3. Entwurf von Algorithmen

3.1 Algorithmen, Programmiersprachen & Programme

Algorithmus:

ein Verfahren zur systematischen, schrittweisen Lösung eines Problems

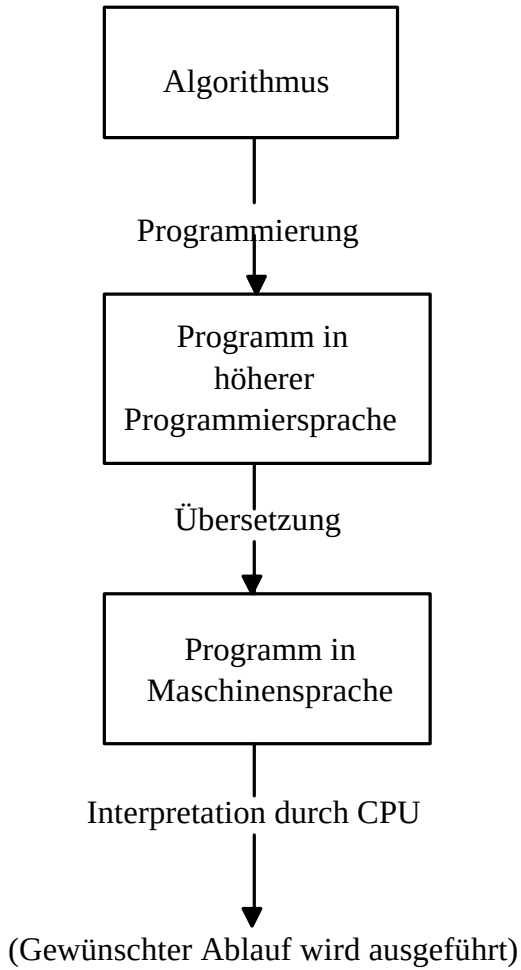
Beschrieben in:

- Umgangssprache
- Programmiersprache
- mathematischem Formalismus (z. B. Turing-Maschine)

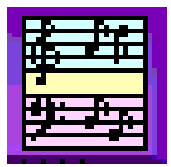
Church-Turing-These

"All reasonable definitions of 'algorithm' are equivalent."

Algorithmus vs. Programm



Beispiele für Algorithmen

Prozess	Algorithmus	Typische Schritte im Algorithmus
Pullover stricken	Strickmuster	Linke Masche, rechte Masche stricken
Modellflugzeug bauen	Montage-anleitung	Leime Rippe A an den Holm B
Kleider nähen	Schnittmuster	Nähe Saum
spiele Musik	Notenblatt	
kochen	Rezept	Zutaten (4 Personen) 400 g Tomatensoße, 1/2 l Wasser, 3 Teelöffel Chilipulver, 1 1/2 l Minutenreis, 1/2 kg zerstoßene Tortilla-Chips, 1/2 kg klein geschnittener Cheddarkäse Anleitung: vermenge Tomatensoße, Wasser, Chilipulver in einem mittelgroßen Topf und bringe es zum Kochen. Rühre den Reis ein, decke den Topf ab, nehme ihn vom Feuer und lasse ihn 10 Minuten stehen. Streue Tortilla-Chips und Käse darüber. Serviere das Essen, wenn gewünscht, mit Salat-blättern und saurer Sahne. Aus: Syd's Kochbuch.

Algorithmen als fundamentales Konzept

- unabhängig von der Programmiersprache
- unabhängig von der Rechner-Hardware

Der Entwurf von Algorithmen ist ein kreativer Prozeß, der nicht automatisiert werden kann.

Beispiel

Summiere alle Zahlen zwischen 1 und 100:

Lösung 1

1+2+3+4…+99+100 = 5050

Lösung 2

(1+100)+(2+99)+…(50+51) = 50*101 = 5050

Wir sehen: Die **Effizienz** eines Algorithmus ist ein wichtiges Entwurfskriterium.

Richtlinien

- Ist das Problem lösbar?
- Welche Schritte müssen ausgeführt werden, um einen **korrekten** Algorithmus zu erhalten?
  Korrektheit heißt:
  - Der Algorithmus erzeugt ein Resultat (ist endlich).
  - Das Resultat ist korrekt.
- Ist der Algorithmus effizient?

Beispiele

- **Unlösbares Problem**
  Halteproblem
  Finde einen Algorithmus, der feststellt, ob ein beliebiger Algorithmus in endlicher Zeit irgendein Ergebnis liefert (=anhält).
- **Sehr komplexes Problem**
  Problem des Handlungsreisenden (traveling salesman)
  Für eine beliebige Topologie: Finde den kürzesten Weg, der alle Städte verbindet, die der Handlungsreisende aufsuchen soll.

Schwierigkeiten beim Algorithmenentwurf

- Es existiert keine präzise Definition des Problems.
- Spezialfälle kommen vor, die nicht betrachtet wurden.
- Es ist schwer zu zeigen, dass der Algorithmus für *alle möglichen* Eingabedaten korrekt abläuft.

Beispiele für Schwierigkeiten (1)

Beispiel 1: Wegbeschreibung

1. am Laden rechts abbiegen
2. geradeaus bis zur nächsten Kreuzung
3. rechts abbiegen
4. dritte Querstraße links

An welchem Laden? (UNPRÄZISE)

Beispiele für Schwierigkeiten (2)

Beispiel 2: Finde Süden mit der Uhr

1. Richte kleinen Zeiger auf die Sonne.
2. Berechne die Winkelhalbierende zwischen dem kleinen Zeiger und 12.00 h.
3. Winkelhalbierende zeigt nach Süden.

Ausrichten des kleinen Zeigers durch Drehen am Stellknopf oder Drehen der gesamten Uhr? (UNPRÄZISE)

Funktioniert nicht auf der Südhalbkugel! (Sonderfall vergessen)

Beispiele für Schwierigkeiten (3)

Beispiel 3: Repariere Wasserhahn

1. Wasserhahn aufdrehen
2. warten, bis kein Wasser mehr fließt
3. Hahn abschrauben
4. Dichtung gegen neue Dichtung auswechseln
5. Hahn anschrauben
6. Hahn zudrehen
7. Haupthahn öffnen

Endet nicht; Schritt 2 unendlich; es wurde vergessen, zu Anfang den Haupthahn zuzudrehen!

Folgerung

Der Entwurf von Algorithmen muss **auf systematische Weise diszipliniert** durchgeführt werden. Methoden zum systematischen Entwurf und zur Analyse von Algorithmen (Komplexität und Korrektheit) sind hilfreich.

In einem zweiten Schritt wird der Algorithmus als Computerprogramm formuliert. Dieser Schritt wird als **Programmierung** bezeichnet. Durch die Verwendung einer in Syntax und Semantik wohldefinierten Sprache wird sichergestellt, dass die einzelnen Anweisungen

- unmissverständlich sind
- auf dem Zielrechner ausführbar sind (Mächtigkeit, Detaillierungsgrad).

Formulierungsfehler

Annahme:

Es existiert ein Algorithmus, der nicht zu komplex ist. Welche Fehler können in der Formulierung auftauchen?

Wir unterscheiden drei Klassen von Fehlern:

- syntaktische Fehler
- semantische Fehler
- logische Fehler

Klassifikation von Fehlern (1)

a) Syntaktische Fehler

Werden durch den Compiler oder Interpreter gefunden und als solche gekennzeichnet. Die Instruktion wird nicht ausgeführt.

Beispiel:

```
a = b * + c;
```

Klassifikation von Fehlern (2)

b) Semantische Fehler

Werden entweder während der Ausführung gefunden (Java Exception, *segmentation fault* im Betriebssystem) oder schlimmstenfalls nie gefunden.

Beispiel: Feldgrenzenverletzung. Der Vektor a habe 100 Elemente.

```
i = 101;
a[i] = 17;
```

Beispiel: Teilen durch 0

```
r = 0.0;
s = 170/r;
```

Klassifikation von Fehlern (3)

c) Logische Fehler

Das Programm tut nicht, was der Programmierer eigentlich wollte.

Beispiel:

```
// berechne Kreisumfang
circumference = PI * radius;
// anstatt 2*PI*radius
```

Viele semantischen und alle logischen Fehler können während des Testens und durch **formale Verifikation** gefunden werden.

Wichtig: Tests sind im Gegensatz zur formalen Verifikation keine Korrektheitsbeweise für ein Programm!

3.2 Systematischer Entwurf von Algorithmen

a) Problemstellung

(a) Ist die Problemstellung verstanden bzw. wie lautet das eigentliche Problem?

- Wenn die Problemstellung nicht verstanden ist, dann weitere Informationen einholen. Wenn der Kern des Problems klar hervorgehoben ist, dann das eigentliche Problem formulieren.

(b) Ist die Problemstellung klar und exakt?

- Wenn nein, dann weitere Informationen zur Problemstellung beschaffen oder Problemstellung selbst präzisieren.

(c) Hat das Problem bereits einen Namen?

- Wenn nein, dann dem Problem einen kurzen und treffenden Namen geben.

Problemstellung (2)

(d) Was ist bekannt und was ist gegeben?

- Zusammenstellen aller bekannten Eingabegrößen bzw. –objekte. Jedem Eingabeobjekt einen Namen geben und seinen Typ bzw. seine Art beschreiben.

(e) Was ist unbekannt bzw. was ist gesucht?

- Zusammenstellen aller bekannten Ausgabegrößen bzw. Ausgabe-objekte. Jedem Ausgabeobjekt einen Namen geben und seinen Typ festlegen.

(f) Welche Bedingungen für Ein- und Ausgabeobjekte müssen erfüllt sein bzw. werden gefordert?

- Zusammenstellen dieser Bedingungen (z. B. formelmäßige Zusam-menhänge).

Anmerkung zur Problemstellung

In der Praxis ist nur in den seltensten Fällen das zu lösende Problem von vorne herein klar beschrieben (spezifiziert)!

Gründe

Derjenige, der das Problem hat, und derjenige, der den Algorithmus entwerfen soll, sind oft nicht identisch.

Sie haben meist unterschiedliches Vorwissen über die Anwendungsumgebung.

Sie machen meist unterschiedliche „selbstverständliche“ Annahmen.

Deshalb:

Spezifikationen sollten gemeinsam erarbeitet werden.

Beispiel für eine unklare Aufgabenstellung (1)

Problemstellung

Berechne die Funktion f(x) = log(x)

Fragen

- Was ist die Basis des Logarithmus?
- Wie ist x gegeben? In welchem Datentyp? Welchen Wertebereich hat x?
- Wie soll f(x) ausgegeben werden?
- Mit welcher Genauigkeit soll gerechnet werden?
- Was soll passieren, wenn ein unzulässiges x eingegeben wird?

Beispiel für eine unklare Aufgabenstellung (2)

Spezifikation

$f(x) = \log_{10}(x)$

x, f(x) als IEEE *double precision floating point*-Werte

x: Eingabe

f(x): Ausgabe

Wertebereich: [0.3, 30]; in allen anderen Fällen Fehlermeldung ausgeben.

Systematischer Entwurf von Algorithmen (1)

b) Lösungsplan

(a) Ist dasselbe Problem oder ein ähnliches bzw. vergleichbares Problem bekannt?

- Wenn ja, dann versuche man, Kenntnisse über die Lösung zu erhalten.

(b) Ist ein allgemeineres Problem bekannt?

- Wenn ja, dann versuche man, Kenntnisse über die Lösung dieses Problems zu erhalten. Man prüfe, ob das gegebene Problem als Sonderfall des allgemeinen Problems behandelt werden kann. Wenn ja, dann wende man die allgemeine Problemlösung an. Wenn sich das Problem verallgemeinern lässt, ohne dass die Lösung erheblich schwerer wird, dann löse man das allgemeinere Problem.

Systematischer Entwurf von Algorithmen (2)

(c) Lässt sich das Problem in ein einfacheres Teilproblem oder in mehrere einfachere, in sich geschlossene Teilprobleme aufteilen?

- Wenn ja, dann teile man das Problem auf und löse die Teilprobleme.

(d) Man stelle einen in Schritte gegliederten Lösungsplan auf.

- Sind die Teilprobleme zu lösen, dann nehme man für jedes Teilproblem einen Schritt.

Kommentare zum Lösungsplan

a.) **Problem**: Zeichne ein Dreieck auf dem Bildschirm.

b.) **Allgemeineres Problem**: Zeichnen eines Polygons mit n Ecken auf dem Bildschirm. (Dreieck: n = 3, Rechteck: n = 4)

Lösung des allgemeineren Problems erheblich schwerer?

Dreieck:	Linie von Ecke 1 nach Ecke 2
	Linie von Ecke 2 nach Ecke 3
	Linie von Ecke 3 nach Ecke 1
Polygon:	Linie von Ecke 1 zu Ecke 2
	Linie von Ecke 2 nach Ecke 3
	...
	Linie von Ecke n nach Ecke 1

c.) **Teilproblem**: Zeichnen einer Linie zwischen zwei Punkten auf dem Bildschirm.

Was ist ein guter Algorithmus?

Kriterien

- Ausführungszeit (Komplexität)
- Speicherbedarf für Programm und Daten
- Entwicklungszeit. Wird zunehmend wichtiger!
- Gut strukturiertes, wartbares Programm: sehr wichtig!

3.3 Schrittweise Verfeinerung

Zerlegung eines Algorithmus in Teilalgorithmen, von denen jeder einzelne einfacher und überschaubarer ist als der ursprüngliche Algorithmus. Dadurch verbessern sich die Chancen, einen insgesamt korrekten Algorithmus zu entwerfen.

Die Verfeinerung wird so lange fortgesetzt, bis die einzelnen Schritte unmittelbar auf dem Prozessor ausgeführt werden können: vom Menschen bis hin zur Programmiersprache, vom Compiler bis hin zur Maschinensprache.

Beispiel für schrittweise Verfeinerung (1)

Algorithmus "Koche Kaffee"

1. Koche Wasser
2. Gib Kaffeepulver in die Tasse
3. Fülle Wasser in die Tasse

Beispiel für schrittweise Verfeinerung (2)

Erste Verfeinerung

(1.1) Fülle Wasserkessel

(1.2) Schalte Herdplatte an

(1.3) Warte, bis das Wasser kocht

(1.4) Schalte Herdplatte aus

(2.1) Öffne Kaffeeglas

(2.2) Entnehme einen Löffel Kaffee

(2.3) Kippe Löffel in die Tasse

(2.4) Schließe Kaffeeglas

(3.1) Gieße Wasser aus dem Kessel in die Tasse, bis die Tasse voll ist

Beispiel für schrittweise Verfeinerung (3)

Zweite Verfeinerung

{Algorithmus zur Zubereitung einer Tasse Kaffee}

{Zuerst Wasser kochen}

(1.1.1.) Stelle Kessel unter Wasserhahn

(1.1.2.) Drehe Wasserhahn auf

(1.1.3.) Warte, bis Kessel voll ist

(1.1.4.) Drehe Wasserhahn zu

(1.2.) Schalte Herdplatte an

(1.3.1.) Warte, bis Kessel pfeift

(1.4.) Schalte Herdplatte aus

Beispiel für schrittweise Verfeinerung (4)

Zweite Verfeinerung (Fortsetzung)

{Gib Kaffeepulver in die Tasse}

(2.1.1.) Nehme Kaffeeglas aus dem Fach

(2.1.2.) Entferne Deckel vom Kaffeeglas

(2.2) Entnehme einen Löffel Kaffee

(2.3.) Kippe Löffel in die Tasse

(2.4.1.) Schraube Deckel auf das Kaffeeglas

(2.4.2.) Stelle Kaffeeglas in das Fach zurück

{Fülle Wasser in die Tasse}

(3.1.) Gieße Wasser aus dem Kessel in die Tasse, bis die Tasse voll ist