

Vorlesung

Effiziente Algorithmen und Komplexitätstheorie

Sommersemester 2008

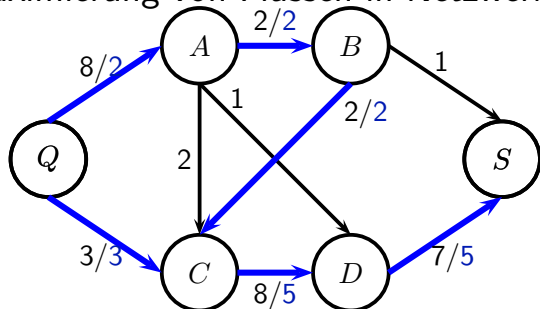
Ingo Wegener

Was bisher geschah...

- starke Zusammenhangskomponenten ✓
- Matchings
 - Algorithmus von Hopcroft und Karp für bipartite Graphen:
 $O(\sqrt{n} \cdot e)$ ✓
 - Algorithmus von Micali und Vazirani für allgemeine Graphen:
 $O(\sqrt{n} \cdot e)$ ✓ \rightsquigarrow "The General Maximum Matching Algorithm of Micali and Vazirani" von Peterson und Loui (1988)
<http://www.springerlink.com/content/x4m27854n315p60t/>

heute neues Thema Flussproblem
neu, aber verwandt

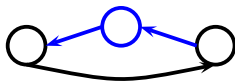
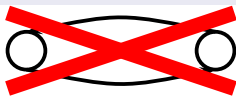
Maximierung von Flüssen in Netzwerken



Definition 4.1

Ein Netzwerk ist ein gerichteter, **asymmetrischer**, gewichteter Graph $G = (V, E, c)$ mit Quelle $Q \in V$ und Senke $S \in V$ und einer Kapazitätsfunktion $c: E \rightarrow \mathbb{N}_0$.

Quelle hat Eingangsgrad 0.



Definitionen

Definition 4.1

Ein Netzwerk ist ein gerichteter, asymmetrischer, gewichteter Graph $G = (V, E, c)$ mit Quelle $Q \in V$ und Senke $S \in V$ und einer Kapazitätsfunktion $c: E \rightarrow \mathbb{N}_0$.

Quelle hat Eingangsgrad 0.

- $\phi: E \rightarrow \mathbb{R}_0^+$ heißt **Fluss**, wenn gilt:
 - ① $\forall x \in V \setminus \{Q, S\}: \sum_{e=(x, \cdot)} \phi(e) = \sum_{e=(\cdot, x)} \phi(e)$ (**Kirchhoff-Regel**)
 - ② $\forall e \in E: \phi(e) \leq c(e)$ (**Kapazitäten respektiert**)
- Fluss ϕ heißt **ganzzahlig**, wenn $\forall e \in E: \phi(e) \in \mathbb{N}_0$
- **Wert** eines Flusses ϕ ist $w(\phi) = \sum_{e=(Q, \cdot)} \phi(e)$.
- Fluss ϕ heißt **maximal**, wenn $\forall \phi'$ Fluss: $w(\phi') \leq w(\phi)$

Beispielanwendung

maximale Matchings in bipartiten Graphen

Sei $G = (U \cup V, E)$ ein bipartiter Graph.

Definiere Netzwerk $N(G) := (V_N, E_N, c_N)$ durch

- $V_N := \{Q\} \cup \{S\} \cup U \cup V$
- $E_N := \{(Q, u) \mid u \in U\} \cup \{(u, v) \mid u \in U, v \in V, \{u, v\} \in E\} \cup \{(v, S) \mid v \in V\}$
- $c_N(e) := 1$ für alle $e \in E_N$

Beobachtung erinnert an Hopcroft/Karp

Lemma

Sei G ein ungerichteter, bipartiter Graph und ϕ ein ganzzahliger, maximaler Fluss auf $N(G)$.

$M := \{\{u, v\} \mid \phi(u, v) = 1\}$ ist maximales Matching in G .

Beweis des Lemmas

Beweis.

