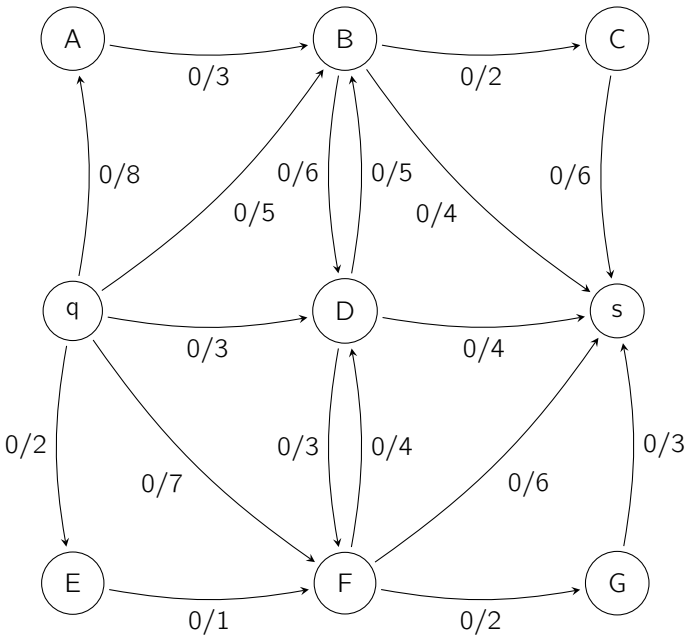
Prof. Dr. Leif Kobbelt
Stefan Dollase, Ira Fesefeldt, Alexandra Heuschling, Gregor Kobsik

Lösung - Übung 11

Tutoraufgabe 1 (Ford-Fulkerson Methode):

Betrachten Sie das folgende Flussnetzwerk mit Quelle q und Senke s:

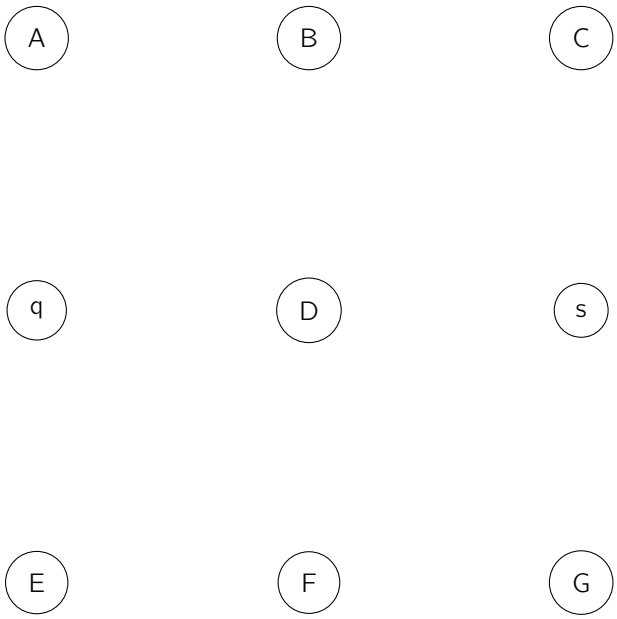


a) Berechnen Sie den maximalen Fluss in diesem Netzwerk mithilfe der *Ford-Fulkerson Methode*. Geben Sie dazu *jedes Restnetzwerk* sowie *nach jeder Flussvergrößerung* den aktuellen Zustand des Flussnetzwerks an. Die vorgegebene Anzahl an Lösungsschritten muss nicht mit der benötigten Anzahl solcher Schritte übereinstimmen.

b) Geben Sie außerdem den *Wert des maximalen Flusses* an.

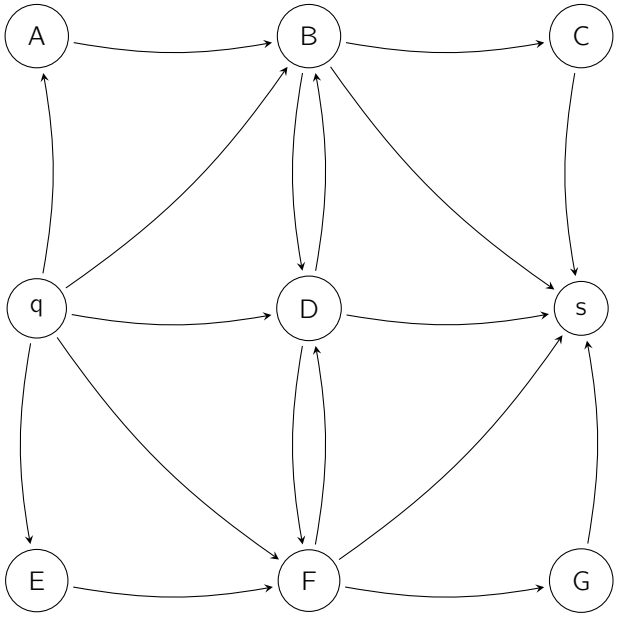
Schritt 1:

Restnetzwerk:



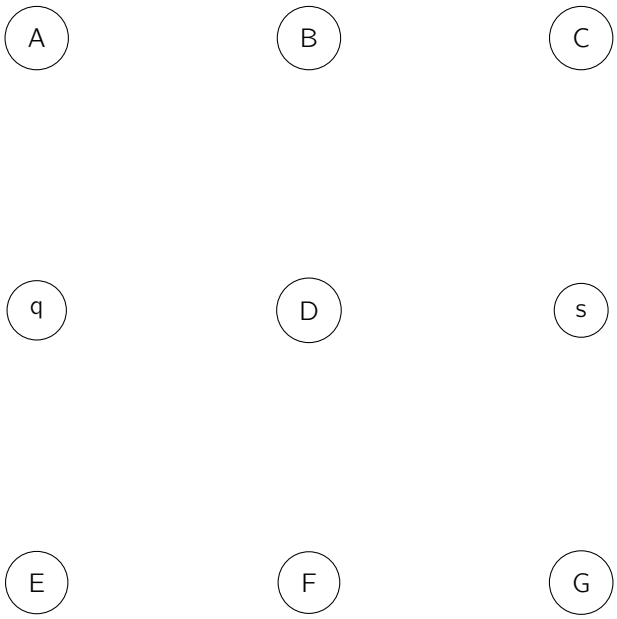
Schritt 2:

Nächstes Flussnetzwerk mit aktuellem Fluss:



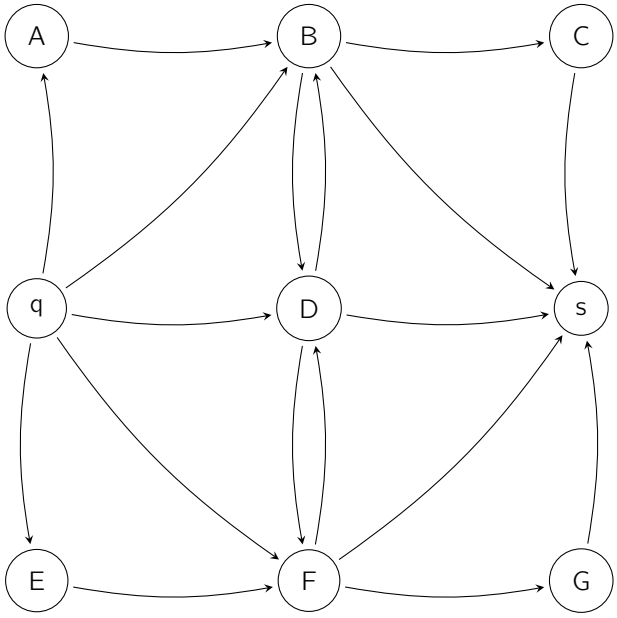
Schritt 3:

Restnetzwerk:



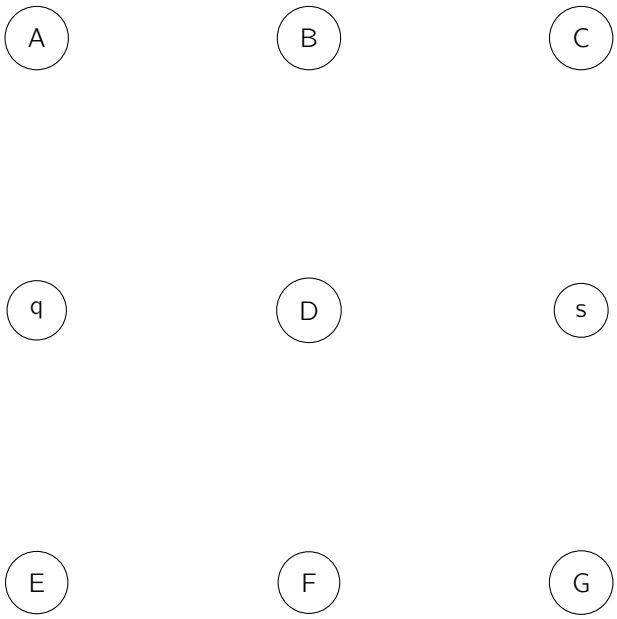
Schritt 4:

Nächstes Flussnetzwerk mit aktuellem Fluss:



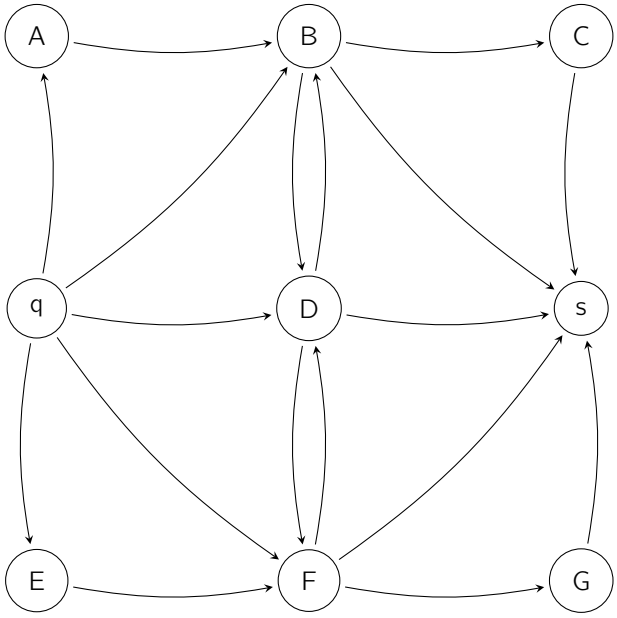
Schritt 5:

Restnetzwerk:



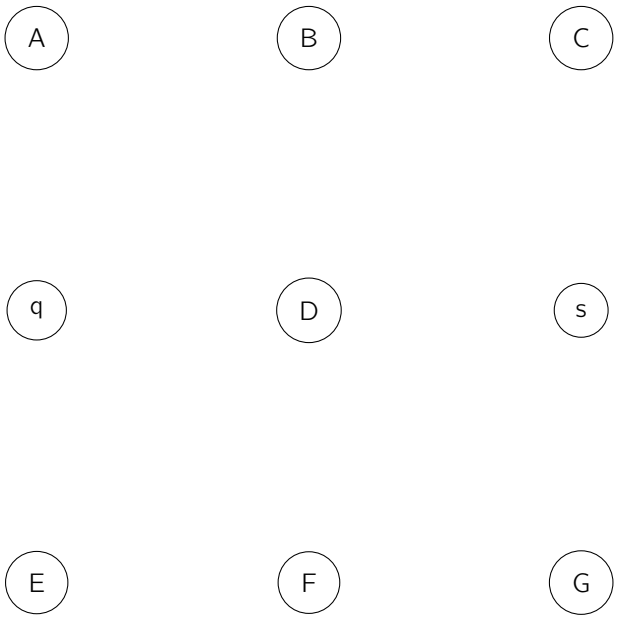
Schritt 6:

Nächstes Flussnetzwerk mit aktuellem Fluss:



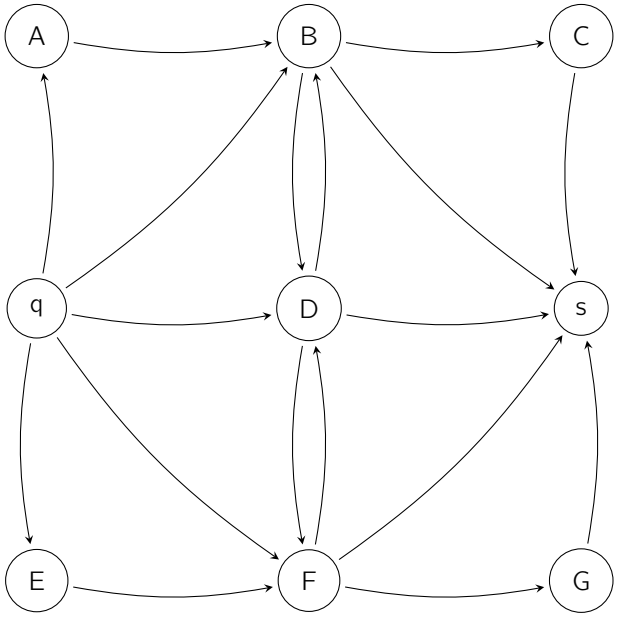
Schritt 7:

Restnetzwerk:



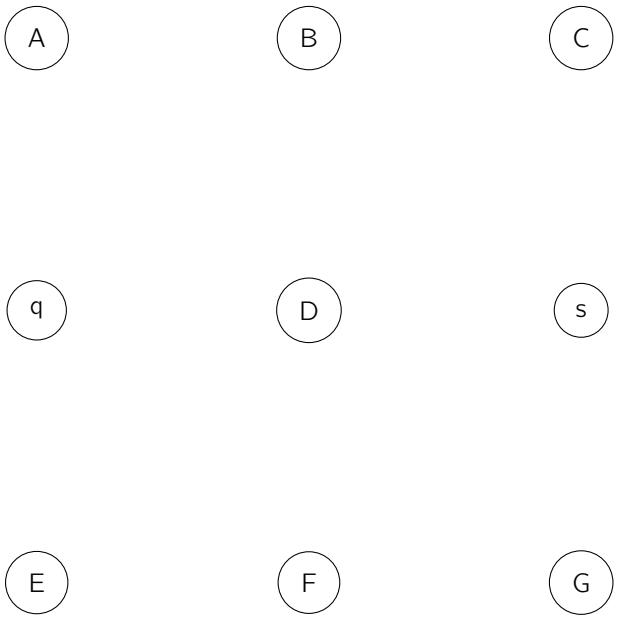
Schritt 8:

Nächstes Flussnetzwerk mit aktuellem Fluss:



