

Informatik

Noah Aichhorn



2024/25

SIemens AG

Inhaltsverzeichnis

[Algorithmen 2](#_Toc183538329)

[Eigenschaften 2](#_Toc183538330)

[ Endlichkeit: Die Beschreibung des Algorithmus muss endlich sein. 2](#_Toc183538331)

[ Terminierung: nach endlich vielen Schritten muss der Algorithmus enden und ein Ergebnis liefern. 2](#_Toc183538332)

[ Determiniertheit: Der Algorithmus muss bei gleichen Voraussetzungen stets das gleiche Ergebnis liefern. 2](#_Toc183538333)

[ Determinismus: Zu jedem Zeitpunkt der Ausführung besteht höchstens eine Möglichkeit der Fortsetzung. Der Folgeschritt ist also eindeutig bestimmt. 2](#_Toc183538334)

[Beispiele für Algorithmen im Alltag 2](#_Toc183538335)

[Struktogramm eines euklidischen Algorithmus 3](#_Toc183538336)

[Beispiel eines euklidischen Algorithmus am Papier: 3](#_Toc183538337)

[Daten Strukturen 4](#_Toc183538338)

[Überblick 4](#_Toc183538339)

[Daten- 5](#_Toc183538340)

[Allgemeine und Binärbäume 5](#_Toc183538341)

[AVL-Bäume 6](#_Toc183538342)

# Algorithmen

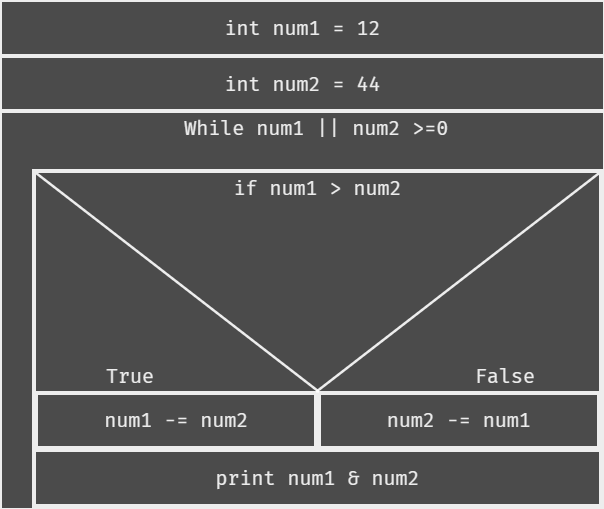
Ein Algorithmus bezeichnet eine systematische, logische Regel oder Vorgehensweise, die zur Lösung eines vorliegenden Problems führt. In der Informatik sind sie die Grundlage für Softwareprogramme und bestimmen, wie Daten verarbeitet, analysiert und genutzt werden.

Eigenschaften

* Eindeutigkeit: Ein Algorithmus darf keine widersprüchliche Beschreibung haben. Sie muss eindeutig sein.
* Ausführbarkeit: jeder Einzelschritt muss ausführbar sein.
* Endlichkeit: Die Beschreibung des Algorithmus muss endlich sein.
* Terminierung: nach endlich vielen Schritten muss der Algorithmus enden und ein Ergebnis liefern.
* Determiniertheit: Der Algorithmus muss bei gleichen Voraussetzungen stets das gleiche Ergebnis liefern.
* Determinismus: Zu jedem Zeitpunkt der Ausführung besteht höchstens eine Möglichkeit der Fortsetzung. Der Folgeschritt ist also eindeutig bestimmt.

### Beispiele für Algorithmen im Alltag

Spiele, Suchmaschinen, Maschinelles Lernen

Struktogramm eines euklidischen Algorithmus

## Beispiel eines euklidischen Algorithmus am Papier:

110:74=1 R36

110 = 1\*74+36 74:36 = 2 R2

74 = 2\*36+2 36:2 = 18 R0

36 = 2\*18+0 = 0

# Daten Strukturen

### Überblick

* Array
* Verkettete Liste
* Queue (FIFO)
* Stack (LIFO)
* Heap (hierarchische Datenstruktur)
* Graph (Google Maps – Graphentheorie)
* Heap (Baum)
* Hashtabelle (Hashfunktion berechnet die Position eines Objektes in einer Tabelle)

##### Verkette Liste

* Head
* Tail
* Datentypen
* Cursor
* prevElement
* nextElement

Methoden:

* Insert()
* Remove()
* Next()
* Prev()
* Get()

Vertauschen von Elementen:

Ein Bild, das Schrift, Grafiken, Grafikdesign, Design enthält.