- ① Jeder solche Fluss Φ def. Matching M mit $|M| = w(\Phi)$:
 Definiere M durch $e \in M \Leftrightarrow \Phi(e) = 1$.
 M ist Matching, da alle Knoten $u \in U$ genau eine eingehende Kante und alle Knoten $v \in V$ genau eine ausgehende Kante haben und darum nur Fluss 1 durch jeden Knoten gehen kann.
- ② Jedes Matching M def. solchen Fluss Φ mit $|M| = w(\Phi)$:
 Definiere $\phi(Q, u_i) := \phi(u_i, v_i) := \phi(v_i, S) := 1$ für alle $\{u_i, v_i\} \in M$.
 ϕ ist ganzzahliger Fluss mit $w(\phi) = |M|$.



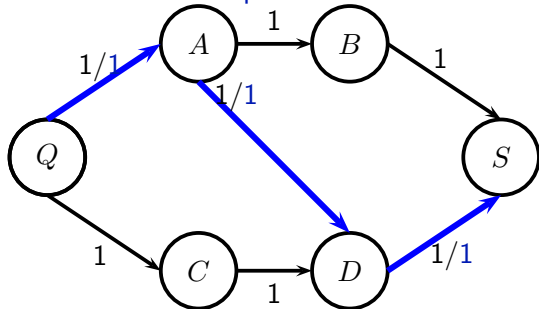
Aber gibt es einen ganzzahligen maximalen Fluss?

Wir erarbeiten uns **konstruktiven Beweis** Algorithmus,
 der ganzzahligen maximalen Fluss berechnet

Eine einfache Idee

- 1 Starte mit dem leeren Fluss $\phi \equiv 0$.
- 2 Suche einen Weg von der Quelle zur Senke ausschließlich über Kanten mit freien Kapazitäten.
- 3 Vergrößere den Fluss, indem du die kleinste Restkapazität auf diesem Weg auf den Fluss der betroffenen Kanten addierst.
- 4 Weiter bei 2.

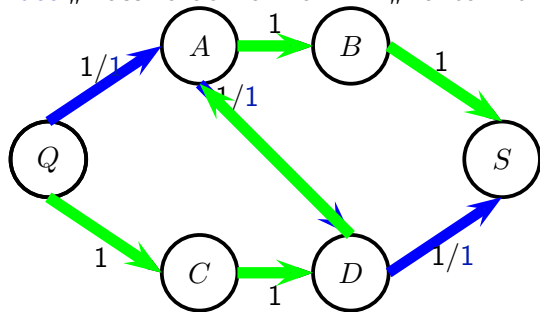
Ein warnendes Beispiel



Man muss schlechte Entscheidungen rückgängig machen können.

Das „warnende“ Beispiel

Idee „Fluss zurücknehmen“ \leftrightarrow „Kante mit Fluss rückwärts gehen“



Der Restgraph

Definition

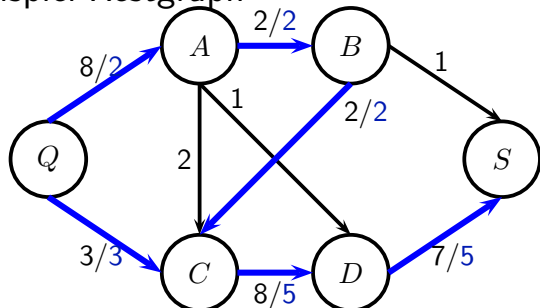
Zu $e = (x, y) \in E$ heißt $\text{rev}(e) = (y, x)$ **Rückwärtskante** von e .

Definition 4.2 (Restgraph)

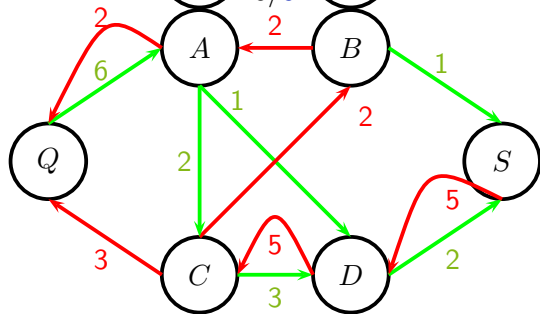
Sei $G = (V, E, c)$ ein Netzwerk, $\phi: E \rightarrow \mathbb{R}_0^+$ ein Fluss auf G . Der **Restgraph** $\text{Rest}_\phi = (V, E_\phi, r_\phi)$ hat die gleichen Knoten wie G und folgende Kanten:

- für $e \in E$ mit $\phi(e) < c(e)$ enthält E_ϕ die Kante e mit Kapazität $r_\phi(e) = c(e) - \phi(e)$,
- für $e \in E$ mit $\phi(e) > 0$ enthält E_ϕ die Kante $e' = \text{rev}(e)$ mit der Kapazität $r_\phi(e') = \phi(e)$.

