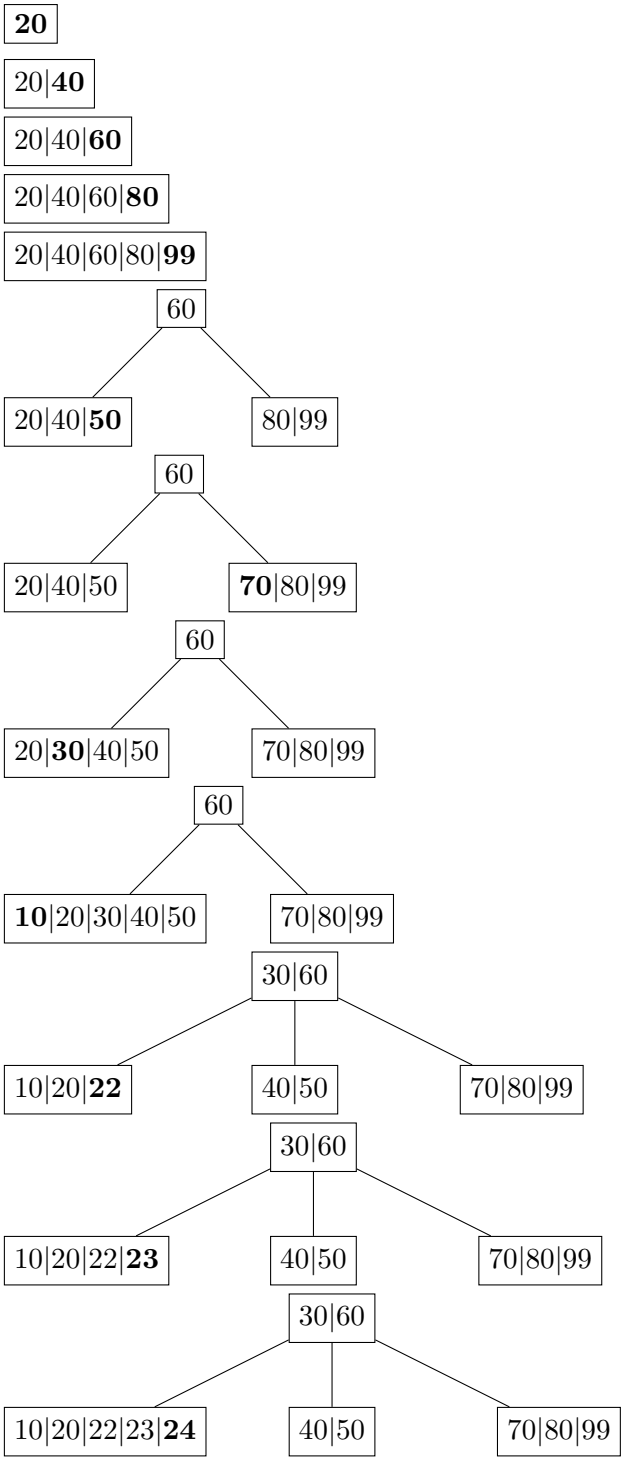
Alexander Neuwirth (439218)
Leonhard Segger (440145)
Jonathan Sigrist (441760)

