

2020_t1_A

⚠ Disclaimer

Alles was hier drinnen steht kann Fehler enthalten!, Falls dir etwas auffällt melde dich gerne auf Discord bei mir ([@xmozz](#))

A1 - Algorithmenanalyse

Stoff: [2. Analyse von Algorithmen](#)

a)

(6 Punkte) Ordnen Sie folgende Funktionen nach Dominanz (\ll), beginnend mit der asymptotisch am schwächsten wachsenden. Es genügt die Funktionen zu reihen, ein Beweis der Gültigkeit der Relationen ist nicht erforderlich.

$$\left(\frac{4}{5}\right)^{2n}, \quad (2n)!, \quad \frac{3n^6 + 5n^3}{7n^2}, \quad \log(n^n), \quad 2 \cdot \sqrt{10n^7}, \quad n \cdot 100, \quad (1.01)^{3n}$$

$$\left(\frac{4}{5}\right)^{2n} \ll n \cdot 100 \ll \log(n^n) \ll 2 \cdot \sqrt{10n^7} \ll \frac{3n^6 + 5n^3}{7n^2} \ll (1.01)^{3n} \ll (2n)!$$

b)

(6 Punkte) Gegeben sind die folgenden Funktionen:

$$f(n) = n^2 + \log(n + 10)$$

$$g_1(n) = \sqrt{n^4} \cdot \log n$$

$$g_2(n) = \frac{n^5 - 5n^3}{3n^3}$$

$$g_3(n) = \begin{cases} (2n)^3 & \text{falls } n \text{ gerade} \\ \frac{3n^2}{\sqrt{n}} & \text{sonst} \end{cases}$$

$$f(n) = \Theta(n^2)$$

$$g_1(n) = \Theta(n^2 \cdot \log n)$$

$$g_2(n) = \Theta(n^2)$$

$$g_3(n) = \Theta(n^3 \text{ oder weniger als } n^2)$$

$f(n)$ ist in	$\Theta(\cdot)$	$O(\cdot)$	$\Omega(\cdot)$	keines
$g_1(n)$		x		
$g_2(n)$	x	x	x	
$g_3(n)$				x

c)

(8 Punkte) Bestimmen Sie die Laufzeiten der unten angegebenen Algorithmen in Abhängigkeit vom Eingabeparameter n in Θ -Notation. Verwenden Sie hierfür möglichst einfache Terme.

n
 $k \leftarrow 0$
 | $\text{for } i = 1, \dots, n$
 | | $\text{for } j = 1, \dots, i$ | n
 | | | $k \leftarrow k + j$
 | | | $\text{for } j = i + n, \dots, 3n$ | $2n \dots 3n$
 | | | | $k \leftarrow k - j$ | $\max 3n \rightarrow O(n)$
 | | | $\text{return } k$

 $\Rightarrow O(n^2)$

$2n$
 $z \leftarrow 0$
 | $\text{for } i = 1, \dots, 2n$
 | | $j \leftarrow n + 5$
 | | $\text{while } j > 1$
 | | | $j \leftarrow \frac{j}{2}$
 | | | $z \leftarrow z + j$
 | | | $\text{return } z$

 $\Rightarrow O(n \log(n))$

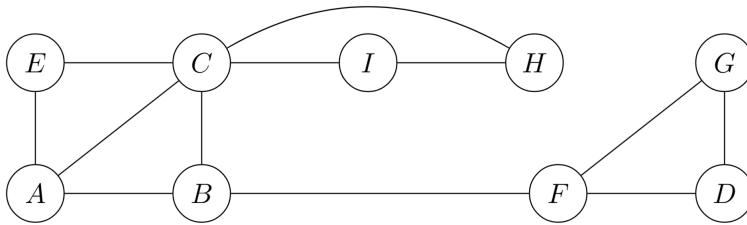
A2 - Graphen

Stoff: 3. Graphen

a)

(4 Punkte) Führen Sie auf dem nachfolgenden Graphen die Breiten- und Tiefensuche entsprechend den Algorithmen in den Vorlesungsfolien durch. Geben Sie dabei jeweils die Reihenfolge an, in der die Knoten besucht werden.

Verwenden Sie jeweils F als Startknoten. Haben Sie die Wahl zwischen mehreren Knoten, gehen Sie alphabetisch vor.



