7. Sortieralgorithmen und Komplexität

In vielen Fällen müssen Datensätze sortiert werden. Je nach Größe des Datensatzes, verwendeter Programmiersprache, Verwendung von Zwischenergebnissen, verfügbarem Speicher und vielem mehr eignen sich verschiedene Sortieralgorithmen. Hier werden zwei Algorithmen vorgestellt und ihre Laufzeiten untersucht.

7.1. Bubblesort

Bubblesort

Bubblesort ist ein einfacher Sortieralgorithmus. Ein Array a wird in Bubblephasen von links nach rechts durchlaufen. Je zwei benachbarte Elemente werden miteinander verglichen und sollten sie noch nicht in der richtigen Reihenfolge sein (links < rechts), werden sie miteinander vertauscht. Der höchste Wert steigt also wie eine Blase nach oben, daher auch der Name *Bubble*sort.

Beispiel: Sortiere das Array (5,1,4,9,0,8,6)

1. Bubblephase	**2. Bubblephase**	**3. Bubblephase**	**4. Bubblephase**
(**5,1**,4,9,0,8,6) *tausch*	(**1,4**,5,0,8,6,9)	(**1,4**,0,5,6,8,9)	(**1,0**,4,5,6,8,9) *tausch*
(1,**5,4**,9,0,8,6) *tausch*	(1,**4,5**,0,8,6,9)	(1,**4,0**,5,6,8,9) *tausch*	(0,**1,4**,5,6,8,9)
(1,4,**5,9**,0,8,6)	(1,4,**5,0**,8,6,9) *tausch*	(1,0,**4,5**,6,8,9)	(0,1,**4,5**,6,8,9)
(1,4,5,**9,0**,8,6) *tausch*	(1,4,0,**5,8**,6,9)	(1,0,4,**5,6**,8,9)	(0,1,4,5,6,8,9)
(1,4,5,0,**9,8**,6) *tausch*	(1,4,0,5,**8,6**,9) *tausch*	(1,0,4,5,6,8,9)	
(1,4,5,0,8,**9,6**) *tausch*	(1,4,0,5,6,8,9)		
(1,4,5,0,8,6,9)			

In der **5. Bubblephase** stellt der Algorithmus fest, dass er fertig ist (wenn so ein vorzeitiges Ende implementiert ist, sonst durchläuft er noch eine 6. Phase).

125. Aufgabe

Aufgabe Bubblesort

Sortiere (e,d,a,b,f,e,c) mit Bubblesort. Notiere dabei, wie viele Vergleiche, Vertauschungen und Bubblephasen durchgeführt werden müssen. Durchlaufe nur so viele Phasen, wie nötig sind.

