Datenstrukturen und Algorithmen

Sommersemester 2013

Positionsbestimmung

Informatik =

Problemlösung

- geg: Problem
- ges: Lösung

Positionsbestimmung

Informatik = Wissenschaft von der

Problemlösung

- geg: Problem
- ges: Lösung

Positionsbestimmung

Informatik = Wissenschaft von der

Problemlösung

- geg: Problem(klasse)
- ges: Lösung

Positionsbestimmung

Informatik = Wissenschaft von der **algorithmischen** Problemlösung

- geg: Problem(klasse)
- ges: Lösung

Positionsbestimmung

Informatik = Wissenschaft von der **algorithmischen** Problemlösung

- geg: Problem(klasse)
- ges: Lösungsprozedur

Positionsbestimmung

Informatik = Wissenschaft von der **algorithmischen** Problemlösung

- geg: Problem(klasse)
- ges: (automatisierbare) Lösung aus elementaren Operationen

Informationsverarbeitung

- **WAS ?**
  - Repräsentation/Organisation von Daten

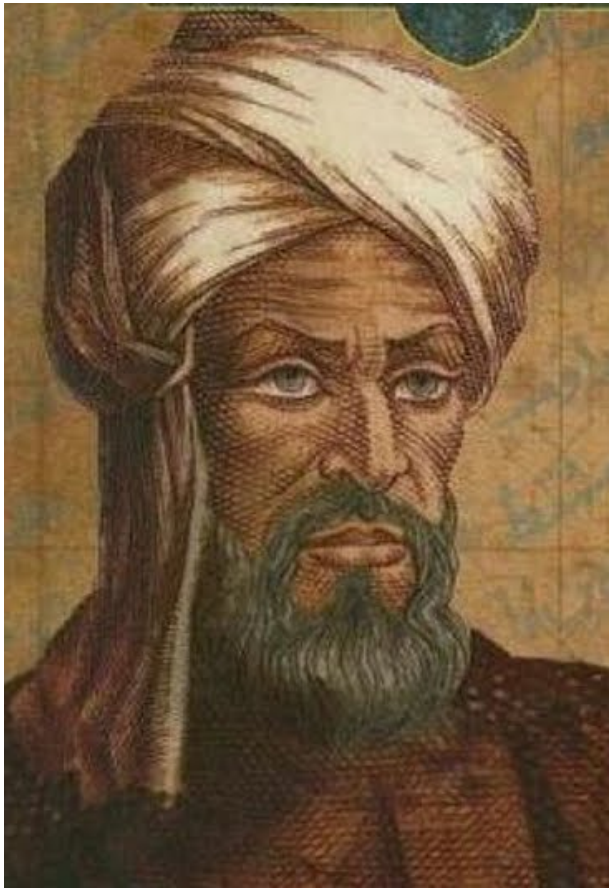
    → **Datenstrukturen**

- **WIE ?**
  - (schrittweise) Modifikation von Daten

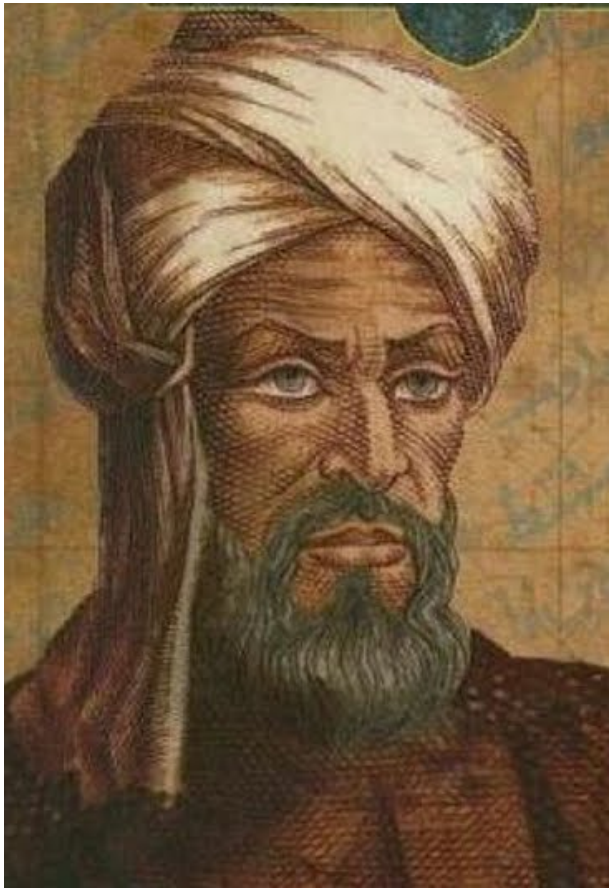
    → **Algorithmen**

- Software = Datenstrukturen + Algorithmen

Abu Dscha'far Muhammad ibn Musa al-Chwarizmi



Abu Dscha'far Muhammad ibn Musa al-Chwarizmi



Al-gorithmus

Rechenverfahren

```
13734 : 42 = 327
126
 113
  84
  294
  294
    0
```


Definition: Algorithmus

- Definition („Rezept“, „Prozedur“, …)

