# Energieeffizienz von Algorithmen

*Am Beispiel gängiger Sortierverfahren*

Eine Facharbeit von Jan Thomas

Fach: Informatik

Gymnasium Essen-Werden

Essen, Juni 2021

# Kapitel 1: Einleitung

## 1.1 Vorwort

In der folgenden Arbeit habe ich die Energieeffizienz von Algorithmen untersucht. Energieeffizienz spielt inzwischen in sämtlichen Bereichen eine größer werdende Rolle. Zum einen schafft der Klimawandel und die steigende Zahl an Elektrogeräten ein größeres Bewusstsein für die Energieeffizienz. Zum anderen werden mehr und mehr mobile Computer gebaut. Tablets, Handys und Laptops sollen viel leisten und der Akku, der in diesen Geräten verbaut ist, soll im besten Falle wenig geladen werden müssen. Neben der Entwicklung neuer Akkutechnologien ist es auch ein Ansatz, die Energieeffizienz von Programmen und Algorithmen zu messen und ein System zu konstruieren, was den Nutzeranforderungen entspricht.

Im Generellen interessiere ich mich für Energiewirtschaft – besonders in Verbindung mit Computer-Systemen. Durch den Unterrichtsstoff, der sich auch mit der Komplexität von Algorithmen befasste, kam mir die Idee, Sortieralgorithmen in die Richtung der benötigten Energie auszuwerten, um ein weiteres Kriterium zur Bewertung von Sortierverfahren zu erhalten.

## 1.2 Einführung

In der theoretischen Informatik werden Algorithmen durch ihre Laufzeit- und Speichereffizienz bewertet. Für gewisse Anforderungen in Computer-Systemen versucht man so, den besten Algorithmus zu finden oder zu entwerfen.

In großen Rechenzentren aber auch im privaten Bereich wird die Energieeffizienz zunehmend als weiteres Bewertungskriterium interessant, da es wirtschaftliche und funktionale Vorteile bietet. In dieser Facharbeit wird das Thema der Energieeffizienz aufgeschlüsselt. Durch Analysen bezüglich der Laufzeit- und Speichereffizienz und Wissen über das betriebene System wird versucht, einen Indikator für die Energieeffizienz zu finden, mit dem sich gängige Sortierverfahren in ihrer Energieeffizienz bewerten lassen.

## 1.3 Strukturierung

Die Facharbeit strukturiert sich in 4 Kapitel. Nach der Einleitung werden in den Grundlagen die wichtigen Aspekte zum Verständnis der Erarbeitungen dargelegt. Kapitel 3 ist der Hauptteil der Arbeit mit den Analysen zu den drei Sortierverfahren Quicksort, Mergesort und Countingsort . Darüber hinaus werden in diesem Kapitel die Verfahren auch hinsichtlich ihrer Effizienz ausgewertet. Im letzten Kapitel werden die Ergebnisse der Arbeit zusammengefasst. Es findet auch ein Ausblick auf andere Forschungsarbeiten zur Energieeffizienz statt. Die Anhänge umfassen ein Glossar mit den wichtigsten Begriffen, ein Literatur- und Abbildungsverzeichnis

sowie sonstige Anhänge mit weiterführenden Abbildungen und einer Sammlung von Wertetabellen, die die Messwerte und errechneten Daten darstellen.

### 1.4 Umsetzung & Umfang

Diese Facharbeit wurde mit Analysen an einem normalen Heim-PC durchgeführt. Dieser hat gute Leistungsdaten, stellt aber immer noch ein durchaus übliches System dar. Analysen zu den Sortierverfahren wurden mit der Java-Entwicklungsumgebung „BlueJ“ durchgeführt und auch implementiert. Die Facharbeit umfasst alle Ergebnisse dieser Analysen sowie Erklärungen, die durch äußere Quellen gestützt werden.

## Kapitel 2: Grundlagen

### 2.1 Was ist Energieeffizienz?

„Energieeffizienz beschreibt allgemein das Verhältnis eines bestimmten Nutzens – zum Beispiel die Bereitstellung von Licht oder Wärme – zu dessen Energieeinsatz. Je weniger Energie eingesetzt werden muss, umso energieeffizienter ist ein Produkt oder eine Dienstleistung“ (Bundesregierung, 2013).

Energieeffizienz ist ein interessanter Aspekt für Computersysteme, sowohl im privaten als auch im industriellen Sinne. Da immer mehr mobile Rechner gebaut werden (Tablets, Smartphones, Laptop-PCs), wachsen die Anforderungen an die Energieversorgung durch Akkus. Der Verbraucher wünscht längere Akkulaufzeiten. Da die Entwicklung neuer leistungsstärkerer Akkus sehr aufwendig ist, ist es eine gute Alternative, die Systeme energieeffizienter zu machen (vgl. Rivoire, et al., 2021 p. 1).

Im industriellen Bereich ist eine höhere Energieeffizienz im Wesentlichen aus Kostengründen interessant. Durch den Betrieb eines Großrechners mit einer Betriebsleistung von 10 MW können im Jahr 7 Millionen Dollar Kosten für elektrischen Strom und weitere 4-8 Millionen Dollar für die Kühlung anfallen (vgl. Rivoire, et al., 2021 p. 1). Sobald die Berechnungen, die auf diesen Rechnern stattfinden, weniger energieaufwendig werden, lassen sich diese Kosten pro sortierter Datenstruktur (oder ähnlicher Aufgaben) senken. Einen weiteren positiven Effekt stellt die Skalierbarkeit eines Großrechners dar, wenn dieser energieeffizient ist. Da die Systeme weniger Kühlung benötigen, lassen sich mehr Rechenkerne installieren, die wiederum zu schnelleren Berechnungen führen können.

### 2.2 Sortierverfahren & Auswahl

Diese Facharbeit soll die Energieeffizienz vor allem am Beispiel gängiger Sortierverfahren untersuchen. Für diese Untersuchungen wurden drei Sortierverfahren ausgewählt, die

unterschiedliche Vor- bzw. Nachteile aufweisen. Diese drei Verfahren werden im Folgenden vorgestellt.

### 2.2.1 Countingsort

Der sogenannte Countingsort ist ein gängiges Sortierverfahren für Arrays. Dem Namen entsprechend sortiert dieser Algorithmus die Datenstruktur, indem er die einzelnen Elemente abzählt (vgl. Studyflix, 2021). Dabei wird zur Hilfe eine weitere Datenstruktur, das Hilfsarray, erstellt. In das Hilfsarray werden dann jeweils die Kardinalitäten der Daten der ersten Struktur eingetragen.

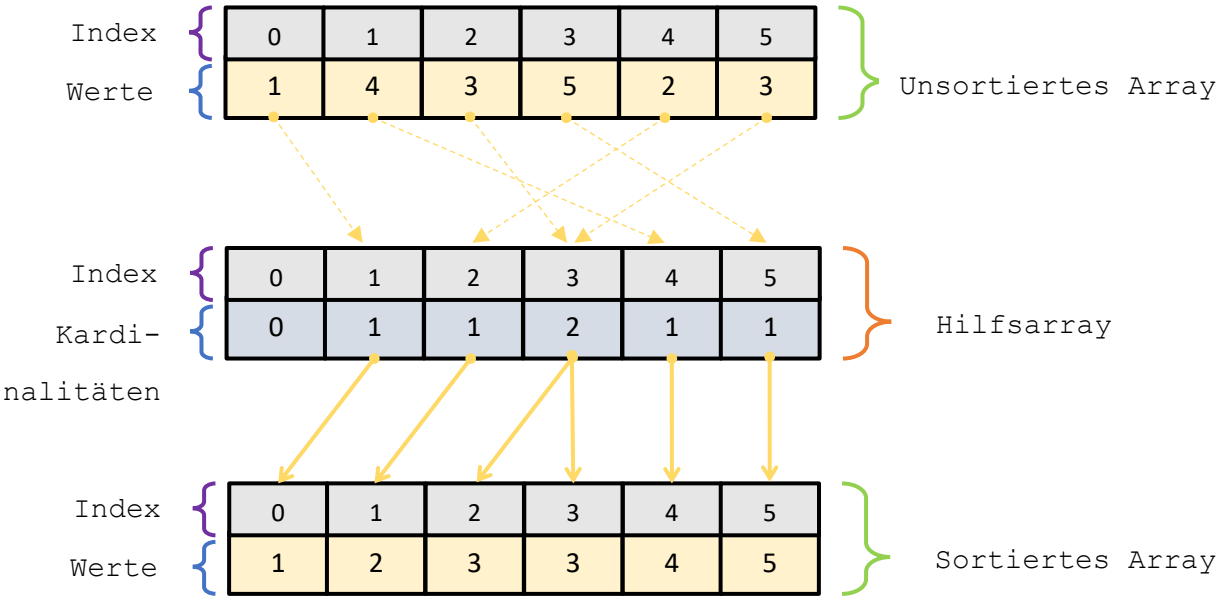


*Abbildung 1: Modellierte Arbeitsweise des Countingsort-Algorithmus*

In der Abbildung 1 ist die Arbeitsweise des Algorithmus dargestellt. Demnach muss für die Sortierung die Größe des größten Objekts bekannt sein. Das größte Objekt wird dann als Index eines neuen Hilfsarrays festgelegt, damit für jeden Wert eine Stelle im Hilfsarray vorhanden ist.

Nun folgt der erste Teil des Algorithmus. Von vorne (in der Abbildung links) beginnend wird das Array durchlaufen. Für den Wert der Zahl, an der sich der Zähler befindet, wird beim entsprechenden Index im Hilfsarray der Wert „1“ ergänzt. Nach dem Durchlauf spiegeln die Zahlen am entsprechenden Index die Häufigkeit der Zahl mit dem Wert des Indexes wider.

Wird das Hilfsarray in das sortierte Array übersetzt, so wird es von vorne bis hinten durchlaufen. An jeder Stelle, an der der Wert im Hilfsarray nicht „0“ ist, wird im finalen Array die Zahl der Array-Stelle ergänzt. Das wird so häufig wiederholt, bis der Wert an der entsprechenden Stelle „0“ ist. Für die Umsetzung sind also sowohl im Hilfsarray als auch im sortierten Array ein Zähler von Nöten.

### 2.2.2 Quicksort

Der Quicksort-Algorithmus ist ein Sortieralgorithmus, der rekursiv ein Array aufteilt, um es zu sortieren und somit nach dem „Teile und herrsche“-Verfahren arbeitet. Der Algorithmus kann durch Faktoren wie die verwendete Programmiersprache in seiner Arbeitsweise leicht abgewandelt sein (vgl. Studyflix, 2021).

Zur Sortierung wird im Quicksort-Algorithmus zunächst frei ein Pivotelement definiert. Dieses Element dient nachher als Vergleichselement. Anschaulich kann davon ausgegangen werden, dass es das Ziel ist, die Position des Pivotelements im unsortierten Array zu finden. Um diese Position zu finden, durchlaufen zwei Hilfsvariablen das Array. Diese werden stets mit dem Pivotelement verglichen. Am Ende dieses iterativen Vorgangs konnte die Stelle des Pivotelements definiert werden. Das Array wird dadurch aufgespalten, dass es nun einen Teil vor und einen Teil des Arrays hinter dem Pivotelement gibt. Daran wird der Algorithmus nun rekursiv ausgeführt. Eine „optimale Rekursion“ (Studyflix, 2021) wird durch „den Median“ (Studyflix, 2021) des Arrays als Pivotelement erreicht.

### 2.2.3 Mergesort

Der Mergesort-Algorithmus arbeitet ebenfalls nach dem „Teile und herrsche“-Verfahren. In der vorliegenden Implementierung besteht er – anders als Countingsort und Quicksort – aus zwei Methoden.

Zunächst teilt die erste Methode das Ausgangsarray so lang, bis so viele Arrays mit jeweils einem Objekt (einer Zahl) als Inhalt vorliegen wie das Ursprungs-Array lang ist. Die zweite Methode fügt diese Arrays nun zusammen, indem sie sie sortiert. Diese Sortierung findet so lang statt, bis nur noch ein sortiertes Array vorliegt. Nähere Informationen finden sich in der Quelle (vgl. Studyflix, 2021).

### 2.2.4 Auswahl

Diese drei Sortierverfahren wurden für die Analysen der Energieeffizienz ausgewählt, weil sie zu den schnellsten Sortierverfahren zählen. Dabei sind die Ansprüche an den Speicher unterschiedlich komplex, wodurch der Vergleich für die Energieeffizienz interessant wird, weil manche Sortierverfahren den Prozessor stärker beanspruchen als den Arbeitsspeicher oder umgekehrt.

## 2.3 Bewertungskriterien für Sortierverfahren

Zur Bewertung dieser Sortierverfahren gibt es im Allgemeinen zwei Kriterien: die Laufzeit- und die Speichereffizienz des Algorithmus.

