

2021_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)

(4 Punkte) Bestimmen Sie die Laufzeit des angegebenen Algorithmus in Abhängigkeit vom Eingabeparameter n in Θ -Notation. Verwenden Sie hierfür möglichst einfache Terme.

```

 $a \leftarrow n^3 + n$ 
 $b \leftarrow 1$ 
while  $a > 4$ 
     $a \leftarrow 2 * a / 5$ 
     $b \leftarrow a + b$ 
return  $b$ 

```

Laufzeit in Θ -Notation: $\Theta(\log n)$

b)

(8 Punkte) Bestimmen Sie die Laufzeit und den Rückgabewert des angegebenen Algorithmus in Abhängigkeit vom Eingabeparameter n in Θ -Notation. Verwenden Sie hierfür möglichst einfache Terme.

```

 $b \leftarrow 2$ 
for  $i = 1, \dots, n$ 
     $k \leftarrow 1$ 
    for  $j = 1, \dots, \lceil \log_2(1+i) \rceil$ 
         $k \leftarrow 2 * k$ 
     $b \leftarrow b + k$ 
     $j \leftarrow n/2 + i$ 
    while  $j > n/4$ 
         $j \leftarrow j - 1$ 
return  $b$ 

```

Laufzeit in Θ -Notation: $\Theta(n^2)$

Rückgabewert in Θ -Notation: $\Theta(n^2)$

$\approx n \cdot (2^{\log(n)}) = n \cdot (n^{\log(2)}) = n \cdot n^1 = n^2$

$\Rightarrow R_{ct} : \Theta(n^2)$

b $\leftarrow 2$
for $i = 1, \dots, n$
 $k \leftarrow 1$
 for $j = 1, \dots, \lceil \log_2(1+i) \rceil$ ————— $k=1, 2, 4, 8\dots$
 $k \leftarrow 2 * k$
 $b \leftarrow b + k$
 $j \leftarrow n/2 + i$
 while $j > n/4$
 $j \leftarrow j - 1$
return b

e^n —————

c)

(6 Punkte) Gegeben sind die folgenden Funktionen:

$$f(n) = n + \log(n^4 + 10)$$

$$g_1(n) = \sqrt{n^5} / \log n$$

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

$$g_3(n) = \begin{cases} 2n + 7 & \text{falls } 5n < n^2 - 3 \\ \frac{3n^2 + 5n + 6}{7} & \text{sonst} \end{cases}$$

$$f(n) = \Theta(n + \log n)$$

f(n) ist in	$\Theta(\cdot)$	$O(\cdot)$	$\Omega(\cdot)$	keines
g1(n)		x		
g2(n)		x		
g3(n)	x	x	x	

d)

- d) (2 Punkte) Nehmen Sie an, wir haben für ein $c > 0$ und ein $n_0 \in \mathbb{N}^+$ gezeigt, dass für alle $n \geq n_0$ gilt

$$\sqrt{4n+3} \leq cn.$$

Was haben wir somit gezeigt? Kreuzen/Geben Sie alle zutreffenden Aussagen an:

- (MC1) $\sqrt{4n+3}$ ist in $O(n)$
 (MC2) $\sqrt{4n+3}$ ist in $\Theta(n)$
 (MC3) $\sqrt{4n+3}$ ist in $\Omega(n)$
 (MC4) keine der zuvor genannten Aussagen

A2 - Graphen

Stoff: 3. Graphen

a)

- (Q1) Ein gerichteter azyklischer Graph mit 17 Knoten hat mindestens 11 starke Zusammenhangskomponenten.

Wahr Falsch

- (Q2) Jeder gerichtete Graph mit einer Quelle besitzt eine topologische Sortierung.

Wahr Falsch

- (Q3) Jeder zusammenhängende, ungerichtete Graph mit n Knoten und mehr als $n + 1$ Kanten enthält mindestens einen Kreis.

Wahr Falsch

- (Q4) In einem ungerichteten Graphen, bei dem alle Kanten ein Gewicht von 1 haben, lassen sich kürzeste Pfade mithilfe einer Breitensuche finden.