BFS: F, B, D, G, A, C, E, H, I

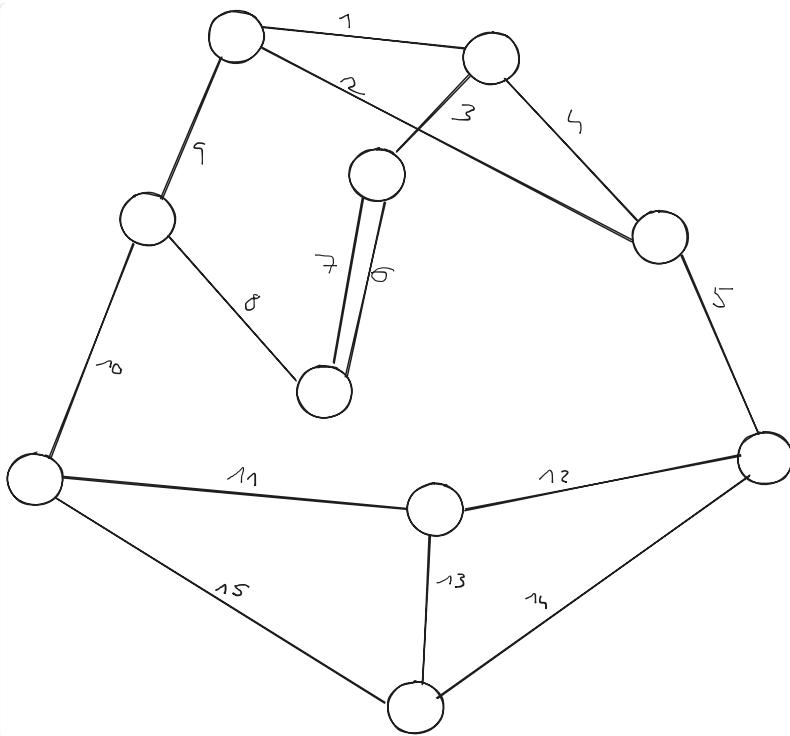
DFS: F, B, A, C, E, H, I, D, G

b)

(4 Punkte) Betrachten Sie die folgenden Behauptungen in Bezug auf ungerichtete und gerichtete Graphen. Kreuzen Sie an, ob diese wahr oder falsch sind.

(+1 Punkt für jede richtige, -1 Punkt für jede falsche und 0 Punkte für keine Antwort, keine negativen Punkte)

- Es gibt gerichtete Graphen, die eine topologische Sortierung besitzen und einen gerichteten Kreis enthalten.
 Wahr Falsch
- Jeder zusammenhängende, ungerichtete Graph mit n Knoten und mehr als $n + 1$ Kanten enthält mindestens einen Kreis.
 Wahr Falsch
- Jeder stark zusammenhängende, gerichtete Graph mit 8 Knoten enthält mindestens einen gerichteten Kreis.
 Wahr Falsch
- Jeder ungerichtete Graph mit 10 Knoten, in dem jeder Knoten den Knotengrad 3 hat, enthält genau 15 Kanten.
 Wahr Falsch



- Es gibt gerichtete Graphen, die eine topologische Sortierung besitzen und einen gerichteten Kreis enthalten.
 - Falsch. Eine topologische Sortierung ist nur für gerichtete azyklische Graphen (DAGs) definiert. Ein Graph, der einen gerichteten Kreis enthält, ist nicht azyklisch.
- Jeder zusammenhängende, ungerichtete Graph mit n Knoten und mehr als $n+1$ Kanten enthält mindestens einen Kreis.
 - Wahr. Ein zusammenhängender ungerichteter Graph mit n Knoten und ohne Kreise ist ein Baum, der genau $n - 1$ Kanten hat. Wenn ein solcher Graph mehr als $n - 1$ Kanten besitzt, muss er mindestens einen Kreis enthalten. Da $n + 1 > n - 1$ (für $n > 0$), ist die Aussage wahr.
- Jeder stark zusammenhängende, gerichtete Graph mit 8 Knoten enthält mindestens einen gerichteten Kreis.
 - Wahr. Ein stark zusammenhängender gerichteter Graph mit mehr als einem Knoten muss immer mindestens einen gerichteten Kreis enthalten. Wäre er azyklisch, gäbe es Quell- und Senkenknoten, was der starken Zusammenhangseigenschaft widersprechen würde.
- Jeder ungerichtete Graph mit 10 Knoten, in dem jeder Knoten den Knotengrad 3 hat, enthält genau 15 Kanten.
 - Wahr. Gemäß dem Handschlaglemma ist die Summe der Grade aller Knoten in einem ungerichteten Graphen gleich dem Doppelten der Anzahl der Kanten ($\sum \deg(v) = 2 \cdot |E|$). Bei 10 Knoten, die jeweils Grad 3 haben, beträgt die Summe der Grade $10 \times 3 = 30$. Daraus folgt $2 \cdot |E| = 30$, also $|E| = 15$.

c)

(12 Punkte) Gegeben ist die folgende Beschreibung einer Ausführung des *Algorithmus von Dijkstra* auf einen gerichteten Graphen in der Implementierung mit einer Liste.

