Algorithmen und Datenstrukturen
Übungszettel 10

Martin Avanzini <martin.avanzini@uibk.ac.at>
Thomas Bauereiß <thomas.bauereiss@uibk.ac.at>
Herbert Jordan <herbert@dps.uibk.ac.at>
René Thiemann <rene.thiemann@uibk.ac.at>

31. Mai, zur Besprechung am 7. Juni

Aufgabe 1) Klassifizierung von Kanten In der Vorlesung wurde die Klassifizierung von Kanten eines Graphen als Baumkanten, Rückwärtskanten, Vorwärtskanten oder Querkanten vorgestellt. Implementieren Sie einen Algorithmus, der diese Klassifizierung für alle Kanten eines gegebenen Graphen durchführt.

1. Implementieren Sie das Interface `EdgeClassifier`. Die Methode `classifyEdges` soll für einen gegebenen Graphen `g` eine Liste von klassifizierten Kanten in Form von Instanzen der Klasse `EdgeClassification` zurückliefern. Greifen Sie bei Ihrer Implementierung auf die Methoden des Interfaces `Graph` aus dem vorherigen Übungsblatt zurück.

```
1  public interface EdgeClassifier {
2
3      /**
4       * Klassifiziert die Kanten eines gerichteten Graphen.
5       *
6       * @param  g  ein gerichteter Graph
7       * @return eine Liste von klassifizierten Kanten
8       */
9      public List<EdgeClassification> classifyEdges(Graph g);
10
11 }
```

```
1  public class EdgeClassification {
2
3      /** Konstanten fuer Typen von Kanten. Kann von aussen z.B. per
4       * EdgeClassification.Type.TREE verwendet werden. */
5      enum Type { TREE, BACK, FORWARD, CROSS };
6
7      /** Quellknoten, aus dem diese Kante herausfuehrt. */
8      int source;
9      /** Zielknoten, in den diese Kante hineinfuehrt. */
10     int target;
11     /** Typ dieser Kante */
12     Type type;
```

```
13
14      public EdgeClassification(int source, int target, Type type) {
15          this.source = source;
16          this.target = target;
17          this.type = type;
18      }
19  }
```

2. Testen Sie Ihre Implementierung mit Hilfe der Klasse `ClassifierTest`, die den Beispielgraphen von Folie 329 enthält. Setzen Sie dafür in Zeile 7 von `ClassifierTest.java` den Klassennamen Ihrer Implementierung ein. Testen Sie Ihre Implementierung auch mit anderen Graphen.

Aufgabe 2) Starke Zusammenhangskomponenten (verschoben auf Blatt 11)

1. Geben Sie ein Beispiel für einen Graphen mit mindestens zwei starken Zusammenhangskomponenten.

2. Wenden Sie den Algorithmus von Kosarajus zum Bestimmen von starken Zusammenhangskomponenten auf den Graph in Abbildung 1 an. Geben Sie die gefundenen starken Zusammenhangskomponenten und die Zwischenergebnisse (Endzeiten) der beiden Tiefensuchen an.

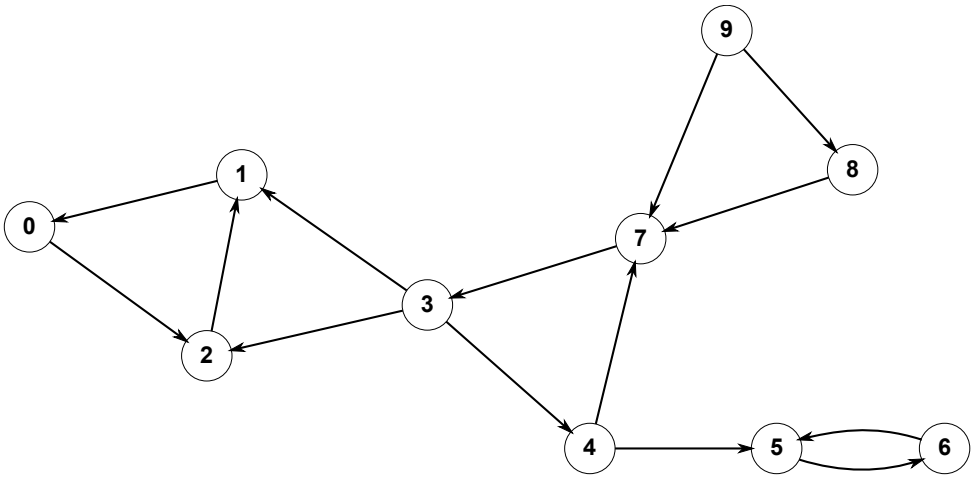

Abbildung 1: Graph