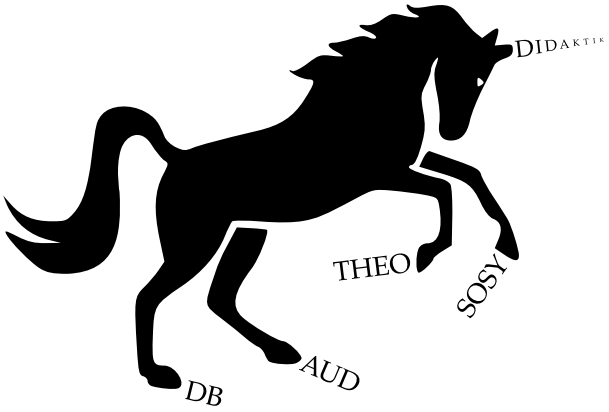
46115
Frühjahr
2018

Theoretische Informatik / Algorithmen / Datenstrukturen (nicht vertieft)

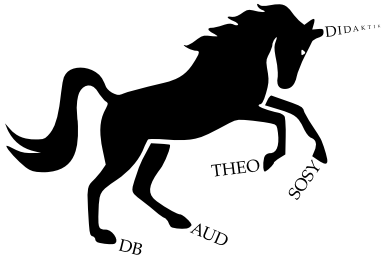
Aufgabenstellungen mit Lösungsvorschlägen



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

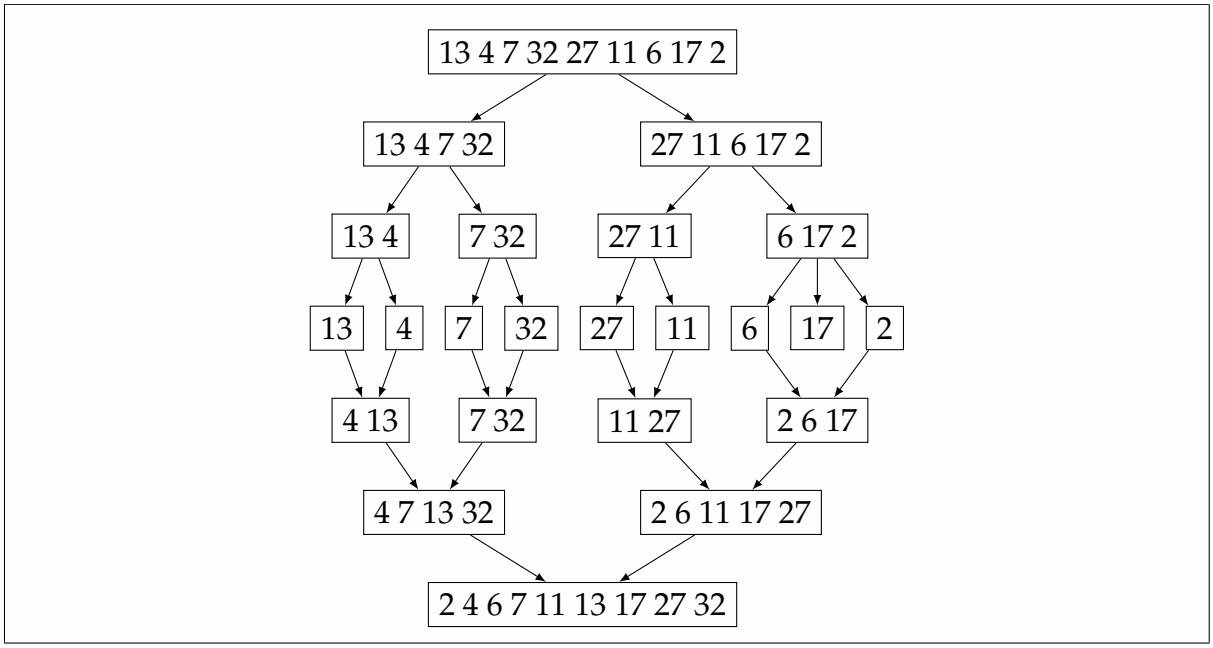
Aufgabe zum Mergesort

(a) Gegeben ist das folgende Array von Zahlen:

$$[13, 4, 7, 32, 27, 11, 6, 17, 2]$$

Sortieren Sie das Array mittels Mergesort aufsteigend von links nach rechts. Das Aufteilen einer Liste soll in der Mitte erfolgen und, falls notwendig, die zweite Liste ein Element länger sein als die erste. Listen der Länge zwei dürfen durch direkten Vergleich sortiert werden. Geben Sie die Eingabe und das Ergebnis jedes (rekursiven) Aufrufs an. Geben Sie abschließend die sortierte Liste an.

