Algorithmen und Komplexität

Semester Nr. 2 Algorithmen und Komplexität bei Karl Stroetmann

Inhaltsverzeichnis

[Überblick 3](#_Toc42547905)

[Groß-O-Notation 4](#_Toc42547906)

[Motivation 4](#_Toc42547907)

[Definition der Groß-O-Notation 4](#_Toc42547908)

[Aussagen der Groß-O-Notation 4](#_Toc42547909)

[Reflexivität der Groß-O-Notation 4](#_Toc42547910)

[Multiplikation von Konstanten 4](#_Toc42547911)

[Addition 5](#_Toc42547912)

[Transitivität der Groß-O-Notation 5](#_Toc42547913)

[Grenzwert der Groß-O-Notation 6](#_Toc42547914)

[Lösen einer Rekurrenzgleichung 6](#_Toc42547915)

[Ausnahmefälle 6](#_Toc42547916)

[Arten der Groß-O-Notation 6](#_Toc42547917)

[Master-Theorem 7](#_Toc42547918)

[Bedingungen für das Master-Theorem 7](#_Toc42547919)

[Anwendung des Master-Theorems 7](#_Toc42547920)

[Sortieren 8](#_Toc42547921)

[Das Sortierproblem 8](#_Toc42547922)

[Definition Lineare Ordnung 8](#_Toc42547923)

[Definition Quasiordnung 8](#_Toc42547924)

[Definition Sortier-Problem 8](#_Toc42547925)

[Insertion Sort 9](#_Toc42547926)

[Selection Sort 10](#_Toc42547927)

[Merge Sort 11](#_Toc42547928)

[Quick Sort 12](#_Toc42547929)

# Überblick

1. Komplexität von Algorithmen

* Groß O-Notation

1. Rekurrenz Gleichungen

* Master Theorem

1. Sortier-Algorithmen:

* Sortieren durch einfügen (Insertion Sort)
* Sortieren durch Auswahl (Selection Sort)
* Sortieren durch Mischen (Merge Sort)
* Quicksort
* Radix Sort
* Heapsort

1. Abstrakte Datentypen
2. Dictionaries (und Mengen)

* Binäre Bäume
* AVL – Bäume + 2-3-Bäume
* Hash Tabellen
* Tries (Spezialfall Strings)

1. Prioritäts Warteschlangen
2. Graphentheoretische Algorithmen

Man muss nun zwischen einem Algorithmus und einem Programm unterscheiden. Ein Algorithmus ist eine abstrakte Darstellung, wie ein gegebenes Problem gelöst werden kann. Ein Programm hingegen ist die konkrete Implementierung dessen.

Des Weiteren sollte ein Algorithmus drei Kriterien erfüllen:

1. Ein Algorithmus muss korrekt sein
2. Ein Algorithmus sollte effizient sein in Bezug auf die Rechenzeit und den Speicherverbrauch
3. Ein Algorithmus sollte einfach sein.

# Groß-O-Notation

## Motivation

Wie berechnen Rechner die Zeiten eines Algorithmus?

1. Implementierung in Programmiersprache
2. Zählen von arithmetischen Operationen und Speicherzugriffen
3. Nachschlagen der Zeit der Operationen im Prozessorhandbuch
4. Berechnung der Rechenzeit

Ein Bild, das Karte enthält.

Automatisch generierte BeschreibungDie Groß-O-Notation ist eine abstrakte Möglichkeit, die das Wachstum der Rechenzeit in Abhängigkeit von der Größe der Eingabe beschreiben. Die O-Notation soll von konstanten Faktoren und unwesentlichen Termen abstrahieren.



Man definiert einen X-Wert (k), ab dem die Funktion  (rot) immer unter  liegt. Das c muss ebenfalls definiert werden und gibt einen Faktor der Funktion g(x) (blau) an, für welche dann das Wachstum von  ab k immer unterhalb von  verläuft.



k

### Definition der Groß-O-Notation



## Aussagen der Groß-O-Notation

### Reflexivität der Groß-O-Notation

Ein Bild, das Screenshot, Vogel enthält.