Wahr Falsch

1. Nein weil mindestens 17

2. Nein muss azyklisch sein

b)

(6 Punkte) Betrachten Sie folgenden Algorithmus:

Mystery(G, s):

Discovered[s] $\leftarrow 0$

Discovered[v] $\leftarrow -1$ für alle anderen Knoten $v \in V$

Queue $Q \leftarrow \{s\}$

while Q ist nicht leer

 Entferne ersten Knoten u aus Q

forall Kante (u, v) inzident zu u **do**

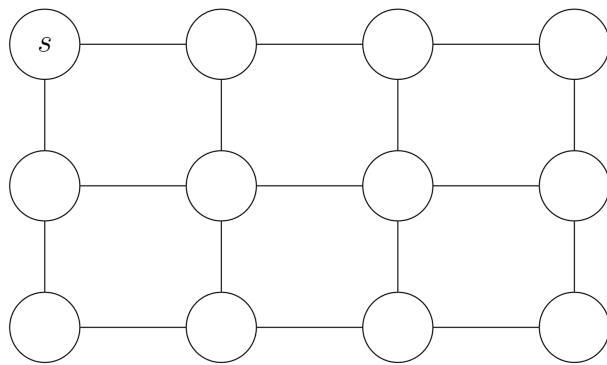
if Discovered[v] == -1 **then**

 Discovered[v] \leftarrow Discovered[u] + 1

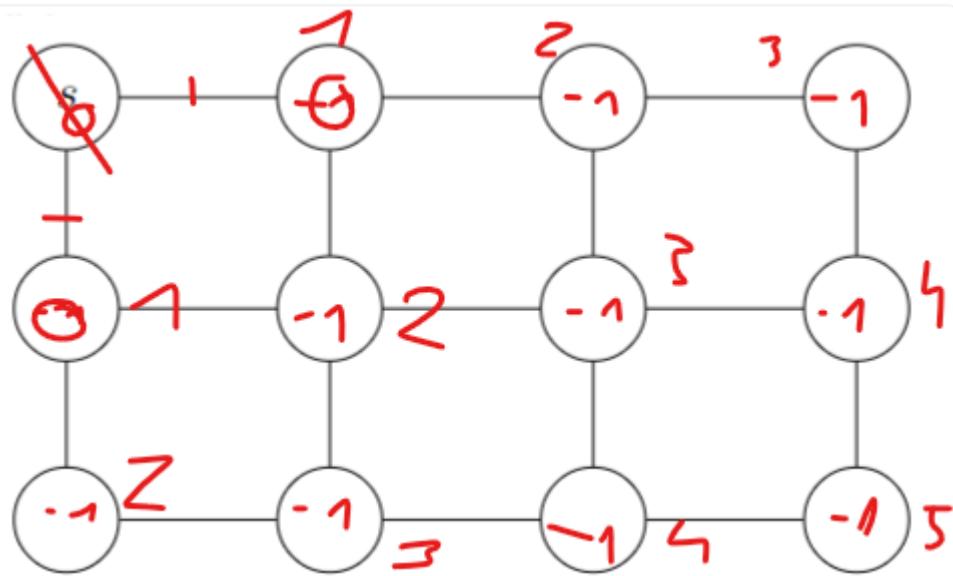
 Füge v zu Q hinzu

return maximalen Eintrag in Discovered

Welche Zahl gibt der Algorithmus aus, wenn man ihn auf folgendem Graphen mit Startknoten s aufruft?



Er gibt 5 aus



c)

Das Straßennetzwerk einer Stadt besteht aus Punkten, die ausschließlich durch Einbahnstraßen verbunden sind. Nun hat der Verkehrsminister die Bedenken geäußert, dass Folgendes passieren könnte: Wenn man von einem Punkt A zu einem Punkt B fährt, kommt man unter Umständen nicht mehr von B nach A zurück. Wir wollen untersuchen, unter welchen Umständen es solche zwei Punkte A und B gibt.

(i)

Welcher Typ von Graph eignet sich um dieses Problem zu modellieren?

Gerichteter Graph

(ii)

Welche Rolle haben die Knoten in Ihrem Graph?

Punkte

(iii)

Wann sind zwei Knoten mit einer Kante verbunden?

