Aufgabe 1: Zimmerbelegung

Team: BANDO
Einsendenummer: 00222

1. November 2017

Inhaltsverzeichnis

1 Lösungsidee

Eine Dame soll als Knoten in einem ungerichtetem Graphen dargestellt werden. Zwei Damen, die im gleichem Zimmer sein müssen, da es sich mindestens eine von ihnen wünscht, werden mit einer Kante der Gewichtung 1 miteinander verbunden. Analog werden zwei Damen mit einer Kante der Gewichtung -1 miteinander verbunden, wenn sich mindestens eine von ihnen wünscht nicht mit der anderen ein Zimmer zu teilen.

Eine Zimmerbelegung existiert nun genau dann nicht, wenn mindestens ein Paar aus zwei Knoten existiert, die sowohl mit einem Pfad aus ausschließlich Kanten der Gewichtung 1 miteinander verbunden sind als auch mit einer Kante der Gewichtung -1 direkt miteinander verbunden sind.

Für den Fall, dass eine gültige Zimmerbelegung existiert, sollen sich zwei Damen genau dann das gleiche Zimmer teilen, wenn diese durch einen Pfad aus ausschließlich Kanten der Gewichtung 1 miteinander verbunden sind.

2 Umsetzung

Bei der Bestimmung eines Ergebnisses ist es am pragmatischsten anzunehmen, dass eine Zimmerbelegung existiert, und sie nach den obigen Vorgaben zu generieren. Ist die Annahme falsch, so wird das Programm unweigerlich auf einen Widerspruch stoßen, was das Ergebnis 'Zimmerbelegung nicht möglich.' zurückwirft.

Das Programm beginnt, indem es eine Dame einer Liste anfügt. Alle Damen, die mit einer Kante der Gewichtung 1 mit dieser verbunden sind, werden nun ebenfalls dieser Liste angefügt. Für die neu angefügten Damen wiederhole man diesen Vorgang, bis keine anzufügende Dame mehr existiert. Die Liste beschreibt nun die Damen, die sich einen Raum teilen. Sollte es noch Damen geben, die noch nicht einer Liste angefügt worden wurden, so nehme man einer dieser und füge diese einer neuen Liste an. Diesen Vorgang wiederhole man nun, bis alle Damen einer Liste angefügt worden sind.

Die dadurch generierte Zimmerbelegung soll nun auf die am Anfang genannte Widerspruchsfreiheit überprüft werden. Hierfür wird untersucht, ob zwei Damen, die mit einer Kante der Gewichtung -1 miteinander verbunden sind, Elemente der gleichen Liste sind. Trifft dies zu, so ist ein Widerspruch mit dem enstprechdem Ergebnis gefunden.

Es ist anzumerken, dass die graphentheoretische Lösung nicht mit Hilfe einer Adjazenzmatrix realisiert wurde, sondern mit dem iterativem Durchlaufen der im Form einer Liste gespeicherten Wünsche der Damen. Das Prinzip und die mathematische Richtigkeit bleiben jedoch unverändert.

3 Beispiele

3.1 zimmerbelegung1.txt

Output:
"Keine Zimmerbelegung möglich."

Der Grund hierfür ist, dass Anna mit Paula ein Zimmer teilen möchte, Paula jedoch nicht.

3.2 zimmerbelegung2.txt

Output:
('Alina', 'Lilli'),
('Emma', 'Mia', 'Zoe'),
('Lara')

3.3 zimmerbelegung3.txt

Output:
('Anna', 'Julia', 'Lisa', 'Emily', 'Sofia', 'Marie', 'Jana', 'Johanna', 'Carolin', 'Michelle', 'Emma', 'Nele', 'Antonia', 'Larissa'),
('Hannah'),
('Lea', 'Celine', 'Clara', 'Lena', 'Lina'),
('Sarah', 'Sophie', 'Alina', 'Josephine', 'Leonie', 'Lilli', 'Vanessa', 'Annika', 'Pia', 'Melina', 'Kim', 'Pauline', 'Katharina'),
('Laura', 'Charlotte', 'Lara', 'Jasmin', 'Merle', 'Luisa', 'Celina', 'Nina', 'Jessika', 'Miriam')

3.4 zimmerbelegung4.txt

Output:
('Anna', 'Julia', 'Miriam', 'Jessika', 'Pauline', 'Jasmin'),
('Hannah', 'Celine', 'Annika', 'Pia', 'Lisa', 'Vanessa', 'Nina', 'Clara', 'Kim', 'Celina', 'Luisa', 'Carolin', 'Lara', 'Charlotte', 'Michelle', 'Nele', 'Emma', 'Lina', 'Merle'),
('Lea', 'Laura', 'Katharina', 'Sarah', 'Lena', 'Jana', 'Josephine', 'Marie', 'Johanna', 'Larissa', 'Sophie', 'Alina', 'Melina', 'Sofia', 'Leonie', 'Lilli', 'Emily'),
('Antonia')

3.5 zimmerbelegung5.txt

Ouput:
('Anna'),
('Hannah', 'Johanna', 'Sarah', 'Emily', 'Lena', 'Lisa'),
('Lea'),
('Leonie', 'Luisa', 'Kim'),
('Marie'),
('Laura'),
('Lara', 'Pauline'),
('Julia'),
('Alina'),
('Emma'),
('Nele'),

