Albert-Ludwigs-Universität
Institut für Informatik
Prof. Dr. F. Kuhn
P. Bamberger
M. Fuchs

Graphentheorie
Sommersemester 2021
Musterlösung Übungsblatt 4

Abgabe: 15.06, 16:00 Uhr.

Aufgabe 1: Breitensuchen und Tiefensuche *(6 Punkte)*

a) Die Tiefensuche (Algorithmen 7.1 und 7.2) konstruiert einen Partialgraph G_π welcher den eigentlichen Graphen G in einen Wald aus Wurzelbäumen zerlegt. Kann man mit der Breitensuche (Algorithmen 3.2 und 3.3) auch einen (gerichteten, einfachen, schwach Zusammenhängenden) Graphen $G = (V, R)$ in einen Wald aus Wurzelbäumen zerlegen? Geben Sie an welche Kanten die Baumkanten $R_\pi \subseteq R$ sind und zeigen sie dass hier tatsächlich ein Wald entsteht. *(2 Punkte)*

b) Sowohl die Tiefensuche wie auch die Breitensuche kann dazu benutzt werden Spannbäume/Spannwälder in ungerichteten Graphen zu finden. Sei im folgenden ein einfacher ungerichteter Graph G gegeben. Nehmen sie außerdem an dass G zusammenhängend ist, was im ungerichteten Fall impliziert dass die Tiefen- bzw. Breitensuche einen Spannbaum konstruiert.

Sei T der Spannbaum der entsteht wenn wir eine Tiefensuche ausführen, welche bei einem Knoten u startet. Angenommen die Breitensuche mit gleichem Startknoten u produziert den exakt gleichen Baum T. Beweisen Sie dass in diesem Fall $G = T$ gelten muss. *(3 Punkte)*

c) Zeigen Sie dass die Aussage aus b) nicht für gerichtete Graphen gilt, selbst wenn alle Knoten vom Startknoten u aus erreichbar sind. *(1 Punkt)*

Musterlösung

a) Ja mit der Breitensuche kann man einen Wald aus Wurzelbäumen erzeugen. Zur Konstruktion von R_π: R_π wird zu Beginn von Algorithmus 3.3 mit $\emptyset$ initialisiert und als zusätzlicher Parameter dem Algorithmus 3.2 (ERREICHBAR) mitgegeben.

Algorithmus 3.2 nimmt sich den ersten Knoten aus $v \in L$ und löscht diesen aus L. Nun werden alle nicht-markierten benachbarten Knoten von v zu L hinzugefügt (und selbst markiert). Wir fügen in diesem Ablauf nun zusätzlich eine Kante $r = (v, w)$ zu R_π hinzu, nachdem w zu L hinzugefügt wurde (wichtig ist dass r nur dann hinzugefügt wird wenn w noch nicht markiert war).

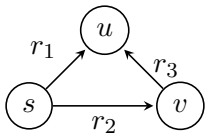
Beweis dass keine Kreise entstehen: Angenommen eine Kante $(v, w) = r$ erzeugt beim Einfügen in R_π einen Kreis, dann müssen sowohl v wie auch w schon markiert worden sein. Da r nur hinzugefügt wird wenn w noch nicht markiert ist, ergibt sich ein Widerspruch.

b) Beweis durch Widerspruch. Angenommen es gibt eine Kante $e = \{x, y\}$ die in G aber nicht in T vorkommt. Da e nicht in T vorkommt bedeutet dies dass die Breitensuche x und y markiert hat bevor entlang dieser Kante ein neuer Knoten 'erforscht' wird. Sei im folgenden t_x (bzw. t_y) der kürzeste Weg von s nach x (bzw. y) auf T. Nehmen wir im folgenden an dass o.B.d.A die Breitensuche x vor y erforscht hat. Dann muss, aufgrund der Existenz von e gelten, dass entweder

x und y den gleichen Abstand zu s haben oder dass y um eine Kante weiter entfernt ist (Wäre y 2 oder mehr Kanten weiter entfernt würde die Breitensuche y über die Kante e erforschen was eine Verletzung der Annahme darstellt). Es gilt also $t_x \leq t_y \leq t_x + 1$.

