




15. JANUAR 2025

SORTIERALGORITHMEN

EINFÜHRUNG IN DIE SORTIERALGORITHMEN

ANTHONY TIMMEL
ENTWICKLUNGSGESELLSCHAFT NIEDERSCHLESISCHE OBERLAUSITZ MBH
Elisabethstraße 40, 02826 Görlitz



Inhaltsverzeichnis

Begriffserklärung.....	3
Sortiervverfahren.....	3
Algorithmus.....	3
Regeln für den Algorithmus	3
Rekursion	3
Cases	3
Average-Case.....	3
Best-Case.....	3
Worst-Case.....	4
Tupel.....	4
Array	4
Datensatz.....	4
Lexikographische Ordnung	4
Zeichenketten.....	4
Node.....	4
Buckets.....	4
Sortieralgorithmen	4
Vergleichbares Sortieren	4
Binary Tree Sort (höhen-balanciert).....	5
Bubblesort	5
Combsort	5
Gnomesort	5
Heapsort	5
Insertionsort	6
Introsort.....	6
Merge Insertion	6
Mergesort	6
Quicksort.....	6
Samplesort	7
Selectionsort	7
Shakesort	7

Shellsort	7
Smoothsort.....	7
Stoogesort	7
Swap-Sort.....	7
Timsort	8
Bogosort	8
Slowsort	8
Nicht-vergleichbares Sortieren	8
Bucketsort	8
Coutingsort.....	8
Radixsort	8
Flashsort	8
Wann sollte man welchen Algorithmus verwenden?.....	9
Sortieralgorithmen im Detail	9
Quellenangabe	9

Begriffserklärung

Sortierv Verfahren

Unter dem Sortierv Verfahren versteht man in der Informatik einen [Algorithmus](#), der dazu dient, eine Liste (auch [Array](#) oder [Tupel](#) genannt) zu ordnen. Dazu wird auf das Element in der Liste zurückgegriffen. Dabei wird vorausgesetzt, dass es eine strenge schwache Ordnung, wie die [lexikographische Ordnung](#) von [Zeichenketten](#) oder die numerische Ordnung von Zahlen definiert ist.

Algorithmus

Ein Algorithmus ist eine eindeutige Handlungsvorschrift zur Lösung eines Problems oder eine Klasse von Problemen. Algorithmen bestehen aus endlich vielen einzelnen wohldefinierten Einzelschritten. Dadurch können sie zur Ausführung in einem Computerprogramm implementiert werden, als auch in menschlicher Sprache formuliert werden. Einen Algorithmus kann man als solches Bezeichnen, wenn er die folgenden Regeln einhält.

Regeln für den Algorithmus

- Der Algorithmus muss bei denselben Eingabewerten dasselbe Ergebnis hervorbringen.
- Jedes Verfahren muss in einem endlichen Text beschreibbar sein.
- Jeder Schritt des Verfahrens muss tatsächlich ausführbar sein.
- Das Verfahren darf nur eine bestimmbare Anzahl an Schritten benötigen
- Die nächste anzuwendende Regel im Verfahren ist zu jedem Zeitpunkt eindeutig.

Rekursion

Die Rekursion ist ein unendlicher Vorgang, der sich selbst als Teil enthält, oder mit sich selbst definierbar ist.

Cases

Cases sind verschiedene Möglichkeiten wie etwas eintreten kann. In diesem Kontext bezieht sich ein Case auf die Schwierigkeit eine endliche Menge an Elementen zu sortieren.

Average-Case

Die normalste und / oder häufigste Variante die der Algorithmus benötigt.

Best-Case

Die beste Variante die der Algorithmus benötigt.

Worst-Case

Die schlimmste Variante die der Algorithmus benötigt.

Tupel

Tupel sind in dem mathematischen Aspekt mathematische Objekte. Sie agieren als eine Liste endlicher vieler, nicht notwendigerweise unterschiedlicher Objekte. Dabei spielt, im Gegensatz als bei Mengen, die Reihenfolge der Objekte innerhalb des Tupels eine Rolle. Die Tupel findet man in der Mathematik bei der Wiedergabe als Koordinaten von Punkten oder als Vektoren. In der Informatik werden sie auch als Synonym für unveränderliche [Datensätze](#) verwendet.

