Protokoll AlgoDat Beispiel 2

calcBalance:

Die Funktion traversiert den Baum in Post-Order Reihenfolge beginnend mit dem rechten Teilbaum.

Zu jeder Node wird dann die Höhe des rechten und des linken Teilbaumes berechnet und dann die Differenz als Klasseneigenschaft „balance“ gespeichert.

Der Aufwand für die Berechnung setzt sich also aus dem Untersuchen der jeweiligen Nodes(O(n)) sowie der Berechnung der Höhe Teilbäume (O(log(n)) zusammen. Daraus ergibt sich ein Gesamtaufwand von O(n\*log(n))

SubtreeSearch

Bevor subtreeSearch() ausgeführt wird, wird zunächst die findRoot() funktion ausgeführt. Diese findet heraus wo sich die root des subtrees im main tree befinden (nullptr wenn diese node nicht existiert), sollte sich nur eine einzelne Node angegeben sein und diese wird gefunden gibt das Programm ein sofortiges „found“ aus. Wenn es sich um 2 oder mehr Nodes handelt, wird nun von dem zuvor gefunden Root aus rekursive der Tree traversiert und überprüft ob alle Elemente, die vom User übermittelt wurden, von besagten Root-Node hinweg erreicht werden können. Dies wird gelöst, indem die vom User eingegebenen Zahlen in einem vector gespeichert werden und dann mitgezählt wie viele Elemente gefunden wurden. Sobald diese Zahl erreicht wird, wird die function abgebrochen da alle Elemente gefunden wurden und somit der Subtree gefunden wurde.

Im best case O(log(n)) worst case O(n).

Der best case ist ein im linken Branch befindender Root mit einer geringen Anzahl an weitern Nodes, da dann weniger als die Hälfte des Baumes durchsucht werden muss. Der Worst case ist, wenn der Root das letzte Element im rechten Branch ist, da dann der ganze Baum durchsucht werden muss.