Datenstrukturen und Algorithmen: Blatt 10

Bernhard Dietrich (6256800)
Lars Fernhomberg (6256030)
Sebastian Kniesburges (6257120)
Marcus Köthenbürger (6258550)

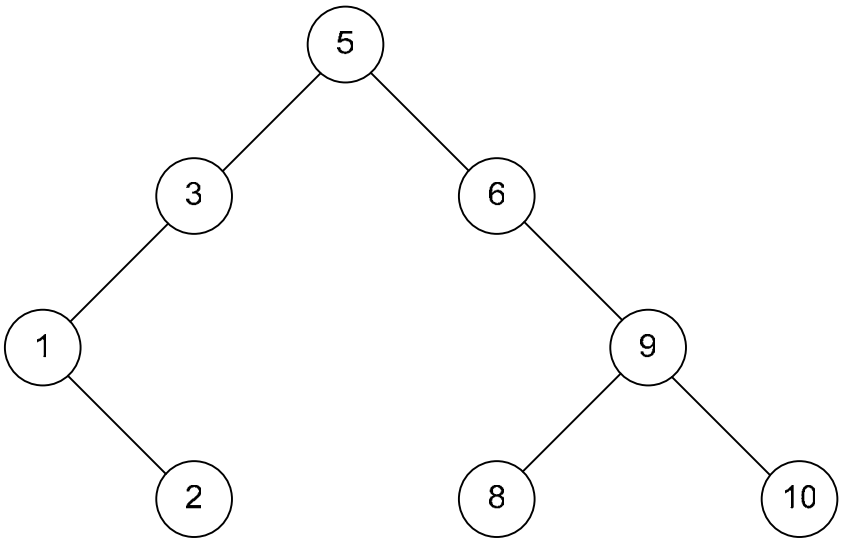
Übungsgruppe 11
Mittwoch, 11:00-13:00 Uhr in D1.303
Matthias Ernst

Aufgabe	1	2	3	4	Σ	Korrektor
Punkte						
von	4	6	6	4	20	

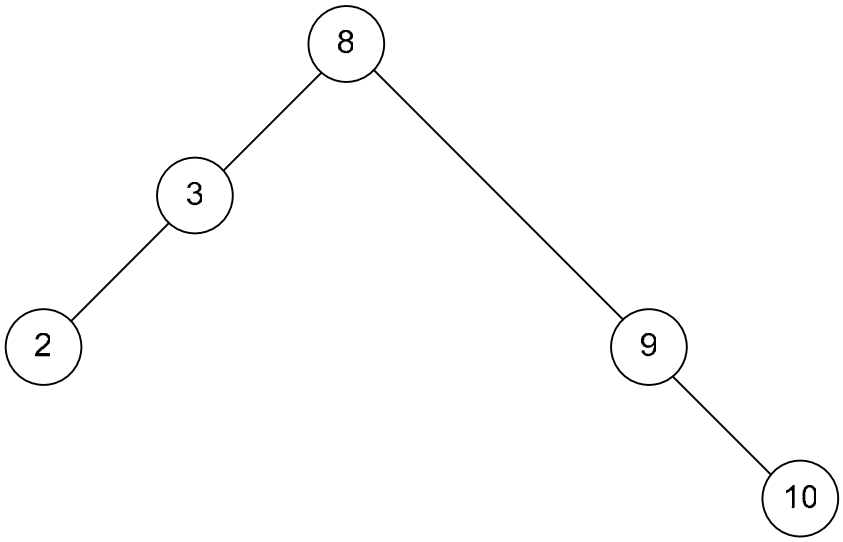
Aufgabe 1

Wir betrachten binäre Suchbäume.

1. In einen zunächst leeren Suchbaum werden nacheinander Knoten mit den Schlüsseln 5, 3, 6, 1, 2, 9, 8, 10 eingefügt. Wie sieht der Suchbaum nach diesen Operationen aus?



2. Aus dem Suchbaum aus dem ersten Teil der Aufgabe werden die Knoten mit den Schlüsseln 6, 5, 1 nacheinander entfernt. Wie sieht der Suchbaum nach diesen Operationen aus?