Array

Ein Array ist in der Informatik eine Datenstruktur ähnlich die das [Tupel](#).

Datensatz

Ein Datensatz ist eine Gruppe von inhaltlich zusammenhängenden Attributen

Lexikographische Ordnung

Die lexikographische Ordnung ist eine Methode, um aus einer linearen Ordnung für einfache Objekte, wie alphabetisch angeordnete Buchstaben, eine lineare Ordnung für aus Buchstaben zusammengesetzte Wörter, zu erhalten.

Zeichenketten

Zeichenketten, Zeichenfolgen oder auch Zeichenreihen sind eine endliche Folge von Zeichen wie z.B. Buchstaben, Ziffern, Sonderzeichen und/oder Steuerzeichen aus einem vordefinierten Zeichensatz. Ein Zeichensatz ist eine Form des [Datensatzes](#).

Node

Eine Node ist in der binären Baumstruktur ein Knotenpunkt. Dieser beinhaltet zu einem das tatsächliche Element, welches zu sortieren ist und ebenfalls zusätzliche Parameter für den Algorithmus das Element zu sortieren.

Buckets

Ein Bucket ist Behälter für Daten, genauso wie das [Array](#).

Sortieralgorithmen

Vergleichbares Sortieren

Das vergleichbare Sortieren basiert auf dem paarweisen Vergleichen von Elementen in einer Liste.

Binary Tree Sort (höhen-balanciert)

In dem Binary Tree Sort wird das zu sortierende Element in einer [Node](#) eingebunden. Durch das lineare Vergleichen des zu sortierenden Elementes, in der Node, kann der Algorithmus festlegen in welcher Höhe das Element platziert werden muss. Wenn sich nun der Algorithmus sich das erste Element greift, überprüft er alle Abzweigungen. Jede Abzweigung hat ebenfalls wieder zwei weitere Abzweigungen. Das Programm muss also durch den ganzen Datensatz hindurch, um das Element an die richtige Stelle zu sortieren.

Bubblesort

Bei dem Bubblesort werden die zu sortierende Elemente nicht in einer [Node](#) eingebunden. Der Bubblesort vergleicht jedes Element mit seinem rechten Nachbar, dem nächsten Element in der Liste. Sollte nun der Wert des Elementes höher sein als der des rechten Elementes, tauschen die beiden Elemente ihre Plätze. Dieses Verfahren wiederholt sich so lange, bis keine Änderungen, beim Vergleichen der Parameter, mehr auftreten.

Combsort

Der Combsort ähnelt dem [Bubblesort](#), hier existiert jedoch eine Lücke zwischen den zu vergleichenden Elementen. Diese Lücke wird mit jedem Durchgang um 1,3 verkleinert, dadurch kann gewährleistet werden, dass sich keine Cluster, eine Ballung an Elementen, bildet. Sollte die Lücken zwischen den Elementen am Ende „1“ betragen wird der Combsort zu einem Bubblesort, weil die zu vergleichende Elemente nun direkt nebeneinander, wie beim Bubblesort, liegen.

Gnomesort

Gnomesort führt dieselben Vertauschungsoperationen wie [Insertionsort](#) durch, vergleicht aber einige Elemente mehrmals. Wenn das Programm einen Parameter findet, der nicht in die Reihenfolge passt, schiebt er das Element so lange zurück, bis das Element in die Reihenfolge hineinpasst. Der Algorithmus ist vollendet, wenn es die letzte Position der Liste erreicht.

Heapsort

Der Heapsort hat zwei Phasen. In der ersten Phase werden die Elemente zu einem Binärbaum umgebaut. Sie besitzen wie zwei Unterknoten, die zusammen mit dem Hauptknoten jetzt verglichen werden. Sollte einer dieser Unterknoten größer als der Hauptknoten sein, wird er mit diesem ausgetauscht. Dieses Verfahren wiederholt sich für alle Zweige. In Phase zwei befindet sich die größte Zahl im [Array](#) an der ersten Position, im Hauptknotenpunkt. Dieses Element wird dann an das Ende des Arrays geschoben und für den restlichen Prozess ignoriert. Anschließend beginnt die Phase 1 und danach Phase 2 erneut, solange bis nur noch ein Element in dem Binärbaum übrigbleibt. Dieses ist dann das niedrigste Element in dem [Datensatz](#) und wird an den Anfang des Arrays hinzugefügt.

