Datenbanken, Hands-on 3

Henry Haustein

Aufgabe 1

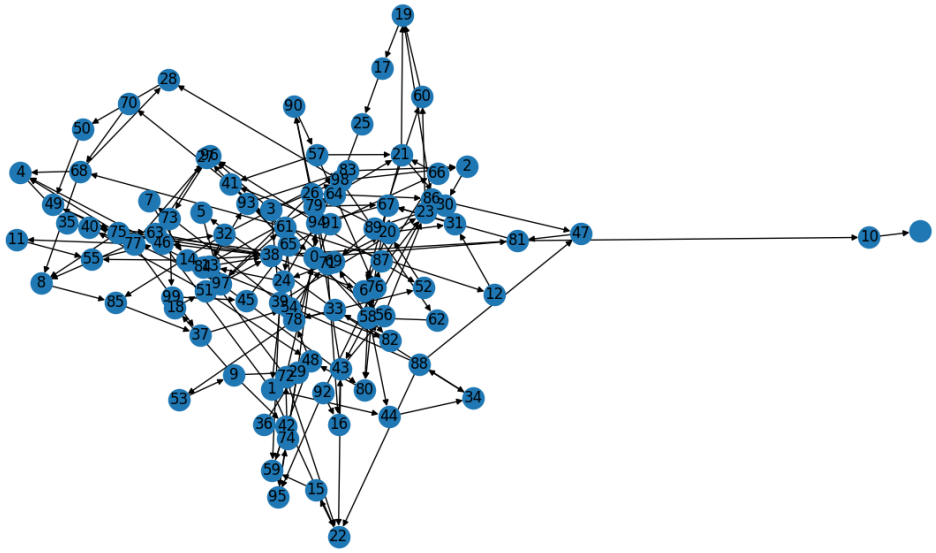
Ich habe ein Python-Skript geschrieben, was die Aufgaben löst. Dafür habe ich die Bibliothek `networkx` für die Graphen-Implementation und `matplotlib` zum Zeichnen des Graphen benutzt.

```
1  import networkx as nx
2  import matplotlib.pyplot as plt
3
4  filename = "graph_17.txt"
5
6  def reachableNodes(graph, start, exploredNodes):
7    for vertex in graph.neighbors(start):
8      if vertex not in exploredNodes:
9        exploredNodes.append(vertex)
10       moreVertices = reachableNodes(graph, vertex, exploredNodes)
11       exploredNodes.extend(moreVertices)
12   # gleich Duplikate herausfiltern um Rechenleistung und RAM zu
         sparen
13   return list(set(exploredNodes))
14
15 def getNodesPartOfCycle(graph):
16   nodesPartOfCycle = []
17   for vertex in G.nodes:
18     if vertex not in nodesPartOfCycle:
19       if vertex in reachableNodes(graph, vertex, []):
20         nodesPartOfCycle.append(vertex)
21   return nodesPartOfCycle
22
23 # Zuordnung zwischen einem Knoten und dem Zeitstempel
24 verticesWithTimestamp = {}
25
26 print("Start Einlesen der Kanten")
27 with open(filename) as f:
28   lines = f.read().splitlines()
29   for line in lines:
30     if line == "<<objects>>":
31       # Start der Liste an Knoten
32       numberVertices = 0
33       continue
34     if line == "<<edges>>":
```

