Basisinformationstechnologie I

Wintersemester 2022/23. Programmiersprachen – Grundlagen und Konzepte. *Basierend auf Jan Wieners' Folien*

Institut für Digital Humanities, Historisch-Kulturwissenschaftliche Informationsverarbeitung | Prof. Dr. Øyvind Eide | Slavina Stoyanova

Informatik und Programmierung

Structure and Interpretation of Computer Programs. Video Lectures by Hal Abelson and Gerald Jay Sussman. Lecture 1a: Overview and Introduction to Lisp

https://www.youtube.com/watch?v=XYKRVNQ_MqE

Cf. Abelson, H. und Sussman, G.J. (1996). *Struktur und Interpretation von Computerprogrammen : eine Informatik-Einführung.* 4., durchges. Aufl. Berlin: Springer

Abelson, H. and Sussman, G.J. (1996). *Structure and interpretation of computer programs.* Cambridge, Mass.: MIT Press.

Programmiersprachen und ihre Unterschiede

Differenzierung anhand von C++ und JavaScript:

- Compiler vs. Interpreter
- Lexik, Syntax, Semantik, Pragmatik
- Paradigmen
- Typisierung: Dynamisch vs. statisch
- Variablen: Deklaration vs. Initialisierung
- Hardwarenahe Programmierung: C++ und Zeiger
- Gemeinsamkeiten: Auswahlanweisungen und Kontrollstrukturen
- (Objektorientierung: Klassen, Geheimnisprinzip, Kapselung)

Strukturierte Computerorganisation

Assembler

Beispiel: „Hello World“ :

```
DATA SEGMENT                ;- Beginn des Datensegments
Meldung db "Hello World"    ;- Die Zeichenkette "Hello World"
        db "$"              ;- Endzeichen der Zeichenkette
DATA ENDS                   ;- Ende des Datensegment
CODE SEGMENT                ;- Beginn des Codesegements
ASSUME CS:CODE,DS:DATA      ;- Dem Assembler die Segmente mitteilen
Anfang:                     ;- Label für den Anfang des Programms
mov ax, DATA                ;- das Daten...
mov ds, ax                  ;  ...segment festlegen
mov dx, offset Meldung      ;- den Text in das auf DS bezogene Datenregister laden
mov ah, 09h                 ;- Die Unterfunktion 9 des Betriebssysteminterrupts 21h auswählen
int 21h                     ;- den Betriebssysteminterrupt 21h (hier erfolgt
                               Ausgabe des Texts) aufrufen
mov ax, 4C00h               ;- Die Unterfunktion 4Ch (Programmbeendigung) des
                               Betriebssysteminterrupts 21h festlegen
int 21h                     ;- diesen Befehl wiederum ausführen
CODE ENDS                   ;- Ende des Codesegments
END Anfang                  ;- dem Assembler das Ende des Labels Anfang mitteilen
```

http://de.wikipedia.org/wiki/Assemblersprache

C++

```
std::cout << “Hello World“;
```

Programmiersprachen

- Programmiersprache: Eine zum Formulieren von **Programmen** geschaffene **künstliche Sprache**
- Anweisungen, die wir dem Computer geben, werden als Text formuliert, z.B. in Python:

```
print "Hello World!"
```

- In C++:

```
cout << „Hello World“;
```

- In Common LISP

```
(princ "Hello World")
```

Programmiersprachen

- Programmtext ist formuliert nach festen Regeln: Festgelegt durch die **Grammatik** einer Programmiersprache
- Beispiel C++:

```
cout << „Hello World“;
```

  - Grammatik schreibt u.a. vor, dass der Ausdruck `cout << „Hello World“` mit einem Semikolon abgeschlossen werden muss

Programmiersprachen

Wer prüft zu welchem Zeitpunkt die Grammatik?

- Programme, die in einer höheren Programmiersprache formuliert sind, können nicht unmittelbar von einem Rechner ausgeführt werden, sondern müssen erst in eine Folge von Maschinenbefehlen übersetzt werden

- Maschinenbefehle == elementare Operationen, z.B.:
  - Daten aus dem Speicher lesen
  - Elementare arithmetische Operationen ausführen
  - Daten in den Speicher schreiben
  - Sprünge: Berechnungen an bestimmter Stelle fortsetzen

Programmiersprachen

Compiler: Computerprogramm, das ein in einer Hochsprache (C++, etc.) formuliertes Programm, das sog. **Quellprogramm** in ein **Zielprogramm**, z.B.:

- Bytecode (Sammlung von Befehlen für eine virtuelle Maschine)
  oder
- Maschinencode (Instruktionen, die der entsprechende Prozessor (Hardware) direkt umsetzen kann)

übersetzt

„kompilieren“: Verb, Anwendung eines Compilers auf ein Quellprogramm

Compiler

Kompilierung zumeist in mehreren Phasen:

- **Analysephase**
  - **Lexikalische** Analyse: Unterteilung des Quelltextes in bedeutungstragende Elemente, z.B. Schlüsselwörter, Zahlen, Operatoren
    → Dieser Teil der Kompilierung wird als **Scanner** oder **Lexer** bezeichnet
  - **Syntaktische** Analyse: Überprüft, ob der Quellcode der Syntax entspricht, d.h. ein wohldefinierter Satz der entsprechenden Programmiersprache ist
    → **Parser**
  - **Semantische** Analyse: Überprüft weiterführende Anforderungen; z.B. muss in einigen Programmiersprachen gewährleistet sein, dass jeder Bezeichner vor seiner Benutzung deklariert worden ist
- **Synthesephase**
  - Zwischencodeerzeugung
  - Programmoptimierung
  - Codegenerierung

Compiler vs. Interpreter

- Ein **Compiler** übersetzt immer einen kompletten Programmtext in eine Folge von Maschinenbefehlen, bevor die erste Programmanweisung ausgeführt wird.
- Ein **Interpreter** dagegen übersetzt immer nur **eine einzige Programmanweisung** in ein kleines **Unterprogramm aus Maschinenbefehlen** und führt dieses sofort aus. Anschließend wird mit der nächsten Anweisung genauso verfahren.

- **Interpreter Pro**: Einfacher zu konstruieren als Compiler
- **Interpreter Contra**: Ein Befehl, der mehrfach ausgeführt wird, muss jedes mal erneut übersetzt werden

Programmiersprachen: Das muss sein

Bei der Definition einer Programmiersprache muss ihre **Lexik, Syntax, Semantik** und **Pragmatik** definiert werden:

- **Lexik:** Definiert die gültigen Zeichen und Wörter, aus denen Programme der Programmiersprache zusammengesetzt sein dürfen.
- **Syntax:** Definiert den korrekten Aufbau der Sätze aus gültigen Zeichen bzw. Wörtern, d.h. sie legt fest, in welcher Reihenfolge lexikalisch korrekte Zeichen bzw. Wörter im Programm auftreten dürfen.
- **Semantik:** Definiert die Bedeutung syntaktisch korrekter Sätze, d.h. sie beschreibt, was passiert, wenn bspw. bestimmte Anweisungen ausgeführt werden.
- **Pragmatik:** Definiert ihren Einsatzbereich, d.h. sie gibt an, für welche Arten von Problemen die Programmiersprache besonders gut geeignet ist.

Quelle: Boles, Dietrich (2015): Programmieren spielend gelernt mit dem Java-Hamster-Modell. http://www.java-hamster-modell.de/eBooks/hamster1.pdf.

Programmiersprachen: Klassifizierung

Grobunterscheidung:

- niedere (maschinennahe) Programmiersprachen
- Höhere Programmiersprachen

Programmiersprachen: Klassifizierung

Höhere Programmiersprachen:

- Weitaus einfacher zu handhaben als z.B. Assembler
- Problemorientiert

Unterscheidung höherer Programmiersprachen in fünf Kategorien / Programmierparadigmen:

- **Imperative** Programmiersprachen:
  Programme bestehen aus Folgen von Befehlen (BASIC, PASCAL, MODULA-2).
- **Funktionale** Programmiersprachen:
  Programme werden als mathematische Funktionen betrachtet (LISP, MIRANDA).
- **Prädikative/deklarative** Programmiersprachen:
  Programme bestehen aus Fakten und Regeln, die beschreiben, wie aus gegebenen Fakten neue Fakten hergeleitet werden können (PROLOG).
- **Regelbasierte** Programmiersprachen:
  Programme bestehen aus "wenn-dann-Regeln"; wenn eine angegebene Bedingung gültig ist, dann wird eine angegebene Aktion ausgeführt (OPS5).
- **Objektorientierte** Programmiersprachen:
  Programme bestehen aus Objekten, die bestimmte (Teil-)Probleme lösen und zum Lösen eines Gesamtproblems mit anderen Objekten über Nachrichten kommunizieren können (SIMULA, SMALLTALK).