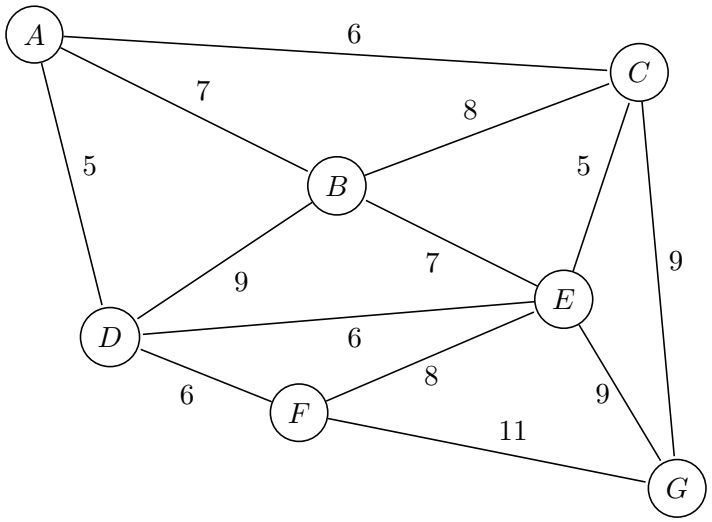
Da die Tiefensuche nach Voraussetzung den gleichen Baum T findet, müssen bezüglich T alle Knoten den gleichen Abstand zu s haben. Wenn nun die Tiefensuche den Knoten x erforscht hat, arbeitet sie jede Kante von x ab welche nicht zu einem bereits erforschten Nachbarn führt. Unter diesen Kanten befindet sich auch e. Da e nicht in T liegt muss die Tiefensuche um y von x zu erreichen über mindestens einen weiteren Knoten laufen (es gibt ja keine parallelen Kanten!), also gilt $t_y \geq t_x + 2$. Widerspruch.

c) In folgender Abbildung können sowohl die Tiefen- wie auch die Breitensuche einen Baum T finden, welcher nicht die Kante r_3 inkludiert. Da $T \neq G$ gilt die Aussage nicht.



Aufgabe 2: MSF *(4 Punkte)*

Betrachten Sie folgenden Graphen G:



a) Finden Sie einen minimalen Spannbaum für G mit Hilfe des **Algorithmus von Kruskal**. Geben Sie dafür in jedem Iterationsschritt die Menge E_F an. *(1 Punkt)*

b) Führen Sie den **Algorithmus von Prim** beginnend mit Startknoten B aus. Dabei soll der selbe Spannbaum wie in a) entstehen. Geben Sie auch hier in jedem Schritt die Mengen S und E_T an. *(1 Punkt)*

c) Stellen sie eine eindeutige Sortierung der (Kanten)Kosten bezüglich der Identität der Endknoten her, so dass man den **Algorithmus von Borůvka** ausführen kann.

Hinweis: Sie können hierfür die lexikographische Ordnung auf den Knoten nutzen.

Führen Sie anschließend den **Algorithmus von Borůvka** aus (Sie müssen hier keine Zwischenschritte angeben). *(1 Punkt)*

d) Der **Algorithmus von Kruskal** hat eine Laufzeit von $O(m \log m)$. Wie würde sich seine Laufzeit verbessern, wenn die Kantengewichte $c(e_1), ..., c(e_m)$ schon in sortierter Reihenfolge ($c(e_1) \leq c(e_2) \leq ... \leq c(e_m)$) gegeben sind? *(1 Punkt)*

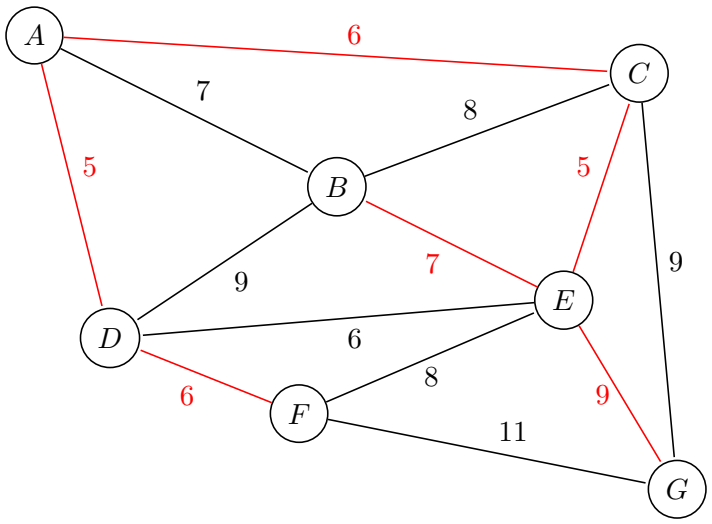
Musterlösung

a) Für den Algorithmus von Kruskal benötigt man eine Sortierung der Kanten (bezüglich deren Kosten). Eine Mögliche Sortierung ist folgende:

$$\{A,D\},\{E,C\},\{D,F\},\{A,C\},\{D,E\},\{B,E\},\{A,B\},\{B,C\},\{F,E\},\{E,G\},\{B,D\},\{C,G\},\{F,G\}$$

Im folgenden die Menge E_F in jedem Iterationsschritt und der resultierende Spannbaum nach Ausführung aller Schritte.

- $E_F = \emptyset$
- $E_F = \{\{A,D\}\}$
- $E_F = \{\{A,D\},\{E,C\}\}$
- $E_F = \{\{A,D\},\{E,C\},\{D,F\}\}$
- $E_F = \{\{A,D\},\{E,C\},\{D,F\},\{A,C\}\}$
- $E_F = \{\{A,D\},\{E,C\},\{D,F\},\{A,C\},\{B,E\}\}$
- $E_F = \{\{A,D\},\{E,C\},\{D,F\},\{A,C\},\{B,E\},\{E,G\}\}$

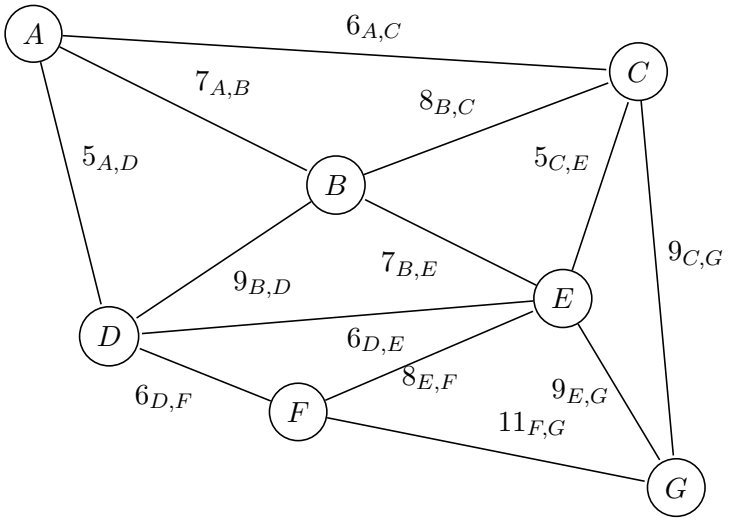


Hinweis: Die hier angegebene Lösung ist nur eine von vielen.

b) Im Algorithmus von Prim wird in jedem Schritt die momentane Zusammenhangkomponente (S) um einen Knoten erweitert, und zwar um den Knoten der durch die Kante mit minimalen Kosten erreichbar ist.