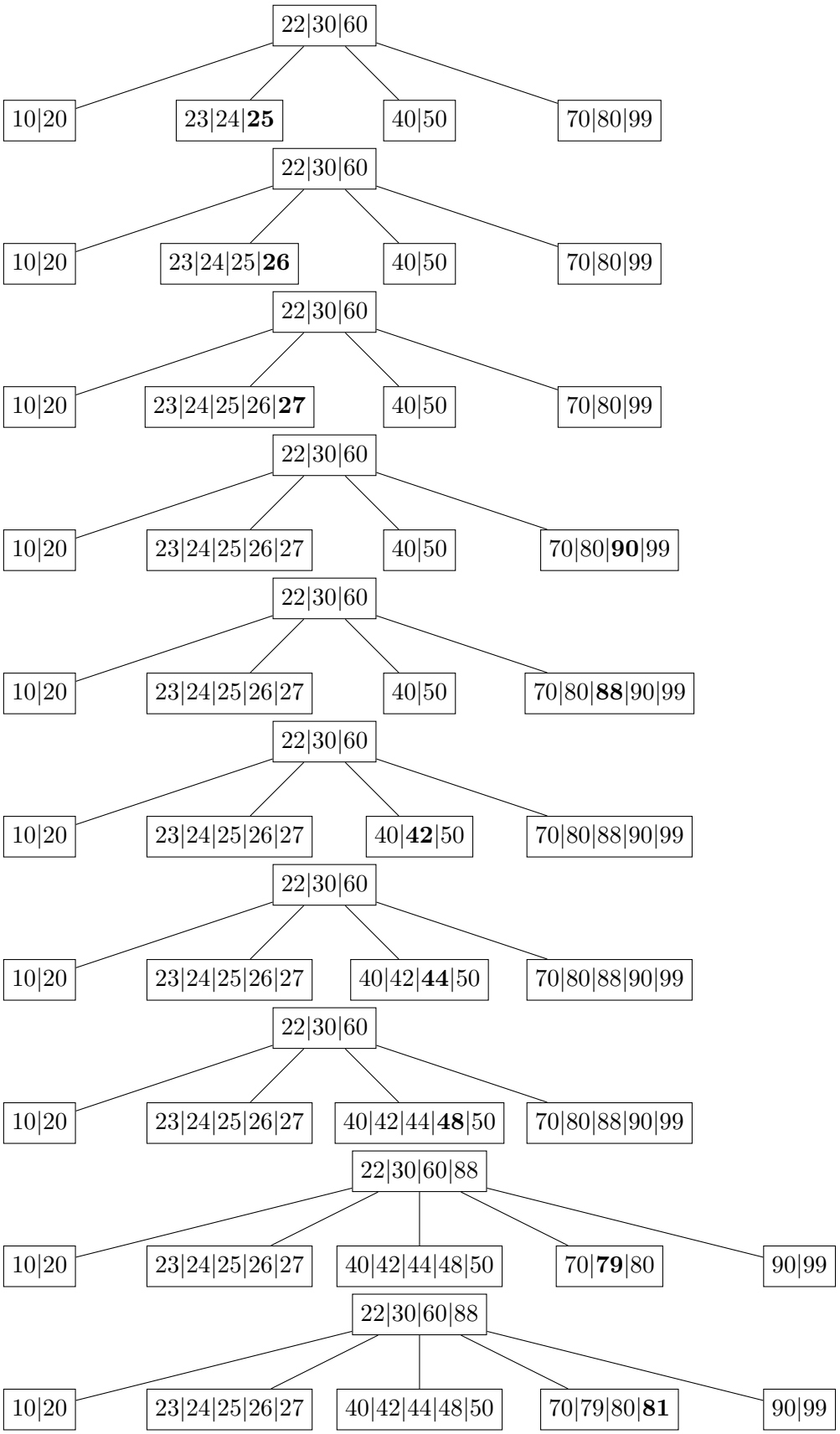
**Aufgabe 36.**
Die Methode hat eine Laufzeit von $\mathcal{O}(n)$, da jedes Element genau einmal aufgerufen wird.

