

# Algodat Übung 3

treecheck

Thomas Rauhofer

*if15b029*

Tobias Watzek

*if15b038*

# 1. Inhalt

[Inhalt](#)

[Rekursive Funktionen](#)

[Insert](#)

[AVL\\_height](#)

[AVL\\_check](#)

[check\\_max und check\\_min](#)

[check\\_avg](#)

[Aufwandsabschätzung](#)

[Insert](#)

[AVL\\_check](#)

[check\\_max und check\\_min](#)

[check\\_avg](#)

## 2. Rekursive Funktionen

### 3. Insert

Diese Funktion nimmt ein Argument und setzt es an die richtige Stelle in unserem Baum.

Unsere insert Funktion ist überladen und kann public mit nur einem Argument aufgerufen werden, wonach sie dann die private version aufruft und root als zweites Argument einsetzt.

Wenn root NULL ist (es wird im constructor mit NULL initialisiert) wird der übergebene int Wert als Wert in der root eingesetzt.

Wenn root ungleich NULL ist wird überprüft ob der übergeben int Wert kleiner ist als der Wert des aktuellen Knoten oder größer. Bei kleiner wird überprüft ob der linke Ast des Knotens NULL ist und wenn dann wird insert für den linken Anst aufgerufen, bei größer passiert das selbe mit dem rechten knoten. Wenn der Wert des überprüften Astes nicht NULL ist, wird so lange rekursiv durch den Baum gegangen bis ein Knoten mit NULL gefunden wird und dort wird der Wert eingefügt.

Im laufe des inserts werden auch der maximale und der minimale eingegebene Wert festgestellt und gespeichert.

### 4. AVL\_height

Diese Funktion kann für jeden Knoten im Baum aufgerufen werden und überprüft rekursiv wie viele Ebenen sich unter ihm befinden und returned diese Wert.

Vom übergebenen Knoten aus überprüft die Funktion ob beide Äste NULL sind. Wenn beide NULL sind wird 1 returned und wenn einer der beiden (oder beide) nicht NULL sind wird die Funktion für beide noch einmal aufgerufen.

## 5. AVL\_check

Diese Funktion überprüft ob der Baum ein AVL Baum ist, gibt für jeden Knoten den AVL-Wert an und falls für einen der Knoten die AVL Bedingung nicht erfüllt wird, wird ausgegeben das der Baum kein AVL Baum ist.

Auch diese Funktion ist überladen und kann public aufgerufen werden, wonach sie dann die private version aufruft und root als Argument einsetzt.

Vom ersten Knoten aus wird überprüft ob der rechte Ast NULL ist und wenn nicht wird AVL\_check rekursiv auf ihn aufgerufen und der avl\_wert per AVL\_height berechnet. Das gleiche gilt für den linken Knoten.

## 6. check\_max und check\_min

Diese Funktionen ermitteln respektive den größten und den kleinsten im Baum befindlichen Wert und schreiben diesen in die Klassenvariablen.

Beide Funktionen werden wieder per public Funktion auf root aufgerufen und überprüfen dort ob der linke beziehungsweise rechte Ast NULL ist und wenn nicht wird die funktion auf den erneut aufgerufen. Wenn der linke beziehungsweise rechte Ast NULL ist wird der Wert in min\_value bzw max\_value gespeichert.

Es wird sozusagen für das minimum der linkeste Wert und für das maximum der rechteste Wert im Baum ermittelt.

## 7. check\_avg

Diese Funktion summiert alle Werte im Baum und zählt die Anzahl der Werte.

Die Funktionen wird wie immer per public Funktion auf root aufgerufen, addiert den key\_value zu der Summe der Werte, erhöht den counter für die Werte um eins und wenn einer der beiden Äste ungleich NULL ist wird die Funktion auf diesen(oder beide) Ast wieder aufgerufen.

## 8. Aufwandsabschätzung

### 9. Insert

Beim inserten von Werten wird ein Aufwand von  $O(1)$  im best case und  $O(n)$  im worst case erwartet, wobei  $N$  die Anzahl der einzufügenden Werte ist.

Wenn der Baum ein AVL Baum ist, dann ist ein Aufwand von  $O(\log N)$  zu erwarten.

### 10. AVL\_check

Wie beim insert ist für den AVL\_check ein best case von  $O(1)$  und ein worst case von  $O(N)$  zu erwarten wobei hier  $N$  für die Anzahl der Knoten des Baumes steht.

### 11. check\_max und check\_min

Hier ein geringer Aufwand erwartet da nur bis zum linkesten bzw. rechtesten element durchgegangen werden muss.

### 12. check\_avg

Hier wird der Aufwand  $O(N)$  sein wobei  $N$  die Anzahl der eingefügten Werte ist, da jedes Element angesprochen werden muss.