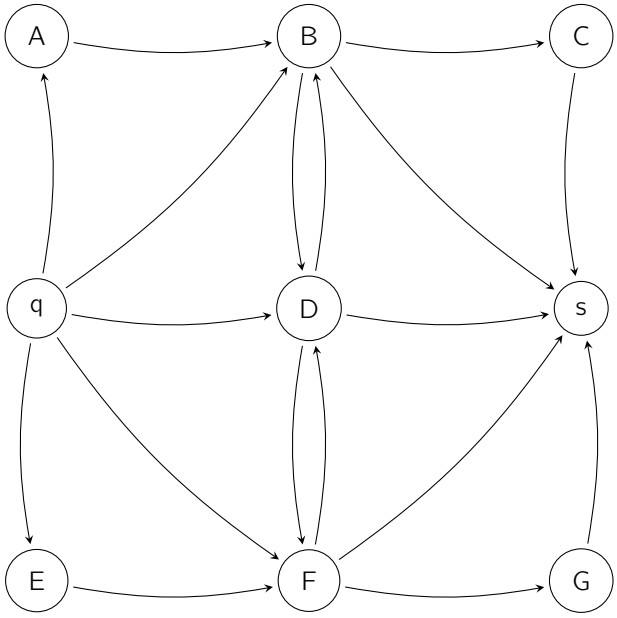
Schritt 9:

Restnetzwerk:



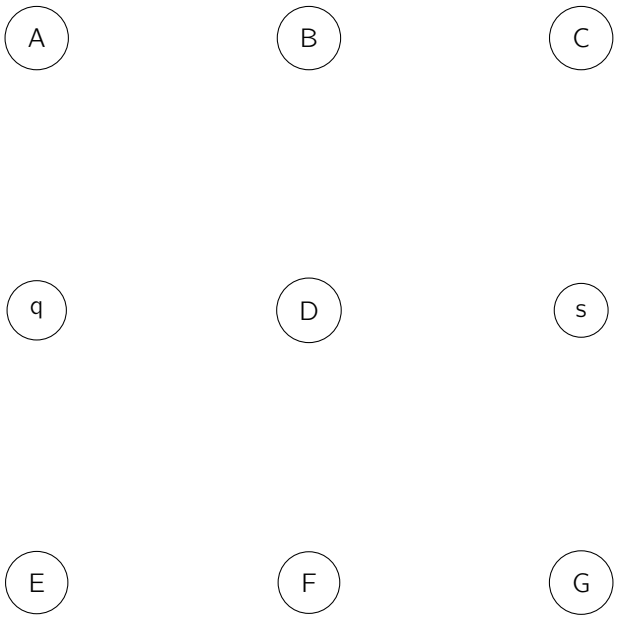
Schritt 10:

Nächstes Flussnetzwerk mit aktuellem Fluss:



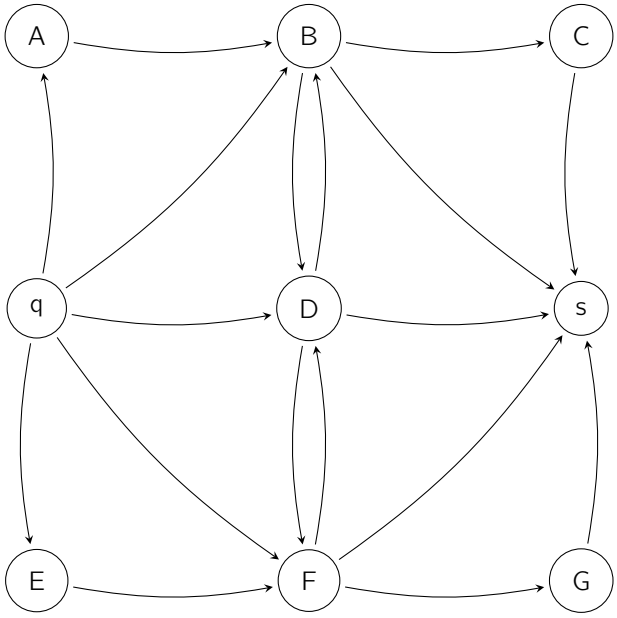
Schritt 11:

Restnetzwerk:



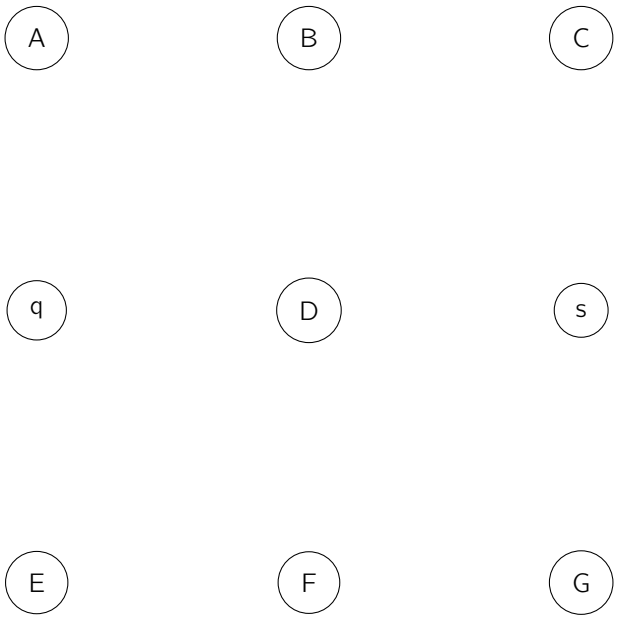
Schritt 12:

Nächstes Flussnetzwerk mit aktuellem Fluss:



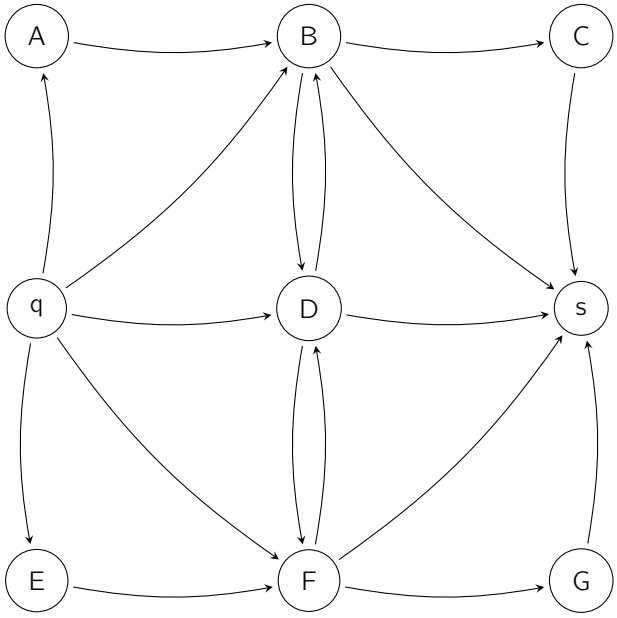
Schritt 13:

Restnetzwerk:



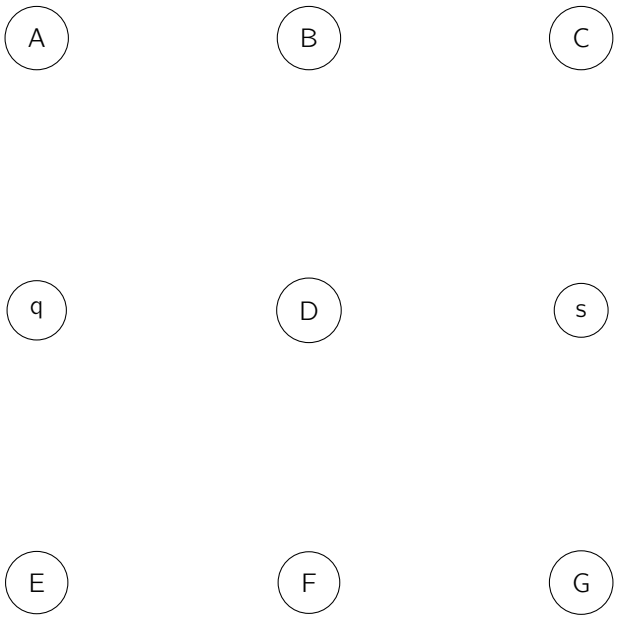
Schritt 14:

Nächstes Flussnetzwerk mit aktuellem Fluss:



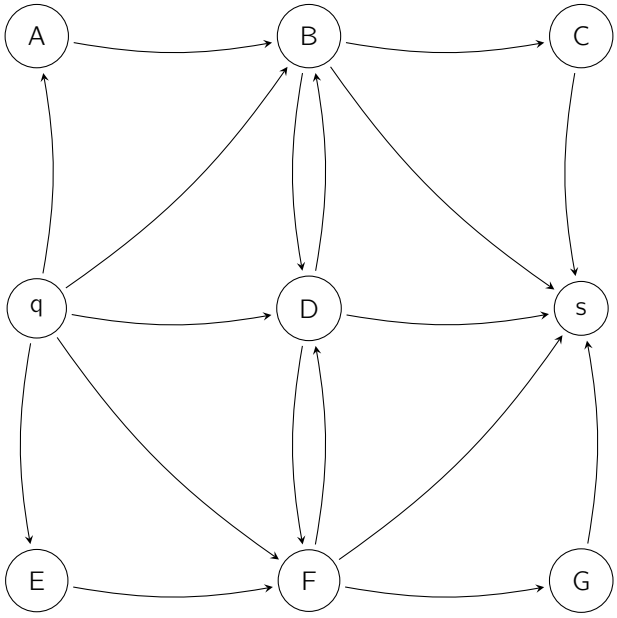
Schritt 15:

Restnetzwerk:



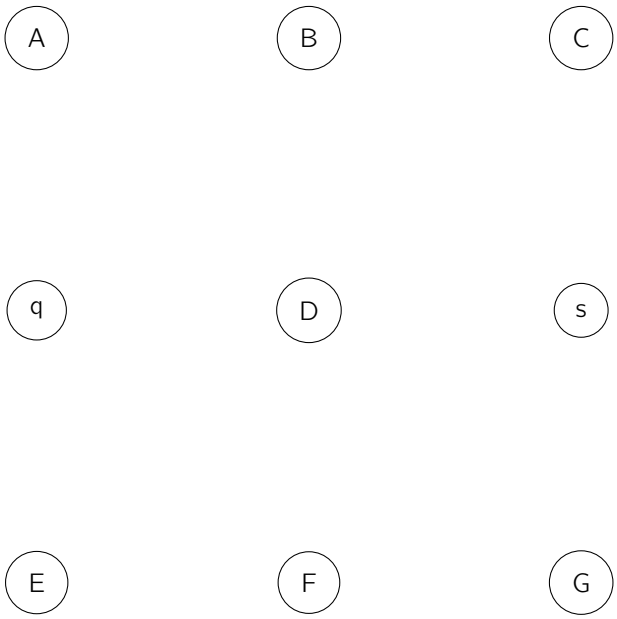
Schritt 16:

Nächstes Flussnetzwerk mit aktuellem Fluss:



Schritt 17:

Restnetzwerk:



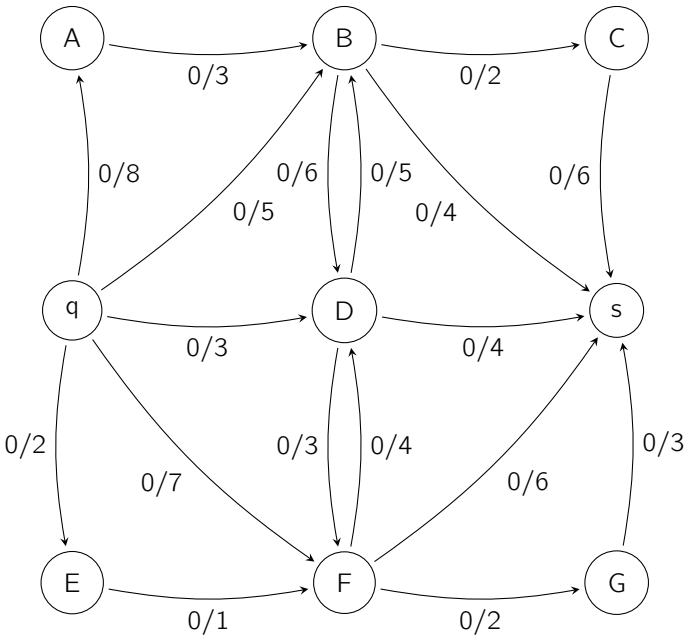
Der maximale Fluss hat den Wert:

Lösung

a)

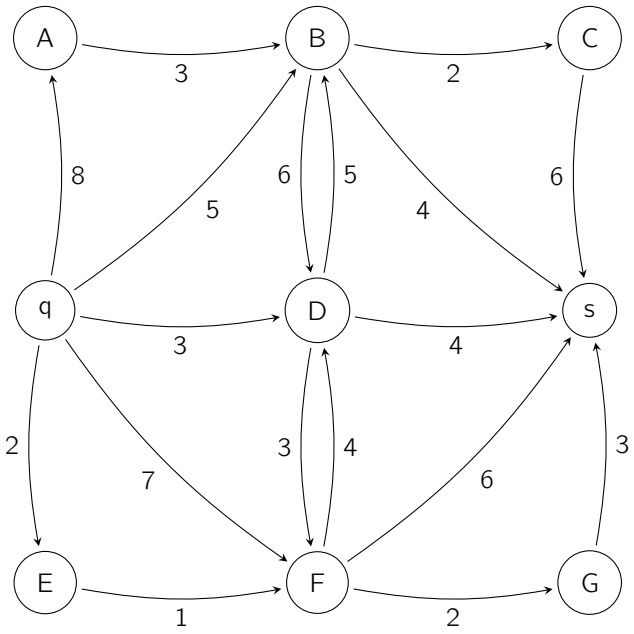
Schritt 0:

Initiales Flussnetzwerk:



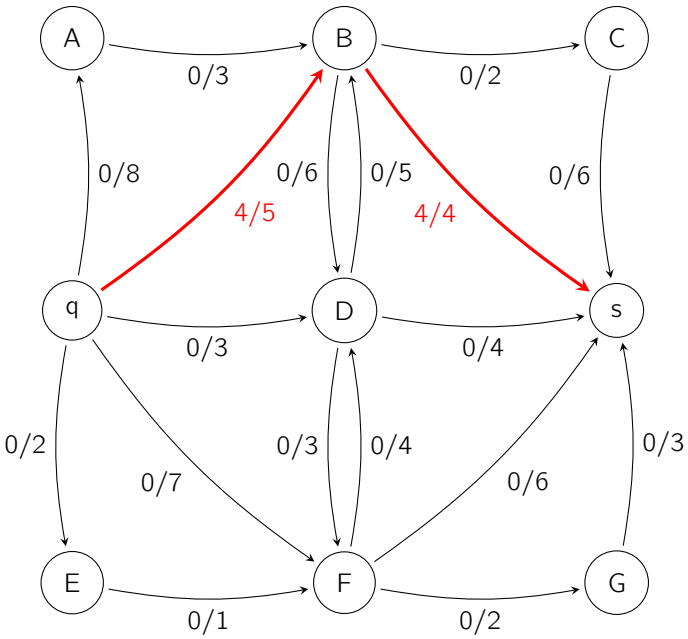
Schritt 1:

Restnetzwerk:



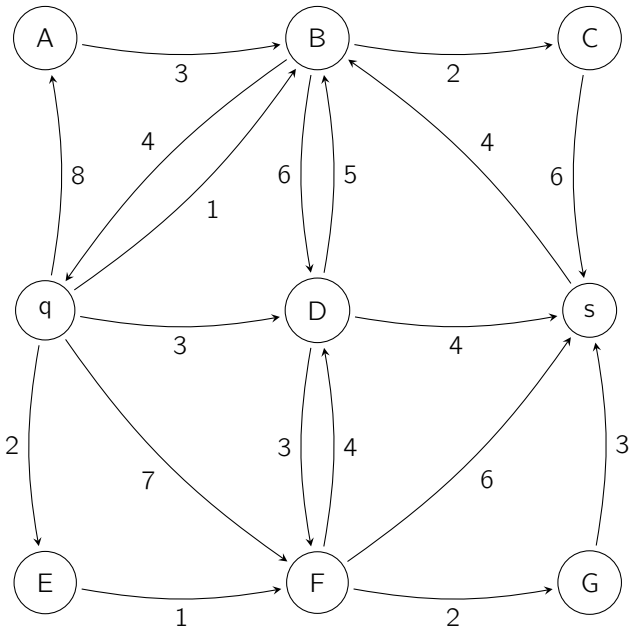
Schritt 2:

Nächstes Flussnetzwerk mit aktuellem Fluss:



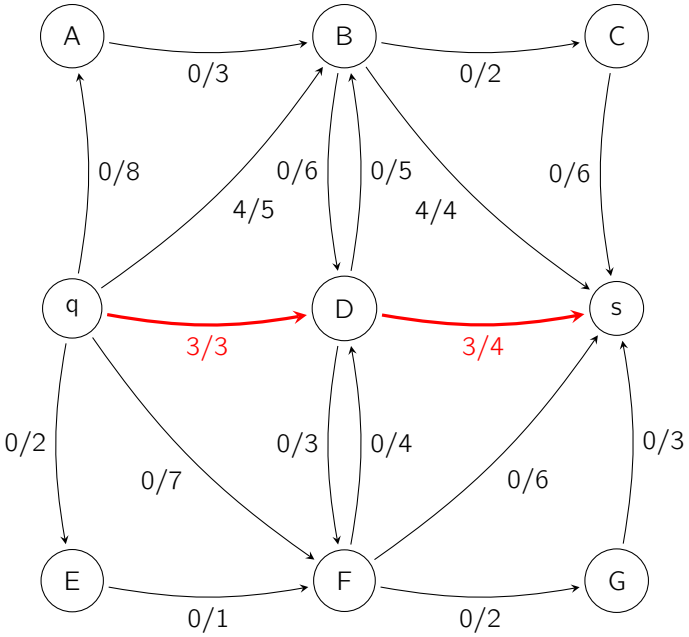
Schritt 3:

Restnetzwerk:



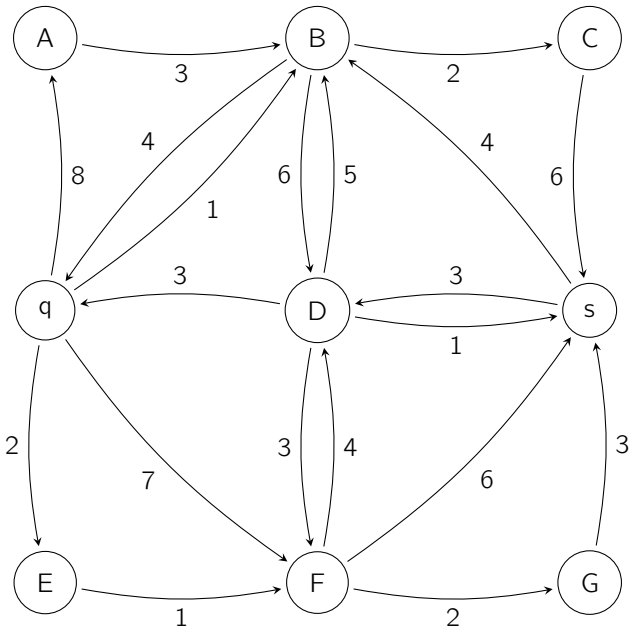
Schritt 4:

Nächstes Flussnetzwerk mit aktuellem Fluss:



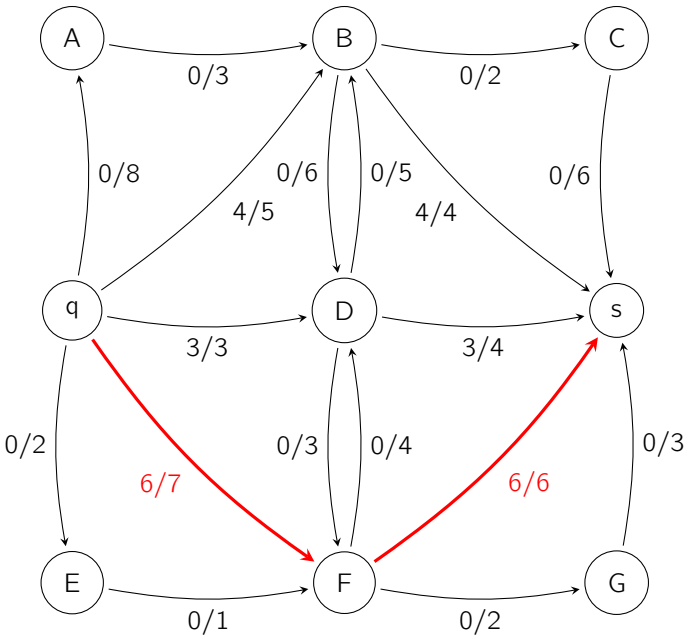
Schritt 5:

Restnetzwerk:



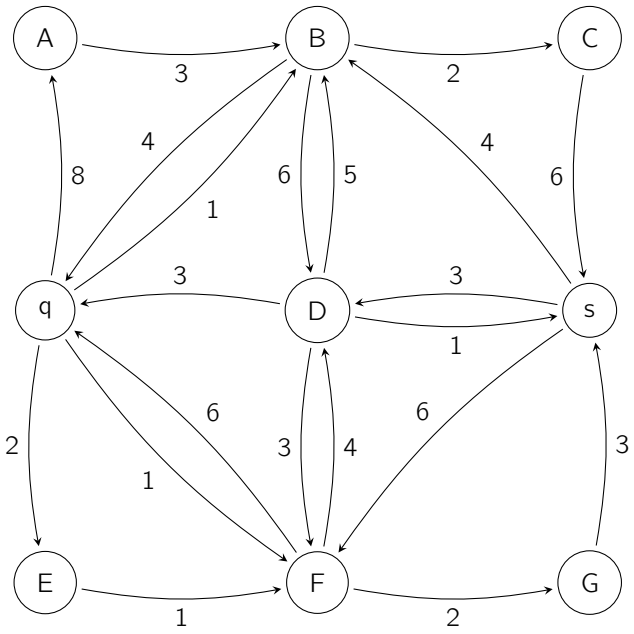
Schritt 6:

Nächstes Flussnetzwerk mit aktuellem Fluss:



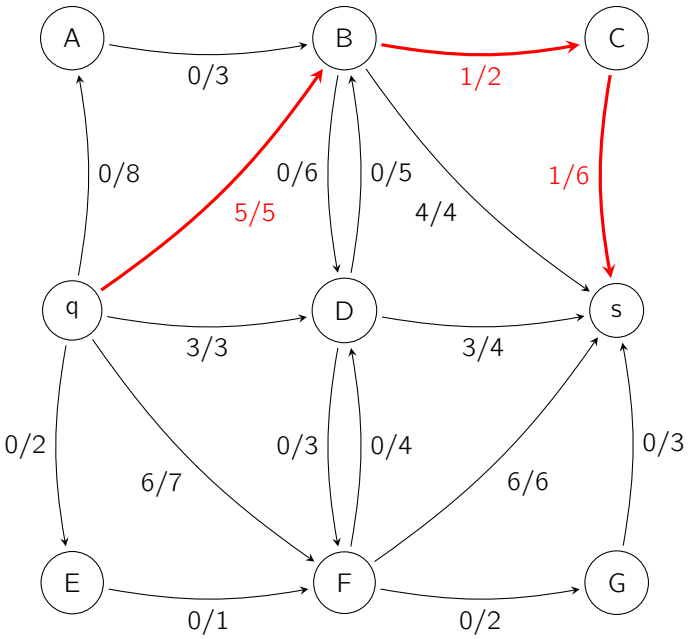
Schritt 7:

Restnetzwerk:



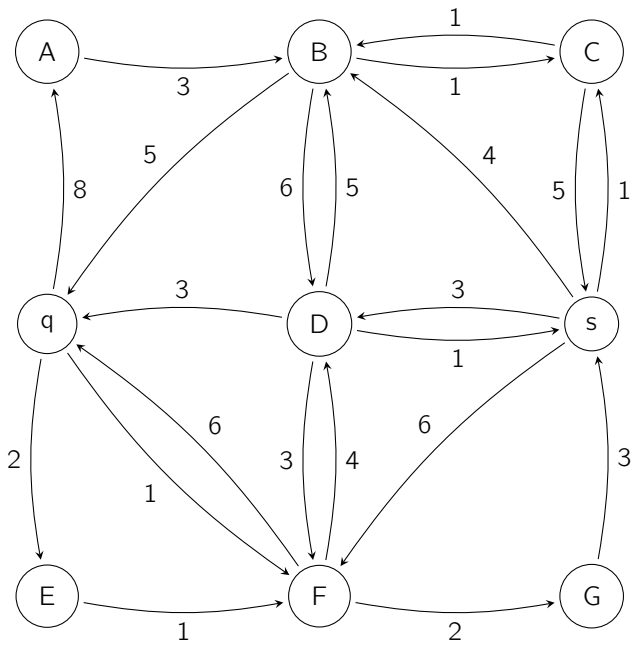
Schritt 8:

Nächstes Flussnetzwerk mit aktuellem Fluss:



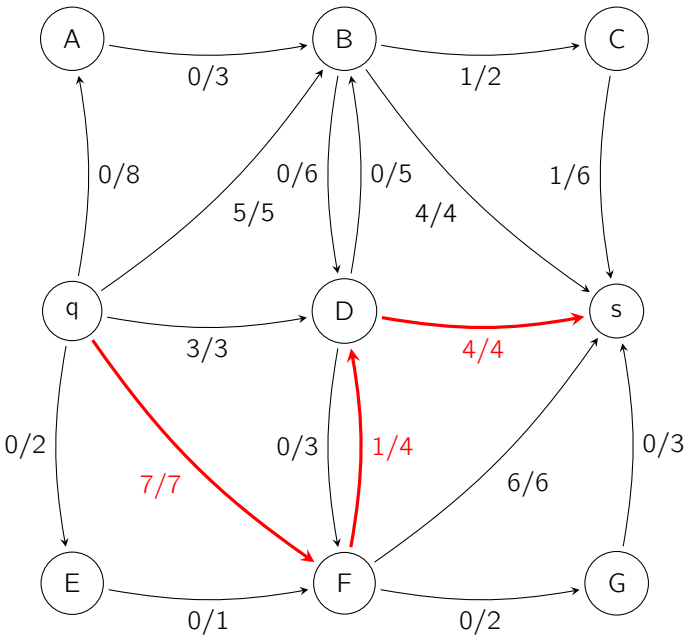
Schritt 9:

Restnetzwerk:



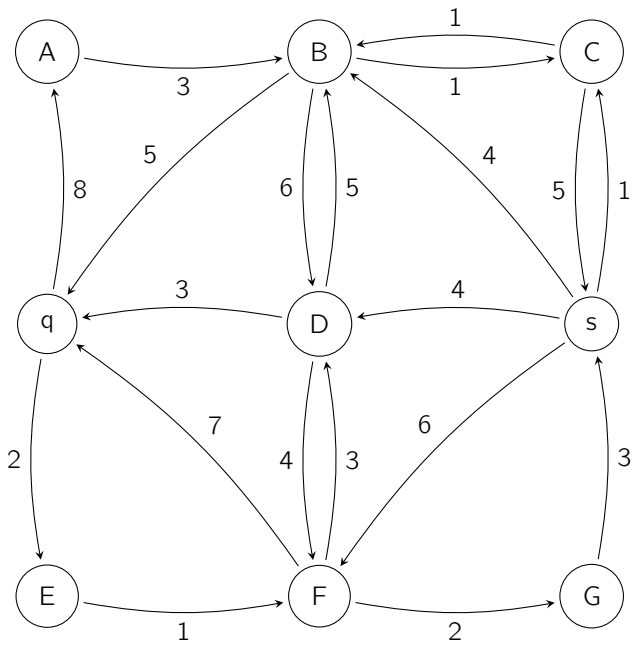
Schritt 10:

Nächstes Flussnetzwerk mit aktuellem Fluss:



