5.6 Exkurs: Codierung von Zeichen

Einzelne Zeichen werden durch Tabellen in natürliche Zahlen übersetzt:

ASCII-Code (8 Bit)

’A’ $= 65 = 0x41$	’B’ $= 66 = 0x42$	’C’ $= 67 = 0x43$
’a’ $= 97 = 0x61$	’b’ $= 98 = 0x62$	’c’ $= 99 = 0x63$
’0’ $= 48 = 0x30$	’1’ $= 49 = 0x31$	’2’ $= 50 = 0x32$
’␣’ $= 32 = 0x20$	’@’ $= 64 = 0x40$	

Unicode (UTF-8, variable Länge mit Blöcken zu je 8 Bit)

- 1 Byte: 0xxx xxxx = 7 Bit ASCII
- 2 Bytes: 110x xxxx 10xx xxxx = 11 Bit Codierung
- 3 Bytes: 1110 xxxx 10xx xxxx 10xx xxxx = 16 Bit Codierung
- ...

5.7 Exkurs: Rechnen mit Folgen und Reihen

Bemerkung. Obwohl der Computer endlich ist, lassen sich (gewisse) unendliche Mengen (und Funktionen auf unendliche Mengen etc.) explizit darstellen.

Beispiel.

```python
def a(n):
  return 1/n
```

ist die Codierung einer Folge reeller Zahlen (im Rahmen der Maschinengenauigkeit).

Bemerkung. Lambda-Funktionen kommen ursprünglich aus der funktionalen Programmierung. Eben genannte Folge kann man in Python auch schreiben als

```python
a = lambda n: 1/n
```

- Die Nutzung ist gleich
- Kein return-Statement

```python
print(list(map(a, [1,2,4,8])))
>>> [1, 0.5, 0.25, 0.125]
```

Addition:

```
def add(a, b):
  return lambda n: a(n) + b(n)
```

Subtraktion:

```
def sub(a,b):
  return lambda n: a(n) - b(n)
```

Reihen:

```
def finite_subsequence(a, k):
  return [a(n) for n in range(0, k)]

def series(a):
  return lambda n: sum(finite_subsequence(a, n))
```

5.8 Exkurs: Riemannsche Zeta-Funktion

Für komplexe Zahlen $s = \sigma \cdot \tau i \in \mathbb{C}$ mit $\sigma > 1$ konvergiert Dirichlet-Reihe

$$\zeta(s) = \sum_{n=0}^{\infty} \frac{1}{n^s} = 1 + \frac{1}{2^s} + \frac{1}{3^s} + \frac{1}{4^s} + \dots$$

Die Riemannsche Zeta-Funktion ist die eindeutige analytische Fortsetzung auf ganz $\mathbb{C}$. Die Riemannsche Vermutung ist: Alle Nullstellen (mit $\sigma > 0$) erfüllen $\sigma = \frac{1}{2}$.

```
def zeta(s):
  return series(lambda n: 1/(pow(n+1, z)))
```

Satz 5.1 (Riemannsches Umordnungsgesetz). *Es seien $a_n \in \mathbb{R}$ für $n \in \mathbb{N}$ und $\sum_{n=0}^{\infty} a_n$ sei konvergent, aber nicht absolut konvergent. Weiter sei $x \in \mathbb{R}$ beliebig. Dann existiert eine bijektive Abbildung $\phi : \mathbb{N} \to \mathbb{N}$ so, dass*

$$\sum_{n=0}^{\infty} a_{\phi}(n) = x$$

```
"""
  a is a series such that the series of partial sums converges, but not absolutely.
  x is the float to converge to for the reordering.
  returns first k indices of reordering bijection phi
"""
def riemann(a, x, k):
```

```
    partialsum = 0
    i = 0
    ipos, ineg = (0,0)
    reordering = []
    for j in range(0, k):
      if(partialsum < x):
        i = next_positive_index(a, ipos)
        ipos = i + 1
      else:
        i = next_negative_index(a, ineg)
        ineg = i + 1
      reordering.append(i)
      partialsums += a(i)
    return reordering

```

Listing 1: Riemann-Reordering

6 Graphen

Definition 6.1. Ein ungerichteter Graph G ist ein Tripel $G = (V, E, \Psi)$, wobei V und E endliche Mengen sind, $V \neq 0$ und

$$\Psi : E \to \{X \mid X \subseteq V, |X| = 2\}$$

Elemente in $V = V(G)$ heißen Knoten. Elemente in $E = E(G)$ heißen Kanten.