Quelle: Boles, Dietrich (2015): Programmieren spielend gelernt mit dem Java-Hamster-Modell. http://www.java-hamster-modell.de/eBooks/hamster1.pdf.

Objektorientierte Programmiersprachen: C++

C++:

- Ermöglicht **maschinennahe** Programmierung (Stichw. „Zeiger“), als auch abstrakte Programmierung (i.e. **Objektorientierung**)
- Kompilierung über g++ Compiler, Microsoft Visual C++ Compiler, etc.

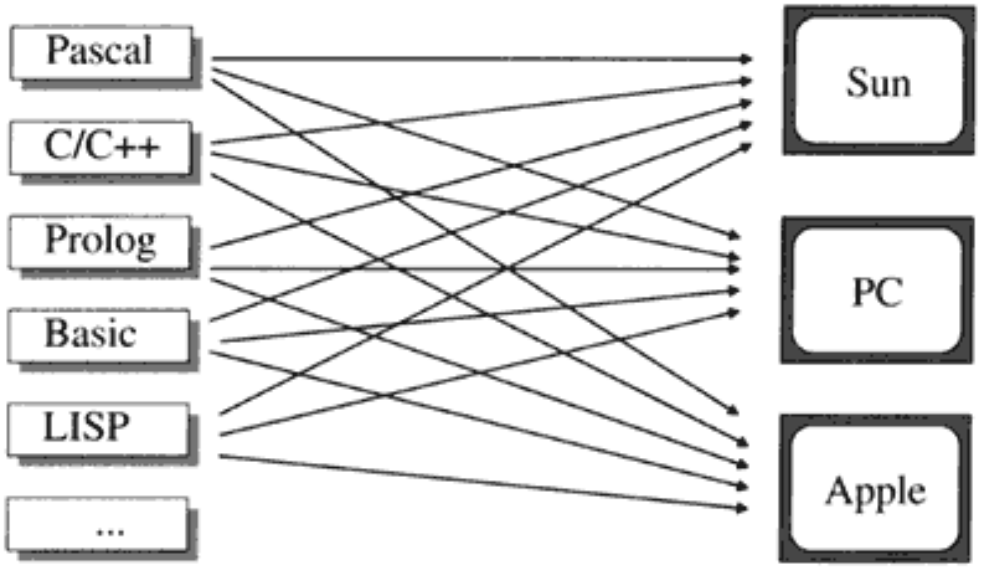


Abb.: n ×m viele Compiler

Objektorientierte Programmiersprachen: Java

Java (Java ist NICHT JavaScript!)

- Besonderheit: Java-Programme werden in Bytecode übersetzt, anschließend in einer Java-Laufzeitumgebung ausgeführt → **Virtuelle Maschine** (VM)

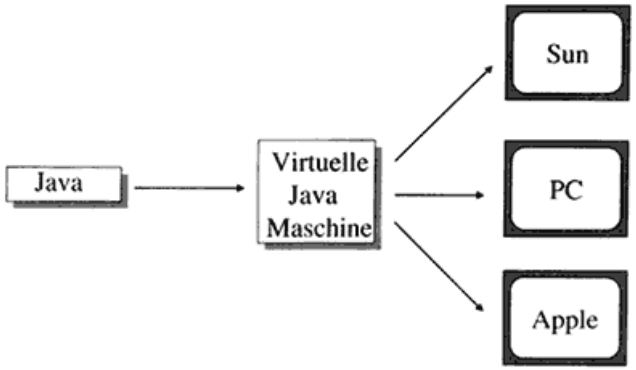


Abb.: Realität: Virtuelle Java-Maschine

- Vorteil: Plattformunabhängigkeit: Java-Programme laufen zumeist ohne weitere Anpassungen auf unterschiedlichen Computer- und Betriebssystemen, für die eine Java-VM existiert

Überblick

Suchen (und finden)