Wenn eine Straße von $Punkt_1$ zu $Punkt_2$ führt geht eine Kante von $Knoten_1$ zu $Knoten_2$

(iv)

Beschreiben Sie mithilfe der Begriffe von starken und schwachen Zusammenhang welche Eigenschaften der Graph genau erfüllen muss, damit es solche zwei Punkte A und B gibt. Insbesondere erwähnen Sie, in welcher Beziehung die starken und schwachen Zusammenhangskomponenten von A und B zueinander stehen müssen.

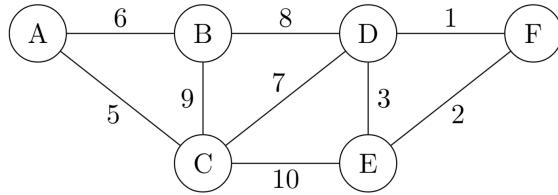
Sie dürfen in keiner starken Zusammenhangskomponente sein sondern nur in einer schwachen.

A3 - Greedy

Stoff: [4. Greedy-Algorithmen](#)

a)

(6 Punkte) Berechnen Sie einen minimalen Spannbaum des untenstehenden Graphen. Geben Sie die Reihenfolge an, in der der Algorithmus von **Prim** (mit Startknoten A) die Kanten des Graphen in einen Spannbaum einfügt. Geben Sie die Reihenfolge als Liste von Knotenpaaren an (z.B.: AB, BC, CD, ...).



AC, AB, CD, DF, FE

b)

(2 Punkte) Hat der obige Graph einen eindeutigen minimalen Spannbaum? Begründen Sie Ihre Antwort.

Ja es gibt nämlich keine Kanten mit gleichen Knotengewichten

c)

(Q1) Das Kreislemma besagt, dass ein minimaler Spannbaum eines Graphen G zumindest eine Kante jedes Kreises von G enthält.

Wahr Falsch

(Q2) Jeder Graph hat einen minimalen Spannbaum.

Wahr Falsch

(Q3) Sei S eine Teilmenge von Knoten eines zusammenhängenden Graphen und e eine Kante minimalen Gewichts, die genau einen Endknoten in S hat. Dann gibt es einen minimalen Spannbaum, der e enthält.

Wahr Falsch

(Q4) Der Algorithmus von Kruskal hat auf dünnen Graphen eine bessere asymptotische Laufzeit als der Algorithmus von Prim.

Wahr Falsch

1. Kreislemma: Maximale Kante vom Kreis ist nie im MST

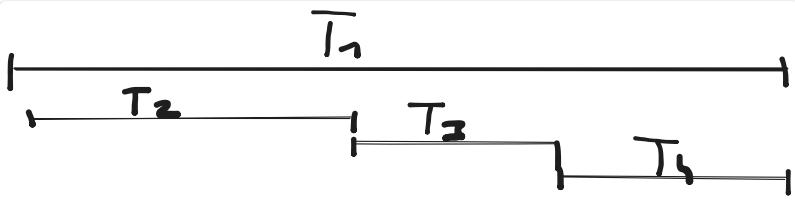
2. Nein muss zusammenhängend sein

d)

(4 Punkte) Begründen Sie mit einem Gegenbeispiel, warum die folgende Greedy-Strategie für das **Interval Scheduling** Problem im Allgemeinen keine optimale Lösung liefert:

Man sortiert die Jobs aufsteigend nach Startzeitpunkt und fügt sie in dieser Reihenfolge ein (wenn sie mit den bisher gewählten Jobs kompatibel sind).

Geben Sie Ihr Gegenbeispiel als eine Liste von Paaren $(s_1, f_1), (s_2, f_2), \dots$ der Startzeiten s_i und Endzeiten f_i von Jobs an.