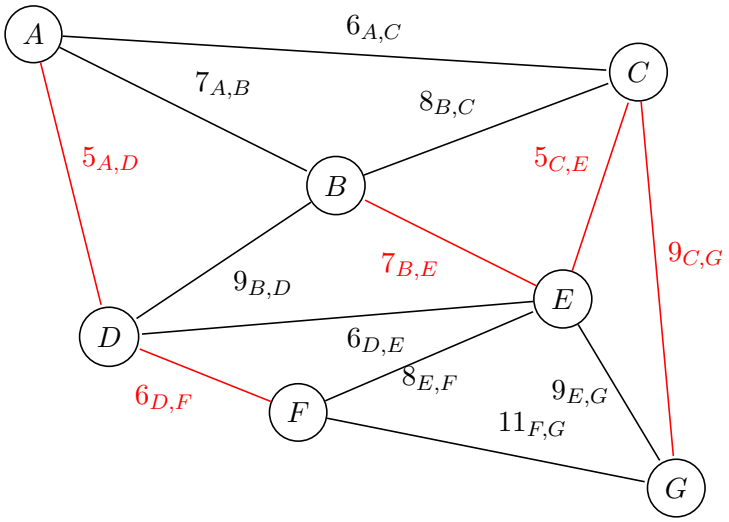
- $E_F = \emptyset$ $S = \{B\}$
- $E_F = \{\{B,E\}\}$ $S = \{B,E\}$
- $E_F = \{\{B,E\},\{E,C\}\}$ $S = \{B,E,C\}$
- $E_F = \{\{B,E\},\{E,C\},\{C,A\}\}$ $S = \{B,E,C,A\}$
- $E_F = \{\{B,E\},\{E,C\},\{C,A\},\{A,D\}\}$ $S = \{B,E,C,A,D\}$
- $E_F = \{\{B,E\},\{E,C\},\{C,A\},\{A,D\},\{D,F\}\}$ $S = \{B,E,C,A,D,F\}$
- $E_F = \{\{B,E\},\{E,C\},\{C,A\},\{A,D\},\{D,F\},\{E,G\}\}$ $S = \{B,E,C,A,D,F,G\}$

c) Im folgenden der Graph mit einer eindeutigen Identität bzg. der Endknoten. Im folgenden nun die

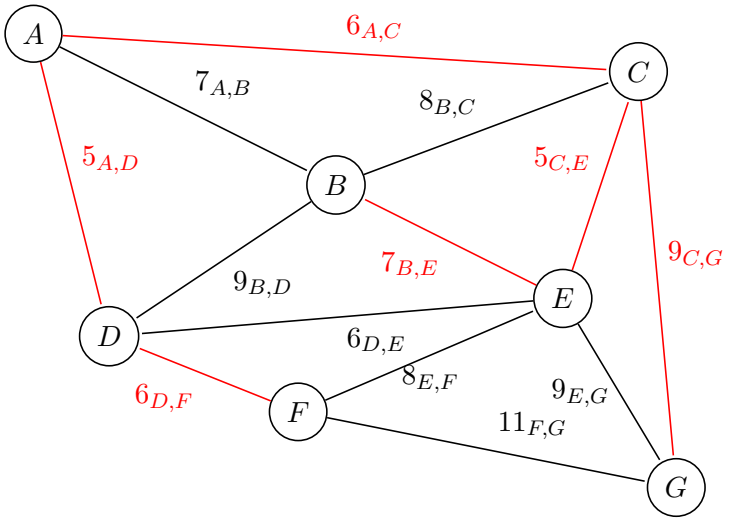


weiteren Schritte des Algorithmus (nur das Resultat wurde in der Aufgabenstellung gefordert):

- Erster Schritt: Es entstehen die Zusammenhangskomonenten $\{A, D, F\}$ und $\{B, E, C, G\}$



- Zweiter Schritt: Obige Zusammenhangkomponenten werden zusammengeführt.



d) Die Laufzeit setzt sich zusammen aus der Sortierung der Kanten ($O(m \log m)$) und dem Test auf Kreisfreiheit in allen m Iterationen ($O(m \cdot \alpha(n))$). Da $\alpha(n)$ eine sehr langsam wachsende Funktion ist (der Funktionswert ist ≤ 4 für quasi jede praktische Anwendung), dominiert die Laufzeit für das Sortieren die Gesamtlaufzeit. Wenn nun aber die Kanten schon sortiert sind, ergibt sich eine Gesamtlaufzeit von $O(m \cdot \alpha(n))$.

