Inhaltsverzeichnis

[Anpassungen an bestehenden Algorithmen 2](#_Toc483750406)

[Verfahren zum Ermitteln der Summe zwischen zwei Werten 2](#_Toc483750407)

[Grafische Auswertung – Aufwandsanalyse 4](#_Toc483750408)

[Tabellarische Auswertung 5](#_Toc483750409)

[Quellen 5](#_Toc483750410)

[Glossar 5](#_Toc483750411)

**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. Dies wird anhand des verlinkten Binärbaumes umgesetzt.*

## 

## 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. „sumBetweenMinMax“: 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() - getMaxRightside((Knoten<Integer>) wurzel, max)

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

1. „getMinLeftSide“: 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 „sumBetweenMinMax“ 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. „getMaxRightSide“: 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 „sumBetweenMinMax“ 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.

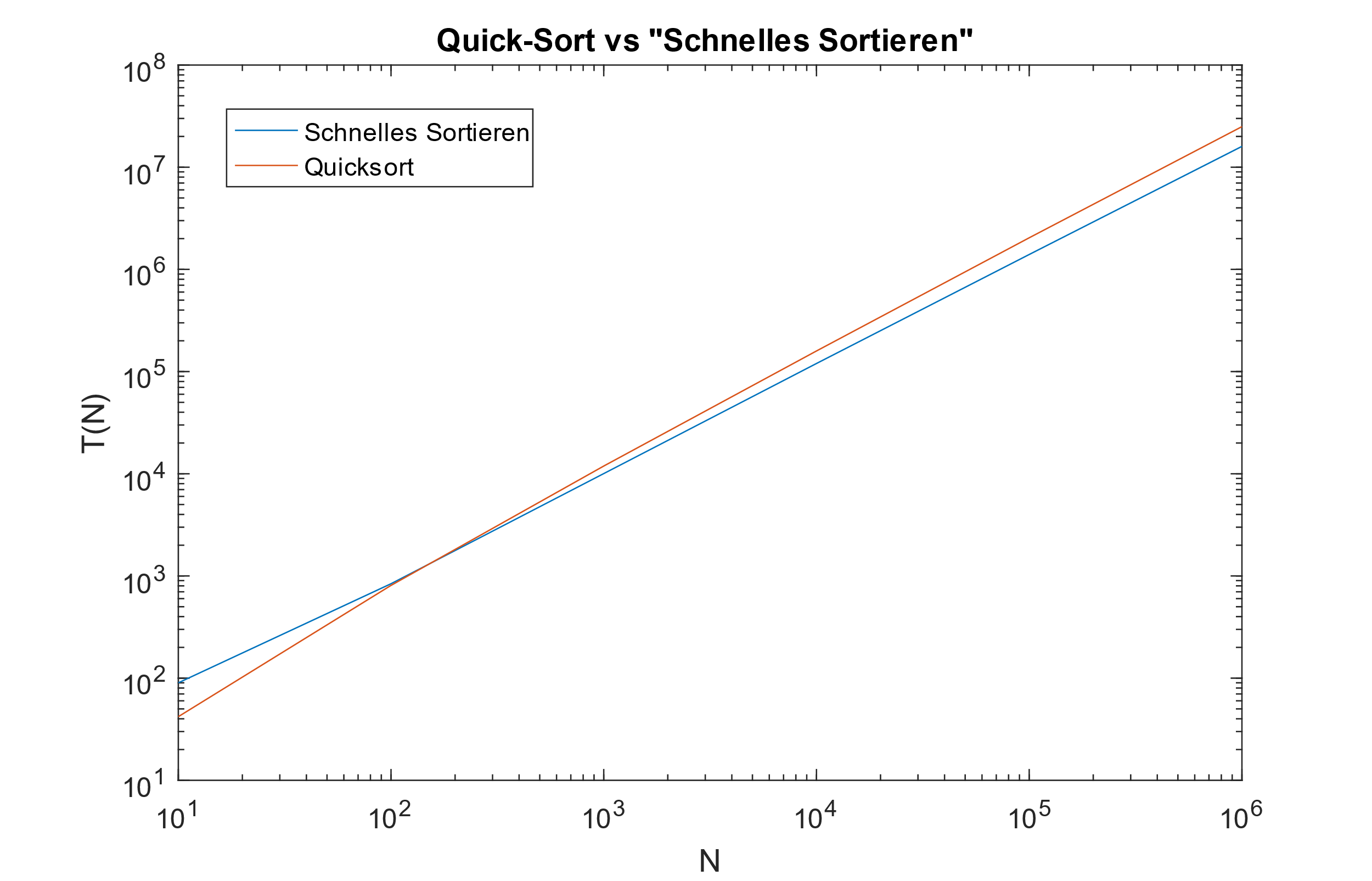
## Grafische Auswertung – Aufwandsanalyse

*Die Werte von Quicksort wurden aus 10 verschiedenen Messungen für jede Größe gemittelt. Die Werte von „Schnelles Sortieren“ sind immer gleich und stammen daher aus nur einer Messung.*

Der Wertebereich, der zu sortierenden Werte, erstreckt sich von 700\*N <= Wertebereich < 800 \*N, wobei N =, k =1,….,6 *.*Die Zwei Verfahren wurden mit zu sortierenden Listen in Größe 10, 100, 1000, 10000, 100000 und 1000000 getestet. Im Grafik 1 befinden sich beide Aufwände Grafisch in einer doppelt Logarithmischen Darstellung wieder.

Bis zu einer Listengröße von 100 ist noch der Vorsprung von Quicksort zu sehen. Ab einer Größe von 1000 steigt der Vorsprung vom „schnellen Sortieren“ dadurch, dass Quicksort im Aufwand weiter absteigt und nicht gleichbleibt. Der Unten angegebenen Tabelle können genaue Unterschiede in Zahlen ausgedrückt entnommen werden.

**Grafik 1**



## Tabellarische Auswertung

|  |  |  |
| --- | --- | --- |
| **ListenGroesse** | **Quicksort** | **"Schnelles Sortieren"** |
| 10 | 42 | 90 |
| 100 | 809 | 840 |
| 1000 | 11908 | 10050 |
| 10000 | 159178 | 120060 |
| 100000 | 2039082 | 1400070 |
| 1000000 | 24963195 | 16000080 |

## Quellen

Bild 1,2 und 3: <http://www.geeksforgeeks.org/counting-sort/>

Bild 4: <https://dotgeoteam.wordpress.com/2013/05/05/radix-sort/>

## Glossar

Ziffer: In diesem Dokument ist das Wort Ziffer gleichbedeutend mit dem Wort „Digit“ im englischen. Es ist also keine Zahl, sondern es beschreibt die Position in einer mehrstelligen Zahl.

Pivotelement: Ein Element, das zur Aufteilungsgrenze beim Sortieren genutzt wird.

Doppelt-Logarithmische Darstellung: Bei der doppelt-logarithmischen Darstellung sind beide Koordinatenachsen [logarithmisch](https://elearning.physik.uni-frankfurt.de/data/FB13-PhysikOnline/lm_data/lm_281/modul_2/teil_6/node18.html#logarfunkt) unterteilt.