Algorithmen und Datenstrukturen

Übungsgruppe 14

Utz Pöhlmann
4poehlma@informatik.uni-hamburg.de
6663579

Louis Kobras
4kobras@informatik.uni-hamburg.de
6658699

Rene Ogniwek
reneogniwek@gmx.net
6428103

Steffi Kaussow
s.kaussow@gmail.com
6414862

4. Januar 2016

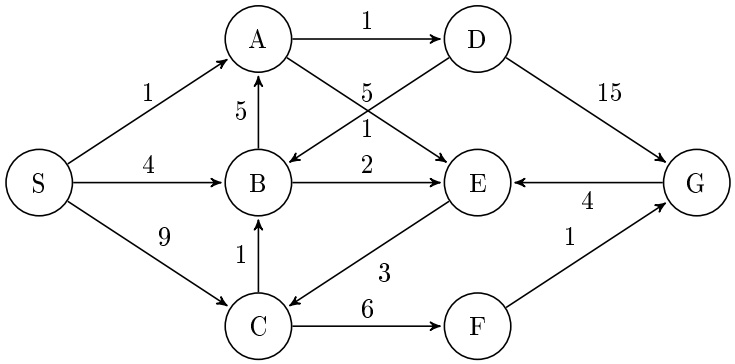
Punkte für den Hausaufgabenteil:

6.1	6.2	Σ

Zettel vom 10. 12. – Abgabe: 04. 01. 2016

Übungsaufgabe 6.1

[| 10]

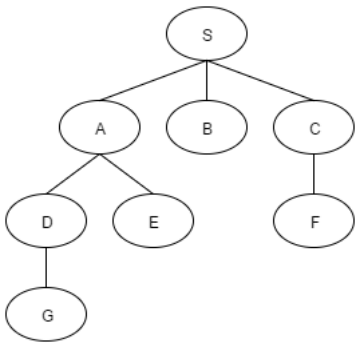


Betrachten Sie obigen Graphen G. Wenden Sie jeweils das verlangte Verfahren an bzw. beantwortet Sie die Frage oder begründen Sie, warum dies nicht nicht. (Bei den ersten sechs Teilaufgaben spielen die Kantengewichte keine Rolle.)

Aufgabe 6.1.1

Ermittel Sie mit der Breitensuche einen Breitensuchbaum. Starten Sie den Algorithmus bei s. (1 Pkt.)

Ergebnis der Breitensuche:
S A B C D E F G
BFS-Baum:



Aufgabe 6.1.2

Ist das Ergebnis der Breitensuche eindeutig? (1 Pkt.)

Ja, ist es, da die Reihenfolge der Kanten, die gewählt werden, gegeben ist durch die Struktur der Adjazenzmatrix des Graphen. Zumindest ist dies der Fall bei den bisher im Modul gegebenen Algorithmen. Liegt keine Adjazenzmatrix vor oder arbeitet ein Algorithmus mit einer Modifikation der BFS, so kann es, gegeben es gibt mehrere gültige Breitensuchbäume, dazu kommen, dass unterschiedliche Ergebnisse zurückgegeben werden.

Aufgabe 6.1.3

Ermitteln Sie mit der Tiefensuche einen Tiefensuchwald und insb. die Zeiten $d[v]$ und $f[v]$ für jeden Knoten v. Starten Sie den Algorithmus wieder bei s. (1 Pkt.)

Ergebnis	**Stack**	$d[v]$	$f[v]$
/	S	–	–
S	SA	S:0	–
SA	SAD	A:1	–
SAD	SADB	D:2	–
SADB	SADBE	B:3	–
SADBE	SADBEC	E:4	–
SADBEC	SADBECF	C:5	–
SADBECF	SADBECFG	F:6	–
SADBECFG	SADBECFG	G:7	–

Ergebnis	**Stack**	$d[v]$	$f[v]$
SADBECFG	SADBECF	–	G:8
SADBECFG	SADBEC	–	F:9
SADBECFG	SADBE	–	C:10
SADBECFG	SADB	–	E:11
SADBECFG	SAD	–	B:12
SADBECFG	SA	–	D:13
SADBECFG	S	–	A:14
SADBECFG	/	–	S:15

Knoten	$d[v]$	$f[v]$
S	0	15
A	1	14
B	3	12
C	5	10
D	2	13
E	4	11
F	6	9
G	7	8

Aufgabe 6.1.4

Geben Sie eine topologische Sortierung des Graphen G an. (1 Pkt.)

Es ist nicht möglich, eine topologische Sortierung für G anzugeben, da G zyklisch ist und somit unabhängig von der gewählten Reihenfolge keine Möglichkeit besteht, rückwärtige Kanten zu vermeiden.