Lösungsvorschlag



(b) Beantworten Sie folgende Fragen jeweils ohne Begründung oder Beweis.

(i) Welche Worst-Case-Laufzeit ($\mathcal{O}$-Notation) hat Mergesort für n Elemente?

Lösungsvorschlag

$\mathcal{O}(n \cdot log(n))$ im Best-, Average- und Worst-Case

(ii) Welche Laufzeit hat Mergesort für n Elemente im Best-Case?

Lösungsvorschlag

$\mathcal{O}(n \cdot log(n))$ im Best-, Average- und Worst-Case

(iii) Kann basierend auf paarweisen Vergleichen von Werten schneller (Laufzeitkomplexität) als Mergesort sortiert werden?

Lösungsvorschlag

Nein. Es lässt sich beweisen, dass ein vergleichsbasiertes Sortierverfahren nicht schneller als $\Omega(n \cdot \log(n))$ sein kann. [a]

[a] https://de.wikipedia.org/wiki/Sortierverfahren

Aufgabe [Prim nach Adjazenzmatrix, Tripelnotation]

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

$$\begin{array}{c} \\ s \\ a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{array} \begin{array}{c} \begin{array}{ccccccccc} s & a & b & c & d & e & f & g & h \end{array} \\ \begin{pmatrix} * & - & 3 & - & - & 7 & - & - & - \\ - & * & - & 0 & 8 & - & 11 & - & - \\ 3 & - & * & - & 5 & - & 10 & - & - \\ - & 0 & - & * & - & - & 1 & - & - \\ - & 8 & 5 & - & * & 2 & 3 & - & 6 \\ 7 & - & - & - & 2 & * & - & - & 11 \\ - & 11 & 10 & 1 & 3 & - & * & 7 & - \\ - & - & - & - & - & - & 7 & * & 4 \\ - & - & - & - & 6 & 11 & - & 4 & * \end{pmatrix} \end{array}$$

(a) 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 Sie sein Gewicht an.

Lösungsvorschlag

Der Graph muss nicht gezeichnet werden. Der Algorithmus kann auch nur mit der Adjazenzmatrix durchgeführt werden. Möglicherweise geht das Lösen der Aufgabe schneller mit der Matrix von der Hand.

Kompletter Graph

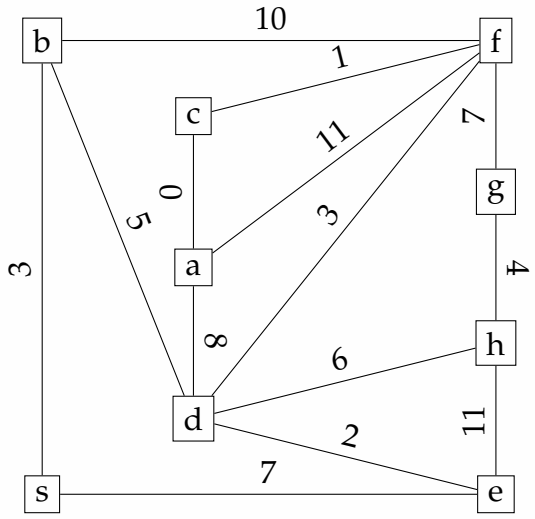
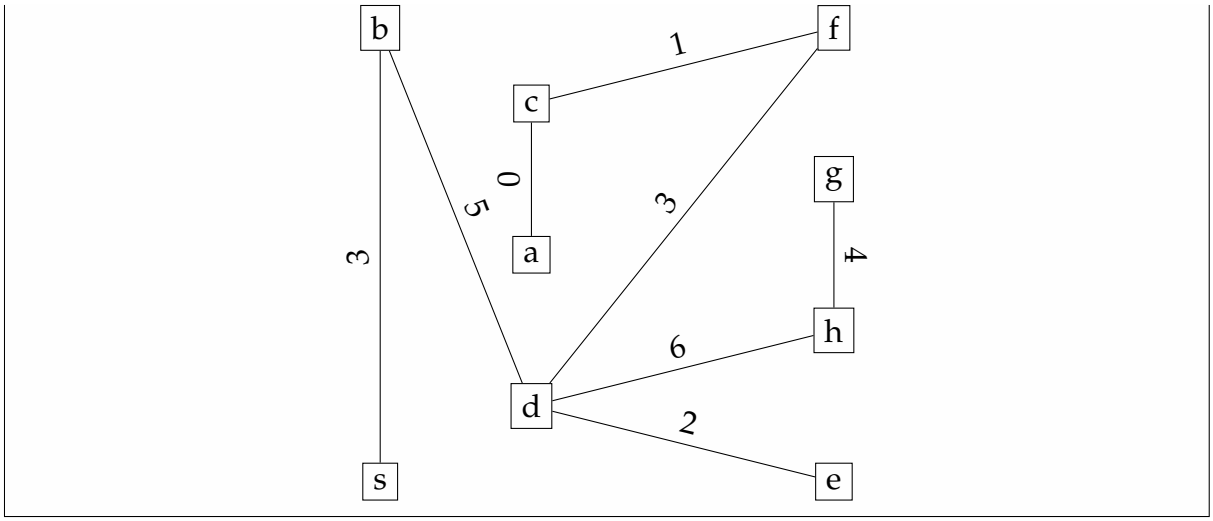