Definition 6.2. Ein gerichteter Graph (oder Digraph) G ist ein Tripel $G = (V, E, \Psi)$, wobei V und E endliche Mengen sind, $V \neq 0$ und

$$\Psi : E \to \{(v, w) \in V \times V \mid v \neq w\}$$

6.1 Terminologie für Graphen

Definition 6.3. Auf einem gerichteten oder ungerichteten Graphen verbindet die Kante $e = \{v, w\}$ bzw. $e = (v, w)$ die Knoten v und w. Dann heißen die Knoten v und w benachbart oder *adjazent*. Sie sind Endknoten der Kante e und mit dieser *inzident*.
Zwei Kanten $e, e' \in E$ heißen parallel, falls $\Psi(e) = \Psi(e')$.

Definition 6.4. Auf einem gerichteten oder ungerichteten Graphen $G = (V, E, \Psi), v \in V$

- Ist G ungerichtet, so ist
$$\delta(v) := \{e \in E \mid v \in \Psi(e)\}$$
die Menge der zu v inzidenten Kanten.

- Der Grad eines Knotens v ist $|\delta(v)|$.

- Ist G ungerichtet, so ist
$$\begin{aligned}\delta^+(v) &:= \{e \in E \mid \exists\, w \in V : \Psi(e) = (v, w)\}\\ \delta^-(v) &:= \{e \in E \mid \exists\, w \in V : \Psi(e) = (w, v)\}\\ \delta(v) &:= \delta^+(v) \cup \delta^-(v)\end{aligned}$$

- Der *Eingangsgrad* von v ist $|\delta^-(v)|$.
- Der *Ausgangsgrad* von v ist $|\delta^+(v)|$.

Definition 6.5. Ein gerichteter oder ungerichteter Graph ohne parallele Kanten wird *einfach* genannt.

- Für einfache Graphen identifiziert man $e \in E$ mit $\Psi(e)$ und schreibt
$$\begin{aligned} G = (V, E) \text{ mit } & E \subseteq \{\{v, w\} \mid v, w \in V, v \neq w\} \\ \text{bzw } & E \subseteq \{(v, w) \mid v, w \in V, v \neq w\} \end{aligned}$$
- Insbesondere ist dann die Kantenmenge E eines einfachen Graphen eine Relation auf V (d. h. auf $V \times V$).
- Die Kantenmenge E eines ungerichteten Graphen kann als symmetrische Relation R_E auf V aufgefasst werden:
$$(v, w) \in R_E :\Longleftrightarrow \{v, w\} \in E$$

6.2 Handschlaglemma

Lemma 6.6. *Es sei $G = (V, E, \Psi)$ ungerichteter oder gerichteter Graph.*

(a) Die Anzahl Knoten ungeraden Grades ist gerade, denn
$$\sum_{v\in V} |\delta(v)| = 2|E|$$

(b) Ist G gerichtet, dann gilt:
$$\sum_{v\in V} |\delta^+(v)| = |E| = \sum_{v\in V} |\delta^-(v)|$$

Beweis. (a)
$$\begin{aligned} \sum_{v\in V} |\delta(v)| &= \sum_{v\in V} \big|\{e \in E \mid \Psi(e) = \{v, w\} \text{ für ein } w \in V\}\big| \\ &= \sum_{e\in E} 2 = 2|E| \end{aligned}$$

(b)
$$\begin{aligned} \sum_{v\in V} |\delta^+(v)| &= \sum_{v\in V} |\{e \in E \mid \Psi(e) = (v, w) \text{ für ein } w \in V\}| \\ &= \sum_{e\in E} 1 = |E| = \sum_{v\in V} |\delta^-(v)| \end{aligned}$$

□

Definition 6.7. Ein ungerichteter Graph heißt regulär (oder d-regulär), falls jeder Knoten Grad d besitzt.

6.3 Teilgraphen, Wege, Kreise

Definition 6.8. Sei $G = (V(G), E(G), \Psi_G)$ ein ungerichteter oder gerichteter Graph.

(a) Graph $H = (V(H), E(H), \Psi_H)$ ist *Teilgraph* von G, falls

$$V(H) \subseteq V(G), E(H) \subseteq E(G), \Psi_H = \Psi_{G|E(H)}$$

(b) Ist zusätzlich $V(H) = V(G)$, so ist H *aufgespannter Teilgraph.*

(c) H ist induzierter Teilgraph, falls

$$\begin{aligned} E(H) &= \{e \in E(G) \mid \Psi_G(e) = \{v, w\} : v, w \in V(H)\} \\ \text{bzw. } E(H) &= \{e \in E(G) \mid \Psi_G(e) = (v, w) : v, w \in V(H)\} \end{aligned}$$

In diesem Falle schreiben wir auch $H = G[V(H)]$.