2. Algorithmen und deren Darstellung

2.2 Algorithmen Definition

Ein Algorithmus ist eine Bearbeitungsvorschrift zur Lösung eines Problems, die die folgenden Eigenschaften hat:

a) Die Vorschrift ist mit endlichen Mitteln beschreibbar. **(Finitheit)**
Nach endlich vielen Schritten zu einem Ergebnis

b) Es ist immer eindeutig definiert, welche Regel in der Vorschrift als nächstes anzuwenden ist. **(Determinismus)**
Der Weg ist zu jedem Zeitpunkt fest vorgeschrieben. Es gibt zu jedem Zeitpunkt der Ausführung nur ***eine*** *Möglichkeit der Fortsetzung.*

c) Die Ausgabe ist immer eindeutig bestimmt **(Determiniertheit)**
Bei gleichen Startbedingungen immer das gleiche Ergebnis, dabei ist der Weg egal.

Zusätzlich wird man zumeist noch fordern:

d) Das Verfahren darf zu jedem Zeitpunkt nur angemessen viel Speicherplatz benötigen. **(Platzkomplexität)**

e) Das Verfahren sollte zu einem angemessenen Zeitpunkt terminieren. **(Zeitkomplexität, Terminiertheit)**

(wenn der Algorithmus deterministisch ist, determiniert er aufjedenfall auch)

Beispiel 2.7: Algorithmus: Binäre Suche

```
def binaereSuche(w, L) {
    first = 0
    last = L.size() - 1
    while (first <= last) {
      int middle = (first+last)/2
      if (L[middle] < w) {
          first = middle + 1
      }
      else {
          if (L[middle] == w) return true
          last = middle - 1
      }
    }
    return false
}
```

First=0, last=L.size-1 (first & last sind die Begrenzer)

Solange first <=last ist
wird auf middle = (first+last)/2 gespeichert
beim ersten Durchgang wäre er bei 7.
Wenn L<w
dann middle +1
Sonst

L==w ist dann True
Last wird -1

Die Suche beginnt nochmal. Nach dem ersten Durchgang entweder oberhalb oder unterhalb der 7. Es wird immer um die Hälfte geteilt und verglichen

L=[1,5,7,23,44,234]
w=44
println binaereSuche (w,L)

Ablaufbeschreibung detailliert.

Zuerst wird geschaut, ob die 44 genau in der Mitte steht, also an Position L_{middle}, wobei middle=(first+last)/2 ist. Dies ist somit die 7. Der Vergleich zeigt, die 44 ist größer als die 7. Deshalb setzt man den Begrenzer first mit middle+1. Die liste wird wieder halbiert und der wert verglichen.
Binär suchen heißt, die Liste immer wieder binär (also in zwei Hälften) zu teilen. Der Algorithmus wird damit immer kleinere Listen produzieren, in denen noch zu suchen ist.

Operation	Lineare Suche	Binäre Suche
Namen einfügen	Schnell	Langsam
Name suchen	Langsam	schnell

3. Grundlegende mathematische Definitionen

{a,b,c} = {a,c,b} ← Menge, dort ist es egal
[a,b,c] = [a,b,c] ← Liste, dort ist es nicht egal

Definition 3.5: kartesische Produkt

seien zwei Mengen M1, M2 gegeben. Wir bezeichnen mit M1 × M2 = {(x1, x2) | x1 ∈ M1 und x2 ∈ M2} das kartesische Produkt (oder Kreuzprodukt) beider Mengen.
x1 ist in M1 und x2 ist in M2
Bsp.: M1 = {1,2,3} und M2 = {11,13}. Dann erhält man das kartesische Produkt als M1 × M2 = {(1, 11), (1, 13), (2, 11), (2, 13), (3, 11), (3, 13)}

Definition 3.6: Zweistellige Relation

Sei eine Menge M gegeben. Jede Teilmenge R ⊆ M × M beschreibt eine **zweistellige Relation**.
Eine Relation ist damit eine Teilmenge des kartesischen Produktes einer Menge mit sich selbst. Damit setzt sie Elemente der Menge mit anderen Elementen in Beziehung.
Bsp.:

1. Gegeben sind $M = \{1,2,3,4\}$ und $R_1 = \{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$. R1 beschreibt die Beziehung *Kleiner* als. {X kleiner Y}
2. Gegeben sind M3 = {1, 2, 12, 17} und R_1 = {(1, 12), (2, 17)}. R_1 beschreibt die Beziehung 5 · x + 7 = y

- R ist **reflexiv**, falls für alle $x \in$ M gilt: xRx
- R ist **symmetrisch**, falls für alle $x, y \in$ M gilt. Wenn xRy, dann auch yRx.
- R ist **antisymmetrisch**, falls für alle $x, y \in$ M gilt. Wenn xRy und $x \neq y$, dann gilt niemals yRx. xRx ist erlaubt.
- R ist **asymmetrisch**, falls für alle $x, y \in$ M gilt. Wenn xRy, dann niemals yRx. xRy ist nicht erlaubt
- R ist **transitiv**, falls für alle $x, y, z \in$ M gilt. Wenn xRy **und** yRz, dann auch xRz.

Bsp.

1. Sei R1 = {(1, 2), (2, 1)} über M1 = {1, 2, 3}.

$R_1 = \quad x\ R\ y, y\ R\ x \rightarrow symmetrisch$

2. Sei R3 = {(1, 2), (2, 3), (2, 2), (1, 3)} über M1.
   $R_3 = xRy, xRx, yRz, xRz = antisymmetrisch, Reflexiv, transitiv$

- R heißt **Halbordnung** oder **partielle Ordnung**, falls R reflexiv, antisymmetrisch und transitiv ist.
- R heißt **Äquivalenzrelation**, falls R reflexiv, symmetrisch und transitiv ist.

1. Betrachte die Relation "x und y besitzen dieselbe ISB-Nummer" auf der Grundmenge aller bisher gedruckten Buchexemplare.
Welche Eigenschaften besitzt diese Relation?

"*reflexiv:*" Für jedes Buchexemplar x gilt: x und x besitzen dieselbe ISB-Nummer. Sprich: ein Buchexemplar hat immer dieselbe ISB-Nummer wie es selbst.

•

"*symmetrisch:*" Wenn x und y dieselbe ISB-Nummer besitzen, dann besitzen auch y und x dieselbe ISB-Nummer.

•

"*nicht antisymmetrisch:*" Es gibt mindestens zwei verschiedene Buchexemplare x und y, die dieselbe ISB-Nummer besitzen. Für diese beiden Exemplare steht zwar x in Relation zu y und y in Relation zu x, aber es ist $x \neq y$.

•

"*transitiv:*" Wenn die Buchexemplare x und y dieselbe ISB-Nummer besitzen und die Buchexemplare y und z dieselbe ISB-Nummer besitzen, dann besitzen auch x und z dieselbe ISB-Nummer.

Operationen

- $Op_1(a,b) = a + b$. Die Operation Op1 ist damit die Additionsoperation
- $Op_2(a,b) = a\%b$. Die Operation Op2 berechnet den Rest, der bei der Division übrigbleibt.
- $Op_3(a,b) = a$. Die sogenannte Projektion gibt immer den ersten Eingabewert zurück.
- $Op_4(a,b) = 4$. Die Operation Op4 gibt immer den konstanten Wert 4 zurück.

- Op heißt total, wenn für alle Kombinationen aus Eingabewerten ein Wert aus M definiert ist. Formal: ∀x1, x2 ∈ M ∃y ∈ M mit Op (x1, x2) = y.
  Für alle x1,x2 der Menge existiert y in der Menge mit Op(x1,x2)=y
- Op heißt Injektiv, wenn jede Kombination aus Eingabewerten einen verschiedenen Wert aus M liefert. Mit anderen Worten, jeder Wert aus M ist höchstens einmal das Ergebnis der Anwendung der Operation auf Ausgangswerte. Formal: ∀y ∈ M: |{(x1, x2) | x1, x2 ∈ M und Op (x1, x2) = y}| ≤ 1.
- Op heißt surjektiv, wenn jeder Wert aus M wenigstens einmal das Ergebnis der Anwendung der Operation auf Ausgangswerte ist. Formal: ∀y ∈ M: |{(x1, x2) | x1, x2 ∈ M und Op (x1, x2) = y}| ≥ 1.
- Op heißt bijektiv, wenn Op sowohl Injektiv als auch surjektiv ist. Formal: ∀y ∈ M: |{(x1, x2) | x1, x2 ∈ M und Op(x1, x2) = y}| = 1

∀x ∈ M: für alle x der Menge M

∃x ∈ M: es existiert ein x in der Menge M

Algebra Struktur Definition

Die Struktur $A = (M_1, \ldots, M_m, Op_1, \ldots, Op_n$ wird als **mehrsortige Algebra** bezeichnet. Die Algebra nennt man universell, wenn die Operationen wiederum nur Werte aus Mi liefern. Wenn m = 1, dann spricht man nur von einer **Algebra**.