### 2.3.1 Laufzeiteffizienz

Die Laufzeiteffizienz untersucht einen Algorithmus danach, wie schnell er berechnet werden kann. Dieser Wert wird bei Sortieralgorithmen in Abhängigkeit von den zu sortierenden Objekten

gebildet. Wie aus Abbildung 8 (vgl. Anhang 1) hervorgeht, gibt es bei Sortieralgorithmen vier gängige Verläufe: Den exponentiellen, den quadratischen, den linearen und den logarithmischen Verlauf. Diese hängen alle von der Anzahl der zu sortierenden Objekte ab. Demnach ist der günstigste Verlauf der Laufzeiteffizienz der logarithmische Verlauf, da dieser am geringsten ansteigt. Manche der modernsten Sortierverfahren weisen einen linearen Verlauf auf, dieser ist allerdings schon sehr effizient.

#### 2.3.2 Speichereffizienz

Die Speichereffizienz betrachtet den Speicher, der zur Sortierung einer Datenstruktur zusätzlich belegt wird. Sie ist Teil der Komplexität eines Algorithmus. Dieser Speicher spiegelt sich auf der technischen Ebene im Arbeitsspeicher wieder, steht also – im Vergleich zum Festplattenspeicher – nur in geringem Maße zur Verfügung. Die Speichereffizienz richtet sich in der Regel auch nach der Anzahl der zu sortierenden Objekte.

#### 2.3.3 Gewichtung in der Komplexität

Je nach Anforderung an das System werden die einzelnen Effizienzen unterschiedlich gewichtet. In der Regel muss also ein Anforderungsprofil festgelegt werden, um definieren zu können, ob ein Algorithmus geeignet ist, oder nicht.

# Kapitel 3: Erarbeitung

In diesem Kapitel soll die Vorgehensweise zur Analyse gängiger Sortierverfahren dargelegt werden. Um ein beherrschbares Maß an Sortierverfahren auszuwerten, werden die drei in Kapitel 2 erläuterten Sortierverfahren Quicksort, Mergesort und Countingsort zur Analyse herangezogen.

## 3.1 Durchführung der Analyse

Energieeffizienz am Computer, insbesondere im Hinblick auf konkrete Algorithmen, ist schwer zu messen. Es existiert kein Messverfahren, bei dem man nur bestimmte Programme auswerten kann. Somit ist es notwendig, dass andere Strategien entwickelt werden, die Aussagen über den tatsächlichen Stromverbrauch eines Algorithmus treffen lassen.

#### 3.1.1 Ziel der Analysen

Ziel der Analysen im Rahmen dieser Facharbeit ist es, Aussagen über den Stromverbrauch der verwendeten Sortierverfahren treffen zu können. Dabei muss auf die grundsätzliche Nutzung oder Effizienz geachtet werden. Damit ist gemeint, dass sowohl der spezielle Energieverbrauch des einzelnen Sortierverfahrens, als auch der komplette Energieauffand, der bei der Berechnung entsteht, erfasst und bewertet wird. Es sollen also verschiedene Parameter zur Auswertung von Algorithmen herangezogen werden.

#### 3.1.2 Strategie

Um die Effizienz der einzelnen Sortieralgorithmen herauszufinden, sollen Laufzeit-, Speichereffizienz sowie die Anzahl an Vergleichen für die Analyse herangezogen werden. Aus

Informationen der Hersteller werden Daten entnommen, wie viel Strom die einzelnen Rechenoperationen in Anspruch nehmen. So lassen sich Aussagen zur Energieeffizienz gut treffen.

Um die Messwerte möglichst frei von Messfehlern zu halten, sollen besonders viele Daten zu jedem einzelnen Algorithmus aufgenommen werden. Die Idee ist, dass alle drei Algorithmen das gleiche, zufällige Array sortieren. Dieses Array ist befüllt mit Zahlenwerten von 0 bis 1.000.000.000, füllt also gut die Hälfte des 32-Bit-Integer-Kontingents im positiven Bereich aus.

In einem selbst programmierten Analysealgorithmus sollen die Daten gesammelt werden. Dabei sollen unterschiedlich viele Objekte sortiert werden, das Array soll anschaulich also eine unterschiedliche Größe bekommen. So lassen sich in den einzelnen Parametern Aussagen über den Verlauf der benötigten Laufzeit, des benötigten Speichers oder der benötigten Vergleiche in Abhängigkeit der Anzahl an zu sortierenden Objekten treffen.

### 3.1.3 Auswertungskriterien

Im Allgemeinen sollen zur Analyse der Sortierverfahren drei Auswertungskriterien verwendet werden.

Die Laufzeit beschreibt die Zeit in Millisekunden, die der Algorithmus zur Sortierung benötigt. Dieser Wert ist ausschlaggebend für die Energie, weil die Laufzeit eines Algorithmus zum einen die Belastungsdauer der Recheneinheiten im Computer widerspiegelt, zum anderen aber auch die Menge an benötigten Grundverbrauchs-Strom darstellt, die zur Analyse der Energieeffizienz wichtig sein kann.

Der verwendete Speicher zielt lediglich auf den Arbeitsspeicher des Computers ab. Er gibt einen Indikator dafür, wie viel Strom durch den Arbeitsspeicher des Computers verbraucht worden ist, da mit mehr Speicher die Belastung steigt.

Die Anzahl der Vergleiche zeigt den ungefähren Arbeitsaufwand an und gibt darüber hinaus noch ein Indiz für einen möglichen Laufzeit-Verlauf.

Um ein Bewertungskriterium für den energieeffizientesten Algorithmus zu finden, muss ein Wertemaßstab festgelegt werden. Dazu wurde ein Index erstellt, der die beschriebenen Messwerte mit einbezieht und einen Anhaltspunkt für die Energieeffizienz gibt. Dieser Index ist:

$$\ni = \left( \frac{Speicher\ (Bytes)}{Arbeitsspeicher\ (Bytes)} + (Laufzeit * Grundverbrauch) + \left( \frac{Vergleiche}{(Prozessorleistung - Grundverbrauch)} * Taktfrequenz \right) \right) / 100.000$$

Die drei Bewertungskriterien werden in diesem Index unter Berücksichtigung des tatsächlichen Stromverbrauchs einbezogen. Somit wird die Aussage, die der Index möglich macht, besonders realitätsnah. Der Index lässt sich in drei „Elemente“ aufteilen.

Zunächst wird der verwendete Speicher (Speicher) durch die Speichergröße des Arbeitsspeichers (in diesem Falle 16 GB) geteilt. Es wird davon ausgegangen, dass zu dem 1 Watt Grundverbrauch im Arbeitsspeicher, unter Last noch 1 Watt hinzukommt, was sich darin ausdrückt. Der Anteil am Gesamtverbrauch ist sehr gering.

$$Element\ 1: Speichereffizienz = \frac{Speicher\ (Bytes)}{Arbeitsspeicher\ (Bytes)}$$

Danach wird die Laufzeit in den Index miteinbezogen. Diese wird mit dem Grundverbrauch multipliziert, da die Laufzeit ausschlaggebend für diesen Stromwert ist.

$$Element\ 2: Laufzeiteffizienz = (Laufzeit * Grundverbrauch)$$

Der letzte Teil der Formel zielt auf die Anzahl und die Berechnungsdauer der Vergleiche ab. Dieser Teil der Formel orientiert sich an der Berechnung der Prozessorleistung in „Instruktionen pro Sekunde“. Die Formel dafür lautet: $L = \frac{1}{CPI*T}$ [IPS] (vgl. Die Chemie-Schule, 2021).

$$Element\ 3: Vergleiche = \left(\frac{Vergleiche}{(Prozessorleistung - Grundverbrauch)} * Taktfrequenz\right)$$

Dabei stellt L die Leistung des Prozessors dar, die in IPS (Instruktionen pro Sekunde) oder MIPS (Millionen Instruktionen pro Sekunde) angegeben wird. CPI ist die Anzahl von Zyklen, die zum Ausführen eines Vorgangs benötigt werden. T stellt die Taktfrequenz des Prozessors dar. Da noch der Stromverbrauch des Prozessors einbezogen werden musste, der nach Angaben des Herstellers bei durchschnittlich 65 Watt liegt (vgl. Advanced Micro Devices, 2021) und davon ausgegangen wurde, dass für einen Vergleich nur ein Zyklus benötigt wird, ergab sich dieses Element durch:

$$E_{Vergleiche} = \frac{Vergleiche}{IPS * (Prozessorverbrauch - Grundverbrauch_{Prozessor})}$$

V = Vergleiche

IPS = Instruktionen pro Sekunde

Prozessorverbrauch = 65 Watt

Grundverbrauch = 1,15 Watt

CPI = 1

$$E_{Vergleiche} = \frac{Vergleiche}{\frac{1}{CPI * T} * (65 - 1{,}15)}$$

$$E_{Vergleiche} = \frac{Vergleiche}{\frac{1}{1 * T} * 63{,}85}$$

$$E_{Vergleiche} = \frac{Vergleiche * T}{63{,}85}$$

Schlussendlich wird die errechnete Summe aller drei Elemente durch 100.000 geteilt, um den Wert zu senken. Das ist möglich, weil es sich bei der Berechnung um einen Index handelt.

Im weiteren Verlauf werden einzelne Teile der Gleichung als „Elemente“ bezeichnet. Diese Elemente sind hier grafisch dargestellt.

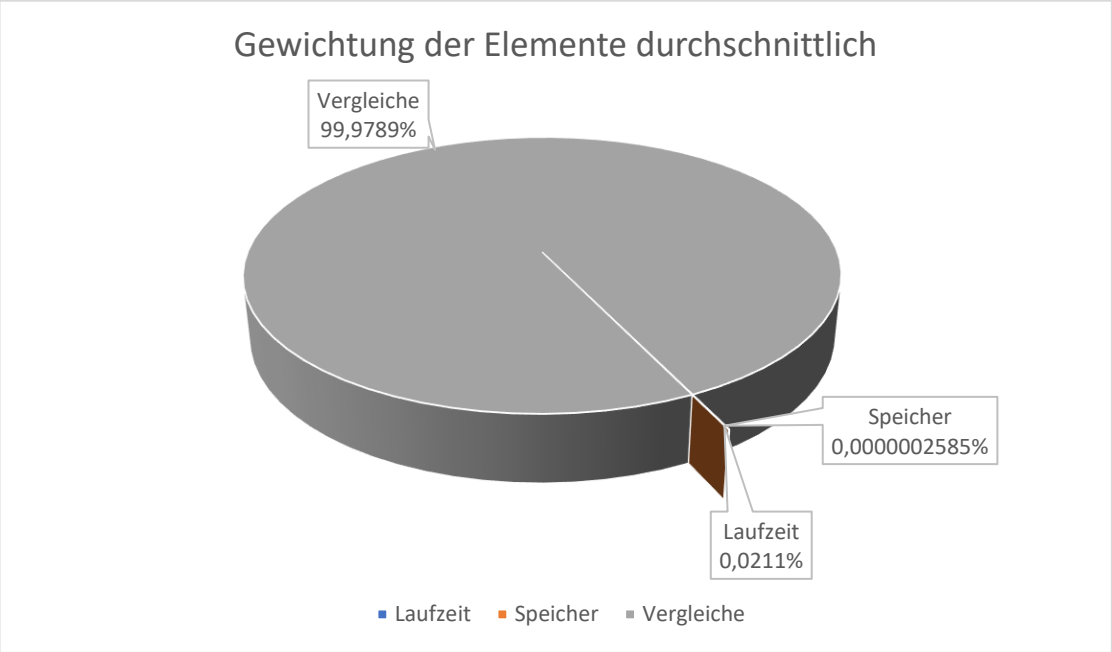


*Abbildung 2: Darstellung der drei Index-Elemente und ihrer Gewichtung (real)*

In der Abbildung 2 ist die Gewichtung der drei Elemente Speicher, Laufzeit und Vergleiche im Durchschnitt über die Index Berechnungen aller drei Sortierverfahren in einem Objektanzahl-Bereich von 0 bis 200.000 dargelegt. Es zeigt sich, dass die Anzahl der Vergleiche am wichtigsten ist, da der Prozessor die Komponente im PC ist, die am meisten Strom verbraucht. In Abbildung 9 im Anhang dieser Facharbeit ist die Gewichtung der drei Elemente für die einzelnen Sortierverfahren separat dargestellt.

## 3.2 Implementierung eines Analyse-Verfahrens

Um die vorgestellten Sortierverfahren analysieren zu können, war es wichtig, einen Algorithmus zu entwerfen, der diese Aufgabe übernimmt. Das hat den Vorteil, dass man viele Messwerte auswerten kann, da der Algorithmus diese automatisch und systematisch aufnimmt.

### 3.2.1 Vorgehensweise

Die Idee bestand darin, einen Algorithmus zu entwerfen, der für die drei gewählten Sortierverfahren (Countingsort, Quicksort und Mergesort) Analysen zur Laufzeit, zum verwendeten Speicherplatz im Arbeitsspeicher und zu den getätigten Vergleichen während der Ausführung durchführt.

