Protokoll Programmierbeispiel 2

Armin NASERI, Andor KURUCZ

Einleitung:

Das vorliegende Protokoll konzentriert sich auf die detaillierte Untersuchung der BinaryTree-Implementierung, die Funktionalitäten zur Verwaltung und Analyse von binären Suchbäumen bietet. Ein binärer Suchbaum ist eine Datenstruktur, die häufig in der Informatik verwendet wird, um effizienten Zugriff, Einfügungen und Suchvorgänge zu ermöglichen. Diese Implementierung umfasst verschiedene rekursive Funktionen, die für Operationen wie das Einfügen neuer Knoten, die Überprüfung auf AVL-Eigenschaften und die Analyse statistischer Informationen verwendet werden.

Beschreibung der rekursiven Funktionen:

Die rekursiven Funktionen in der main.cpp und BinaryTree.cpp spielen eine entscheidende Rolle bei der Verarbeitung und Manipulation von Baumstrukturen. In der main.cpp-Datei werden isIdentical und searchSubtree verwendet, um die Ähnlichkeit zwischen Bäumen zu überprüfen und Teilbäume in einem größeren Baum zu suchen. Diese Funktionen nutzen die rekursive Natur von Baumstrukturen, um effizient alle Knoten zu durchlaufen und die gewünschten Operationen durchzuführen.

In BinaryTree.cpp sind die Funktionen wie checkAVL, height und printStatistics intern rekursiv. Diese Funktionen sind für die Verwaltung und Analyse von Binärbäumen verantwortlich. Zum Beispiel verwendet checkAVL eine rekursive Methode, um den Balancierungsfaktor jedes Knotens im Baum zu überprüfen und AVL-Verletzungen zu identifizieren.

O-Notation und Laufzeitvergleich:

Die Effizienz der Operationen auf Binärbäumen wird durch ihre Laufzeiten und ihre O-Notation bestimmt. Die O-Notation beschreibt das asymptotische Verhalten der Algorithmen in Bezug auf die Eingabegröße.

Die meisten Operationen auf Binärbäumen haben eine logarithmische oder lineare Laufzeitkomplexität. Zum Beispiel hat das Einfügen eines Elements in einen ausbalancierten Binärbaum eine O(log n) Laufzeit, während das Durchlaufen des Baumes für statistische Analysen eine lineare O(n) Laufzeit hat.

Es ist wichtig zu beachten, dass die Effizienz der Operationen auch von der Struktur des Baumes abhängt. Ein ausbalancierter Baum bietet optimale Laufzeiten für alle Operationen, während ein ungleichmäßiger Baum die schlechtesten Fallzeiten verursachen kann.

Optimale Anwendung:

WICHTIG

Es ist wichtig anzumerken, dass damit das Programm auch funktioniert, ist es wichtig eine Linux/Unix Umgebung zu verwenden. In unserem Beispiel haben wir Mingw64 unter Windows verwendet, welche eine Linux Umgebung simuliert. Die .exe des Programmes soll in den binary-Ordner reinkopiert werden, damit das Programm von überall aus innerhalb des Terminals angewendet werden kann.