Beispiel Restgraph



Netzwerk mit Fluss



Restgraph

Über den Nutzen von Restgraphen

Lemma 4.3

Sei $(G = (V, E), c)$ Netzwerk, Φ Fluss auf G , $\text{Rest}_\Phi = ((V, E'), r_\Phi)$ Restgraph dazu, $P = (e_1, e_2, \dots, e_l)$ einfacher gerichteter Weg in Rest_Φ mit Start in Q und Ende in S , $r := \min\{r_\Phi(e) \mid e \in P\}$.
Betrachte $\Phi' : E \rightarrow \mathbb{R}_0^+$ mit

$$\Phi'(e) := \begin{cases} \Phi(e) + r & \text{falls } e \in P, \\ \Phi(e) - r & \text{falls } \text{rev}(e) \in P, \\ \Phi(e) & \text{sonst.} \end{cases}$$

Φ' ist ein Fluss für (G, c) mit $w(\Phi') = w(\Phi) + r > w(\Phi)$.

Offensichtlich $w(\Phi') = w(\Phi) + r > w(\Phi)$

zu zeigen Ist Φ' wirklich ein Fluss?

Beweis von Lemma 4.3

schon gesehen Kirchhoff-Regel kritisch

klar für Q und S nichts zu zeigen ✓

Betrachte $e_i = (u, v)$, $e_{i+1} = (v, w)$ mit $v \in V \setminus \{Q, S\}$

Voraussetzung für Φ Kirchhoff-Regel für v erfüllt

1. Fall $e_i \in E$, $e_{i+1} \in E$

Beobachtung eingehende und ausgehende Summe wachsen um r ✓

2. Fall $e_i \notin E$, $e_{i+1} \notin E$

Beobachtung eingehende und ausgehende Summe fallen um r ✓

3. Fall $e_i \in E$, $e_{i+1} \notin E$

Beobachtung eingehende Summe wächst wegen e_i um r

Beobachtung eingehende Summe fällt wegen e_{i+1} um r ✓

4. Fall $e_i \notin E$, $e_{i+1} \in E$

Beobachtung ausgehende Summe wächst wegen e_{i+1} um r

Beobachtung ausgehende Summe fällt wegen e_i um r ✓

□

Algorithmus von Ford und Fulkerson

Definition

Ein Weg $Q \rightsquigarrow S$ im Restgraphen heißt **flussvergrößernd (FV-Weg)**.

1. Start mit dem leeren Fluss $\phi \equiv 0$.
2. Berechne den Restgraphen.
3. Markiere Q . *{* Wir suchen FV-Weg. *}*
4. So lange S nicht markiert ist
5. Wenn es im Restgraphen einen markierten Knoten x ,
 einen nicht markierten Knoten y und eine Kante (x, y)
 gibt, markiere y mit dem Vermerk „erreicht von x “.
6. Sonst STOP.
7. Betrachte den markierten Weg P von S nach Q .
8. $r := \min\{r_\phi(e) \mid e \in P\}$ *{* kleinste „Kapazität“ *}*
9. $\forall e \in E \cap P: \phi'(e) := \phi(e) + r$ *{* Vorwärtskante $+r$ *}*
10. $\forall \text{rev}(e) \in E \cap P: \phi'(e) := \phi(e) - r$ *{* Rückwärtskante $-r$ *}*
11. Weiter bei 2.

Einfache Beobachtungen

- Eine Runde läuft in Zeit $O(|V| + |E|)$.
- P wird in Zeit $O(|V|)$ berechnet.
- ϕ ist nach jeder Runde ein Fluss.
- Wenn S markiert wird, ist Φ nicht maximal.
- Der berechnete Fluss ist ganzzahlig.