Um einen solchen Algorithmus zu entwerfen, wurde zunächst ein Java-Code in der Entwicklungsumgebung von „BlueJ“ angelegt. Für jedes Sortierverfahren wurde eine Klasse erstellt. Darüber hinaus wurde auch eine Klasse „Analyse“ erstellt, die Methoden zur Analyse der Sortierverfahren enthält.

Die drei Sortier-Klassen „QuickSort“, „MergeSort“ und „CountingSort“ weisen im Allgemeinen den gleichen Aufbau auf.

Im Konstruktor dieser Klassen muss als Parameter ein Array bestehend aus Integer-Werten eingegeben werden. Die Klasse ruft anschließend im Konstruktor eine Methode auf, die den Namen der Klasse trägt, also das jeweilige Sortierverfahren. Diese Methode spiegelt die eigentliche Sortierung wider. Am Ende der Methode wird das Array zurückgegeben. Alle Methoden zum Sortieren sind „private“ also nicht von außen aufrufbar.

Eine Besonderheit stellt die Implementierung der Klasse „MergeSort“ dar. Diese besteht nämlich aus zwei Methoden. Zunächst wird durch den Konstruktor die Methode „mergesort()“ aufgerufen. Diese teilt das Array. Die Arbeitsweise dieses Algorithmus ist rekursiv. Anschließend werden dann die einzelnen geteilten Arrays rekursiv durch die Methode „combine()“ aufgerufen. Es entsteht ein sortiertes Array.

Die Klasse „Analyse“ wurde so entworfen, dass sie die Klassen der Sortierverfahren für ausgiebige Tests benutzen kann. Um diese Tests durchzuführen ist kein Parameter im Konstruktor nötig. Tests müssen mit einem manuellen Methodenaufruf gestartet werden. Zum Ausführen der einzelnen Tests wurden verschiedene Methoden entworfen.

Die Vorgehensweise sah vor, dass die Klasse „Analyse“ umfangreiche Tests automatisch durchführt. Das bedeutet, dass die Anzahl an Objekten während eines Tests gesteigert wird, um dann Messdaten entnehmen zu können. Diese Messdaten müssen während der Ausführung gespeichert werden. Auch ist zu beachten, dass die Messdaten später gut entnehmbar sein mussten, um sie später mit einer Excel-Tabelle auswerten zu können.

### 3.2.2 Arbeitsweise des Algorithmus

Dieser Abschnitt beschreibt nun konkret die Arbeitsweise der Klasse „Analyse“. Diese Klasse weist einige Besonderheiten auf.

Der Algorithmus dient dazu, umfangreiche Tests mit den drei ausgewählten Sortierverfahren durchzuführen. Diese Tests werden in der Methode „allAnalysis()“ durchgeführt und die Messdaten ebenfalls abgespeichert und anschließend ausgegeben.

Zunächst erstellt diese Methode insgesamt 9 Arrays des Datentyps „long“. In diesen Arrays sollen für jeden der drei Algorithmen für bis zu 100 unterschiedliche Messungen die verwendete Laufzeit (Array: quick, count, merge), der verwendete Speicherplatz im Arbeitsspeicher (Array: qumem, comem, memem) und die Anzahl der getätigten Vergleiche (Array: quvgl, covgl, mevgl) ausgegeben werden. Darüber hinaus werden neben diesen neun Arrays drei Integer-Werte erstellt. Der Integer-Wert „step“ gibt den Abstand als Zahl an Objekten zwischen zwei Messungen an. Das Integer „border“ stellt die obere Grenze an zu sortierenden Objekten dar. Ist diese Grenze erreicht, werden keine Analysen mehr durchgeführt. Das Integer „counter“ dient als Hilfsvariable bei der Befüllung der neun Arrays zur Speicherung der Messdaten. Die beschriebenen Variablen helfen bei der Automatisierung der Messdaten, da sie dynamisch anzupassen sind, und die Messwerte speichern können. Die Methode kann, aufgrund der Länge der Arrays, maximal 100 Messungen durchführen.

Nach der Deklaration dieser Hilfsvariablen findet die eigentliche Analyse statt.

Die Laufzeit wird bei jedem Algorithmus durch den Methodenaufruf der Methode „analysis<NamedesSortierverfahrens>()“ analysiert. Die aufgerufene Methode bekommt als Parameter die Anzahl an zu sortierenden Objekten und die sogenannte „quantity“, also Häufigkeit, übermittelt. Die Methode ruft dann den Sortieralgorithmus mit einem Array der angegebenen Länge auf. Durch den zweiten Parameter „quantity“ wird ein Durchschnittswert erstellt. Die Sortiermethode wird auf die jeweilige Länge des Arrays aufgerufen, so oft, wie der Wert dieser Variable ist. Danach wird durch den Wert geteilt, wodurch ein arithmetisches Mittel der Messwerte entsteht, welches dem „Average-Case“ näherkommt. Im Fall dieser Methode wurden die Tests stets 3-fach ausgeführt. Jeder Messwert der Laufzeit ist also der Durchschnittswert dreier Methodenaufrufe.

Der Speicher jedes Sortierverfahrens wird hochgezählt. Dieser wurde durch Betrachtung des Quelltextes individuell ausgewertet und richtet sich bei jedem Sortierverfahren nach der Anzahl der zu sortierenden Objekte und entsprechender Hilfsvariablen. Die Ausgabe erfolgt in Bytes, da die errechneten Werte stets noch durch 8 geteilt werden.

Auch die Anzahl der getätigten Vergleiche erfolgt in Form einer Berechnung. Die Berechnung findet aufgrund des ermittelten „Average-Case“ des Sortierverfahrens statt.

### 3.2.3 Besondere Codezeilen

Um die Auswertung der Messwerte zu vereinfachen, wurden besondere Aspekte der Programmierung angewandt.

Einer dieser Aspekte ist die Verwendung des Datentyps „long“. Dieser Datentyp hat ein Speichervolumen von -9.223.372.036.854.775.808 bis 9.223.372.036.854.775.807 und kann somit deutlich größere Zahlen abspeichern als der Datentyp Integer (vgl. Microsoft Corporation, 2021). Darin liegt der Vorteil den er im Quelltext hat. Manche Messdaten könnten nämlich über das Integer-Kontingent hinauswachsen.

Darüber hinaus wurden zwei Java-Klassen des „java.util“ Paketes verwendet, welche den Programmtext erweitern.

Die Klasse „Random“ ist notwendig, damit Zufallswerte für die Belegungen im zu sortierenden Array gebildet werden können. Sie erzeugt dabei eine Zahl, die nur eine Näherung zum Zufall darstellt, also als „pseudorandom“ (Oracle, 2021) bezeichnet werden kann. Für die Tests ist diese Näherung allerdings ausreichend.

Die Klasse „Scanner“ stammt ebenfalls aus „java.util“ und ermöglicht es, Eingaben in der Konsole vom Programm auslesen zu lassen. Sie wurde angewandt, damit beim Anzeigen der Messwerte eine Trennung der einzelnen Kategorien einfacher wird und die Messwerte in eine Excel-Tabelle kopiert werden können. Zur Funktionalität trägt sie nicht bei.

### 3.2.4 Güte der Messwerte

Die Messwerte der Laufzeit basieren auf der Systemzeit des Computers, die in Millisekunden ausgegeben wird und sehr genau ist. Die Länge der Berechnung bei einzelnen Sortierverfahren kann durch Hintergrundprozesse allerdings verlängert werden, was durch die Bildung des Durchschnitts (siehe 3.2.2 Arbeitsweise des Algorithmus

) verhindert wird.

Der Speicher wird aus den Daten zu den Hilfsvariablen und der Struktur der Sortieralgorithmen gelesen. Diese Messwerte können vom tatsächlichen Speicher abweichen, weil eventuell in der Verarbeitung gewisse Teile komprimiert werden, beziehungsweise nicht alle Hilfsvariablen tatsächlich bis zum Ende benötigt werden. Solche Messungenauigkeiten sind zwischen allen Algorithmen gleich, wodurch der Wert belastbar bleibt.

### 3.2.5 Schwierigkeiten

Bei der Implementierung ergab sich die Schwierigkeit, dass der Countingsort-Algorithmus durch seine Syntax ein Problem bei der Analyse von großen Objektanzahl-Bereichen hatte. Die zugelassene Array-Länge durch den Arbeitsspeicher und die Array-Datenstruktur an sich machten eine Sortierung von bis zu 200.000 Objekten im Bereich zwischen 0 und 1.000.000.000 möglich.

Die anderen Sortierverfahren können diesen Bereich weit überschreiten und deutlich größere Arrays sortieren. Dieses Problem ließ sich dadurch lösen, dass besonders viele Messwerte im unteren Bereich aufgenommen wurden, die dann über eine Ausgleichsgerade auf höhere Werte extrapoliert werden können.

## 3.3 Messungen

In diesem Abschnitt sollen die Analysen zu den einzelnen Sortierverfahren dargelegt werden. Die Aussagen zur Energieeffizienz stammen aus dem erstellten Index (siehe 3.1.3 Auswertungskriterien

### 3.3.1 Mittel & Durchführung der Analyse

Um Messwerte aufnehmen zu können, wurden verschiedene Analysen an einem Heim-PC durchgeführt. Dieser Computer hat einen 8-Kern Prozessor (AMD Ryzen 7 1700 Eight-Core Processor) mit 3,2 GHZ und 16GB Arbeitsspeicher.

Der Grundverbrauch des PC lässt sich über Daten des Herstellers aufschlüsseln.

Nach einem Test des Prozessor-Herstellers AMD wies der Prozessor des Computers bei einem Test der Leistung eine Arbeit von 115 Watt auf. Diese Arbeit wurde unter voller Belastung des Prozessors gemessen. Geht man davon aus, dass der Test des Herstellers unter einer Auslastung von 100 % durchgeführt wurde, liegt der Grundverbrauch im Prozessor bei 1,15 Watt (vgl. Advanced Micro Devices, 2021).

Die Grafikkarte, die an den Sortiertests keinen Beitrag hat, trägt zum Grundverbrauch des PCs bei und wird deshalb mit einberechnet. Der Grundverbrauch wurde durch eine Metrik-Anzeige auf 8 Watt bestimmt.

Der tatsächliche Stromverbrauch des Arbeitsspeichers lässt sich nicht feststellen. Häufig sind Angaben, dass der Arbeitsspeicher kaum Leistung des Prozessors ausmacht. Ein unterstützendes Indiz ist, dass der RAM nicht separat mit Strom versorgt wird, und die Versorgung über das Mother-Board vollkommen ausreicht. Es wird daher die Annahme getroffen, dass der Arbeitsspeicher im Ruhemodus 1 Watt verbraucht.

Nach dieser Berechnung liegt der Grundverbrauch des Computers bei 10,15 Watt. Weitere Komponenten des PC könnten diesen Wert zwar erhöhen (vgl. cheapenergy24, 2021), können aber außer Acht gelassen werden, da die Analysen auf eine Aussage zur Effizienz unter Berücksichtigung angenommener „Großverbraucher“ in Form eines Index abzielen.

### 3.3.2 Analyse am Beispiel kleiner Computer (PC)

Nun werden die Ergebnisse der Analysen dargelegt. Es wurden Tests im Wertebereich zwischen 0 und 16 Millionen Objekten durchgeführt, wobei nicht alle Anzahlen in jedem Verfahren angewandt wurden (siehe 3.2.5 Schwierigkeiten

). Analysen ohne Objekte (0) wurden auch durchgeführt, tragen aber nicht zu den Aussagen bei, da sie keine Sortierung darstellen.