Schrittweise Modifikation von Daten zur Lösung eines Problems

- Eigenschaften (Donald Knuth)
  - Determinismus
  - Input (# ⩾ 0)
  - Output (# ⩾ 1)
  - Terminierung

Definition: Algorithmus

- Definition („Rezept“, „Prozedur“, …)

Schrittweise Modifikation von Daten zur Lösung eines Problems

- Eigenschaften (Donald Knuth)
  - Determinismus *(Randomisierung? Indeterminismus?)*
  - Input (# ⩾ 0)
  - Output (# ⩾ 1)
  - Terminierung

Definition: Algorithmus

- Definition („Rezept“, „Prozedur“, …)

Schrittweise Modifikation von Daten zur Lösung eines Problems

- Eigenschaften (Donald Knuth)
  - Determinismus *(Randomisierung? Indeterminismus?)*
  - Input (# ⩾ 0)
  - Output (# ⩾ 1)
  - Terminierung *(Online-Algorithmen? Anytime-Algorithmen?)*

Analyse von Algorithmen

- partielle Korrektheit
- totale Korrektheit
- Komplexität
  (Speicherplatz, Rechenzeit)
- Robustheit
  (bei inkorrekten Eingaben)

Das “Halteproblem”

```
while (n ≠ 1)
    if n is even then
        n ← n / 2
    else
        n ← 3 * n + 1
```

Das “Halteproblem”

```
while (n ≠ 1)
    if n is even then
        n ← n / 2
    else
        n ← 3 * n + 1
```

1

Das “Halteproblem”

```
while (n ≠ 1)
    if n is even then
        n ← n / 2
    else
        n ← 3 * n + 1
```

1 2

Das “Halteproblem”

```
while (n ≠ 1)
    if n is even then
        n ← n / 2
    else
        n ← 3 * n + 1
```

1 2 4

Das “Halteproblem”

```
while (n ≠ 1)
    if n is even then
        n ← n / 2
    else
        n ← 3 * n + 1
```

1 2 4 8

Das “Halteproblem”

```
while (n ≠ 1)
    if n is even then
        n ← n / 2
    else
        n ← 3 * n + 1
```

1 2 4 8 16

Das “Halteproblem”

```
while (n ≠ 1)
    if n is even then
        n ← n / 2
    else
        n ← 3 * n + 1
```

5

1 2 4 8 16 32

Das “Halteproblem”

```
while (n ≠ 1)
    if n is even then
        n ← n / 2
    else
        n ← 3 * n + 1
```

5 10

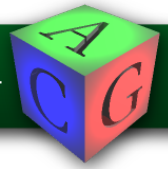
1 2 4 8 16 32 64

Das “Halteproblem”

```
while (n ≠ 1)
    if n is even then
        n ← n / 2
    else
        n ← 3 * n + 1
```

3 5 10 20 21

1 2 4 8 16 32 64 128



Das “Halteproblem”

- Angenommen, es existiere ein Programm

  bool terminator(P,n),

  das vorhersagen kann, ob das Programm P mit Input n terminieren wird.

Das “Halteproblem”

- Angenommen, es existiere ein Programm

  bool terminator(P,n),

  das vorhersagen kann, ob das Programm P mit Input n terminieren wird.

- X(n) = { if (terminate(X,n) == true) then
  while(true)
  else
  stop }

Das “Diagonale Halteproblem”

- Angenommen, es existiere ein Programm

  bool terminator(P,P),

  das vorhersagen kann, ob das Programm P mit dem eigenen Code als Input terminieren wird.

Das “Diagonale Halteproblem”

- Angenommen, es existiere ein Programm

  bool terminator(P,P),

  das vorhersagen kann, ob das Programm P mit dem eigenen Code als Input terminieren wird.

- X(P) = { if (terminate(P,P) == true) then
  while(true)
  else
  stop }

Das “Diagonale Halteproblem”

- Angenommen, es existiere ein Programm

```
bool terminate(
```

Wie verhält sich das Programm X mit dem Input X ?

X(X) ?

gramm P mit
de als Input terminieren wird.

- X(P) = { if (terminate(P,P) == true) then
  while(true)
  else
  stop }

Analyse von Algorithmen

- **Komplexität**
  - Effizienz (Praxistauglichkeit)
  - worst / best / average case
  - wie viel länger dauert die Berechnung, wenn der Input verdoppelt wird?
  - gibt es einen besseren Algorithmus? (Problem-Reduktion)

Einbettung in die Informatik

Problem → Spezifikation

Spezifikation → Algorithmus

Algorithmus → Programm

Programm → Computer

Top-Down Software-Entwurf

- Spezifiziere Eingabe / Ausgabe
- Problemlösung
  - Beziehungen, Referenzen, Daten
    → **Datenstrukturen**
  - Wesentliche Verarbeitungsschritte
    → Module, Prozeduren, **Algorithmen**
- Analyse
  - Korrektheit, Aufwand / Komplexität
- Implementieren / Testen / Dokumentieren

Software-Entwicklung

- **Datenstrukturen**
  - statische Beziehungen
  - (explizite) funktionale Zusammenhänge