- Discovered = \emptyset
 $L = \{s, a, b, c, d, e, u\}$

Knoten	s	a	b	c	d	e	u
$d(\cdot)$	0	∞	∞	∞	∞	∞	∞

- Discovered = $\{s\}$
 $L = \{a, b, c, d, e, u\}$

Knoten	s	a	b	c	d	e	u
$d(\cdot)$	0	17	5	∞	3	∞	∞

- Discovered = $\{s, d\}$
 $L = \{a, b, c, e, u\}$

Knoten	s	a	b	c	d	e	u
$d(\cdot)$	0	17	4	∞	3	∞	∞

- Discovered = $\{s, d, b\}$
 $L = \{a, c, e, u\}$

Knoten	s	a	b	c	d	e	u
$d(\cdot)$	0	12	4	5	3	∞	∞

- Discovered = $\{s, d, b, c\}$
 $L = \{a, e, u\}$

Knoten	s	a	b	c	d	e	u
$d(\cdot)$	0	9	4	5	3	∞	15

- Discovered = $\{s, d, b, c, a\}$
 $L = \{e, u\}$

Knoten	s	a	b	c	d	e	u
$d(\cdot)$	0	9	4	5	3	11	12

- Discovered = $\{s, d, b, c, a, e\}$
 $L = \{u\}$

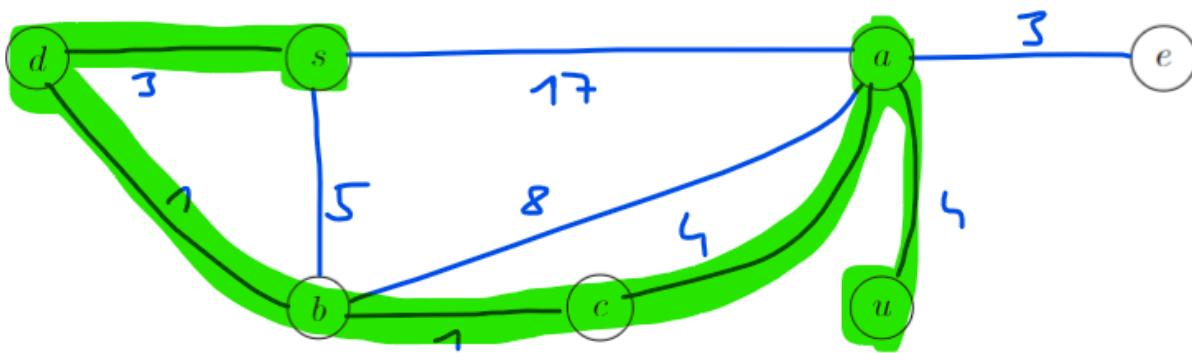
Knoten	s	a	b	c	d	e	u
$d(\cdot)$	0	9	4	5	3	11	12

- Discovered = $\{s, d, b, c, a, e, u\}$
 $L = \emptyset$

Knoten	s	a	b	c	d	e	u
$d(\cdot)$	0	9	4	5	3	11	12

(i)

Extrahieren Sie aus diesem Ablauf den kürzesten Pfad von s nach u sowie seine Länge. Geben Sie den Pfad als Liste von Knoten an.

