Aufgabe 1

(a)

Die Methode *ss* ruft lediglich die Methode *ssr* auf. Da dabei kein Vergleich durchgeführt wird, entspricht die Laufzeit von *ss* der Laufzeit von *ssr*. In der Methode *ssr* findet zunächst immer ein Vergleich statt, um die Abbruchbedingung zu testen. In der anschließenden **for**-Schleife werden genau k-1 Vergleiche durchgeführt. Anschließend wird *ssr* rekursiv mit k-1 aufgerufen. Wir erhalten also:

$$T(n) = \begin{cases} T(n-1)+n & \text{für } n>1 \\ 1 & \text{für } n=1 \end{cases}$$

Zählt man die Vergleiche für das Hochzählen der Variablen in der **for**-Schleife mit, werden bei k>1 zusätzlich k Vergleiche durchgeführt, man erhält:

$$T(n) = \begin{cases} T(n-1)+2n & \text{für } n>1 \\ 1 & \text{für } n=1 \end{cases}$$

(b)

Wir vermuten als Formel:

$$f(n) = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

Offenbar ist der Induktionsanfang (n=1) erfüllt. Der Induktionsschritt ist:

$$n \to n+1 : T(n+1) = T(n)+n+1 = \frac{n(n+1)}{2} + n + 1 = \frac{n(n+1)+2n+2}{2} = \frac{(n+1)(n+2)}{2}$$

Bei Mitzählen der Vergleiche in der **for**-Schleife ergibt sich:

$$f(n) = 1 + \sum_{i=2}^{n} 2i = 1 + 2\sum_{i=1}^{n} i - 2 = n^2 + n - 1$$

Auch hier ist der Induktionsanfang offenbar erfüllt. Der Induktionschritt ist:

$$\begin{aligned} n \to n+1 : T(n+1) &= T(n) + 2(n+1) = n^2 + n - 1 + 2n + 2 \\ &= (n^2+2n+1)+(n+1)-1 = (n+1)^2 + (n+1) - 1 \end{aligned}$$

(c)

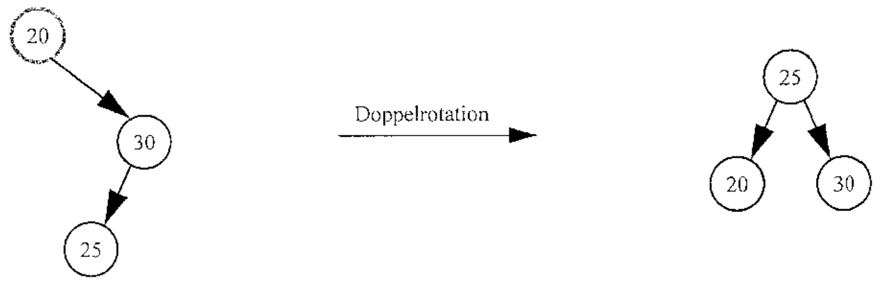
Wir erhalten: $(n^2 + n) / 2 = O(n^2)$

bzw. $(n^2 + n - 1) = O(n^2)$

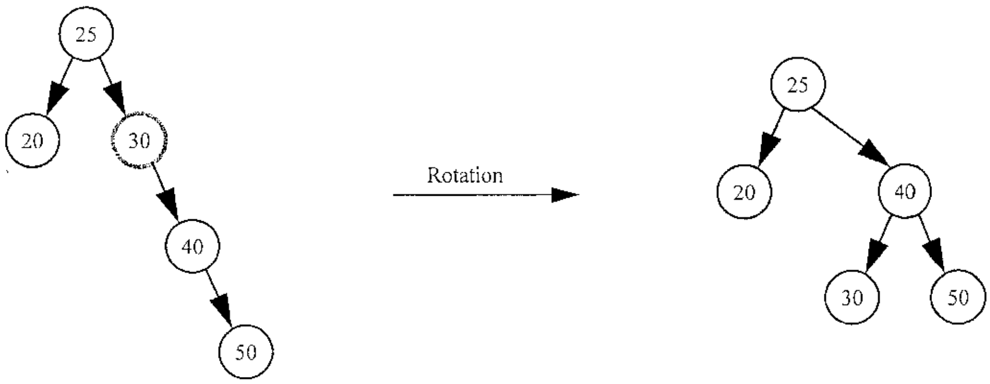
Aufgabe 2

(a)