Aufgabe 3: Eindeutige Minimale Spannwälder *(4 Punkte)*

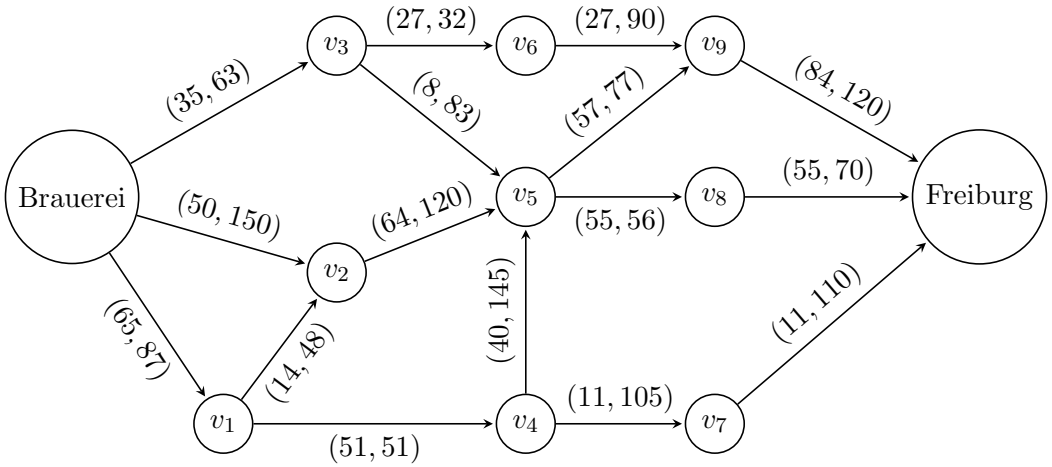
Sei $G = (V, E)$ ein ungerichteter Graph mit **paarweise verschiedenen** Kantenkosten bezüglich einer Kostenfunktion $c : E \to \mathbb{R}$. Zeigen Sie, dass G einen *eindeutigen* minimalen Spannwald hat.

Musterlösung

Seien T und T' zwei minimale Spannbäume (von G) mit Kanten $e_1, \ldots, e_{n-1}$ und $e'_1, \ldots, e'_{n-1}$, jeweils aufsteigend nach Gewicht sortiert. Nehme an es gelte $T \neq T'$. Sei j der größte Index, für den $e_j \neq e'_j$ gilt. Da die Gewichte paarweise verschieden sind, muss auch $c(e_j) \neq c(e'_j)$ gelten. O.b.d.A. sei $c(e_j) < c(e'_j)$. Der Graph $T' \setminus \{e'_j\}$ hat zwei Zusammenhangskomponenten mit Knoten S und $V \setminus S$. Sei e_k eine Kante aus T, welche S und $V \setminus S$ verbindet. Da T' nur eine Kante zwischen S und $V \setminus S$ haben kann, muss $k \leq j$ sein (da $e_k = e'_k$ für $k > j$). Nun ist aber $(T' \setminus \{e'_j\}) \cup \{e_k\}$ wieder ein Spannbaum und da $c(e'_j) > c(e_j) \geq c(e_k)$ hat dieser ein kleineres Gewicht als T' im Widerspruch dazu, dass T' minimal ist.

Aufgabe 4: Flüsse und Schnitte *(6 Punkte)*

Es ist mitten im Sommer und wie jedes Jahr müssen die Getränke erst einmal von der Brauerei im Schwarzwald zu den Häusern der Studenten gelangen. Doch das Getränke-Verteil-System ist alt und ineffizient. Betrachten Sie den nachfolgenden Graphen, der das Getränke-Verteil-System darstellt. Jede Kante ist mit (x/X) beschriftet, wobei x die Menge an Getränketransportern und X die maximal erlaubte Anzahl an Getränketransportern auf dieser Strecke ist.



a) Wie viele Getränketranporter kommen aktuell in Freiburg an? *(1 Punkt)*

b) Zeigen Sie anhand des Residualnetzes, dass der Fluss nicht maximal ist. *(1 Punkt)*

