Albert-Ludwigs-Universität
Institut für Informatik
Prof. Dr. F. Kuhn
M. Fuchs, G. Schmid

Algorithmen und Datenstrukturen
Sommersemester 2026
Musterlösung Übungsblatt 8

Abgabe: Dienstag, 23. Juni, 2026, 10:00 Uhr

Aufgabe 1: Tiefensuche *(6 Punkte)*

Wir definieren für jeden Knoten 2 Zeitpunkte (wie in Slide 29):

- $t_{v,1}$: Zeitpunkt wenn Knoten v von der DFS-Suche grau gefärbt wird
- $t_{v,2}$: Zeitpunkt wenn Knoten v von der DFS-Suche schwarz gefärbt wird

Sei ausserdem folgender *gerichteter* Graph $G = (V, E)$ gegeben mit

- $V = \{u_1, u_2, u_3, u_4, u_5\}$
- $E = \{(u_1, u_2), (u_1, u_3), (u_2, u_3), (u_3, u_4), (u_4, u_1), (u_5, u_1), (u_5, u_3), (u_5, u_4)\}$

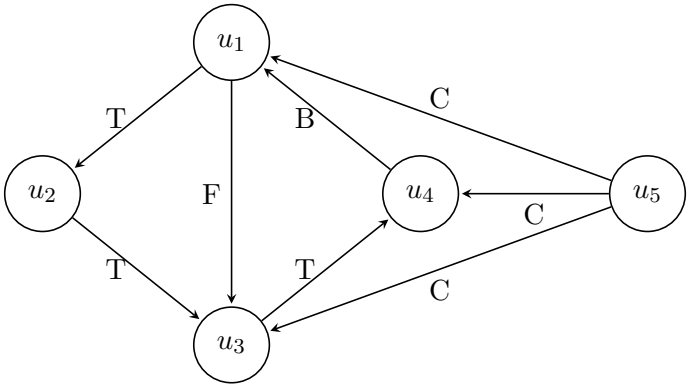
(a) Zeichnen Sie G. *(2 Punkte)*

(b) Schreiben Sie zu jedem Knoten in G das Bearbeitungsintervall $[t_{v,1}, t_{v,2}]$. Wenn mehrere Knoten als nächstes von der Tiefensuche besucht werden könnten, wird immer derjenige mit kleinstem Index gewählt (und wir starten somit auch bei u_1). *(2 Punkte)*

(c) Schreiben Sie zu jeder Kante ob es sich um eine Baumkante, Rückwärtskante, Vorwärtskante oder eine Querkanten handelt. *(2 Punkte)*

Musterlösung

ac) Wir bennenen Baumkanten mit T (Tree Edge), Rückwärtskanten mit B (Backward Edge), Vorwärtskanten mit F (Forward Edge) und Querkanten mit C (Cross Edge).



b)
- $u_1 : [1, 8]$
- $u_2 : [2, 7]$

- $u_3 : [3, 6]$
- $u_4 : [4, 5]$
- $u_5 : [9, 10]$

Aufgabe 2: Breitensuche auf dem Schauspielergraph *(6 Punkte)*

In der `movies.csv` Datei sind die Metadaten von über 4000 Filmen gespeichert. Zu jedem dieser Filme sind bis zu 3 Schauspieler angegeben, welche in diesem Film mitgespielt haben. Wir definieren uns den *Schauspielergraphen* G, als den Graphen dessen Knoten gerade die Schauspieler aus der Datei representieren und zwei Knoten eine Kante haben, wenn die beiden Schauspieler in mindestens einem Film gemeinsam aufgetreten sind. Im template `bfs.py` ist schon eine Funktion vorgegeben welche die entsprechende Adjazenzliste erstellt. Nun sollen Sie die Breitensuche aus der Vorlesung implementieren/erweitern und folgende Aufgaben lösen[1]:

(a) Die Breitensuche hat als Input einen *Startknoten*. Von diesem werden alle anderen (erreichbaren) Knoten erkundet. Erweitern Sie die Breitensuche aus der Vorlesung, welche zu jedem Knoten der neu erkundet wird, das level relativ zum Startknoten ausgibt. Sprich, der Startknoten selbst hat level 0, alle Knoten die direkt mit dem Startknoten verbunden sind haben das level 1, alle Knoten welche direkt mit einem Knoten der Stufe 1 verbunden sind (aber selbst noch nicht besucht wurden) haben das level 2 usw. Nutzen Sie ihre Implementierung um folgende Frage zu lösen: **Welche Distanz haben 'Tom Hanks' und 'Orlando Bloom' voneinander im Schauspielergraph?** (Alternativ Forumuliert: Wenn die BFS bei 'Tom Hanks' gestartet wird, welches level wird für 'Orlando Bloom' ausgegeben?) Schreiben Sie ihre Lösung in die Abgabe. *(3 Punkte)*

(b) Schreiben Sie eine weitere Funktion die ihre Breitensuche aus (a) nutzt um die Anzahl aller Zusammenhangskomponenten zu berechnen. Was bedeutet es wenn zwei Schauspieler in der gleichen Zusammenhangskomponente sind? Schreiben Sie ihre Lösung in die Abgabe. *(3 Punkte)*

(c) **Bonusaufgabe:** Wie groß ist die größte Zusammenhangskomponente, wie klein ist die kleinste? Schreiben Sie ihre Lösung in die Abgabe. *(2 bonus Punkte)*

Musterlösung

Der Python Code ist in `bfs.py`. Wenn man die Main nutzt um die Aufgaben zu lösen wird folgendes ausgegeben:

(a)
```
Searching for the distance between 'Tom Hanks' and 'Orlando Bloom'...
Success!  Found a path with distance 3.
```

(b)
```
Found 397 many components.
```

(c)
```
Largest component is the component of CCH Pounder with 4216 actors.
Second largest component is the component of Yuliya Snigir with 11 actors.
The smallest component is the component of Robert Redford with 1 actors.
```

Hinweis: Die Antworten sind nicht komplett eindeutig, da es viele Komponenten aus nur einem Schauspieler gibt. Ebenso können andere Repräsentanten für eine Komponente gewählt werden.

Aufgabe 3: Zyklenfreie Graphen *(8 Punkte)*

(a) Wie viele Kanten m kann ein ungerichteter zusammenhägender Graph mit n Knoten maximal haben (Unter der Annahme dass es keine Schleifen[2] gibt.)? Begründen Sie ihre Antwort. *(2 Punkte)*

[1] Submitten Sie nicht die `movies.csv` Datei ins SVN.

[2] Eine Schleife ist eine Kante der Form $\{v, v\}$

(b) Zeigen Sie, dass jeder ungerichteter zusammenhägende Graph welcher keinen Zyklus[3] enthält, genau $n-1$ Kanten hat (wobei n die Anzahl der Knoten des Graphen ist). *(3 Punkte)*
Hinweis: Sie können diese Aussage zum Beispiel per Induktion über $n \geq 1$ zeigen.

(c) Gegeben ein ungerichteter zusammenhängender Graph $G = (V, E)$ mit $n = |V|$. Geben Sie einen Algorithmus an der in Zeit $O(n)$ entscheidet ob G einen Zyklus enthält oder nicht. Spezifizieren Sie dafür explizit in welcher Datenstruktur G gegeben sein soll.