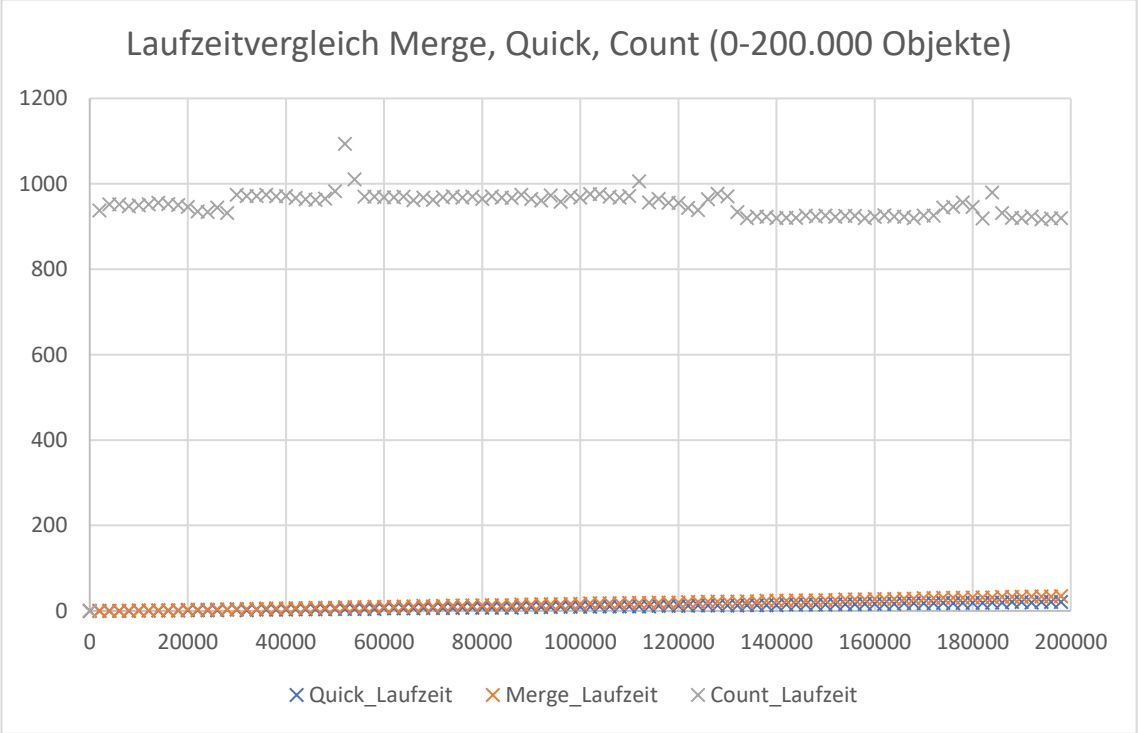


*Abbildung 3:Laufzeitanalysen von Quick-, Merge- und Countigsort*

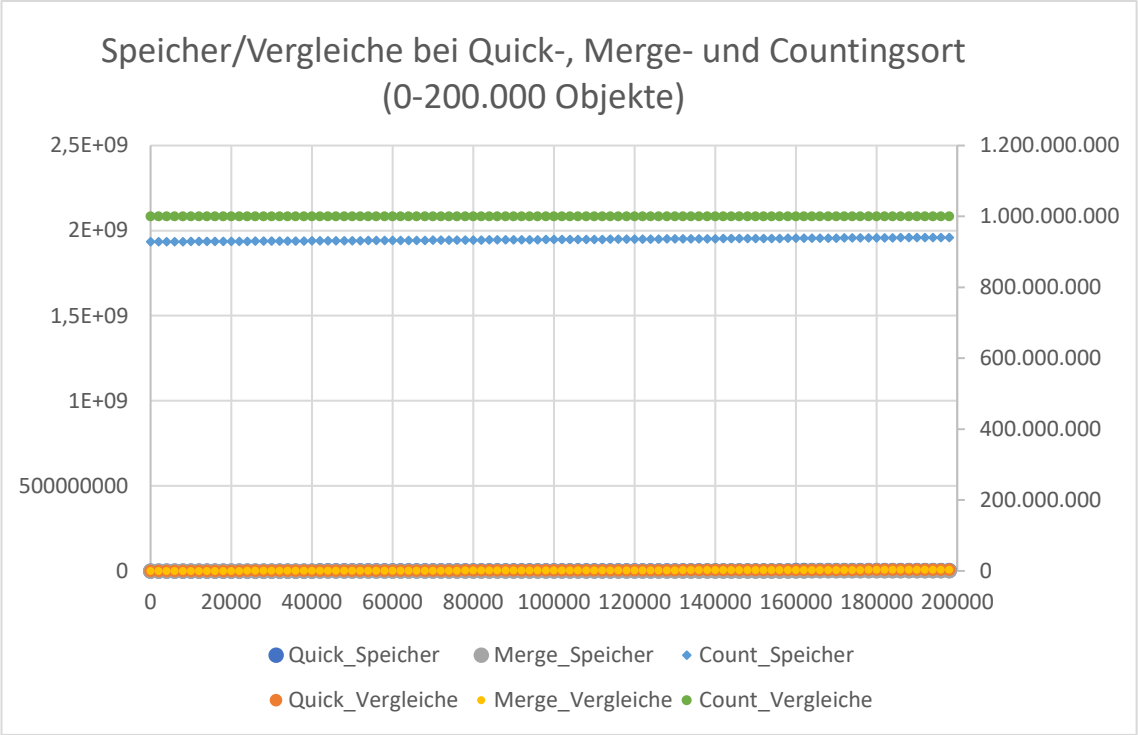


*Abbildung 4: Speicher-/Vergleichsanalyse von Merge-, Quick- und Countingsort*

In diesen Abbildungen zeigen sich aufgeschlüsselt die bloßen Messdaten des Analyseverfahrens. In diesem Beispiel wurde in einem Wertebereich von 0-198000 Objekten jedes Verfahren analysiert. Deutlich zu erkennen ist, dass in diesem Bereich kein markantes Wachstum deutlich wird, der Countingsort allerdings in jedem Messwert höhere Werte als die anderen Verfahren erzielt.

Zum Vergleich der Daten wurden sie mit dem Energieeffizienzindex berechnet, wobei mit Merge- und Quicksort Analysen bis zu Werten von 16 Millionen Objekten durchgeführt wurden. Die Testdaten wurden zusammengefasst.

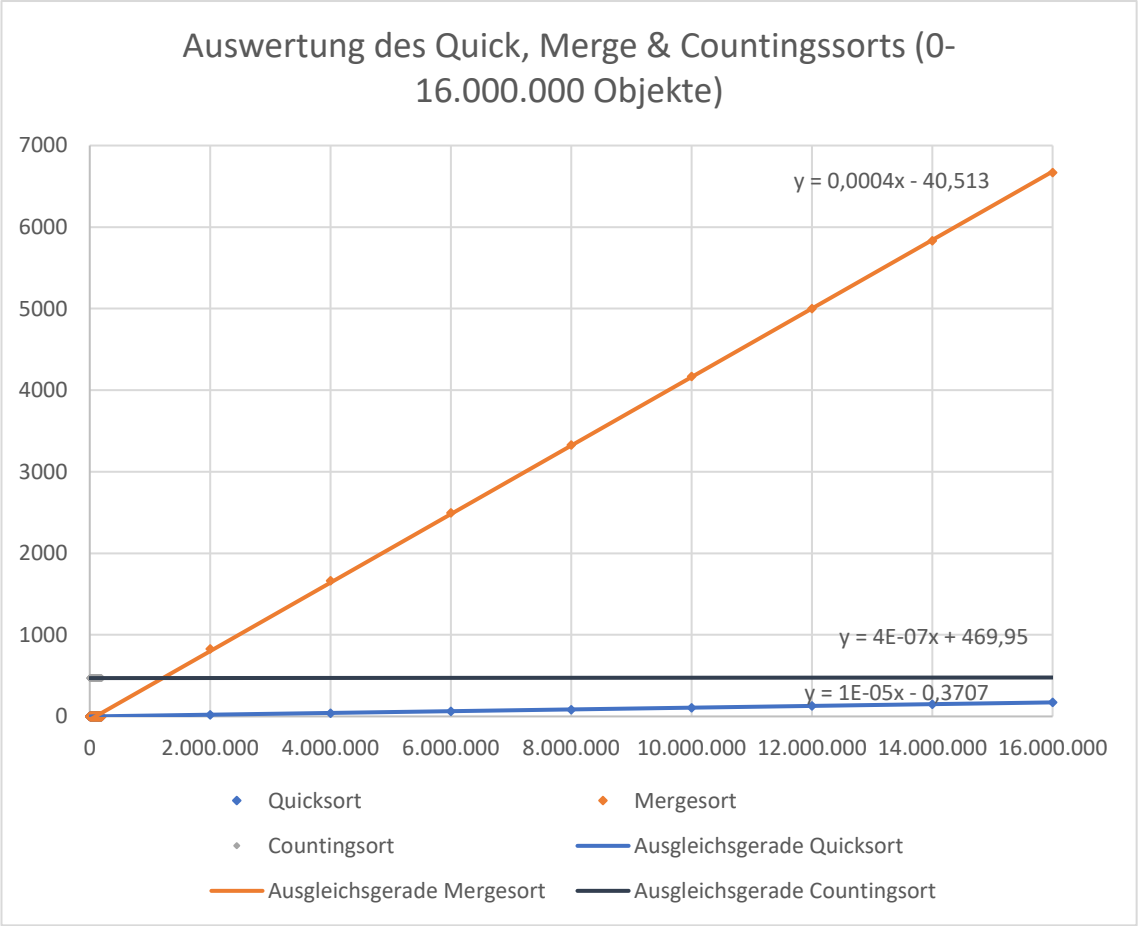


*Abbildung 5: Die Auswertung der drei Sortierverfahren mit dem Energieeffizienz-Index*

Das Diagramm zeigt die Berechnungswerte des Energieeffizienzindex mit den vorliegenden Messwerten, die durch eine Ausgleichsgerade in ihrem Verlauf gekennzeichnet wurden. Es ist zu erkennen, dass der Mergesort das ineffizienteste Verfahren ist, da er größere Werte als der Quicksort erzielt.

Den Messwerten zufolge ist der Quicksort-Algorithmus der energieeffizienteste Sortier-Algorithmus dieser drei Beispiele, da er auch bei einer Objektanzahl von 16 Millionen Objekten noch deutlich unter dem Wert des Countingsorts liegt.

### 3.3.2 Bedeutung für große Rechenzentren

Die Bewertung der Energieeffizienz für große Rechenzentren erfordert, dass die Sortierverfahren in größeren Objekt-Anzahl-Bereichen untersucht werden. Um eine Aussage über die Energieeffizienz der Algorithmen für große Rechenzentren treffen zu können, werden die Messwerte des Heim-PCs erweitert.

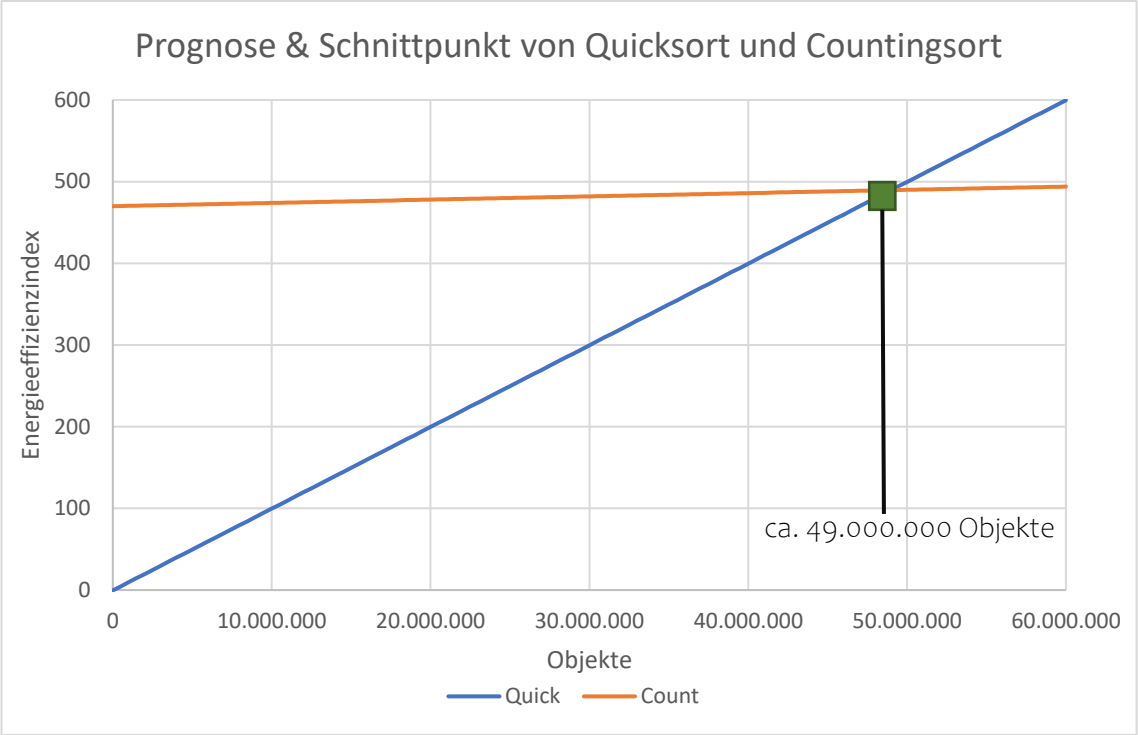


*Abbildung 6: Skalierung der beiden energieeffizienten Sortierverfahren auf Heim-PCs. Die Daten sind prognostiziert.*

Um diese Skalierung möglich zu machen, wurden die Formeln der Ausgleichsgeraden aus Abbildung 5 entnommen und auf höhere Werte erweitert. Es wird deutlich, dass der Quicksort ab einer Objektanzahl von etwa 49.000.000 Objekten nach dem Energieeffizienzindex ineffizienter wird, als der Countingsort-Algorithmus. Diese Anzahl an Objekten ist in der Industrie gering, da dort in der Regel große Mengen an Daten verwaltet werden müssen.