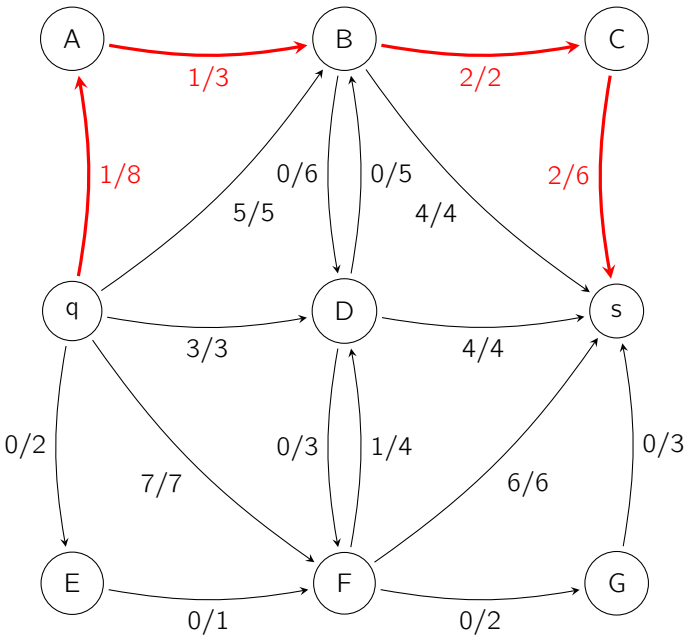
Schritt 11:

Restnetzwerk:



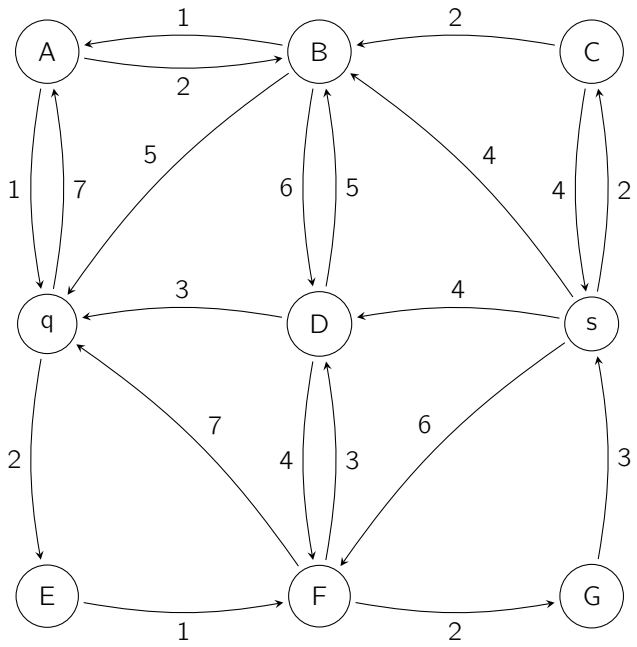
Schritt 12:

Nächstes Flussnetzwerk mit aktuellem Fluss:



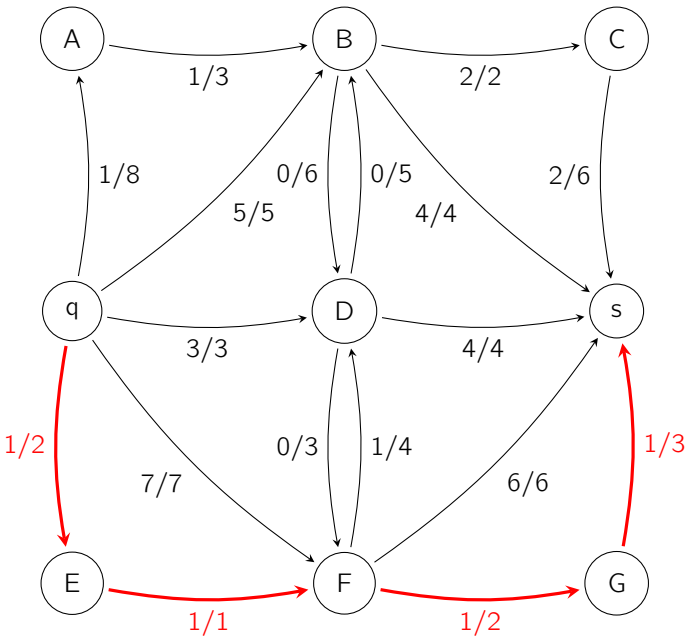
Schritt 13:

Restnetzwerk:



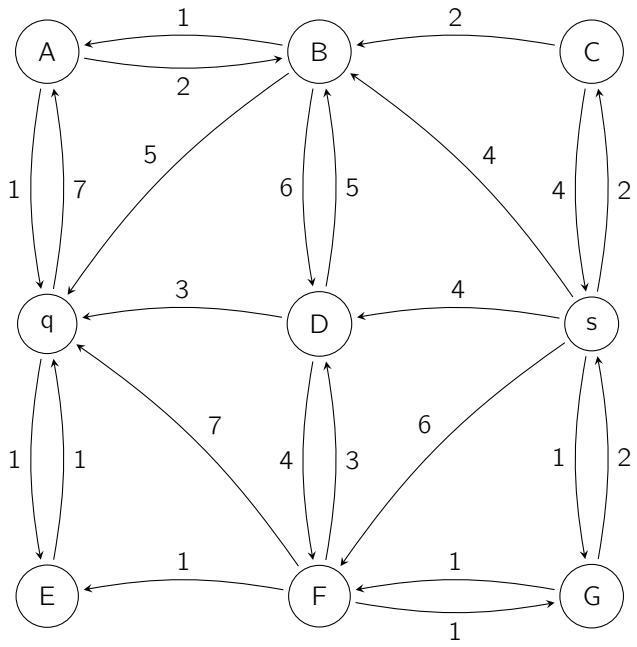
Schritt 14:

Nächstes Flussnetzwerk mit aktuellem Fluss:



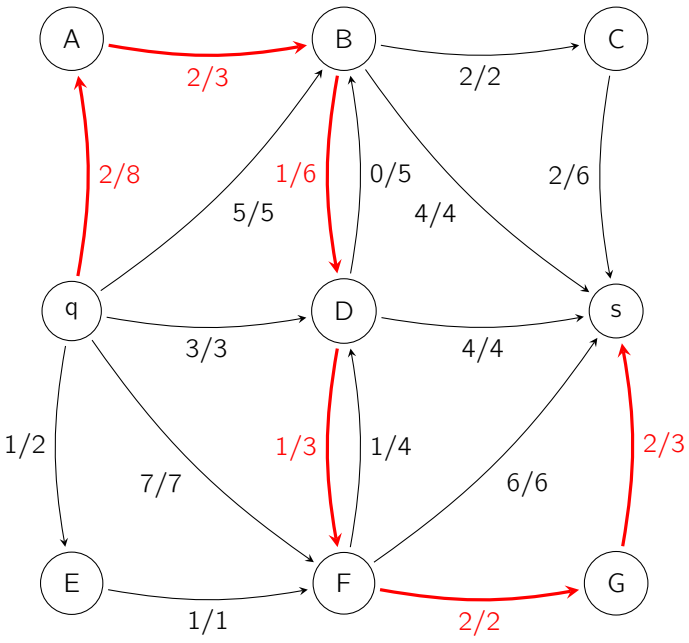
Schritt 15:

Restnetzwerk:



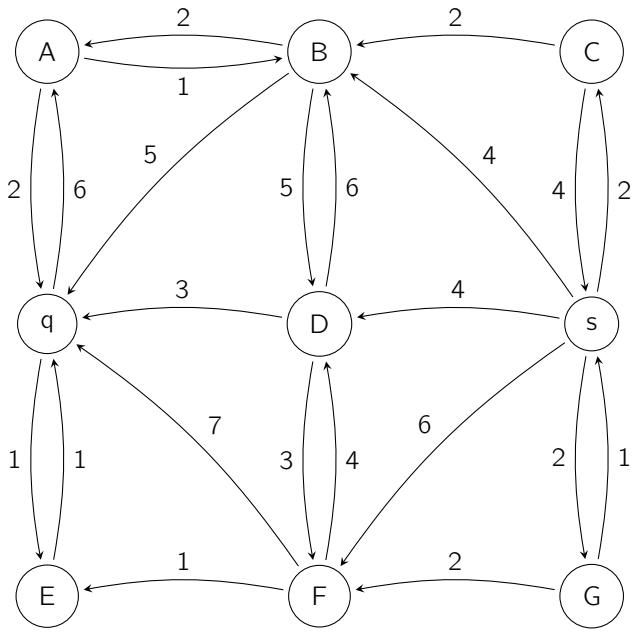
Schritt 16:

Nächstes Flussnetzwerk mit aktuellem Fluss:



Schritt 17:

Restnetzwerk:



b) Der maximale Fluss hat den Wert: 18

Tutoraufgabe 2 (Modellierung mit Flussnetzwerken – Kartenspieler):

Sei K die Menge aller Kartenspieler. Es gibt Partien $S_1, \dots S_m \subseteq K$ die aus den Kartenspielern bestehen, die an dieser Partie teilnehmen wollen. Zu jeder Partie S_i muss es einen Organisator $o_i \in S_i$ geben, der an der Partie teilnimmt. Jeder Kartenspieler $T_i \in K$ ist bereit bis zu $f(T_i)$ viele Partien zu organisieren. Die Kartenspieler K, die Funktion $f : K \mapsto \{0, \dots m\}$ und die Partien $S_1, \dots S_m$ sind bekannt.

a) Wie kann man effizient Organisatoren den Partien zuordnen? Geben Sie eine Beschreibung Ihres Verfahrens an.

b) Welche Laufzeit hat das Verfahren? Begründen Sie Ihre Antwort.

Lösung

a) Wir nutzen hierfür Flussnetzwerke und die Ford-Fulkerson-Methode. Wir bezeichnen die Spieler als T_1 bis T_n und die Partien als S_1 bis S_m. Wir kodieren das Problem folgendermaßen als Fluss $G = (V, E, g)$:

- $V = \{q, s, T_1 \dots T_n, S_1 \dots S_m\}$ wobei T_i der i-te Teilnehmer und S_j die j-te Partie ist.
- $(q, T_i) \in E$ mit $g(q, T_i) = f(T_i)$ kodiert, dass jeder Teilnehmer T_i bereit ist, $f(T_i)$ viele Partien zu organisieren.
- $(T_i, S_j) \in E$ mit $g(T_i, S_j) = 1$ falls $T_i \in S_j$ kodiert, dass T_i in S_j mitspielt und daher die Organisation übernehmen könnte.
- $(S_j, s) \in E$ mit $g(S_j, s) = 1$ kodiert, dass jede Partie höchstens einen Organisator haben sollte.

Ein Fluss auf der Kante (T_i, S_j) gibt nun an, dass T_i die Partie S_j organisiert. Dies kann nur maximal eine Person sein, da nur ein Gewicht aus S_j rausfließt. Eine Person T_i kann nicht mehr als $f(T_i)$ Partien organisieren, da nicht mehr Fluss von q in den Knoten eingeht. Ein maximaler Fluss von m wäre erreicht, wenn jede Partie genau ein Organisator bekommt - dann wäre nämlich die Kapazität aller Kanten (S_j, s) ausgeschöpft.

b) Die Komplexität dieses Algorithmus ist genau die Komplexität der Ford-Fulkerson-Methode, da die Flusstransformation in linearer Zeit abhängig von der Eingabe zu berechnen ist.

Tutoraufgabe 3 (Modellierung mit Flussnetzwerken – ISS):

Die internationale Raumstation ISS steht auch Weltraumtouristen offen. Sie sollen nun entscheiden, welche Touristen Sie mitnehmen wollen, um möglichst viel Geld zu verdienen.
Gegeben sind Kandidaten $K_1, \ldots, K_n$, welche jeweils bereit sind, $k_1, \ldots, k_n$ US-Dollar zu zahlen. Allerdings sind sie anspruchsvoll und erwarten auf der ISS auch ein Unterhaltungsprogramm (der Erstbesucher Cameron wollte zum Beispiel einen Weltraumspaziergang machen). Zu diesem Zweck stehen eine Menge „Attraktionen“ $Z_1, \ldots, Z_m$ zur Verfügung. Bei der Bereitstellung einer Attraktion zur ISS entstehen allerdings jeweils Kosten $z_1, \ldots, z_m$. Der Kandidat K_i ist nur bereit zu zahlen, wenn die Attraktionen $R_i \subseteq \{Z_1, \ldots, Z_m\}$ bereit gestellt werden.

a) Entwerfen Sie einen effizienten Algorithmus, der eine Menge von Kandidaten auswählt, um die Einnahmen (also die gezahlten Gebühren der Touristen minus die Kosten für die Attraktionen) zu maximieren. Jede Attraktion muss nur einmal organisiert werden, selbst wenn mehrere es benutzen wollen.

b) Welche Laufzeit hat das Verfahren? Begründen Sie Ihre Antwort.

Lösung

a) Wir konstruieren auf der Knotenmenge $\{q, s\} \cup \{K_1, \ldots, K_n\} \cup \{Z_1, \ldots, Z_m\}$ das folgende Flussproblem:

a) Von der Quelle q läuft zu jedem Knoten K_i eine Kante mit Kapazität k_i.

b) Für jedes $Z_j \in R_i$ läuft eine Kante mit unendlicher Kapazität von K_i nach Z_j.

c) Von jedem Knoten Z_i läuft eine Kante mit Kapazität z_i in die Senke s.

Jeder (nicht unendliche) Schnitt im oben beschriebenen Netzwerk stellt eine Lösung dar: Wir organisieren alle Attraktionen deren Kante geschnitten wird und nehmen alle Kandidaten mit deren Kante *nicht* geschnitten wird. Da unendliche Kanten nicht im Schnitt liegen, sind alle Wünsche erfüllt.

Sei $M = \sum_{i=1}^{n} k_i$ der maximal mögliche Gewinn. Die gestrichenen Kanten entsprechen jeweils der Differenz zu M (durch Verzicht auf einen Touristen oder Verzicht auf die Ersparnis durch Nichtorganisation einer Attraktion).

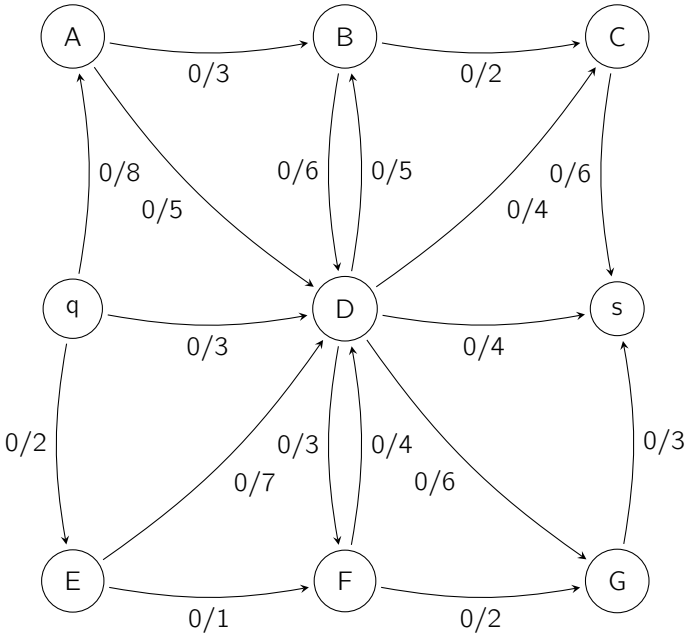
Ein Min-Cut entspricht somit einer Lösung mit maximalem Gewinn. Ein Min-Cut für dieses Netzwerk ist natürlich endlich, da es endliche Schnitte gibt (beispielsweise die Menge aller Kanten, die in q beginnen).