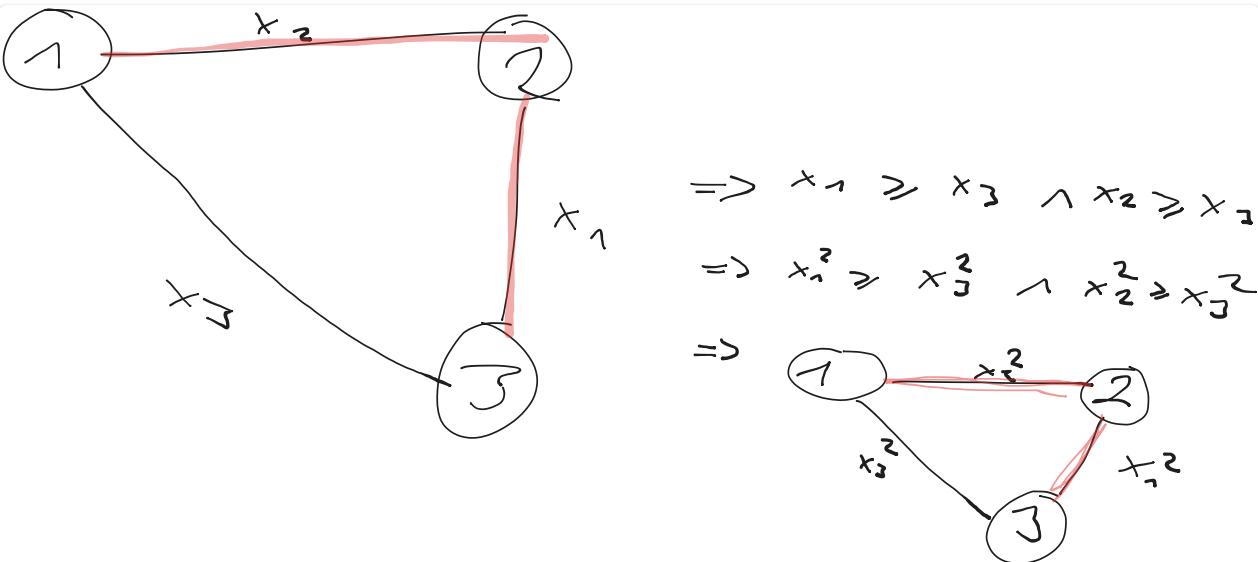


$(0, 10), (1, 4), (4, 7), (7, 8), (8, 10)$

e)

(4 Punkte) Angenommen, G ist ein Graph mit positiven Kantengewichten c_e und T ein minimaler Spannbaum von G . Sei G' der Graph mit der gleichen Knoten- und Kantenmenge wie G und quadrierten Kantengewichten $c'_e = c_e^2$. Ist T ein minimaler Spannbaum von G' ? Begründen Sie Ihre Antwort mit einer Beweisskizze oder einem Gegenbeispiel.