Aufgabe 6.1.5

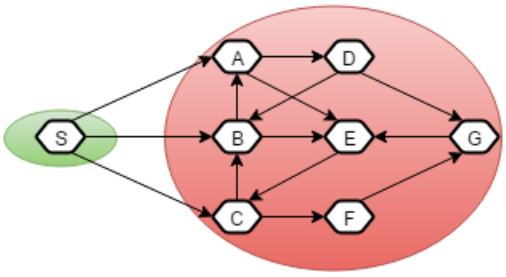
Bestimmen Sie mit dem Algorithmus aus der Vorlesung die starken Zusammenhangskomponenten von G. Geben Sie dazu das Ergebnis von $DFS(G)$ und $DFS(G^T)$ sowie die starken Zusammenhangskomponenten an. (1 Pkt.)

$DFS(G)$: s. 6.1.3.
$DFS(G^T)$: nicht möglich, da keine topologische Sortierung vorliegt.
Starke Zusammenhangskomponenten:
[S], [A,B,C,D,E,F,G]

Aufgabe 6.1.6

Geben Sie den Komponentengraphen von G an. (1 Pkt.)

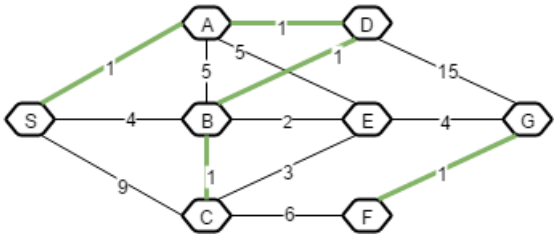
Der Übersicht halber wurden die Kantengewichte, da sie für diese Aufgabe ohnehin irrelevant sind, weggelassen.



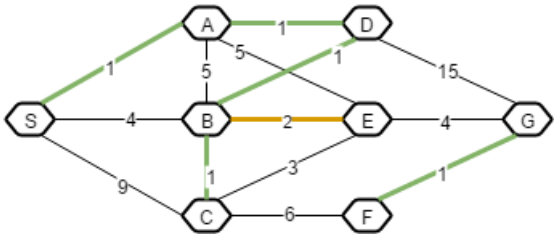
Aufgabe 6.1.7

Interpretieren Sie G nun als ungerichteten Graphen. Wenden Sie den Algorithmus von Kruskal an, um einen minimalen Spannbaum zu bestimmen. (Es genügt hier anzugeben welche Kanten in den minimalen Spannbaum aufgenommen werden und in welcher Reihenfolge dies geschieht!) (1 Pkt.)

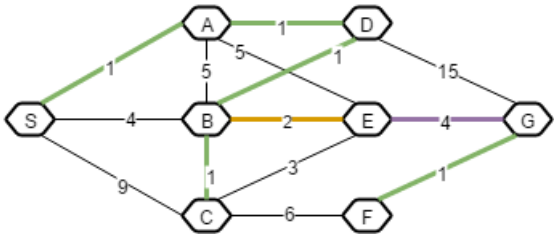
Schritt 1: Hinzufügen aller Kanten mit Gewicht 1.
$\{(S,A),(A,D),(B,D),(B,C),(F,G)\}$



Schritt 2: Hinzufügen aller Kanten mit Gewicht 2.
$\{(B,E)\}$



Schritt 3: Kanten mit Gewicht 3 sind redundant. Fahre fort mit Kanten mit Gewicht 4.
$\{(E,G)\}$

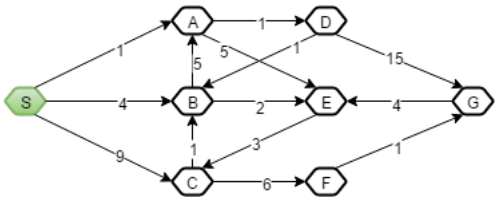


Aufgabe 6.1.8

Bestimmen Sie mit dem Algorithmus von Dijkstra kürzeste Pfade von s zu allen anderen Knoten. Geben Sie hierzu jeweils den Graph mit den Werten von $d[v]$ nach jeder Iteration der while-Schleife an und machen Sie $\pi[v]$ z.B. durch dickere Kanten deutlich. (2 Pkt.)

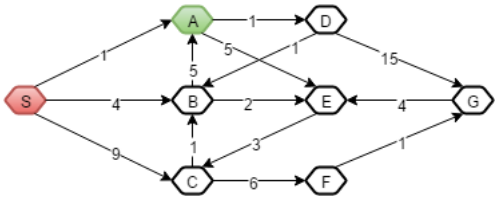
Sei $v \in V(G)$. Dann ordnet $d(v)$ jedem Knoten seine Distanz zum Startknoten zu. Sei desweiteren die Startdistanz jedes Knoten zum Startknoten zunächst ∞. Außerdem werden fertig bearbeitete Knoten mit roter Schrift markiert, der als nächstes geprüfte Knoten in grüner Schrift.