Musterlösung

(a) Ein Graph hat die maximale Anzahl von Kanten wenn jeder Knoten mit jedem Verbunden ist. Bedeutet als jeder Knoten hat den Grad $n-1$. Wir fixen nun einer Reihenfolge der Knoten $v_1, ..., v_n$ und zählen jeweils die "noch nicht gezählten" Kanten. Da hat v_1 gerade $n-1$ viele, v_2 hat noch $n-2$ viele (da die Kante zwischen v_1 und v_2 ja schon gezählt wurde), v_3 hat $n-3$ viele usw. Es gilt also:

$$m \leq \sum_{i=1}^{n}(n-i) = \sum_{i=1}^{n-1} i = \frac{(n-1)\cdot n}{2}$$

Ein anderer Ansatz wäre zu berechnen wie viele 2 elementige Teilmengen es von einer n elementigen Menge gibt. Davon gibt es gerade $\binom{n}{2} = \frac{n!}{2!\cdot(n-2)!} = \frac{n\cdot(n-1)}{2!} = \frac{(n-1)\cdot n}{2}$.

(b) Ein zusammenhängender Graph ohne Zyklus hat genau $n-1$ Kanten. Beweis per Induktion.
Induktionsanfang: Für $n = 1$ hat der Graph keine Kante, die Aussage ist also korrekt.
Induktionsvoraussetzung: Jeder zyklenfreie zusammenhängende Graph mit $k \leq n-1$ Knoten hat $k-1$ Kanten.
Induktionsschritt: Wir zeigen nun dass die Voraussetzung auch für einen n-Knoten zyklenfreien zusammenhängenden Graphen G wahr ist. Jeder Graph G mit n Knoten kann zusammengesetzt werden aus einem Knoten v der verbunden ist mit $l \geq 1$ disjunkten Subgraphen $G_1, ..., G_l$ von G. Da G zyklenfrei ist, ist auch jeder dieser Subgraphen zyklenfrei und die einzige Verbindung zwischen 2 Subgraphen ist über den Knoten v. Ohne Beschränkung der Allgemeinheit, sagen wir dass G_i grade n_i Knoten hat (für jeden dieser Subgraphen). Da $n_i \leq n-1$ für alle i, gilt durch die Induktionsvoraussetzung dass G_i gerade $n_i - 1$ Kanten hat. Wir können nun die Anzahl der Kanten m in G wie folgt berechnen:

$$m = deg(v) + \sum_{i=1}^{l}(n_i - 1) = l + \sum_{i=1}^{l} n_i - \sum_{i=1}^{l} 1 = \sum_{i=1}^{l} n_i = n-1$$

Dabei gilt $deg(v) = l$, da v ja gerade mit jedem der l Subgraphen verbunden ist und es gilt $\sum_{i=1}^{l} n_i = n-1$ da dies ja gerade die Summe über alle Knoten in G ohne v repräsentiert.

(c) Diese Aufgabe könnte man prinzipiell sowohl mit Tiefen- wie auch Breitensuche lösen. Wir nutzen hier die Breitensuche und gehen davon aus dass G als Adjazenliste vorliegt. Wir führen die Breitensuche "normal" aus, speichern uns aber für jeden Knoten v, von welchem Knoten u aus er zu erst erreicht wurde. Diesen Knoten u nennen wir **parent** von v. Wenn nun v einen markierten Nachbar hat, der nicht sein parent ist, dann existiert ein Zyklus im Graph und wir geben *false* zurück. Diese Prozedur hat die selbe Laufzeit wie die Breitensuche, also $O(n+m)$. Wenn m nun wie in Aufgabe a) gerade $O(n^2)$ ist, ist die Laufzeit natürlich zu schlecht. Wir wissen aber von b), dass wenn G zyklenfrei ist, er nur $n-1$ Kanten hat. Wir können also die Prozedur nach $n-1$ Schritten abbrechen und *false* zurückgeben sollte es noch unbesuchte Knoten in der Queue geben. Die Laufzeit ist somit $O(n)$.

[3] Ein Zyklus (oder auch Kreis) ist ein Pfad $v_1, \ldots, v_k \in V$ in einem Graphen bei dem zusätlich eine Kante zwischen dem Start und dem Endknoten existiert, also $\{v_1, v_k\} \in E$.