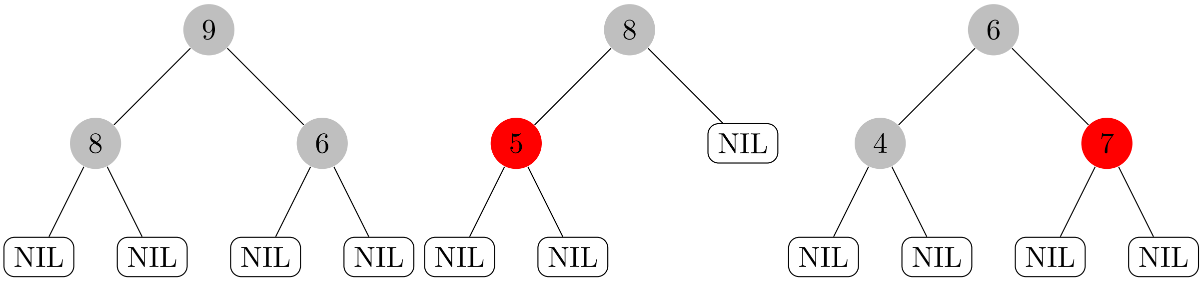
Albert-Ludwigs-Universität
Institut für Informatik
Prof. Dr. F. Kuhn
M. Fuchs, G. Schmid

Algorithmen und Datenstrukturen
Sommersemester 2024
Musterlösung Übungsblatt 7

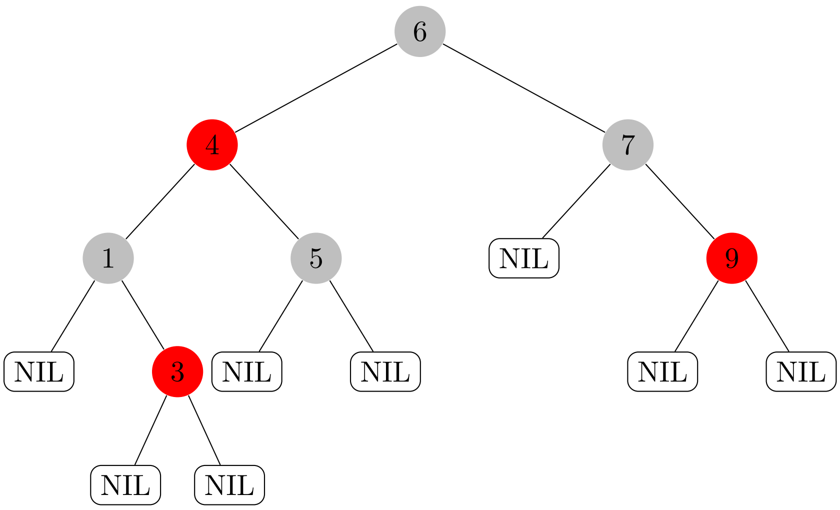
Abgabe: Dienstag, 11. Juni, 2024, 10:00 Uhr

Aufgabe 1: Rot-Schwarz Bäume *(10 Punkte)*

(a) Entscheiden Sie für jeden der folgenden Bäume, ob es sich um einen Rot-Schwarz Baum handelt und falls nicht, welche Eigenschaft verletzt ist: *(3 Punkte)*



(b) Führen Sie auf folgendem Rot-Schwarz Baum zuerst die Operation `insert`(8) und danach `delete`(5) aus. Zeichnen Sie den resultierenden Baum. Dokumentieren Sie Ihre Zwischenschritte.[1] *(7 Punkte)*



