IWI Fakultät für Informatik und Wirtschaftsinformatik
Julian Schäfer, M.Sc.
Prof. Dr. Heiko Körner

Hochschule Karlsruhe Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES
Näher dran.

Probeklausur zur Vorlesung Algorithmen auf Graphen

Musterlösungen

Aufgabe 1:

a) Nach der Ausführung von BFS(G) enthält d die folgenden Werte:

Knoten v	a	b	c	d	e	f	g	h
Distanz $d[v]$	0	1	2	3	1	2	3	4

b) Das Verfahren STARKE ZUSAMMENHANGSKOMPONENTEN findet die folgenden Komponenten:

$$\begin{aligned} C_1 &= \{a\} \\ C_2 &= \{e\} \\ C_3 &= \{b\} \\ C_4 &= \{c, f, g\} \\ C_5 &= \{d, h\} \end{aligned}$$

Nach der ersten Phase enthalten d und f die folgenden Werte:

Knoten v	a	b	c	d	e	f	g	h
$d[v]$	13	1	8	5	14	2	3	4
$f[v]$	16	12	9	6	15	11	10	7

Nach der ersten Phase gilt für den Stapelspeicher $L = [d, h, c, g, f, b, e, a]$. Und nachdem das Verfahren terminiert ist enthalten d und f die folgenden Werte:

Knoten v	a	b	c	d	e	f	g	h
$d[v]$	1	1	2	2	1	1	3	1
$f[v]$	2	2	5	3	2	6	4	4

c) Es genügt im obigen Graphen die Kanten (a, b), (b, f) und (c, d) zu drehen, damit der Graph genau eine SCC besitzt. Die Menge ist nicht eindeutig, denn alternativ kann anstatt (b, f) kann auch (b, c) gedreht werden.

d) Die Lösung der Aufgabe setzt sich aus drei Teilen zusammen. Zunächst werden wir zeigen wie die gegebene Eigenschaft verwenden werden kann um die Kreisfreiheit eines ungerichteten Graphen zu zeigen. Im Anschluß implementieren wir ein passendes Verfahren und beweisen im letzen Schritt, dass dieses Verfahren korrekt arbeitet.

- Aus der in der Vorlesung bewiesenen Eigenschaft ergibt sich folgende logische Ableitung:

$$
\begin{array}{rcll}
\begin{array}{r}G \text{ ist ungerichtet} \\ \wedge\ G \text{ ist kreisfrei}\end{array} & \Longrightarrow & \text{n = m + p} & \\
n \neq m + p & \Longrightarrow & \neg(G \text{ ist ungerichtet} \wedge G \text{ ist kreisfrei}) & \text{[Kontraposition]} \\
n \neq m + p & \Longrightarrow & \neg\ G \text{ ist ungerichtet}) \vee \neg(G \text{ ist kreisfrei}) & \text{[De Morgan]} \\
n \neq m + p & \Longrightarrow & \begin{array}{l}G \text{ ist nicht ungerichtet} \\ \vee\ G \text{ ist nicht kreisfrei}\end{array} & \text{[De Morgan]}
\end{array}
$$

Ist $G = (V, E)$ ein ungerichteter Graph mit n Knoten, m Kanten gund p Zusammenhangskomponenten. Wenn n nicht der Summe aus m und p entspricht, dann kann G nicht kreisfrei sein. Denn wir wissen, dass G ungerichtet ist.

- Wir können folgendes Verfahren implementieren:

```
1  boolean istKreisfrei(G) {
2    // Verfahren aus der Vorlesung
3    List<Zusammenhangskomponenten> komponenten =
        STARKE_ZUSSAMENHANGSKOMPONENTEN(G);
4    // Berechnung der Anzahl aller Komponenten
5    int p = komponenten.size();
6    // Prüfung der hinreichenden Bedingung n = m + p
7    int n = G.V.size();
8    int m = G.E.size();
9    return n == m + p;
10 }
```

- Zu zeigen ist noch ob das Verfahren korrekt arbeitet (es liefert den Wert false gdw. der Graph eine Kreis enthält):
  - Wir führen den Beweis durch Widerspruch.
  - Angenommen es gibt einen ungrichteten Graph G mit einem Kreis für den das obige Verfahren den falschen Wert (true) liefert.
  - In Zeile 9 muss gelten, dass $n \neq m + p$.
  - Da in Zeile 7-8 die Werte n und m direkt aus G ausgelesen werden, muss die Berechnung der starken Zusammenhangskomponenten in Zeile 3 fehlerhaft sein.
  - In der Vorlesung haben wir gezeigt, dass das Verfahren STARKE ZUSAMMENHANGSKOMPONENTEN korrekt arbeitet.
  - Dies führt direkt zu einem Widerspruch, d.h. es kann keinen solchen Graph G geben für den das Verfahren den falschen Wert liefert

