Prof. Dr. Ir. Joost-Pieter Katoen

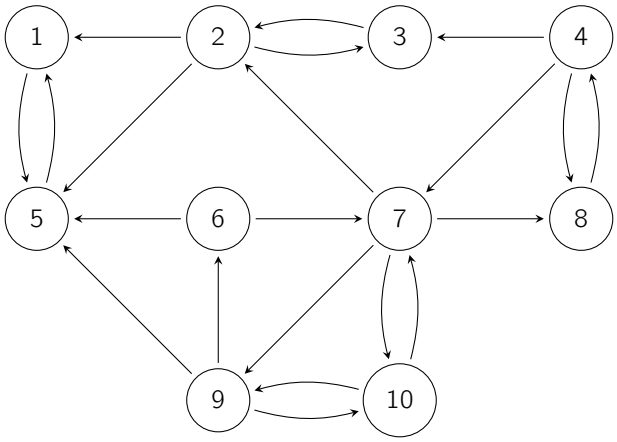
Christian Dehnert, Friedrich Gretz, Benjamin Kaminski, Thomas Ströder

Allgemeine Hinweise:

- Die **Hausaufgaben** sollen in Gruppen von je **2-3 Studierenden** aus der **gleichen Kleingruppenübung (Tutorium)** bearbeitet werden. **Namen und Matrikelnummern** der Studierenden sind auf jedes Blatt der Abgabe zu schreiben. **Heften bzw. tackern Sie die Blätter!**
- Die **Nummer der Übungsgruppe** muss **links oben** auf das **erste Blatt** der Abgabe geschrieben werden. Notieren Sie die Gruppennummer gut sichtbar, damit wir besser sortieren können.
- Die Lösungen müssen **bis Mittwoch, den 24.06.2015 um 12:15 Uhr** in den entsprechenden Übungskasten eingeworfen werden. Sie finden die Kästen am Eingang Halifaxstr. des Informatikzentrums (Ahornstr. 55). Alternativ können Sie die Lösungen auch vor der Abgabefrist direkt bei Ihrer Tutorin/Ihrem Tutor abgeben.
- In Aufgaben, bei denen Sie Algorithmen implementieren sollen, dürfen Sie Ihre Lösung als Pseudo-Code abgeben. Abgaben in verbreiteten imperativen Sprachen wie Java oder C++ sind ebenfalls erlaubt.

Tutoraufgabe 1 (Starke Zusammenhangskomponenten):

Wenden Sie *Sharir's Algorithmus* aus der Vorlesung an, um die starken Zusammenhangskomponenten des folgenden Graphen zu finden. Geben Sie das Array `color` und den Stack `S` nach jeder Schleifeniteration der ersten und zweiten Phase (also nach Zeile 17 und nach Zeile 22 im Code auf den Folien) an, falls `DFS1` bzw. `DFS2` ausgeführt wurde. Geben Sie zudem das Array `scc` nach jeder Schleifeniteration der zweiten Phase (also nach Zeile 22) an, falls `DFS2` ausgeführt wurde. Nehmen Sie hierbei an, dass `scc` initial mit Nullen gefüllt ist und die Knoten im Graphen und in allen Adjazenzlisten aufsteigend nach ihren Schlüsselwerten sortiert sind, also der Knoten mit Schlüssel 1 vom Algorithmus als erstes berücksichtig wird usw.



Tutoraufgabe 2 (Kondensationsgraph):

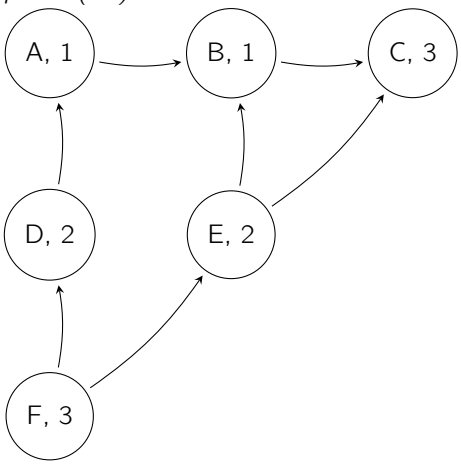
a) Zeigen Sie, dass für jeden gerichteten Graphen G gilt, dass der transponierte Graph des Kondensationsgraphen von G gleich dem Kondensationsgraphen des transponierten Graphen von G ist:

$$(G\downarrow)^T \;=\; (G^T)\downarrow$$

b) Entwerfen Sie einen Algorithmus, der für einen gegebenen gerichteten Graphen $G = (V, E)$ in Laufzeit $\mathcal{O}(|V| + |E|)$ den Graph $G\downarrow$ berechnet. Stellen Sie insbesondere sicher, dass es in $G\downarrow$ höchstens eine Kante von einem Knoten v zu einem anderen Knoten w gibt (d.h. es darf keine doppelten Kanten geben).