Musterlösung

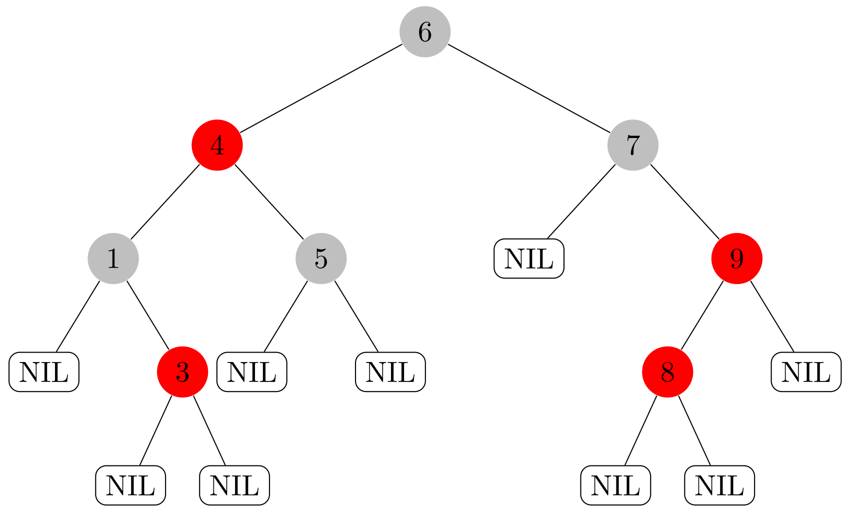
(a) Von links nach rechts:

1) Kein Rot-Schwarz-Baum, da kein binärer Suchbaum (die Wurzel hat ein rechtes Kind mit kleinerem Schlüssel).
2) Rot-Schwarz-Baum

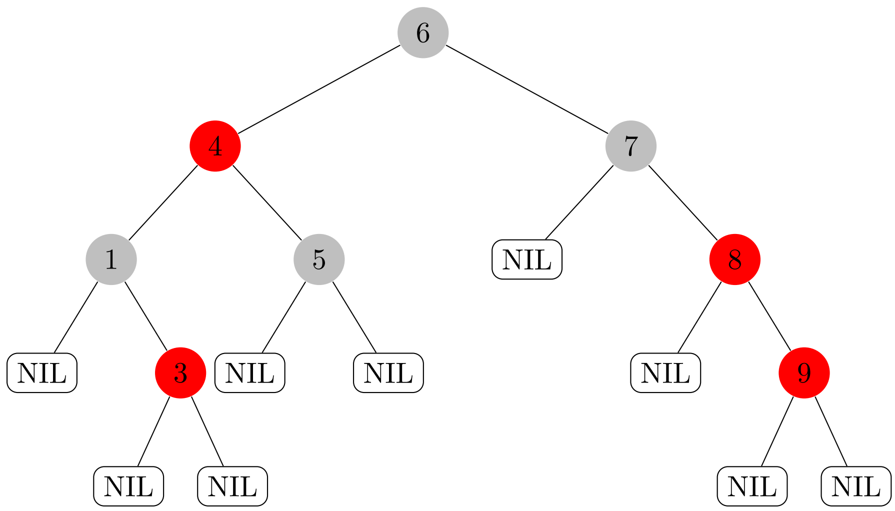
[1] Wie haben hier einen Beispielcode zum Zeichnen solcher Bäume in LaTeX bereitgestellt.

3) Kein Rot-Schwarz-Baum: Die Anzahl schwarzer Knoten auf einem Pfad von der Wurzel zu einem Blatt ist größer, wenn man durch den linken Teilbaum geht.