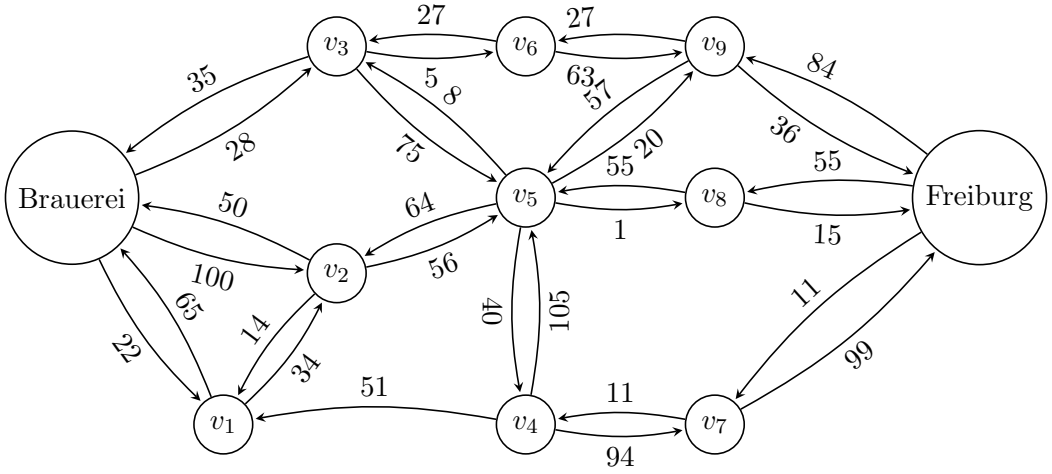
c) Die Brauerei möchte das Getränke-Verteil-System ausbauen. Diese hat Ihnen hierzu eine Vollmacht ausgestellt, mit der Sie Getränketransporter hinzufügen und umverteilen dürfen. Maximieren sie die Anzahl der in Freiburg ankommenden Getränketransporter, indem sie die neuen Flusswerte angeben. *(3 Punkte)*

d) Geben sie einen minimalen $(Brauerei, Freiburg)$-Schnitt an. *(1 Punkt)*

Musterlösung

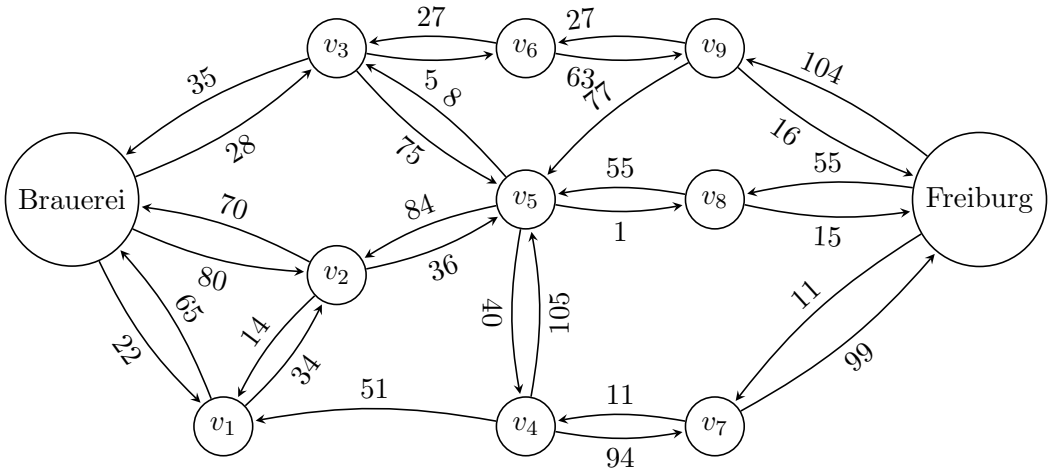
a) Es kommen 150 Transporter an.

b) Nach Beobachtung 9.11 wissen wir dass der Fluss nicht maximal ist solange ein flussvergrößernder Pfad existiert. Ein solcher Pfad im residualnetz (siehe unten) wäre z.B. folgender: *Brauerei* $\to$ $v_2 \to v_5 \to v_9 \to$ *Freiburg*.