Um einen Min-Cut im obigen konstrukierten Flussnetzwerk zu berechnen, berechnen wir zunächst einen maximalen Fluss f. Anschließend berechnen wir im Restnetzwerk G_f alle von q erreichbaren Knoten und speichern diese in der Menge Q, beispielsweise mithilfe einer Tiefensuche. Der Schnitt $(Q, V \setminus Q)$ ist nun ein Min-Cut.

b) Die Komplexität dieses Algorithmus ist genau die Komplexität der Ford-Fulkerson-Methode, da die Erreichbarkeitsanalyse nur zusätzlich eine Tiefensuche benötigt.

Aufgabe 4 (Ford-Fulkerson Methode): 8 + 2 = 10 Punkte

Betrachten Sie das folgende Flussnetzwerk mit Quelle q und Senke s:

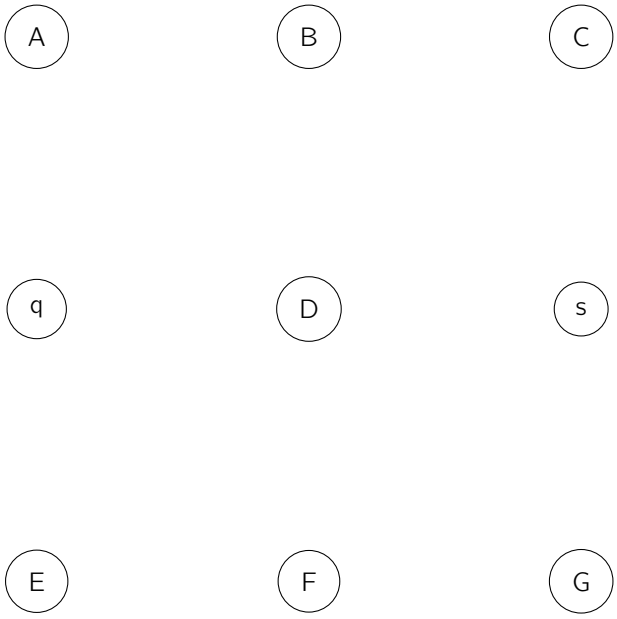


a) Berechnen Sie den maximalen Fluss in diesem Netzwerk mithilfe der *Ford-Fulkerson Methode*. Geben Sie dazu *jedes Restnetzwerk* sowie *nach jeder Flussvergrößerung* den aktuellen Zustand des Flussnetzwerks an. Die vorgegebene Anzahl an Lösungsschritten muss nicht mit der benötigten Anzahl solcher Schritte übereinstimmen.

b) Geben Sie außerdem den *Wert des maximalen Flusses* an.

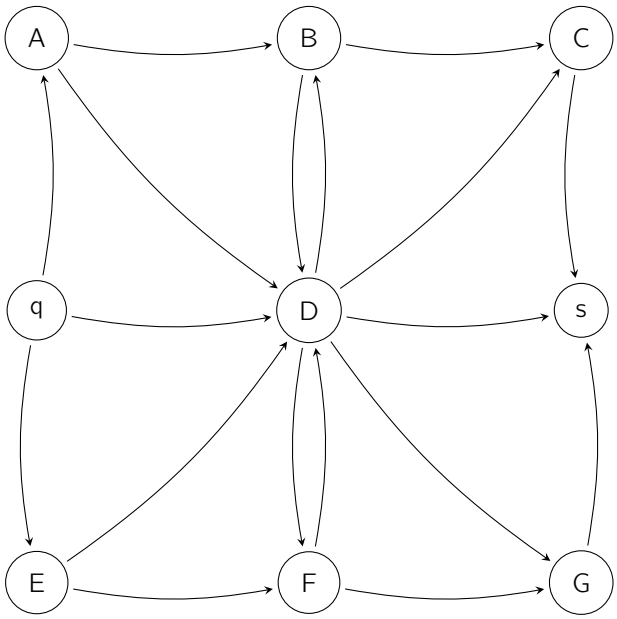
Schritt 1:

Restnetzwerk:



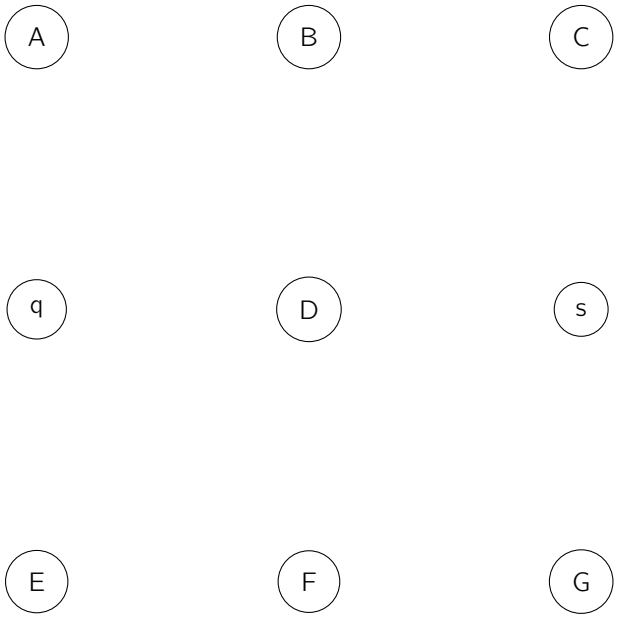
Schritt 2:

Nächstes Flussnetzwerk mit aktuellem Fluss:



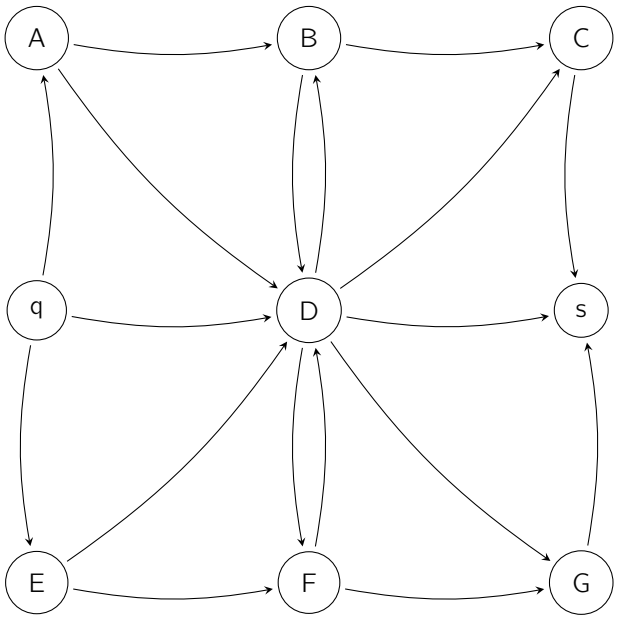
Schritt 3:

Restnetzwerk:



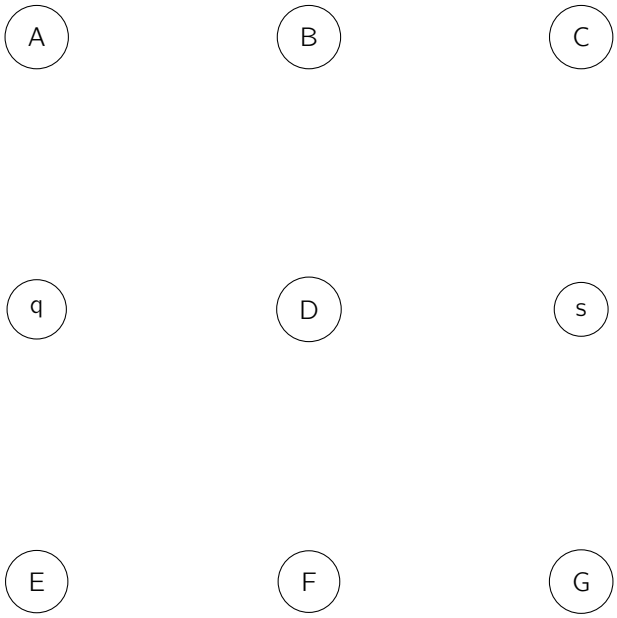
Schritt 4:

Nächstes Flussnetzwerk mit aktuellem Fluss:



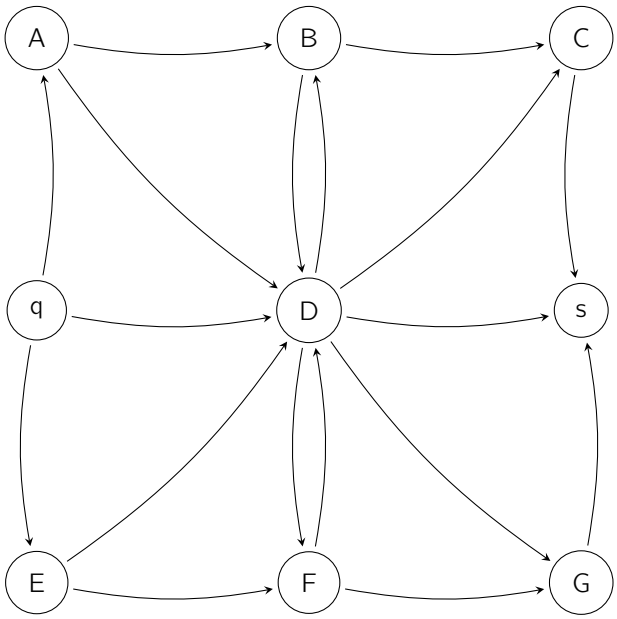
Schritt 5:

Restnetzwerk:



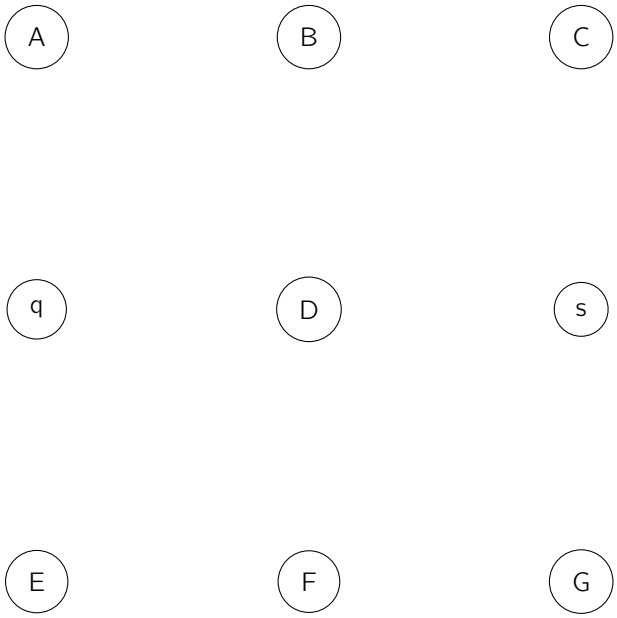
Schritt 6:

Nächstes Flussnetzwerk mit aktuellem Fluss:



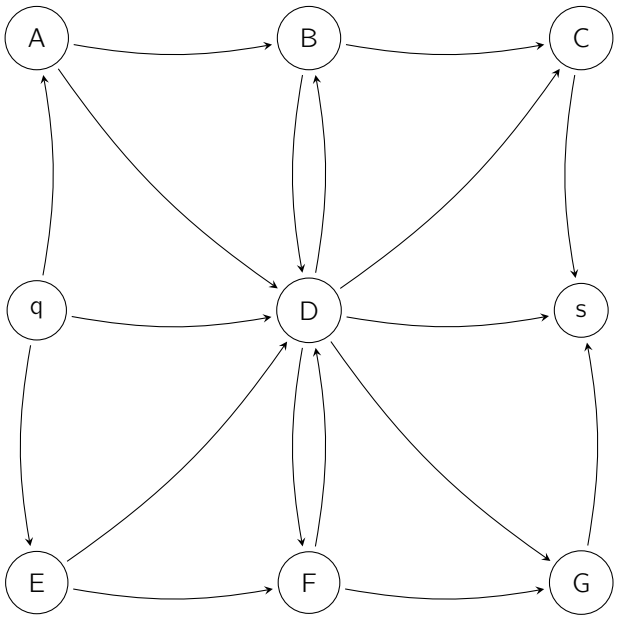
Schritt 7:

Restnetzwerk:



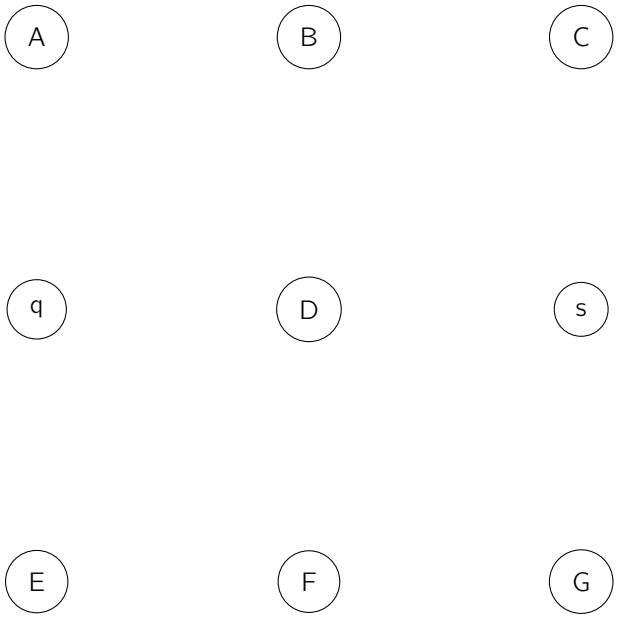
Schritt 8:

Nächstes Flussnetzwerk mit aktuellem Fluss:



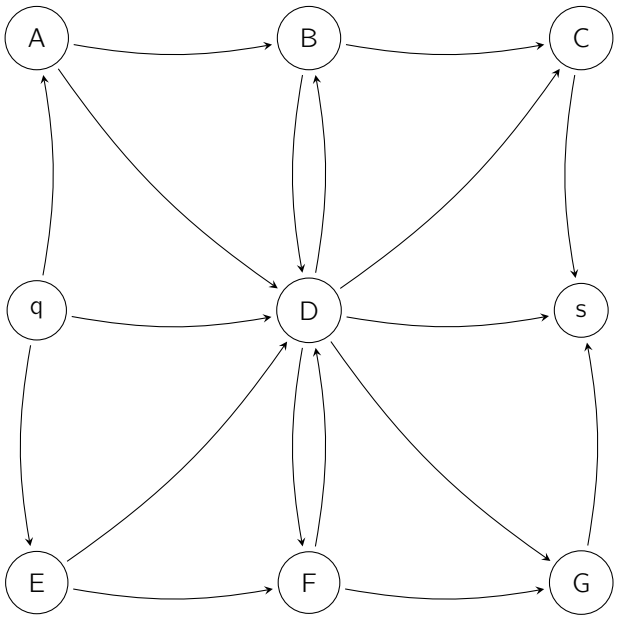
Schritt 9:

Restnetzwerk:



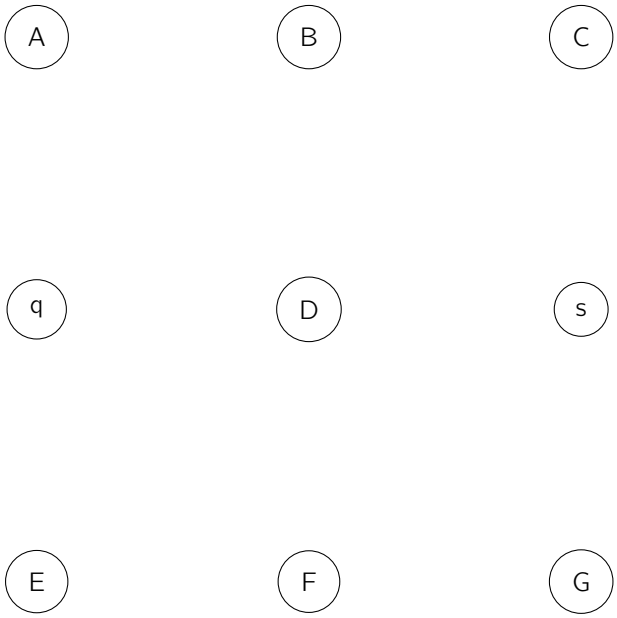
Schritt 10:

Nächstes Flussnetzwerk mit aktuellem Fluss:



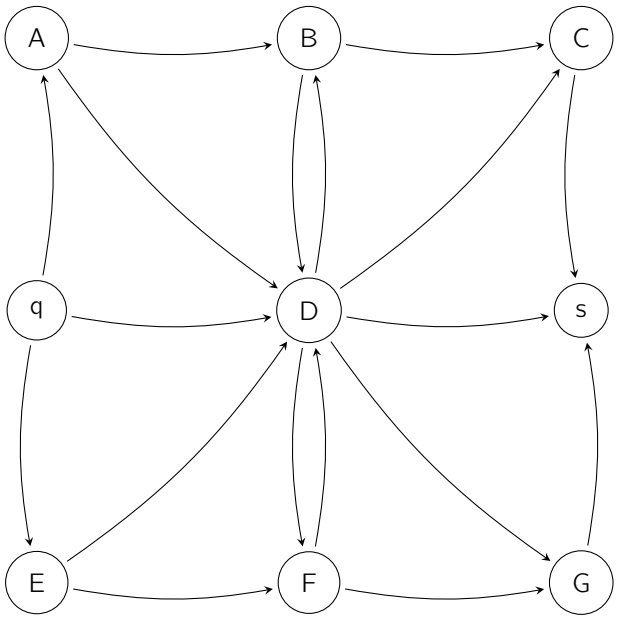
Schritt 11:

Restnetzwerk:



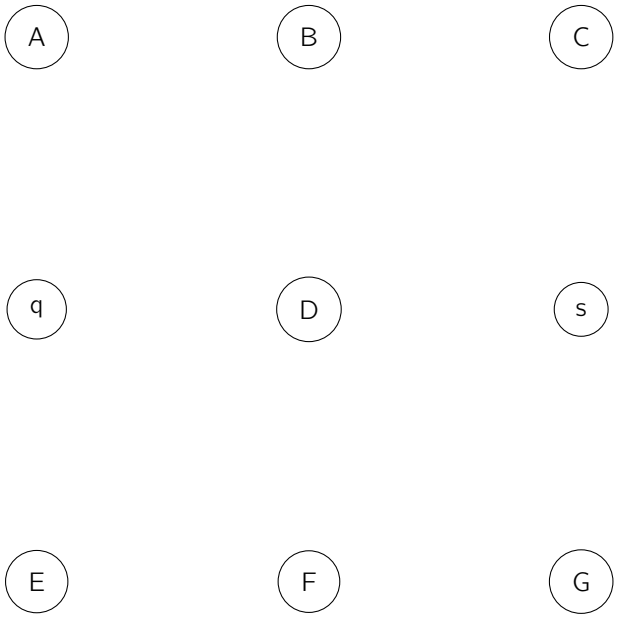
Schritt 12:

Nächstes Flussnetzwerk mit aktuellem Fluss:



Schritt 13:

Restnetzwerk:



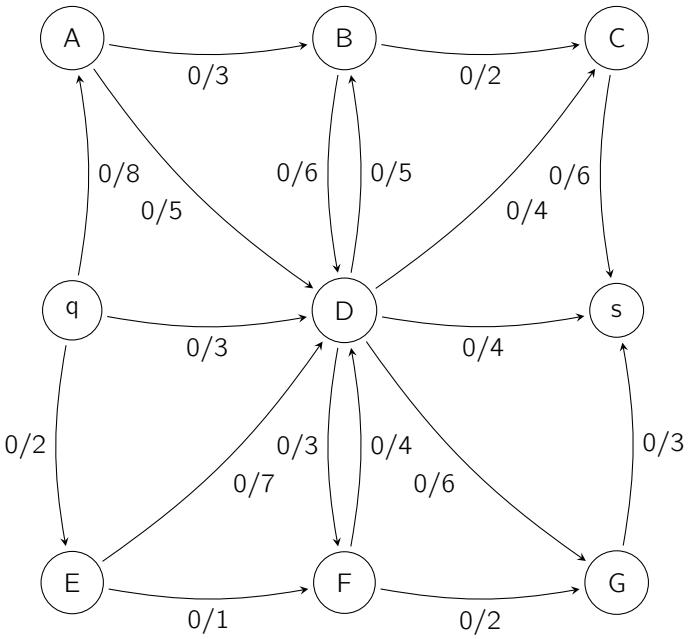
Der maximale Fluss hat den Wert:

Lösung

a)

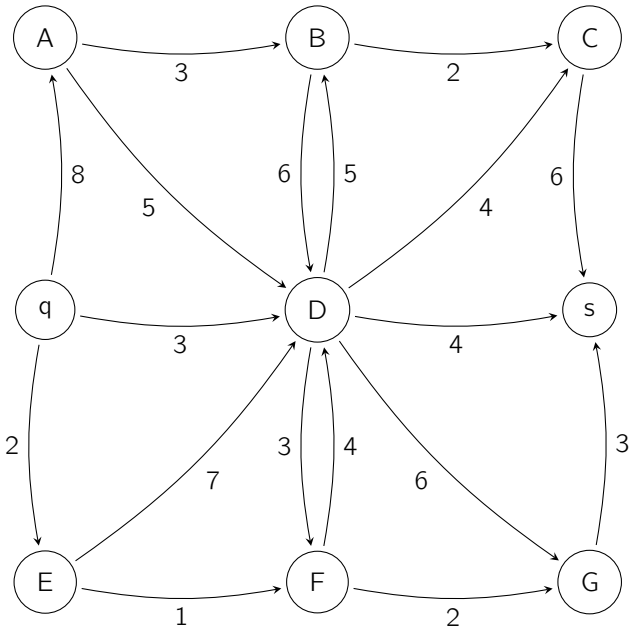
Schritt 0:

Initiales Flussnetzwerk:



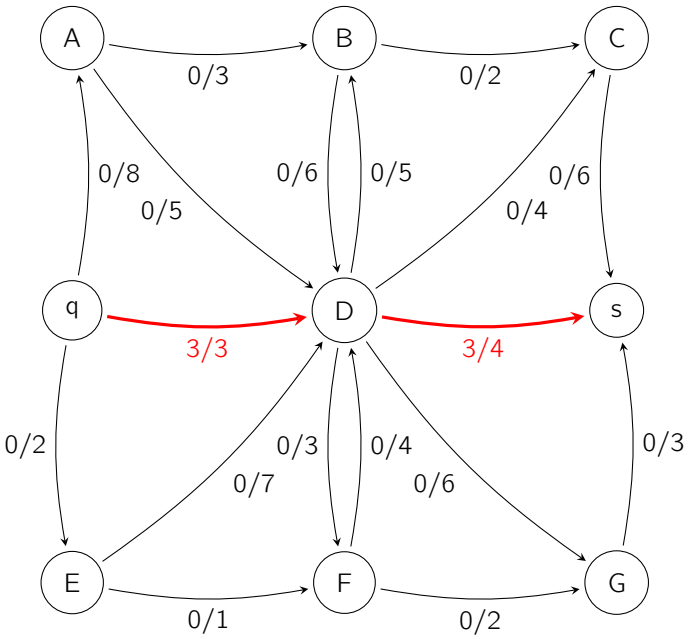
Schritt 1:

Restnetzwerk:



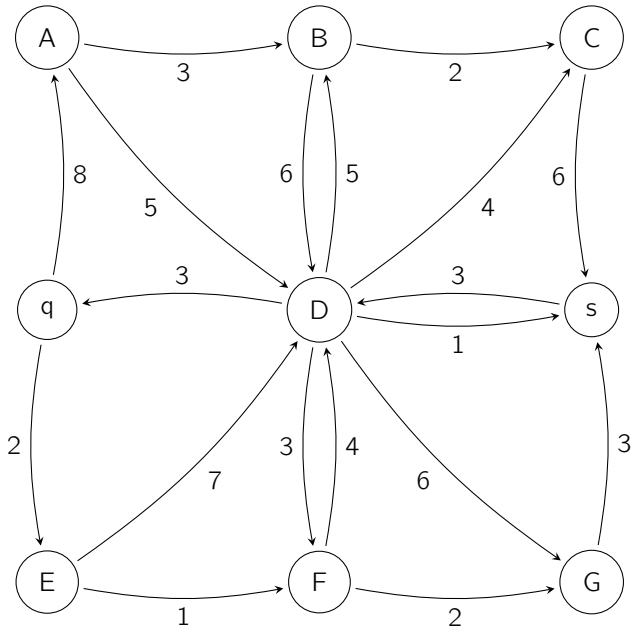
Schritt 2:

Nächstes Flussnetzwerk mit aktuellem Fluss:



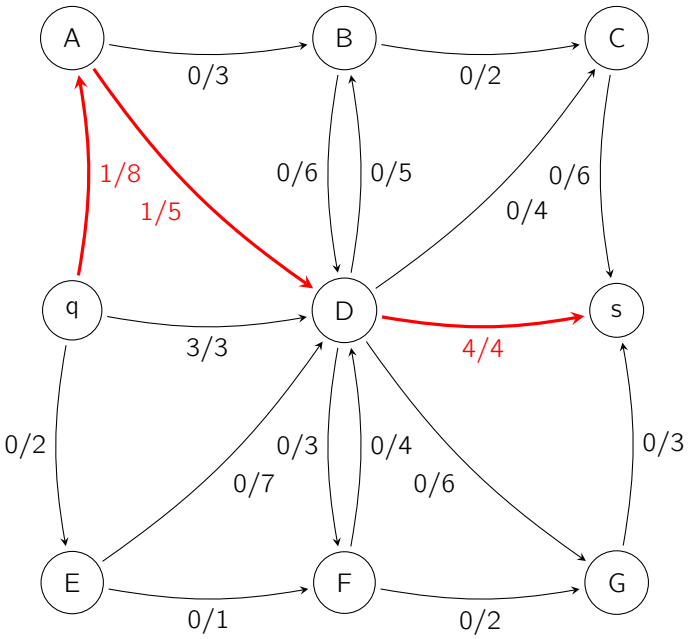
Schritt 3:

Restnetzwerk:



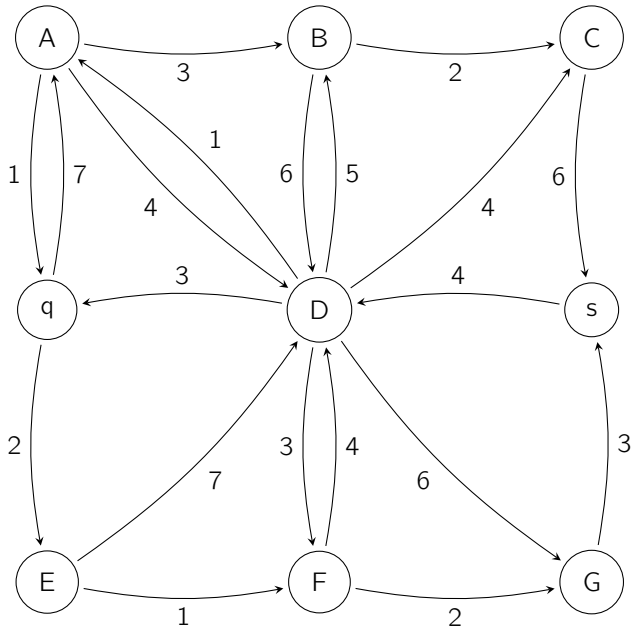
Schritt 4:

Nächstes Flussnetzwerk mit aktuellem Fluss:



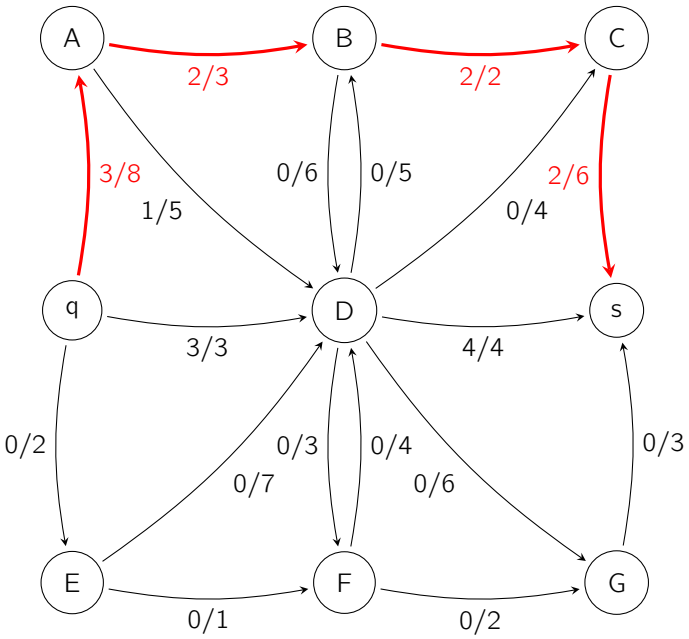
Schritt 5:

Restnetzwerk:



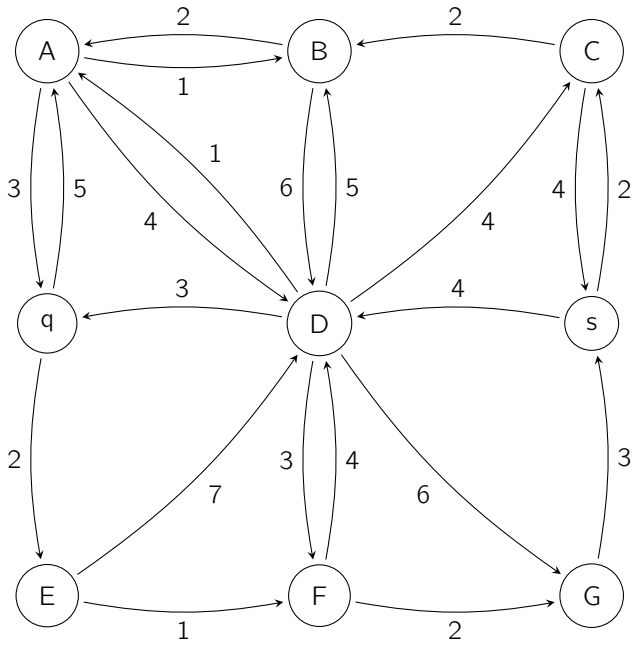
Schritt 6:

Nächstes Flussnetzwerk mit aktuellem Fluss:



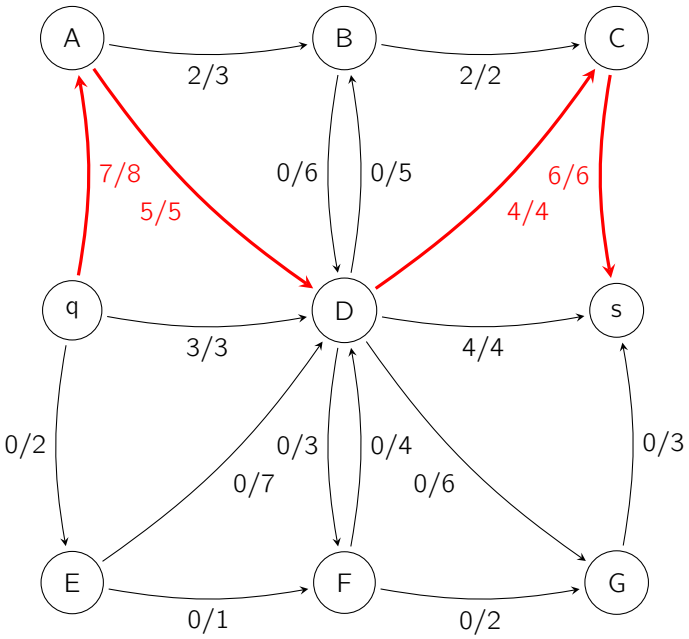
Schritt 7:

Restnetzwerk:



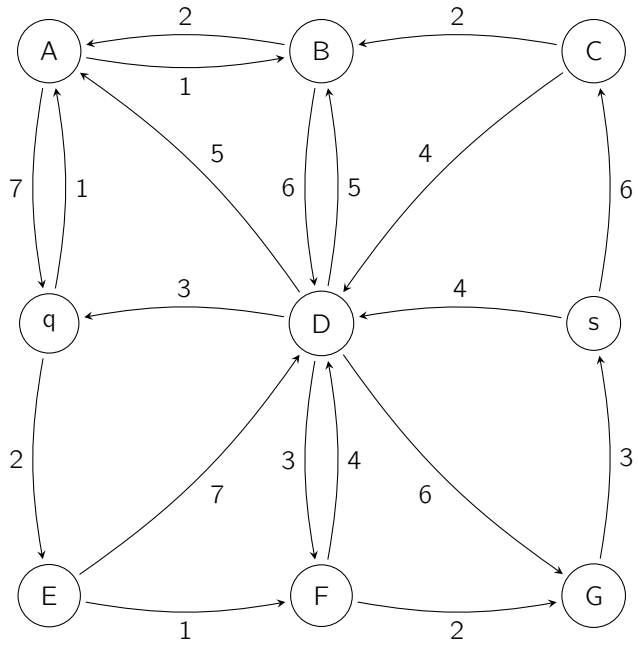
Schritt 8:

Nächstes Flussnetzwerk mit aktuellem Fluss:



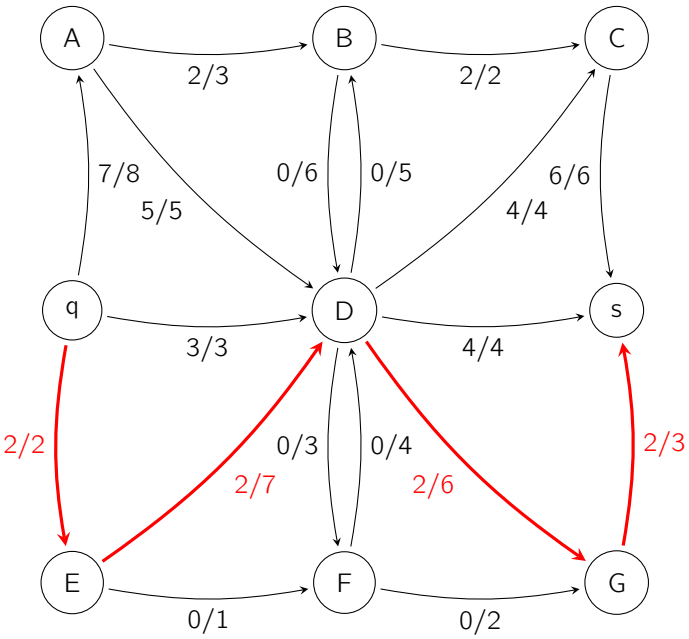
Schritt 9:

Restnetzwerk:



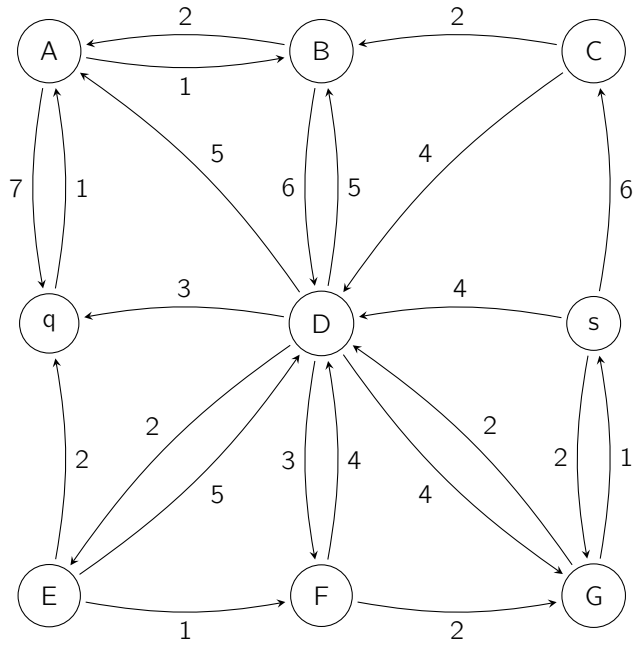
Schritt 10:

Nächstes Flussnetzwerk mit aktuellem Fluss:



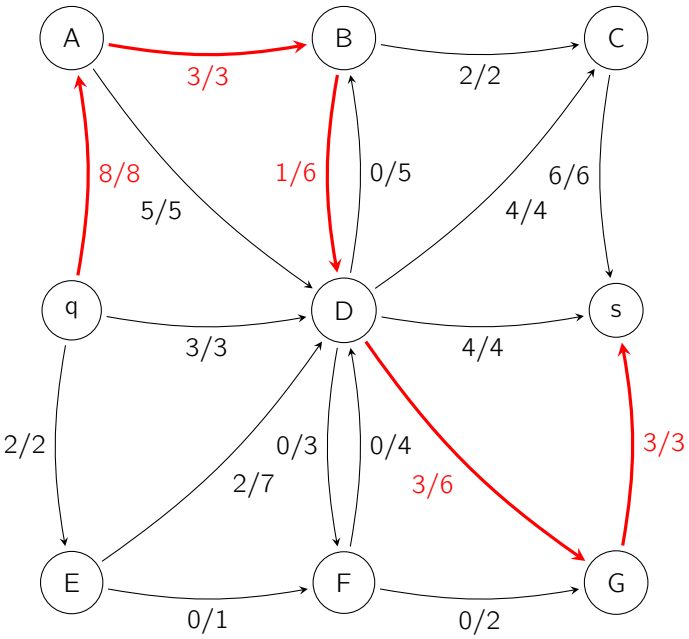
Schritt 11:

Restnetzwerk:



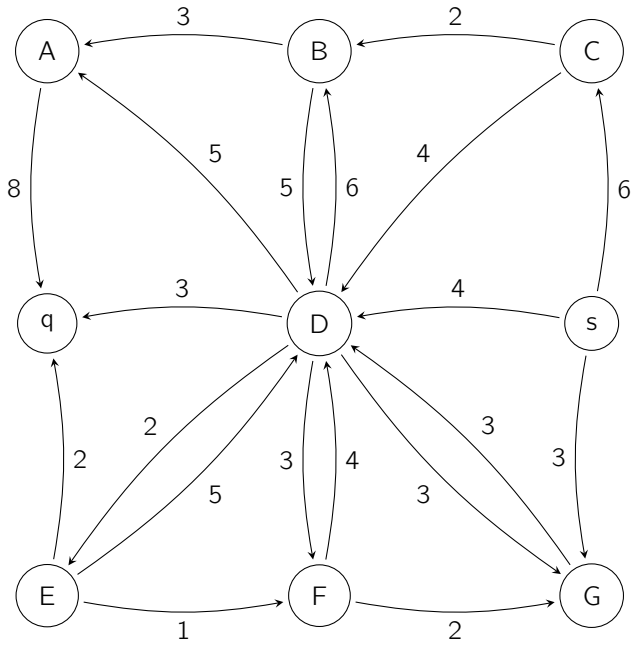
Schritt 12:

Nächstes Flussnetzwerk mit aktuellem Fluss:



Schritt 13:

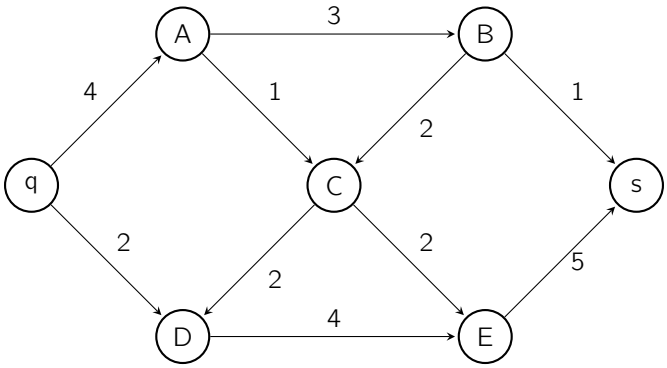
Restnetzwerk:



b) Der maximale Fluss hat den Wert: 13

Aufgabe 5 (Algorithmus von Dinic): 8 + 2 = 10 Punkte

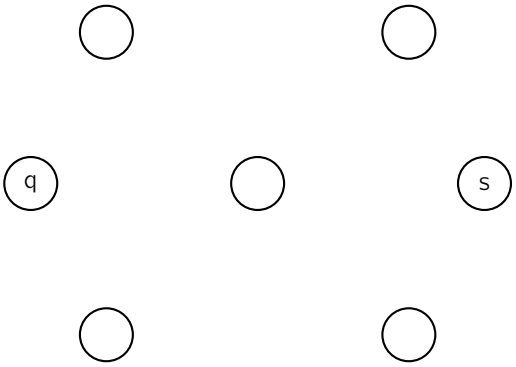
Gegeben ist folgendes Flussnetzwerk G:



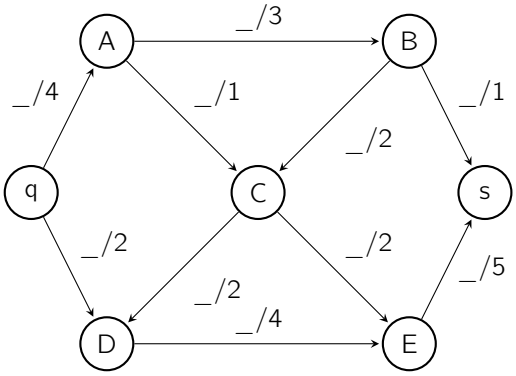
a) Berechnen Sie den Fluss von G mithilfe von Dinics Algorithmus. Geben Sie dafür stets in jeder Iteration erst das Niveaunetzwerk, dann den Sperrfluss und anschließend den aktualisierten Fluss im Flussnetzwerk an. Sie können dafür unten stehende Vorlagen nutzen.

b) Was ist der Wert des maximalen Flusses des Flussnetzwerkes G?