Korrektheit

Ist der Ford-Fulkerson-Algorithmus endlich?

klar

- $B := \sum_{e=(Q,\cdot)} c(e)$ ist obere Schranke für ϕ
- ϕ wächst je Runde um ≥ 1
- Algorithmus stoppt nach $\leq B$ Runden

Ist der Fluss maximal?

etwas Vorarbeit nötig...

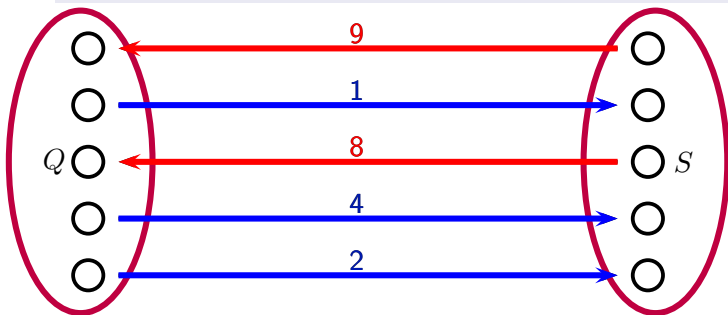
Q-S-Schnitte

Definition 4.5

(V_Q, V_S) mit $V_Q \cup V_S = V$, $Q \in V_Q$ und $S \in V_S$ heißt **Q-S-Schnitt**.

Der **Wert** eines Q-S-Schnitts ist definiert durch

$$w(V_Q, V_S) := \sum_{e \in E \cap (V_Q \times V_S)} c(e).$$



$$w = 7 = 2 + 4 + 1$$

Max Flow = Min Cut

Theorem 4.6 (Max Flow = Min Cut)

Der Wert eines maximalen Flusses ist gleich dem Wert eines minimalen Q - S -Schnittes.

Beweis. „ $\forall \phi, (V_Q, V_S): w(\phi) \leq w(V_Q, V_S)$ “

Erinnerung $w(\Phi) = \sum_{e=(Q,\cdot)} \Phi(e)$ (Definition)

Beobachtung $\forall v \in V \setminus \{Q, S\}: \sum_{e=(v,\cdot)} \Phi(e) = \sum_{e=(\cdot,v)} \Phi(e)$

(Kirchhoff)

$\Leftrightarrow \forall v \in V \setminus \{Q, S\}: \sum_{e=(v,\cdot)} \Phi(e) - \sum_{e=(\cdot,v)} \Phi(e) = 0$

also $\forall v \in V_Q \setminus \{Q\}: \sum_{e=(v,\cdot)} \Phi(e) - \sum_{e=(\cdot,v)} \Phi(e) = 0$

darum $\sum_{v \in V_Q} \left(\sum_{e=(v,\cdot)} \Phi(e) - \sum_{e=(\cdot,v)} \Phi(e) \right) = w(\Phi)$

Beweis „Max Flow = Min Cut“

haben
$$\sum_{v \in V_Q} \left(\sum_{e=(v, \cdot)} \Phi(e) - \sum_{e=(\cdot, v)} \Phi(e) \right) = w(\Phi)$$

Idee **Partitioniere** Kanten in drei Teilmengen
 $E \cap (V_Q \times V_Q)$, $E \cap (V_Q \times V_S)$, $E \cap (V_S \times V_Q)$

Beobachtung

- $E \cap (V_Q \times V_S)$ **ausschließlich** in $\sum_{e=(v, \cdot)} \Phi(e)$
- $E \cap (V_S \times V_Q)$ **ausschließlich** in $-\sum_{e=(\cdot, v)} \Phi(e)$
- $E \cap (V_Q \times V_Q)$ **jeweils** einmal in beiden Summen \rightsquigarrow Beitrag 0

also
$$\sum_{e \in E \cap (V_Q \times V_S)} \Phi(e) - \sum_{e \in E \cap (V_S \times V_Q)} \Phi(e) = w(\Phi)$$

