

# Kapitel 4

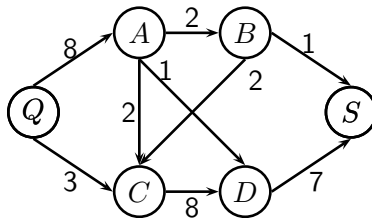
## Flüsse in Graphen

Effiziente Algorithmen, SS 2018

Professor Dr. Petra Mutzel

VO 6 am 26. April 2018

## 4.1 Das Flussproblem



### Definition 4.1

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

Die Quelle hat Eingangsgrad 0, die Senke hat Ausgangsgrad 0.

## Bemerkung zur Definition

- Asymmetrisch bedeutet:  $\forall u, v \in V : (u, v) \in E \Rightarrow (v, u) \notin E$ .
- Mit dieser Bedingung, auf die auch verzichtet werden könnte, ist die Beschreibung der Algorithmen einfacher.
- Praktisch bedeutet sie keine Einschränkung, weil man im Falle einer Verletzung künstliche Knoten hinzufügen kann  $\Rightarrow$  asymmetrisch.



# Definitionen

## Definition 4.1

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

Die Quelle hat Eingangsgrad 0. Die Senke hat Ausgangsgrad 0.

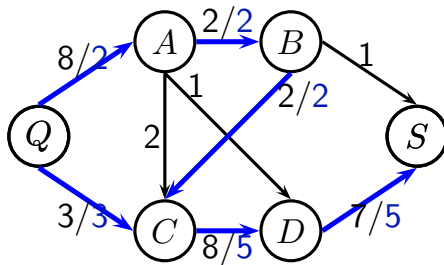
- In einem Netzwerk heißt die Abbildung  $\phi: E \rightarrow \mathbb{R}_0^+$  **Fluss**, wenn gilt:
  - ①  $\forall e \in E: \phi(e) \leq c(e)$  (**Kapazitäten respektierend**)
  - ②  $\forall v \in V \setminus \{Q, S\}: \sum_{e=(u,v)} \phi(e) = \sum_{e=(v,u)} \phi(e)$  (**Kirchhoff-Regel**)
- Ein Fluss  $\phi$  heißt **ganzzahlig**, wenn  $\forall e \in E: \phi(e) \in \mathbb{N}_0$

# Definitionen

## Definition 4.1 ff

- Der **Wert** eines Flusses  $\phi$  ist  $w(\phi) = \sum_{e=(Q,v)} \phi(e)$ .
- Ein Fluss  $\phi$  heißt **maximal**, wenn  $\forall \phi'$  Fluss:  $w(\phi') \leq w(\phi)$ .
- Das **Flussproblem**: Berechne **maximalen Fluss** in einem gegebenen Netzwerk  $N$

# Flussproblem – Beispiel



Fluss  $\phi$  mit Wert  $w(\phi) = 5$ .

# Motivation

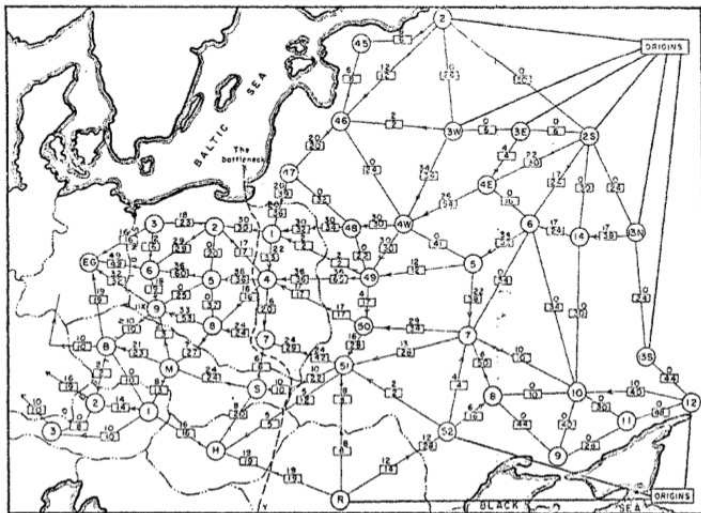
**Ziel:** möglichst viel Substanz (z.B. Wasser, Öl, Gas) durch ein Netzwerk von einer Quelle zu einer Senke liefern

Viele praktische Anwendungen, wie z.B.

- Energietransporte, wie z.B. Fernwärme
- Telekommunikation
- Gütertransporte (wenn Güter klein sind oder teilbar)
- Paketzustelldienst
- Multimodale Transporte (Speditionen, Green Logistics)
- ...