**Aufgabe 37.**

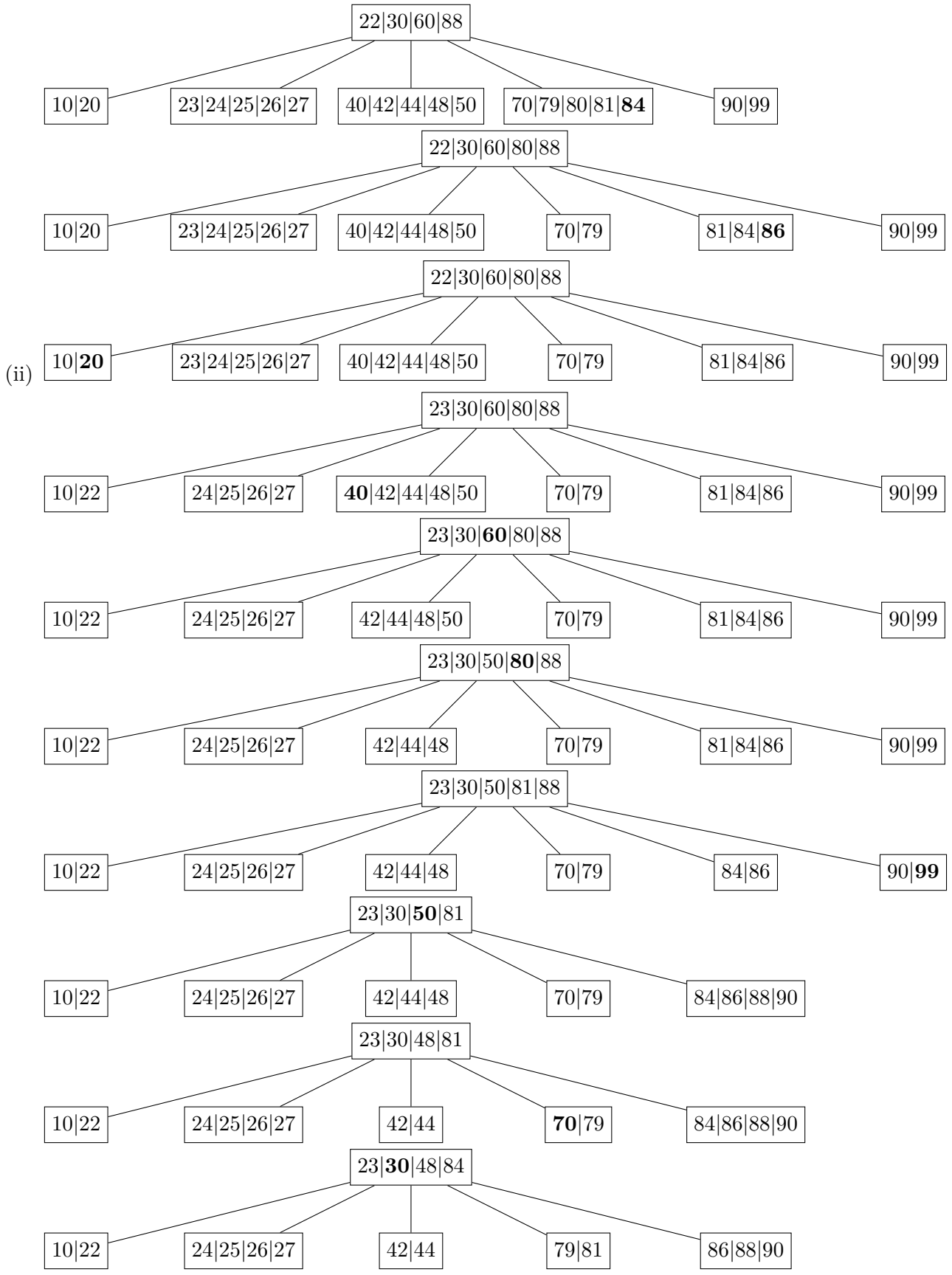
(a) (i)



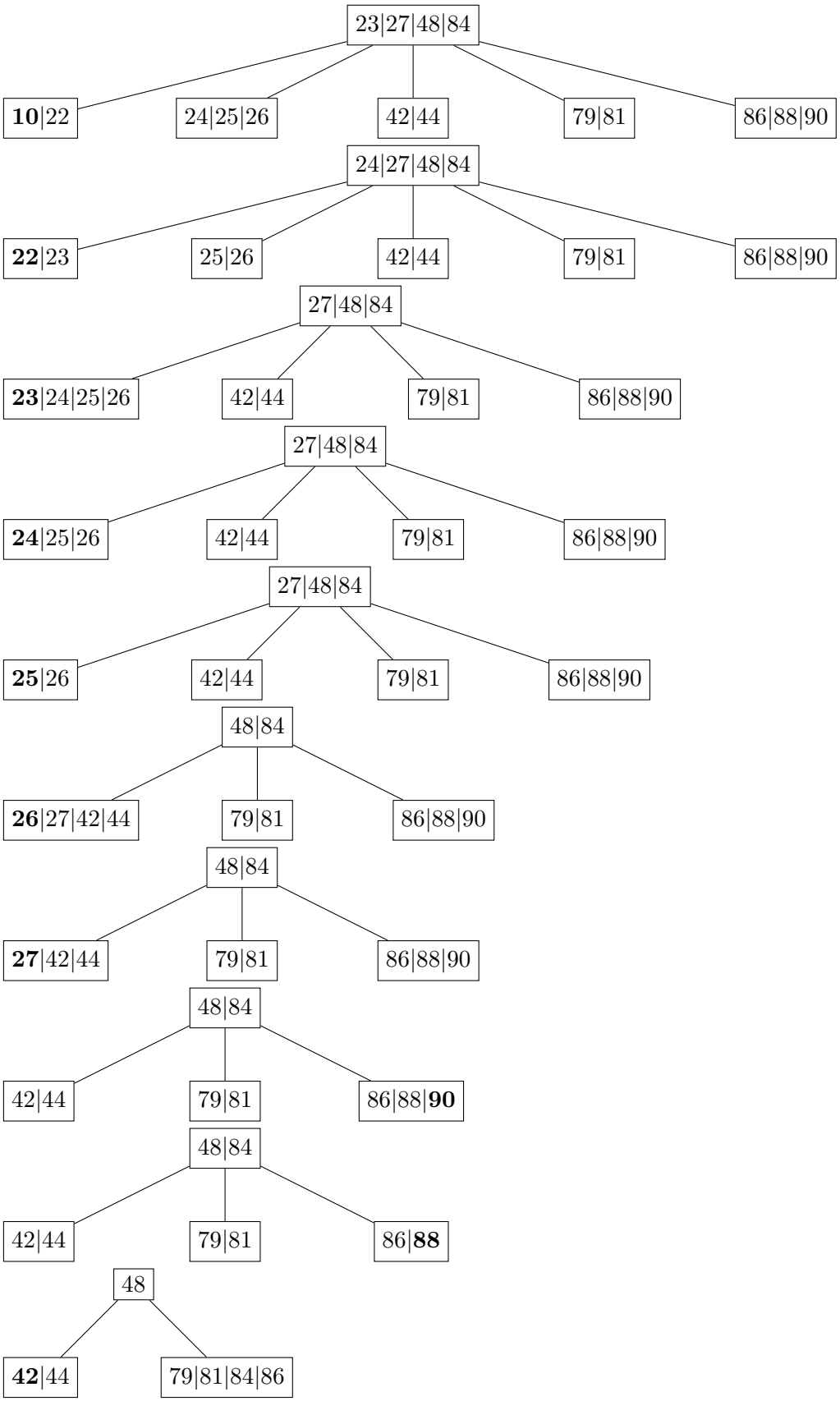
Alexander Neuwirth (439218)
Leonhard Segger (440145)
Jonathan Sigrist (441760)