Aufgabe 2

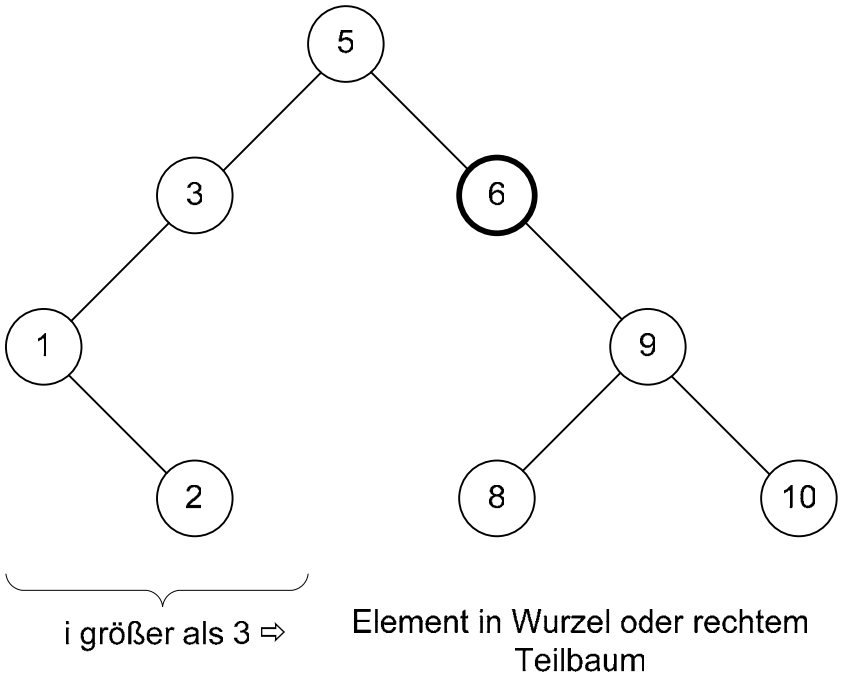
Gegeben eine Menge von Objekten mit unterschiedlichen Schlüsseln aus $\mathbb{N}$, soll die Operation SELECT(i) das Objekt mit dem i-kleinsten Schlüssel bestimmen. Erweitern Sie die Datenstruktur der binären Bäume so, dass sie neben den Wörterbuchoperationen INSERT, SEARCH, usw. auch die Operation SELECT unterstützt. Beschreiben Sie dazu informell, wie die Operation SELECT realisiert werden soll und wie ggf. die Operationen INSERT und DELETE angepasst werden müssen. Die Laufzeit für SELECT soll $O(h)$ sein, wobei h die Höhe des Baums T ist.

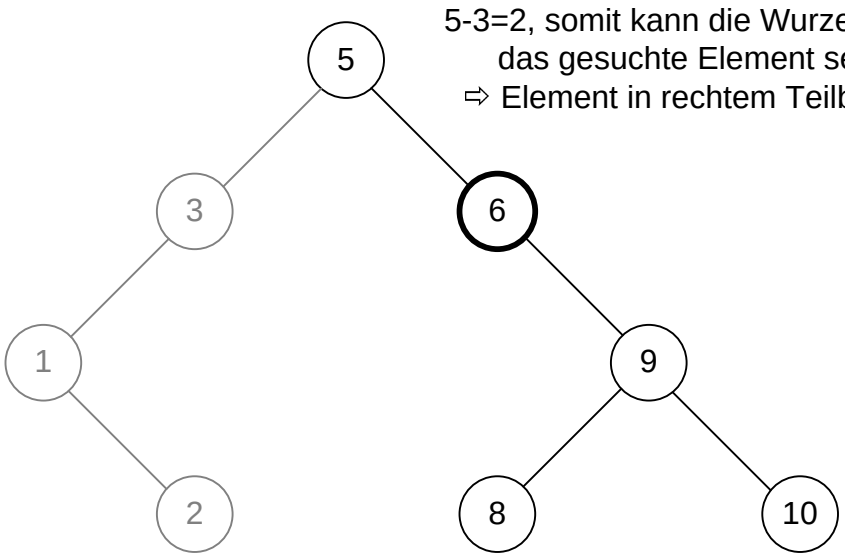
Um diese Aufgabe zu lösen, muss zusätzlich in jedem Knoten des Baums gespeichert werden, wie viele Knoten sich in dem Teilbaum mit dem entsprechenden Knoten als Wurzel befinden. Um dieses zu erreichen müssen die Operationen INSERT und DELETE so angepasst werden, dass sie nach Beendigung des Einfüge- bzw. Löschvorgangs bei sämtlichen Elternknoten (also sowohl die unmittelbaren Elternknoten als auch die Großelternknoten, Urgroßelternknoten, … bis zur Wurzel des Baums) den neu eingeführten Wert für die Anzahl der Knoten in den Teilbäumen um je eins erhöhen bzw. um je eins verringern.