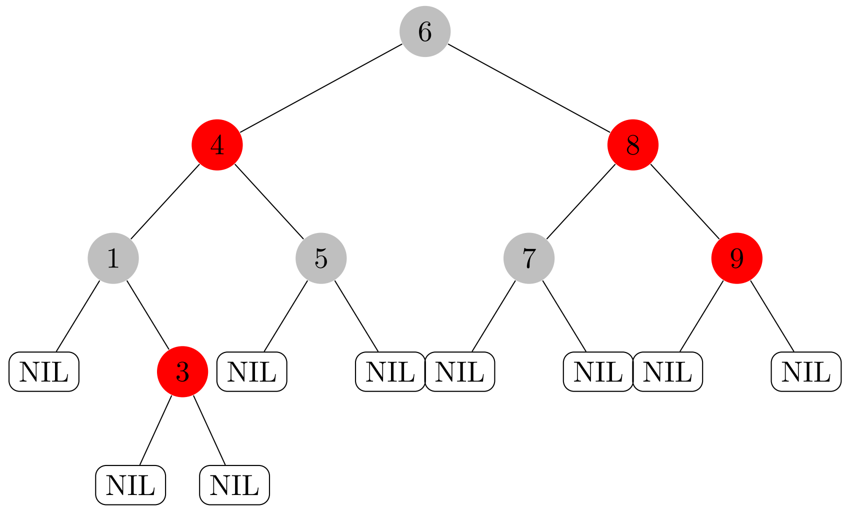
(b) Wir fügen zuerst einen roten Knoten mit Schlüssel 8 ein, so wie man allgemein Schlüssel in binäre Suchbäume einfügt.



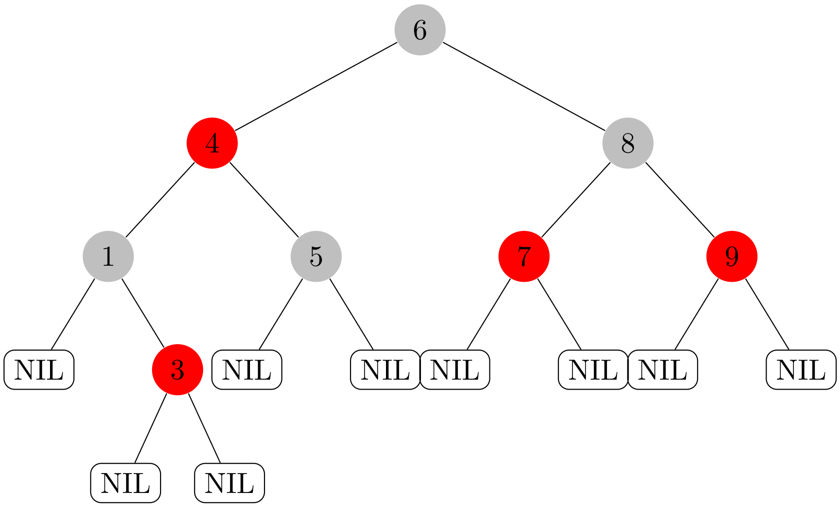
Wir befinden uns im Fall 1b aus der Vorlesung. Wir machen also ein right-rotate(9,8),



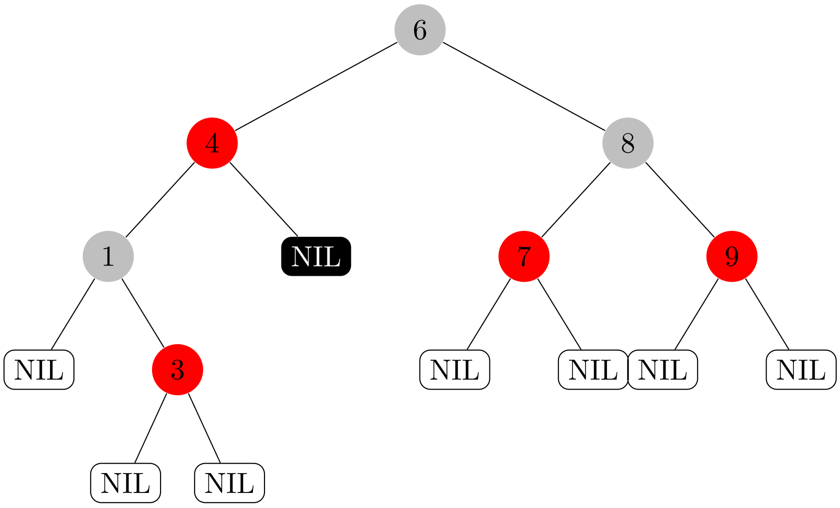
danach ein left-rotate(7,8)



