Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen
Dr. Hanjo Täubig
Tobias Lieber

Sommersemester 2011
Abschlussklausur
5. August 2011

Grundlagen: Algorithmen und Datenstrukturen

Name	Vorname	Studiengang	Matrikelnummer
. . . . .	. . . . .	☐ Bachelor ☐ Inform. ☐ Master ☐ Mathe. ☐ Lehramt ☐ Bio-Inf.	. . . . .

Hörsaal	Reihe	Sitzplatz	Unterschrift
. . . . .	. . . . .	. . . . .	. . . . .

Allgemeine Hinweise zur Klausur

- Bitte füllen Sie obige Felder in Druckbuchstaben aus und unterschreiben Sie!
- Schreiben Sie nicht in *roter* oder *grüner* Farbe bzw. mit Bleistift!
- Außer Ihrem Schreibgerät und einem handbeschriebenen DIN-A4-Blatt sind keine weiteren Hilfsmittel erlaubt.
- Die Bearbeitungszeit beträgt 75 Minuten.
- Alle Antworten sind in die geheftete Angabe auf den jeweiligen Seiten (bzw. Rückseiten) der betreffenden Aufgaben einzutragen. Begründen Sie alle Ihre Schritte. Auf dem Schmierblatt können Sie Nebenrechnungen machen. Der Schmierblattbogen muss ebenfalls abgegeben werden, wird aber in der Regel nicht bewertet.

Hörsaal verlassen von . . . . . bis . . . . . / von . . . . . bis . . . . .

Vorzeitig abgegeben um . . . . .

Besondere Bemerkungen:

	A1	A2	A3	A4	A5	Σ	Korrektor
Erstkorrektur							
Zweitkorrektur							

Aufgabe 1 (9 Punkte)

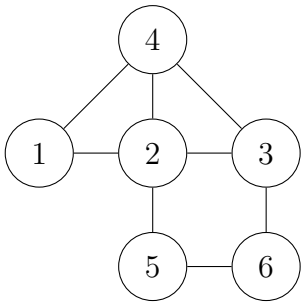
Bearbeiten Sie die folgenden Aufgaben:

1) Welche Voraussetzung muss ein kantengewichteter Graph $G = (V, E)$ erfüllen, damit der Dijkstra-Algorithmus die kürzesten Wege von einem Knoten $s \in V$ zu allen anderen Knoten korrekt berechnet?

Lösungsvorschlag

Die Kantengewichte müssen positiv sein.

2) Geben Sie für den folgenden Graphen die Adjazenzmatrix an:



Lösungsvorschlag

0	1	0	1	0	0
1	0	1	1	1	0
0	1	0	1	0	1
1	1	1	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0

3) Wir betrachten die exakten Laufzeiten von zwei Algorithmen auf Instanzen der Eingabegröße $n \in \mathbb{N}$:

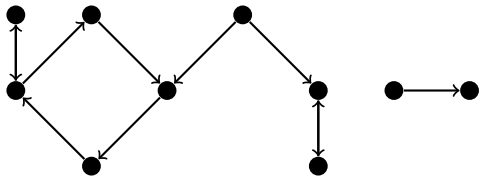
- Algorithmus 1 verwendet $4 \cdot n$ Elementaroperationen.
- Algorithmus 2 verwendet $n\lceil \log \log \log n \rceil$ Elementaroperationen.

Für welche Probleminstanzgrößen würden Sie Algorithmus 1 bevorzugen, für welche Algorithmus 2?

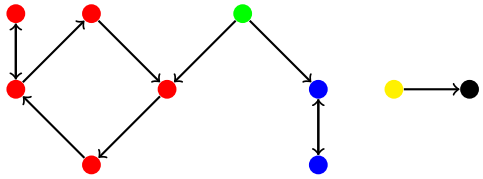
Lösungsvorschlag

Wenn $n \geq 2^{2^{2^4}}$ ist Algorithmus 1 effizienter. Sonst Algorithmus 2.

4) Zeichnen Sie die starken Zusammenhangskomponenten des gegebenen gerichteten Graphen in den Graphen ein.