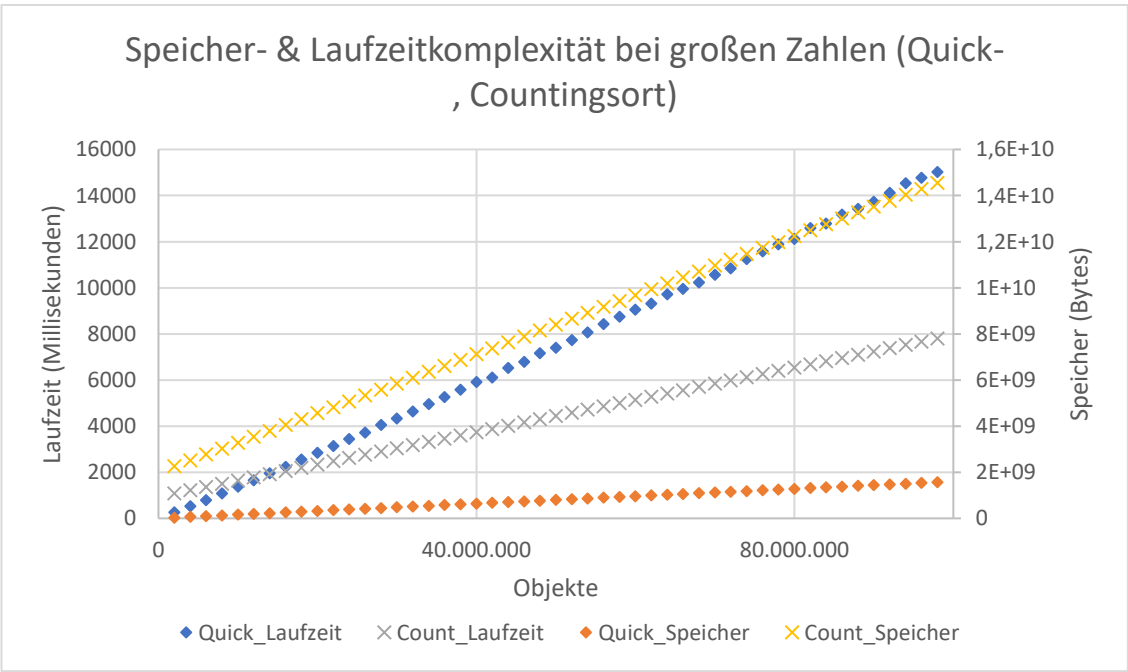


*Abbildung 7: Darstellung der Speicher- und Laufzeitkomplexität für den Quicksort und den Countingsort. Die Daten sind prognostiziert.*

Es wird deutlich, dass die Laufzeit des Quicksorts die des Countingssorts früh überschreitet. Da die Laufzeit (siehe 3.2.2 Arbeitsweise des Algorithmus

der wichtigste Aspekt des Energieeffizienz-Indexes ist, wird der Algorithmus so auf lange Sicht ineffizienter.

Wenn man nun die Messwerte eines kleinen Computers also auf die Anwendung großer Rechenzentren hochskaliert, dann zeigt sich, dass der Countingsort an die Stelle des Quicksorts tritt und den energieeffizientesten Algorithmus darstellt.

# Kapitel 4: Zusammenfassung

## 4.1 Zusammenfassung der eigenen Ergebnisse

Grundsätzlich lässt sich festhalten, dass Energieeffizienz ein wichtiger Aspekt der Komplexitätsanalyse für Sortierverfahren ist. In der privaten Anwendung kann durch Energieeffizienz vor allem die geforderte Akkulaufzeit verlängert werden und portable Geräte können praktikabler werden. In industrieller Hinsicht ist Energieeffizienz ebenfalls wichtig, da die großen Rechenzentren sehr viel elektrischen Strom verbrauchen, was zu Steigerungen in den Betriebskosten führt. Darüber hinaus muss der Strom produziert werden, was zu großen Teilen nicht klimaneutral möglich ist, wodurch Energieeffizienz ökonomischen und ökologischen Nutzen bekommt.

Sortierverfahren werden in ihrer Komplexität meistens über die Laufzeit- und Speichereffizienz aufgeschlüsselt, die eng mit den getätigten Vergleichen verknüpft sind. Aus Messungen zu diesen Parametern und Wissen über das Gerät, welches man zur Berechnung benutzt, können näherungsweise Aussagen zur Energieeffizienz des Algorithmus getroffen werden.

Auf kleinen Rechnern, die verhältnismäßig wenig Objekte zu sortieren haben, eignet sich der Quicksort als energieeffizienter Algorithmus. Dieser weist den Vorteil auf, dass er sehr Speicher effizient ist.

In großen Rechenzentren wandelt sich das Bild. Da in solchen Zentren eher mehr Objekte sortiert werden müssen und die Rechenoperationen parallel durchgeführt werden, empfiehlt sich der Countingsort. Dieser wird schätzungsweise ab einer Objekt-Anzahl von etwa 49 Millionen Objekten effizienter als der Quicksort und eignet sich somit besser zur Sortierung großer Mengen von Nutzerdaten.

Grundsätzlich muss zwischen Heim- und Großrechnern unterschieden werden. Es zeigt sich aber generell, dass die Energieeffizienz von Algorithmen eng mit der Laufzeit, dem verwendeten Speicher und den Vergleichen zusammenhängt. Als Grundaussage lässt sich formulieren: Ein Algorithmus, der wenig Vergleiche zum Sortieren braucht, ist energieeffizient.

Insgesamt zeigt sich, dass das wichtigste Bauteil, um einen energieeffizienten Rechner zu bauen, der verwendete Prozessor ist. Dieser muss bei möglichst wenig Verbrauch an elektrischem Strom möglichst schnell Berechnungen durchführen können.

## 4.2 Ausblick: JouleSort

In der Wissenschaft rund um die Energieeffizienz von Algorithmen haben sich bereits Verfahren entwickelt, die Energieeffizienz aufschlüsseln. Eines dieser Verfahren ist der JouleSort, der hier kurz beschrieben werden soll.

JouleSort ist ein sogenanntes „benchmark“, also ein Analyseprogramm, welches dazu dient, die Energieeffizienz von Sortieralgorithmen in drei Kategorien zu nutzen. Nach dem Aufsatz zur Vorstellung dieses Verfahrens gibt es heutzutage drei Anwendungsbereiche für Algorithmen: „mobile, desktop, and server“ (Rivoire, et al., 2021 p. 2). Diese drei Sortierverfahren überdecken die Computer-Arten vom Smartphone über den normalen PC zuhause bis hin zu großen Rechenzentren. Dabei werden alle drei Systeme getestet, um die Energieeffizienz ermitteln zu können. Die Vergleichbarkeit wird dadurch hergestellt, dass die Verfahren unter „peak use“ (Rivoire, et al., 2021 p. 2), also unter höchster Auslastung aller Komponenten, durchgeführt werden. Dabei zeigte sich allerdings – wie in den Messungen dieser Arbeit – dass der Prozessor kaum belastet wird (vgl. Rivoire, et al., 2021 p. 7). Das genaue Verfahren sieht vor, dass eine konstante Anzahl von Daten sortiert wird, die in drei Speicherkategorien durchgeführt wird. Das effektivste Verfahren in den drei Kategorien ist das, welches insgesamt am wenigsten Energie für die Sortierung verwendet hat.

JouleSort zeigt, dass die Frage der Energieeffizienz immer wichtiger wird, da sie gerade in der Industrie ökonomischen Nutzen hat.

## 4.3 Anmerkungen

Der in dieser Facharbeitet verwendete Quellcode für die Sortieralgorithmen stammt aus verschiedenen Quellen. Die Implementierung zum Quicksort und Countingsort stammt aus dem BlueJ-Projekt „Sortiertest-Implementierung“ aus dem Informatikunterricht und wurde lediglich für die Analyse angepasst. Die Mergesort-Implementierung leitet sich aus einem Codebeispiel der Website „Studyflix“ (Studyflix, 2021) ab. Das eigentliche Analyse-Verfahren wurde selber entworfen und auch implementiert.

# Glossar

**Array**
Ein Array ist eine Datenstruktur, die Objekte eines bestimmten Typs oder Daten eines Datentyps abspeichern kann. Das Array übernimmt dabei den Datentyp. Die Länge eines Arrays ist vordefiniert und nicht veränderbar. Ein eindimensionales Array – wie es in dieser Arbeit ausschließlich verwendet wird – ähnelt anschaulich einer Tabelle mit zwei Zeilen, die eine ist der Indikator der Stelle, die andere der Datentyp mit dem entsprechenden Wert (vgl. Studyflix, 2021).

**Average-Case**
Der Average-Case „beschäftigt sich […] mit" der Komplexität von Algorithmen „bei ‚typischen' Eingaben" (Average-Case Analyse, 2016). Der Average-Case in einer bestimmten Komplexitäts-Kategorie (Laufzeit/Speicher/Vergleiche) gibt also den Durchschnittswert an, der bei einer entsprechenden Messung des Algorithmus entstehen sollte.

**CPI**
„Die CPI-Rechenleistung beschreibt die Anzahl der für die Ausführung der einzelnen Instruktionen benötigten Taktzyklen. Bei der Ermittlung des CPI-Wertes dividiert man die Anzahl der für die Ausführung eines Programms erforderlichen Taktzyklen durch die Anzahl der Instruktionen." (Dipl.-Ing. Lipinski, et al., 2006).

**Datentyp**
„Ein Datentyp beschreibt eine Menge von Datenobjekten, die alle die gleiche Struktur haben und mit denen die gleichen Operationen ausgeführt werden können." (INF-Schule, 2021). Beispiele für Datentypen sind das „Integer" (int), der „String" oder der „Boolean" (boolean). Es existieren noch weitere Datentypen.

**Desktop**
Ein Desktop-PC ist ein Computer, der stationär installiert ist. Dem englischen Wortursprung gemäß stellt der Desktop oder Desktop-PC einen „Schreibtischrechner" dar, der für die Benutzung im privaten Bereich oder in Büros eingesetzt wird und eine verhältnismäßig geringe Rechenleistung hat (vgl. Nell, 2019).

**Integer**
Ein Integer-Datentyp stellt einen ganzzahligen Wert in der Java-Programmierung dar. In der Java-Syntax wird der Integer sowohl als Datentyp in Form von „int" verwendet, wie auch als Objekt der Klasse „Integer" (Mertens, 2020).

**IPS**
„Die Instruktionen pro Sekunde (kurz IPS, von englisch instructions per second), meist als Millionen Instruktionen pro Sekunde (MIPS, von engl. million instructions per second […]) angegeben, ist eine Maßeinheit für die Rechenleistung von Computern, dabei insbesondere die Leistungsfähigkeit der CPU." (Die Chemie-Schule, 2021).

**PC**
Ein PC (engl. für: Personal Computer) ist ein Computer, der für die Nutzung durch Privatpersonen vorgehsehen ist und verhältnismäßig wenig Rechenleistung aufweist (vgl. Faust, 2020).

**Rechenzentrum**
Ein Rechenzentrum ist ein Zentrum von „Rechen- und Speicherressourcen“ (1&1, 2020), das der Berechnung von großen Datenmengen dient. Dabei ist der Begriff „Rechenzentrum“ häufig synonym zu Großrechner. Dem Namen nach ist ein Rechenzentrum besonders leistungsfähig. Rechenzentren haben verschiedene Einsatzgebiete wie beispielsweise bei Internetprovidern, in der Forschung oder als Renderfarms in der Animationstechnik (vgl. 1&1, 2020).

**Server**
„Ein Server ist ein leistungsstarker Netzwerkrechner, der seine Ressourcen für andere Computer oder Programme bereitstellt. Diese greifen meist über das Netzwerk auf die Daten zu. Ein Server als Software ist ein Computerprogramm, welches mit dem Client (englisch für Kunde) kommuniziert und ihm Zugang zu bereitgestellten Daten verschafft.“ (Aschermann, 2016).

**String**
„Ein String ist eine Zeichenkette. Der Datentyp String hat als Wertebereich die Menge aller Zeichenketten. Strings werden in Anführungszeichen gesetzt; Beispiele für Strings sind "abba" […] sowie auch "". Die Länge eines Strings ist die Anzahl der Zeichen, aus denen er besteht. So hat beispielsweise der String "abba" die Länge 4. Der leere String "" hat die Länge 0, da er 0 Zeichen enthält. Die einzelnen Zeichen eines Strings der Länge n werden von 0 bis n-1 nummeriert.“ (Hochschule Flensburg, 2021).

**Taktfrequenz**
Die Taktfrequenz gibt die Anzahl der Zyklen an, die der Prozessor eines Computersystems pro Sekunde durchführen kann. Sie ist eine Einheit für die Geschwindigkeit des Prozessors und wird in GHz (Gigahertz) angegeben (vgl. Intel Corporation, 2021).

**Watt**
Die Einheit Watt ist eine Basiseinheit (SI-Einheit) für die Leistung. Sie setzt sich aus Arbeit pro Zeit zusammen und hat das Formelzeichen „W“ (vgl. Physik-Schule, 2021).

# Verweise- Literaturverzeichnis

[Online]

**1&1. 2020.** ionos.de. *Was ist ein Rechenzentrum.* [Online] IONOS, 15. September 2020. [Zitat vom: 05. Juni 2021.] https://www.ionos.de/digitalguide/server/knowhow/was-ist-ein-rechenzentrum/.