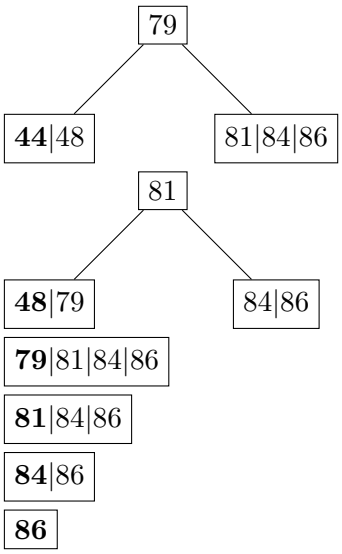
Alexander Neuwirth (439218)
Leonhard Segger (440145)
Jonathan Sigrist (441760)



Alexander Neuwirth (439218)
Leonhard Segger (440145)
Jonathan Sigrist (441760)



Alexander Neuwirth (439218)
Leonhard Segger (440145)
Jonathan Sigrist (441760)




(b) (i) **1.Fall: $\mathcal{B}$ ist Blatt** Der Vorgänger ist der linke Nachbar von $\mathcal{B}$. Falls dieser nicht existiert, ist der Vorgänger der linke Vaterknoten. Falls auch dieser niht existiert, so wiederholt sich dieser Prozess für den Vaterknoten des Vaterknotens und so weiter. Also ist der erste (d. h. $n$ kleinst möglich) linke $n$-fache-Vaterknoten ist der Vorgänger. (außer Minimum)

Der Nachfolger ist dementsprechend der rechte Nachbar. Falls dieser nicht existiert, so ist der rechte Vaterknoten der Nachfolger und so weiter. Also ist der erste ($n$ kleinst möglich) rechte $n$-fache Vaterknoten der Nachfolger. (außer Maximum)

**2.Fall: $\mathcal{B}$ ist innerer Knoten** Vorgänger ist das Maximum des rechten Kinderknotens. Nachfolger das Minimum des linken Kinderknotens.

**Laufzeit:** Das Finden des $\mathcal{B}$-Elementes braucht $\mathcal{O}(\log n)$. Da die Position des Maximums bzw. des Minimums eines Teilbaums bekannt ist, beträgt die Laufzeit dafür $\mathcal{O}(1)$. Im Worst-Case muss der Baum bis zur Wurzel ($\mathcal{O}(\log n)$) geprüft werden, danach ist der Vorgänger/Nachfolger mit $\mathcal{O}(1)$ bekannt. Es folgt eine gesammte Laufzeit von $\mathcal{O}(\log n)$ für die Bestimmung von Vorgänger und Nachfolger.