Insertionsort

In dem Insertionsort ist ein stabiles Sortierverfahren. Hierbei werden alle Elemente linear durchgegangen. Sollte auf ein Element getroffen werden, welches niedriger als das vorherige Element ist, wird das Element zurückverschoben, bis es höher als das nächste vorherige Element ist. Der Insertionsort ist mit dem Heapsort verwandt, weshalb auch hier alle Elemente sortiert sind, wenn Algorithmus am Ende des [Arrays](#) angelangt ist.

Introsort

Der Begriff Introsort ist eine Kurzform für „introspektives Sortieren“. Der Algorithmus ist eine Variation von Quicksort, welches im [Worst-Case](#) auf ein anderes Sortierverfahren zurückfällt, wie z.B. [Heapsort](#). Dabei wird zu Beginn des [Rekursionsschrittes](#) eine Bewertungsfunktion ausgeführt, die anschließend entscheidet, ob ein anderer Algorithmus für die Sortierung der Tabelle verwendet werden sollte. Dadurch wird die Sortiergeschwindigkeit erhöht.

Merge Insertion

Merge Insertion ist die Verbesserung des [Mergesort](#) Algorithmus. Denn wo Mergesort immer eine gleiche Anzahl an Vergleichen abhängig von der Eingabelänge benötigt, kommt Merge Insertion auch mit weniger Vergleichen aus. Dabei wird die Eingabe nicht in ungefähr gleich große Teillisten aufgespalten, sondern immer in die nächstgrößere Zweierpotenz. Jedoch wird dadurch der Sortieralgorithmus komplex und schwer zu verstehen.

Mergesort

Bei Mergesort werden die Elemente, in Phase 1, in einzelne Listen aufgeteilt. In der zweiten Phase werden zwei Listen sortiert zusammengeführt. Phase zwei wird so lange wiederholt, bis nur noch eine Liste, mit den ganzen Elementen übrigbleibt.

Quicksort

Der Quicksort sortiert das [Array](#) an Elementen in kürzeren Abschnitten. Dabei setzt es ein so genanntes „Pivot“ ein. Dieses Element bestimmt, wie der Algorithmus in diesem Durchgang die Elemente einordnet. Dabei gibt es zwei verschiedene Optionen. Entweder ist das Element kleiner als das Pivot-Element, dann wird es vor diesem eingeordnet, ist es jedoch größer, wird es nach diesem Pivot-Element eingeordnet. Dieser Zyklus wiederholt sich so lange, bis jedes Element in Reihenfolge ist. Bei dem Quicksort kommt die [Rekursion](#) zum Einsatz. Dieser ermöglicht es, einen simpleren Algorithmus zu definieren, der sich dann unendlich oft wiederholen kann. Diese Technik kommt ebenfalls beim Quicksort zum Einsatz. Bei jedem Zyklus wird ein neues Pivot-Element gewählt, sollte irgendwann die nur noch dieses Element in dem Abschnitt verfügbar sein, gilt das Element als sortiert.

Samplesort

Der Samplesort Algorithmus agiert ähnlich wie der Quicksort, jedoch benötigt der Samplesort 15% weniger Vergleiche. Diese Optimierung kommt durch den Einsatz von mehreren Pivot-Elementen in im gesamten Prozess. Dadurch wird die Menge an [Rekursionen](#) gesenkt und auch die Dauer des gesamten Prozesses wird verkürzt.

Selectionsort

Bei dem Selectionsort gibt es zwei verschiedene Bereiche in einem [Array](#), Bereich eins ist der sortierte Teil, auch S genannt. Der zweite Bereich verfügt über die unsortierten Elemente, wird U genannt. Wenn nun der Algorithmus die Elemente aufsteigend sortieren soll, muss er zuerst alle Elemente im Bereich U durchgehen. Er merkt sich dabei immer die Position des niedrigsten Elementes. Sobald er die Teilliste durchgegangen ist, verschiebt er das Element nun zu S. Dann wiederholt sich der Algorithmus von vorne.

Shakesort

Der Shakesort Algorithmus sortiert die Elemente abwechseln von oben nach unten. Dabei werden jeweils zwei benachbarte Elemente verglichen und gegebenenfalls vertauscht.

