

Algorithmen und Datenstrukturen - eine Einführung

Dozent: Prof. Dr. Michael Eichberg
Kontakt: michael.eichberg@dhbw.de , Raum 149B
Version: 1.2.1.2
Quelle: Die Folien sind teilweise inspiriert von oder basierend auf:
Lehrmaterial von Prof. Dr. Scherer und Prof. Dr. Baumgart

Kontrollfragen: <https://delors.github.io/theo-algo-und-ds-einfuehrung/kontrollfragen.de.md.html>

Folien: <https://delors.github.io/theo-algo-und-ds-einfuehrung/folien.de.md.html>
<https://delors.github.io/theo-algo-und-ds-einfuehrung/folien.de.md.html.pdf>

Fehler melden: <https://github.com/Delors/delors.github.io/issues>

KI Verwendung: Bei der Erstellung der Unterlagen wurden KI Assistenten (insbesondere Claude aber ggf. auch ChatGPT, Ollama mit Gemma/LLama/Qwen oder OpenCode mit Kimi/Qwen/Deepseek...) unterstützend eingesetzt. Dies erfolgte insbesondere zur Unterstützung bei der Generierung von Grafiken (d. h. SVG Dateien), oder um sich Übersichtstabellen generieren zu lassen. Weiterhin wurde KI zur allgemeinen Qualitätssicherung eingesetzt. Inhalte, die ggf. von der KI vorgeschlagen wurden, wurden im Falle der Übernahme explizit validiert und angepasst.

1. Einführung

Was ist ein Algorithmus?

Definition

Ein Algorithmus ist eine endliche, wohl-definierte Abfolge von Anweisungen, die zur Lösung eines Problems führen.

Beispiel

- Berechnen / Arithmetik
- Suchen
- Sortieren
- Planen
- Simulieren
- Visualisieren und Interagieren

Die Spezifikation eines Algorithmus kann erfolgen...

- in einer Programmiersprache (z. B. Java, Python)
- in einer formalen Sprache (z. B. Z3, Alloy)
- in einer natürlichen Sprache (z. B. Deutsch, Englisch)
- in einer graphischen Sprache (Diagramme, z. B. UML)

Was ist eine Datenstruktur?

Definition

Eine Datenstruktur beschreibt eine Möglichkeit...

- Daten zu organisieren,
- Daten zu verwalten und
- Daten zu speichern,
- um sie effizient zu nutzen.

Beispiel

- *Primitive*
- Arrays
- Listen
- Bäume
- Graphen
- ...

2. Algorithmen

Algorithmus: Beispiel

Aufgabe

Gegeben ein nicht-sortiertes Array X mit N Integer Elementen. Was ist das größte Element?

Algorithmus - natürlich sprachlich

1. Nehme an, das erste Element ist das Maximum
2. Vergleiche das nächste Element des Arrays mit dem bisherigen Maximum.
3. Wenn es größer ist, aktualisiere das Maximum auf dieses Element
4. Wiederhole 2 und 3 bis wir am Ende des Arrays angelangt sind

🐘 Algorithmus

```
1 public int max(int[] x) {  
2     int max = x[0];  
3     for (int i = 1; i < x.length; i++) {  
4         if (x[i] > max) {  
5             max = x[i];  
6         }  
7     }  
8     return max;  
9 }
```

🐘 Algorithmus

```
1 def max(x):  
2     max = x[0]  
3     for i in range(1, len(x)):  
4         if x[i] > max:  
5             max = x[i]  
6     return max
```

Analyse (aller drei Implementierungen)

Zeitkomplexität:	hängt direkt von der Anzahl der Elemente ab
Speicherkomplexität:	eine Variable für das Maximum
deterministisch:	ja
endliche Ausführung:	ja

Übung

2.1. Mehrheitselement finden

1. Schreiben Sie einen Algorithmus, um das *Mehrheitselement* in einem unsortierten Array von Integer Werten zu finden. Das *Mehrheitselement* ist das Element, das **mehr als** $\lfloor n/2 \rfloor$ mal vorkommt. Sie können davon ausgehen, dass das Mehrheitselement immer vorhanden ist.
2. Wie schnell ist Ihr Algorithmus? Wovon hängt dies maßgeblich ab? (Best- und Worst-Case)
3. Wie viel Speicherplatz benötigt Ihr Algorithmus?