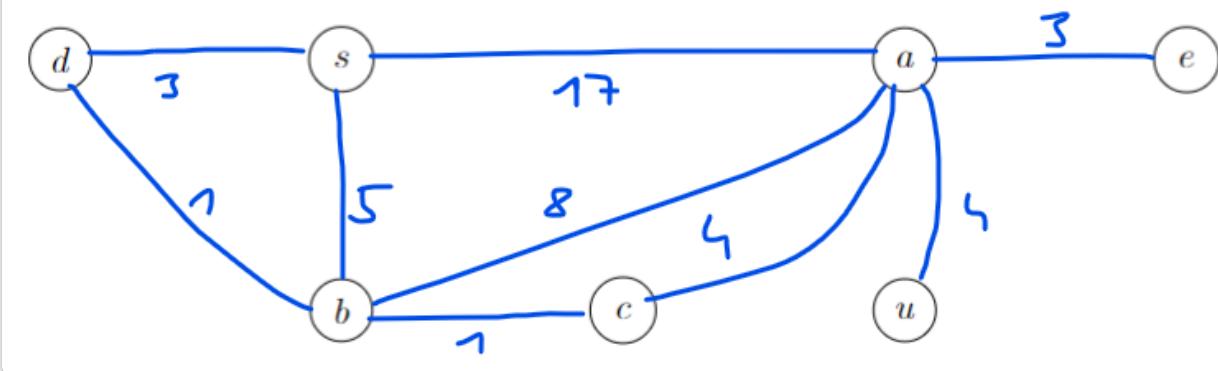


Pfad: $s \rightarrow d \rightarrow b \rightarrow c \rightarrow a \rightarrow u$

Gewicht: 12

(ii)

Zeichnen Sie die Kanten und Kantengewichte eines Graphen der zu einem solchen Ablauf führt in der folgenden Grafik ein.

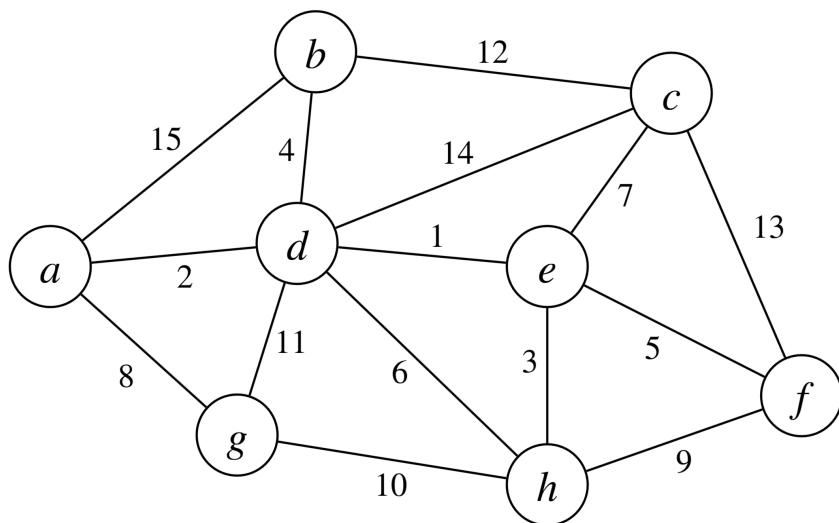


A3 - Greedy

Stoff: 4. Greedy-Algorithmen

a)

(10 Punkte) Ermitteln Sie mit dem Verfahren von Kruskal für den unten dargestellten gewichteten Graphen einen minimalen Spannbaum. Markieren Sie dazu die ausgewählten Kanten oder geben Sie die Kantenmenge in Form einer Liste an. Geben Sie außerdem die Reihenfolge an, in der die Kanten des Spannbaums hinzugefügt werden.



1. DE
2. AD
3. EH
4. BD
5. EF
6. CE
7. AG

b)

(3 Punkte) Sei $G = (V, E)$ ein *nicht zusammenhängender* Graph und c Kantengewichte von E . Berechnet der Algorithmus von *Prim* mit der Eingabe (G, c) einen minimalen Spannbaum für jede Zusammenhangskomponente, wenn mit einem beliebigen Startknoten $v \in V$ begonnen wird? Begründen Sie Ihre Antwort.

Ja Nein

Begründung:

Nein, da er ja nicht in die anderen Teilgraphen ohne direkte Kantenverbindung initialisiert wird und durch das einfache ausführen vom Algorithmus im Anfangsteilbaum bleibt.

c)

(3 Punkte) Sei $G = (V, E)$ ein *nicht zusammenhängender* Graph und c Kantengewichte von E . Berechnet der Algorithmus von *Kruskal* mit der Eingabe (G, c) einen minimalen Spannbaum für jede Zusammenhangskomponente? Begründen Sie Ihre Antwort.

Ja Nein

Begründung:

Ja, da wir hier nicht das Problem haben, dass wir von schon den besuchten Kanten weitergehen sondern in sortierter Reihenfolge voranschreiten, egal ob es eine Verbindung dazu gibt.