**Advanced Micro Devices. 2021.** amd.com. *AMD Ryzen 7 1700 Prozessor.* [Online] AMD, 2021. [Zitat vom: 30. Mai 2021.] https://www.amd.com/de/products/cpu/amd-ryzen-7-1700.

**Aschermann, Tim. 2016.** chip.de. *Was ist ein Server? Definition und Funktion einfach zusammengefasst.* [Online] CHIP, 19. April 2016. [Zitat vom: 05. Juni 2021.] https://praxistipps.chip.de/was-ist-ein-server-definition-und-funktion-einfach-zusammengefasst_12282.

*Average-Case Analyse.* **Prof. Dr. Albers, Susanne und Dr. Souza, Alexander. 2016.** Freiburg, George-Köhler-Allee : Universität Freiburg, 2016.

**Buchner, Florian. 2019.** florianbuchner.com. *Sortierverfahren: Countingsort.* [Online] 2018. Oktober 2019. https://florianbuchner.com/sortierverfahren-countingsort/.

**Bundesregierung. 2013.** Umwelt Bundesamt. *Was bedeutet "Energieeffizienz?".* [Online] Bundesrepublik Deutschland, 03. August 2013. [Zitat vom: 06. Juni 2021.] https://www.umweltbundesamt.de/service/uba-fragen/was-bedeutet-energieeffizienz.

**cheapenergy24. 2021.** cheapenergy24.de. *Wie hoch ist der Stromverbrauch eines PCs?* [Online] cheapenergy24, 2021. [Zitat vom: 30. Mai 2021.] https://www.cheapenergy24.de/news/stromverbrauch-pc/.

**Die Chemie-Schule. 2021.** chemie-schule.de. *Instruktionen pro Sekunde.* [Online] Die Chemie-Schule, 2021. [Zitat vom: 2021. Juni 05.] https://www.chemie-schule.de/KnowHow/MIPS_(Einheit).

**Dipl.-Ing. Lipinski, Klaus, et al. 2006.** itwissen.info. *CPI (cycles per instruction).* [Online] ITWissen.info, 27. September 2006. [Zitat vom: 05. Juni 2021.] https://www.itwissen.info/CPI-cycles-per-instruction.html.

**Faust, Daniel. 2020.** biteno.com. *Was ist ein Personal Computer (PC)?* [Online] Biteno GmbH, 05. August 2020. [Zitat vom: 05. Juni 2021.] https://www.biteno.com/was-ist-ein-personal-computer-pc/.

**Hochschule Flensburg. 2021.** if.hs-flensburg.de. *Datentyp String.* [Online] 30. Mai 2021. https://www.inf.hs-flensburg.de/lang/prog/string.htm.

**INF-Schule. 2021.** inf-schule.de. *Fachkonzept-Datentyp.* [Online] 30. Mai 2021. https://www.inf-schule.de/programmierung/imperativeprogrammierung/konzepteimp/datentypen/konzept_datentyp.

**Intel Corporation. 2021.** intel.de. *Was ist Taktfrequenz.* [Online] intel, 2021. [Zitat vom: 05. Juni 2021.] https://www.intel.de/content/www/de/de/gaming/resources/cpu-clock-speed.html.

**Mertens, Robert. 2020.** lernenprogrammieren.com. *Der Datentyp Integer.* [Online] Lernenprogrammieren, 13. September 2020. [Zitat vom: 30. Mai 2021.] https://www.inf-schule.de/programmierung/imperativeprogrammierung/konzepteimp/datentypen/konzept_datentyp.

**Microsoft Corporation. 2021.** microsoft.com. *Long data type (Visual Basic).* [Online] Mai 29, 2021. https://docs.microsoft.com/en-us/dotnet/visual-basic/language-reference/data-types/long-data-type.

**Nell, Maximilian. 2019.** Chip. *Was ist ein Desktop PC: Einfach erklärt - CHIP.* [Online] CHIP, 13. März 2019. [Zitat vom: 05. Juni 2021.] https://praxistipps.chip.de/was-ist-ein-desktop-pc-einfach-erklaert_108897.

**Oracle. 2021.** oracle.com. *Random.* [Online] 2021. [Zitat vom: 29. Mai 2021.] https://docs.oracle.com/javase/8/docs/api/java/util/Random.html.

**Physik-Schule. 2021.** physik.cosmos-indirekt.de. *Watt (Einheit).* [Online] 2021. [Zitat vom: 05. Juni 2021.] https://physik.cosmos-indirekt.de/Physik-Schule/Watt_(Einheit).

**Rivoire, Suzanne, et al. 2021.** *JouleSort: A Balanced Energy-Efficiency Benchmark.* s.l. : Standford University, 2021.

**Schneider, Michael. 2001.** Sortieralgorithmen. *Seminar "Algorithmen und Zufälligkeit" (Prof. Dr. G. Kersting).* Frankfurt am Main, Hessen, Deutschland : Johann-Wolfgang-Goethe Universität, 2001.

**Studyflix. 2021.** studyflix.com. *Countingsort.* [Online] 14. Mai 2021. https://studyflix.de/informatik/counting-sort-1407.

**—. 2021.** studyflix.de. *Quicksort.* [Online] studiflix, 2021. [Zitat vom: 14. Mai 2021.] https://studyflix.de/informatik/quicksort-1322.

**—. 2021.** studyflix.de. *Mergesort.* [Online] Studyflix, 2021. [Zitat vom: 2021. Mai 14.] https://studyflix.de/informatik/mergesort-1324.

**—. 2021.** studyflix.de. *Arrays.* [Online] Studyflix, 2021. [Zitat vom: 2021. Juni 05.] https://studyflix.de/informatik/arrays-802.

# Abbildungsverzeichnis

# Anhang



## Interessante Abbildungen

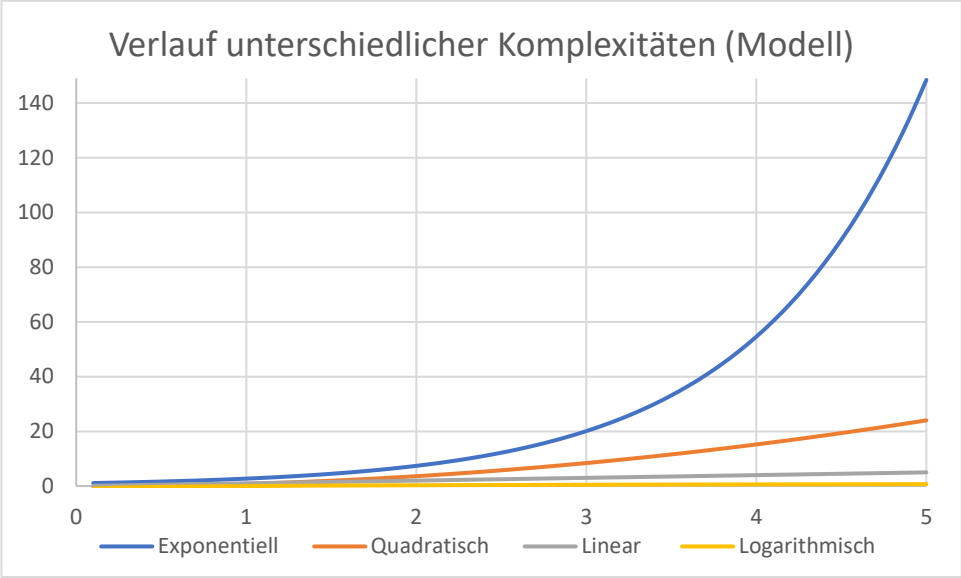


*Abbildung 8: Darstellung unterschiedlicher Komplexitäts-Verläufe anhand eines Modells. X-Achse: Objekte, Y-Achse: fiktiver Wert*

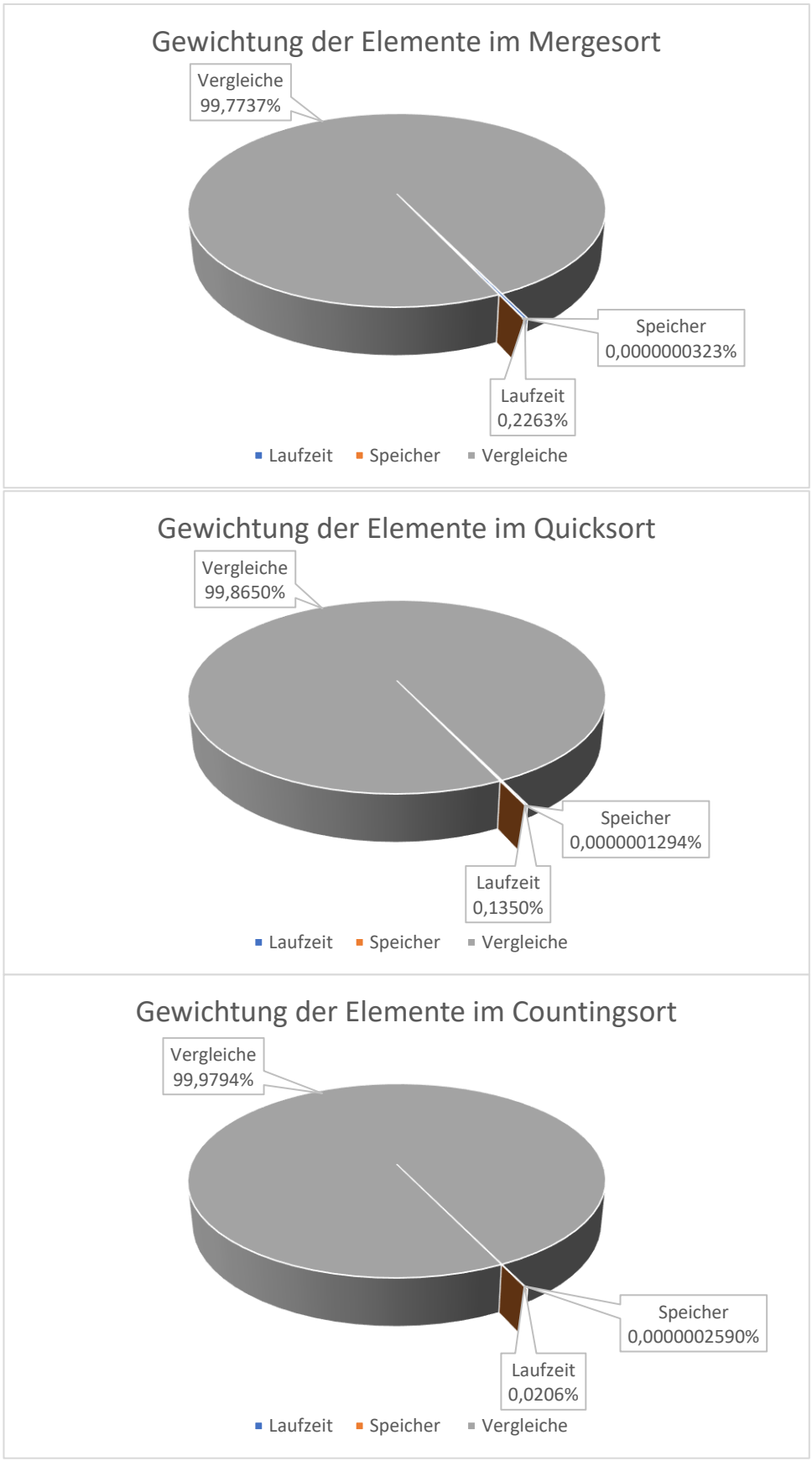


*Abbildung 9: Darstellung der Gewichtungen im Index der einzelnen Sortierverfahren (real)*