Beweis $\forall \Phi, (V_Q, V_S): w(\Phi) \leq w(V_Q, V_S)$ (Fortsetzung)

haben
$$\sum_{e \in E \cap (V_Q \times V_S)} \Phi(e) - \sum_{e \in E \cap (V_S \times V_Q)} \Phi(e) = w(\Phi)$$

Erinnerung
$$w(V_Q, V_S) = \sum_{e \in E \cap (V_Q \times V_S)} c(e) \text{ (Definition)}$$

also
$$w(V_Q, V_S) \geq \sum_{e \in E \cap (V_Q \times V_S)} \Phi(e)$$

weil sogar $\forall e: c(e) \geq \Phi(e)$

klar
$$- \sum_{e \in E \cap (V_S \times V_Q)} \Phi(e) \leq 0$$

weil $\forall e: \Phi(e) \geq 0$

also $\forall \Phi, (V_Q, V_S): w(V_Q, V_S) \geq w(\Phi) \checkmark$

Beweis von Max Flow = Min Cut (Fortsetzung)

haben $\forall \Phi, (V_Q, V_S): w(V_Q, V_S) \geq w(\Phi)$

wollen Max Flow = Min Cut

zeigen $\exists \Phi^*, (V_Q^*, V_S^*): w(V_Q^*, V_S^*) = w(\Phi^*)$

Beobachtung daraus folgt Behauptung

Definition $\Phi^* :=$ Ergebnis von Ford/Fulkerson

Definiton für (V_Q^*, V_S^*) betrachte markierten Restgraphen Rest_{Φ^*}

$V_Q^* := \{v \mid v \text{ markiert}\}$

$V_S^* := \{v \mid v \text{ nicht markiert}\}$

Beobachtung für $e \in V_Q^* \times V_S^* : \Phi(e) = c(e)$
sonst $e \in \text{Rest}_{\Phi^*}$ und Endpunkt markierbar

Beobachtung für $e \in V_S^* \times V_Q^* : \Phi(e) = 0$
sonst $\text{rev}(e) \in \text{Rest}_{\Phi^*}$ und Startpunkt markierbar

Zusammenfassung für Φ^* und (V_Q^*, V_S^*)

haben

- $\forall e \in V_Q^* \times V_S^*: \Phi(e) = c(e)$
- $\forall e \in V_S^* \times V_Q^*: \Phi(e) = 0$

$$\begin{aligned}
 \text{also } w(\Phi^*) &= \sum_{e \in E \cap (V_Q^* \times V_S^*)} \Phi(e) - \sum_{e \in E \cap (V_S^* \times V_Q^*)} \Phi(e) \\
 &= \sum_{e \in E \cap (V_Q^* \times V_S^*)} c(e) - \sum_{e \in E \cap (V_S^* \times V_Q^*)} 0 \\
 &= \sum_{e \in E \cap (V_Q^* \times V_S^*)} c(e) = w(V_Q^*, V_S^*)
 \end{aligned}$$



Folgerungen aus Max Flow = Min Cut

- Wenn der Ford-Fulkerson-Algorithmus nicht die Senke S markiert, ist ϕ maximal.
- Der Ford-Fulkerson-Algorithmus ist korrekt.

Theorem 4.7

Der Algorithmus von Ford und Fulkerson (Algorithmus 4.4) berechnet einen maximalen Fluss in Zeit $O(B \cdot (|V| + |E|))$ mit

$$B = \min \left\{ \sum_{e=(Q,\cdot)} c(e), \sum_{e=(\cdot,S)} c(e), |V| \cdot \max\{c(e) \mid e \in E\} \right\}.$$

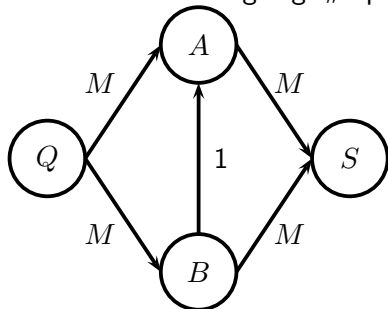


Definition Algorithmen mit Laufzeit polynomiell in Eingabelänge und Größe der größten Zahl heißen **pseudopolynomiell**

also Algorithmus von Ford und Fulkerson **nur** pseudopolynomiell
klar auch $B = \max\{w(\Phi) \mid \Phi \text{ Fluss}\}$ korrekt

Laufzeit des Ford-Fulkerson-Algorithmus

Unsere Abschätzung sagt „superpolynomiell“. Ist das realistisch?



also $2M$ Flussvergrößerungen wenn man Pech hat
bei Eingabelänge $\Theta(\log M)$, also wirklich pseudopolynomiell

Andere Flussalgorithmen

Ford-Fulkerson pseudopolynomiell – Geht es nicht besser? Doch!