ja weil sich die Verhältnisse nicht ändern



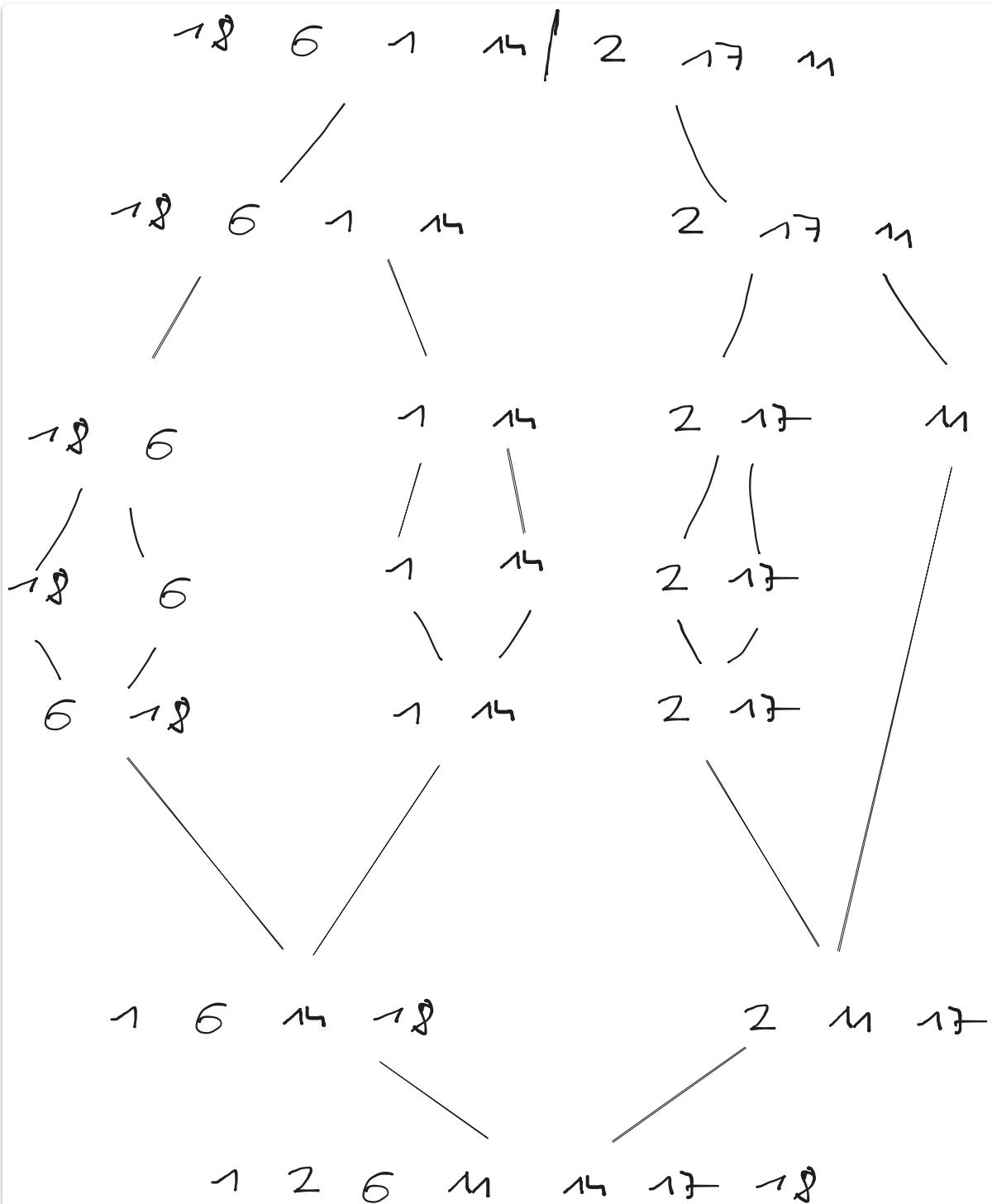
Da die Gewichte positiv sind, kann man das machen.

A4 - Sortierverfahren und Divide-and-Conquer

Stoff: 5. Divide and Conquer

a)

(8 Punkte) Benutzen Sie **Mergesort**, um eine aufsteigende Sortierung des Arrays [18, 6, 1, 14, 2, 17, 11] zu berechnen. Geben Sie, wie Sie das in der Vorlesung und Übung kennen gelernt haben, alle Zwischenschritte an.



b)

- (Q1) Für Arrays mit n Zahlen im Bereich 0 bis z mit $z < n$ hat Countsort eine bessere asymptotische **Worst-Case** Laufzeit als Insertionsort.
- Wahr Falsch
- (Q2) Jedes allgemeine Sortierverfahren benötigt zum Sortieren von n paarweise verschiedenen Schlüsseln mindestens $\Omega(n \log(n))$ Laufzeit im **Best-Case**.
- Wahr Falsch
- (Q3) Im **Worst-Case** benötigt Mergesort asymptotisch gesehen mehr zusätzlichen Speicher als Quicksort.
- Wahr Falsch

2. Bestcase $O(n)$

3. Beide $O(n)$

c)

(6 Punkte) Bei folgender Mergesort-Variante wird das Eingabearray in jedem Rekursionsschritt in drei statt zwei Teile geteilt:

```
ThreewayMergesort(A, l, r):
if l < r then
    m1 ← l + ⌊(r - l)/3⌋
    m2 ← l + 2 · ⌊(r - l)/3⌋ + 1
    ThreewayMergesort(A, l, m1)
    ThreewayMergesort(A, m1 + 1, m2)
    ThreewayMergesort(A, m2 + 1, r)
    ThreewayMerge(A, l, m1, m2, r)
```

Sei $C(n)$ die Anzahl der Schlüsselvergleiche in ThreewayMergesort bei einer Eingabegröße n . Geben Sie eine Rekursionsgleichung zur Berechnung von $C(n)$ an. Zur Vereinfachung nehmen wir an, dass die Funktion ThreewayMerge drei Teilarrays mit einer Gesamtlänge von n immer mit $3n$ Schlüsselvergleichen verschmilzt.