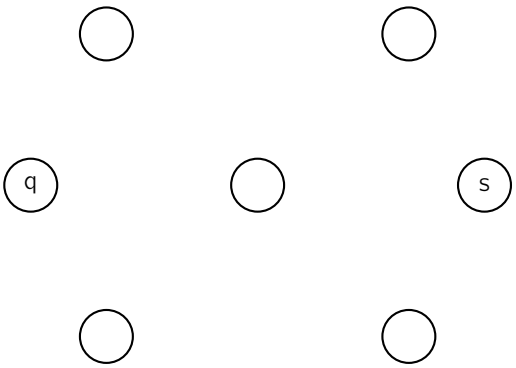
Niveaunetzwerk mit Sperrfluss



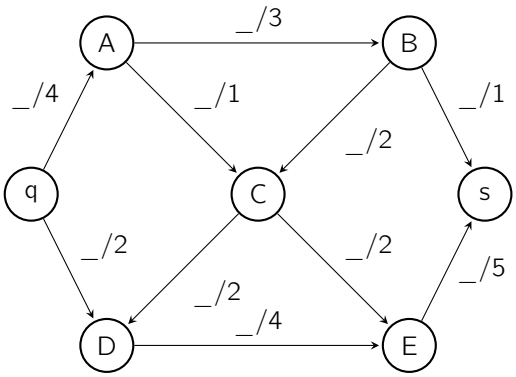
Fluss im Flussnetzwerk



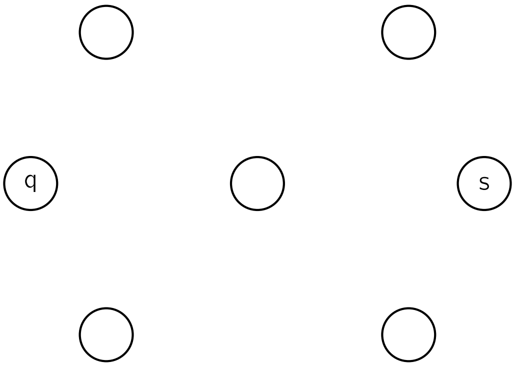
Niveaunetzwerk mit Sperrfluss



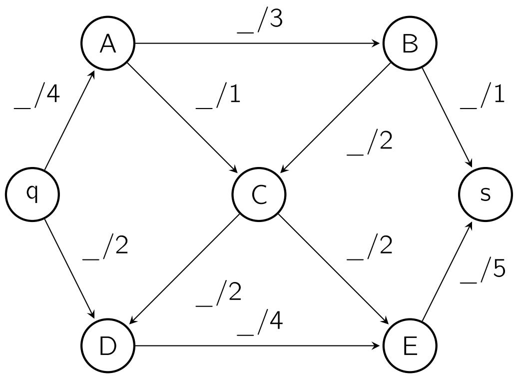
Fluss im Flussnetzwerk



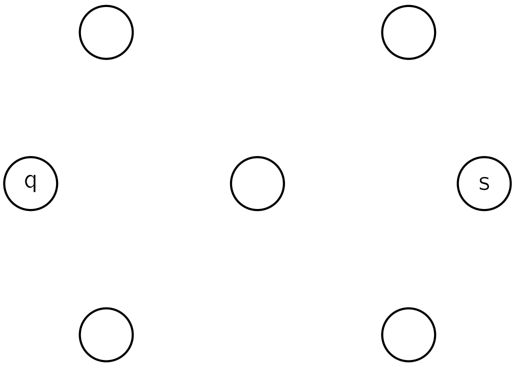
Niveaunetzwerk mit Sperrfluss



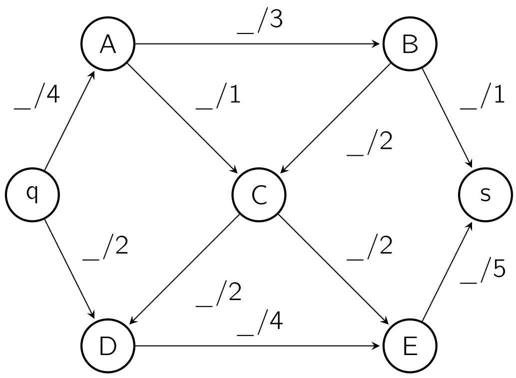
Fluss im Flussnetzwerk



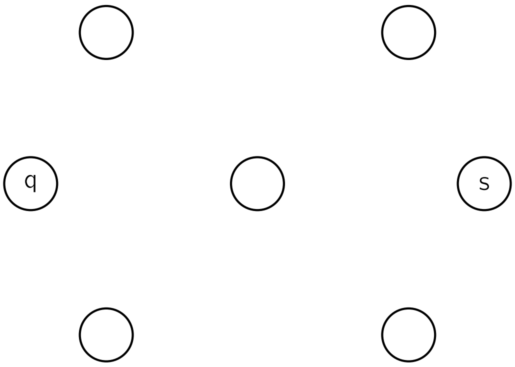
Niveaunetzwerk mit Sperrfluss



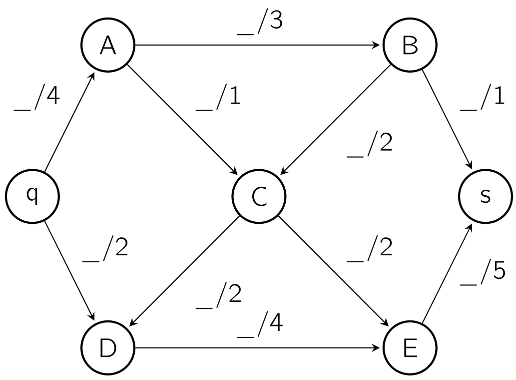
Fluss im Flussnetzwerk



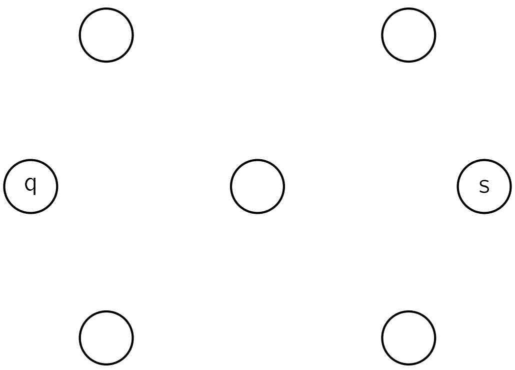
Niveaunetzwerk mit Sperrfluss



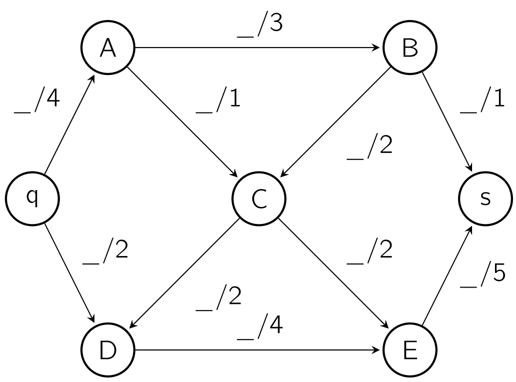
Fluss im Flussnetzwerk



Niveaunetzwerk mit Sperrfluss



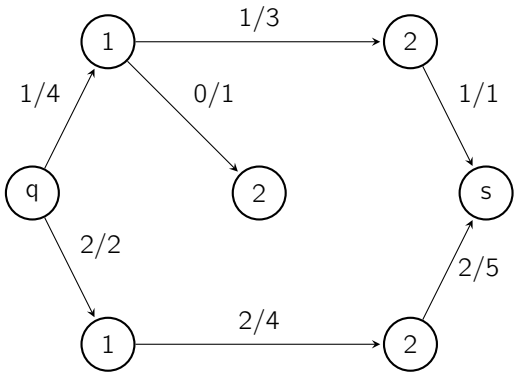
Fluss im Flussnetzwerk



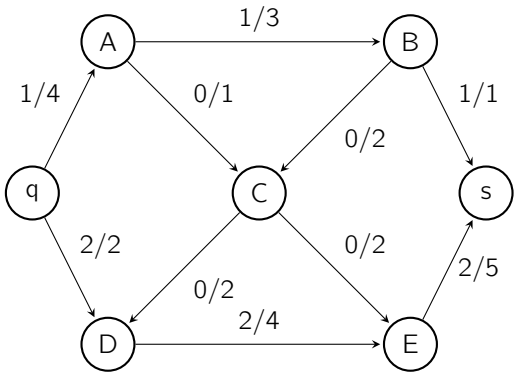
Lösung

a)

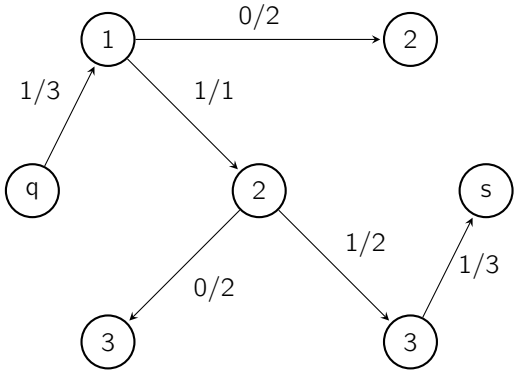
Niveaunetzwerk mit Sperrfluss



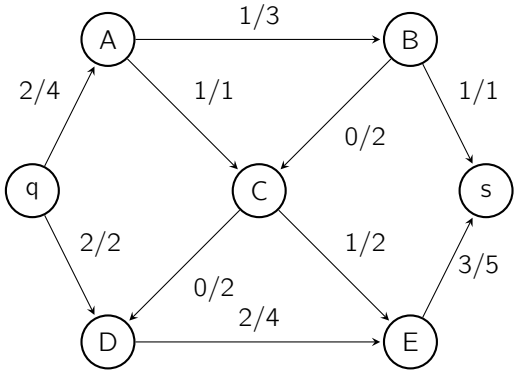
Fluss im Flussnetzwerk



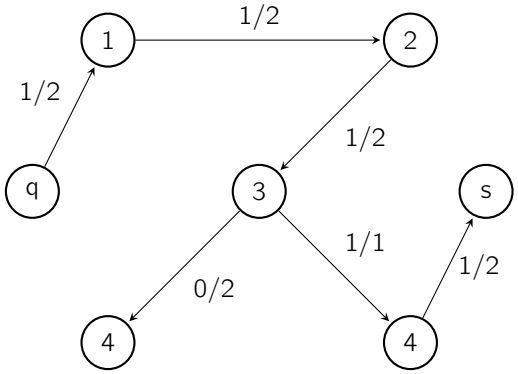
Niveaunetzwerk mit Sperrfluss



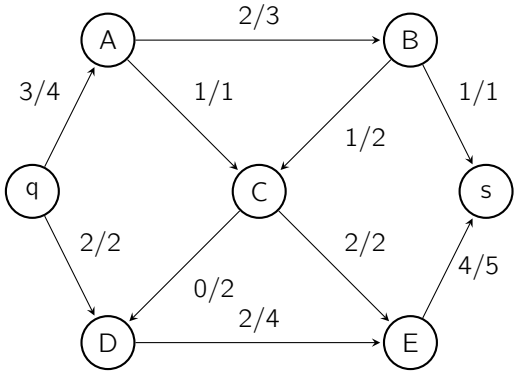
Fluss im Flussnetzwerk

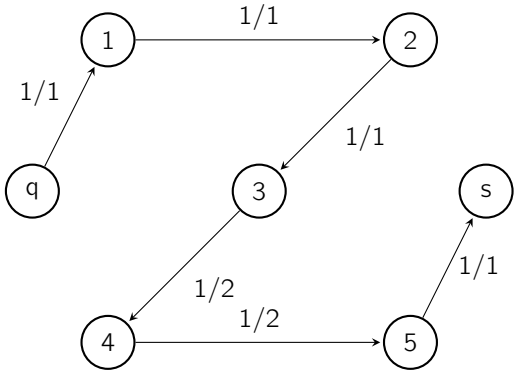


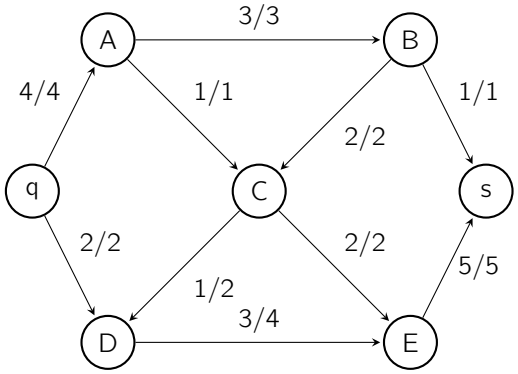
Niveaunetzwerk mit Sperrfluss



Fluss im Flussnetzwerk







b) Der maximale Fluss ist 6 (da aus der Quelle 6 Fluss ausgeht).

Aufgabe 6 (Programmierung in Python - Graphalgorithmen):

5 + 5 + 6 + 4 = 20 Punkte

Bearbeiten Sie die Python Programmieraufgaben. In dieser praktischen Aufgabe werden Sie sich mit Graphalgorithmen auseindersetzen. Diese Aufgabe dient dazu einige Konzepte der Vorlesung zu wiederholen und zu vertiefen.
Zum Bearbeiten der Programmieraufgabe können Sie einfach den Anweisungen des Notebooks *blatt11-python.ipynb* folgen. Das Notebook steht in der .zip-Datei zum Übungsblatt im Lernraum zur Vergügung.
Ihre Implementierung soll immer nach dem *# YOUR CODE HERE* Statement kommen. Ändern Sie keine weiteren Zellen.
Laden Sie spätestens bis zur Deadline dieses Übungsblatts auch Ihre Lösung der Programmieraufgabe im Lernraum hoch. Die Lösung des Übungsblatts und die Lösung der Programmieraufgabe muss im Lernraum an derselben Stelle hochgeladen werden. Die Lösung des Übungsblatts muss dazu als .pdf-Datei hochgeladen werden. Die Lösung der Programmieraufgabe muss als .ipynb-Datei hochgeladen werden.

Übersicht der zu bearbeitenden Aufgaben:

a) Implementierung von Edmonds-Karp

- breadth_first_search()
- edmonds_karp_impl()

b) Labyrinthe: Generieren und Lösen

- create_graph()
- extract_assignment()

Hinweise:

Uns ist ein kleiner Fehler in der Beschreibung der Aufgabe *a) create_graph()* unterlaufen. Die Beispiele, sowie Unittests sind korrekt und nicht von diesem Fehler betrofen. Aus Kompabilitätsgrunden möchten wir kein neues Notebook hochladen und die Beschreibung an dieser Stelle korrigieren:

Folgende Schritte sind zu beachten:

- Einfügen einer Kante von der Quelle 's' zu jedem Studenten. Die Kapazität dieser Kanten ist 1.
- Einfügen einer Kante von jedem Tutorium zu der Senke 't'. Die Kapazität dieser Kanten ist **c**.
- Verbinden der Studenten mit den ausgewählten Tutorien. Die Kapazität dieser Kanten ist 1.

Lösung

Die Lösung der Programmieraufgaben finden Sie im Lernraum. Die Datei trägt den Namen *blatt11-python-solution.ipynb*.