$n = |V|, e = |E|, U = \max\{c(e) \mid e \in E\}$

Jahr	Autoren	Zeit in n, e, U	Zeit bei $e = \Omega(n^2)$
1969	Edmonds/Karp	$O(ne^2)$	$O(n^5)$
1970	Dinic	$O(n^2e)$	$O(n^4)$
1974	Karzanov	$O(n^3)$	$O(n^3)$
1977	Cherkasky	$O(n^2e^{1/2})$	$O(n^3)$
1978	Malhotra/Pramodh Kumar/Maheshwari	$O(n^3)$	$O(n^3)$
1978	Galil	$O(n^{5/3}e^{2/3})$	$O(n^3)$
1978	Galil/Naamad, Shiloach	$O(ne \log^2 n)$	$O(n^3 \log^2 n)$
1980	Sleator/Tarjan	$O(ne \log n)$	$O(n^2 \log n)$
1980	Sleator/Tarjan	$O(ne \log n)$	$O(n^2 \log n)$
1982	Shiloach/Vishkin	$O(n^3)$	$O(n^3)$
1983	Gabow	$O(ne \log U)$	$O(n^3 \log U)$
1984	Tarjan	$O(n^3)$	$O(n^3)$
1985	Goldberg	$O(n^3)$	$O(n^3)$
1986	Goldberg/Tarjan	$O(ne \log(n^2/e))$	$O(n^3)$
1986	Ahuja/Orlin	$O(ne + n^2 \log U)$	$O(n^3 + n^2 \log U)$

Distanzen in Graphen

Betrachte gerichteten Graphen $G = (V, E)$, Knoten $Q \in V$.
Für $v \in V$ bezeichnet $d(v)$ die Länge (=Anzahl Kanten) eines kürzesten Weges von Q nach v .

Lemma 4.8

Gegeben $v, w \in V$, Distanzen d , nach Einfügen von (v, w)
Distanzen d^+ , nach Entfernen von (v, w) Distanzen d^- .

- 1 $(d(v) \geq d(w) - 1) \Rightarrow (\forall u \in V: d^+(u) = d(u))$
- 2 $\forall u \in V$ mit $d(u) \leq d(w) - 1: d^-(u) = d(u)$
- 3 $\forall u \in V: d^-(u) \geq d(u)$

Beweis von Lemma 4.8

① zu zeigen $(d(v) \geq d(w) - 1) \Rightarrow (\forall u \in V : d^+(u) = d(u))$

trivial $\forall u \in V : d^+(u) \leq d(u)$

klar $(d^+(u) < d(u)) \Rightarrow (v, w)$ liegt auf kürzestem Weg

Beobachtung $d(w) = d^+(w)$

Beobachtung Länge kürzester Weg zu w über (v, w)

$= d(v) + 1 \geq d(w)$ (Voraussetzung) also $d^+(u) \geq d(u)$

also Kante (v, w) keine Abkürzung ✓

② zu zeigen $\forall u \in V$ mit $d(u) \leq d(w) - 1 : d^-(u) = d(u)$

Voraussetzung (v, w) existiert in G

also $d(w) \leq d(v) + 1$ wir haben $d(u) \leq d(w) - 1 \leq d(v)$

also (v, w) auf kürzestem Weg zu u nicht vorhanden ✓

③ trivial ✓



Zwei Ideen zur Beschleunigung

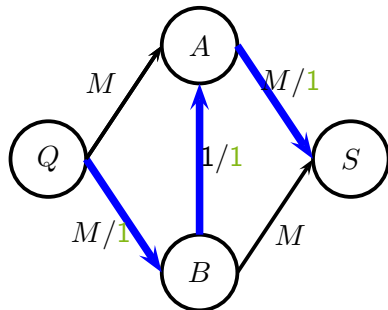
ab jetzt Netzwerk $G = (V, E, c)$ mit $|V| = n$ und $|E| = e$

Beobachtung 1: Ford-Fulkerson braucht
Zeit $O(n + e)$ für einen FVW

Frage Kann man (in der Zeit) vielleicht
mehrere disjunkte FVW berechnen?