3. Datenstrukturen

Array

≡ Definition

Ein Array ist eine Datenstruktur, die eine Sammlung von Elementen gleicher Größe in einem kontinuierlichen, d. h. zusammenhängenden Speicherbereich speichert.

📄 Beispiel

```
1 var myIntArray = new int[5] {1,2,3,9,-5};  
1 var myBooleanArray = new boolean[2] {true,false};  
1 var myDoubleArray = new double[2] {1.0,-3};
```

? Frage

- Wie groß ist der Speicherplatz, den ein Array mit N Elementen benötigt?
- Wie schnell ist der Zugriff auf ein Element (n)?

Dynamische Arrays

Definition

Ein dynamisches Array (Liste) ist eine Datenstruktur, die eine Sammlung von Elementen gleicher Größe in einem kontinuierlichen, d. h. zusammenhängenden Speicherbereich speichert und erweiterbar ist.

- Ein dynamisches Array ist ein Array, das die Größe anpassen kann.

Typischerweise wird ein neues Array mit doppelter Größe erstellt, wenn der Speicherplatz erschöpft ist. Die Elemente des alten Arrays werden dann in das neue Array kopiert.

- dynamische Arrays werden basierend auf (statischen) Arrays implementiert

? Frage

- Wie schnell ist das Einfügen eines neuen Elements?
- Wie schnell ist das Löschen eines Elements?

Verkettete Listen



Definition

Eine verkettete Liste ist eine Datenstruktur, die eine Sammlung von Elementen in Knoten speichert.

- Jeder Knoten enthält einen Zeiger auf das nächste Element und speichert den aktuellen Wert.
- Im Falle einer doppelt verketteten Liste enthält jeder Knoten auch einen Zeiger auf das vorherige Element.
- Verkettete Listen sind dynamisch und können beliebig wachsen.
- Verkettete Listen sind nicht auf einen festen Speicherplatz angewiesen.



Übung - Verkettete Listen

3.1. Einfach verkettete Liste

1. Modellieren Sie die Klassen und Ihre Beziehungen mit Hilfe eines UML Klassendiagramms.
2. Implementieren Sie eine generische, einfach verkettete Liste in Java.
 - Implementieren Sie die Klasse für die Knoten als innere Klasse, der Klasse `LinkedList`.
 - Implementieren Sie folgende Methoden: `isEmpty()`, `prepend()`, `head()` und `deleteHead()`.
 - Achten Sie darauf, dass Ihr Programm immer aussagekräftige Ausnahmen wirft.
 - Implementieren Sie die notwendigen Interfaces, um eine foreach Schleife nutzen zu können, um über die Liste zu iterieren.

Übung - Verkettete Listen

3.2. Doppelt verkettete Liste

1. Implementieren Sie eine generische, doppelt verkettete Liste in Java. Der generische Typparameter sei `T`.

Orientieren Sie sich ggf. an der Implementierung für eine einfach verkettete Liste.

2. Implementieren Sie die folgenden Methoden. Bestimmen Sie das *worst-case* Verhalten der Operationen (Laufzeit) in Abhängigkeit von der Anzahl der Elemente N , die bereits in der Liste gespeichert sind.

- Einfügen eines neuen Wertes (am Ende der Liste; d. h. `append`)
- Einfügen eines neuen Wertes (am Anfang der Liste; d. h. `prepend`)
- Löschen eines Wertes (d. h. `remove(T value)`)
- Überprüfen ob ein Wert vorhanden ist
- eine Funktion (`forEachReverse`), um von hinten nach vorne durch die Liste zu iterieren

Warteschlangen (Queues)

- Queues realisieren das Konzept einer Warteschlange bei der die Elemente, die zuerst hinzugefügt wurden, auch zuerst wieder entfernt werden (FiFo).
- Eine Implementierungsstrategie für Queues ist die Verwendung einer verketteten Liste.