Inhaltsverzeichnis

[Anpassungen an bestehenden Algorithmen 2](#_Toc483922927)

[Verfahren zum Ermitteln der Summe zwischen zwei Werten 2](#_Toc483922928)

[Komplexitätsuntersuchung 3](#_Toc483922929)

[Quellen 4](#_Toc483922930)

[Glossar 4](#_Toc483922931)

**Binärer Suchbaum Teil 1+2**

Aufgabenblatt 7+8

**Abstract**

*Im Folgenden wird ein Algorithmus beschrieben, welcher in einem Binärbaum die Summe Zwischen zwei Werten ermitteln kann. Der Algorithmus mit den umgesetzten Binärbäumen verwendet werden. Die verwendeten Knoten erhalten eine Zusatzinformation: die Summe aller Söhne. Durch diese Information lässt sich die Summe zwischen zwei Werten wie folg berechnen: Die Summe aller Knoten größer als das Maximum, die Summe aller Knoten kleiner als das Minimum subtrahiert von der Summe aller Knoten. Es müssen nur die beiden Knoten der Werte gefunden werden um das Ergebnis berechnen zu können.*

## 

## Anpassungen an bestehenden Algorithmen

Zum erfolgreichen Methodenaufruf des Algorithmus zum Finden der Summe zwischen zwei Werten im Binärbaum muss der bestehende Algorithmus des Binärbaums und die Klasse des Knotens angepasst werden.

Jeder Knoten hat jetzt eine weitere Variable „sum“, welche beim Einfügen eines Knotens in einen Binärbaum, den Wert von allen hinzugefügten Söhnen akkumuliert. Dies wird in der Methode zum Einfügen eines Knotens umgesetzt, damit jeder Knoten über die Summe all seiner Söhne verfügt.

Warum die Summe aller Söhne in seinem Vater gespeichert wird, wird im nächsten Kapitel beschrieben.

## Verfahren zum Ermitteln der Summe zwischen zwei Werten

*Um die Funktionalität dieses Algorithmus zu gewährleisten müssen die „Anpassungen an bestehenden Algorithmen“ umgesetzt worden sein.*

Das Verfahren zum Ermitteln der Summe zwischen zwei Werten besteht aus drei Methoden:

1. „summeZwischenMinMax“: Diese Methode erhält als Argument die Wurzel des Baums, den minimalen Wert und den Maximalen Wert zum Ermitteln der Summe zwischen diesen Beiden. Sie subtrahiert die Summe aller Werte, die kleiner als der minimale Wert sind, und aller Werte, die größer als der Maximale Wert sind, von der Gesamtsumme alle vorkommenden Werte im Baum. Dadurch wird sichergestellt, nur die Summe zwischen den beiden gesuchten Werten akkumuliert wird. Um alle Werte kleiner als Min und alle Werte größer als Max zu subtrahieren wird jeweils eine rekursive Methode (unten in der Formel zu sehen) verwendet. Diese Methoden werden in Punk 2 und Punkt 3 näher beschrieben.

**return** wurzel.getSum() - getSummeGroesserAlsMax((Knoten<Integer>) wurzel, max)

- getSummeKleinerAlsMin((Knoten<Integer>) wurzel, min);

1. „getSummeKleinerAlsMin“: Diese rekursive Methode erhält als Argument einen Knoten und den minimalen Wert, welcher vom Anwender eingegeben wurde. Diese Methode überprüft ob der Wert des Übergebenen Knotens kleiner, größer oder gleich dem gesuchten minimalen Wert ist.

**Falls er kleiner ist**: Die Funktion wird erneut auf den Linken Sohn des übergebenen Knotens aufgerufen.

**Falls er größer ist:** Nun muss anstatt nach Links, nach rechts im Baum navigiert werden um den minimalen Wert zu finden. Das bedeutet, dass alle Werte die Links und der Wert des Knotens selber auf jeden Fall Kleiner als der gesuchte minimale Wert sind. In diesem Fall muss die Summe dieser Werte beim erneuten Methodenaufruf auf den nächsten rechten Knoten mitgenommen werden. Die Werte werden bei der Rückgabe zu dem erneuten Methodenaufruf addiert. So wird sichergestellt, dass in der Methode „summeZwischenMinMax“ wirklich alle Werte die kleiner sind als der minimale Wert von der Gesamtheit subtrahiert werden.

**Falls er gleich ist:** Um alle Werte die kleiner als der gesuchte minimale Wert sind zu finden, wird die Summe des linken Sohns des gefundenen Knotens zurückgegeben. Falls es diesen Sohn nicht gibt, ist der Rückgabewert 0, da es nichts kleineres als diesen Wert gibt.

1. „getSummeGroesserAlsMax“: Diese rekursive Methode erhält als Argument einen Knoten und den maximalen Wert, welcher vom Anwender eingegeben wurde. Diese Methode überprüft ob der Wert des Übergebenen Knotens kleiner, größer oder gleich dem gesuchten minimalen Wert ist.

**Falls er größer ist**: Die Funktion wird erneut auf den Rechten Sohn des übergebenen Knotens aufgerufen.

**Falls er kleiner ist:** Nun muss anstatt nach rechts, nach Links im Baum navigiert werden um den Maximalen Wert zu finden. Das bedeutet, dass alle Werte die rechts und der Wert des Knotens selber auf jeden Fall größer als der gesuchte maximale Wert sind. In diesem Fall muss die Summe dieser Werte beim erneuten Methodenaufruf auf den nächsten linken Knoten mitgenommen werden. Die Werte werden bei der Rückgabe zu dem erneuten Methodenaufruf addiert. So wird sichergestellt, dass in der Methode „summeZwischenMinMax“ wirklich alle Werte die größer sind als der maximale Wert von der Gesamtheit subtrahiert werden.

**Falls er gleich ist:** Um alle Werte die kleiner als der gesuchte minimale Wert sind zu finden, wird die Summe des rechten Sohns des gefundenen Knotens zurückgegeben. Falls es diesen Sohn nicht gibt, ist der Rückgabewert 0, da es nichts kleineres als diesen Wert gibt.

## Komplexitätsuntersuchung

*Der Aufwand wird im Folgenden als Anzahl der Methodenaufrufe verstanden.*

**Worst-Case Szenario:**

Im Worst-Case-Szenario ist der Aufwand um die Summe zwischen den zwei gesuchten Werten zu finden: „2\*Höhe des Baums“. Das ist genau der Abstand zwischen den Werten, wenn der kleinste Wert links außen im Baum zu finden ist und der größte Wert rechts außen zu finden ist und nach diesen beiden Werten kein weiterer Wert weiter unten im Baum vorkommt.

**Best-Case Szenario:**

Im Best-Case-Szenario ist der Aufwand um die Summe zwischen den zwei gesuchten Werten zu finden: „2“. Dieser fall Tritt auf, wenn Min und Max der übergebenen Argumente genau der nächst kleinere und nächst größerer Wert nach der Wurzel sind und diese beiden Werte nicht in dem Baum vorkommen.

Von einer empirischen Untersuchung wird abgesehen, da der Aufwand je nach Position der zu suchenden Werte variiert. Ein durchschnittlicher Wert ist nicht repräsentabel, da dieser ein aus einem zu großen Spektrum entspringen würde. Im allgemeinen lässt sich sagen, dass sich der Aufwand wie folgt zusammensetzt: (Höhe des Knotens vom minimalen Limit) + (Höhe des Knotens vom maximalen Limit).

## UML Klassendiagramm

## Quellen

## Glossar