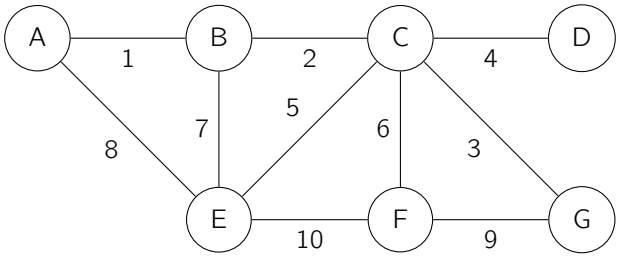
Tutoraufgabe 3 (Kritischer Pfad):

Bestimmen Sie eine *topologische Sortierung* unter Verwendung des in der Vorlesung vorgestellten Algorithmus für den folgenden Graphen. Die Knoten in diesem Graphen sind mit jeweils einem Schlüssel und einer Dauer beschriftet. Im gesamten Algorithmus werden Knoten in aufsteigender alphabetischer Reihenfolge ihrer Schlüssel berücksichtigt. Geben Sie als Ergebnis die Liste der Knotenschlüssel zusammen mit ihrem jeweiligen *frühesten Endzeitpunkt (eft)* in aufsteigender Reihenfolge der Topologiewerte an. Markieren Sie außerdem einen *kritischen Pfad*, indem Sie die zugehörigen Knotenschlüssel unterstreichen, und geben Sie den gesamten *frühesten Endzeitpunkt (eft)* an.



Tutoraufgabe 4 (Minimaler Spannbaum):

Führen Sie Prim's Algorithmus auf dem folgenden Graphen aus.



Der Startknoten hat hierbei den Schlüssel A. Geben Sie dazu *vor* jedem Durchlauf der äußeren Schleife an,

1. welche Kosten die Randknoten haben (d. h. für jeden Knoten `v` in `pq` die Priorität von `v`, wobei ∞ angibt, dass der entsprechende Knoten noch nicht zum Randbereich gehört)
2. und welchen Knoten `pq.getMin()` wählt, indem Sie den Kosten-Wert des gewählten Randknoten in der Tabelle unterstreichen (wie es in der ersten Zeile bereits vorgegeben ist).

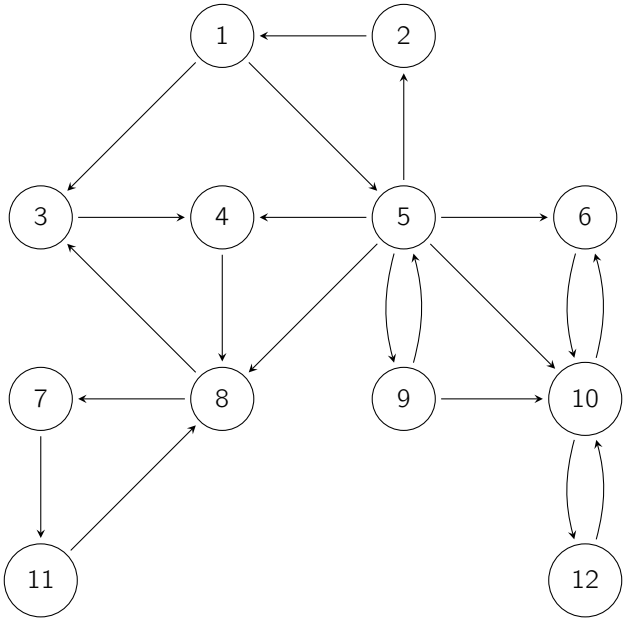
Geben Sie zudem den vom Algorithmus bestimmten minimalen Spannbaum an.

#Iteration	A	B	C	D	E	F	G
1	$\underline{0}$	∞	∞	∞	∞	∞	∞
2							
3							
4							
5							
6							
7							

Minimaler Spannbaum:

Aufgabe 5 (Starke Zusammenhangskomponenten): (8 Punkte)

Wenden Sie *Sharir's Algorithmus* aus der Vorlesung an, um die starken Zusammenhangskomponenten des folgenden Graphen zu finden. Geben Sie das Array `color` und den Stack `S` nach jeder Schleifeniteration der ersten und zweiten Phase (also nach Zeile 17 und nach Zeile 22 im Code auf den Folien) an, falls `DFS1` bzw. `DFS2` ausgeführt wurde. Geben Sie zudem das Array `scc` nach jeder Schleifeniteration der zweiten Phase (also nach Zeile 22) an, falls `DFS2` ausgeführt wurde. Nehmen Sie hierbei an, dass `scc` initial mit Nullen gefüllt ist und die Knoten im Graphen und in allen Adjazenzlisten aufsteigend nach ihren Schlüsselwerten sortiert sind, also der Knoten mit Schlüssel 1 vom Algorithmus als erstes berücksichtig wird usw.



