Übungsfragen 1 — Lösung — Thema A

(a)

$84 : 2 = 42$ R0
$42 : 2 = 21$ R0
$21 : 2 = 10$ R1
$10 : 2 = 5$ R0
$5 : 2 = 2$ R1
$2 : 2 = 1$ R0
$1 : 2 = 0$ R1

Reste von unten nach oben lesen: $84_{10} = 1010100_2$

(b)

$A_{16} = 10$, $B_{16} = 11$, $C_{16} = 12$, $D_{16} = 13$, $E_{16} = 14$, $F_{16} = 15$

$3E9C_{16} = 0011\,1110\,1001\,1100$

(c)

$2 = 0010_2 \quad \Rightarrow \quad \overline{2} = 1101_2 \quad \Rightarrow \quad -2 = \overline{2} + 1 = 1101_2 + 1_2 = 1110_2$

$5 = 0101_2$

	1110_2	-2
$+$	0101_2	5
$=$	$(1)0011_2$	3

Der Übertrag am Schluss kann ignoriert werden, da die Summe einer negativen 4-Bit-Zahl und einer positiven 4-Bit-Zahl keinen Überlauf erzeugen kann.

(d)

1	1	0	0	0	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Vorzeichen: $S = 1$

$$s = (-1)^S = -1$$

Exponent: $E = 100000010_2 = 128 + 2 = 130$

$$e = 130 - 127 = 3$$

Mantisse: $M = 1111$

$$m = 1.1111_2$$

Normalform: $s \cdot m \cdot 2^e = -1.1111_2 \cdot 2^3 = -1111.1_2 = -15.5$

Übungsfragen 1 **Lösung** **Thema B**

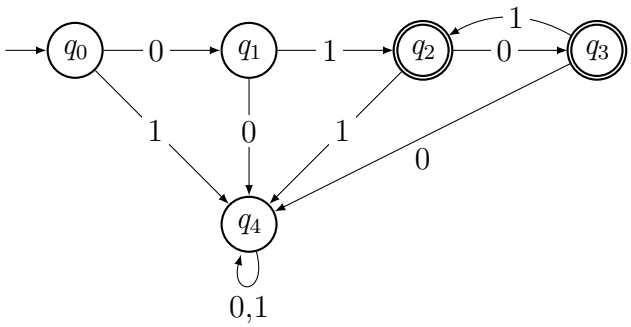
(a)

$\Sigma = \{A, B\}$

Länge	Wörter
0	ε (leeres Wort)
1	A, B
2	AA, AB, BA, BB
3	$AAA, AAB, ABA, BAA, ABB, BAB, BBA, BBB$

(b)

DFA über dem Alphabet $\Sigma = \{0, 1\}$, der alle Wörter w mit $|w| \geq 2$ akzeptiert, die mit einer Null beginnen und in denen kein Zeichen zweimal nacheinander vorkommt.



Übungsfragen 2 — Lösung — Thema A

(a)

Ein Algorithmus ist eine Vorschrift zur *Lösung eines Problems*, die *endlich*, *deterministisch* (eindeutig) und *effektiv* (ausführbar) ist.

(b)

$T(n) = C \cdot n^2$

$T(100) = C \cdot 100^2 = 20\text{s}$ $(*)$

$T(200) = C \cdot 200^2 = C \cdot (2 \cdot 100)^2 = 2^2 \cdot C \cdot 100^2 = 4 \cdot 20\text{s} = 80\text{s}$

Allgemein: In $O(n^2)$ bewirkt das Verdoppeln der Problemgrösse eine Vervierfachung der Laufzeit.

Man hätte auch die erste Gleichung nach C auflösen und diesen Wert in die zweite Gleichung einsetzen können. Meist lässt sich die Rechung jedoch in der oben beschriebenen Weise „kurzschliessen“.