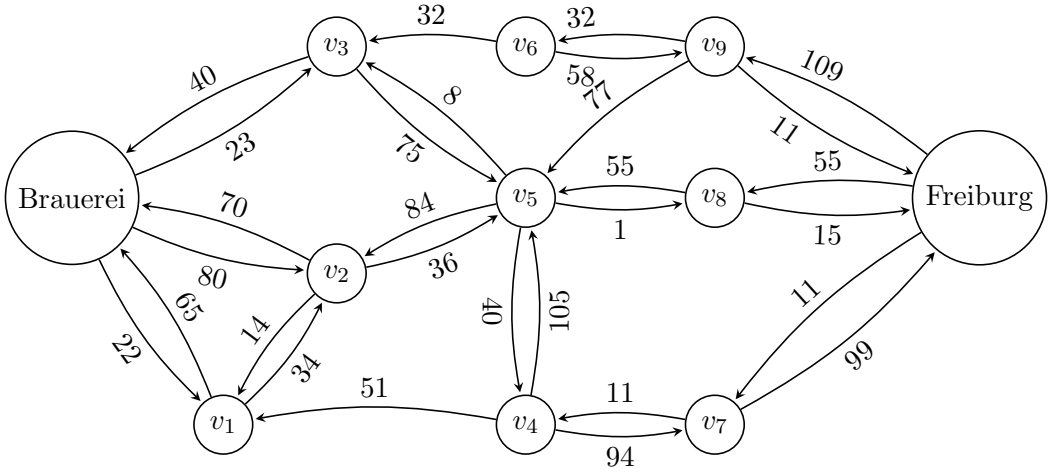


c) Die Idee ist im Residualnetz so lange flussvergrößernde Pfade zu finden bis es keine mehr gibt.

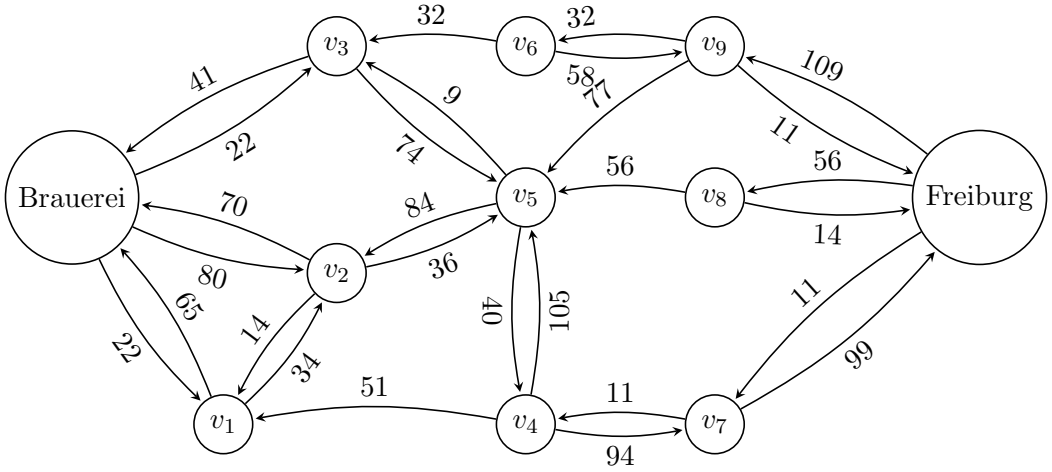
- *Brauerei* $\to v_2 \to v_5 \to v_9 \to$ *Freiburg*:



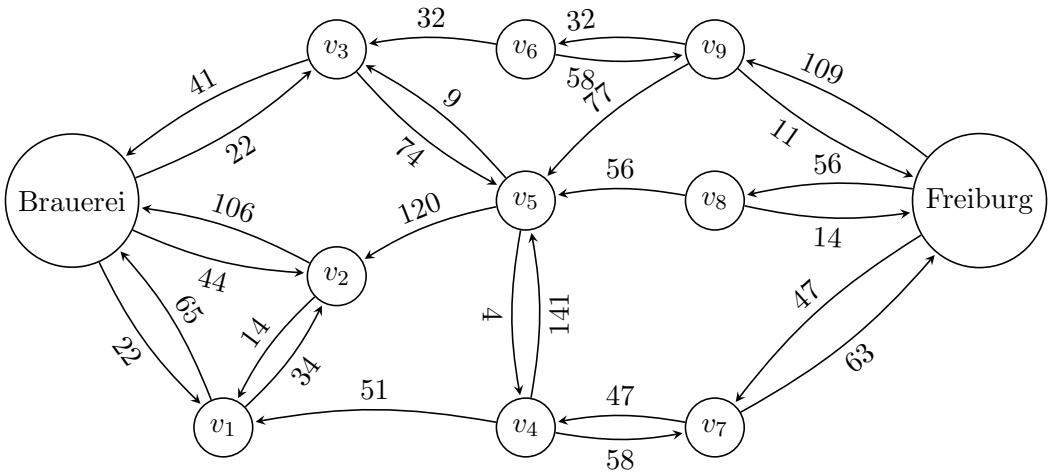
- *Brauerei* $\rightarrow v_3 \rightarrow v_6 \rightarrow v_9 \rightarrow$ *Freiburg*:



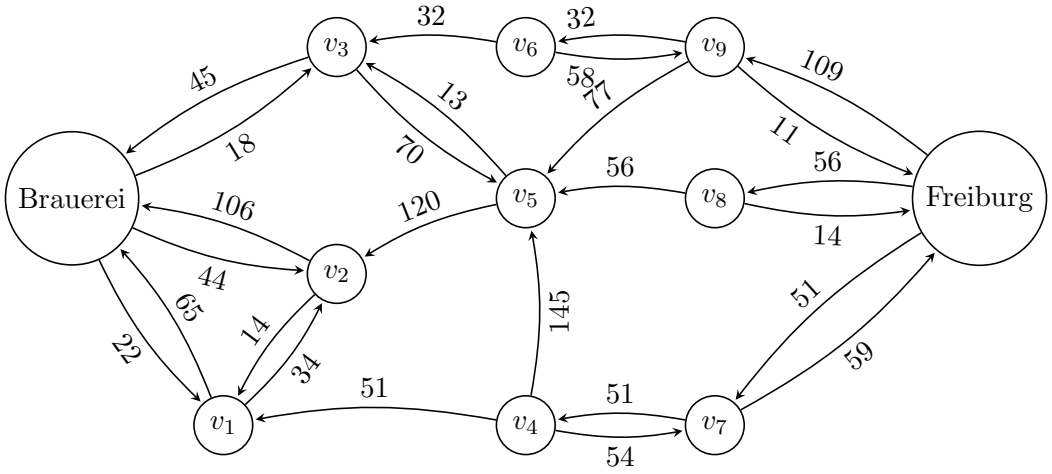
- *Brauerei* $\rightarrow v_3 \rightarrow v_5 \rightarrow v_8 \rightarrow$ *Freiburg*:



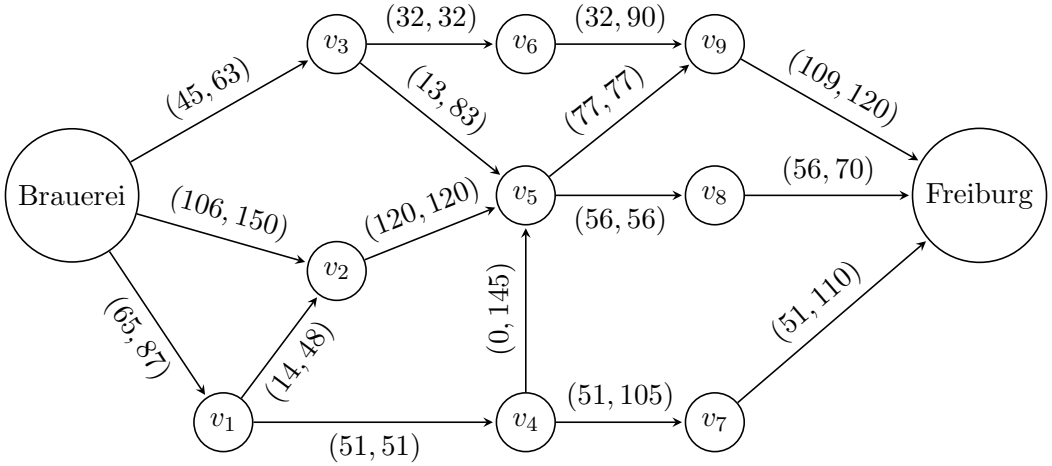
- *Brauerei* $\rightarrow v_2 \rightarrow v_5 \rightarrow v_4 \rightarrow v_7 \rightarrow$ *Freiburg*:



- letzter Pfad: *Brauerei* $\to v_3 \to v_5 \to v_4 \to v_7 \to$ *Freiburg*:



Nun können wir den letzten Residualgraph wieder zurück transformieren: Der maximale Fluss ist



dementsprechend $45 + 106 + 65 = 216 = 109 + 56 + 51$.

d) Der minimale Schnitt muss wie der maximale Fluss einen Wert von 216 haben. Das erreichen wir zum Beispiel durch folgende Belegung: $S = \{Brauerei, v_1, v_2, v_3, v_5\}$ (T ist entsprechend $V \setminus S$).