Algorithmen und Datenstrukturen
Protokoll: Treecheck

Karlheinz Lunatschek und Alexander Weidinger

23. April 2024

1 Rekursive Funktionen

1.1 Insertion

Es wird geprüft, ob man für diesen Wert einen Platz findet und wenn es keinen gibt, dann versucht das Programm einen Platz zu finden, indem er die Funktion selbst wieder aufruft. Wenn der Wert größer ist als die der Node, dann wird auf der rechten Seite eine Node erstellt und der Wert eingefügt, ansonsten umgekehrt.

1.2 Print Tree

Es wird geprüft, ob ein Baum erstellt wurde. Wenn ja, dann wird die Funktion zweimal rekursiv aufgerufen, und zwar einmal mit der linken und dann mit der rechten Node. Das geht so lange weiter, bis es keine weiterfolgenden Nodes mehr gibt.

1.3 Suchen mit einem Wert

Hier wird überprüft ob ein Wert im Baum existiert. Dies tut er indem er den Wert von der Node checkt und mit dem zusuchenden Wert vergleicht. Wenn der Wert kleiner ist, geht er in den linken Teilbaum oder in den rechten Teilbaum falls der Wert größer ist. Während dem Durchlaufen von dem Baum, speichert die Suchfunktion Werte in einem Array, damit bei der Ausgabe ersichtlich ist, durch welche Nodes er gegangen ist. Am Ende zeigt das Programm an, ob es den Wert gibt oder nicht.

1.4 Suchen mit mehreren Werten

In dieser Funktion versuchen wir herauszufinden, ob ein Subtree in einen anderen Tree enthalten ist. Dies tun wir indem die Funktion rekursiv aufgerufen wird. Im Endeffekt versucht die Funktion herauszufinden, ob ein Wert irgendwo in dem Originaltree enthalten ist. Es gibt zwei Möglichkeiten wie sich die Funktion rekursiv aufrufen kann. Wenn die Node-values von beiden Trees gleich sind, dann wird in beiden Trees rekursiv nach einem weiteren Wert gesucht. Sind jedoch die Werte ungleich dann wird nur im originalen Tree rekursiv weitergesucht, bis die Werte der beiden Trees wieder stimmen. Falls der Searchtree zusätzliche Werte hat, dann ist es kein Subtree und der Boolwert wird als falseäusgegeben. Falls es doch ein Subtree ist, dann wird in der Konsole SSubtree foundäusgegeben.

Aufwandsabschätzung	Best Case	Avg. Case	Worst Case
Insert	O(logN)	O(N-logN)	O(N)
Printtree	O(N)	O(N)	O(N)
SimpleSearch	O(1)	O(logN-1)	O(logN)
SubtreeSearch	O(logN)	O(N-logN)	O(N)

Tabelle 1: Aufwandsabschätzung von mehreren Funktionen im Code