Lösungsvorschlag



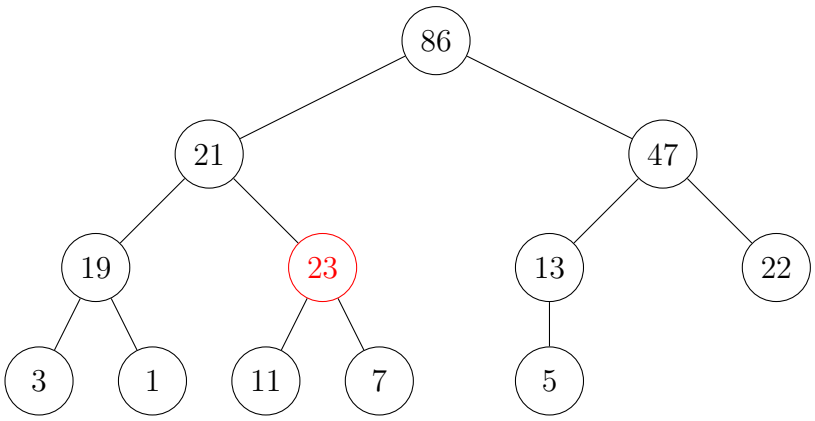
5) Sei $c > 0$. Geben Sie eine obere Schranke für die Anzahl von Split-/Merge-Operationen an, die ein (a, b)-Baum für eine Folge von n insert/remove Operationen benötigt, wenn für den (a, b)-Baum $2a + c \leq b$ gilt.

Lösungsvorschlag

Nach Vorlesung ist die Anzahl der Splitoperationen durch $\mathcal{O}(n)$ beschränkt, sobald $2a \leq b$ gilt.

6) Gegeben sei das Array $(86, 21, 47, 19, 23, 13, 22, 3, 1, 11, 7, 5)$. Repräsentiert das Array einen binären Max-Heap? Begründen Sie Ihre Antwort.

Lösungsvorschlag



7) Welche Eigenschaft von (a, b)-Bäumen garantiert die logarithmische Laufzeit der `insert`- und der `delete`-Operation und worin liegt sie begründet?

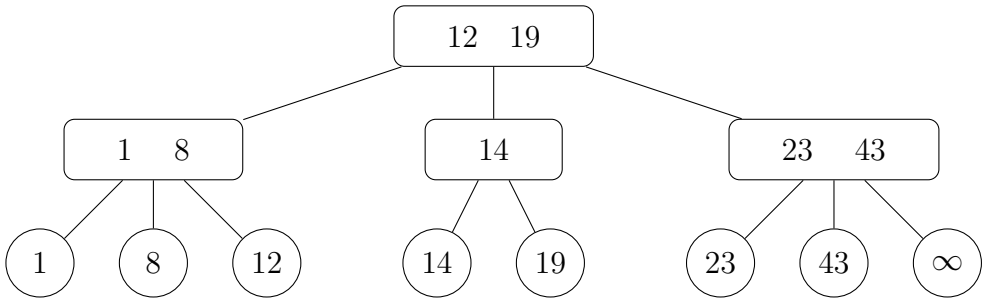
Lösungsvorschlag

(a) Die Höhe des Suchbaums ist logarithmisch in der Anzahl der Knoten.

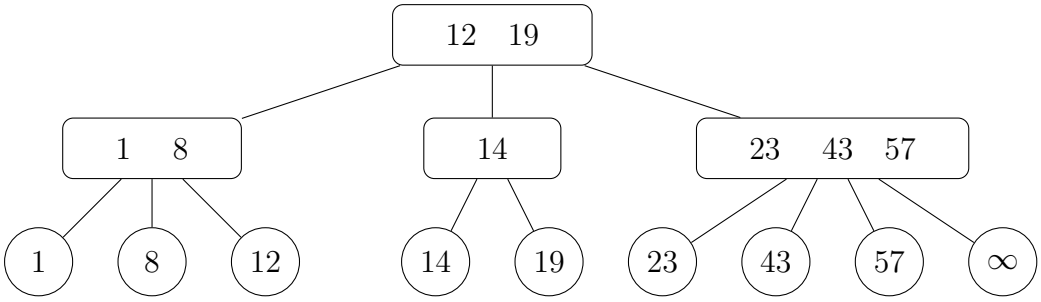
(b) Und zwar weil der Grad von jedem Knoten größer gleich 2 ist,

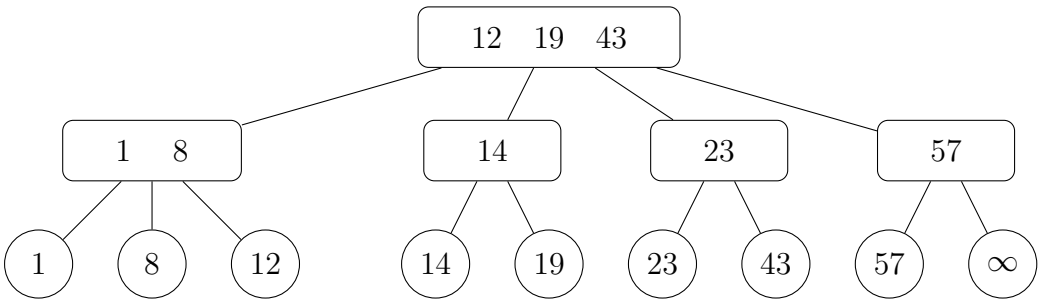
(c) und alle Blätter die selben Tiefe haben.