(ii) Jeder Knoten enthält zusätzlich einen Speicher für Vorgänger und Nachfolger. Falls nun ein neues Element eingefügt wird, werden Vorgänger und Nachfolger dessen bestimmt und deren Speicher des Nachfolgers bzw. des Vorgängers auf das neue Element gesetzt. Auch diese Operation braucht $\mathcal{O}(\log n)$.

Beim Löschen lassen sich die Nachfolger und Vorgänger des zu löschenden Elements auch mit $\mathcal{O}(\log n)$ bestimmen und deren Vorgänger- bzw. Nachfolgervariablen auf einander verweisen lassen. Somit hat auch diese Operation noch eine Laufzeit von $\mathcal{O}(\log n0)$.

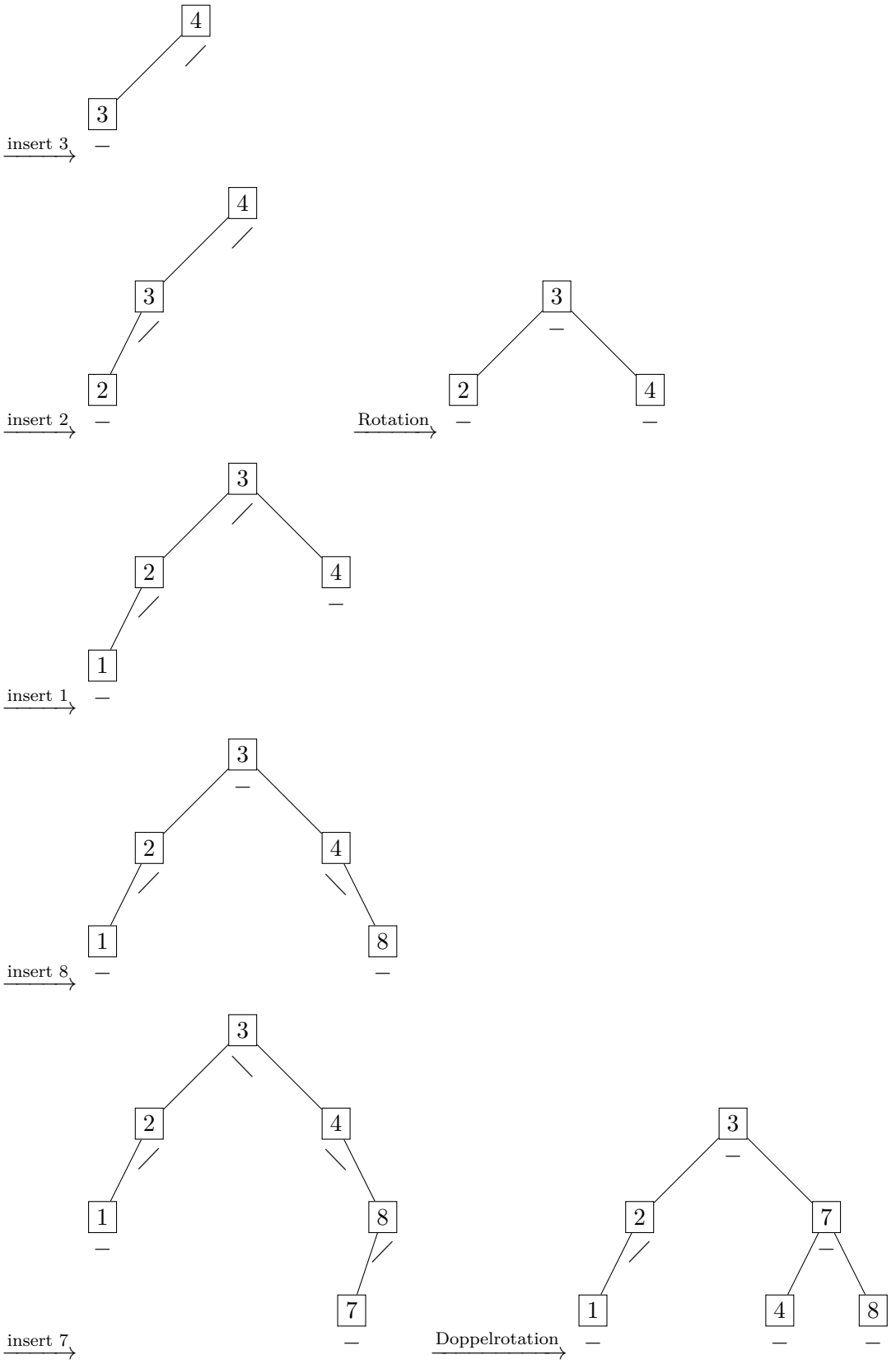
Vorgänger und Nachfolger lassen sich nun mit $\mathcal{O}(1)$ bestimmen.