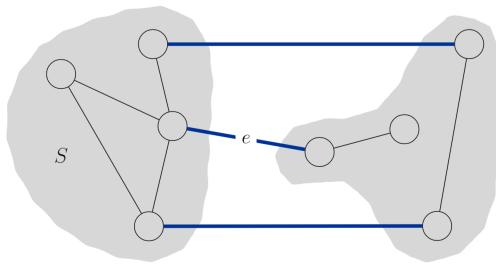
d)

- Das Kreislemma besagt, dass die günstigste Kante jedes in einem Graphen enthaltenen Kreises Teil des minimalen Spannbaumes dieses Graphen sein muss.
 Wahr Falsch
- Der minimale Spannbaum eines Graphen enthält zumindest eine Kante minimalen Gewichtes innerhalb jeder Kantenschnittmenge.
 Wahr Falsch
- Der Algorithmus von Prim und der Algorithmus von Kruskal berechnen gültige minimale Spannbäume und retournieren somit zwangsläufig denselben Spannbaum.
 Wahr Falsch
- Der Algorithmus von Kruskal ist in der Praxis auf *dünnen* Graphen gegenüber dem Algorithmus von Prim zu bevorzugen.
 Wahr Falsch

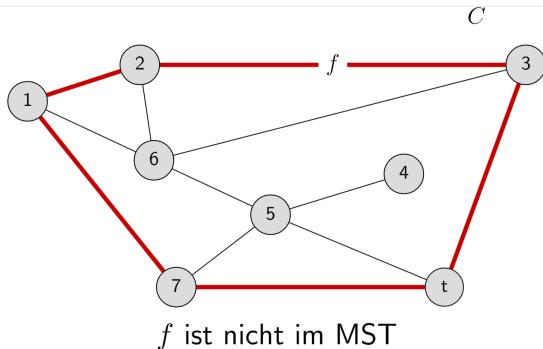
Lemmata

Kantenschnittlemma: Sei $S \subseteq V$ eine beliebige Teilmenge von Knoten und sei e die minimale gewichtete Kante mit genau einem Endknoten in S . Dann enthält der MST T die Kante e .

Kreislemma: Sei C ein beliebiger Kreis und sei f die maximal gewichtete Kante in C . Dann enthält der MST T die Kante f nicht.



e ist im MST



f ist nicht im MST

Vereinfachende Annahme: Alle Kantengewichte sind unterschiedlich, dadurch ist der MST eindeutig.

26

A4 - Sortierverfahren und Divide-and-Conquer

Stoff: 5. Divide and Conquer

Führen Sie **Quicksort** auf dem unten gegebenen Array aus. Wählen Sie stets das letzte Element als Pivot-Element aus und implementieren Sie den Schritt des Aufteilens so, dass die Elemente einer Subfolge immer in derselben Reihenfolge angeordnet werden wie in der Originalfolge. Geben Sie, wie in der Vorlesung und Übung kennen gelernt, die Zwischenschritte an. Markieren Sie in jedem Zwischen-Schritt das ausgewählte Pivot-Element durch einkreisen.

8	2	4	7	1	9	3
---	---	---	---	---	---	---

8 2 4 7 1 9 3

2 7 3 4 7 9 8

1 2 3 4 7 8 9

1 2 3 4 7 8 9

So wie in der vor beschrieben wurde würde das so aussehen... aber wirkt bisschen wenig...

b)

Sei $C(n)$ die Anzahl der Schlüsselvergleiche in Mergesort bei einer Eingabegröße n . Geben Sie die Rekursionsgleichung zur Berechnung von $C(n)$ von Mergesort an.

Sei $C(n)$ die Anzahl der Schlüsselvergleiche in Mergesort bei einer Eingabegröße n . Die Rekursionsgleichung zur Berechnung von $C(n)$ von Mergesort ist wie folgt:

$$C(n) = C(\lceil n/2 \rceil) + C(\lfloor n/2 \rfloor) + n - 1$$

Erläuterung:

- $C(\lceil n/2 \rceil)$: Dies ist die Anzahl der Vergleiche für die Sortierung der ersten Hälfte des Arrays.
- $C(\lfloor n/2 \rfloor)$: Dies ist die Anzahl der Vergleiche für die Sortierung der zweiten Hälfte des Arrays.
- $n - 1$: Dies repräsentiert die maximale Anzahl der Vergleiche, die im Merging-Schritt (Zusammenführen der beiden sortierten Hälften) benötigt werden. Im schlimmsten Fall werden $n - 1$ Vergleiche durchgeführt, um n Elemente zu verschmelzen.