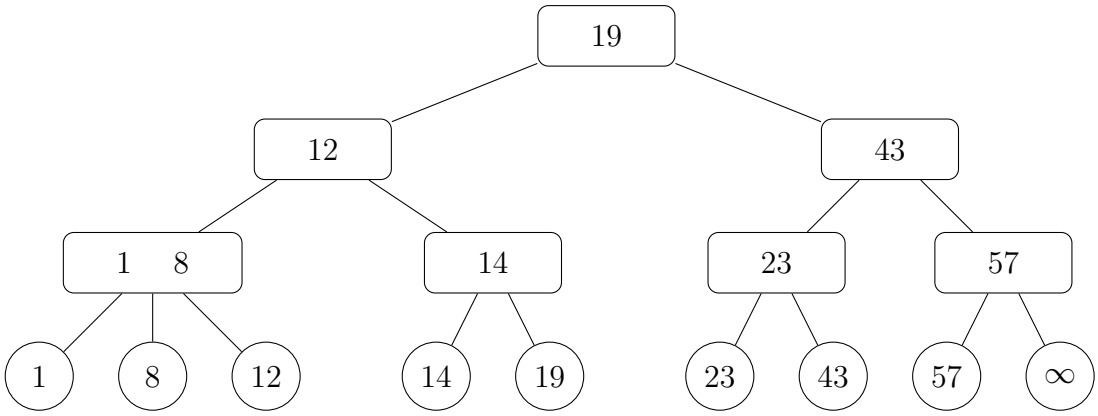
8+9) Fügen Sie in den angegebenen $(2, 3)$-Baum das Element mit dem Schlüssel 57 ein und zeichnen Sie die Zwischenschritte und das Resultat.



Lösungsvorschlag




12 19 43
1 8
14
23
57
1
8
12
14
19
23
43
57
∞


19
12
43
1 8
14
23
57
1
8
12
14
19
23
43
57
∞

Aufgabe 2 (8 Punkte)

a) Wir betrachten den gegebenen Vektor $M := (50, 53, 6, 43, 61, 85, 72, 90, 21, 30, 10, 23, 19)$. Erzeugen Sie aus dem Vektor M einen Min-Binomial-Heap H. Verwenden Sie hierzu den in den Übungen vorgestellten Algorithmus, der in linearer Zeit abläuft. Sortieren Sie die Vektoren *nicht*, sondern verwenden Sie die Elemente in der gegebenen Reihenfolge.

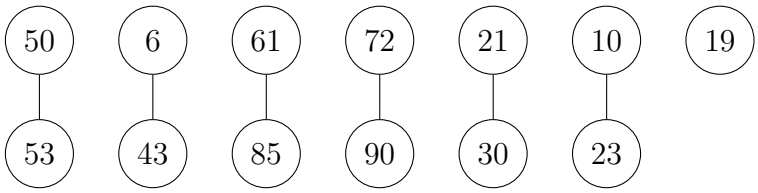
b) Fügen Sie nun in den Heap H zuerst ein Element mit der Priorität 45 und dann ein Element mit der Priorität 7 ein. Zeichnen Sie den resultierenden Heap H.

c) Führen Sie auf dem Heap H eine `deleteMin` Operation aus. Wenn Sie Bäume vereinigen müssen und die Vereinigung nicht eindeutig ist, so vereinigen Sie immer die Bäume mit den größten Wurzelelementen zuerst. Zeichnen Sie den resultierenden Heap H.

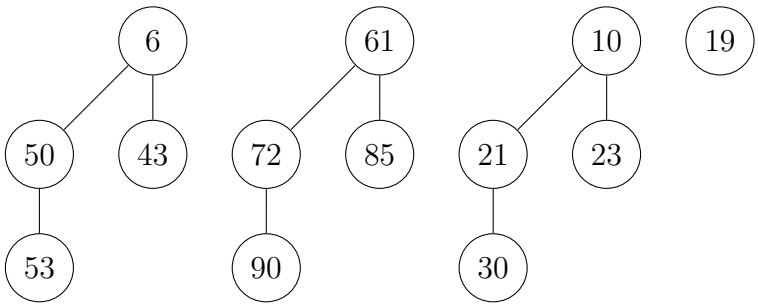
d) Gegeben seien drei Heaps mit 456, 302 bzw. 289 Elementen. Geben Sie an, welche Ränge die in den Heaps enthaltenen Binomialbäume besitzen. Werden die drei Heaps zu einem Heap vereinigt, so entsteht ein neuer Heap. Berechnen Sie auch für den resultierenden Heap die Ränge der im Heap enthaltenen Binomialbäume.