- **Algorithmen**
  - dynamische Prozesse
  - (implizite) funktionale Zusammenhänge

Beispiel: Menge

Daten-struktur

$S \in 2^{IN}$ (d.h. $S \subseteq IN$)

$\min: 2^{IN} \rightarrow IN$

$\min(S) \in S \ \wedge \ \forall x \in S : \min(S) \leq x$

Implemen-tierung

```
int  S[ ], size, min()
```

Algorithmus

```
int i, min = S[ 0 ];
for (i=1 ; i<size ; i++)
    if (S[ i ] < min) then
        min = S[ i ];
```

Ziele der Vorlesung

- Grundlegende Konzepte für den Entwurf und die Analyse von Algorithmen
- Effiziente Implementierung (nicht: Code-Level Optimierung)
- Komplexitätsanalyse
- Repertoire an Standardalgorithmen

Übersicht

- **Datenstrukturen**
  - Konzepte
  - Standard-Typen
- **Algorithmen**
  - Entwurf und Analyse
  - Standard-Verfahren

Datenstrukturen

- abstrakte Datentypen
- konkrete Datentypen

Datenstrukturen

- abstrakte Datentypen
- konkrete Datentypen

lineare Strukturen	hierarchische Strukturen	Graph-Strukturen

Datenstrukturen

- abstrakte Datentypen
- konkrete Datentypen

lineare Strukturen	hierarchische Strukturen	Graph-Strukturen

Arrays
Listen
Stacks (Keller)
Queues (Warteschlangen)

Datenstrukturen

- abstrakte Datentypen
- konkrete Datentypen

lineare Strukturen	hierarchische Strukturen	Graph-Strukturen

Bäume
(binär, rot-schwarz, B-, …)
Prioritätswarteschlangen

Datenstrukturen

- abstrakte Datentypen
- konkrete Datentypen

lineare Strukturen

hierarchische Strukturen

Graph-Strukturen

gerichtet / ungerichtet
planare Graphen
...

Algorithmen

- abstrakt
  - allgemeine Entwurfsparadigmen
  - Komplexitätsanalyse

- konkret
  - Sortieren
  - Suchen
  - Optimieren

The Matrix

	linear	hierarchisch	Graphen
Sortieren			
Suchen			
Optimieren			

The Matrix

	linear	hierarchisch	Graphen
Sortieren			
Suchen			
Optimieren			

The Matrix

	linear	hierarchisch	Graphen
Sortieren			
Suchen			
Optimieren			

The Matrix

	linear	hierarchisch	Graphen
Sortieren			
Suchen			
Optimieren			

The Matrix

	linear	hierarchisch	Graphen
Sortieren			
Suchen			
Optimieren			

The Matrix

	linear	hierarchisch	Graphen
Sortieren			
Suchen			
Optimieren			

The Matrix

	linear	hierarchisch	Graphen
Sortieren	einfache Verfahren		
Suchen			
Optimieren			

The Matrix

	linear	hierarchisch	Graphen
Sortieren	einfache Verfahren	höhere Verfahren	
Suchen			
Optimieren			

The Matrix

	linear	hierarchisch	Graphen
Sortieren	einfache Verfahren	optimale Verfahren	
Suchen			
Optimieren			

The Matrix

	linear	hierarchisch	Graphen
Sortieren	einfache Verfahren	noch bessere Verfahren	
Suchen			
Optimieren			

The Matrix

	linear	hierarchisch	Graphen
Sortieren	einfache Verfahren	höhere Verfahren	
Suchen	Hashing K-Selektion		
Optimieren			

The Matrix

	linear	hierarchisch	Graphen
Sortieren	einfache Verfahren	höhere Verfahren	
Suchen	Hashing K-Selektion	Suchbäume (...)	
Optimieren			

The Matrix

	linear	hierarchisch	Graphen
Sortieren	einfache Verfahren	höhere Verfahren	
Suchen	Hashing K-Selektion	Suchbäume (...)	Traversen
Optimieren			

The Matrix

	linear	hierarchisch	Graphen
Sortieren	einfache Verfahren	höhere Verfahren	topol. Sort.
Suchen	Hashing K-Selektion	Suchbäume (...)	Traversen
Optimieren			

The Matrix

	linear	hierarchisch	Graphen
Sortieren	einfache Verfahren	höhere Verfahren	topol. Sort.
Suchen	Hashing K-Selektion	Suchbäume (...)	Traversen
Optimieren	Greedy		

The Matrix

	linear	hierarchisch	Graphen
Sortieren	einfache Verfahren	höhere Verfahren	topol. Sort.
Suchen	Hashing K-Selektion	Suchbäume (...)	Traversen
Optimieren	Greedy	Dyn. Programming Divide & Conquer	

The Matrix

	linear	hierarchisch	Graphen
Sortieren	einfache Verfahren	höhere Verfahren	topol. Sort.
Suchen	Hashing K-Selektion	Suchbäume (...)	Traversen
Optimieren	Greedy	Dyn. Programming Divide & Conquer	Spannbäume kürzeste Pfade MinCut