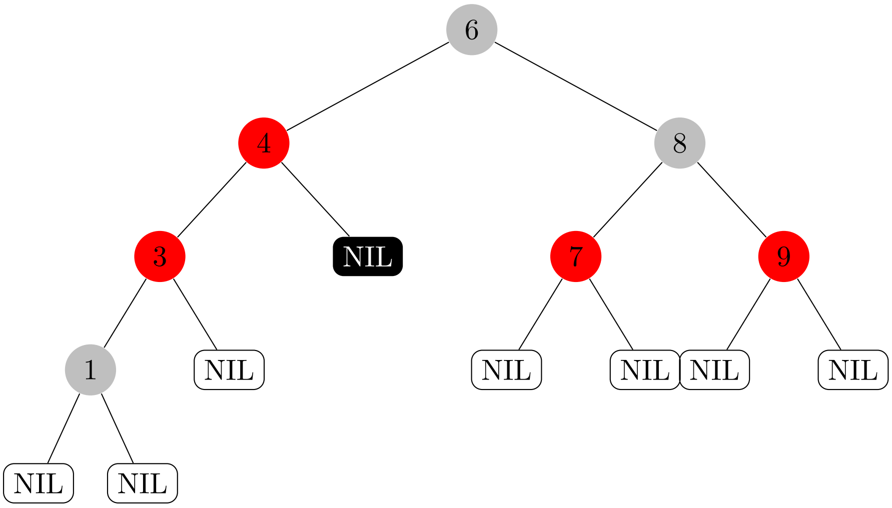
und färben schließlich Knoten 8 schwarz und Knoten 7 rot.



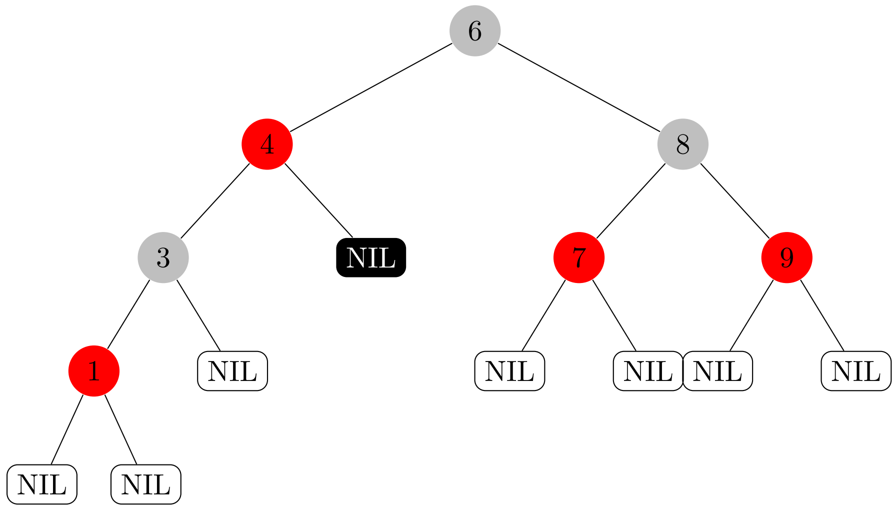
Nun führen wir `delete`(5) aus. Wir befinden uns im Fall 2b aus der Vorlesung (schwarzer Knoten mit zwei NIL-Kindern). Zunächst entfernen wir Knoten 5 aus dem Baum und färben –um die Schwarztiefe zu korrigieren– das rechte NIL-Kind von 4 doppelt schwarz.