Lösungsvorschlag

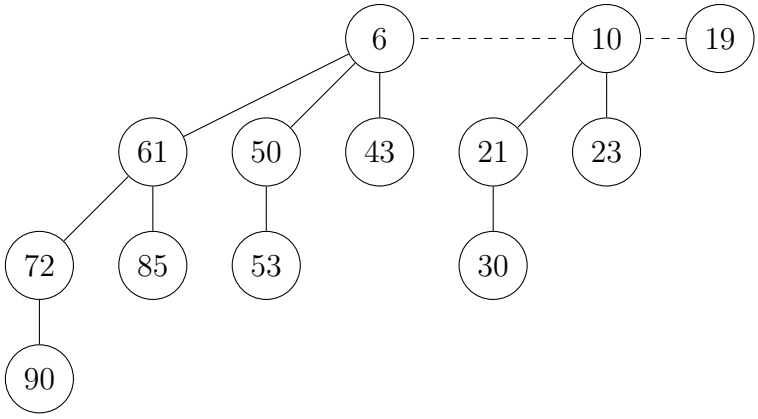
a) Bilden von Paaren:



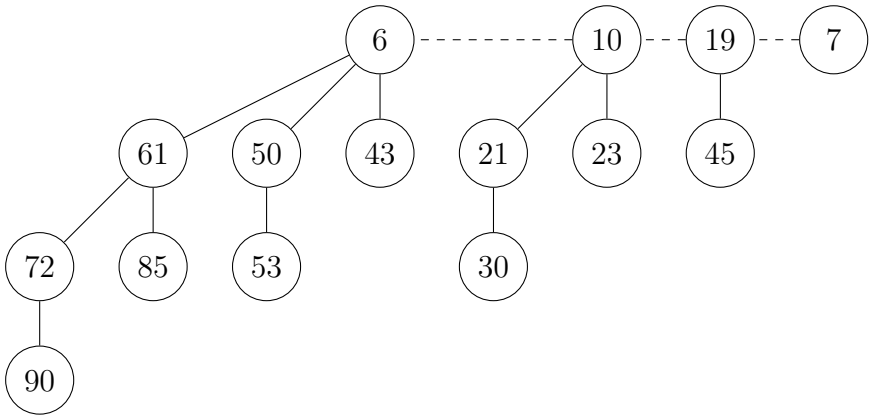
Bilden von Bäumen mit Rang 2:



Bilden von Bäumen mit Rang 3:

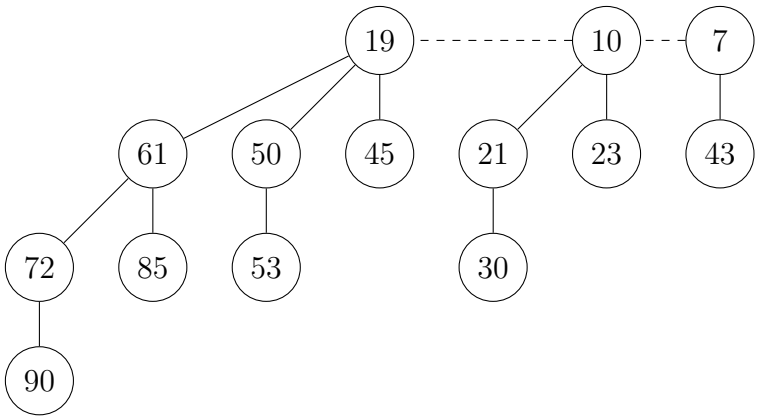


Der Min-Pointer zeigt auf das Element 6.



b)

Der Min-Pointer zeigt auf das Element 6.



c)

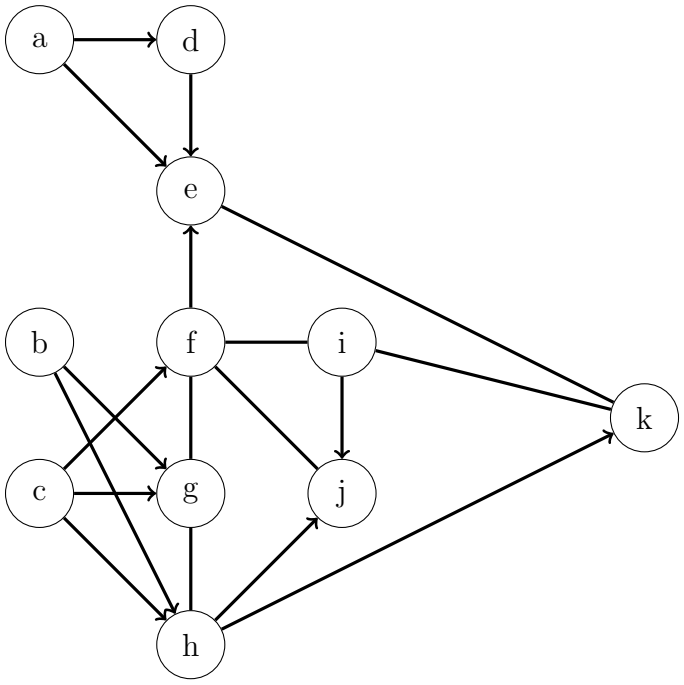
Der Min-Pointer zeigt auf das Element 7.

