Probeklausur: Lösungsvorschläge

Lösungen von Sebastian Kuhnert

Besprechung in den Übungen

Die Lösungen sind ohne Gewähr – Verbesserungsvorschläge sind willkommen.

Aufgabe 1 Beweisen oder widerlegen Sie. ***20 Punkte***

(a) $\sum_{i=1}^{n} \log i = \Theta(n \log n)$,

Lösung:

Die Aussage gilt. Abschätzung nach oben:

$$\sum_{i=1}^{n} \log i \leq \sum_{i=1}^{n} \log n = n \log n = \mathcal{O}(n \log n)$$

Abschätzung nach unten:

$$\sum_{i=1}^{n} \log i \geq \sum_{i=\lceil n/2 \rceil}^{n} \log i \geq \sum_{i=\lceil n/2 \rceil}^{n} \log \frac{n}{2} \geq \frac{n}{2} \log \frac{n}{2} = \frac{1}{2} n (\log n - \log 2) = \Omega(n \log n)$$

(b) $3^{\sqrt{n}} = 2^{o(n)}$,

Lösung:

Die Aussage gilt. Wegen $3^{\sqrt{n}} = 2^{\log_2(3^{\sqrt{n}})} = 2^{\sqrt{n} \cdot \log_2 3}$ genügt es zu zeigen, dass $\sqrt{n} \cdot \log_2 3 = o(n)$. Dies folgt aus

$$\lim_{n \to \infty} \frac{\sqrt{n} \cdot \log_2 3}{n} = \lim_{n \to \infty} \frac{\log_2 3}{\sqrt{n}} = 0.$$

(c) $n - \sqrt{n} - \log n = o(n)$,

Lösung:

Die Aussage ist falsch, da

$$\lim_{n \to \infty} \frac{n - \sqrt{n} - \log n}{n} = \lim_{n \to \infty} 1 - \frac{1}{\sqrt{n}} - \frac{\log n}{n} = 1 - 0 - 0 = 1 > 0.$$

(d) $n / \log \log n = o(n)$.

Lösung:

Die Aussage gilt:

$$\lim_{n \to \infty} \frac{\frac{n}{\log \log n}}{n} = \lim_{n \to \infty} \frac{1}{\log \log n} = 0$$

Aufgabe 2 Gegeben sei das Muster $y = \mathsf{ACACAGACACCA}$. ***15 Punkte***

(a) Konstruieren Sie den DFA M_y.

Lösung:

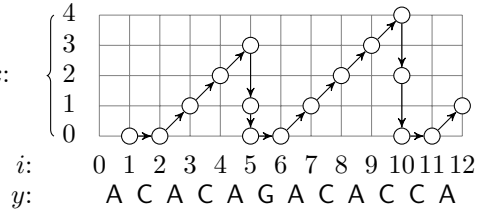
Es wird der DFA mit der folgenden Überführungsfunktion δ generiert:

	0	1	2	3	4	5	6	7	8	9	10	11	12
A	1	1	3	1	5	1	7	1	9	1	5	12	1
C	0	2	0	4	0	4	0	8	0	10	11	0	2
G	0	0	0	0	0	6	0	0	0	0	0	0	0

(b) Geben Sie das Ablaufprotokoll von `KMP-Prefix`(y) und die Präfixfunktion π an.

Lösung:

Es ergibt sich folgender Ablauf:



Damit ergibt sich:

i	0	1	2	3	4	5	6	7	8	9	10	11	12
y_i		A	C	A	C	A	G	A	C	A	C	C	A
$\pi(i)$	0	0	0	1	2	3	0	1	2	3	4	0	1

Aufgabe 3 ***25 Punkte***

(a) Geben Sie einen Algorithmus an, der das folgende Entscheidungsproblem in Zeit $\mathcal{O}(n \log n)$ löst:

Gegeben: Zwei Folgen $(a_1, \ldots, a_m), (a_{m+1}, \ldots, a_n)$ mit $a_i \in \mathbb{N}$.