Beispiele für kleinere n (aus den gegebenen handschriftlichen Notizen):

- $C(1) = 0$
- $C(2) = 1$
- $C(3) = 3$
- $C(4) = 5$
- $C(5) = 8$
- $C(6) = 11$
- $C(7) = 14$

c)

Sei die Eingabe von nachfolgenden Sortieralgorithmen ein Array mit n Zahlen im Bereich 0 bis z mit $z < n$. Ergänzen Sie folgende Laufzeiten von den in der Vorlesung kennengelernten Sortieralgorithmen in Abhängigkeit von n .

- Die Worst-Case Laufzeit von Countsort ist in

$$\Theta(n)$$

- Die Average-Case Laufzeit von Quicksort ist in

$$\Theta(n \log n)$$

- Die Best-Case Laufzeit von Mergesort ist in

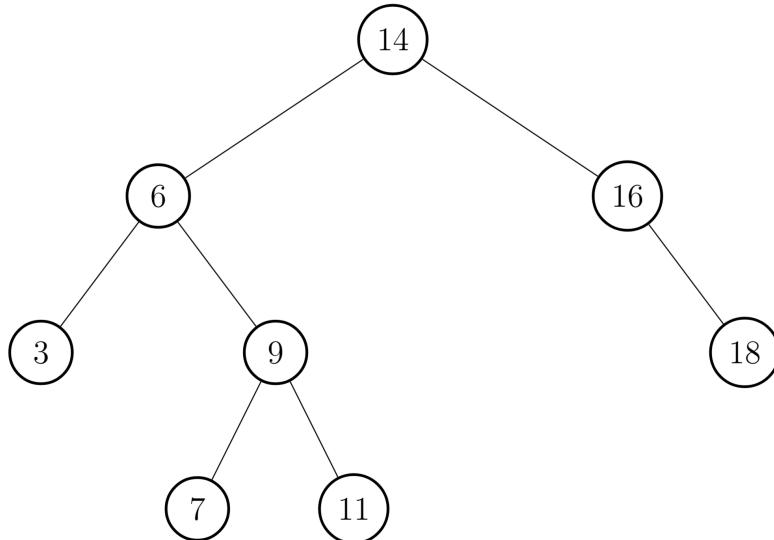
$$\Theta(n \log n)$$

A5 - Suchbäume und Hashing

Stoff: [6. Suchbäume](#) und [7. Hashing](#)

a)

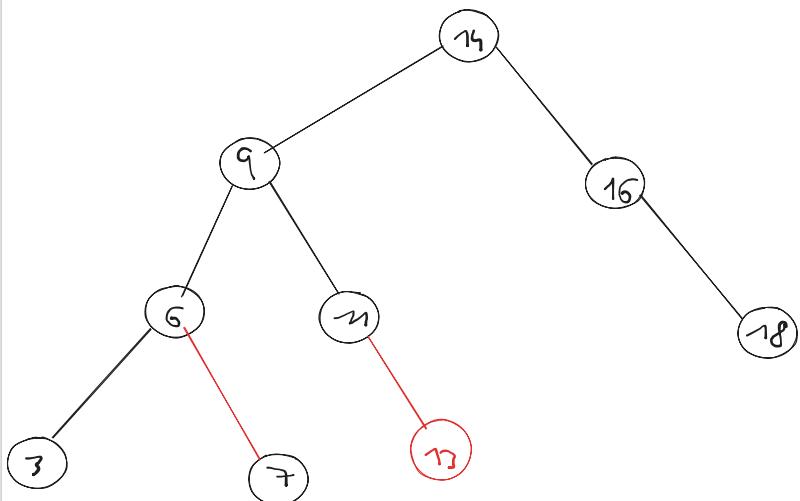
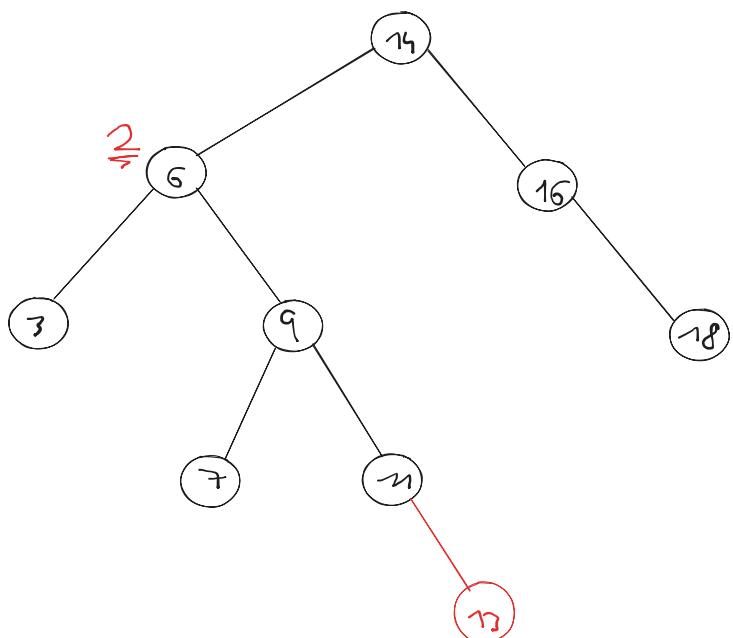
Gegeben ist folgender **AVL-Baum**:



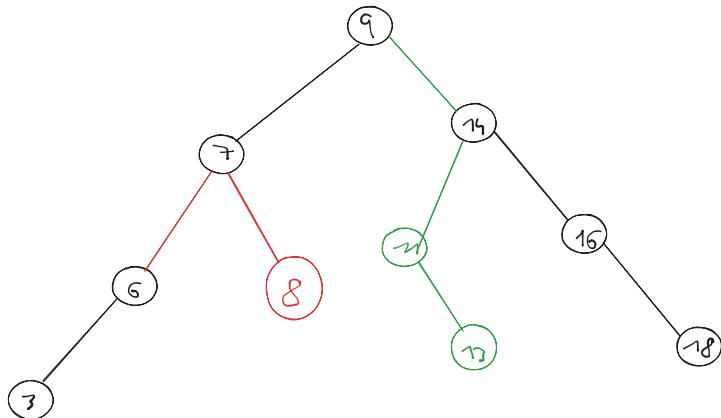
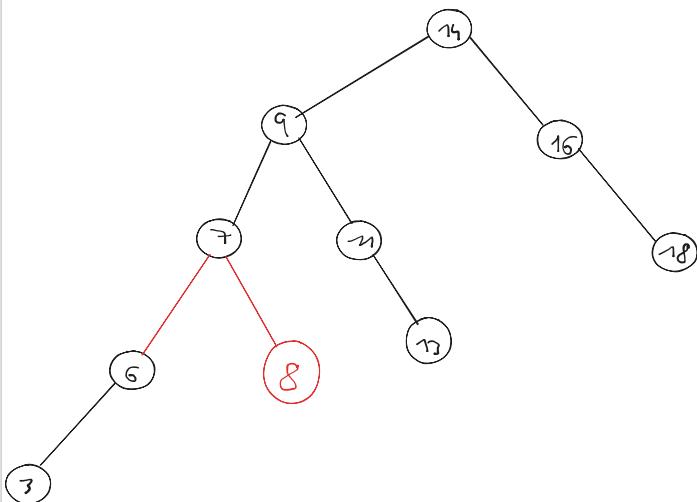
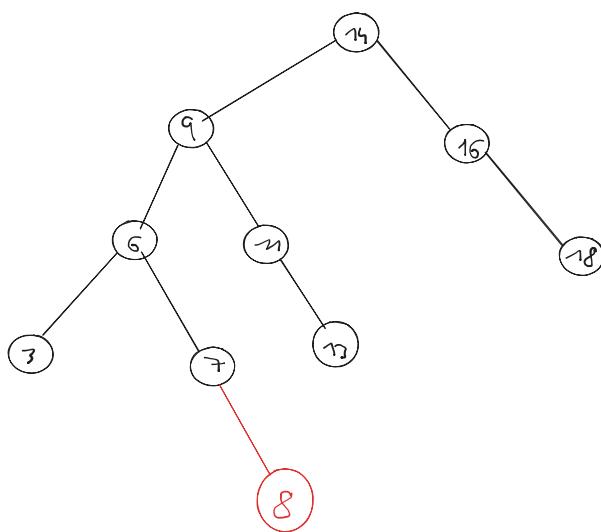
(i)

(6 Punkte) Fügen Sie die Schlüssel 13 und 8 in dieser Reihenfolge in diesen AVL-Baum ein. Falls notwendig, so rebalancieren Sie den Baum nach jedem Einfügen mit geeigneten Rotationsoperationen, um wieder einen gültigen AVL-Baum zu erhalten. Zeichnen Sie den vollständigen Baum direkt nach dem Einfügen und nach jedem Rotationsschritt.