d) Die Binärdarstellung der Zahlen 456, 301 und 289 gibt die Ränge der Binomialbäume an. Somit gibt es in einem Heap mit 456 Elementen einen Baum mit Rang 8, 7, 6 und Rang 3. In einem Heap mit 302 Elementen gibt es jeweils einen Baum mit Rang 8, 5, 3, 2 und Rang 1. In einem Heap mit 289 Elementen gibt es jeweils einen Baum mit Rang 8, 5 und Rang 0.

Der vereinigte Baum hat 1047 Elemente und daher jeweils einen Baum mit Rang 10, 4, 2, 1 und Rang 0.

Aufgabe 3 (10 Punkte)

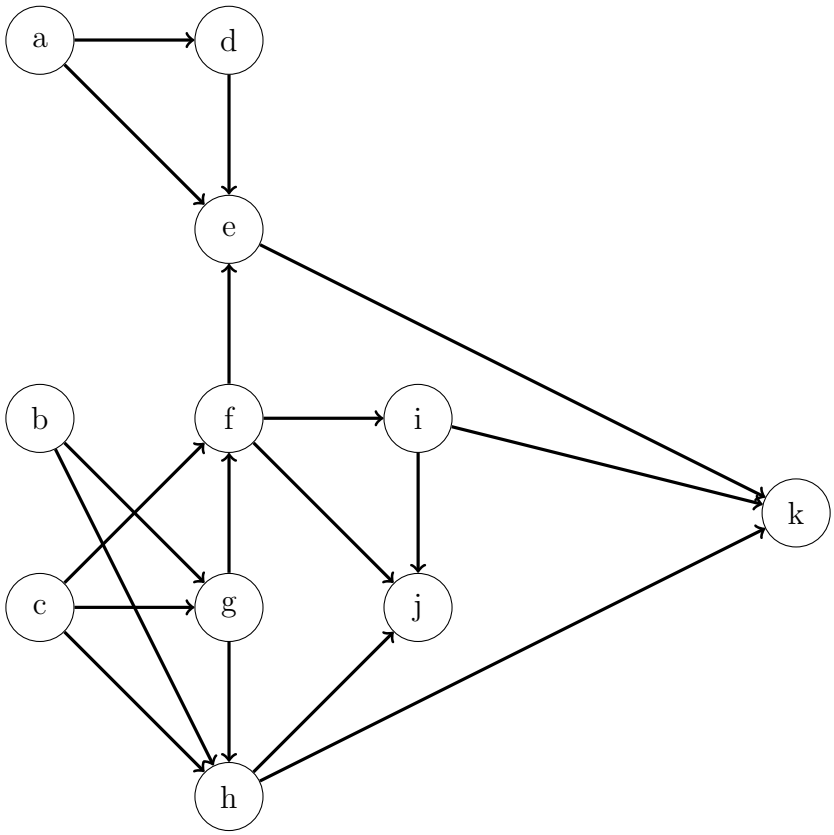
Gegeben sei der untenstehende DAG $G = (V, E)$ und das Berechnungsprotokoll zum Berechnen einer topologischen Sortierung nach dem in der Vorlesung gegebenen Algorithmus. Wir verwenden jedoch nicht wie in der Vorlesung eine Queue sondern einen Stack um Knoten ohne eingehende Kanten von noch nicht fertig bearbeiteten Knoten zu speichern. In die Knotenliste werden Knoten, ähnlich wie bei einem Stack vorne eingefügt und entnommen. Desweiteren werden Knoten in umgekehrter alphabetischer Reihenfolge in die Knotenliste eingefügt (Sie stehen danach also in alphabetischer Ordnung in der Liste). Werden also von einem Knoten x die Knoten y und z entdeckt (und haben diese Knoten keine weiteren eingehenden Kanten von noch nicht fertig bearbeiteten Knoten) und werden in die Liste (u, w) eingefügt, ist die Liste anschließend (y, z, u, w).
Vervollständigen Sie das Berechnungsprotokoll, die topologische Sortierung der Knoten, sowie die Richtung der Kanten im Graphen.
Die Knotenliste beginnt mit den Knoten a, b, c. Danach bekommt der Knoten a den Wert 1 für die topologische Sortierung zugewiesen.