Gefragt: Ist $\{a_1, \ldots, a_m\} \subseteq \{a_{m+1}, \ldots, a_n\}$?

Lösung:

```
sortiere die Liste (a_{m+1}, ..., a_n)
for i ← 1 to m do
   führe Binärsuche von a_i in der sortierten Liste durch
   if nicht gefunden then verwirf
akzeptiere
```

(b) Begründen Sie die Laufzeit und Korrektheit Ihres Algorithmus'.

Lösung:

Sortieren ist z. B. mit `HeapSort` in $\mathcal{O}(n \log n)$ möglich. Die Schleife wird $m \leq n$ mal ausgeführt und eine Binärsuche dauert $\log(n-m) \leq \log n$ viele Schritte, sodass auch die Schleife nur $\mathcal{O}(n \log n)$ Schritte benötigt.

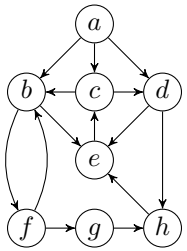
Nein-Instanzen werden verworfen, weil es mindestens ein a_i gibt, das nicht in der Liste $(a_{m+1}, \ldots, a_n)$ enthalten ist. Ja-Instanzen werden nicht verworfen, weil für jedes a_i $(1 \leq i \leq m)$ ein a_j $(m+1 \leq j \leq n)$ mit $a_i = a_j$ existiert und dieses durch Binärsuche auf der sortierten Liste sicher gefunden wird.

(c) Schätzen Sie die asymptotische Laufzeit eines beliebigen vergleichsbasierten Algorithmus' für obiges Entscheidungsproblem ab, der nur ja-nein-Fragen der Form $a_i \leq a_j$ stellen darf.

Lösung:

In (a) wurde bereits ein Algorithmus mit $\mathcal{O}(n \log n)$ Vergleichen vorgestellt, wir zeigen nun die untere Schranke $\Omega(n \log n)$. Hierfür beschränken uns auf Instanzen mit $n = 2m$, in denen in beiden Listen keine Zahl doppelt vorkommt. Die grundlegende Beobachtung ist, dass jeder Algorithmus eine Zuordnung $f\colon \{1, \ldots, m\} \to \{m+1, \ldots, n\}$ der Elemente der ersten Liste zu denen der zweiten Liste finden muss mit $a_i = a_{f(i)}$. Ordnet er eines nicht zu, kann er nicht zwischen Instanzen unterscheiden, in denen dieses Element ausgetauscht wird. Unter diesen Instanzen sind sowohl Ja- als auch Nein-Instanzen, sodass der Algorithmus nicht korrekt arbeiten würde. Da es $m!$ verschiedene solche Zuordnungen gibt, muss der Entscheidungsbaum jedes korrekten Algorithmus für dieses Problem mindestens $m!$ Blätter haben. Damit muss der Entscheidungsbaum mindestens $\Omega(m \log m) = \Omega(n \log n)$ tief sein, was eine untere Schranke für die Anzahl der Vergleiche im Worst-Case ergibt. (Mit Aufgabe 17 gilt die unter Schranke auch für den Average-Case.)

Aufgabe 4 Gegeben sei folgender Digraph G. ***15 Punkte***



(a) Geben Sie für jeden Knoten v von G einen Kreis minimaler Länge an, der v enthält (sofern v auf einem Kreis liegt).

(b) Entwerfen Sie einen Linearzeit-Algorithmus, der für einen gegebenen Digraphen G und Knoten v feststellt, ob v auf einem Kreis liegt, und ggf. einen v enthaltenden Kreis minimaler Länge berechnet. Begründen Sie sowohl die Laufzeit als auch die Korrektheit Ihres Algorithmus.

Hinweis: Verwenden Sie Breitensuche.