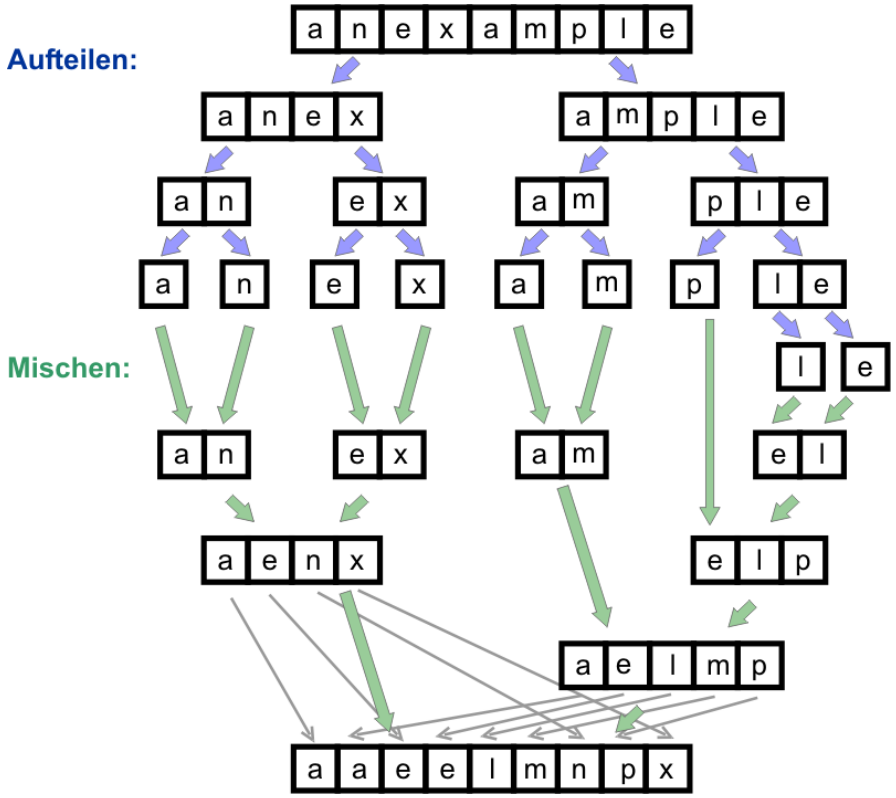
7.2. Mergesort

Mergesort

Beim **Mergesort** wird das zu sortierende Array in kleinere Arrays (bis nur noch ein Element in jedem Array ist) aufgeteilt. Dann werden die kleinen Arrays einzeln sortiert und zusammengeführt (mischen). Beim Zusammenführen zweier Arrays kann sehr schnell vollzogen werden, da diese ja bereits sortiert sind (siehe Video für eine genaue Erklärung).

Der Algorithmus lässt sich einfach rekursiv implementieren.

Beispiel: Sortiere das Array (a,n,e,x,a,m,p,l,e)



Von --Jkrieger 9. Jul 2005 16:01 (CEST) - selbst gezeichnet, Bild-frei, https://de.wikipedia.org/w/index.php?curid=793375

126. Aufgabe

Aufgabe Mergesort

Sortiere (3,1,4,5,2,5,7,8,6,5,2) mit Mergesort. Lass dabei das rechte Array mehr Elemente haben, als das linke, wenn nötig.

7.3. Laufzeiten

Laufzeiten

Damit man die Laufzeiten von Algorithmen besser vergleichen kann, werden nicht tatsächliche „Zeiten“ miteinander verglichen, denn diese sind ja von der Hardware der jeweiligen Rechner abhängig, sondern die Anzahl der Operationen, die bei einem Algorithmus ausgeführt werden müssen. Man notiert dies mit der **Landau-Notation** (oder $\mathcal{O}$**-Notation**). Kurz gesagt, beschreibt die Landau-Notation, den Verlauf einer Funktion f, indem eine Vergleichsfunktion g mit gleichem Verlauf angegeben wird. Wenn das

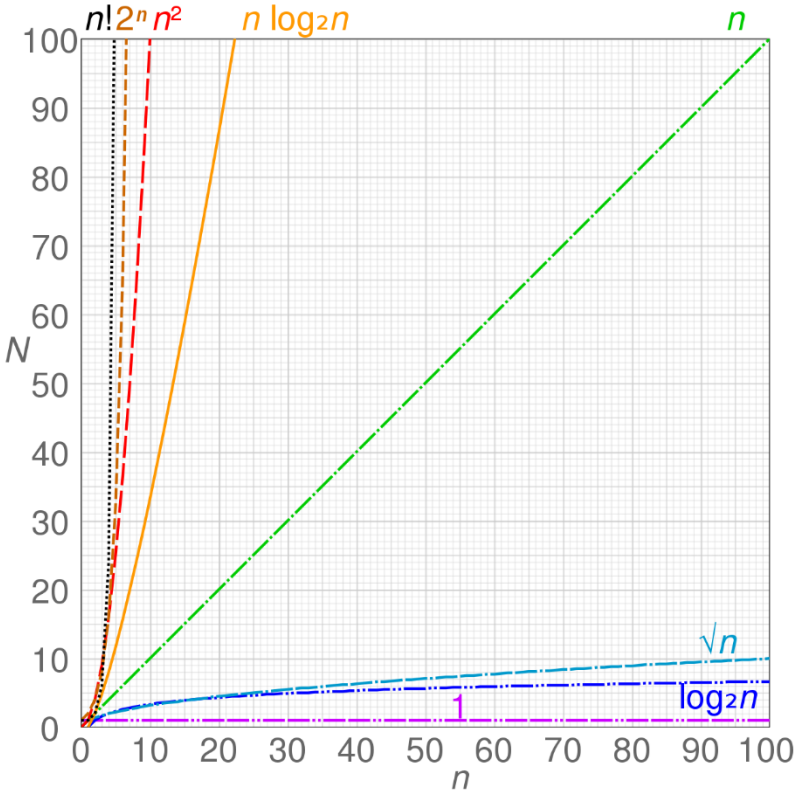
Wachstum von f kleiner oder gleich dem Wachstum von g ist (bis auf einen Steigungsfaktor $c \in \mathbb{R}$ und eine y-Achsenverschiebung), so schreiben wir $f \in \mathcal{O}(g)$.