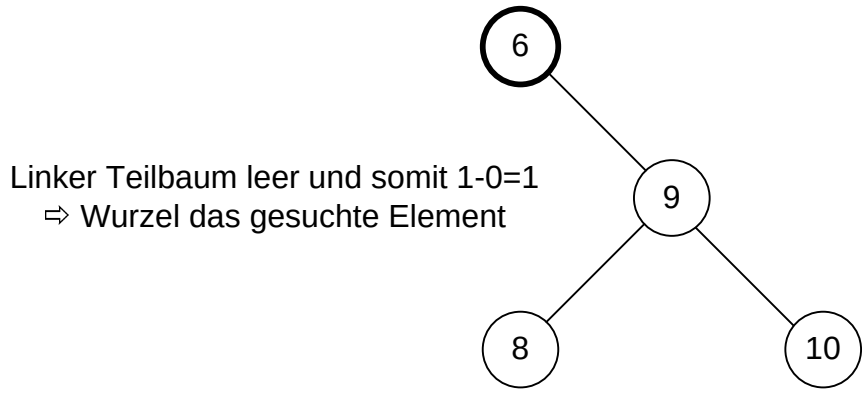
Der SEARCH-Algorithmus überprüft nun, ob i größer als die Anzahl der Elemente im linken Teilbaum ist.

Wenn dies der Fall ist, bedeutet dies, dass das gesuchte Element im rechten Teilbaum oder in der Wurzel vorliegt. Ermittelt die Differenz $i - \text{Anzahl im linken Teilbaum}$ den Wert 1, so bedeutet dies, dass die Wurzel des aktuellen Baums das gesuchte Element ist und der Algorithmus ist fertig. Ermittelt die Differenz einen Wert größer als 1, so bedeutet dies, dass das gesuchte Element im rechten Teilbaum liegt und die Überprüfung erneut gestartet werden muss, wobei die Wurzel des rechten Teilbaums die Wurzel des Baums wird, der in diesem nächsten Schritt beachtet wird. Dabei wird nicht mehr mit dem Wert i operiert sondern mit der Differenz $i - \text{Anzahl im linken Teilbaum-1}$, welche alle bisher ausgeschlossenen Knoten – die Knoten im linken Teilbaum sowie die Wurzel – subtrahiert und somit die Position des Knotens an die veränderte Struktur des Baums anpasst.

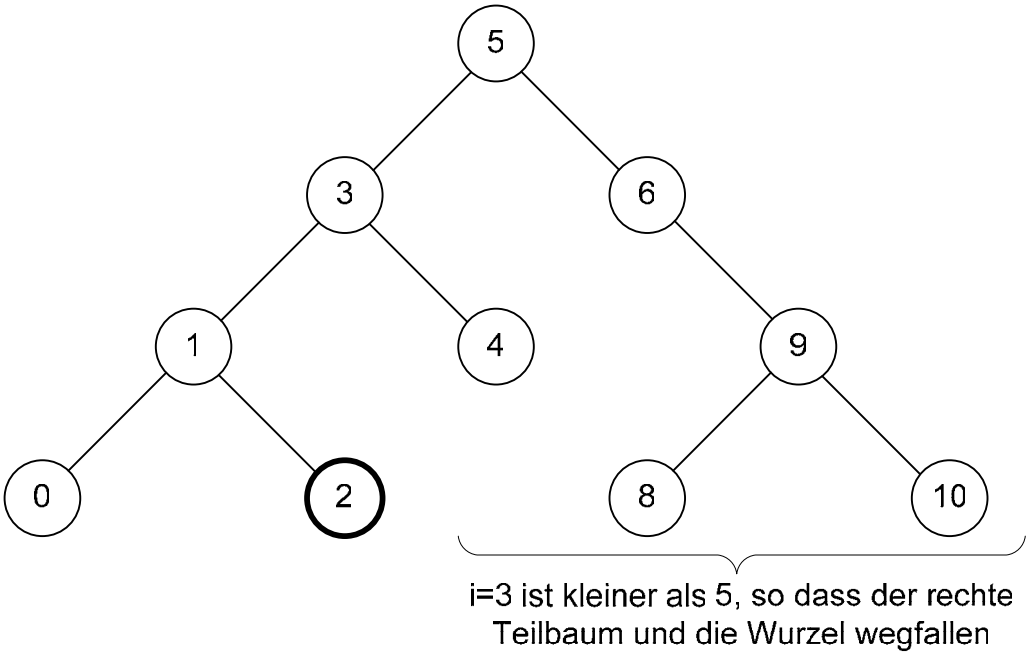
Illustration mit $i = 5$:

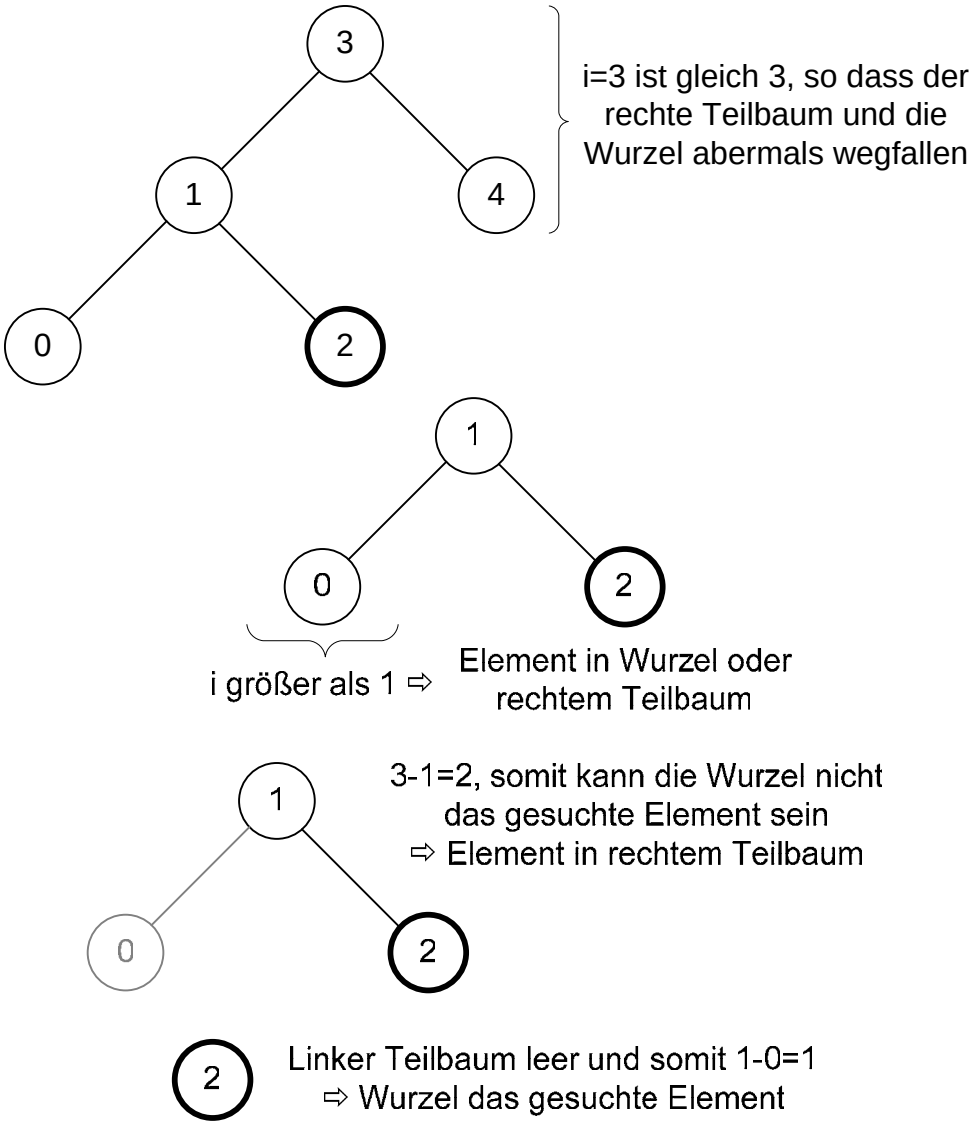






Wenn i aber kleiner als die Anzahl der Elemente im linken Teilbaum oder gleich mit dieser ist, so bedeutet dies, dass sich das gesuchte Element im linken Teilbaum befindet und die Wurzel sowie der rechte Teilbaum für das Suchproblem irrelevant sind. Die Suche wird somit nur für den linken Teilbaum mit dem linken Kind der Wurzel als neue Wurzel weiter ausgeführt.
Illustration für $i = 3$:





Während jedes Schleifendurchlaufs führt der Algorithmus lediglich Operationen mit konstantem Zeitbedarf aus, so dass jeder Schleifendurchlauf in konstanter Zeit $O(1)$ durchgeführt wird. Die Anzahl der Schleifendurchläufe hängt von der Höhe des Baums ab, wobei bei jedem Schleifendurchlauf die Höhe um eine Ebene verringert wird, da jeweils ein kompletter Teilbaum sowie die entsprechende Wurzel ausgeschlossen werden. Insgesamt kann die Schleife also nur h mal durchlaufen werden. Es ergibt sich somit die Laufzeit von $O(h)$.

Aufgabe 3

1. Zeigen Sie, dass ein AVL-Baum mit Höhe h mindestens $F_h - 1$ innere Knoten enthält. Dabei ist F_i die i-te Fibonaccizahl und es gilt $F_0 = 1$, $F_1 = 2$ und $F_{i+1} = F_i + F_{i-1}$ für $i \geq 1$.