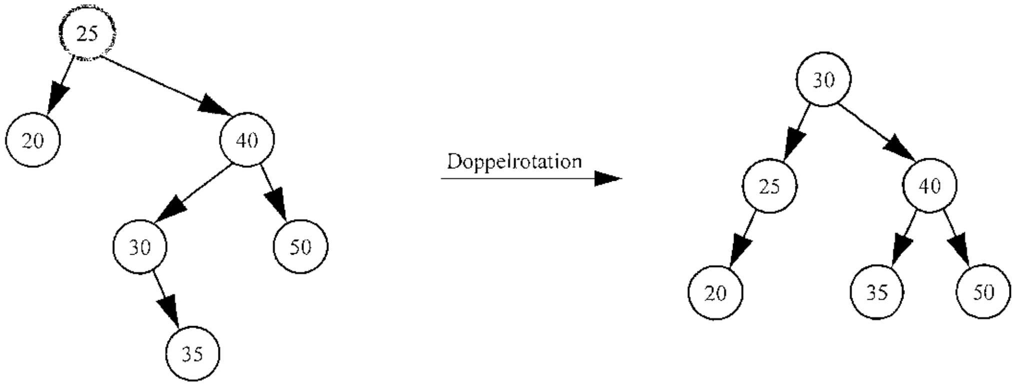
Beim Einfügen der 25 kommt es zu einer Verletzung der Balance am markierten Knoten:



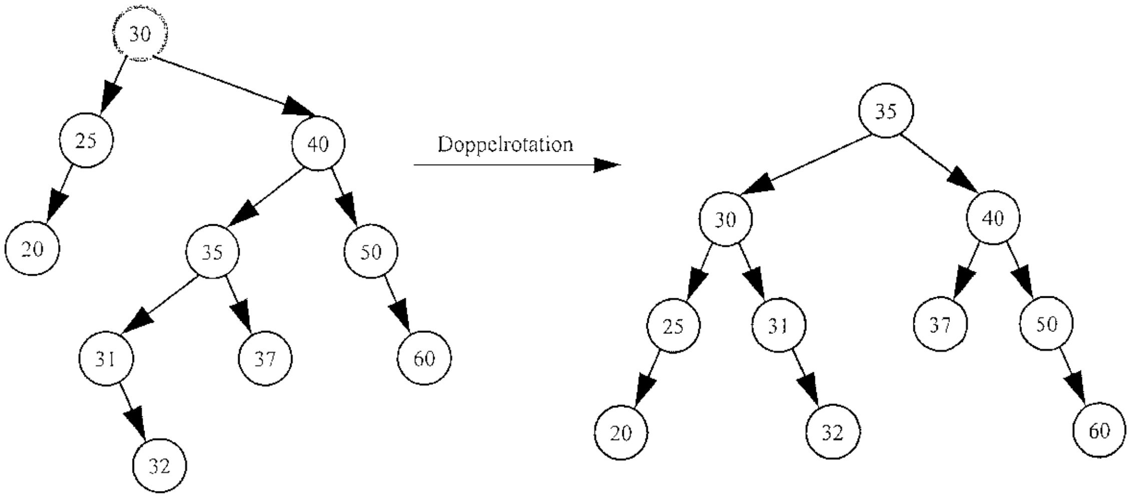
Das Balancekriterium wird beim Einfügen der 50 wieder verletzt:



Das Einfügen der 35 erfordert ebenfalls eine Rebalancierung:

