

## Präsenzaufgaben 2: Graphen

**31.03.2025**

Die Lösung der Aufgaben wird am Ende von Ihnen vorgestellt.

### Aufgabe 1:

#### Graphen.

- a) Sei  $\Sigma_1 = \{a, b, c\} \wedge L_1 = \{w \in \Sigma_1^* \mid |w|_a = |w|_b = \left\lfloor \frac{|w|_c}{3} \right\rfloor\}$ . Welche der folgenden Worte sind Teil der Sprache  $L_1$ ? Falls nein, begründen Sie kurz Ihre Antwort.
- ab
  - ccacbc
  - cc
  - acacacacaccccccbbcbcbcc
  - $c(acbc)^*c$
- b) Sei  $\Sigma_2 = \{a, b, c, d\} \wedge L_2 = \{w \in \Sigma_2^* \mid |w| < 5\}$ . Wie viele Wörter enthält  $L_2$ ?
- c) Sei  $L_3 = \{w \in \Sigma_2^* \mid \forall i \in \Sigma_2 : |w|_i = 1 \wedge |w| = 4\}$ . Wie viele Wörter enthält  $L_3$ ?
- d) Sei  $L_4 = \{w \in \Sigma_2^* \mid \exists i \in \Sigma_2 : |w|_i \geq 2 \wedge |w| = 4\}$ . Wie viele Wörter enthält  $L_4$ ?
- e) Zeichnen Sie den ungerichteten Graphen

$$G = (\{v_1, v_2, v_3, v_4\}, \{(v_1, v_2), (v_1, v_3), (v_3, v_4), (v_3, v_2)\})$$

und geben Sie die dazugehörige Adjazenzmatrix an.

- f) Zeichnen Sie einen vollständigen, ungerichteten und gewichteten Graphen mit  $|V| = 4$  ohne self-loops. Beschriften Sie die Knoten mit  $v_i$ , wobei  $i \in \{1, 2, 3, 4\}$ . Die Gewichtsfunktion ist folgendermaßen definiert:

$$\phi((v_i, v_j)) := \min(i, j)$$

- g) Zeichnen Sie den ungewichteten und gerichteten Graphen, der durch folgende Adjazenzmatrix beschrieben ist:

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

### Aufgabe 2:

#### Graphen.

Wir möchten eine Klasse für gerichtete Graphen programmieren.

- a) Dafür benötigen wir die Klassen Edge und Vertex.
- Ein Vertex/Knoten hat eine id, die als Integer gespeichert wird. Mit der `toString`-Methode soll die id ausgegeben werden. Zudem muss Vertex das Interface Comparable implementieren, um Knoten miteinander vergleichen zu können. Die Ordnung von vertices/Knoten ergibt sich über deren id.

- Eine Edge/Kante hat einen start- und end-vertex/Knoten. Fügen Sie zusätzlich geeignete Konstruktoren hinzu.

Zudem müssen Sie die Methode equals(Object o) überschreiben. Es soll sichergestellt werden, dass zwei Kanten gleich sind, wenn start und ende gleich sind. Außerdem müssen Sie noch folgende Methode hinzufügen:

Listing 3.1: hashCode in Klasse Edge

```
@Override
public int hashCode() {
    return Objects.hash(start, end);
}
```

- Die Klasse Graph soll als Attribute zwei Mengen haben. Die Menge der vertices/-Knoten und die Menge der edges/Kanten. Um in Java Mengen abzubilden, gibt es das Interface **Set**. Nutzen Sie die Implementation **HashSet**. Fügen Sie zusätzlich einen geeigneten Konstruktor hinzu.
- Implementieren Sie die `toString`-Methode, die den Graphen als Adjazenzmatrix ausgibt. Dieser Teil von `main`:

Listing 3.2: main in Klasse Graph

```
Vertex v2 = new Vertex(2);
Vertex v1 = new Vertex(1);
HashSet<Vertex> tempV = new HashSet<Vertex>();
tempV.add(v2);
tempV.add(v1);

Edge e1 = new Edge(v1,v2);
HashSet<Edge> tempE = new HashSet<Edge>();
tempE.add(e1);

Graph test = new Graph(tempV,tempE);
test.toString();
```

soll zu folgender Ausgabe führen:

```
1: 0 1
2: 0 0
```