   Wir definieren $A(h) = \text{Anzahl innere Knoten bis Ebene } h$. Die Wurzel des Baums hat zwei Teilbäume, wobei zwischen beiden Teilbäumen die Höhe nur maximal um eine Ebene differenziert. Der eine Teilbaum hat $A(h-1)$ innere Knoten und der andere Teilbaum hat somit mindestens $A(h-2)$ innere Knoten.

Die Gesamtanzahl aller Knoten lässt sich somit
durch $A(h) =$ Knoten größerer Teilbaum + Knoten kleiner Teilbaum + Wurzel
$= A(h-1) + A(h-2) + 1$ beschreiben.

Induktionsanfang: Anschaulich sieht man, dass für $h = 0$ $A(0) = 0 = F_0 - 1$ gilt sowie für $h = 1$ $A(1) = 1 = F_1 - 1$ gilt.

Induktionsvorraussetzung: Für ein beliebiges $n \in \mathbb{N}$ gelte $A(h) = F_h - 1$.

Induktionsschluss: $h \to h+1$

$$\begin{aligned} A(h+1) &= A(h) + A(h-1) + 1 \\ &\overset{\text{I.V.}}{=} F_h - 1 + F_{h-1} - 1 + 1 \\ &= F_h + F_{h-1} - 1 \\ &= F_{h+1} - 1 \end{aligned}$$

Somit gilt für minimale AVL-Bäume bei Höhe h: $A(h) = F_h - 1$

2. Zeigen Sie, dass für die Höhe h eines AVL-Baums T mit n Knoten gilt $h = O(\log(n))$.
Sie dürfen hierbei benutzen, dass es eine Konstante $c > 1$ gibt, so dass $F_h = \Omega(c^h)$.

Aus $F_h = \Omega(c^h)$ folgt $F_h \geq a \cdot c^h + b - 1$ und somit $a \cdot c^h + b - 1 \leq F_h - 1 \leq n$.