Lösung:

Zu (a) Kürzeste Kreise:

- b und f liegen auf dem Kreis (b, f, b) der Länge 2.
- c, d und e liegen auf dem Kreis (c, d, e, c) der Länge 3.
- h liegt auf dem Kreis (h, e, c, d, h) der Länge 4.
- g liegt auf dem Kreis (g, h, e, c, b, f, g) der Länge 6.
- a liegt auf keinem Kreis.

Zu (b) Der Algorithmus startet eine Breitensuche in v. Wird dabei eine Kante mit Zielknoten v gefunden, wurde ein Kreis gefunden, der ausgegeben wird.

Die Laufzeit ist linear, weil Breitensuche in Linearzeit möglich ist.

Zur Korrektheit: Es ist klar, dass wenn v auf einem Kreis liegt, dieser auch gefunden wird. Es muss nur noch gezeigt werden, dass der ausgegebene Kreis immer ein kürzester Kreis durch v ist. Dies folgt aus der einfachen Beobachtung, dass Kreise durch v mit Länge k gefunden werden, während die $(k-1)$-te Schicht des Breitensuchbaums aufgebaut wird.

Aufgabe 5 Beweisen oder widerlegen Sie. ***20 Punkte***

(a) Ein azyklischer Graph mit $n \geq 2$ Knoten hat mindestens zwei Blätter.

Lösung:

Die Aussage gilt, denn jede Zusammenhangskomponente ist entweder ein isolierter Knoten oder hat mindestens zwei Blätter. Angenommen, eine Zusammenhangskomponente hat weniger als zwei Blätter, d. h. höchstens ein Knoten hat Grad 1 und alle übrigen mindestens Grad 2 (Grad 0 kann nicht vorkommen, da zusammenhängend und mindestens 2 Knoten). Starte in dem Knoten mit kleinstem Grad und baue schrittweise einen Weg auf, der niemals die gleiche Kante zweimal in Folge besucht. Dieser Weg kann unendlich lang verlängert werden, da alle bis auf den ersten Knoten mindestens Grad 2 haben. Da es nur endlich viele Knoten gibt, muss dieser Weg einen Kreis enthalten.

(b) Ein azyklischer Digraph mit $n \geq 2$ Knoten hat mindestens zwei Blätter.

Lösung:

Die Aussage gilt nicht. Gegenbeispiel: Der Graph •⟶• hat nur ein Blatt (d. h. Knoten mit Ausgangsgrad 0).

(c) Ein azyklischer Digraph mit $n \geq 2$ Knoten hat mindestens ein Blatt.

Lösung:

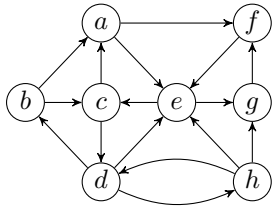
Die Aussage gilt. Starte in einem beliebigen Knoten. Solange der aktuelle Knoten kein Blatt ist, gehe zu einem beliebigen Nachfolger. Da der Graph azyklisch ist, kann jeder Knoten dabei nur einmal besucht werden und der Prozess endet nach endlich vielen Schritten in einem Blatt.

(d) Ein stark zusammenhängender Digraph mit $n \geq 2$ Knoten hat $m \geq n$ Kanten.

Lösung:

Die Aussage gilt. In jeden Knoten muss mindestens eine Kante münden, da sonst der Knoten von den übrigen Knoten aus nicht erreichbar sein kann.

Aufgabe 6 Gegeben sei der folgende Digraph G. ***20 Punkte***



(a) In welcher Reihenfolge werden die Knoten bei einer Tiefen- bzw. Breitensuche mit Startknoten a jeweils zum ersten und letzten Mal besucht? Bei Wahlmöglichkeit sollen die Nachbarn in alphabetischer Folge besucht werden.

(b) Geben Sie die zugehörigen Tiefen- und Breitensuchbäume mit allen Vorwärts-, Rückwärts- und Querkanten an.