Shellsort

Shellsort basiert auf dem Insertionsort. Der erste Schritt bei diesem Algorithmus ist die Definierung des Abstandes zwischen den zu sortierenden Elementen, je größer der Abstand, desto kürzer ist die allgemeine Laufzeit der Sortierung. In jedem Zyklus wird der Abstand kleiner, bis er eins beträgt. Um die Elemente nun zu sortieren, greift Shellsort auf den [Insertionsort](#) zurück.

Smoothsort

Der Smoothsort ist eine Variation von Heapsort. Jedoch unterscheidet sich dieser Algorithmus in dem [Average-Case](#) und [Worst-Case](#) nicht dem Heapsort. Die einzige Verbesserung zum Heapsort ist im [Best-Case](#) zu finden.

Stooagesort

Der Stooagesort Algorithmus hat eine schlechtere Laufzeit als [Bubblesort](#), weshalb er in der realen Welt auch nur zur Anschauung genutzt wird. Der Algorithmus vergleicht die ersten zwei Elemente miteinander, wenn mehr als zwei Elemente in der Liste existieren, setzt er den Prozess fort, solange bis alle Elemente sortiert sind.

Swap-Sort

Bei dem Swap-Sort wird jedes Element beim ersten Verschieben, direkt an die richtige Stelle geschoben. Dafür wird für jedes Element, die kleineren Elemente gezählt damit sichergestellt werden kann, dass das ausgetauschte Element bereits in der richtigen, also

endgültigen Stelle steht. Jedoch weist der Algorithmus ein Nachteil auf. Jedes Element darf in dem zu sortierenden [Array](#) nur einmal vorkommen.

Timsort

Timsort ist ein hybrider Algorithmus, der von [Mergesort](#) und [Insertionsort](#) abgeleitet ist. Das Sortiervfahren erkennt bereits sortierte Abschnitte, absteigend sortierte Abschnitte werden umgedreht und abhängig von der Abschnittslänge eines [Arrays](#) wendet Timsort entweder ein optimiertes Mergesort oder ein optimiertes Insertionsort an. Dadurch kann der Algorithmus eine sehr schnelle Laufzeit hervorbringen.

Bogosort

Bei diesem Algorithmus wird so lange das Array zufällig gemischt, bis die Elemente in einer Reihenfolge sind. Dieser Algorithmus wird aufgrund seiner schlechten Laufzeit nicht in der realen Welt eingesetzt und gilt lediglich als Beispiel eines schlechten Algorithmus.

Slowsort

Der Slowsort Algorithmus sucht nach dem größten Element in dem [Array](#) und setzt es an das Ende des Arrays. Anschließend wiederholt sich der Prozess von vorne, solange bis das Array sortiert ist.

Nicht-vergleichbares Sortieren

Nicht-vergleichbares Sortieren sind alle Sortiervfahren die die zu sortierende Objekte nicht untereinander auf „kleiner“, „größer“ und „gleich“ verglichen werden können.

Bucketsort

Im Bucketsort werden die Eingabe Elemente in mehrere Buckets aufgeteilt. Die Elemente werden jedoch nicht nur in die Buckets verschoben. Anschließend können diese Elemente in den Buckets mit einem vergleichbaren Sortieralgorithmus sortiert werden. Am Ende werden die Buckets zusammengeführt in ein [Array](#).

Countingsort

Countingsort vergleicht die Elemente nicht direkt, sondern nur deren Häufigkeit, wie viel sie in dem [Array](#) vorkommen.

Radixsort

Radixsort ist ein auf [Bucketsort](#) oder [Countingsort](#) basierender Algorithmus. Es benutzt einen Unteralgorithmus um die Zahlen, nach jeder Stelle zu sortieren.

Flashsort

Flashsort versucht unmittelbar abzuschätzen, wohin ein Element im Rahmen der Sortierung etwa positioniert werden muss, wenn das niedrigste und höchste Element bekannt sind.

Wann sollte man welchen Algorithmus verwenden?