Knotennummer in TopoSort	Knoten x	Knotenliste nach Knoten x
1	a	d, b, c
2		
3		
4		
5	g	
6		
7		i, h
8		
9		
10		
11		

Lösungsvorschlag

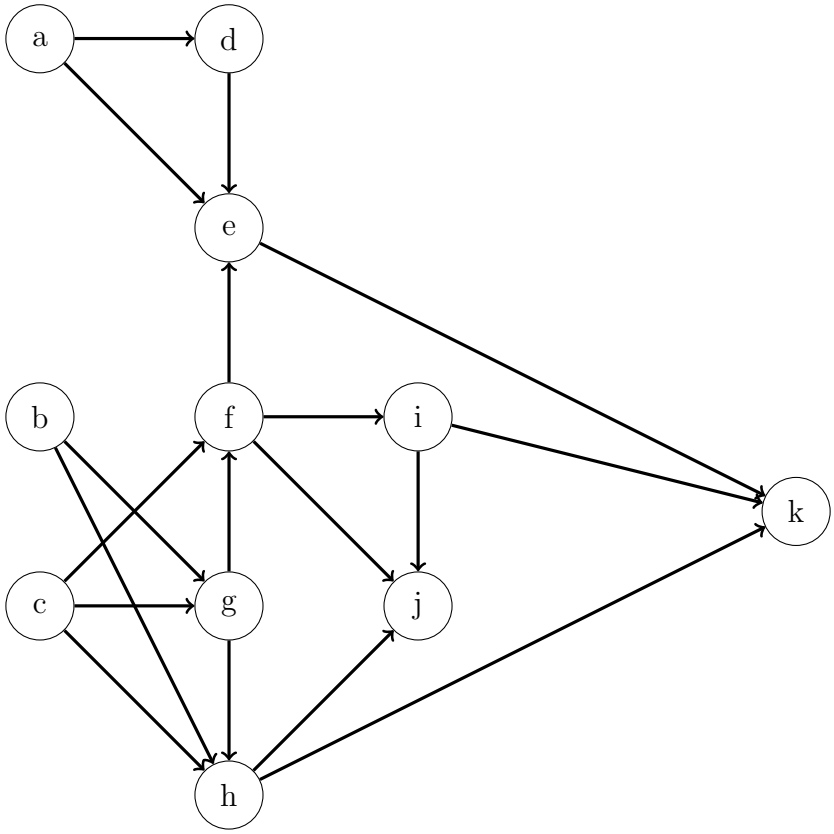
Nach dem Knoten die d und b abgearbeitet sind, wird vorerst kein weiterer Knoten in die Liste eingefügt.



Knotennummer in TopoSort	Knoten x	Knotenliste nach Knoten x
1	a	d, b, c
2	d	b, c
3	b	c
4		
5	g	
6		
7		i, h
8		
9		
10		
11		

Nach dem Knoten c abgearbeitet ist, muss g eingefügt werden. Da dieser die nächste Nummer in der topologischen Sortierung erhält. Daher kann die Kante nur von g nach f gerichtet werden. Da g die nächste Nummer erhält kann auch h keine Kante zu g bekommen.

Damit nun i nach zwei Zuweisungen an erster Stelle stehen kann müssen auch alle Kanten von f ausgehend sein, wie auch von e. Einfache Abarbeitung der Liste ergibt:



Knotennummer in TopoSort	Knoten x	Knotenliste nach Knoten x
1	a	d, b, c
2	d	b, c
3	b	c
4	c	g
5	g	f, h
6	f	e, i, h
7	e	i, h
8	i	h
9	h	j,k
10	j	k
11	k	

Aufgabe 4 (8 Punkte)

a) Führen Sie die folgenden Operationen nacheinander auf dem unten gezeichneten AVL-Baum aus.

i) `insert( 3 )`

ii) `insert( 26 )`

iii) `insert( 34 )`

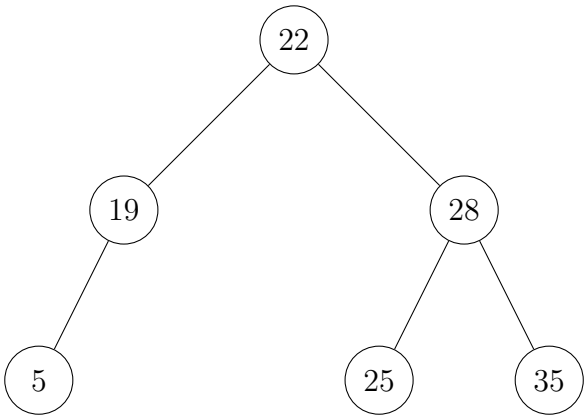
iv) `insert( 24 )`

v) `insert( 27 )`

vi) `delete( 24 )`

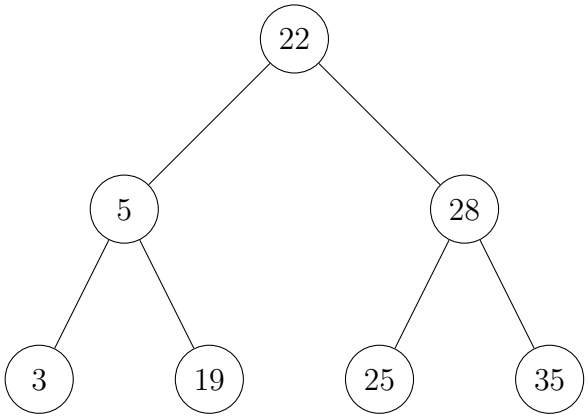
Geben Sie nach jeder Operation den resultierenden AVL-Baum an.