**Aufgabe 38.**

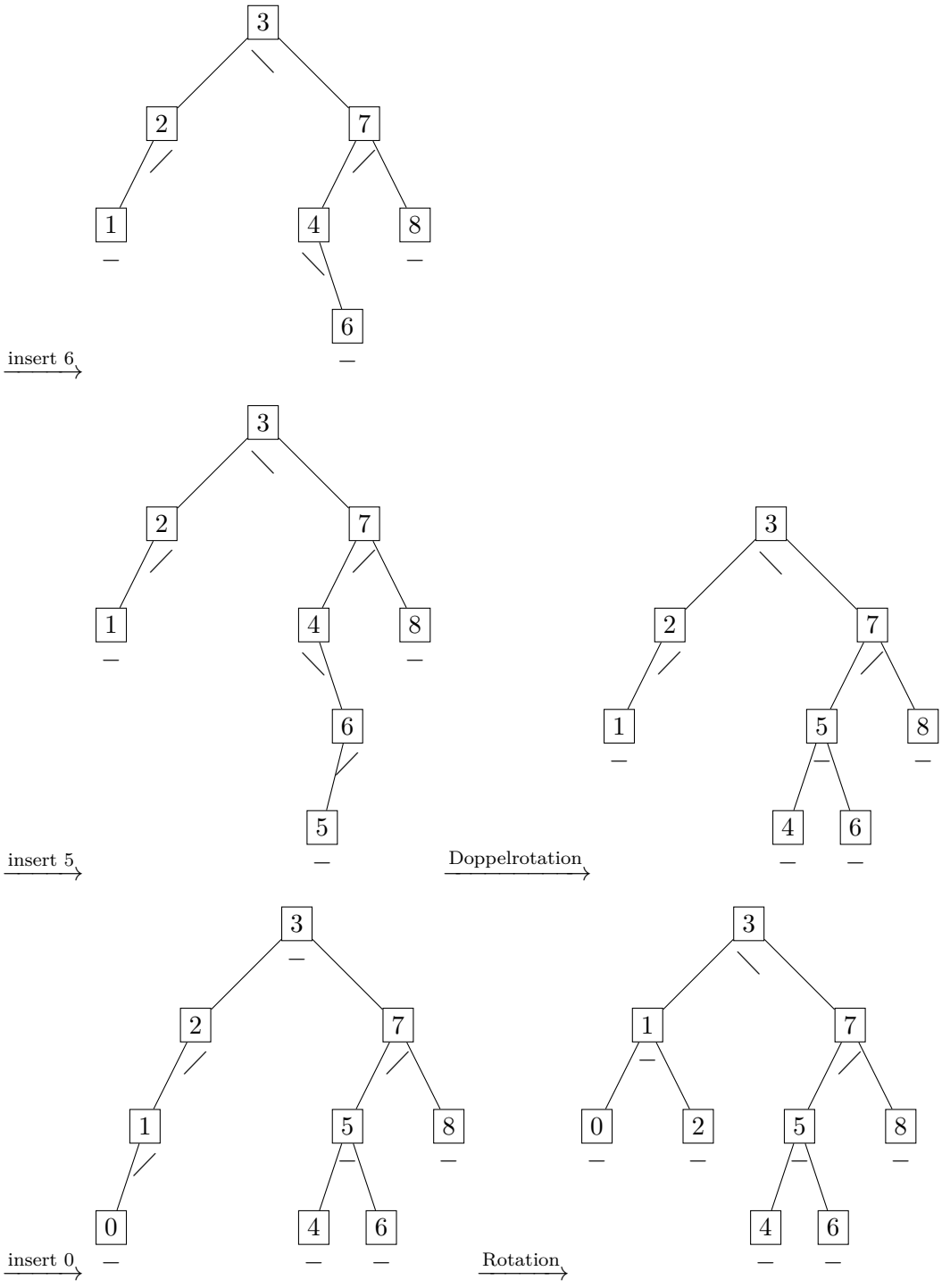
$\boxed{4}$

$\xrightarrow{\text{insert } 4}$ —

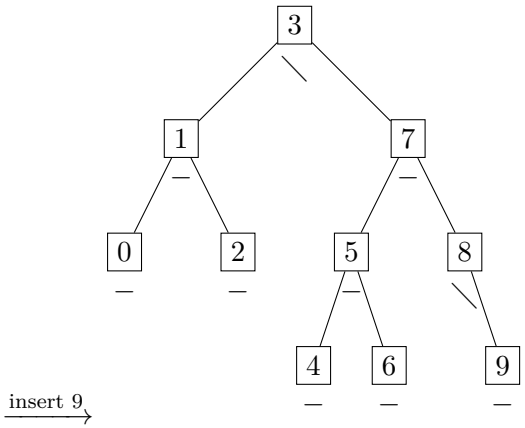
Alexander Neuwirth (439218)
Leonhard Segger (440145)
Jonathan Sigrist (441760)



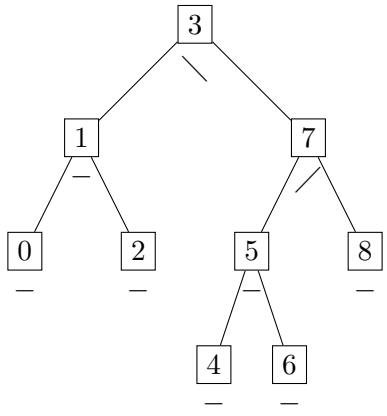
Alexander Neuwirth (439218)
Leonhard Segger (440145)
Jonathan Sigrist (441760)



