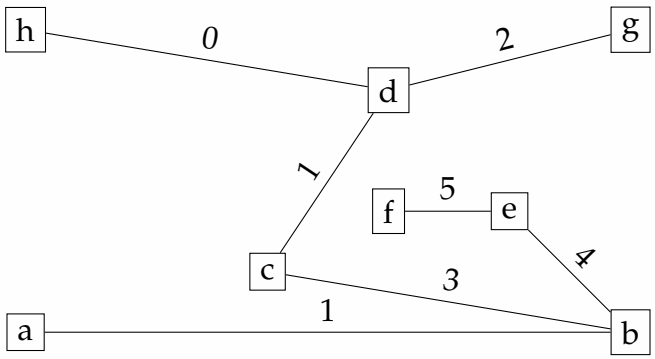


Erstellen Sie dazu eine Tabelle mit zwei Spalten und stellen Sie jeden einzelnen Schritt des Verfahrens in einer eigenen Zeile dar. Geben Sie in der ersten Spalte denjenigen Knoten v, der vom Algorithmus als nächstes in den Ergebnisbaum aufgenommen wird (dieser sog. „schwarze“ Knoten ist damit fertiggestellt), als Tripel (v, p, δ) mit v als Knotenname, p als aktueller Vorgängerknoten und δ als aktuelle Distanz von v zu p an. Führen Sie in der zweiten Spalte alle anderen vom aktuellen Spannbaum direkt erreichbaren Knoten v (sog. „graue Randknoten“) ebenfalls als Tripel (v, p, δ) auf.

Zeichnen Sie anschließend den entstandenen Spannbaum und geben sein Gewicht an.

Lösungsvorschlag

„schwarze“	„graue“ Randknoten
(a, NULL, ∞)	(b, a, 1) (c, a, 4) (h, a, 5) (d, a, 6)
(b, a, 1)	(c, b, 3) (e, b, 4) (h, a, 5) (d, a, 6) (g, b, 7)
(c, b, 3)	(d, c, 1) (e, b, 4) (h, a, 5) (g, b, 7)
(d, c, 1)	(h, d, 0) (g, d, 2) (e, b, 4) (f, d, 6)
(h, d, 0)	(g, d, 2) (e, b, 4) (f, d, 6)
(g, d, 2)	(e, b, 4) (f, d, 6)
(e, b, 4)	(f, e, 5)
(f, e, 5)	



Minimales Kantengewicht: 16

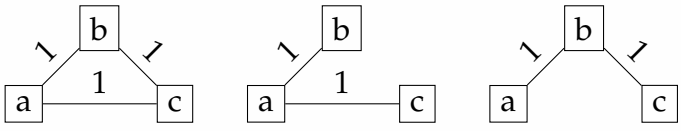
(b) Welche Worst-Case-Laufzeitkomplexität hat der Algorithmus von Prim, wenn die grauen Knoten in einem Heap (= Halde) nach Distanz verwaltet werden? Sei dabei n die Anzahl an Knoten und m die Anzahl an Kanten des Graphen. Eine Begründung ist nicht erforderlich.

Lösungsvorschlag

$\mathcal{O}(n \cdot \log(n) + m)$

(c) Zeigen Sie durch ein kleines Beispiel, dass ein minimaler Spannbaum eines ungerichteten Graphen nicht immer eindeutig ist.

Lösungsvorschlag



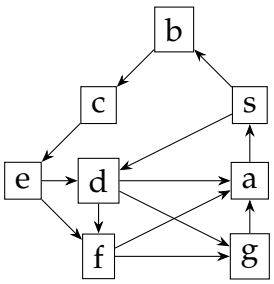
(d) Skizzieren Sie eine Methode, mit der ein maximaler Spannbaum mit einem beliebigen Algorithmus für minimale Spannbäume berechnet werden kann. In welcher Laufzeitkomplexität kann ein maximaler Spannbaum berechnet werden?

Lösungsvorschlag

Alle Kantengewichte negieren. In $\mathcal{O}(n \cdot \log(n) + m)$ wie der Algorithmus von Prim.

Aufgabe 11 [Graph a-g, Startknoten s]

Gegeben sei der folgende gerichtete Graph G:



Traversieren Sie G ausgehend vom Knoten s mittels

(a) Tiefensuche (DFS),

Lösungsvorschlag

Rekursiv ohne Keller:

0	1	2	3	4	5	6	7
s	b	c	e	d	a	f	g

(b) Breitensuche (BFS)

Lösungsvorschlag

mit Warteschlange:

0	1	2	3	4	5	6	7
s	b	d	c	a	f	g	e

und geben Sie jeweils die erhaltene Nummerierung der Knoten an. Besuchen Sie die Nachbarn eines Knotens bei Wahlmöglichkeiten immer in alphabetisch aufsteigender Reihenfolge.