Angenommen $f: \mathbb{N} \to \mathbb{N}$ beschreibt, wie viel Arbeitsschritte ein Programm F benötigt, abhängig von der der Größe der Eingabe n. Dann können z.B. folgende Laufzeiten auftreten (der Größe nach geordnet):

Notation	Bedeutung	Anschauliche Erklärung	Beispiel
$f \in \mathcal{O}(1)$	f ist konstant.	f hat immer dieselbe Laufzeit, egal wie groß die Eingabe ist.	Feststellen, ob eine Binärzahl gerade ist (kleinsten Bit Prüfen)
$f \in \mathcal{O}(\log n)$	f wächst logarithmisch.	f wächst ungefähr um einen konstanten Betrag, wenn sich n verdoppelt.	Suchen von Elementen in einem AVL-Baum (Suchbaum mit optimaler Datenverteilung)
$f \in \mathcal{O}(n)$	f wächst linear.	f wächst ungefähr auf das Doppelte, wenn sich n verdoppelt.	Durchlaufe ein Array mit n Elementen mit einer for-Schleife.
$f \in \mathcal{O}(n \cdot \log n)$	f wächst super-linear.		Sortieren von n Zahlen mit Mergesort.
$f \in \mathcal{O}(n^2)$	f wächst quadratisch.	f wächst ungefähr auf das Vierfache, wenn sich n verdoppelt.	Ein Programm mit 2 verschachtelten for-Schliefen, die jeweils n Elemente durchlaufen.
$f \in \mathcal{O}(n^k)$	f wächst polynomiell.	f wächst ungefähr auf das 2^k-fache, wenn sich n verdoppelt.	Ein Programm mit k verschachtelten for-Schleifen, die jeweils n Elemente durchlaufen.
$f \in \mathcal{O}(2^n)$	f wächst exponentiell.	f wächst ungefähr auf das Doppelte, wenn sich n um 1 erhöht.	Rekursiver Algorithmus, der sich selbst 2-mal aufruft.
$f \in \mathcal{O}(n!)$	f wächst faktoriell.	f wächst ungefähr auf das n+1-fache, wenn sich n um 1 erhöht.	Rekursiver Algorithmus, der sich selbst n-mal aufruft.

Betrachtet man die Vergleichsfunktionen graphisch, fällt auf, wie diese sich unterschiedlich für große Eingaben entwickeln:

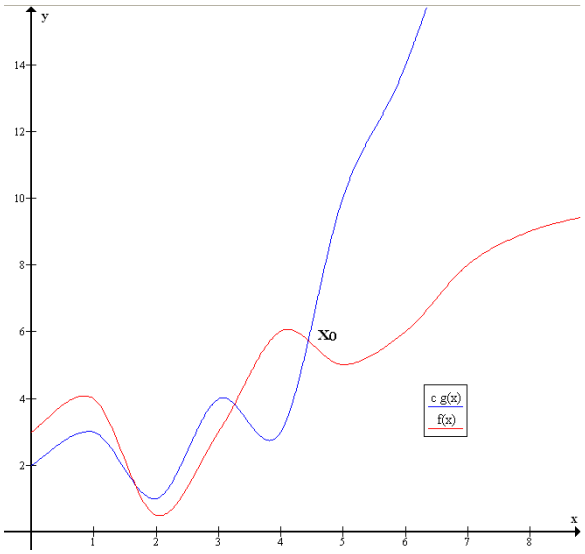
Beispiel



By Cmglee - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=50321072

Die Laundau-notation beschreibt, wie sich Funktionen für sehr große Eingaben entwickeln. Sie beschreibt nicht den genauen Verlauf der Funktion.

Beispiel: Die blaue Funktion g hat auf Dauer ein stärkeres Wachstum als die rote f. Also $\mathcal{O}(f) \leq \mathcal{O}(g)$



127. Aufgabe

Aufgabe Laufzeiten

Recherchiere die Laufzeiten von Bubblesort und Mergesort. Leite daraus ab für welche Aufgaben sich die Sortieralgorithmen jeweils eignen.

7.4. Übungsaufgaben

128. Aufgabe

Erstelle ein kleines Array mit zufälligen Werten (z.B. Würfel oder Zufallszahlengenerator). Sortiere diese dann mit Bubble- oder Mergesort. Vergleiche deine Ergebnisse mit https://www.hackerearth.com/practice/algorithms/sorting/bubble-sort/practice-problems/

7.5. Checkliste

- ☐ Ich kenne Bubblesort und Mergesort und kann diese anwenden.
- ☐ Ich kenne die durchschnittlichen Laufzeiten von Bubblesort und Mergesort.
- ☐ Ich kenne die Landaunotation und kann an dieser erkennen, ob ein Algorithmus schnell oder langsam arbeitet.
- ☐ Anhand der Laufzeiten eines Sortieralgorithmus kann ich erkennen, für welche Aufgaben er geeignet sein kann.
- ☐ Ich kann einen neuen Sortieralgorithmus mit Hilfe einer Anleitung verstehen und anwenden.