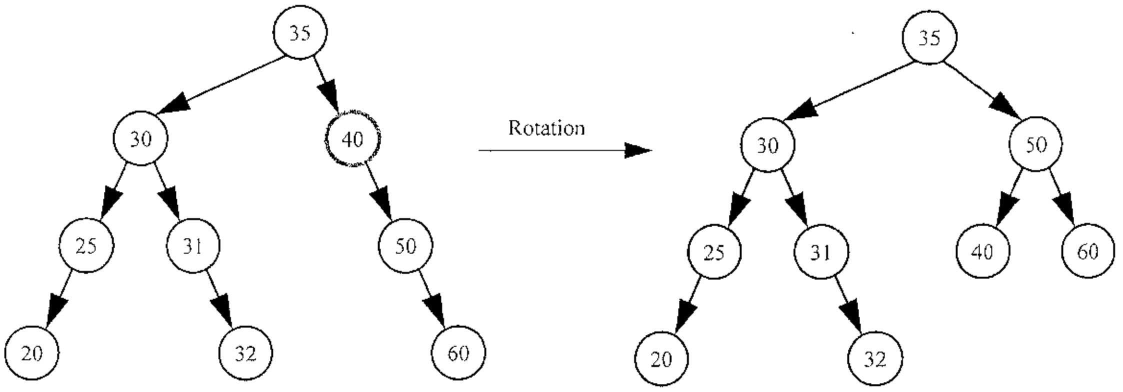


Die nächste Doppelrotation wird durch das Einfügen der 32 notwendig:

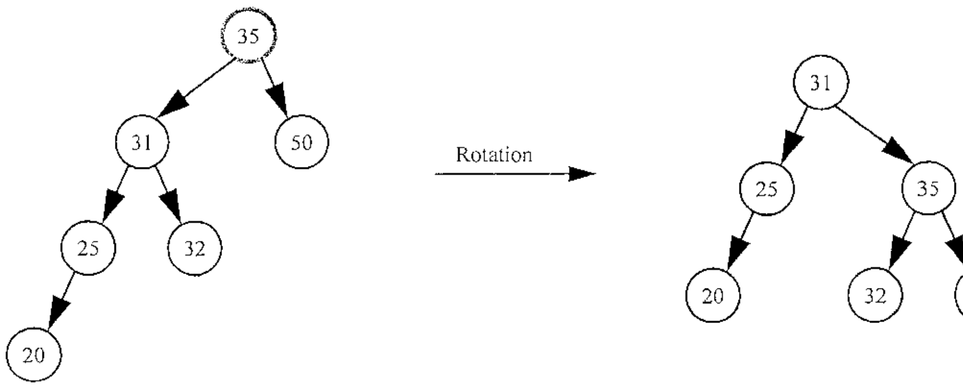


(b)

Das Löschen des Schlüssels 37 erfordert eine Rebalancierung am Knoten 40:



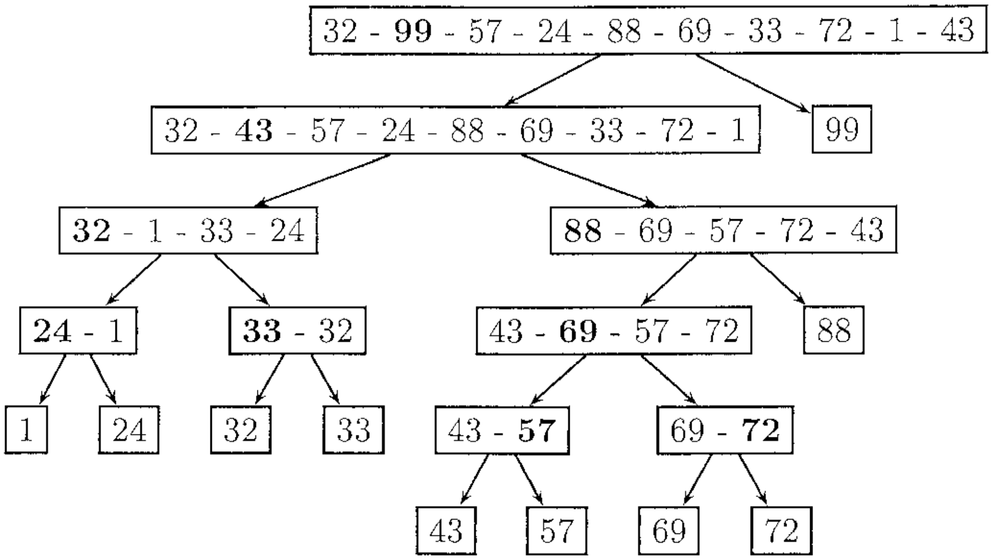
Die Knoten 30 und 40 lassen sich ohne Rebalancierungen löschen. Erst das Löschen des Schlüssels 60 erfordert eine Rebalancierung:



Aufgabe 3

(a)

Im Divideschritt ergibt sich folgende Aufteilung:



(b)

Bei Quicksort findet die eigentliche Arbeit im Divideschritt statt, während im Mergeschritt nur noch die Ergebnislisten aneinander gehängt werden. Bei Mergesort wird im Divideschritt nur das Array in der Mitte aufgespalten und die eigentliche Arbeit erst im Mergeschritt erledigt.

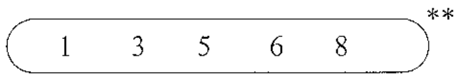
(c)

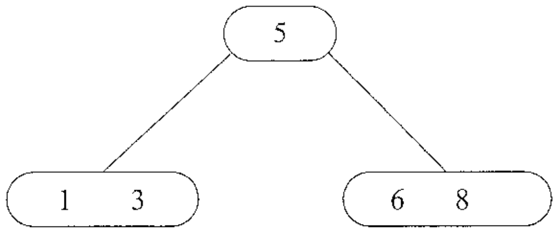
Quicksort und Mergesort sind Ausprägungen des Divide-and-Conquer-Verfahrens.

Aufgabe 4

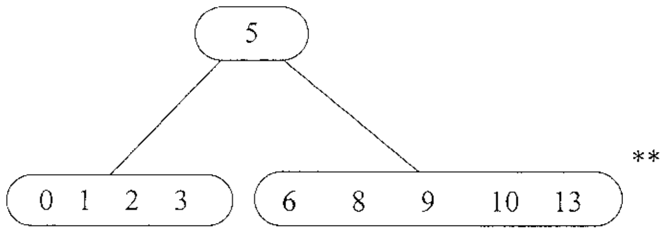
(a)

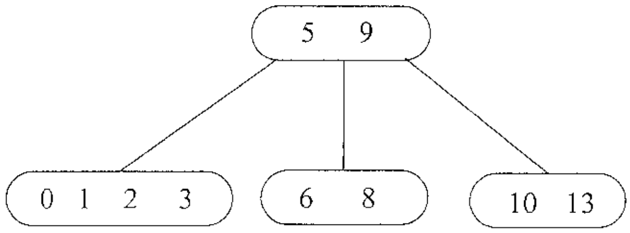
Nach dem Einfügen der Schlüssel 8, 5, 6, 3 und 1 ist die erste Overflow-Behandlung notwendig. Die betroffenen Knoten werden mit ** markiert.



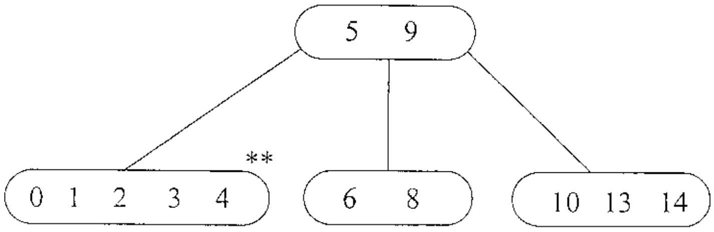


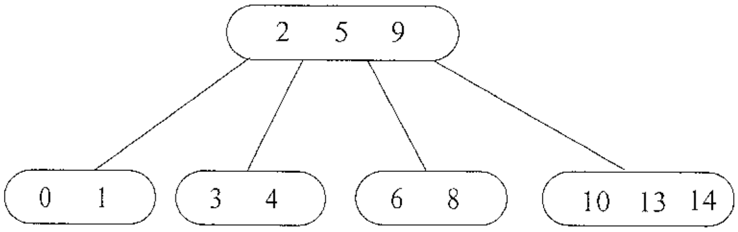
Der nächste Overflow tritt beim Einfügen des Schlüssels 9 auf.



