Algorithmen und Datenstrukturen - eine Einführung

Dozent: Prof. Dr. Michael Eichberg
Kontakt: michael.eichberg@dhbw.de, Raum 149B

Version: 1.1

Quelle: Die Folien sind teilweise inspiriert von oder basierend auf:
Lehrmaterial von Prof. Dr. Scherer und Prof. Dr. Baumgart

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

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

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

```java
public int max(int[] x) {
    int max = x[0];
    for (int i = 1; i < x.length; i++) {
        if (x[i] > max) {
            max = x[i];
        }
    }
    return max;
}
```

Algorithmus

```python
def max(x):
    max = x[0]
    for i in range(1, len(x)):
        if x[i] > max:
            max = x[i]
    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

```
var myIntArray = new int[5] {1,2,3,9,-5};
```

```
var myBooleanArray = new boolean[2] {true,false};
```

```
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.

  Typischwerweise 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

3.1. Doppelt verkettete Liste

1. Implementieren Sie eine generische, doppelt verkettete Liste in Java.
   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
   - Löschen eines Wertes
   - Überprüfen ob ein Wert vorhanden ist
   - eine Funktion (forEachReverse), um von hinten nach vorne durch die Liste zu iterieren