Einfügen von 13

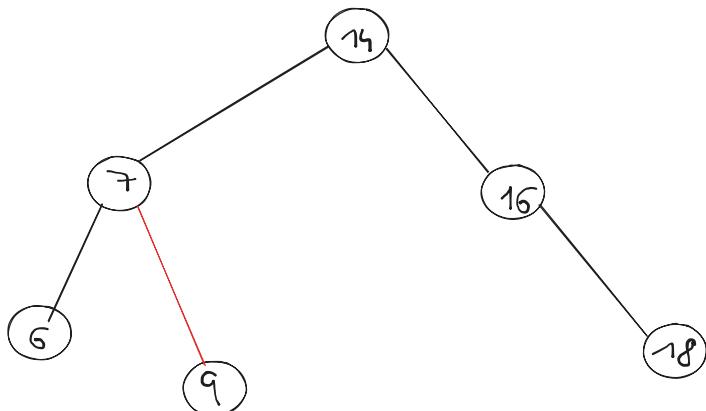
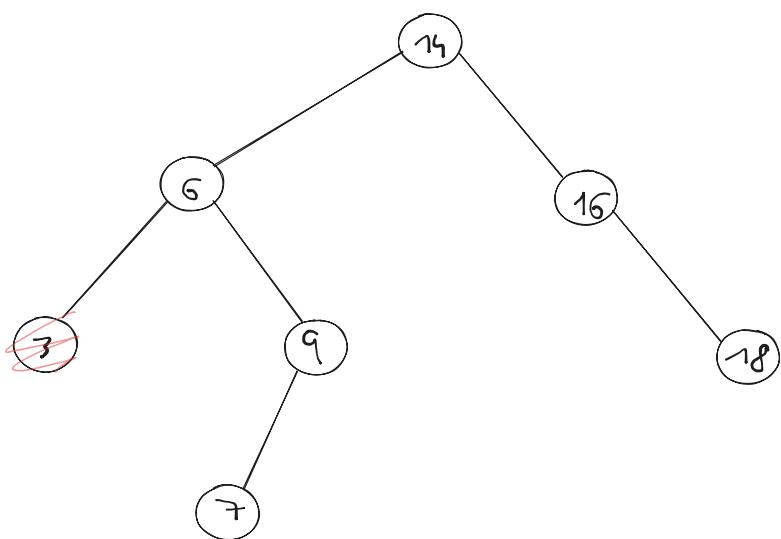
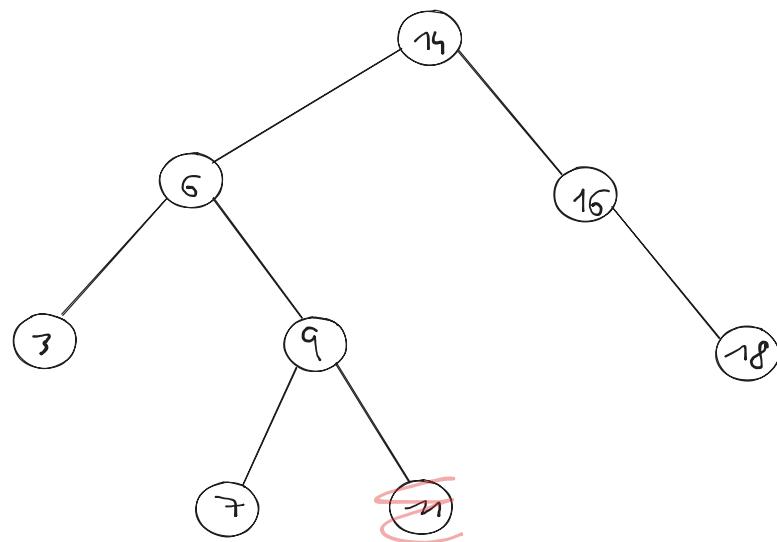


Einfügen von 8



(ii)

(6 Punkte) Löschen Sie aus dem *ursprünglichen*, gegebenen AVL-Baum die Schlüssel 11 und 3 in dieser Reihenfolge. Falls notwendig rebalancieren Sie den Baum nach jedem Löschvorgang mit geeigneten Rotationsoperationen. Zeichnen Sie wiederum den vollständigen Baum direkt nach dem Löschen und nach jedem Rotationsschritt.

**b)**

(8 Punkte) Fügen Sie die Elemente der Folge

$$\langle 18, 6, 7, 13, 14, 20, 22, 26, 35, 10 \rangle$$

in dieser Reihenfolge in einen anfangs leeren **B-Baum der Ordnung 3** ein. Zeichnen Sie den B-Baum jeweils vor und nach jeder Reorganisationsmaßnahme und geben Sie den endgültigen B-Baum an.