## Wertetabellen

**Index-Auswertungen (0-16.000.000)**

| Objekte | Quick | Merge | Count |
|---|---|---|---|
| 0 | | | |
| 2.000 | 0,00939702 | 0,00939702 | 469,947362 |
| 4.000 | 0,02067345 | 0,02067345 | 469,949722 |
| 6.000 | 0,03382929 | 0,03382929 | 469,950662 |
| 8.000 | 0,04510572 | 0,04520722 | 469,951196 |
| 10.000 | 0,06118216 | 0,06118216 | 469,952339 |
| 12.000 | 0,07339829 | 0,07339829 | 469,953481 |
| 14.000 | 0,08561442 | 0,08571592 | 469,954827 |
| 16.000 | 0,09783055 | 0,09793205 | 469,955462 |
| 18.000 | 0,11850401 | 0,11860551 | 469,956199 |
| 20.000 | 0,13176134 | 0,13186284 | 469,956631 |
| 22.000 | 0,14491717 | 0,14501867 | 469,956556 |
| 24.000 | 0,15807301 | 0,15817451 | 469,957292 |
| 26.000 | 0,17122884 | 0,17143184 | 469,959349 |
| 28.000 | 0,18448618 | 0,18458768 | 469,958969 |
| 30.000 | 0,19764201 | 0,19784501 | 469,964273 |
| 32.000 | 0,21069634 | 0,21100084 | 469,964908 |
| 34.000 | 0,23992862 | 0,24013162 | 469,965848 |
| 36.000 | 0,25402416 | 0,25432866 | 469,967092 |
| 38.000 | 0,26811969 | 0,26842419 | 469,967626 |
| 40.000 | 0,28221523 | 0,28251973 | 469,968667 |
| 42.000 | 0,29641227 | 0,29671676 | 469,969099 |
| 44.000 | 0,3105078 | 0,3108123 | 469,969734 |
| 46.000 | 0,32460334 | 0,32500934 | 469,970573 |
| 48.000 | 0,33869887 | 0,33910487 | 469,971817 |
| 50.000 | 0,35289591 | 0,35320041 | 469,974482 |
| 52.000 | 0,36699145 | 0,36739745 | 469,986688 |
| 54.000 | 0,38108698 | 0,38149298 | 469,979203 |
| 56.000 | 0,39518252 | 0,39558852 | 469,975982 |
| 58.000 | 0,40927806 | 0,40978556 | 469,976921 |
| 60.000 | 0,42347509 | 0,42388109 | 469,97776 |
| 62.000 | 0,43757063 | 0,43797663 | 469,978699 |
| 64.000 | 0,45166617 | 0,45217367 | 469,97974 |
| 66.000 | 0,49677188 | 0,49727938 | 469,979868 |
| 68.000 | 0,51190862 | 0,51241612 | 469,981417 |
| 70.000 | 0,52694386 | 0,52734986 | 469,981849 |
| 72.000 | 0,5419791 | 0,5424866 | 469,983398 |
| 74.000 | 0,55701434 | 0,55762334 | 469,98454 |
| 76.000 | 0,57215108 | 0,57265858 | 469,985176 |
| 78.000 | 0,58718632 | 0,58769381 | 469,986318 |
| 80.000 | 0,60222155 | 0,60283055 | 469,986751 |
| 82.000 | 0,61725679 | 0,61786579 | 469,988299 |
| 84.000 | 0,63229203 | 0,63290103 | 469,988935 |

| | | | |
|---|---|---|---|
| 86.000 | 0,64732727 | 0,64793627 | 469,989773 |
| 88.000 | 0,66246401 | 0,66307301 | 469,991524 |
| 90.000 | 0,67749925 | 0,67810825 | 469,991449 |
| 92.000 | 0,69253449 | 0,69324499 | 469,991983 |
| 94.000 | 0,70756973 | 0,70828023 | 469,994141 |
| 96.000 | 0,72270647 | 0,72331546 | 469,993558 |
| 98.000 | 0,7377417 | 0,7384522 | 469,995918 |
| 100.000 | 0,75277694 | 0,75348744 | 469,996452 |
| 102.000 | 0,76781218 | 0,76862418 | 469,998204 |
| 104.000 | 0,78294892 | 0,78365942 | 469,999144 |
| 106.000 | 0,79798416 | 0,79869466 | 469,999474 |
| 108.000 | 0,8130194 | 0,8137299 | 470,000211 |
| 110.000 | 0,82805464 | 0,82886664 | 470,001557 |
| 112.000 | 0,84308988 | 0,84390188 | 470,005947 |
| 114.000 | 0,85812512 | 0,85893711 | 470,001914 |
| 116.000 | 0,87326185 | 0,87407385 | 470,003665 |
| 118.000 | 0,88829709 | 0,88910909 | 470,003691 |
| 120.000 | 0,90333233 | 0,90414433 | 470,00453 |
| 122.000 | 0,91836757 | 0,91928107 | 470,004353 |
| 124.000 | 0,93350431 | 0,93431631 | 470,004785 |
| 126.000 | 0,94853955 | 0,94945305 | 470,008262 |
| 128.000 | 0,96357479 | 0,96448829 | 470,010521 |
| 130.000 | 0,97861003 | 0,97952352 | 470,010852 |
| 132.000 | 1,05566563 | 1,05657912 | 470,008036 |
| 134.000 | 1,07174207 | 1,07265557 | 470,007555 |
| 136.000 | 1,08771701 | 1,08863051 | 470,008901 |
| 138.000 | 1,10369195 | 1,10470695 | 470,009739 |
| 140.000 | 1,11966689 | 1,12068189 | 470,010374 |
| 142.000 | 1,13574333 | 1,13665683 | 470,011415 |
| 144.000 | 1,15171827 | 1,15273327 | 470,012355 |
| 146.000 | 1,16769321 | 1,16870821 | 470,013802 |
| 148.000 | 1,18366816 | 1,18468315 | 470,014539 |
| 150.000 | 1,1996431 | 1,2007596 | 470,015682 |
| 152.000 | 1,21571954 | 1,21673454 | 470,016317 |
| 154.000 | 1,23169448 | 1,23281098 | 470,01746 |
| 156.000 | 1,24766942 | 1,24878592 | 470,018399 |
| 158.000 | 1,26374586 | 1,26476086 | 470,018831 |
| 160.000 | 1,2797208 | 1,2808373 | 470,020076 |
| 162.000 | 1,29569574 | 1,29681224 | 470,021421 |
| 164.000 | 1,31167069 | 1,31278718 | 470,022057 |
| 166.000 | 1,32774713 | 1,32886363 | 470,022895 |
| 168.000 | 1,34372207 | 1,34483857 | 470,02353 |
| 170.000 | 1,35969701 | 1,36091501 | 470,025079 |
| 172.000 | 1,37567195 | 1,37688995 | 470,026018 |
| 174.000 | 1,39174839 | 1,39296639 | 470,028887 |
| 176.000 | 1,40762183 | 1,40894133 | 470,029928 |
| 178.000 | 1,42369827 | 1,42491627 | 470,031984 |

| | | | |
|---|---|---|---|
| 180.000 | 1,43967322 | 1,44099271 | 470,031807 |
| 182.000 | 1,45564816 | 1,45696766 | 470,030006 |
| 184.000 | 1,4717246 | 1,4729426 | 470,037138 |
| 186.000 | 1,48769954 | 1,48901904 | 470,033205 |
| 188.000 | 1,50377598 | 1,50499398 | 470,033029 |
| 190.000 | 1,51975092 | 1,52096892 | 470,033867 |
| 192.000 | 1,53562436 | 1,53704536 | 470,035212 |
| 194.000 | 1,5517008 | 1,5530203 | 470,035442 |
| 196.000 | 1,56767575 | 1,56899524 | 470,036584 |
| 198.000 | 1,58365069 | 1,58507169 | 470,037626 |
| 2.000.000 | 18,8202356 | 830,797297 | |
| 4.000.000 | 39,5201805 | 1663,47075 | |
| 6.000.000 | 62,1010528 | 2498,02361 | |
| 8.000.000 | 82,8032307 | 3330,69706 | |
| 10.000.000 | 108,204428 | 4168,06903 | |
| 12.000.000 | 129,846816 | 5001,68218 | |
| 14.000.000 | 151,49032 | 5835,29534 | |
| 16.000.000 | 173,131795 | 6668,90849 | |

**Laufzeit- und Speicher-Messwerte (0-16.000.000)**

| Obj. | Quick Laufzeit | Quick Speicher | Merge Laufzeit | Merge Speicher | Count Laufzeit | Count Speicher |
|---|---|---|---|---|---|---|
| 0 | | | | | | 1935484928 |
| 2.000 | 0 | 32000 | 0 | 8032 | 938 | 1935740928 |
| 4.000 | 0 | 64000 | 0 | 16032 | 952 | 1935996928 |
| 6.000 | 0 | 96000 | 0 | 24032 | 952 | 1936252928 |
| 8.000 | 0 | 128000 | 1 | 32032 | 948 | 1936508928 |
| 10.000 | 1 | 160000 | 1 | 40032 | 950 | 1936764928 |
| 12.000 | 1 | 192000 | 1 | 48032 | 952 | 1937020928 |
| 14.000 | 1 | 224000 | 2 | 56032 | 956 | 1937276928 |
| 16.000 | 1 | 256000 | 2 | 64032 | 953 | 1937532928 |
| 18.000 | 1 | 288000 | 2 | 72032 | 951 | 1937788928 |
| 20.000 | 2 | 320000 | 3 | 80032 | 946 | 1938044928 |
| 22.000 | 2 | 352000 | 3 | 88032 | 936 | 1938300928 |
| 24.000 | 2 | 384000 | 3 | 96032 | 934 | 1938556928 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 26.000 | 2 | 416000 | 4 | 104032 | 945 | 1938556928 |
| 28.000 | 3 | 448000 | 4 | 112032 | 932 | 1938812928 |
| 30.000 | 3 | 480000 | 5 | 120032 | 975 | 1939068928 |
| 32.000 | 2 | 512000 | 5 | 128032 | 972 | 1939324928 |
| 34.000 | 3 | 544000 | 5 | 136032 | 972 | 1939580928 |
| 36.000 | 3 | 576000 | 6 | 144032 | 975 | 1939836928 |
| 38.000 | 3 | 608000 | 6 | 152032 | 971 | 1940092928 |
| 40.000 | 3 | 640000 | 6 | 160032 | 972 | 1940348928 |
| 42.000 | 4 | 672000 | 7 | 168032 | 967 | 1940604928 |
| 44.000 | 4 | 704000 | 7 | 176032 | 964 | 1940860928 |
| 46.000 | 4 | 736000 | 8 | 184032 | 963 | 1941116928 |
| 48.000 | 4 | 768000 | 8 | 192032 | 966 | 1941372928 |
| 50.000 | 5 | 800000 | 8 | 200032 | 983 | 1941628928 |
| 52.000 | 5 | 832000 | 9 | 208032 | 1094 | 1941884928 |
| 54.000 | 5 | 864000 | 9 | 216032 | 1011 | 1942140928 |
| 56.000 | 5 | 896000 | 9 | 224032 | 970 | 1942396928 |
| 58.000 | 5 | 928000 | 10 | 232032 | 970 | 1942652928 |
| 60.000 | 6 | 960000 | 10 | 240032 | 969 | 1942908928 |
| 62.000 | 6 | 992000 | 10 | 248032 | 969 | 1943164928 |
| 64.000 | 6 | 1024000 | 11 | 256032 | 970 | 1943420928 |
| 66.000 | 6 | 1056000 | 11 | 264032 | 962 | 1943676928 |
| 68.000 | 7 | 1088000 | 12 | 272032 | 968 | 1943932928 |
| 70.000 | 7 | 1120000 | 11 | 280032 | 963 | 1944188928 |
| 72.000 | 7 | 1152000 | 12 | 288032 | 969 | 1944444928 |
| 74.000 | 7 | 1184000 | 13 | 296032 | 971 | 1944700928 |
| 76.000 | 8 | 1216000 | 13 | 304032 | 968 | 1944956928 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 78.000 | 8 | 1248000 | 13 | 312032 | 970 | 1945212928 |
| 80.000 | 8 | 1280000 | 14 | 320032 | 965 | 1945468928 |
| 82.000 | 8 | 1312000 | 14 | 328032 | 971 | 1945724928 |
| 84.000 | 8 | 1344000 | 14 | 336032 | 968 | 1945980928 |
| 86.000 | 8 | 1376000 | 14 | 344032 | 967 | 1946236928 |
| 88.000 | 9 | 1408000 | 15 | 352032 | 975 | 1946492928 |
| 90.000 | 9 | 1440000 | 15 | 360032 | 965 | 1946748928 |
| 92.000 | 9 | 1472000 | 16 | 368032 | 961 | 1947004928 |
| 94.000 | 9 | 1504000 | 16 | 376032 | 973 | 1947260928 |
| 96.000 | 10 | 1536000 | 16 | 384032 | 958 | 1947516928 |
| 98.000 | 10 | 1568000 | 17 | 392032 | 972 | 1947772928 |
| 100.000 | 10 | 1600000 | 17 | 400032 | 968 | 1948028928 |
| 102.000 | 10 | 1632000 | 18 | 408032 | 976 | 1948284928 |
| 104.000 | 11 | 1664000 | 18 | 416032 | 976 | 1948540928 |
| 106.000 | 11 | 1696000 | 18 | 424032 | 970 | 1948796928 |
| 108.000 | 11 | 1728000 | 18 | 432032 | 968 | 1949052928 |
| 110.000 | 11 | 1760000 | 19 | 440032 | 972 | 1949308928 |
| 112.000 | 11 | 1792000 | 19 | 448032 | 1006 | 1949564928 |
| 114.000 | 11 | 1824000 | 19 | 456032 | 957 | 1949820928 |
| 116.000 | 12 | 1856000 | 20 | 464032 | 965 | 1950076928 |
| 118.000 | 12 | 1888000 | 20 | 472032 | 956 | 1950332928 |
| 120.000 | 12 | 1920000 | 20 | 480032 | 955 | 1950588928 |
| 122.000 | 12 | 1952000 | 21 | 488032 | 944 | 1950844928 |
| 124.000 | 13 | 1984000 | 21 | 496032 | 939 | 1951100928 |
| 126.000 | 13 | 2016000 | 22 | 504032 | 964 | 1951356928 |
| 128.000 | 13 | 2048000 | 22 | 512032 | 977 | 1951612928 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 130.000 | 13 | 2080000 | 22 | 520032 | 971 | 1951868928 |
| 132.000 | 13 | 2112000 | 22 | 528032 | 934 | 1952124928 |
| 134.000 | 14 | 2144000 | 23 | 536032 | 920 | 1952380928 |
| 136.000 | 14 | 2176000 | 23 | 544032 | 924 | 1952636928 |
| 138.000 | 14 | 2208000 | 24 | 552032 | 923 | 1952892928 |
| 140.000 | 14 | 2240000 | 24 | 560032 | 920 | 1953148928 |
| 142.000 | 15 | 2272000 | 24 | 568032 | 921 | 1953404928 |
| 144.000 | 15 | 2304000 | 25 | 576032 | 921 | 1953660928 |
| 146.000 | 15 | 2336000 | 25 | 584032 | 926 | 1953916928 |
| 148.000 | 15 | 2368000 | 25 | 592032 | 924 | 1954172928 |
| 150.000 | 15 | 2400000 | 26 | 600032 | 926 | 1954428928 |
| 152.000 | 16 | 2432000 | 26 | 608032 | 923 | 1954684928 |
| 154.000 | 16 | 2464000 | 27 | 616032 | 925 | 1954940928 |
| 156.000 | 16 | 2496000 | 27 | 624032 | 925 | 1955196928 |
| 158.000 | 17 | 2528000 | 27 | 632032 | 920 | 1955452928 |
| 160.000 | 17 | 2560000 | 28 | 640032 | 923 | 1955708928 |
| 162.000 | 17 | 2592000 | 28 | 648032 | 927 | 1955964928 |
| 164.000 | 17 | 2624000 | 28 | 656032 | 924 | 1956220928 |
| 166.000 | 18 | 2656000 | 29 | 664032 | 923 | 1956476928 |
| 168.000 | 18 | 2688000 | 29 | 672032 | 920 | 1956732928 |
| 170.000 | 18 | 2720000 | 30 | 680032 | 926 | 1956988928 |
| 172.000 | 18 | 2752000 | 30 | 688032 | 926 | 1957244928 |
| 174.000 | 19 | 2784000 | 31 | 696032 | 945 | 1957500928 |
| 176.000 | 18 | 2816000 | 31 | 704032 | 946 | 1957756928 |
| 178.000 | 19 | 2848000 | 31 | 712032 | 957 | 1958012928 |
| 180.000 | 19 | 2880000 | 32 | 720032 | 946 | 1958268928 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 182.000 | 19 | 2912000 | 32 | 728032 | 919 | 1958524928 |
| 184.000 | 20 | 2944000 | 32 | 736032 | 980 | 1958780928 |
| 186.000 | 20 | 2976000 | 33 | 744032 | 932 | 1959036928 |
| 188.000 | 21 | 3008000 | 33 | 752032 | 921 | 1959292928 |
| 190.000 | 21 | 3040000 | 33 | 760032 | 920 | 1959548928 |
| 192.000 | 20 | 3072000 | 34 | 768032 | 924 | 1959804928 |
| 194.000 | 21 | 3104000 | 34 | 776032 | 917 | 1960060928 |
| 196.000 | 21 | 3136000 | 34 | 784032 | 919 | 1960316928 |
| 198.000 | 21 | 3168000 | 35 | 792032 | 920 | 1960572928 |
| 2.000.000 | 258 | 32000000 | 530 | 8000032 | | |
| 4.000.000 | 519 | 64000000 | 1112 | 16000032 | | |
| 6.000.000 | 795 | 96000000 | 1684 | 24000032 | | |
| 8.000.000 | 1078 | 128000000 | 2252 | 32000032 | | |
| 10.000.000 | 1366 | 160000000 | 2872 | 40000032 | | |
| 12.000.000 | 1654 | 192000000 | 3480 | 48000032 | | |
| 14.000.000 | 1953 | 224000000 | 4117 | 56000032 | | |
| 16.000.000 | 2232 | 256000000 | 4767 | 64000032 | | |