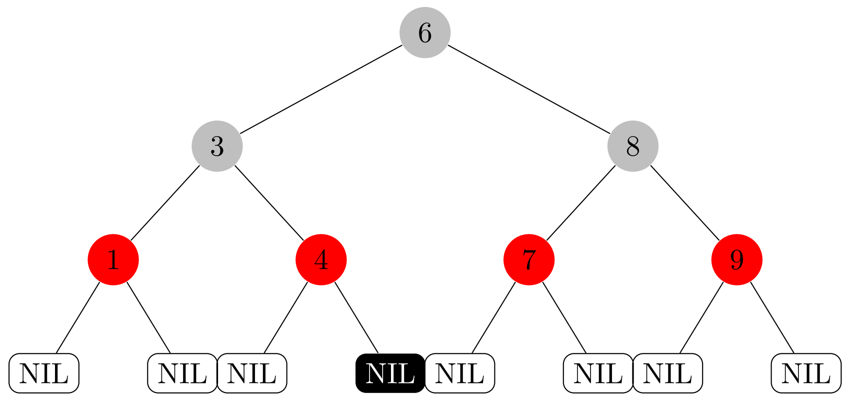
Wir befinden uns im Fall A.2 aus der Vorlesung. Wir machen ein left-rotate(1,3)



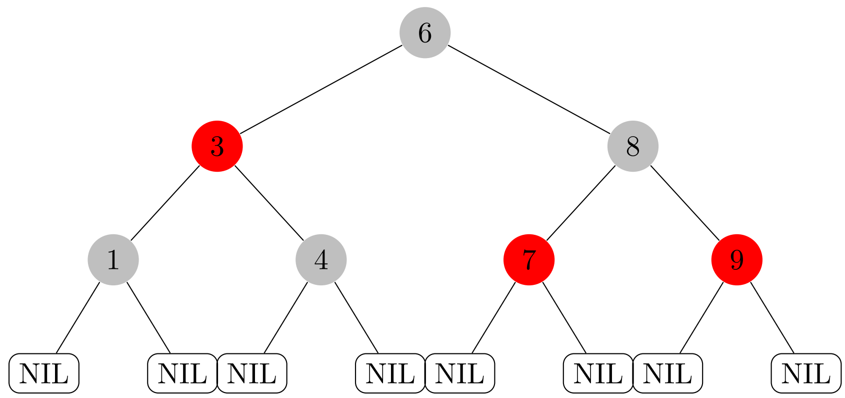
und färben Knoten 1 und 3 um.



Nun befinden wir uns im Fall A.1. Wir machen also ein right-rotate(4,3)



und färben um. Am Ende sieht der Baum so aus



Aufgabe 2: Balancierte Suchbäume *(10 Punkte)*

Sei ein **black-box-tree** ein binärer Suchbaum für den aber ausserdem folgende Zusatzeigenschaft gilt: Für jeden Knoten v unterscheidet sich die Tiefe[2] des rechten und linken Teilbaums um höchstens 1. *Hinweis: Da die Tiefe nicht für leere Teilbäume definiert ist, nutzen wir wie bei Rot-Schwarz Bäumen den Trick einen NIL Knoten an jedes "Blatt" zu hängen. Obige Zusatzeigenschaft muss also nur für die nicht-NIL Knoten gelten.*

(a) Beweisen oder Widerlegen Sie: In einem black-box-tree können Pfade von der Wurzel zu Blättern existieren deren Länge sich um mehr als 1 unterscheidet. *(1 Punkt)*

[2] Die Tiefe ist die Länge des längsten Pfades der Wurzel zu einem Blatt.

(b) Beweisen Sie *induktiv*, dass ein black-box-tree der Tiefe $d \geq 0$ bis zur Tiefe $\lfloor \frac{d}{2} \rfloor$ voll besetzt ist. Wir nehmen hier an dass ein Baum mit Tiefe $d = 0$ aus nur einem NIL Knoten besteht. *(3 Punkte)*

Bemerkung: Ein Baum ist zur Tiefe d' voll besetzt wenn er für alle $x \leq d'$ genau 2^x Knoten mit Tiefe x hat (d.h., Schicht x des Baumes hat die maximale Anzahl Knoten).