Automatisch generierte Beschreibung

### Multiplikation von Konstanten

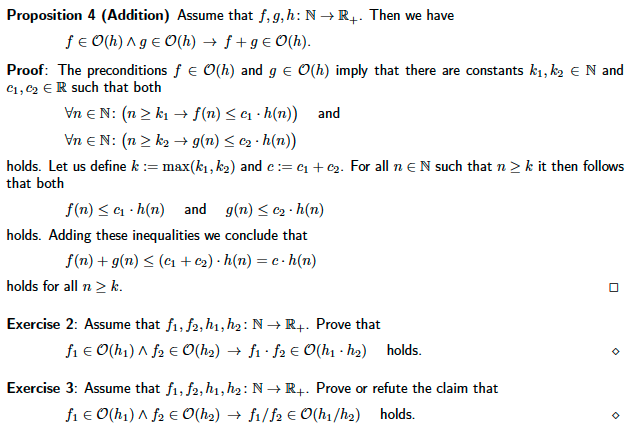
Ein Bild, das Screenshot, Vogel enthält.

Automatisch generierte Beschreibung

Ein Bild, das Vogel enthält.

Automatisch generierte Beschreibung

### Addition

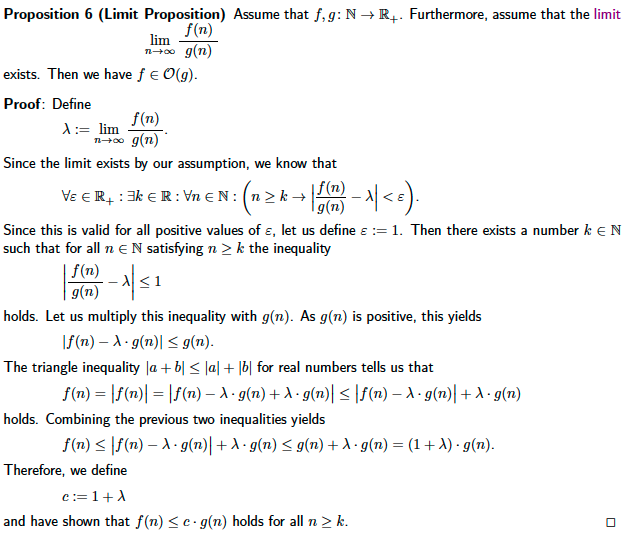


### Transitivität der Groß-O-Notation

Ein Bild, das Tisch, Anzeige, viele, Vogel enthält.

Automatisch generierte Beschreibung

### Grenzwert der Groß-O-Notation



## Lösen einer Rekurrenzgleichung

1. Homogenen Teil lösen ()
   * Die Rekurrenzgleichung ohne +1 am Ende
2. Inhomogenen Teil lösen ()
   * Die gesamte Rekurrenzgleichung
3. Einsetzen in die allgemeine Lösung
   * 
4. Gleichungssystem mit den Anfangsbedingungen Aufstellen
5. Lösen des Gleichungssystems und  &  bestimmen
6. Allgemeine Lösung aufstellen
   * 

### Ausnahmefälle

1. Gilt , dann ist die allgemeine Lösung
   * 
2. Wenn , dann ist der Ansatz zur Lösung der inhomogenen Rekurrenzgleichung
   * 

## Arten der Groß-O-Notation

1. Die O-Notation:
   * 
2. Die -Notation:
   * 
3. Die -Notation:
   * 

# Master-Theorem

Mit dem Master-Theorem kann man einfacher die Komplexität einer rekursiven Funktion bestimmen.

## Bedingungen für das Master-Theorem

1. , so dass , so dass  und
2. die Funktion  erfüllt die Rekurrenzrelation



## Anwendung des Master-Theorems

Man kann durch das Master-Theorem alle benötigten Variablen schnell ablesen und so die Komplexität des Algorithmus in einen der folgenden Fälle einordnen.

1. 
2. 
3. 

# Sortieren

## Das Sortierproblem

Es wird immer davon ausgegangen, dass eine Liste L gegeben ist. Alle Elemente aus L stammen aus der Menge S. S verfügt über eine binäre Relation , welche reflexiv, anti-symmetrisch und transitiv ist.