(c)

```
def gcd(a, b):
    b = abs(b)
    while b > 0:
        a, b = b, a % b
    return a
```

(d)

```
def f(n):
    if n == 0:
        return 3
    else:
        return 2 + f(n-1)

print(f(4))
```

$$
\begin{aligned}
f(4) &= 2 + f(3) \\
&= 2 + (2 + f(2)) \\
&= 2 + (2 + (2 + f(1))) \\
&= 2 + (2 + (2 + (2 + f(0)))) \\
&\overset{*}{=} 2 + (2 + (2 + (2 + 3))) \\
&= 2 + (2 + (2 + 5)) \\
&= 2 + (2 + 7) \\
&= 2 + 9 \\
&= 11
\end{aligned}
$$

(a)

G	G	G	A	A	A	G	G	C	A	T	Vergleiche
G	G	C	A								3
	G	G	C	A							3
		G	G	C	A						2
			G	G	C	A					1
				G	G	C	A				1
					G	G	C	A			1
						G	G	C	A		4
Total											15

(b) $t =$ `aaaaaaa`, $p =$ `aab`

```
a a a a a a a
a a b
  a a b
    a a b
      a a b
        a a b
          a a b
```

Das Worst-Case-Muster der Länge $m = 3$ wird $n - m + 1 = 7 - 3 + 1 = 5$ mal erfolglos mit dem jeweiligen Textabschnitt verglichen.

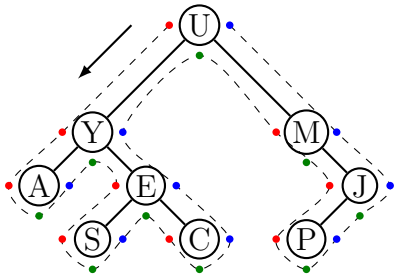
Somit gilt: $O((n - m + 1)m) = O(nm - m^2 + m)$

Gehen wir davon aus, dass der Text sehr viel länger als das Muster ist, ist, können wir die Summanden m und m^2 gegenüber n vernachlässigen und erhalten $O(nm)$.

(a)

```
class Stack:

    def __init__(self):
        self.items = []

    def push(self, item):
        self.items.append(item)

    def pop(self):
        return self.items.pop() # entfernt letztes Element

    def is_empty(self):
        return self.items == []

    def size(self):
        return len(self.items)
```

(b)



Preorder: U Y A E S C M J P

Postorder: A S C E Y P J M U

In jedem Fall wird der Baum rekursiv durchlaufen („Besuche immer zuerst den linken Ast und erst wenn das nicht möglich ist, besuche den rechten Ast.“). Bei der Preorder-Traversierung wird der Knoten verarbeitet (z. B. ausgegeben) *bevor* seine Kinder verarbeitet wurden; bei der Postorder-Traversierung wird er verarbeitet *nachdem* seine Kinder verarbeitet wurden. Bei Binärbäumen (Bäume, deren Knoten maximal zwei Kinder haben) gibt es auch noch eine Inorder-Traversierung, wo die Verarbeitung *zwischen* der Verarbeitung der beiden Äste erfolgt (grüne Punkte).

(c)

Die Breitensuche *Breadth-First-Search* (BFS) ist eine weitere Methode, um in einem Graphen von einem Startknoten aus jeden erreichbaren Knoten zu besuchen. Im Gegensatz zur Tiefensuche liefert der Algorithmus zusätzlich die kürzteste Wege zwischen dem Startknoten und den erreichbaren Knoten.

1. Wähle den Startknoten, markiere ihn als gesehen und füge ihn in eine Warteschlange ein.

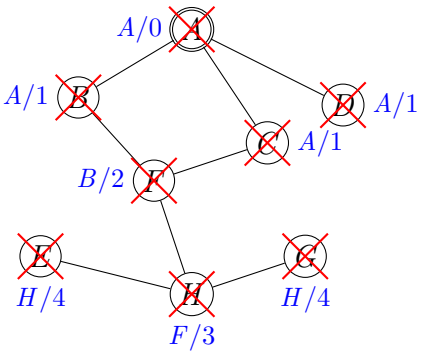
2. Entnimm einen Knoten vom Beginn der Warteschlange.

   (a) Falls das gesuchte Element gefunden wurde, brich die Suche ab und gib *gefunden* zurück.

   (b) Anderenfalls hänge alle bisher unmarkierten Nachbarn dieses Knotens ans Ende der Warteschlange an und markiere sie als gesehen.