(c) Geben Sie die minimale Anzahl von Knoten eines black-box-tree in Abhängigkeit von d als Rekursionsgleichung an und begründen Sie. *(3 Punkte)*

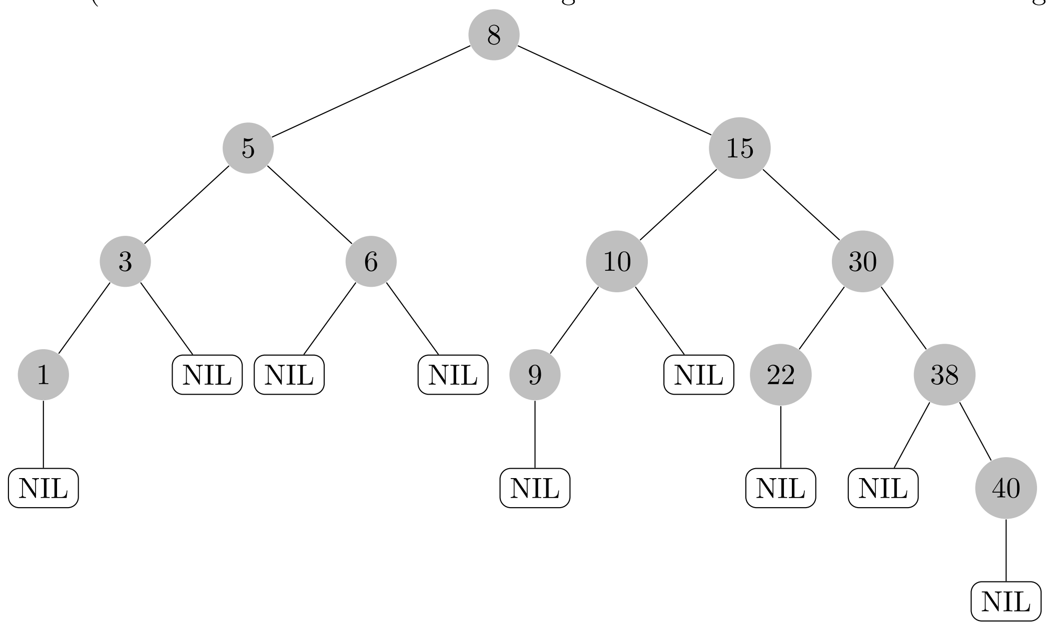
Hinweis: Drücken Sie die minimale Anzahl der Knoten eines black-box-tree der Tiefe d rekursiv mittels der min. Anzahl Knoten niedrigerer black-box-tree aus (mit Basisfällen für Tiefe 0 bzw. 1).

(d) Zeigen Sie dass ein black-box-tree mit n Knoten Tiefe $\mathcal{O}(\log n)$ hat. *(3 Punkte)*

Hinweis: Benutzen Sie dazu entweder Aussage b) oder Aussage c)

Musterlösung

(a) Die Aussage ist wahr, im folgenden ist der Pfad zwischen 8 und 6 um 2 kürzer als der längste Pfad. (Statt 2 NIL Kinder wurde aus Platzgründen immer nur eines der beiden gezeichnet.)



(b) **Induktionsanfang:** Jeder (nicht triviale) Baum hat eine Wurzel, ist also bis zur Tiefe 0 voll besetzt. Die Behauptung gilt also für $d = 0$ und $d = 1$.

Induktionsschluss: Angenommen die Behauptung stimmt für alle black-box-tree mit Tiefe höchstens d. Wir zeigen dass dies auch für einen black-box-tree der Tiefe $d+1$ gilt.