1. **Anfang**



v	S	A	B	C	D	E	F	G
$d(v)$	0	∞	∞	∞	∞	∞	∞	∞

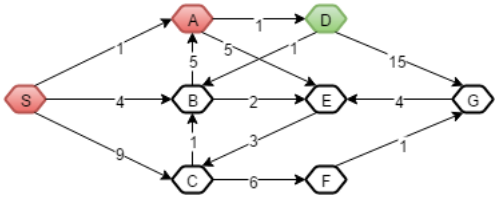
2. **S**



v	S	A	B	C	D	E	F	G
$d(v)$	0	1	4	9	∞	∞	∞	∞

Kanten:
$(s,a),(s,b),(s,c)$

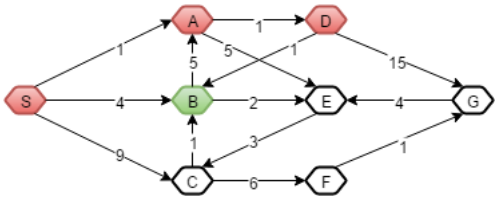
3. **A**



v	S	A	B	C	D	E	F	G
$d(v)$	0	1	4	9	2	6	∞	∞

Kanten:
$(s,a),(s,b),(s,c),(a,d),(a,e)$

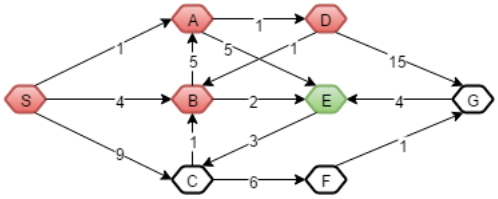
4. **D**



v	S	A	B	C	D	E	F	G
$d(v)$	0	1	4	9	2	6	∞	17

Kanten:
$(s,a),(s,c),(a,d),(a,e),(d,b),(d,g)$

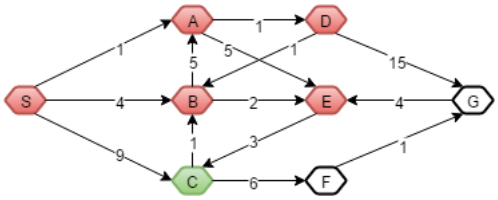
5. **B**



v	S	A	B	C	D	E	F	G
$d(v)$	0	1	3	9	2	5	∞	17

Kanten:
$(s,a),(s,b),(s,c),(a,d),(a,e)$

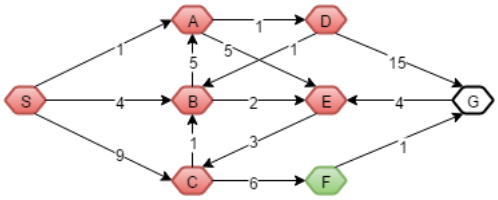
6. **E**



v	S	A	B	C	D	E	F	G
$d(v)$	0	1	3	8	2	5	∞	17

Kanten:
$(s,a),(a,d),(d,b),(d,g),(b,e),(e,c)$

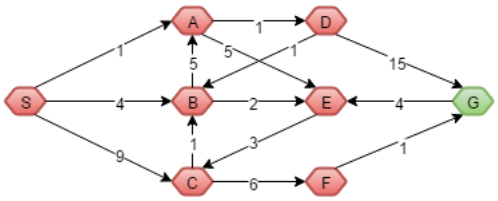
7. **C**



v	S	A	B	C	D	E	F	G
$d(v)$	0	1	3	8	2	5	14	17

Kanten:
$(s,a),(a,d),(d,b),(d,g),(b,e),(e,c),(c,f)$

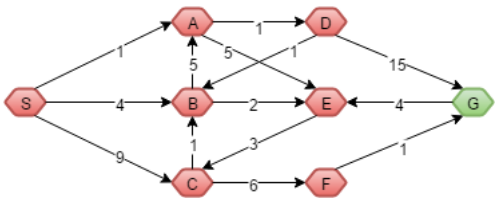
8. **F**



v	S	A	B	C	D	E	F	G
$d(v)$	0	1	3	8	2	5	14	15

Kanten:
$(s,a),(a,d),(d,b),(b,e),(e,c),(c,f),(f,g)$

9. **G**



v	S	A	B	C	D	E	F	G
$d(v)$	0	1	3	8	2	5	14	15

Kanten:
$(s,a),(a,d),(d,b),(b,e),(e,c),(c,f),(f,g)$

Aufgabe 6.1.9

Betrachten Sie den Sourcecode von Dijkstras Algorithmus und vom Algorithmus von Prim. Was ist der hauptsächliche Unterschied der beiden Algorithmen bei der Ausführung? (1 Pkt.)

