Hausaufgaben 2: Graphen 31.03.2025

Abgabe der Lösung am 06.04.2025

Aufgabe 1:

Graphen.

a) Sei $\Sigma_1 = \{a, b, c\} \wedge L_1 = \{w \in \Sigma_1^* | \quad |w|_a + |w|_b = \left\lfloor \frac{|w|_c}{2} \right\rfloor\}$. Welche der folgenden Worte sind Teil der Sprache L_1? Falls nein, begründen Sie kurz Ihre Antwort.

- $cabc$
- $ccacbcc$
- cc
- $acacacacacccccccbcbcbcbcc$
- $c(acccbc)^*cca$

b) Sei $\Sigma_2 = \{a, b, c, d\} \wedge L_2 = \{w \in \Sigma_2^* \quad | \quad |w| < 5 \wedge |w|_a = 1\}$. Wie viele Wörter enthält L_2?

c) Sei $L_3 = \{w \in \Sigma_2^* \quad | \quad |w|_a = 0 \wedge |w| < 5\}$. Wie viele Wörter enthält L_3?

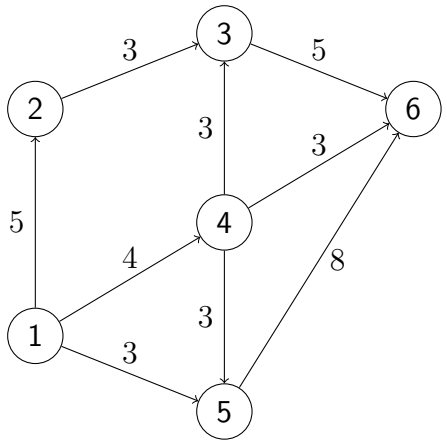
d) Sei $L_4 = \{w \in \Sigma_2^* \quad | \quad |w|_a \geq 1 \wedge |w| = 4\}$. Wie viele Wörter enthält L_4?

e) Zeichnen Sie den gerichteten Graphen

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

und geben Sie die dazugehörige Adjazenzmatrix an.

f) Geben Sie für den folgenden Graphen die Adjazenzmatrix an:



g) Zeichnen Sie den ungerichteten Graphen, der durch folgende Adjazenzmatrix beschrieben ist:

$$\begin{pmatrix} 0 & 3 & 1 & 0 & 0 \\ 3 & 0 & 4 & 0 & 0 \\ 1 & 4 & 0 & 7 & 0 \\ 0 & 0 & 7 & 0 & 1 \\ 0 & 0 & 0 & 1 & 9 \end{pmatrix}$$

Aufgabe 2:

Graphen. Wir möchten unsere Klasse für gerichtete Graphen erweitern, um DEAs umzusetzen. Dafür müssen wir die Klassen Edge und Graph erweitern.

a) Die Klasse Transistion soll die Klasse Edge erweitern. Es muss ein HashSet hinzugefügt werden, in dem die möglichen Symbole, die eine Transition ermöglichen, gespeichert sind.

b) Die Klasse DEA soll die Klasse Graph erweitern. Dafür muss die Oberklasse Graph einen Default-Konstruktor erhalten. Der Konstruktor für die Klasse DEA, soll folgenden Header haben:

```
public DEA(HashSet<Vertex> vertices,
        HashSet<Character> alphabet,
        HashSet<Transition> edges,
        Vertex start,
        HashSet<Vertex> finalStates)
```

c) Implementieren Sie die private-Methode correctDEA(). Diese soll überprüfen, ob für jedes Paar aus Knoten/vertex und Buchstaben aus Σ genau eine Kante existiert. Ist dies nicht der Fall, soll ein Fehler geworfen werden. Diese Methode soll dann am Ende des Konstruktors aufgerufen werden. Hinweis: Die Methoden equals und hashCode müssen für die Klasse Transistion angepasst werden.

d) Implementieren Sie nun eine boolean containsWord(String w) Methode, die überprüft, ob der Automat in einem Endzustand landet.

Listing 3.3: `main` in Klasse `DEA`

```
public static void main(String[] args) {

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

  HashSet<Character> temp1 = new HashSet<Character>();
  temp1.add('a');
  HashSet<Character> temp2 = new HashSet<Character>();
  temp2.add('b');
  HashSet<Character> temp3 = new HashSet<Character>();
  temp3.add('a');
  temp3.add('b');
  HashSet<Character> alphabet = new HashSet<Character>();
  alphabet.add('a');
  alphabet.add('b');

  Transition t1 = new Transition(v1,v2, temp1);
  Transition t2 = new Transition(v1,v1, temp2);
  Transition t3 = new Transition(v2,v2, temp3);
  Transition tFALSE = new Transition(v1,v2, temp3);

  HashSet<Transition> tempT = new HashSet<Transition>();
  tempT.add(t1);
  tempT.add(t2);
  tempT.add(t3);
  //tempT.add(tFALSE);
```

```
    HashSet<Vertex> finalStates = new HashSet<Vertex>();
    finalStates.add(v2);
    DEA test = new DEA(tempV,alphabet,tempT,v1,finalStates);

    System.out.println(test.containsWord("baabaabab"));
    System.out.println(test.containsWord("bb"));
    System.out.println(test.containsWord("a"));
  }
```