Bsp.

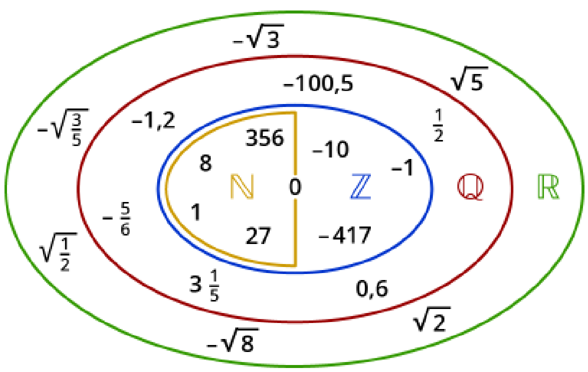
a) $(\mathbb{N}, -)$ ist keine universelle Algebra. $(\mathbb{N} - \mathbb{N} \neq \mathbb{N} \rightarrow 2 - 3 = -1)$ Die Operation der Subtraktion ist aber total.
b) $(\mathbb{Z}, \log())$ ist Algebra, aber nicht universell. Einige Logarithmen sind nicht ganzzahlig.
c) $(\mathbb{R}, \mathbb{N}, +, *)$ ist universell, mehrsortige Algebra. Die Addition ist total auf beiden Trägermengen total, die Subtraktion nicht auf den natürlichen Zahlen

4. Abstrakte Datentypen

4.1
createStack **schafft** einen **neuen** Keller
push **fügt** dem Keller ein Element **hinzu**
pop **entfernt** das oberste Elemente des Kellers
top lässt uns auf den Keller **schauen**
empty zeigt an, ob der Keller **leer** ist
Die *Axiome* sollen nun **widerspiegeln**, welchen Regeln wir diese **Operationen unterwerfen** sollen.

4.3
increment erhöhen
decrement vermindern
init initialisieren
get ausgeben
smaller kleiner als

$\mathbb{N} = \{1; 2; 3; 4 \dots\}$ Menge der natürlichen Zahlen
$\mathbb{N}_0 = \{0; 1; 2; 3 \dots\}$ Menge der natürlichen Zahlen auch die 0
$\mathbb{Z} = \{\dots - 2; -1; 0; 1; 2; \dots\}$ Menge der ganzen Zahlen
$\mathbb{Q} = \left\{0; \frac{1}{2}; -3\frac{3}{7}; \dots\right\}$ Menge der rationalen Zahlen. Sind sie Brüche aus ganzen Zahlen ohne das Wurzelziehen
$\mathbb{R} = \left\{\pi; e; \sqrt{2}; \sqrt{5}; \dots\right\}$ Menge der reellen Zahlen. Brüche sowie alle Wurzeln.



- Gerade Zahlen: 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24
- Ungerade Zahlen: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23

Die Primzahlen bis 100 lauten:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47 , 53, 59,
61, 67, 71, 73, 79, 83, 89, 97

Pseudocode:

```
def mergeSort(A) {
  if (A.size() <= 1) return A

  int pivot = (A.size() / 2)
  def A1 = []
  def A2 = []
```

```
if ((S1.size() > 0) && (S2.size() > 0)) {
  if (S1[0] < S2[0]) {
    Asort << S1[0]
    S1.remove(0)
  }
  else {
    Asort << S2[0]
    S2.remove(0)
```

```
def selection_2(A, k) {
  for (i = 1; i <= k; i++) {
    for (j = 0; j < A.size()-1; j++) {
      if (A[j] > A[j+1]) {
        temp = A[j]
        A[j] = A[j+1]
        A[j+1] = temp
      }
    }
  }
  return A[A.size()-k]
}
```