Während Dijkstra die Entfernung jedes Knotens zum Startknoten speichert, enthalten die Knoten nach Prim jeweils nur die Entfernung zum Vorgängerknoten. Dieser Unterschied wird durch Dijkstra's `Relax` in Zeile 8ff[1] hervorgerufen (Dieser `if`-Block ist äquivalent zur Funktion `Relax` von der Seite vorher).

[1] Foliensatz 9, Version 03.01.2016, Seite 119

Übungsaufgabe 6.2

[| 6]

Aufgabe 6.2.1

Sei $G = (V, E)$ ein ungerichteter Graph. Bei einer k-Färbung von G wird jedem Knoten $v \in V$ eine Farbe $c \in [k] = \{1, 2, \ldots, k\}$ zugewiesen. Die Färbung ist korrekt, wenn zwei benachbarte Knoten nie die gleiche Farbe haben. Formal ist eine *k-Färbung* dann also eine Abbildung $f : V \to [k]$ und eine *korrekte k-Färbung* erfüllt $f(x) \neq f(y)$ für jede Kante $\{x, y\} \in E$. Wir definieren folgendes Problem:

$$k - col = \{\langle G \rangle | G \text{ besitzt eine korrekte } k\text{-Färbung}\}$$

Angenommen Sie wissen, dass 3-col *NP*-vollständig ist. Zeigen Sie, dass auch 8-col *NP*-vollständig ist. (2 Pkt.)

Sei das Zertifikat eine Menge bestehend aus einem Graphen $G = (V, E)$ und einer Färbungsfunktion f. Die Prüfung des korrekten Ergebnisses der Funktion in Form von $f(x) \neq f(y)$ für zwei benachbarte Knoten $x, y \in V$ lässt sich in polynomieller Zeit erledigen, womit 8-col in NP liegt.

Reduktion
Es wird nach Aufgabenstellung 3-col auf 8-col reduziert.
Nach Aufgabenstellung ist bekannt, dass 3-col in NP liegt.

Sei die Gruppe H eine Menge von Knoten, die eine erfüllende Belegung von 3-col darstellt. Zu H wird ein Knoten hinzugefügt, der paarweise mit allen drei Knoten aus H verbunden wird. Da der neue Knoten mit allen anderen Knoten verbunden ist, muss er eine Farbe erhalten, die bisher nicht vergeben ist. H' ist nun eine erfüllende Belegung für 4-col und jede 3 Knoten aus H sind eine erfüllende Belegung für 3-col. Dies wird wiederholt, bis H'_neu 8 Knoten vorweist. Für jede k Knoten gilt hierbei, dass es sich um eine erfüllende Belegung für k-col handelt, und jede Auswahl aus k-1 Knoten aus H eine erfüllende Belegung für (k-1)-col ist. Diese Erweiterung liegt in $\mathcal{O}(k - |V_{H_0}|)$.

Aufgabe 6.2.2

Wir betrachten das folgende Spiel: Auf eiem Feld mit n Spalten, n Zeilen und $n \times n$ Feldern können auf jeder der n^2 Positionen entweder ein weißer oder ein schwarzer Stein liegen oder das Feld kann leer sein. Eine Anfangsmarkierung besteht aus einer Zuweisung von Spielsteinen zu Feldern. Das Spiel wird dann (alleine) gespielt, indem beliebig viele Steine entfernt werden. Ziel ist es, dass jede Spalte nur noch Steine einer Farbe enthält (oder leer ist) und jede Zeile mindestens einen Stein enthält. Ob man dieses Ziel erreichen kann (und damit gewinnt), hängt von der Anfangsmarkierung ab. Die Eingabe ist ein Spiel S bestehend aus dem Feld und der Anfangsmarkierung (für das Feld reicht die Zahl n). Die Eingabe wird akzeptiert, wenn S gewonnen werden kann. Zeigen Sie, dass dieses Problem *NP*-vollständig ist. (4 Pkt.)

Reduktion von 3KNF auf Kreuzgo[1]:
Eine Spalte wird als Klausel interpretiert, die dann wahr ist, wenn für jedes Feld gilt:

$$((schwarz \wedge \neg weiss) \vee (\neg schwarz \wedge weiss) \vee null)$$

je nachdem, ob der erste gefundene Stein in der Spalte schwarz oder weiß ist.
Ebenso wird eine Zeile als Klausel interpretiert, die dann wahr ist, wenn für mindestens ein Feld gilt:

$$(schwarz \vee weiss \vee \neg null)$$

[1] Wir haben das Spiel in Ermangelung eines Namen so getauft

Diese Belegung muss für jede Zeile bzw. jede Spalte respektive gelten; die einzelnen Klauseln sind konjunkt. Gibt es eine erfüllende Belegung für die dadurch entstehende KNF, so ist diese Lösung auch eine Lösung für Kreuzgo.