Automatisch generierte Beschreibung

1.next muss auf B zeigen.

A.prev muss auf B zeigen.

A.next muss auf 2 zeigen.

2.prev muss auf A zeigen.

B.prev muss auf 1 zeigen.

B.next muss auf A zeigen.

Daten-Bäume

## Allgemeine und Binärbäume

Ein Bild, das Kreis, Design enthält.

Automatisch generierte Beschreibung

In Order: 50,60,75,100,150,200,220,250,300

Pre Order: 100,50,75,60,150,250,200,220,300

Post Order: 60,75,50,220,200,300,250,150,100

## AVL-Bäume

##### Balancewerte (Balancefaktor)

Der Balancewert eines Knotens gibt den Höhenunterschied zwischen seinem linken und rechten Teilbaum an.

Balancewert=Ho¨he des linken Teilbaums−Ho¨he des rechten Teilbaums\text{Balancewert} = \text{Höhe des linken Teilbaums} - \text{Höhe des rechten Teilbaums}Balancewert=Ho¨he des linken Teilbaums−Ho¨he des rechten Teilbaums

**Mögliche Werte:**

* **-1**: Der rechte Teilbaum ist höher.
* **0**: Die beiden Teilbäume sind gleich hoch.
* **+1**: Der linke Teilbaum ist höher.

Wenn der Balancewert eines Knotens außerhalb dieses Bereichs liegt (z. B. −2-2−2 oder +2+2+2), ist der Baum **unausgeglichen** und es muss eine Rotation durchgeführt werden.

Ein Bild, das Zeichnung, Darstellung enthält.

Automatisch generierte BeschreibungEin Bild, das Zeichnung, Diagramm, Clipart, Cartoon enthält.

Automatisch generierte Beschreibung

##### Ein Bild, das Diagramm, Kreis, Clipart, Design enthält. Automatisch generierte BeschreibungBeispiel

Ein Bild, das Text, Kreis, Schrift, Clipart enthält.

Automatisch generierte BeschreibungEin Bild, das Kreis, Text, Schrift, Clipart enthält.

Automatisch generierte BeschreibungEin Bild, das Kreis, Diagramm, Schrift, Clipart enthält.

Automatisch generierte BeschreibungEin Bild, das Kreis, Diagramm, Design enthält.

Automatisch generierte BeschreibungEin Bild, das Diagramm, Kreis, Schrift, Clipart enthält.

Automatisch generierte BeschreibungEin Bild, das Diagramm, Kreis, Schrift, Clipart enthält.

Automatisch generierte BeschreibungEin Bild, das Kreis, Diagramm, Clipart, Design enthält.

Automatisch generierte BeschreibungEin Bild, das Kreis, Diagramm, Schrift, Clipart enthält.

Automatisch generierte BeschreibungEin Bild, das Kreis, Diagramm, Schrift, Design enthält.

Automatisch generierte Beschreibung

55 und 24 sind kein Problem, bei der 12 muss man dann eine Drehung nach rechts machen. 17 wieder ganz normal. Damit die 15 passt muss man eine doppelte Rotation nach links machen. Nach der 14 wird eine einfache rechte Rotation gemacht. 35 einfach normal hinzufügen. Die 29 braucht eine einfache Drehung nach rechts. Wenn man 60 hinzufügen will muss man 24 und 35 einmal nach links drehen. Bei der 16 muss man dann eine doppelte links rotation machen. 60 kann ganz einfach gelöscht werden.

##### Rotationsregeln

Es gibt vier Rotationsarten, um den Baum wieder auszugleichen:

**1. Rechtsrotation (Single Right Rotation, SRR):**

* Wird angewendet, wenn ein Knoten **linksübergewichtet** ist (+2+2+2) und der linke Kindknoten ebenfalls einen **positiven** Balancewert (+1+1+1) hat.
* **Ziel**: Den rechten Teilbaum des linken Kindes „hochziehen“ und den Wurzelknoten „senken“.

**2. Linksrotation (Single Left Rotation, SLR):**

* Wird angewendet, wenn ein Knoten **rechtsübergewichtet** ist (−2-2−2) und der rechte Kindknoten ebenfalls einen **negativen** Balancewert (−1-1−1) hat.
* **Ziel**: Den linken Teilbaum des rechten Kindes „hochziehen“ und den Wurzelknoten „senken“.