Tabelle schwarz-graue-Knoten

schwarze	**graue**
(s, null, -)	(b, s, 3); (e, s, 7);
(b, s, 3)	(d, b, 5); (e, s, 7); (f, b, 10);
(d, b, 5)	(a, d, 8); (e, d, 2); (f, d, 3); (h, d, 6);
(e, d, 2)	(a, d, 8); (f, d, 3); (h, d, 6);
(f, d, 3)	(a, d, 8); (c, f, 1); (g, f, 7); (h, d, 6);
(c, f, 1)	(a, c, 0); (g, f, 7); (h, d, 6);
(a, c, 0)	(g, f, 7); (h, d, 6);
(h, d, 6)	(g, h, 4);
(g, h, 4)	

Gewicht des minimalen Spannbaums: 24

Minimaler Spannbaum



(b) Welche Worst-Case-Laufzeitkomplexität hat der Algorithmus von Prim, wenn die grauen Knoten in einem Heap 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

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

(c) Beschreiben Sie kurz die Idee des alternativen Ansatzes zur Berechnung eines minimalen Spannbaumes von Kruskal.

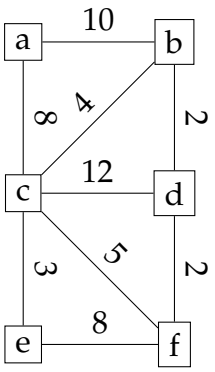
Lösungsvorschlag

Kruskal wählt nicht die kürzeste an einen Teilgraphen anschließende Kante, sondern global die kürzeste verbliebene aller Kanten, die keinen Zyklus bildet, ohne dass diese mit dem Teilgraph verbunden sein muss.

Thema Nr. 2

Aufgabe 4 [Graph a-f]

Sei G der folgende Graph.



(a) Der Algorithmus von Prim ist ein Algorithmus zur Bestimmung des minimalen Spannbaums in einem Graphen. Geben Sie einen anderen Algorithmus zur Bestimmung des minimalen Spannbaums an.

Lösungsvorschlag

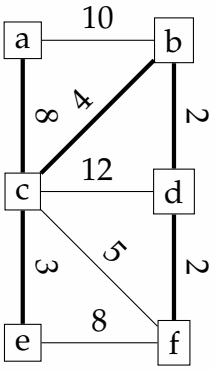
Zum Beispiel der Algorithmus von Kruskal

(b) Führen Sie den Algorithmus von Prim schrittweise auf G aus. Ausgangsknoten soll der Knoten a sein. Ihre Tabelle sollte wie folgt beginnen:

a	b	c	d	e	f	Warteschlange

Die Einträge der Tabelle geben an, wie weit der angegebene Knoten vom aktuellen Baum entfernt ist.

Lösungsvorschlag



a	b	c	d	e	f	Warteschlange
0	∞	∞	∞	∞	∞	a
0	10	8	∞	∞	∞	c, b
0	4	0	12	3	5	e, b, f, d
0	4	0	12	0	5	b, f, d
0	0	0	2	0	5	d, f
0	0	0	0	0	2	f
0	0	0	0	0	0	

(c) Erklären Sie, warum der Kürzeste-Wege-Baum (also das gezeichnete Ergebnis des Dijkstra-Algorithmus) und der minimale Spannbaum nicht notwendigerweise identisch sind.

Lösungsvorschlag

Die Wahl der nächsten Kante erfolgt nach völlig verschiedenen Kriterien:

- Beim Kürzeste-Wege-Baum orientiert sie sich an der Entfernung der einzelnen Knoten vom Startknoten.
- Beim Spannbaum orientiert sie sich an der Entfernung der einzelnen Knoten vom bereits erschlossenen Teil des Spannbaums.