Jeden Sortieralgorithmus kann man wie folgt unterscheiden. Als erstes die Effizienz eines Algorithmus. Je mehr Speicher für den Prozess des Sortierens benötigt wird, desto unfreundlicher wird er zu benutzen. Aber auch ebenfalls ist die Dauer der Sortierung eine wichtige Komponente, denn die Daten sollen nicht lange brauchen, bis sie fertig sortiert sind. Als zweiten Parameter ist die Stabilität eines Sortieralgorithmus, je „intelligenter“ der Algorithmus ist, umso weniger Operationen müssen durchgeführt werden. Als letzten Parameter ist das Design, wie wird der Algorithmus implementiert. Sortieralgorithmen wie Bubble Sort sind einfach zu implementieren, aber ineffizient für große Datensätze. Schnellere Algorithmen wie Quick Sort und Merge Sort sind effizienter, aber komplexer zu implementieren. Insertion Sort ist effizient für kleine oder nahezu sortierte Listen, während Heap Sort gut für Datensätze ist, aber zusätzlichen Speicherplatz benötigt.

Sortieralgorithmen im Detail

Die vier Sortieralgorithmen Bubblesort, Insertionsort, Mergesort und Quicksort die in je zwei unterschiedlichen Programmiersprachen geschrieben werden sollten befinden sich in dem [GitHub Repository](#).

Quellenangabe

Findung der Sortieralgorithmen und der verschiedenen Sortierarten

- <https://de.wikipedia.org/wiki/Sortiervverfahren>

Was sind Algorithmen?

- <https://de.wikipedia.org/wiki/Algorithmus>

Was sind Tupel?

- <https://de.wikipedia.org/wiki/Tupel>

Was sind Arrays?

- [https://de.wikipedia.org/wiki/Array_\(Datentyp\)](https://de.wikipedia.org/wiki/Array_(Datentyp))

Was ist ein Datensatz?

- <https://de.wikipedia.org/wiki/Datensatz>

Was ist lexikographische Ordnung?

- https://de.wikipedia.org/wiki/Lexikographische_Ordnung
- https://de.wikiversity.org/wiki/W%C3%B6rter/Lexikographische_Ordnung/Beispiele

Binary-Tree-Sort (höhen-balanciert)

- https://de.wikipedia.org/wiki/Binary_Tree_Sort

Bubblesort

- <https://de.wikipedia.org/wiki/Bubblesort>

Combsort

- <https://de.wikipedia.org/wiki/Combsort>

Gnomesort

- <https://de.wikipedia.org/wiki/Gnomesort>

Heapsort

- <https://de.wikipedia.org/wiki/Heapsort>
- <https://www.happycoders.eu/de/algorithmen/heapsort/>

Insertionsort

- <https://de.wikipedia.org/wiki/Insertionsort>

Introsort

- <https://de.wikipedia.org/wiki/Introsort>

Merge Insertion

- https://de.wikipedia.org/wiki/Merge_Insertion

Mergesort

- <https://de.wikipedia.org/wiki/Mergesort>

Quicksort

- <https://de.wikipedia.org/wiki/Quicksort>

Rekursion

- <https://de.wikipedia.org/wiki/Rekursion>

Samplesort

- <https://de.wikipedia.org/wiki/Samplesort>
- <http://users.atw.hu/parallelcomp/ch09lev1sec5.html>

Shakesort

- <https://de.wikipedia.org/wiki/Shakersort>

Shellsort

- <https://de.wikipedia.org/wiki/Shellsort>

Smoothsort

- <https://de.wikipedia.org/wiki/Smoothsort>

Stoogesort

- <https://de.wikipedia.org/wiki/Stoogesort>

Swap-Sort

- <https://de.wikipedia.org/wiki/Swap-Sort>

Timsort

- <https://de.wikipedia.org/wiki/Timsort>

Bogosort

- <https://de.wikipedia.org/wiki/Bogosort>

Slowsort

- <https://de.wikipedia.org/wiki/Slowsort>

Bucketsort

- <https://de.wikipedia.org/wiki/Bucketsort>

Buckets

- <https://www.bbc.co.uk/bitesize/guides/z2m3b9q/revision/3#:~:text=Data%20is%20often%20acted%20on,gathered%20back%20into%20a%20list.>

Countingsort

- <https://de.wikipedia.org/wiki/Countingsort>

Radixsort

- <https://de.wikipedia.org/wiki/Radixsort>

Wann/welcher Algorithmus?

- <https://studyflix.de/informatik/sortieralgorithmen-1337>