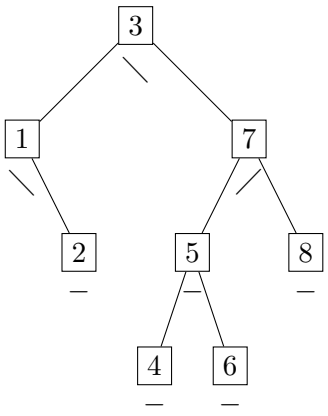
Alexander Neuwirth (439218)
Leonhard Segger (440145)
Jonathan Sigrist (441760)



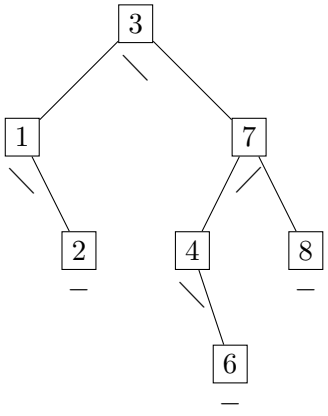

insert 9 →




remove 9 →

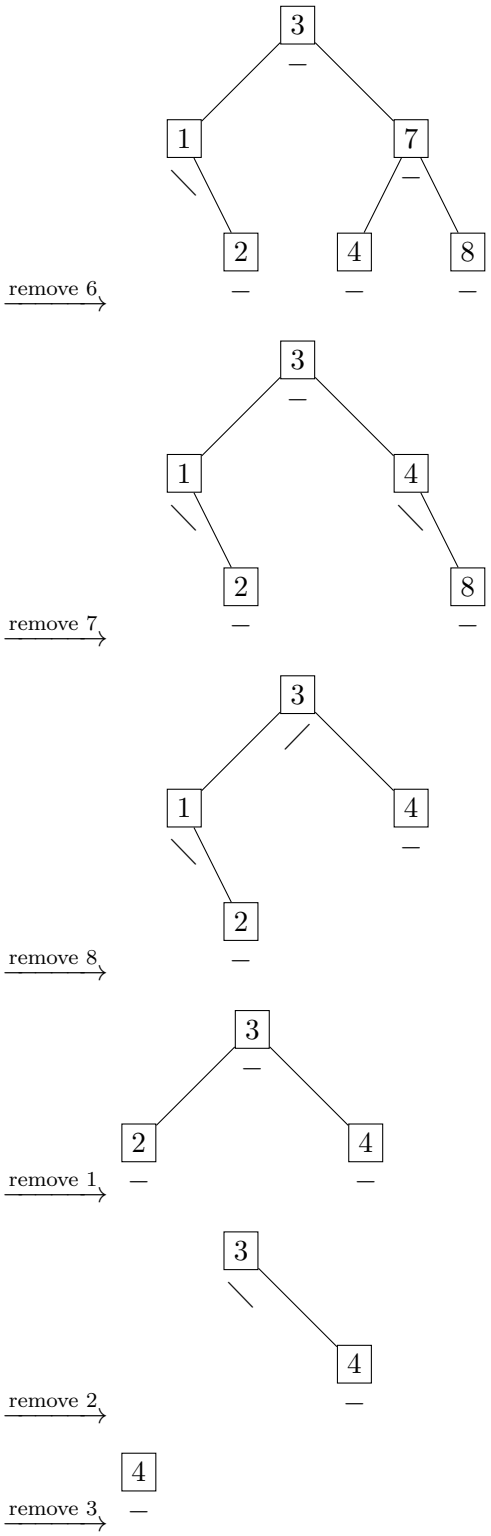



remove 0 →




remove 5 →

Alexander Neuwirth (439218)
Leonhard Segger (440145)
Jonathan Sigrist (441760)




