Lehrstuhl für Informatik I
Algorithmen, Komplexität und wissensbasierte Systeme
Universität Würzburg

Würzburg, den 17. Januar 2019

Prof. Dr. Alexander Wolff
Andre Löffler, M. Sc.
Johannes Zink, M. Sc.

3. Präsenzübungsblatt zur Vorlesung Algorithmen und Datenstrukturen (Winter 2018/19)

Aufgabe 1 – Operationen in Rot-Schwarz-Bäumen

a) In einen anfangs leeren Rot-Schwarz-Baum werden nacheinander die Zahlen $50, 42, 38, 13, 17$ und 5 eingefügt. Zeichnen Sie den Baum nach jeder dieser Einfügeoperationen.

b) Anschließend werden die Zahlen $5, 13, 17$ und 38 nacheinander gelöscht. Zeichnen Sie den Baum nach jeder dieser Löschoperationen.

Stellen Sie sicher, dass rote und schwarze Knoten klar voneinander unterscheidbar sind. Sie brauchen die nil-Blätter nicht zu zeichnen.

Aufgabe 2 – Springer auf Schachfeld

Gesucht ist ein Algorithmus, der ermittelt, wieviele Züge ein Springer benötigt, um auf einem Schachbrett mit $n \times n$ Feldern von Feld (x_1, y_1) zu Feld (x_2, y_2) zu kommen.

a) Wie lässt sich dieses Problem als Kürzeste-Wege-Problem auf einem Graphen modellieren? Was repräsentieren die Knoten und Kanten des Graphen?

b) Ist der Graph zweifärbbar (entsprechend der Definition in Blatt 7 Aufgabe 2)? Begründen Sie Ihre Antwort.

c) Geben Sie für das gewöhnliche Schachbrett mit 8×8 Feldern die genaue Anzahl der Kanten des Graphen an.

d) Geben Sie für ein Schachbrett mit $n \times n$ Feldern eine obere Schranke für die Anzahl der Kanten an. Verwenden Sie dafür die Groß-O-Notation.

e) Geben Sie einen Algorithmus an, der die erforderliche Anzahl an Zügen in $\Theta(n^2)$ Zeit berechnet. Begründen Sie die Laufzeit Ihres Algorithmus.

Aufgabe 3 – Amortisierte Analyse

In der Vorlesung haben Sie die Implementierung der Funktion Successor für binäre Suchbäume kennengelernt, die auch unten angegeben ist. Die Worst-Case-Laufzeit dieser Funktion ist $\Theta(h)$, wobei h die Höhe des Baumes ist.

```
Node Successor(Node x)
if x.right ≠ nil then
    return Minimum(x.right)
y = x.p
while y ≠ nil and x == y.right do
    x = y
    y = y.p
return y
```

Auf einem binären Suchbaum wird die Funktion Successor nun für jeden der n Knoten genau einmal aufgerufen. Begründen Sie, dass in diesem Fall die amortisierte Laufzeit eines einzelnen solchen Aufrufes von Successor in $O(1)$ ist.

Aufgabe 4 – Bäume transformieren

Als ADS-HörerIn stellen Sie sich zu Hause natürlich nicht irgendeine Zimmerpflanze auf, sondern einen binären Suchbaum. Ihr Suchbaum heißt B_1 und hat n Knoten. Ihr(e) Übungspartner(in) hat auch einen binären Suchbaum, der B_2 genannt wird und zufälligerweise genau dieselben Schlüssel wie Ihr Suchbaum enthält. Er sieht aber schöner aus. Deshalb möchten Sie Ihren Suchbaum so transformieren, dass er am Ende wie B_2 aussieht.

Zeigen Sie hierzu, dass Sie stets in der Lage sind B_1 durch $O(n)$ Rotationen in B_2 zu transformieren, egal wie die beiden binären Suchbäume konkret aussehen.

Hinweis: Zeigen Sie zunächst, dass $n - 1$ Rotationen ausreichen, um einen beliebigen Suchbaum in einen Suchbaum zu transformieren, in dem kein Knoten einen linken Kindknoten hat.

Diese Aufgaben werden eventuell gemeinsam in den Übungen am 22. und 23. Januar 2019 gelöst. Sie brauchen Sie nicht vorher zu lösen und auch nicht abzugeben.