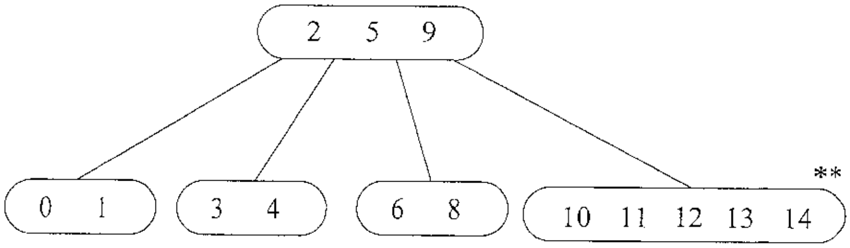


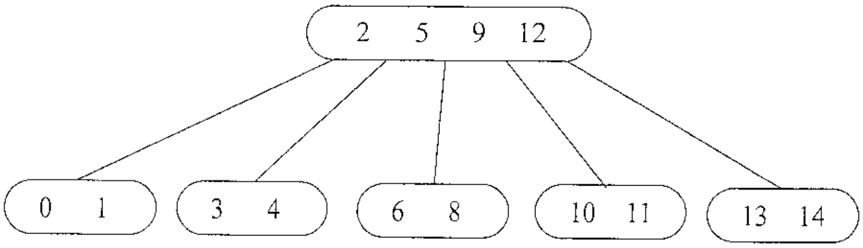
Die nächste Overflow-Behandlung wird mit dem Einfügen des Schlüssels 4 nötig.





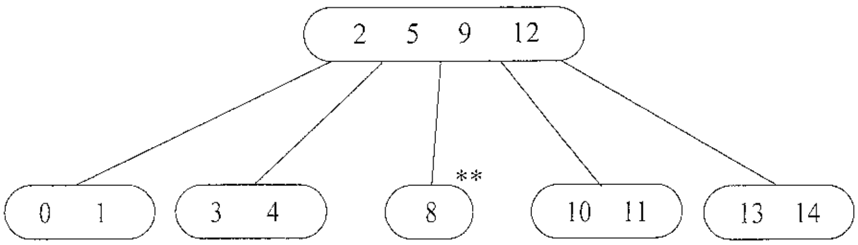
Die letzte Overflow-Behandlung wird mit dem Einfügen des Schlüssels 12 nötig.

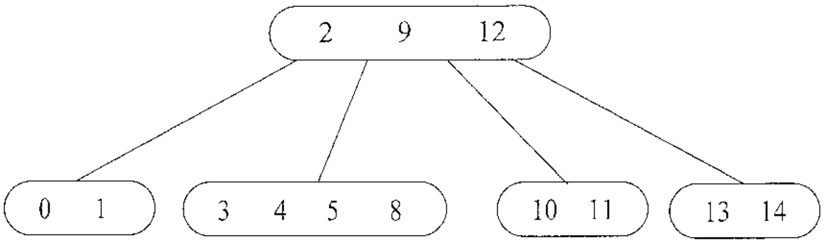




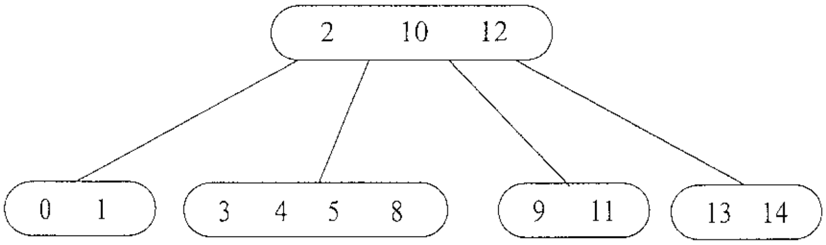
(b)

Der Baum nach dem Löschen des Schlüssels 6:





Der Underflow konnte nicht durch Balancieren beseitigt werden, daher war hier ein Merge notwendig. Beim Löschen des Schlüssels 9 wird dieser zunächst mit einem Schlüssel aus dem Nachfolgerknoten vertauscht:



Nun wird der Schlüssel 9 gelöscht. Es entsteht ein Underflow, der durch Balancieren ausgeglichen werden kann: