1 Algorithmische Grundlagen

1.1 Begriffsklärung

- Fragen
- Begriffsklärungen
  - Abstraktionsebenen für Algorithmen und Datenstrukturen
  - Algorithmus
- Qualität von Algorithmen

Fragen

- Welche Rolle spielen Datenstrukturen und Algorithmen in der Informatik? Wozu werden sie überhaupt gebraucht?
- Welche Möglichkeiten gibt es, Algorithmen und Datenstrukturen formal zu beschreiben?
- Wie unterscheidet man ***was*** ein Algorithmus macht und ***wie*** er es macht? Ist diese Unterscheidung sinnvoll?

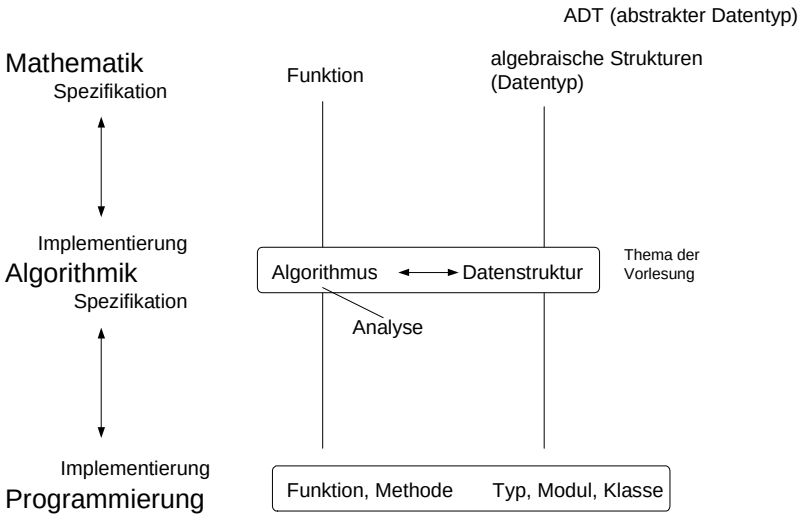
Fragen .

- Was versteht man unter der Qualität eines Algorithmus, und wie kann man diese messen ?
- Welcher Zusammenhang besteht zwischen Datenstrukturen und Algorithmen? Hängt die Qualität eines Algorithmus von der Datenstruktur ab, auf der er operiert?

Abstraktionsebenen

- Mathematik
- „Algorithmik“
- Programmierung

Abstraktionsebenen .



Beispiel - Aufgabe

Gegeben sei eine Menge S von ganzen Zahlen. Stelle fest, ob eine bestimmte Zahl c in S enthalten ist.

Lösung

Die Darstellung bzw. Lösung dieser Aufgabe kann auf jeder der drei Abstraktionsebenen erfolgen

- als mathematische Funktion
- als Algorithmus in einer Metasprache und
- als Programm in einer Programmiersprache

Spezifikation als Funktion

Sei F(Z) die Menge aller endlichen Teilmengen von Z, z.B. $\{1,3,7\} \in F(Z)$, aber für die Menge $M = \{x \mid x \text{ ist gerade}\}$ gilt: $M \notin F(Z)$. Dagegen ist $M \in P(Z)$ (Potenzmenge von Z = Menge aller Teilmengen von Z).

Sei BOOL = {true, false}

Definiere $f: F(Z) \times Z \rightarrow BOOL$

$$f(S, c) = \begin{cases} \text{true}, & \text{falls } c \in S \\ \text{false}, & \text{sonst} \end{cases}$$

Spezifikation als Algorithmus

Als Repräsentation für die Objektmenge wählen wir ein Array.

Algorithmus contains
Input S, ein Integerarray der Länge n
c, eine Integerzahl
Output true, falls c ∈ S, sonst false

Method-1

```
var b : bool;
b := false;
for i := 1 to n do
  if S[i] = c then b := true endif;
endfor;
return b;
end contains.
```

Spezifikation als Programm

```
program  test_membership (input, output);
type  intset = array[1..n] of integer;
    function  contains  (s: intset, c: integer) : boolean;
    var b: boolean;
    begin
        b := false;
        for i := 1 to n do
            if  s[i] = c then b := true;
        contains := b
    end;
    begin
        contains([2,3,6,8], 5)
    end; { test_membership }
```


Definitionen: **Algorithmus**

- Ein ***Algorithmus*** ist ein wohl-definiertes Berechnungsverfahren, das als ***Eingabe*** ein oder mehrere Werte bzw. Mengen von Werten erhält und als ***Ausgabe*** Werte bzw. Mengen von Werten erzeugt. Ein Algorithmus ist damit eine Folge von Berechnungsschritten, die eine Eingabe in eine Ausgabe transformiert.

Definitionen: **Algorithmus .**

- Ein Algorithmus ist ein Werkzeug, das ein wohl-spezifiziertes berechenbares Problem löst.
- Ein Algorithmus ist eine Anleitung zur Lösung eines Problems, wie ein Kochrezept, ein Strickmuster oder eine Reparatur- oder Montageanleitung.

Church‘sche These

Jede im intuitiven Sinne berechenbare Funktion ist auch Turing-berechenbar.

Die Gültigkeit der Church‘schen These würde bedeuten, dass alle verschiedenen Algorithmen-Begriffe nach Turing, Church, etc. äquivalent sind. Da man bisher kein Gegenbeispiel gefunden hat, wird die Church‘sche These von der Informatik als gültig akzeptiert.

Nicht-lösbare Probleme

Es gibt Probleme in der Informatik, von denen man beweisen kann, dass sie nicht lösbar sind. Dazu gehört das **Unvollständigkeitstheorem** von Gödel und das bekannte **Halteproblem**.

Beispiele für Aussagen über natürliche Zahlen

- Jede positive Integerzahl > 1, die keine Primzahl ist, ist zusammengesetzt (composite). Beispiele: 4 = 2 * 2, 111 = 3 * 37, 1001 = 7 * 11 * 13
- Jede positive Zahl > 1 hat einen Prim-Teiler (prime divisor, Nenner)
- Es gibt unendlich viele Primzahlen.
- Theorem: Ist n keine Primzahl, dann hat n einen Primfaktor, der kleiner als die Wurzel aus n ist.
- Das Theorem kann dazu benutzt werden, um alle Primzahlen zu finden, die kleiner oder gleich einer positiven Zahl n sind.

Unvollständigkeitstheorem

Gödel veröffentlichte 1931 sein

Unvollständigkeitstheorem,

welches beweist, dass es keinen Algorithmus gibt, der irgendeine Aussage über natürliche Zahlen aufnimmt und ausgibt, ob die Aussage wahr oder falsch ist.

Halteproblem

Das ***Halteproblem*** ist nicht entscheidbar.

Die Lösung des Halteproblems wäre ein Algorithmus, der für irgendein Programm testen kann, ob das Programm terminiert oder nicht.

Man kann beweisen, dass es einen solchen Algorithmus nicht gibt.

Eigenschaften eines Algorithmus

- Allgemeinheit
- Determiniertheit
- Determinismus
- Terminierung

Allgemeinheit und Determiniertheit

- **Allgemeinheit.** Ein Algorithmus löst im allgemeinen eine Klasse von Problemen. Die Auswahl des Einzelfalls erfolgt meist über Parameter.
- **Determiniertheit**. Algorithmen sind in der Regel determiniert; d.h. bei gleichen Eingabewerten und Startbedingungen laufen sie stets gleich ab und liefern dasselbe Ergebnis. Beispiele für nicht-determinierte Algorithmen sind z.B. stochastische Simulationen, die manche Entscheidungen dem Zufall überlassen.

Determinismus und Terminierung

- **Determinismus.** Ein Algorithmus heißt deterministisch, wenn zu jedem Zeitpunkt seiner Bearbeitung höchstens eine Möglichkeit der Fortsetzung besteht.
- **Terminierung.** In der Regel sind nur solche Algorithmen von Interesse, die für jede Eingabe nach endlich vielen Schritten terminieren, d.h. anhalten. Eine Ausnahme sind hier z.B. Betriebssystem-Funktionen und Prozess-Steuerungen.