3. Ist die Warteschlange leer, dann wurde jeder Knoten untersucht. Beende die Suche und gib *nicht gefunden* zurück.

4. Wiederhole Schritt 2.



Warteschlange

B C D
F

H
E G

Übungsfragen 3 — Lösung — Thema B

(a)

RLE, Steuerzeichen: `'!'`, Zählziffer: 4–9

unkomprimiert:	`AAAAAAABB!CCCCCCCCCCCCCCC`
komprimiert:	`A!7BB!0C!9C!7`

Kompressionsrate: $\frac{|\text{unkomprimierte Daten}|}{|\text{komprimierte Daten}|} = \frac{26}{13} = 2$

(b)

Unkomprimiert: `AAAAAAAAAABC`

Basistabelle: `A=0, B=1, C=2`

Substr	**Code**	**Tab+**
A	0	AA=3
AA	3	AAA=4
AAA	4	AAAA=5
AAAA	5	AAAAB=6
B	1	BC=7
C	2	--

Komprimiert: `0, 3, 4, 5, 1, 2`

Anzahl Bits unkomprimiert: $12 \cdot 2 = 24$

Anzahl Bits komprimiert: $6 \cdot 3 = 18$

Kompressionsrate: $24/18 = 4/3 = 1.33$

```
class Image:

    def __init__(self, width, height):
        self.w = width
        self.h = height
        self.img = [[0 for i in range(width)] for j in range(height)]

    def set_pixel(self, x, y, color=1):
        self.img[y][x] = color

    def write(self, filename):
        fd = open(filename, mode='w')
        fd.write('P1\n')
        fd.write('{0} {1}\n'.format(self.w, self.h))
        for i in range(0, self.h):
            for j in range(self.w):
                fd.write('{0} '.format(self.img[i][j]))
        fd.close()

I = Image(3, 3)
I.set_pixel(0, 0)
I.set_pixel(1, 1)
I.set_pixel(2, 2)
I.write('test.pbm')
```

(a) Das Modul definiert eine Klasse `Image`

- Der Konstruktor `__init__(self, width, height)` initialisiert die Attribute `self.w` und `self.h` mit den Parameterwerten `width` und `height` Dem Attribut `self.img` wird die Nullmatrix mit `height` Zeilen und `width` Spalten zugewiesen.
- Die Methode `set_pixel(...)` setzt in der Zeile `y` und der Kolonne `x` ein Pixel mit der „Farbe“ `color`. Wird dieser Parameter beim Aufruf der Methode weggelassen, so wird `1` verwendet.
- Die Methode `write(...)` öffnet eine neue Datei mit dem angegebenen Dateinamen, und schreibt die Bildinformationen (Bildformat, Dimension, Pixelwerte) in diese Datei.

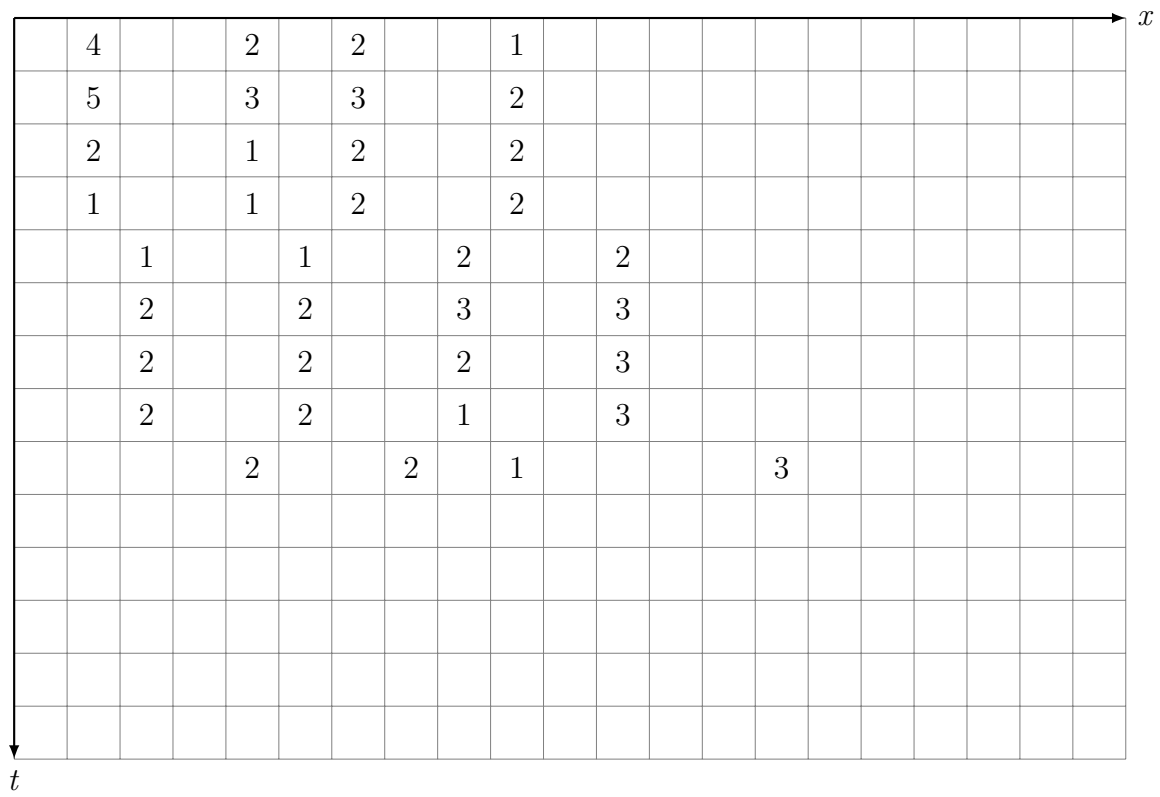
(b)
```
P1
3 3
1 0 0 0 1 0 0 0 1
```

Übungsfragen 4 **Lösung** **Thema B**

(a)

- *Typ:* Abbild (zur Analyse)
- *Verkürzung:*
  - Strasse: Reduktion auf eine Liste von Zellen
  - Auto: Reduktion auf seine Position und Geschwindigkeit
  - Fahrerin: Reduktion auf Beschleunigen, Bremsen und Trödeln
- *Zweck:* Untersuchung des Verkehrsflusses

(b)



Übungsfragen 5 **Lösung** **Thema A**

(a)

```
def selectionsort(L):
    n = len(A)
    for i in range(0,n-1):
        minpos=i
        for j in range(i+1,n):
            if A[j]<A[minpos]:
                minpos=j
        A[minpos],A[i]=A[i],A[minpos]
```

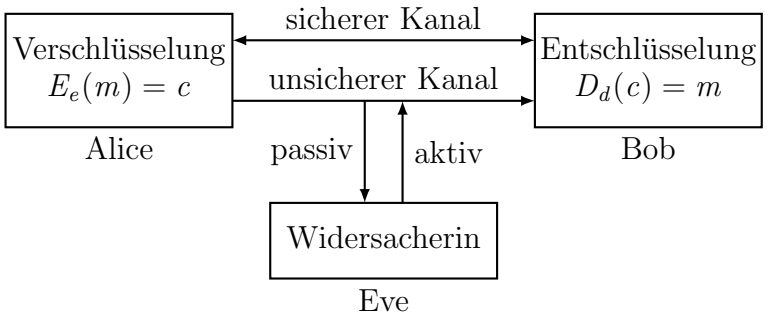
(b)

Insertionsort

					Vergleiche	Verschiebungen
7	9	1	3	5	1	0
7	9	1	3	5	2	3
1	7	9	3	5	3	3
1	3	7	9	5	3	3
1	3	5	7	9		

Übungsfragen 5 **Lösung** **Thema B**

(a)



E Encryption function
D Decryption function

e Dncryption key
d Decryption key

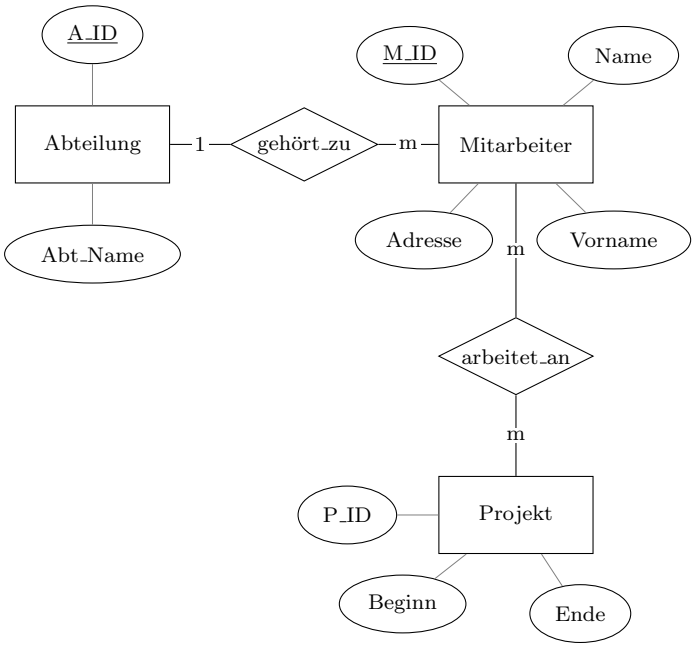
m Message (Klartext)
c Ciphertext (Geheimtext)