Aufgabe 6 (Starke Zusammenhangskoponenten II): (4 + 7 = 11 Punkte)

a) Professor Cartoon (der böse Zwillingsbruder von Professor Katoen) behauptet, dass der Algorithmus zum Erkennen starker Zusammenhangskomponenten vereinfacht werden kann, indem man in der zweiten Tiefensuche den eigentlichen Graphen G statt des transponierten Graphen benutzt und den Stack des Algorithmus durch eine Queue ersetzt (und damit die Reihenfolge der Startknoten der Tiefensuche in der zweiten Phase "umkehrt").

Legen Sie Professor Cartoon das Handwerk indem Sie ihn widerlegen.

b) Geben Sie einen Algorithmus (Pseudocode oder präzise Beschreibung) an, welcher für einen gerichteten Graphen $G = (V, E)$ in Laufzeit $\mathcal{O}(|V| + |E|)$ einen Graphen $G' = (V, E')$ berechnet, so dass

(a) G' die gleichen starken Zusammenhangskomponenten wie G hat,

(b) $G'\downarrow \;=\; G\downarrow$, und

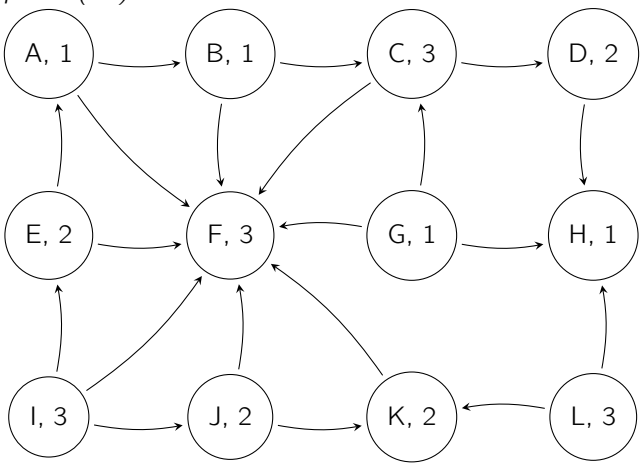
(c) $|E'|$ minimal ist (über alle Graphen, die die ersten beiden Bedingungen erfüllen).

Begründen Sie, wieso Ihr Algorithmus die Laufzeitschranke einhält und wieso er korrekt arbeitet.

Hinweis: Sie dürfen das Ergebnis von Tutoraufgabe 2 b) benutzen.

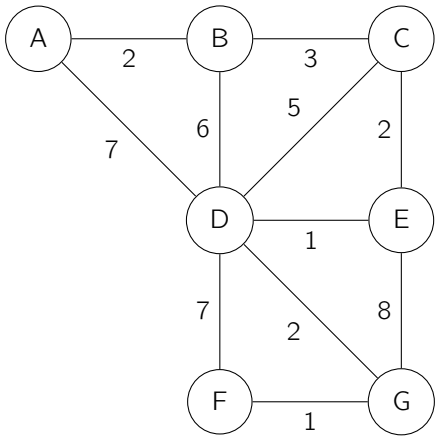
Aufgabe 7 (Kritischer Pfad): (5 Punkte)

Bestimmen Sie eine *topologische Sortierung* unter Verwendung des in der Vorlesung vorgestellten Algorithmus für den folgenden Graphen. Die Knoten in diesem Graphen sind mit jeweils einem Schlüssel und einer Dauer beschriftet. Im gesamten Algorithmus werden Knoten in aufsteigender alphabetischer Reihenfolge ihrer Schlüssel berücksichtigt. Geben Sie als Ergebnis die Liste der Knotenschlüssel zusammen mit ihrem jeweiligen *frühesten Endzeitpunkt (eft)* in aufsteigender Reihenfolge der Topologiewerte an. Markieren Sie außerdem einen *kritischen Pfad*, indem Sie die zugehörigen Knotenschlüssel unterstreichen, und geben Sie den gesamten *frühesten Endzeitpunkt (eft)* an.



Aufgabe 8 (Minimaler Spannbaum): (4 Punkte)

Führen Sie Prim's Algorithmus auf dem folgenden Graphen aus.