$$C(n) = \begin{cases} 0 & \text{wenn } n \leq 1 \\ C(\lceil n/3 \rceil) + C(\lceil n/3 \rceil) + C(\lfloor n/3 \rfloor) + 3n & \text{wenn } n > 1 \end{cases}$$

Erläuterung:

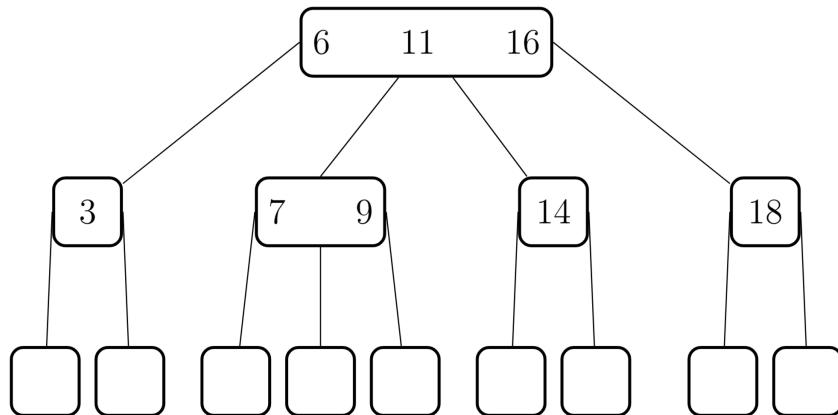
- $C(n)$: Anzahl der Schlüsselvergleiche für ein Array der Größe n .
- Basisfall ($n \leq 1$): Keine Vergleiche für leere oder einelementige Arrays.
- Rekursiver Fall ($n > 1$):
 - Drei rekursive Aufrufe für die drei Teile (Größe ungefähr $n/3$).
 - $3n$ Vergleiche für das anschließende Verschmelzen der drei sortierten Teile.

A5 - Suchbäume und Hashing

Stoff: 6. Suchbäume und 7. Hashing

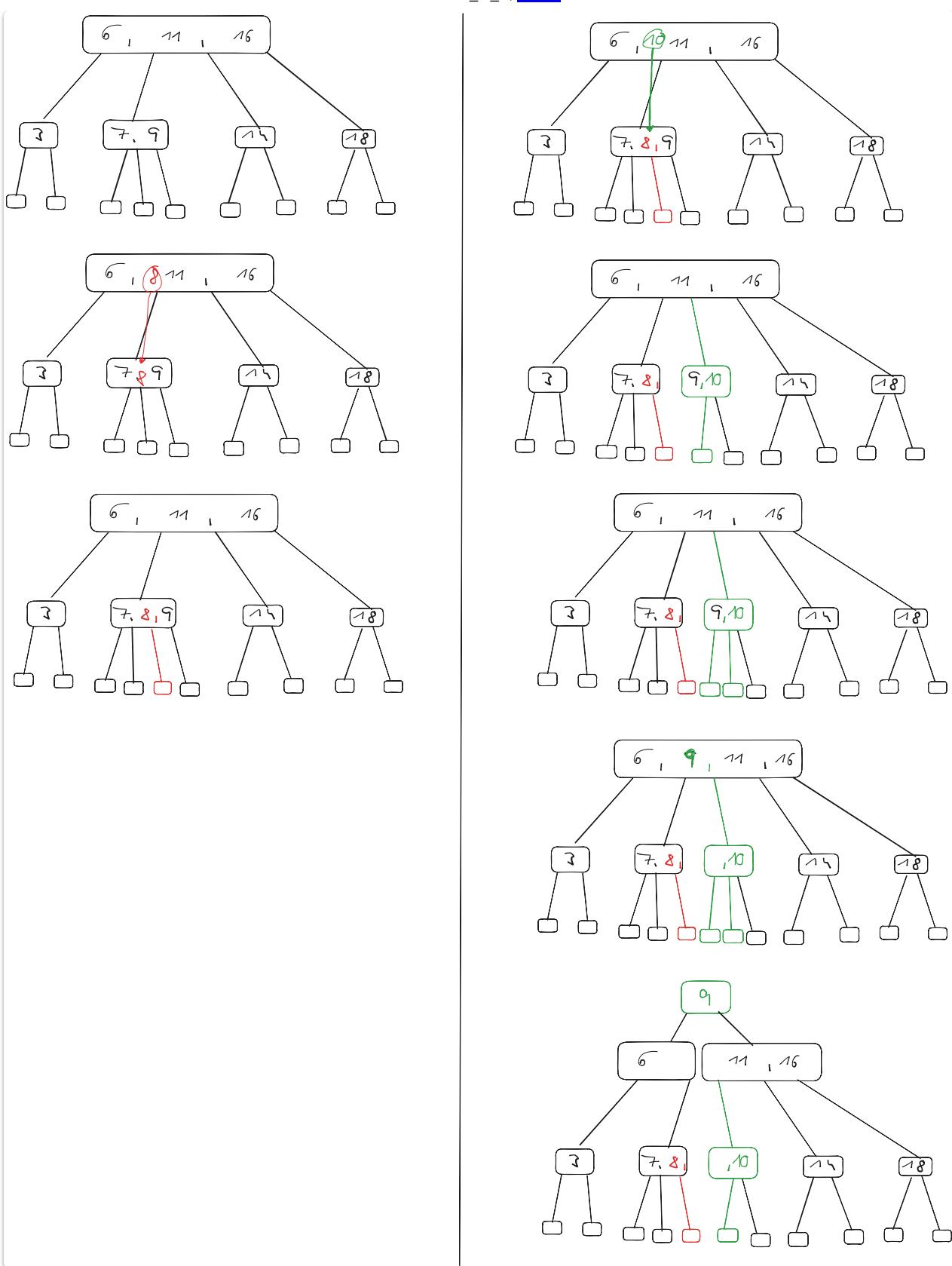
a)