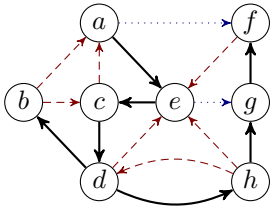
Lösung:

zu (a) Reihenfolge der Besuche:

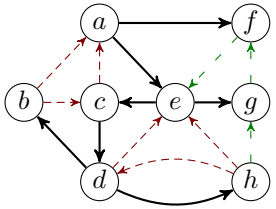
Erste Besuche DFS:	a, e, c, d, b, h, g, f
Letzte Besuche DFS:	b, f, g, h, d, c, e, a
Erste und letzte Besuche BFS:	a, e, f, c, g, d, b, h

zu (b) Baumkanten: ⟶
Vorwärtskanten: ⟶
Rückwärtskanten: ⟶
Querkanten: ⟶

Tiefensuchbaum:

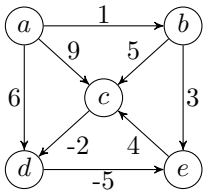


Breitensuchbaum:



Aufgabe 7 ***25 Punkte***

Gegeben sei folgender Digraph G mit Kostenfunktion ℓ.



(a) Bestimmen Sie mit dem Bellman-Ford-Moore-Algorithmus kürzeste Wege von a zu allen anderen Knoten.

(b) Bestimmen Sie mit dem Floyd-Warshall-Algorithmus kürzeste Wege zwischen allen Knoten.

Lösung:

zu (a) Ablaufprotokoll mit Queue-Einträgen (i, u, v, g) für einen Knoten u, der in Runde i von $v = \texttt{parent}(u)$ mit Entfernung g zum Startknoten erreicht wurde:

$$
\begin{array}{l|l|l|l|l|l}
(0,a,\texttt{nil},0) & & & & & \\
 & (1,b,a,1) & & & & \\
 & (1,c,a,9) & (1,c,b,6) & & & \\
 & (1,d,a,6) & (1,d,a,6) & (1,d,c,4) & & \\
 & & (2,e,b,4) & (2,e,b,4) & (2,e,d,-1) & \rightarrow \\
\rightarrow & (3,c,e,3) & & & & \\
 & & (4,d,c,1) & & & \\
 & & & (5,e,d,-4) & &
\end{array}
$$

An dieser Stelle bricht der Algorithmus wegen $i = 5 = n$ ab: Er hat den negativen Kreis gefunden.

zu (b)

d_0	a	b	c	d	e
a	∞	1	9	6	∞
b	∞	∞	5	∞	3
c	∞	∞	∞	-2	∞
d	∞	∞	∞	∞	-5
e	∞	∞	4	∞	∞

d_1	a	b	c	d	e
a	∞	1	9	6	∞
b	∞	∞	5	∞	3
c	∞	∞	∞	-2	∞
d	∞	∞	∞	∞	-5
e	∞	∞	4	∞	∞

d_2	a	b	c	d	e
a	∞	1	**6**	6	**4**
b	∞	∞	5	∞	3
c	∞	∞	∞	-2	∞
d	∞	∞	∞	∞	-5
e	∞	∞	4	∞	∞

d_3	a	b	c	d	e
a	∞	1	6	**4**	4
b	∞	∞	5	**3**	3
c	∞	∞	∞	-2	∞
d	∞	∞	∞	∞	-5
e	∞	∞	4	**2**	∞

d_4	a	b	c	d	e
a	∞	1	6	4	$\mathbf{-1}$
b	∞	∞	5	3	$\mathbf{-2}$
c	∞	∞	∞	-2	$\mathbf{-7}$
d	∞	∞	∞	∞	-5
e	∞	∞	4	2	$\mathbf{-3}$

d_5	a	b	c	d	e
a	∞	1	**3**	**1**	$\mathbf{-4}$
b	∞	∞	**2**	**0**	$\mathbf{-5}$
c	∞	∞	$\mathbf{-3}$	$\mathbf{-5}$	$\mathbf{-10}$
d	∞	∞	$\mathbf{-1}$	$\mathbf{-3}$	$\mathbf{-8}$
e	∞	∞	**1**	$\mathbf{-1}$	$\mathbf{-6}$

Da es auf der Diagonale negative Einträge gibt, ist die Ausgabe des Algorithmus »Negative Kreise«.