□

e) Der Graph G kann nicht topologisch sortiert werden, denn G ist nicht kreisfrei. Für einen Kreis kann keine topologische Sortierung gefunden werden. Zum Beispiel ist in G folgender Kreis enthalten: $c \to f \to g \to c$

f) Der Graph kann topologisch sortiert werden, wenn der Graph kreisfrei ist. Um dies zu erreichen können die Kanten $c \to f$ und $d \to h$ entfernt werden. Die Lösung ist nicht eindeutig, z.B. kann anstatt $d \to h$ die Kante $h \to d$ entfernt werden. Angenommen aus G werden die Kanten $c \to f$ und $d \to h$ entfernt, dann ist $[a, e, b, f, g, h, c, d]$ die topolische Sortierung von G.

Aufgabe 2:

a) Wendet man den Algorithmus von DISJKSTRA auf den Graphen G an, so ergibt sich folgende Reihenfolge der betrachteten Knoten:

$$s, d, a, b, c, e$$

Die Reihenfolge ist nicht eindeutig, denn in der vorletzten Iteration haben sowohl c als auch e die gleichen Kosten. Aus diesem Grund ist auch die folgende Reihenfolge eine korrekte Antwort:

$$s, d, a, b, e, c$$

b) Zum Zeitpunkt nach der Verarbeitung des Kontens b enthält p und k folgende Werte:

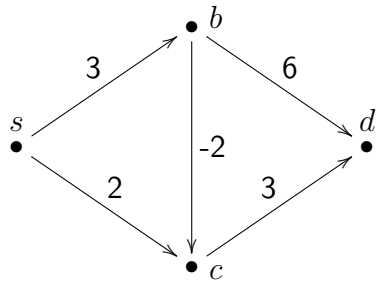
Konten v	s	a	b	c	d	e
$k(v)/p(v)$	$0/-$	$5/s$	$11/d$	$12/b$	$4/s$	$12/a$

Daraus ergeben sich folgende Pfade:

Konten v	Kürzeste Wege von s nach v
a	$s \to a$
b	$s \to d \to b$
c	$s \to d \to b \to c$
d	$s \to d$
e	$s \to a \to e$

c) Wenn es in einem kanten-gewichteten Digraphen eine negative Kante gibt, dann ist die Korrektheit der berechneten kürzesten Pfade nicht mehr garantiert. Angenommen s ist der Startknoten und ein Konten v wird während der Ausführung besucht, dann werden alle Nachfolger von v betrachtet um ggf. einen kürzeren Weg über v zu dem Nachfolger zu finden. Sei w ein solcher Nachfolger von v und ferner wird eine Verbesserung des bisherigen p-Wert mit $p(w) = v$ erzielt. Zusätzlich nehmen wir an, dass die Kante $v \to w$ negative Kosten besitzt ($c((v, w)) < 0$). Wurde w vor v schon besucht, so kann w kein zweites mal besucht werden (dies ist im DIJKSTRA-Algorithmus ausgeschlossen, Knoten werden nach der Initialisierung von Q nur entfernt). D.h. selbst wenn nun ein kürzerer Weg $s \to \ldots \to v \to w$ gefunden wird, führt dies zu keinen Verbesserungen für die Nachfolger von w. Somit kann es zu nicht-optimalen k- bzw. p-Werten für Nachfolger von w kommen.

d) Im folgenden gerichteten Graphen führt das Dijsktra-Verfahren zu einer inkorrekten Lösung. Ausgehend vom Startknoten s wählt das Dijkstra-Verfahren zunächst den Knoten c und versucht mit diesem eine Verbesserung aller direkten Nachfolger zu finden. Dies liefert für d einen verbesserten k-Wert mit $k[d] = 5$. Erst nachdem c aus Q entfernt wurde wird in der nächsten Iteration b gewählt. Mit b kann jetzt für c eine Verbesserung erzielt werden $k[c] = 1$. Leider befindet sich c nicht mehr in Q. Eine erneute Betrachtung der Kante $(c, d) \in E$ ist somit nicht mehr möglich und der korrekte k-Wert wird nicht erzielt ($k[d] = 4$).


b
3
6
s
d
-2
2
3
c