1.  ( ist reflexiv)
2.  ( ist anti-symmetrisch)
3.  ( ist transitiv)

### Definition Lineare Ordnung

Ein Paar  in dem S ein Set ist und  eine Relation auf , welche reflexiv, anti-symmetrisch und transitiv ist, wird partiell geordnet genannt. Wenn zudem noch



gilt, dann ist das Paar  in der totalen Ordnung und die Relation  wird als lineare Ordnung bezeichnet.

### Definition Quasiordnung

Ein Paar  ist in Quasiordnung, genau dann, wenn:

1. 
2.  ( ist reflexiv)
3.  ( ist transitiv)

Wenn eine Quasiordnung vorliegt und dazu noch

 (Linearität)

gilt, dann liegt eine Totale-Quasi-Ordnung vor.

### Definition Sortier-Problem

Gegeben:  ist in Totaler-Quasi-Ordnung. L ist eine Liste aus M.

Gesucht: Liste aus S mit zwei Eigenschaften:

1. Liste S ist aufsteigend sortiert



1. Liste S enthält dieselben Elemente wie L und doppelte Elemente in S kommen auch doppelt in L vor.

## Insertion Sort

Er durchläuft Schritt für Schritt eine Liste und entnimmt dabei aus der unsortierten Eingabefolge ein Element und setzt es dann an der entsprechend richtigen Stelle wieder ein.

#### Formale Definition



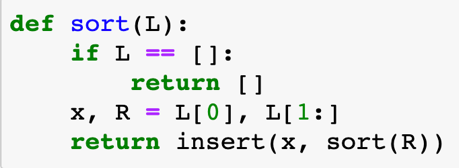
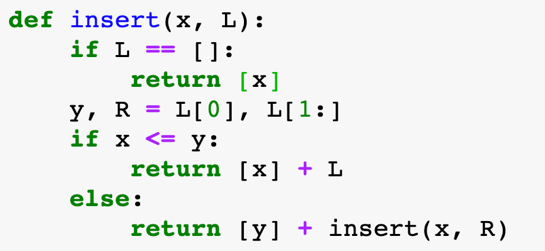








#### Implementierung



#### Komplexität

Insertion Sort ist gerade bei teilweise sortierten Listen sehr effizient.

##### Bester Fall

Liste ist bereits sortiert



##### Worst-case

Liste ist falsch herum sortiert



##### Durchschnittlicher Fall



## Selection Sort

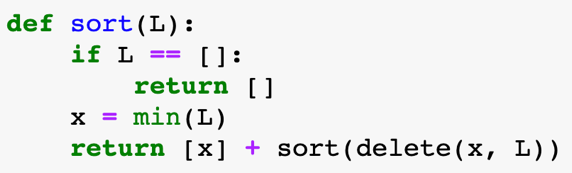
Das Minimum aus der Liste L wird entnommen und dort gelöscht. Das Minimum wird vor die Restliste gesetzt und dies geschieht rekursiv.

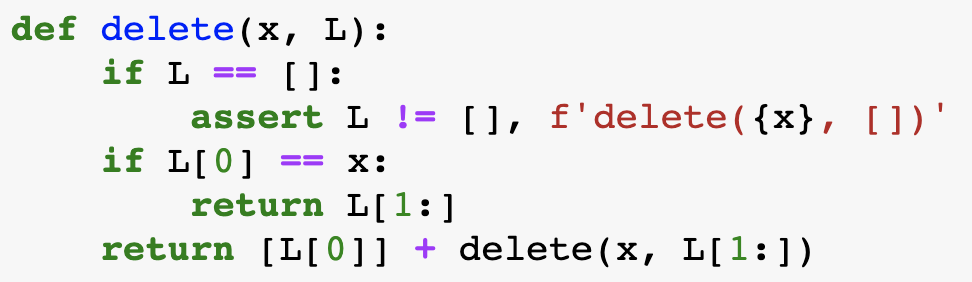
#### Formale Definition





#### Implementierung





#### Komplexität

##### Immer



## Merge Sort

Merge Sort arbeitet nach dem Teilen-und-Herrsche-Prinzip. Es werden immer zwei geleichgroße Teillisten gebildet und sortiert und im Anschluss zusammengemerged.