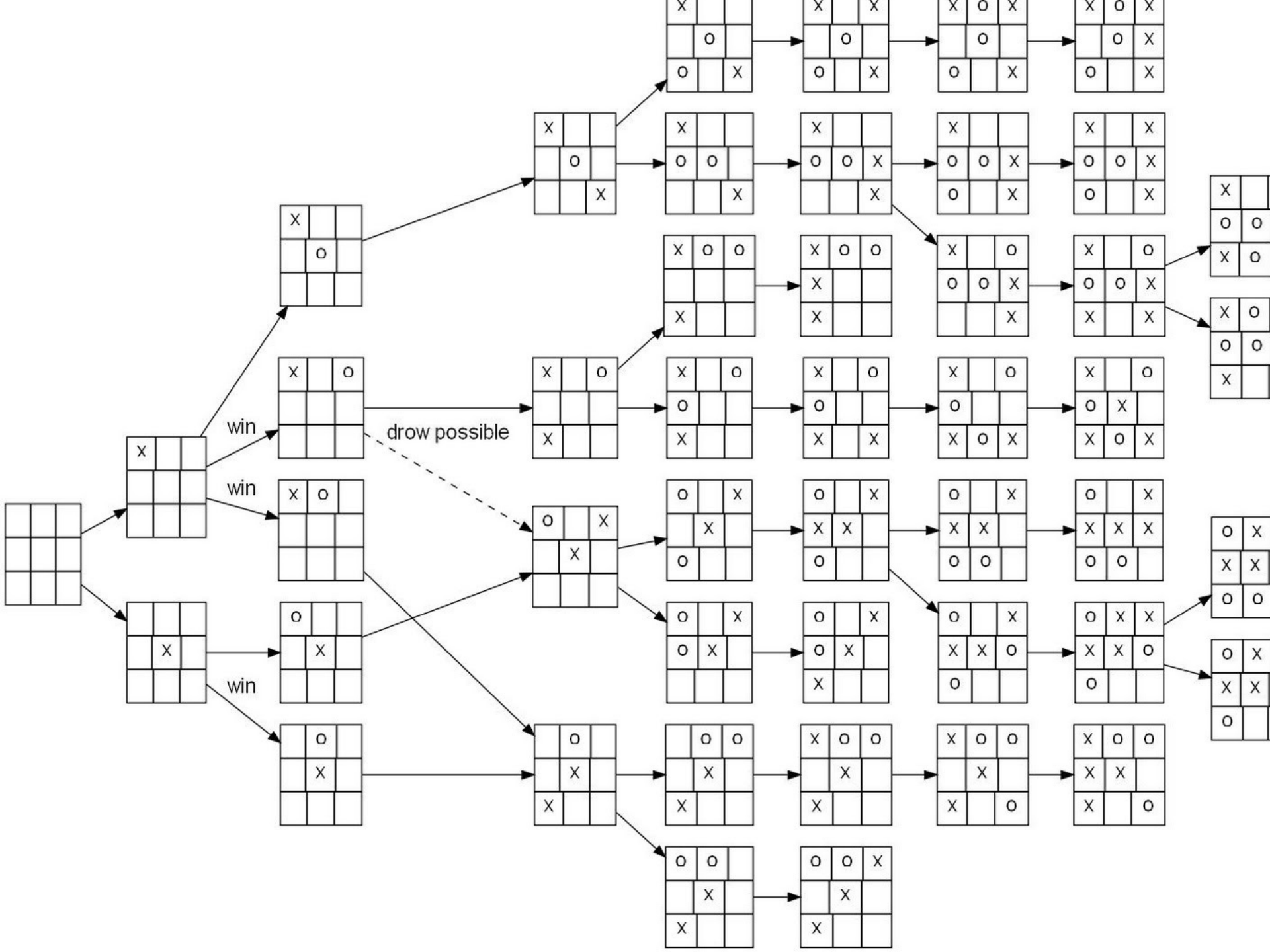
- lineare Suche
- binäre Suche

Sortieren

- Bubble Sort
- (TimSort)

Komplexitätsklassen: Landau-Notation / Groß-O-Notation

Suchen


win
drow possible
win
win


kanYeWest GRADUATION
PLAID & BOB JAROC
RAKIM
WU-TANG CLAN 8 DIAGRAMS
MOBB DEEP / THE INFAMOUS
THE STROKES ROOM ON FIRE
BECK MODERN GUILT
Beanfield Seek
MAD SKILLZ FROM WHERE???
björk debut
THE BEST OF JOY DIVISION
RZA
EGOEXPRESS WE DO WIE DU + HOT WIRE MY HEART
STING SONGS FROM THE LABYRINTH
BLOCKHEAD DOWNTOWN SCIENCE
NIRVANA IN UTERO
QUEENS OF THE STONE AGE
M.I.A. ARULAR
WU-TANG CLAN – WU-TANG FOREVER
SYSTEM OF A DOWN
swayzak
Alice COLTRANE
IMPULSE!
50 CENT THE MASSACRE
CYPRESS HILL III (TEMPLES OF BOOM)
Stunt Records Compilation Vol. 11
SURVIVAL OF THE FATTEST
JOHN COLTRANE MY FAVORITE THINGS
JOHNNY CASH
Massive Töne ...dichter in Stuttgart
CLARK
Booka Shade The Sun & The Neon Light
bass is boss
RADIOHEAD OK COMPUTER
NICK DRAKE Five Leaves Left
M.I.A. KALA
JEFF BUCKLEY LIVE AT SIN-É
NAS STREET'S DISCIPLE
stuff - keep it unreal reissue
THE TRENTEMØLLER CHRONICLES
BEATZ AUS DER BUDE
BECK ODELAY
M.A.N.D.Y.
DIE FIRMA
KAPITULATION
MR OIZO LAMBS ANGER
NICK DRAKE Pink Moon