**Aufgabe 39.**

Alexander Neuwirth (439218)
Leonhard Segger (440145)
Jonathan Sigrist (441760)

(a) (i)

| $S$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 1) | | | | | | 5 | | | |
| 2) | | 28 | | | | 5 | | | |
| 3) | | 28 → 19 | | | | 5 | | | |
| 4) | | 28 → 19 | | | | 5 | 15 | | |
| 5) | | 28 → 19 | 20 | | | 5 | 15 | | |
| 6) | | 28 → 19 | 20 | | | 5 | 15 → 33 | | |
| 7) | | 28 → 19 | 20 | 12 | | 5 | 15 → 33 | | |
| 8) | | 28 → 19 | 20 | 12 | | 5 | 15 → 33 | | 17 |

(ii)

| $S$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 1) | | | | | | 5 | | | |
| 2) | | 28 | | | | 5 | | | |
| 3) | | 28 | 19 | | | 5 | | | |
| 4) | | 28 | 19 | | | 5 | 15 | | |
| 5) | | 28 | 19 | 20 | | 5 | 15 | | |
| 6) | | 28 | 19 | 20 | | 5 | 15 | 33 | |
| 7) | | 28 | 19 | 20 | 12 | 5 | 15 | 33 | |
| 8) | | 28 | 19 | 20 | 12 | 5 | 15 | 33 | 17 |

(iii)

| $S$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 1) | | | | | | 5 | | | |
| 2) | | 28 | | | | 5 | | | |
| 3) | | 28 | 19 | | | 5 | | | |
| 4) | | 28 | 19 | | | 5 | 15 | | |
| 5) | | 28 | 19 | 20 | | 5 | 15 | | |
| 6) | | 28 | 19 | 20 | | 5 | 15 | 33 | |
| 7) | | 28 | 19 | 20 | 12 | 5 | 15 | 33 | |
| 8) | | 28 | 19 | 20 | 12 | 5 | 15 | 33 | 17 |

(iv)

| $S$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 1) | | | | | | 5 | | | |
| 2) | | 28 | | | | 5 | | | |
| 3) | | 28 | | 19 | | 5 | | | |
| 4) | | 28 | | 19 | | 5 | 15 | | |
| 5) | | 28 | 20 | 19 | | 5 | 15 | | |
| 6) | | 28 | 20 | 19 | 33 | 5 | 15 | | |
| 7) | | 28 | 20 | 19 | 33 | 5 | 15 | 12 | |
| 8) | | 28 | 20 | 19 | 33 | 5 | 15 | 12 | 17 |

(b) Zur Speicherung von 6 Elementen werden 3 Bits benötigt. Also $g(x) \to [0, 7]$.

$$
\begin{aligned}
23 &= \texttt{0010111} \\
35 &= \texttt{0100011} \\
56 &= \texttt{0111000} \\
81 &= \texttt{1010001} \\
86 &= \texttt{1011011} \\
91 &= \texttt{1011011}
\end{aligned}
$$

Alexander Neuwirth (439218)
Leonhard Segger (440145)
Jonathan Sigrist (441760)

Damit ergibt sich die Funktion $g(x) =$ `(x»6) | (x & 6)`.

**Aufgabe 40.**

(b) (i) Die Elemente können beim Einfügen zu einer linearen Liste werden. Dann müsste jeder Knoten überprüft werden, also $2^32$ Zugriffe.

(ii) Da der schiefste AVL-Baum eine Höhe von $1.44 \cdot \log_2 n$ hat, müssen im Worst-Caseeben so viele Zugriffe erfolgen.

(iii) Ein B-Baum mit $b = 255$ hat ebenso $a = 128$. Somit folgt eine maximale Höhe von $\log_{128}\left(\frac{2^{32}+1}{2} + 1\right) \geq$ Höhe im Worst-Case. Das sind gerade mal 5 Zugriffe auf Knoten.