$$a \cdot c^h + b - 1 \leq n$$

$$\overset{\text{mit } d = b-1}{\Leftrightarrow} \quad c^h \leq \frac{n-d}{a}$$

$$h \leq \log_c\left(\frac{n-d}{a}\right)$$

$$= \log_c(n-d) - \log_c(a)$$

$$= \frac{\log_2(n-d)}{\log_2 c} - \frac{\log_2(a)}{\log_2 c}$$

Da $\log_2 c$ und $\log_2(a)$ konstant, folgt

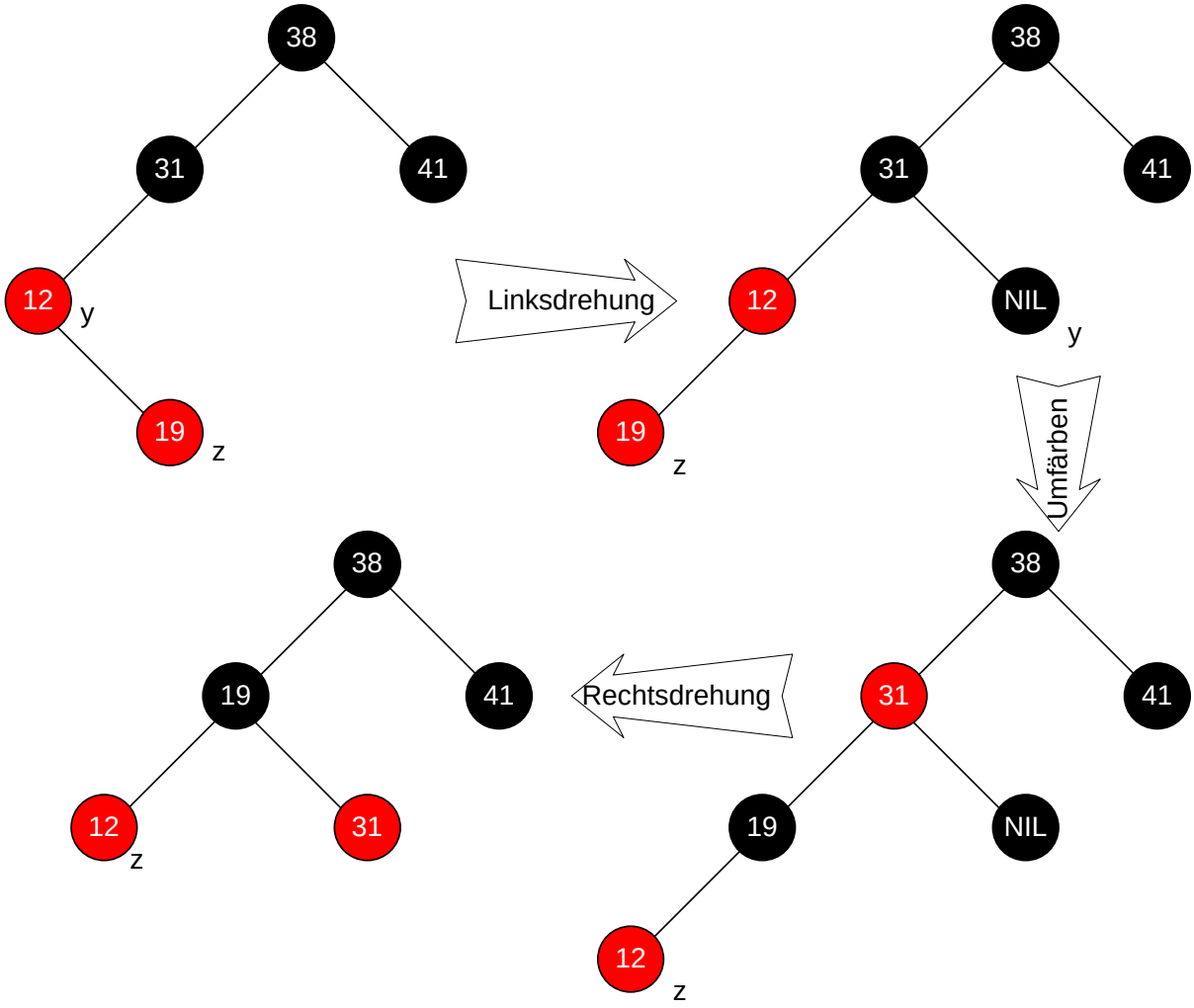
$$h \leq \log(n-d)$$

Da $n - d \leq n$ folgt somit $O(\log(n))$

Aufgabe 4

Fügen Sie in den gegebenen Baum mithilfe der Algorithmen RB-INSERT und RB-INSERT-FIXUP Knoten mit den Schlüsseln 19 und 8 ein. Beschreiben Sie hierzu jeweils die einzelnen Zwischenergebnissen des Algorithmus RB-INSERT-FIXUP.

Einfügen der 19



Einfügen der 8