Beobachtung 2: Kürzere FVW
wären besser gewesen.

Frage Nützt es etwas, **nur**
kürzeste FVW
zu benutzen?



Niveaunetzwerke

für kürzeste Q - S -Wege Betrachte Distanz $d(v)$ in Rest_Φ
 klar $d(v) \geq d(S) \Rightarrow v$ nicht auf kürzestem Q - S -Weg

Definition 4.9

Betrachte $(G = (V, E), c)$ mit Fluss Φ , Rest_Φ , Distanzen d .

Für $0 \leq i < d(s)$ definiere i -tes Niveau

$$V_i := \begin{cases} \{v \in V \mid d(v) = i\} & \text{für } i < d(S), \\ \{S\} & \text{für } i = d(S). \end{cases}$$

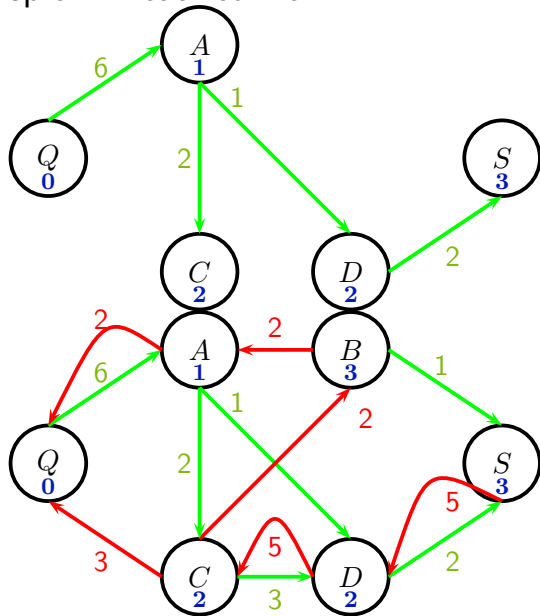
Definiere Niveaunetzwerk $N_\Phi = (V_\Phi, E_\Phi, r_\Phi)$ durch $V_\Phi := \bigcup_{i=0}^{d(S)} V_i$,

$E_i := \{(v, w) \in V_{i-1} \times V_i \mid (v, w) \in \text{Rest}_\Phi\}$, $E_\Phi := \bigcup_{i=1}^{d(S)} E_i$, und

r_Φ wie in Rest_Φ .

klar Niveaunetzwerk berechenbar in Zeit $O(n + e)$

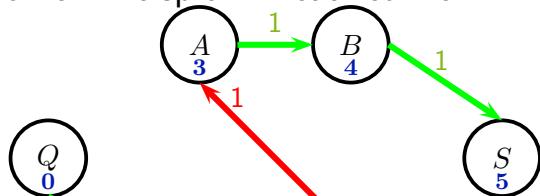
Beispiel Niveaunetzwerk



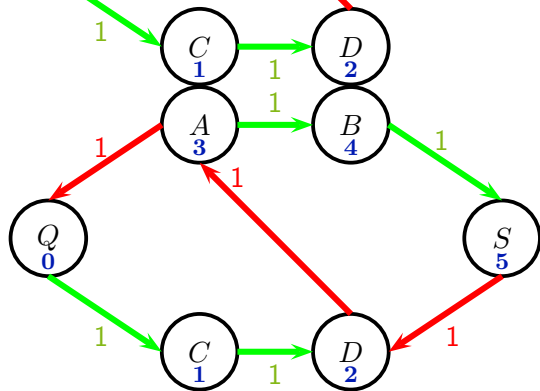
Niveaunetzwerk

Restgraph

Noch ein Beispiel Niveaunetzwerk



Niveaunetzwerk



Restgraph