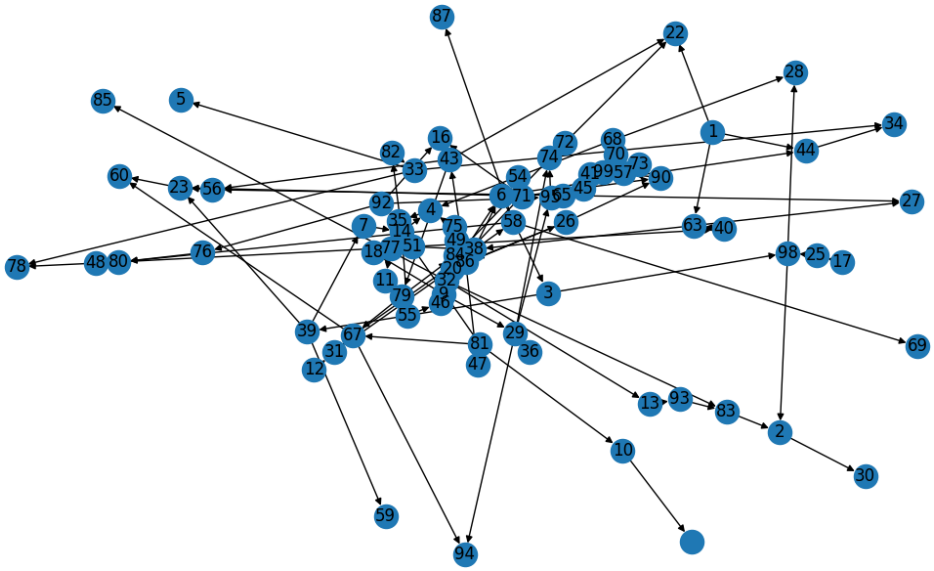
```
35        # Ende der Liste an Knoten
36        break
37      numberVertices = numberVertices + 1
38      id = line.split("\t")[0]
39      timestamp = line.split("\t")[1]
40      verticesWithTimestamp.update({id: timestamp})
41
42
43  G = nx.DiGraph()
44  print("Start Hinzufuegen der Knoten zum Graphen")
45  G.add_nodes_from(list(verticesWithTimestamp.keys()))
46
47  print("Start Hinzufuegen der Kanten zum Graphen")
48  # Datei wird zeilenweise eingelesen, aber der fuer uns wichtige
        Inhalt kommt erst, wenn "<<edges>>" einmal aufgetaucht ist
49  processEdges = False
50  with open(filename) as f:
51    lines = f.read().splitlines()
52    for line in lines:
53      if processEdges:
54        try:
55          start = line.split("\t")[0]
56          enden = line.split("\t")[1].split(",")
57          for ende in enden:
58            G.add_edge(start, ende)
59        except Exception as e:
60          # falls ein Knoten mal keine Enden hat, wird die Zeile
                nicht bearbeitet - ist ja auch nicht notwendig
61          pass
62      if line == "<<edges>>":
63        processEdges = True
64
65
66  # Zeichnen des Graphen
67  plt.box(False)
68  nx.draw_networkx(G)
69  plt.show()
70
71  removedNodes = []
72
73  while True:
74    nodesPartOfCycle = getNodesPartOfCycle(G)
75    NodesWithTimestamps = {}
76    if len(nodesPartOfCycle) > 0:
77      print("========================= Zyklus gefunden
          =========================")
78      # es gibt Zyklen
79      print("Aktuelle Knoten, die Teil eines Zyklus sind: " + str(
          nodesPartOfCycle))
80      for elem in nodesPartOfCycle:
```

```
81          # Zu jedem Knoten in einem Zyklus suchen wir uns den
                passenden Timestamp
82          NodesWithTimestamps.update({elem: verticesWithTimestamp[elem
                ]})
83        # Sortieren nach aufsteigendem Timestamp
84        NodesWithTimestamps = dict(sorted(NodesWithTimestamps.items(),
              key=lambda item: item[1]))
85        print("Sortierte Liste an Knoten mit ihrem Zeitstempel: " +
              str(NodesWithTimestamps))
86        # letztes Element ist der juengste Knoten
87        youngestNode = list(NodesWithTimestamps)[-1]
88        print("Juengster Knoten: " + str(youngestNode))
89        removedNodes.append(youngestNode)
90        G.remove_node(youngestNode)
91      else:
92        # es gibt keine Zyklen mehr -> Programm ist fertig
93        print("========================== Programm fertig
              ==========================")
94        print("Sortierte Liste an entfernten Knoten: " + str(sorted([
              int(node) for node in removedNodes])))
95        plt.box(False)
96        nx.draw_networkx(G)
97        plt.show()
98        break
```

In Zeile 4 muss der Dateiname der Datei mit der Repräsentation des Graphen eingetragen werden. Unsere Gruppe hatten den Graphen 17 zu bearbeiten, der Graph sieht vor der Bearbeitung so aus:



Und so nach dem Algorithmus:



Insgesamt wurden die folgenden Knoten entfernt (hier als Listenrepräsentation): `[0, 8, 15, 19, 21, 24, 37, 42, 50, 52, 53, 61, 62, 64, 66, 88, 89, 91, 96, 97]`