b) Nach dem Einfügen eines Elements in einen AVL-Baum muss die Höhenbedingung unter Umständen wiederhergestellt werden. Wie oft muss dann auf dem Weg zur Wurzel eine Rotation durchgeführt werden? Geben Sie die Laufzeit für die Einfüge-Operation an und wodurch deren Laufzeit beschränkt ist. Begründen Sie Ihre Antworten.

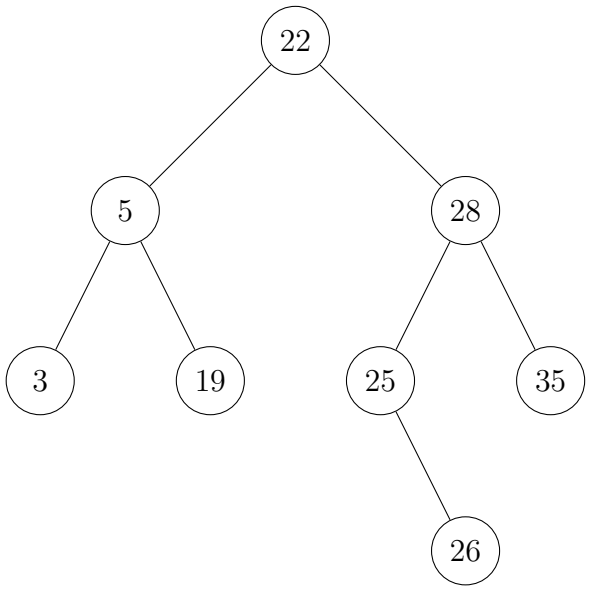


Lösungsvorschlag

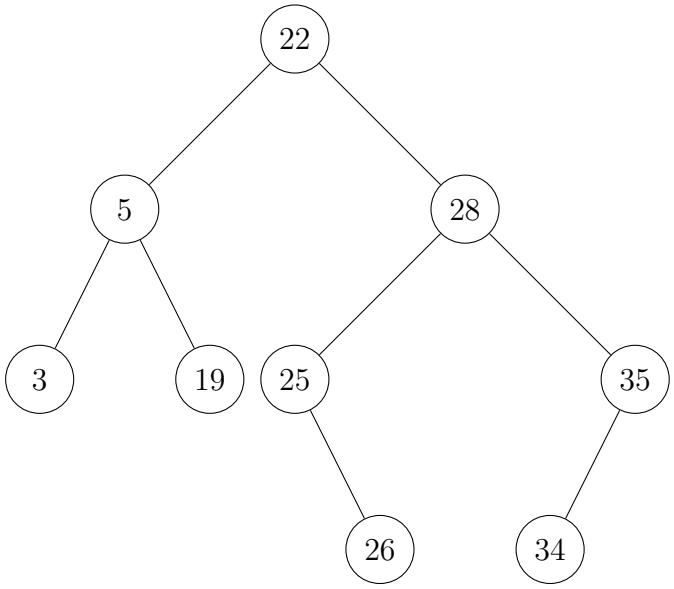
a) i) Einfügen von 3 mit einfacher Rotation in 19:



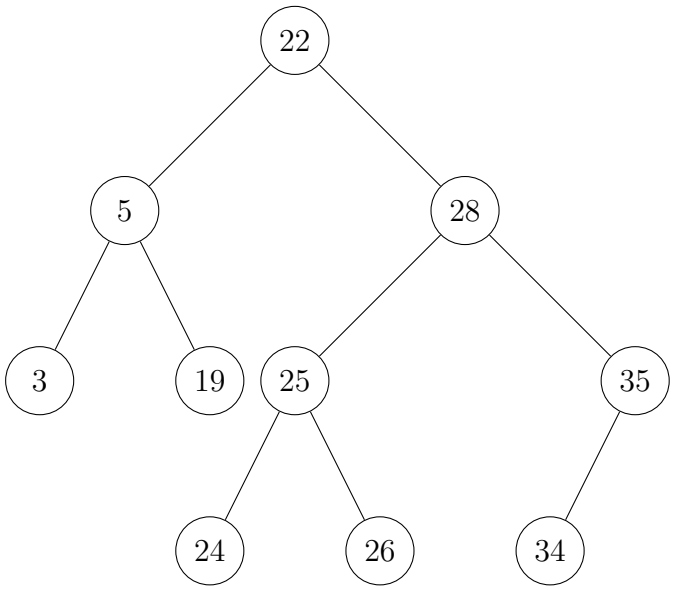
ii) Einfügen von 26 ohne Rotation:



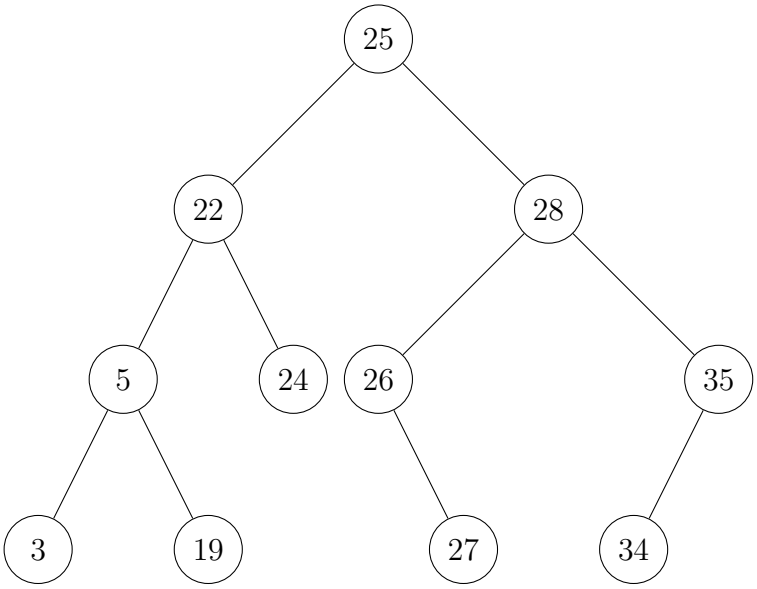
iii) Einfügen von 34 ohne Rotation:



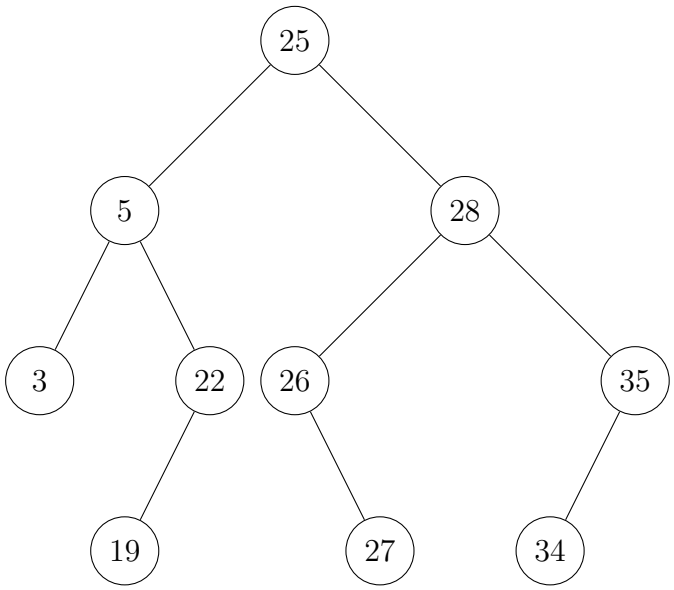
iv) Einfügen von 24 ohne Rotation:



v) Einfügen von 27 mit doppelter Rotation in 22:



vi) Löschen von 24 mit einfacher Rotation in 22:



b) Die Höhenbedingung muss nur einmal korrigiert werden. Danach ist die Höhe des Teilbaums so gross wie der Teilbaum vorher war, und dort galt die Höhenbedingung.

Die Laufzeit der Einfügeoperation ist $\mathcal{O}(\log n)$ da von dem eingefügten Element der Pfad zur Wurzel, der logarithmische Länge hat überprüft werden muss.

Aufgabe 5 (5 Punkte)

Seien die Kosten eines Pfades p in einem gewichteten Graphen G definiert als das Maximum aller Kantengewichte in p. Schlagen Sie einen Algorithmus vor, mit dem man in $\mathcal{O}((n+m)\log n)$ Zeit für einen Knoten s in einem Graphen $G = (V, E)$ mit $|V| = n$, $|E| = m$ und beliebigen Kantengewichten $c : E \mapsto \mathbb{R}$ für alle $t \in V$ die minimalen Pfadkosten von s nach t in G berechnen kann.

Lösungsvorschlag

Wir verwenden eine modifizierte Variante des Dijkstra-Algorithmus. Statt die Wegkosten zu addieren verwenden wir das Maximum der Wegkosten zu v und dem Kantengewicht (v, w).
Daher ist die Laufzeit automatisch die Angegebene. Die Korrektheit folgt aus dem Dijkstra-Algorithmus.