**3. Links-Rechts-Rotation (Double Left-Right Rotation, DLR):**

* Wird angewendet, wenn ein Knoten **linksübergewichtet** ist (+2+2+2) und der linke Kindknoten einen **negativen** Balancewert (−1-1−1) hat.
* **Ablauf**: Zuerst wird eine **Linksrotation** am linken Kindknoten durchgeführt, gefolgt von einer **Rechtsrotation** am ursprünglichen Knoten.

**4. Rechts-Links-Rotation (Double Right-Left Rotation, DRL):**

* Wird angewendet, wenn ein Knoten **rechtsübergewichtet** ist (−2-2−2) und der rechte Kindknoten einen **positiven** Balancewert (+1+1+1) hat.
* **Ablauf**: Zuerst wird eine **Rechtsrotation** am rechten Kindknoten durchgeführt, gefolgt von einer **Linksrotation** am ursprünglichen Knoten.

# B-Bäume

##### Besondere Eigenschaften

**M-Wert**: Ein B-Baum ist ein balancierter Suchbaum, bei dem jeder Knoten mehrere Schlüssel enthalten kann. Die Ordnung mmm (eine feste positive Zahl) bestimmt, wie viele Schlüssel jeder Knoten maximal speichern kann (bis zu m−1m-1m−1) und wie viele Kinder er haben kann (bis zu mmm).

**Selbstbalancierend**: B-Bäume bleiben automatisch balanciert, sodass die Höhe des Baumes logarithmisch zur Anzahl der gespeicherten Elemente bleibt.

**Hoher Verzweigungsgrad**: Aufgrund der Möglichkeit, mehrere Schlüssel in einem Knoten zu speichern, haben B-Bäume eine deutlich größere Verzweigung als binäre Bäume, was die Höhe reduziert.

**Einsatz in Speicherhierarchien**: Die Struktur minimiert Festplattenzugriffe, da mehrere Schlüssel in einem Knoten gespeichert sind und diese in einem einzigen I/O-Block gelesen werden können. Dies macht sie ideal für Datenbanken und Dateisysteme.

##### Einsatzmöglichkeiten

* **Datenbanken:** B-Bäume und ihre Varianten (z. B. B+ Bäume) werden häufig in Datenbanken eingesetzt, um große Datenmengen effizient zu verwalten, da sie Festplattenzugriffe minimieren.
* **Dateisysteme:** Viele Dateisysteme, wie NTFS und EXT4, nutzen B-Bäume für die Organisation von Dateien.
* **Indexstrukturen:** Sie eignen sich hervorragend für Indizes in großen Datenstrukturen, z. B. in SQL-Datenbanken.

# Rot/Schwarz-Bäume

##### Besondere Eigenschaften

**Färbung der Knoten:** Jeder Knoten ist entweder **rot** oder **schwarz** gefärbt. Diese Färbung wird benutzt, um eine relative Balance im Baum sicherzustellen.

**Höhenbeschränkung:** Der Baum ist balanciert, indem sichergestellt wird, dass auf jedem Pfad von der Wurzel zu einem Blatt nicht mehr als doppelt so viele schwarze Knoten wie rote Knoten liegen.

**Rot/Schwarz-Regeln:** Es gibt strenge Regeln für das Färben und Rebalancieren, z. B. keine zwei roten Knoten dürfen direkt aufeinander folgen.

**Einfacheres Balancieren:** Rot/Schwarz-Bäume benötigen weniger Rotationen als AVL-Bäume, da sie eine **lockere Balancierung** garantieren.

##### Vor-/Nachteile gegenüber AVL-Bäumen

**Vorteile:**

* + **Effizientere Updates:** Rot/Schwarz-Bäume benötigen weniger Rotationen bei Einfüge- oder Löschoperationen, da sie weniger streng balanciert sind.
  + **Einfachere Implementierung:** Die Rebalancierung ist weniger kompliziert als bei AVL-Bäumen.
  + **Einsatz in Systemen:** Rot/Schwarz-Bäume werden häufig in der Praxis verwendet, z. B. in der Java-Standardbibliothek (TreeMap und TreeSet) oder der Linux-Kernelstruktur.

**Nachteile:**

* + **Schlechtere Suchperformance:** Wegen der lockereren Balancierung können Rot/Schwarz-Bäume etwas größere Höhen als AVL-Bäume haben, was die Suche minimal verlangsamen kann.
  + **Nicht für hochfrequente Suchoperationen optimiert:** Wenn die Suchgeschwindigkeit entscheidend ist, haben AVL-Bäume einen kleinen Vorteil, da sie stärker balanciert sind.