Die Speicher-Angaben sind in Bytes, die Angaben zur Laufzeit in Millisekunden.

**Vergleichs-Messwerte (0-16.000.000)**

| Objekte | Quick_Vergleiche | Merge_Vergleiche | Count_Vergleiche |
|---|---|---|---|
| 0 | 0 | 0 | 1.000.000.000 |
| 2000 | 20.000 | 20.000 | 1.000.002.000 |
| 4000 | 44.000 | 44.000 | 1.000.004.000 |
| 6000 | 72.000 | 72.000 | 1.000.006.000 |
| 8000 | 96.000 | 96.000 | 1.000.008.000 |
| 10000 | 130.000 | 130.000 | 1.000.010.000 |
| 12000 | 156.000 | 156.000 | 1.000.012.000 |
| 14000 | 182.000 | 182.000 | 1.000.014.000 |
| 16000 | 208.000 | 208.000 | 1.000.016.000 |
| 18000 | 252.000 | 252.000 | 1.000.018.000 |
| 20000 | 280.000 | 280.000 | 1.000.020.000 |
| 22000 | 308.000 | 308.000 | 1.000.022.000 |
| 24000 | 336.000 | 336.000 | 1.000.024.000 |
| 26000 | 364.000 | 364.000 | 1.000.026.000 |
| 28000 | 392.000 | 392.000 | 1.000.028.000 |

| | | | |
|---|---|---|---|
| 30000 | 420.000 | 420.000 | 1.000.030.000 |
| 32000 | 448.000 | 448.000 | 1.000.032.000 |
| 34000 | 510.000 | 510.000 | 1.000.034.000 |
| 36000 | 540.000 | 540.000 | 1.000.036.000 |
| 38000 | 570.000 | 570.000 | 1.000.038.000 |
| 40000 | 600.000 | 600.000 | 1.000.040.000 |
| 42000 | 630.000 | 630.000 | 1.000.042.000 |
| 44000 | 660.000 | 660.000 | 1.000.044.000 |
| 46000 | 690.000 | 690.000 | 1.000.046.000 |
| 48000 | 720.000 | 720.000 | 1.000.048.000 |
| 50000 | 750.000 | 750.000 | 1.000.050.000 |
| 52000 | 780.000 | 780.000 | 1.000.052.000 |
| 54000 | 810.000 | 810.000 | 1.000.054.000 |
| 56000 | 840.000 | 840.000 | 1.000.056.000 |
| 58000 | 870.000 | 870.000 | 1.000.058.000 |
| 60000 | 900.000 | 900.000 | 1.000.060.000 |
| 62000 | 930.000 | 930.000 | 1.000.062.000 |
| 64000 | 960.000 | 960.000 | 1.000.064.000 |
| 66000 | 1.056.000 | 1.056.000 | 1.000.066.000 |
| 68000 | 1.088.000 | 1.088.000 | 1.000.068.000 |
| 70000 | 1.120.000 | 1.120.000 | 1.000.070.000 |
| 72000 | 1.152.000 | 1.152.000 | 1.000.072.000 |
| 74000 | 1.184.000 | 1.184.000 | 1.000.074.000 |
| 76000 | 1.216.000 | 1.216.000 | 1.000.076.000 |
| 78000 | 1.248.000 | 1.248.000 | 1.000.078.000 |
| 80000 | 1.280.000 | 1.280.000 | 1.000.080.000 |
| 82000 | 1.312.000 | 1.312.000 | 1.000.082.000 |
| 84000 | 1.344.000 | 1.344.000 | 1.000.084.000 |
| 86000 | 1.376.000 | 1.376.000 | 1.000.086.000 |
| 88000 | 1.408.000 | 1.408.000 | 1.000.088.000 |
| 90000 | 1.440.000 | 1.440.000 | 1.000.090.000 |
| 92000 | 1.472.000 | 1.472.000 | 1.000.092.000 |
| 94000 | 1.504.000 | 1.504.000 | 1.000.094.000 |
| 96000 | 1.536.000 | 1.536.000 | 1.000.096.000 |
| 98000 | 1.568.000 | 1.568.000 | 1.000.098.000 |
| 100000 | 1.600.000 | 1.600.000 | 1.000.100.000 |
| 102000 | 1.632.000 | 1.632.000 | 1.000.102.000 |
| 104000 | 1.664.000 | 1.664.000 | 1.000.104.000 |
| 106000 | 1.696.000 | 1.696.000 | 1.000.106.000 |
| 108000 | 1.728.000 | 1.728.000 | 1.000.108.000 |
| 110000 | 1.760.000 | 1.760.000 | 1.000.110.000 |
| 112000 | 1.792.000 | 1.792.000 | 1.000.112.000 |
| 114000 | 1.824.000 | 1.824.000 | 1.000.114.000 |
| 116000 | 1.856.000 | 1.856.000 | 1.000.116.000 |
| 118000 | 1.888.000 | 1.888.000 | 1.000.118.000 |
| 120000 | 1.920.000 | 1.920.000 | 1.000.120.000 |
| 122000 | 1.952.000 | 1.952.000 | 1.000.122.000 |

| | | | |
|---|---|---|---|
| 124000 | 1.984.000 | 1.984.000 | 1.000.124.000 |
| 126000 | 2.016.000 | 2.016.000 | 1.000.126.000 |
| 128000 | 2.048.000 | 2.048.000 | 1.000.128.000 |
| 130000 | 2.080.000 | 2.080.000 | 1.000.130.000 |
| 132000 | 2.244.000 | 2.244.000 | 1.000.132.000 |
| 134000 | 2.278.000 | 2.278.000 | 1.000.134.000 |
| 136000 | 2.312.000 | 2.312.000 | 1.000.136.000 |
| 138000 | 2.346.000 | 2.346.000 | 1.000.138.000 |
| 140000 | 2.380.000 | 2.380.000 | 1.000.140.000 |
| 142000 | 2.414.000 | 2.414.000 | 1.000.142.000 |
| 144000 | 2.448.000 | 2.448.000 | 1.000.144.000 |
| 146000 | 2.482.000 | 2.482.000 | 1.000.146.000 |
| 148000 | 2.516.000 | 2.516.000 | 1.000.148.000 |
| 150000 | 2.550.000 | 2.550.000 | 1.000.150.000 |
| 152000 | 2.584.000 | 2.584.000 | 1.000.152.000 |
| 154000 | 2.618.000 | 2.618.000 | 1.000.154.000 |
| 156000 | 2.652.000 | 2.652.000 | 1.000.156.000 |
| 158000 | 2.686.000 | 2.686.000 | 1.000.158.000 |
| 160000 | 2.720.000 | 2.720.000 | 1.000.160.000 |
| 162000 | 2.754.000 | 2.754.000 | 1.000.162.000 |
| 164000 | 2.788.000 | 2.788.000 | 1.000.164.000 |
| 166000 | 2.822.000 | 2.822.000 | 1.000.166.000 |
| 168000 | 2.856.000 | 2.856.000 | 1.000.168.000 |
| 170000 | 2.890.000 | 2.890.000 | 1.000.170.000 |
| 172000 | 2.924.000 | 2.924.000 | 1.000.172.000 |
| 174000 | 2.958.000 | 2.958.000 | 1.000.174.000 |
| 176000 | 2.992.000 | 2.992.000 | 1.000.176.000 |
| 178000 | 3.026.000 | 3.026.000 | 1.000.178.000 |
| 180000 | 3.060.000 | 3.060.000 | 1.000.180.000 |
| 182000 | 3.094.000 | 3.094.000 | 1.000.182.000 |
| 184000 | 3.128.000 | 3.128.000 | 1.000.184.000 |
| 186000 | 3.162.000 | 3.162.000 | 1.000.186.000 |
| 188000 | 3.196.000 | 3.196.000 | 1.000.188.000 |
| 190000 | 3.230.000 | 3.230.000 | 1.000.190.000 |
| 192000 | 3.264.000 | 3.264.000 | 1.000.192.000 |
| 194000 | 3.298.000 | 3.298.000 | 1.000.194.000 |
| 196000 | 3.332.000 | 3.332.000 | 1.000.196.000 |
| 198000 | 3.366.000 | 3.366.000 | 1.000.198.000 |
| 2.000.000 | 40000000 | 40000000 | |
| 4.000.000 | 84000000 | 84000000 | |
| 6.000.000 | 132000000 | 132000000 | |
| 8.000.000 | 176000000 | 176000000 | |
| 10.000.000 | 230000000 | 230000000 | |
| 12.000.000 | 276000000 | 276000000 | |
| 14.000.000 | 322000000 | 322000000 | |
| 16.000.000 | 368000000 | 368000000 | |

# Erklärung zur selbstständigen Verfassung der Arbeit

Ich versichere, dass ich die vorliegende Arbeit selbstständig verfasst, keine anderen als die angegebenen Hilfsmittel benutzt und die Stellen der Facharbeit, die im Wortlaut oder im wesentlichen Inhalt aus anderen Werken entnommen wurden, mit genauer Quellenangabe kenntlich gemacht habe. Verwendete Informationen aus dem Internet können der Lehrkraft auf Verlangen zur Verfügung gestellt werden.

_____ _____

Ort, Datum Unterschrift