Gegeben ist folgender **B-Baum** der Ordnung 4:



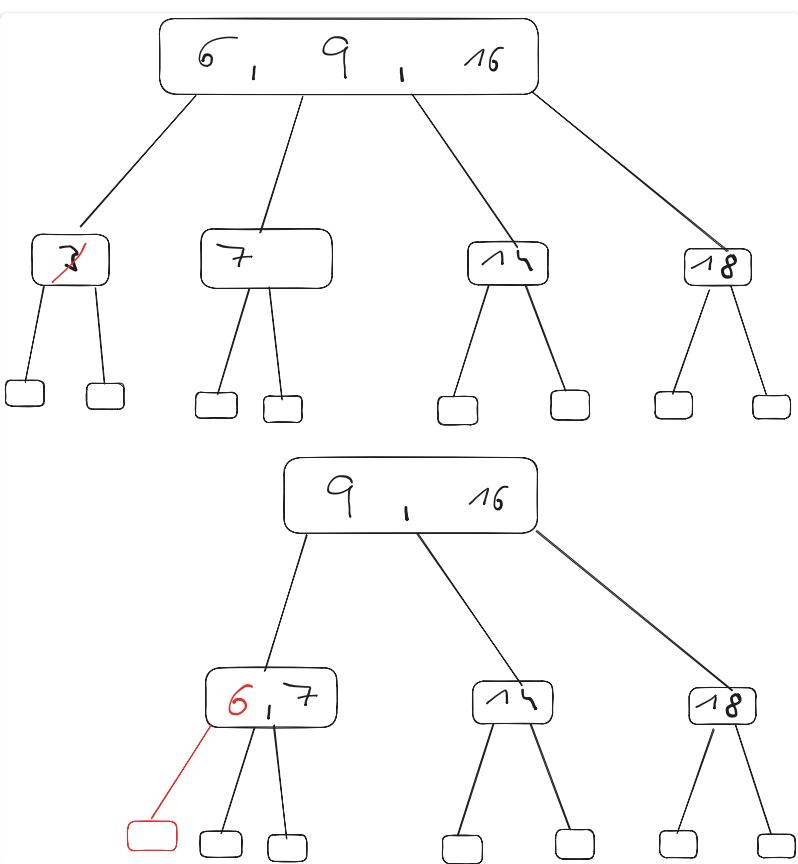
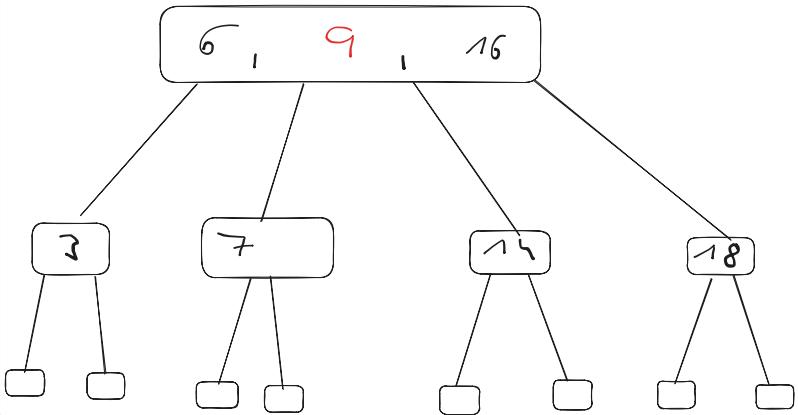
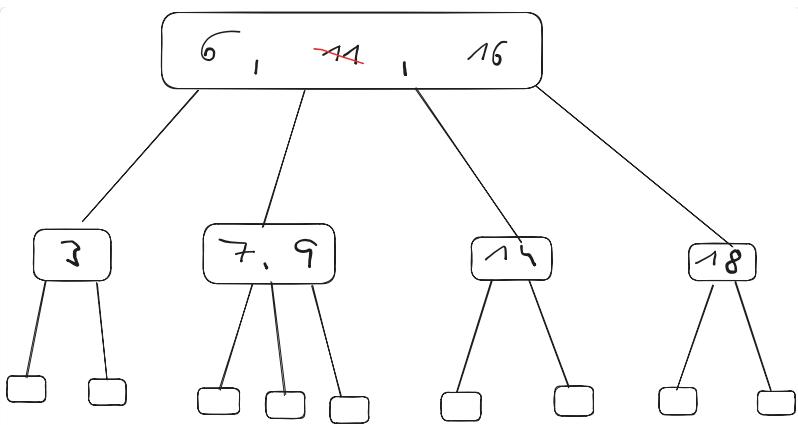
(i)