(b)

1. $p = 19$; $g = 10$ ist ein Generator von $\mathbb{Z}_{19}^*$
2. (a) Alice wählt $a = 2$, berechnet $A = 10^2 \bmod 19 = 5$ und sendet $A = 5$ an Bob.
   (b) Bob wählt $b = 16$, berechnet damit
   $B = 10^{16} \bmod 19$
   $10^2 \bmod 19 = 5$
   $10^4 \bmod 19 = (10^2)^2 \bmod 19 = 5^2 \bmod 19 = 6$
   $10^8 \bmod 19 = (10^4)^2 \bmod 19 = 6^2 \bmod 19 = 17$
   $10^{16} \bmod 19 = (10^8)^2 \bmod 19 = 17^2 \bmod 19 = 11$
   und sendet B an Alice.
3. (a) Alice berechnet $K_a = B^a = 11^2 \bmod 19 = 7$
   (b) Bob berechnet $K_b = A^b = 5^{16} \bmod 19 = 16$
4. Der Schlüssel lautet $K = 11$.

Übungsfragen 6 Lösung Thema A

(a)



1 bedeutet *genau ein*; c bedeutet *kein oder ein* (*can*); m bedeutet *mindestens ein*

Statt $m : n$ wird hier die Schreibweise $m : m$ verwendet.

Hier werden die Kardinalitäten vor der Zielentität einer Beziehung gesetzt. Konsequent umgekehrt ist auch richtig.

(b)

(a)

AVG(a)
2.0

(b)

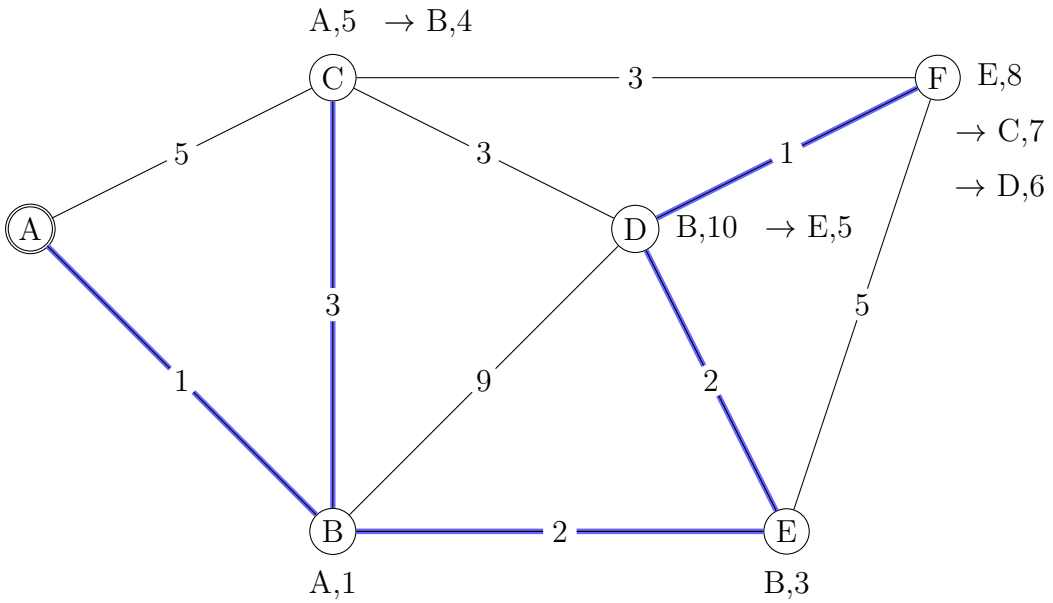
b	c
3.2	t
4.1	s
6.5	r

(c)

SUM(a)	c
4	r
1	s
5	t

Übungsfragen 6 **Lösung** **Thema B**

(a)



(b)

- in Routenplanern
- im Internet für das Routing der Datenpakete. Mit *Routing* wird hier das Festlegen von Wegen für Datenströme in Computernetzwerken bezeichnet.