Sei r die Wurzel eines solchen black-box-tree T und T_ℓ und T_r der linke bzw. rechte Teilbaum an r. Einer der beiden Teilbäume T_ℓ, T_r muss nun Tiefe d haben, da T Tiefe $d+1$ hat. Der andere muss dann Tiefe mindestens $d-1$ haben, aufgrund der black-box-tree-Eigenschaft. Der niedrigere von beiden Teilbäumen ist aufgrund der Induktionsvoraussetzung bis Tiefe $\lfloor \frac{d-1}{2} \rfloor = \lfloor \frac{d+1}{2} - 1 \rfloor = \lfloor \frac{d+1}{2} \rfloor - 1$ voll besetzt. Dies gilt natürlich auch für den anderen (tieferen) Teilbaum. Damit haben wir in T eine voll besetzte Schicht mehr (die Wurzel kommt hinzu) weshalb T bis zur Tiefe $\lfloor \frac{d+1}{2} \rfloor - 1 + 1 = \lfloor \frac{d+1}{2} \rfloor$ voll besetzt ist.

(c) Sei N_d die Mindestanzahl von Knoten eines black-box-tree der Tiefe d. Ein black-box-tree der Tiefe 0 hat genau einen Knoten (die Wurzel), wir setzen daher $N_0 = 1$. Ein Baum der Tiefe 1 hat prinzipiell 2 oder 3 Knoten, da wir aber davon ausgehen dass wir immer NIL Kinder haben, hat ein Baum mit Tiefe 1 sogar genau 3 Knoten. Somit ist $N_1 = 3$.

Sei nun $d \geq 2$. Wir definieren N_d rekursiv. Ein black-box-tree T dieser Tiefe besteht aus einer Wurzel r und einem rechten und linken Teilbaum T_ℓ respektive T_r. Einer der beiden Teilbäume (o.B.d.A. sei das T_r) muss offensichtlich Tiefe $d-1$ haben (da T Tiefe d hat) besteht also aus mindestens N_{d-1} Knoten. Dann muss T_ℓ aufgrund der black-box-tree-Eigenschaft Tiefe mindestens $d-2$ haben besteht also N_{d-2} Knoten. Damit ist die Mindestanzahl der Knoten von T rekursiv gegeben durch

$$N_d = N_{d-1} + N_{d-2} + 1.$$

(d) **Mittels (b)**: Ein black-box-tree der Tiefe d ist nach (a) bis zur $\lfloor \frac{d}{2} \rfloor$-ten Schicht voll besetzt. D.h., dieser Baum hat mindestens $n \geq 2^{\lfloor d/2 \rfloor}$ Knoten (wir brauchen nur eine sehr grobe Abschätzung für die Asymptotik). Damit ist

$$\begin{aligned}
& 2^{\lfloor \frac{d}{2} \rfloor} \leq n \\
\iff \quad & \lfloor \frac{d}{2} \rfloor \leq \log(n) \\
\implies \quad & \frac{d}{2} - \frac{1}{2} \leq \lfloor \frac{d}{2} \rfloor \leq \log(n) \\
\implies \quad & d \leq 2 \log n + 1 \\
\implies \quad & d \in \mathcal{O}(\log(n)).
\end{aligned}$$

Mittels (c): Fast analog zur Fibonacci-Folge aus dem Hinweis haben wir $N_d = N_{d-1} + N_{d-2} + 1 = 2N_{d-2} + N_{d-3} + 2 \geq 2N_{d-2}$. Anschaulich heißt das, dass sich die Mindestanzahl der Knoten in einem black-box-tree mindestens alle zwei "Tiefenschritte" verdoppelt. Konkret heißt das $N_d \geq 2N_{d-2} \geq 2^2 N_{d-4} \geq \cdots \geq 2^{\lfloor d/2 \rfloor} N_{d-2\lfloor d/2 \rfloor} \geq 2^{\lfloor d/2 \rfloor} N_0 = 2^{\lfloor d/2 \rfloor}$. Der Rest geht wie oben.