- (i) (6 Punkte) Fügen Sie die Schlüssel 8 und 10 in dieser Reihenfolge nacheinander in diesen B-Baum ein. Zeichnen Sie den vollständigen Baum direkt nach dem Einfügen jeden Elementes.



(ii)

(6 Punkte) Löschen Sie aus dem **ursprünglichen**, gegebenen B-Baum die Schlüssel 11 und 3 nacheinander in dieser Reihenfolge. Zeichnen Sie den vollständigen Baum direkt nach dem Löschen jeden Elementes.



b)

Gegeben ist folgende Hashtabelle, die Double-Hashing mit $h_1(k) = k \bmod 7$ und $h_2(k) = (k \bmod 6) + 1$ verwendet:

Position	0	1	2	3	4	5	6
Schlüssel	8	1	9	3	-3	12	21

(i)

(4 Punkte) Geben Sie **eine** Reihenfolge an, in der die enthaltenen Elemente in die Hashtabelle eingefügt worden sein könnten, um diesen Zustand ohne irgendwelche anderen Schritte zu erreichen.

$8 \bmod 7 = \cancel{1}$	$1, 9, 3, -3, 12, 8, 21$
$1 \bmod 7 = 1$	
$9 \bmod 7 = 2$	
$3 \bmod 7 = 3$	
$-3 \bmod 7 = 4$	
$12 \bmod 7 = 5$	
$21 \bmod 7 = \cancel{0}$	

(ii)

(4 Punkte) Wie viele Schritte benötigt eine erfolgreiche Suche durchschnittlich in der obigen Hashtabelle?

Hinweis: Die Suche nach Schlüssel 1, benötigt einen Schritt.

$$1 + 1 + 1 + 1 + 1 = 5 \text{ für die ohne Kollision}$$

$$5 + 3 = 8 \text{ für die mit 8}$$

$$8 + 6 = 14 \text{ für die mit 21}$$

Also $\frac{14}{7} = 2$ braucht man durchschnittlich