Aufgabe 1
„Superstar“- Dokumentation

37. Bundeswettbewerb Informatik 2018/19 - 1. Runde

Sebastian Baron, Simon Fiebich, Lukas Rost

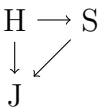
Team-ID: 00036

26. November 2018

Inhaltsverzeichnis

1 Lösungsidee

Zunächst bietet es sich an, die Eingabe in einen Graphen umzuwandeln. Die Knoten entsprechen dabei den TeeniGram-Mitgliedern. Eine gerichtete Kante verläuft von einem Knoten x zu einem Knoten y genau dann, wenn x y im TeeniGram-Netzwerk folgt. Für das in der Aufgabenstellung gegebene Beispiel ergibt sich folgender Graph:



Literatur

[1] Wikipedia-Artikel zur Tiefensuche, `https://de.wikipedia.org/wiki/Tiefensuche`

[2] Steven S. Skiena: The Algorithm Design Manual, ISBN 978-1-84800-069-8

2 Umsetzung

3 Beispiele

4 Quellcode

```
1  package de.lukasrost.bwinf2019.superstar;
2
3  import javax.swing.*;
4  import java.io.BufferedReader;
5  import java.io.File;
6  import java.io.FileReader;
7  import java.io.IOException;
8  import java.util.ArrayList;
9  import java.util.HashMap;
10
11 class SuperstarHelper {
12     private File inputFile;
13     private ArrayList<Vertex> vertices = new ArrayList<>();
14     private HashMap<String,Vertex> nameToVertex = new HashMap<>();
15     private Graph graph;
16     private String superStar;
17
18     void showFileSelectionWindow(){
19         //Benutzerauswahl der einzulesenden Datei und Umwandlung in ein
           ↪  File-Objekt
20         JFileChooser chooser = new JFileChooser();
21         File file = null;
22         int rueckgabeWert = chooser.showOpenDialog(null);
23         if (rueckgabeWert == JFileChooser.APPROVE_OPTION){
24             file = chooser.getSelectedFile();
```

```
25          } else {
26              System.exit(0);
27          }
28          inputFile = file;
29      }
30
31      void readToGraph(){
32          try (BufferedReader br = new BufferedReader(new
            ↪  FileReader(inputFile)))
33          {
34              boolean first = true;
35              for (String line; (line = br.readLine()) != null;)
36              {
37                  if (first){
38                      first = false;
39                      for (String name : line.split(" ")){
40                          Vertex v = new Vertex(name);
41                          vertices.add(v);
42                          nameToVertex.put(name,v);
43                      }
44                  } else {
45                      String[] edge = line.split(" ");
46                      nameToVertex.get(edge[0]).
                        ↪  addAllToAdjacency(nameToVertex.get(edge[1]));
47                  }
48              }
49          } catch (IOException e) {
50              e.printStackTrace();
51          }
52          graph = new Graph(vertices.toArray(new Vertex[0]));
53      }
54
55      void generateSolution(){
56          superStar = graph.modifiedDFS();
57      }
58
59      String getOutput(){
60          if (!"".equals(superStar)){
61              return "Superstar ist " + superStar + ".\nAnzahl der Anfragen: " +
                ↪  graph.getAnfrageCounter();
62          } else {
63              return "Es gibt keinen Superstar.\nAnzahl der Anfragen: " +
                ↪  graph.getAnfrageCounter();
64          }
65      }
66  }
```

Quellcode 1: Ein- und Ausgabe: `SuperstarHelper.java`

```
1  package de.lukasrost.bwinf2019.superstar;
2
3  import java.util.ArrayList;
4  import java.util.Arrays;
```

```
5
6   class Graph {
7       private ArrayList<Vertex> vertices = new ArrayList<>();
8       private Vertex current;
9
10      private int anfrageCounter = 0;
11
12      Graph(Vertex... nodes1){
13          vertices.addAll(Arrays.asList(nodes1));
14          current = vertices.get(0);
15      }
16
17      int getAnfrageCounter() {
18          return anfrageCounter;
19      }
20
21      private boolean hasEdge(Vertex start, Vertex end){
22          anfrageCounter++;
23          return start.getAdjacency().contains(end);
24      }
25
26      String modifiedDFS(){
27          return modifiedDFS(current,new ArrayList<>(),null);
28      }
29
30      private String modifiedDFS(Vertex start, ArrayList<Vertex> visited, Vertex
        ↪  parent){
31          visited.add(start);
32          Vertex vt = null;
33
34          for (Vertex vertex : vertices) {
35              if (!vertex.equals(start) && !visited.contains(vertex) &&
                ↪  hasEdge(start,vertex)){
36                  vt = vertex;
37                  break;
38              }
39          }
40
41          if (vt != null){
42              return modifiedDFS(vt,visited,start);
43          } else {
44              for (Vertex vertex : vertices){
45                  if (!vertex.equals(start) && !vertex.equals(parent) &&
                    ↪  !hasEdge(vertex,start)){
46                      return "";
47                  }
48              }
49              return start.getContent();
50          }
51      }
52  }
```

Quellcode 2: Implementierung des ADT Graph: `Graph.java`