#### Formale Definition





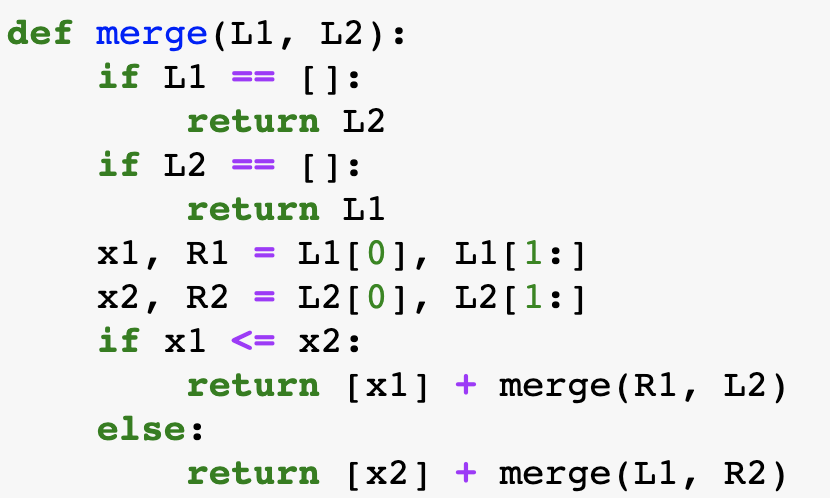
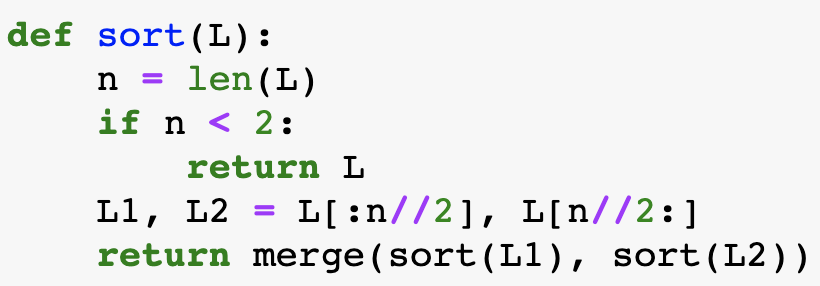








#### Implementierung



#### Komplexität

##### Im Worst-case



## Quick Sort

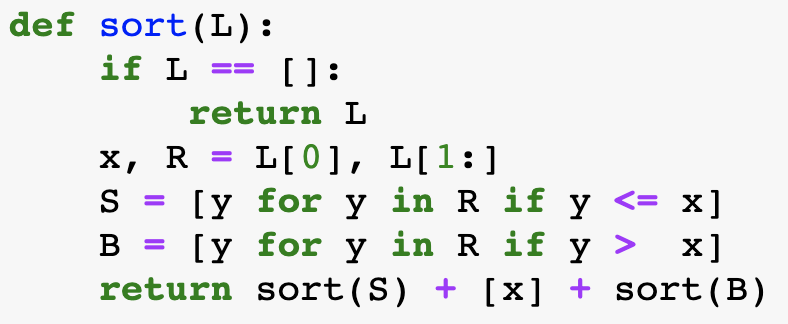
Man wählt ein Pivot-Element und vergleicht die anderen Elementen mit diesem. Ist das Pivot-Element größer, wird das Element in die Linke Teilliste gegeben und für den Fall, dass es kleiner ist in die rechte Teilliste. Dieser Schritt wird rekursiv ausgeführt.

#### Formale Definition





#### Implementierung



#### Komplexität

##### Worst-Case



##### Durchschnittlicher Fall



Also 

## Counting Sort

Counting Sort zählt die Häufigkeit der einzelnen Elemente und fügt diese in Sublisten ein, damit ist die Liste dann sortiert.

##### Formale Definition