('Antonia'),
('Katharina'),
('Sophie'),
('Annika'),
('Jana'),
('Jasmin'),
('Lina'),
('Lilli'),
('Celine'),
('Michelle'),
('Pia'),
('Carolin'),
('Celina'),
('Miriam'),
('Vanessa'),
('Jessika'),
('Merle'),
('Melina'),
('Josephine'),
('Larissa'),
('Nina'),
('Sofia'),
('Charlotte'),
('Clara')

Hier scheitert das Programm an der Tatsache, dass viele Damen 'neutral' sind und theoretisch alle in ein gleiches Zimmer kommen könnten. Dies widerspricht jedoch der Definition, da zwei Damen sich genau dann ein Zimmer teilen, wenn es sich mindestens eine von ihnen wünscht. So oder so ist das obige Ergebnis gültig, jedoch nicht pragmatisch.

3.6 zimmerbelegung6.txt

Output:
('Anna', 'Carolin', 'Clara', 'Lena', 'Nina', 'Antonia', 'Lisa', 'Pia', 'Luisa', 'Lara', 'Laura', 'Nele', 'Leonie', 'Alina', 'Julia', 'Marie', 'Jessika', 'Katharina', 'Pauline', 'Emma', 'Miriam', 'Josephine', 'Lina', 'Kim', 'Charlotte'),
('Hannah', 'Sophie', 'Sarah', 'Vanessa', 'Jana', 'Johanna', 'Melina', 'Emily', 'Lilli', 'Larissa', 'Celine'),
('Lea', 'Michelle'),
('Annika', 'Celina'),
('Jasmin', 'Merle', 'Sofia')

3.7 eigenesBeispiel.txt

	+	-
Alex	Noah	
Noah	Benno	
Benno		Alex

Output:
"Keine Zimmerbelegung möglich."

Der Grund hierfür ist, dass sich Alex und Benno ein Zimmer teilen müssen, da sich Alex Noah wünscht und Noah Benno. Benno möchte mit Alex aber kein Zimmer teilen, weshalb keine Zimmerbelegung möglich ist.

4 Quellcode

Der folgende Auszug ist die Hauptmethode des Programms. Sie beschreibt den in der Umsetzung genannten Vorgang der Findung der Zimmerverteilung:

```
try:
  # wiederhole, solange personen nicht lehr ist
  while len(personen) > 0:
    # erstelle ein neues Zimmer und fuege
    # die erste Person in personen hinzu
    nzimmer = [personen[0]]
    # erstelle das gleich Zimmer in dem am Ende nur die Namen stehen
    reszimmer = []
    # loesche die hinzugefuegte Person aus personen
    del (personen[0])
    # found gibt an, ob in einem Durchlauf
    # ein Treffer gefunden wurde
    found = True
    # wiederhole solange im vorherigen
    # Durchgang ein Treffer gefunden wurde
    while found:
      # setze found standartgemaess auf Falsch
      found = False
      # pruefe auf Uebereinstimmung und speichere
      # den zurueckgegebenen Index als ind
      ind = posk_p(nzimmer)
      # wenn eine Uebereinstimmung gefunden wurde
      if ind != -1:
        # pruefe auf (nicht) negative Uebereinstimmung
        # bei der gefundenen Person
        if not negk_p(ind, nzimmer) and not negp_k(ind, nzimmer):
          # fuege die gefundene Person dem aktuellen Zimmer hinzu
          nzimmer.append(personen[ind])
          # loesche die hinzugefuegte Person aus personen
          del (personen[ind])
          # steze found auf Wahr
          found = True
        # wenn negative Uebereinstimmung gefunden wurde
        else:
          # springe zu Zeile 91 durch einen provozierten Fehler
          abbruch = 1 / 0
      else:
        # pruefe auf Uebereinstimmung und speichere
        # den zurueckgegebenen Index als ind
        ind = posp_k(nzimmer)
        # wenn eine Uebereinstimmung gefunden wurde
        if ind != -1:
          # pruefe auf (nicht) negative Uebereinstimmung
          # bei der gefundenen Person
          if not negk_p(ind, nzimmer) and not negp_k(ind, nzimmer):
            # fuege die gefundene Person dem aktuellen Zimmer hinzu
            nzimmer.append(personen[ind])
            # loesche die gefundene Person aus personen
            del (personen[ind])
            # setze found auf Wahr
            found = True
          # wenn negative Uebereinstimmung gefunden wurde
          else:
```

```
            # springe zu Zeile 91 durch einen provozierten Fehler
55          fail = 1 / 0
    # Durchaufe nzimmer als person
57  for person in nzimmer:
      # fuege reszimmer den Namen von person hinzu
59    reszimmer.append(person[0])
    # fuege das erstellte Zimmer der Liste aller Zimmer hinzu
61  zimmer.append(reszimmer)
# fange jeden "durch Null geteilt" Error auf
63 except ZeroDivisionError:
  # Gebe den text "Keine Zimmerbelegung moeglich!" aus
65  print("Keine Zimmerbelegung moeglich!")
  # Gebe die Liste mit allen Zimmern aus
67  print(zimmer)
```