Aufgabe 7 (Bäume)

Gegeben sei die folgende Realisierung von binären Bäumen (in einer an Java angelehnten Notation):

```
3  class Node {
4    Node left, right;
5    int value;
```

(a) Beschreiben Sie in möglichst wenigen Worten, was die folgende Methode `foo` auf einem nicht-leeren binären Baum berechnet.

```
11   int foo(Node node) {
12     int b = node.value;
13     if (b < 0) {
14       b = -1 * b;
15     }
16     if (node.left != null) {
17       b = b + foo(node.left);
18     }
19     if (node.right != null) {
20       b = b + foo(node.right);
21     }
22     return b;
23   }
```

> Die Methode `foo(Node node)` berechnet die Summe aller Knoten des Unterbaums des als Parameter übergebenen Knotens `node`. Der Schüsselwert des Knotens `node` selbst wird in die Summenberechnung mit einbezogen.

(b) Die Laufzeit der Methode `foo(tree)` ist linear in n, der Anzahl von Knoten im übergebenen Baum `tree`. Begründen Sie kurz, warum `foo(tree)` eine lineare Laufzeit hat.

> Die Methode `foo(Node node)` wird pro Knoten des Baums `tree` genau einmal aufgerufen. Es handelt sich um eine rekursive Methode, die auf den linken und rechten Kindknoten aufgerufen wird. Hat ein Knoten keine Kinder mehr, dann wir die Methode nicht aufgerufen.

(c) Betrachten Sie den folgenden Algorithmus für nicht-leere, binäre Bäume. Beschreiben Sie in möglichst wenigen Worten die Wirkung der Methode `magic(tree)`. Welche Rolle spielt dabei die Methode `max`?

```
25   void magic(Node node) {
26     Node m = max(node);
27     if (m.value > node.value) {
28       // Werte von m und node vertauschen
29       int tmp = m.value;
30       m.value = node.value;
31       node.value = tmp;
32     }
33     if (node.left != null)
34       magic(node.left);
35     if (node.right != null)
```

```
36          magic(node.right);
37      }
38
39      Node max(Node node) {
40        Node max = node;
41        if (node.left != null) {
42          Node tmp = max(node.left);
43          if (tmp.value > max.value)
44            max = tmp;
45        }
46        if (node.right != null) {
47          Node tmp = max(node.right);
48          if (tmp.value > max.value)
49            max = tmp;
50        }
51        return max;
52      }
```

Methode `magic(tree)` vertauscht für jeden Knoten des Unterbaums den Wert mit dem Maximal-Knoten. Die Methode `max` liefert dabei den Knoten mit der größten Schlüsselwert im Unterbaum des übergebenen Knoten (sich selbst eingeschlossen).

(d) Geben Sie in Abhängigkeit von n, der Anzahl von Knoten im übergebenen Baum `tree`, jeweils eine Rekursionsgleichung für die asymptotische Best-Case-Laufzeit $(B(n))$ und Worst-Case-Laufzeit $(W(n))$ des Aufrufs `magic(tree)` sowie die entsprechende Komplexitätsklasse (Θ) an. Begründen Sie Ihre Antwort.

Hinweis: Überlegen Sie, ob die Struktur des übergebenen Baumes Einfluss auf die Laufzeit hat. Die lineare Laufzeit von `max(t)` in der Anzahl der Knoten des Baumes `t` darf vorausgesetzt werden.

Additum: Kompletter Code

```
1  package org.bschlangaul.examen.examen_66115_2019_09;
2
3  class Node {
4    Node left, right;
5    int value;
6
7    public Node(int value) {
8      this.value = value;
9    }
10
11   int foo(Node node) {
12     int b = node.value;
13     if (b < 0) {
14       b = -1 * b;
15     }
16     if (node.left != null) {
17       b = b + foo(node.left);
18     }
19     if (node.right != null) {
20       b = b + foo(node.right);
21     }
```

```
22      return b;
23    }
24
25    void magic(Node node) {
26      Node m = max(node);
27      if (m.value > node.value) {
28        // Werte von m und node vertauschen
29        int tmp = m.value;
30        m.value = node.value;
31        node.value = tmp;
32      }
33      if (node.left != null)
34        magic(node.left);
35      if (node.right != null)
36        magic(node.right);
37    }
38
39    Node max(Node node) {
40      Node max = node;
41      if (node.left != null) {
42        Node tmp = max(node.left);
43        if (tmp.value > max.value)
44          max = tmp;
45      }
46      if (node.right != null) {
47        Node tmp = max(node.right);
48        if (tmp.value > max.value)
49          max = tmp;
50      }
51      return max;
52    }
53
54    public static void main(String[] args) {
55      Node eins = new Node(1);
56      Node zwei = new Node(2);
57      Node drei = new Node(3);
58      Node vier = new Node(4);
59      Node fünf = new Node(5);
60
61      eins.left = zwei;
62      eins.right = drei;
63
64      drei.left = vier;
65      drei.right = fünf;
66
67
        ↪  System.out.println("Die Methode foo() berechnet die Summe aller Schlüsselwerte: 1 + 2 + 3 + 4 + 5 = "
        ↪  + eins.foo(eins));
68
69      System.out.println("Die Methode magic() wird ausgeführt");
70      eins.magic(eins);
71
72      System.out.println("Der Knoten eins hat jetzt den Wert: " + eins.value);
73      System.out.println("Der Knoten zwei hat jetzt den Wert: " + zwei.value);
74      System.out.println("Der Knoten drei hat jetzt den Wert: " + drei.value);
75      System.out.println("Der Knoten vier hat jetzt den Wert: " + vier.value);
76      System.out.println("Der Knoten fünf hat jetzt den Wert: " + fünf.value);
77
78      System.out.println("Die Summe aller Schlüsselwerte bleibt gleich: " +
        ↪  eins.foo(eins));
79    }
80  }
```