Der Startknoten hat hierbei den Schlüssel A. Geben Sie dazu *vor* jedem Durchlauf der äußeren Schleife an,

1. welche Kosten die Randknoten haben (d. h. für jeden Knoten `v` in `pq` die Priorität von `v`, wobei ∞ angibt, dass der entsprechende Knoten noch nicht zum Randbereich gehört)
2. und welchen Knoten `pq.getMin()` wählt, indem Sie den Kosten-Wert des gewählten Randknoten in der Tabelle unterstreichen (wie es in der ersten Zeile bereits vorgegeben ist).

Geben Sie zudem den vom Algorithmus bestimmten minimalen Spannbaum an.

#Iteration	A	B	C	D	E	F	G
1	$\underline{0}$	∞	∞	∞	∞	∞	∞
2							
3							
4							
5							
6							
7							

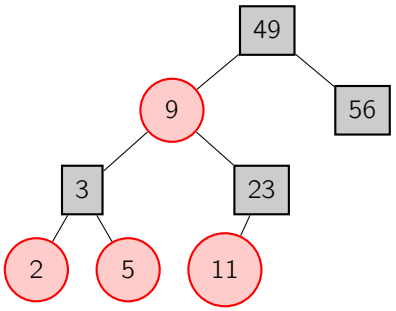
Minimaler Spannbaum:

Aufgabe 9 (Rot-Schwarz-Bäume): (15 Minuten, 2 + 1,5 + 2,5 + 2 = 8 Punkte)

a) Fügen Sie den Wert 7 in den folgenden *Rot-Schwarz-Baum* ein und geben Sie die entstehenden Bäume nach

- jeder *Einfügeoperation*,
- jeder *Rotation* sowie
- jeder *Umfärbung* an.

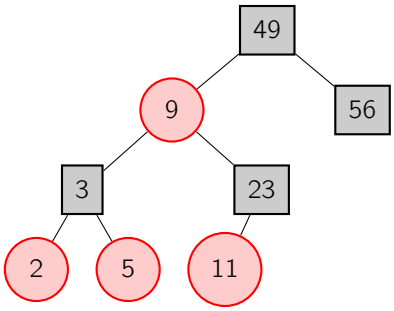
Markieren Sie außerdem zu jeder Rotation, welcher Knoten in welche Richtung rotiert wird. Mehrere Umfärbungen können Sie in einem Schritt zusammenfassen. Beachten Sie, dass rote Knoten rund und schwarze Knoten eckig dargestellt werden.



b) Fügen Sie den Wert 10 in den folgenden *Rot-Schwarz-Baum* ein und geben Sie die entstehenden Bäume nach

- jeder *Einfügeoperation*,
- jeder *Rotation* sowie
- jeder *Umfärbung* an.

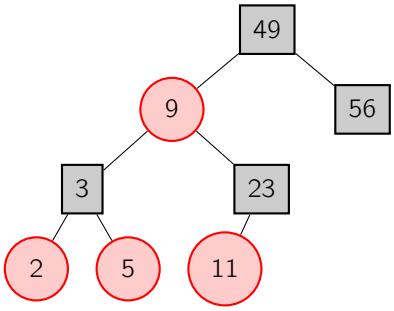
Markieren Sie außerdem zu jeder Rotation, welcher Knoten in welche Richtung rotiert wird. Mehrere Umfärbungen können Sie in einem Schritt zusammenfassen. Beachten Sie, dass rote Knoten rund und schwarze Knoten eckig dargestellt werden.



c) Löschen Sie den Wert 56 aus dem folgenden *Rot-Schwarz-Baum* und geben Sie die entstehenden Bäume nach

- jeder *Löschoperation*,
- jeder *Rotation* sowie
- jeder *Umfärbung* an.

Markieren Sie außerdem zu jeder Rotation, welcher Knoten in welche Richtung rotiert wird. Mehrere Umfärbungen können Sie in einem Schritt zusammenfassen. Beachten Sie, dass rote Knoten rund und schwarze Knoten eckig dargestellt werden.



d) Löschen Sie den Wert 9 aus dem folgenden *Rot-Schwarz-Baum* und geben Sie die entstehenden Bäume nach

- jeder *Löschoperation*,
- jeder *Rotation* sowie
- jeder *Umfärbung* an.

Markieren Sie außerdem zu jeder Rotation, welcher Knoten in welche Richtung rotiert wird. Mehrere Umfärbungen können Sie in einem Schritt zusammenfassen. Beachten Sie, dass rote Knoten rund und schwarze Knoten eckig dargestellt werden.