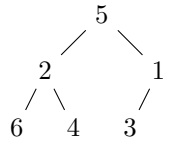
Aufgabe 8 ***30 Punkte***

Betrachten Sie die Eingabefolge $a = (5, 2, 1, 6, 4, 3)$.

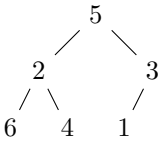
(a) Sortieren Sie die Folge a mit `HeapSort`. Geben Sie alle Zwischenergebnisse und die Anzahl der Vergleiche an.

Lösung:

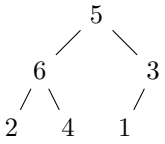
Anfangszustand:



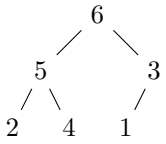
nach `Heapify` auf 1 (1 Vergleich):



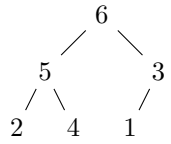
nach `Heapify` auf 2 (2 Vergleiche):



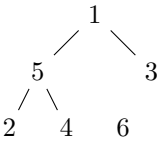
nach `Heapify` auf 5 (2 Vergleiche):



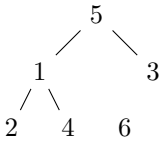
nach `Heapify` auf 5 (2 Vergleiche):



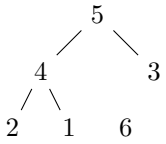
nach Heruntertauschen der Wurzel:



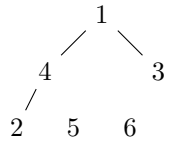
nach `Heapify` auf 1 (2 Vergleiche):



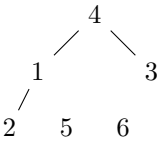
nach `Heapify` auf 1 (2 Vergleiche):



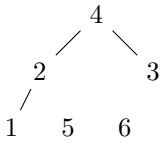
nach Heruntertauschen der Wurzel:



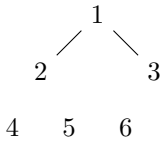
nach `Heapify` auf 1 (2 Vergleiche):



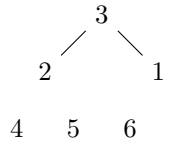
nach `Heapify` auf 1 (1 Vergleich):



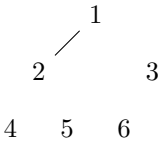
nach Heruntertauschen der Wurzel:



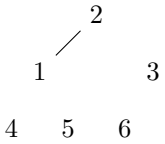
nach `Heapify` auf 1 (2 Vergleiche):



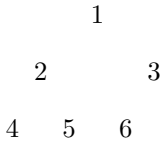
nach Heruntertauschen der Wurzel:



nach `Heapify` auf 1 (1 Vergleich):



nach Heruntertauschen der Wurzel:



Insgesamt kommt es zu 17 Vergleichen.

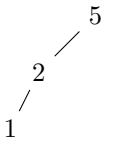
(b) Geben Sie den zur Einfügesequenz a gehörigen AVL-Baum an. Geben sie alle Zwischenergebnisse und die durchgeführten Rotationen an.

Lösung:

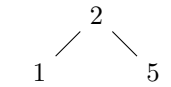
5 einfügen:

2 einfügen:

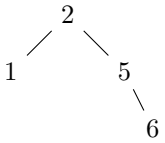
1 einfügen:



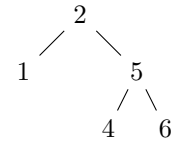
`RightRotate` an 5:



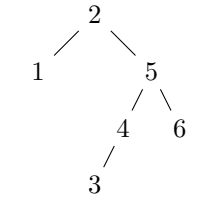
6 einfügen:



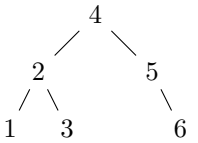
4 einfügen:



3 einfügen:



`RightLeftRotate` an 2:



(c) Betrachten Sie die Sortieralgorithmen `ST-Sort` und `AVL-Sort`, die die Glieder der Eingabefolge der Reihe nach in einen Suchbaum (bzw. AVL-Baum) einfügen und anschließend in sortierter Reihenfolge ausgeben. Wie viele Vergleiche benötigt `AVL-Sort` zum Sortieren der Folge a? Begründen Sie.