Problemstellung Suche

Problemstellung: Wie finden wir eine gesuchte Audio CD?

Umgebungsvariablen:

- CD-Sammlung ist **unsortiert**
  ODER
- **Sortierte** CD-Sammlung

Lineare Suche

Sequenzielle / Lineare Suche in unsortierter Sammlung:

Anzahl **Suchvorgänge** bei n Elementen:

- minimal: 1(**best case**),
- maximal: **n** (**worst case**)
  → Durchschnittlich müssen die Hälfte der Einträge durchgesehen werden, um das gewünschte Element zu finden (**average case**)

- Anzahl der Suchschritte steigt proportional mit der Anzahl der Einträge

Zeitkomplexität eines Problems

Anzahl Suchvorgänge bei linearer Suche:

- minimal: 1(**best case**),
- maximal: **n** (**worst case**)

→ **Komplexitätsklasse O(n)**

Landau-Notation (Groß-O-Notation)

Komplexitätsklassen	Ordnung
Konstant	$O\,(1)$
Logarithmisch	$O\,(\log n)$
Linear	$O\,(n)$
n-log-*n*	$O\,(n \log n)$
Quadratisch	$O\,(n^2)$
polynomiell	$O\,(n^k)$ für k >= 1
exponentiell	$O\,(n^d)$ für d > 1

Sortieren

Sortieralgorithmen

Sortierstrategien / -Verfahren:

- Insertionsort: Sortieren durch Einfügen
  → Analog dem Vorgehen eines Kartenspielers: Neue Karten werden einzeln einsortiert, bevor die nächste Karte aufgenommen wird
- **Bubblesort**: Vergleichsbasierter Sortieralgorithmus
  → quadratische Komplexität mit $\boldsymbol{O\,(n^2)}$
- Heapsort
- Quicksort (vgl. C.A.R. Hoare)
- **TimSort** (verwendet in V8)
  → best case Komplexität: $\boldsymbol{O\,(n)}$
  → worst case Komplexität: $\boldsymbol{O\,(n \log n)}$

Bubblesort

Bubblesort: Sortiert z.B. ein Array von Datensätzen durch wiederholtes Vertauschen von Nachbarfeldern, die in falscher Reihenfolge stehen

Wird so lange wiederholt, bis das Array vollständig sortiert ist.

Dabei wird das **Array** in mehreren **Durchgängen** von **links nach rechts** durchwandert.

Bei jedem Durchgang werden alle Nachbarfelder verglichen und ggf. vertauscht. Nach dem 1. Durchgang hat man folgende Situation:

- Das größte Element ist ganz rechts.
- Alle anderen Elemente sind zwar zum Teil an *besseren* Positionen (also näher an der endgültigen Position), im Allgemeinen aber noch unsortiert.

Wandern des größten Elementes nach rechts → Aufsteigen von Luftblasen: Größte Luftblase steigt nach oben

Bubblesort

Zu sortieren ist die Zahlenfolge 55 07 78 12 42

(vgl. http://de.wikipedia.org/wiki/Bubblesort)

1. Durchlauf:

55 07 78 12 42

07 **55 78** 12 42

07 55 **78 12** 42

07 55 12 **78 42**

? ? ?

Bubblesort

Zu sortieren ist die Zahlenfolge 55 07 78 12 42
(vgl. http://de.wikipedia.org/wiki/Bubblesort)

2. Durchlauf:
07 55 12 42 78
07 **55 12** 42 78
07 12 **55 42** 78
07 12 42 **55 78**
07 12 42 55 78

3. Durchlauf:
07 **12 42** 55 78
07 12 **42 55** 78
07 12 42 **55 78**
07 12 42 55 78 → Fertig sortiert.