Lösung:

Vergleiche der Folgeglieder finden nur beim Einfügen statt (beim Rotieren werden nur Tiefendifferenzen verglichen). Die Zahl der Vergleiche beim Einfügen einer Zahl entspricht gerade der Tiefe des neuen Knotens im Baum (vor etwaigen Rotationen). Diese lässt sich in der Lösung von (b) leicht nachvollziehen:

- Einfügen der 5: 0 Vergleiche
- Einfügen der 2: 1 Vergleich
- Einfügen der 1: 2 Vergleiche
- Einfügen der 6: 2 Vergleiche
- Einfügen der 4: 2 Vergleiche
- Einfügen der 3: 3 Vergleiche

Zusammen ergeben sich also 10 Vergleiche.

(d) Geben Sie möglichst enge obere und untere asymptotische Schranken für den besten und schlechtesten Fall für `ST-Sort` und `AVL-Sort` an. Begründen Sie.

Lösung:

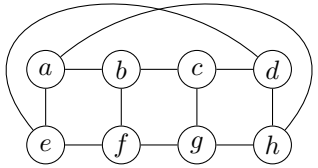
Ein Suchbaum mit n Einträgen kann höchstens Tiefe n haben, sodass für `ST-Sort` höchstens $\sum_{i=1}^{n}(i-1) = \mathcal{O}(n^2)$ Vergleiche benötigt werden. Für eine bereits sortierte Eingabefolge tritt dieser Fall auch ein.

Untere Schranke für `ST-Sort`: Minimal wird die Tiefensumme für einen Heapförmigen Suchbaum, dort beträgt sie $\sum_{i=1}^{n} \log i = \Omega(n \log n)$.
Ein AVL-Baum mit n Einträgen hat nach Vorlesung Tiefe $\Theta(\log n)$, sodass sich für `AVL-Sort` $\sum_{i=1}^{n} \Theta(\log i) = \Theta(n \log n)$ Vergleiche ergeben.

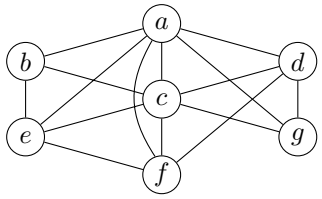
Aufgabe 9 ***10 Punkte***

Welche der beiden folgenden Graphen sind planar und welche nicht. Begründen Sie.

G :



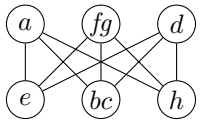
H :



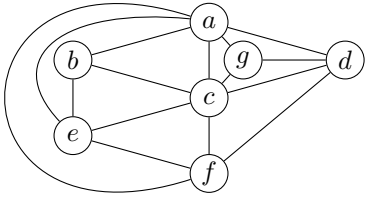
Lösung:

- Der Graph G ist nicht planar, weil er den $K_{3,3}$ als Minor enthält: Verschmelze die Knoten b und c sowie die Knoten f und g. Die eine Partition ist $\{a, fg, d\}$, die andere $\{e, bc, h\}$:

G' :



H :



- Der Graph H ist planar. Dies sieht man zum Beispiel, indem man die Kanten $\{a, e\}$ und $\{a, f\}$ links am restlichen Graphen vorbei zeichnet und den Knoten g in das Dreieck (a, c, d) verschiebt.

Als Bearbeitungszeit sind 180 min (also 1 min pro Punkt) vorgesehen.