Viele Probleme als Flussproblem modellierbar!

# Historische Anwendung: Eisenbahnnetz [Harris, Ross 1955]





# Beispielanwendung

## Maximale Matchings in bipartiten Graphen

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

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

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

**Beobachtung** erinnert an Hopcroft/Karp

### Lemma

*Sei  $G$  ein ungerichteter, bipartiter Graph und  $\phi$  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  $\Phi$  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  $\Phi$  mit  $|M| = w(\Phi)$ :

Definiere  $\phi(Q, u_i) := \phi(u_i, v_i) := \phi(v_i, S) := 1$  für alle  $\{u_i, v_i\} \in M$ .

$\phi$  ist ganzzahliger Fluss mit  $w(\phi) = |M|$ .



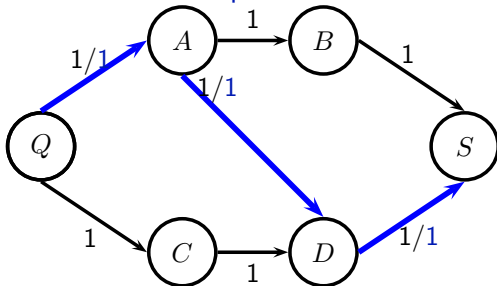
Aber gibt es einen ganzzahligen maximalen Fluss?

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

## 4.2 Algorithmus von Ford und Fulkerson: Erste 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 die kleinste Restkapazität auf diesem Weg auf den Fluss der betroffenen Kanten addiert wird.
- ④ Weiter bei 2.

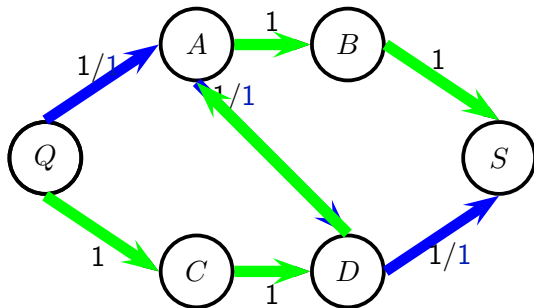
Ein warnendes Beispiel



Man muss  
**schlechte Entscheidungen**  
rückgängig machen  
können.

# Das „warnende“ Beispiel

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



# Der Restgraph

## Definition

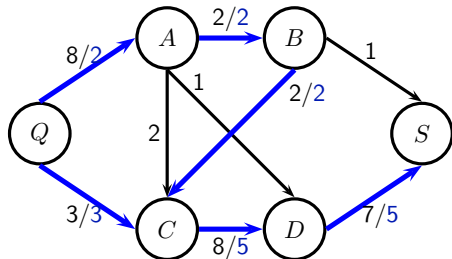
Zu  $e = (x, y) \in E$  heißt  $\text{rev}(e) = (y, x)$  **Rückwärtskante** von  $e$ .

## Definition 4.2 (Restgraph)

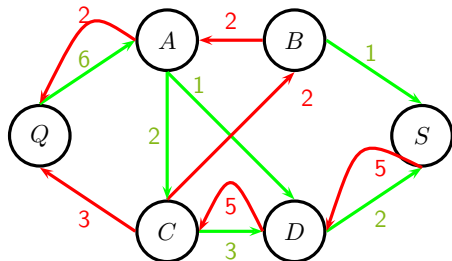
Sei  $G = (V, E, c)$  ein Netzwerk,  $\phi: E \rightarrow \mathbb{R}_0^+$  ein Fluss auf  $G$ . Der **Restgraph**  $\text{Rest}_\phi = (V, E_\phi, r_\phi)$  hat die gleichen Knoten wie  $G$  und folgende Kanten:

- für  $e \in E$  mit  $\phi(e) < c(e)$  enthält  $E_\phi$  die Kante  $e$  mit Kapazität  $r_\phi(e) = c(e) - \phi(e)$ ,
- für  $e \in E$  mit  $\phi(e) > 0$  enthält  $E_\phi$  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

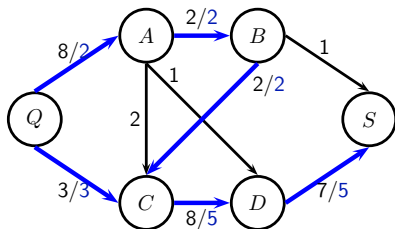
## Lemma 4.3

Sei  $(G = (V, E), c)$  Netzwerk,  $\Phi$  Fluss auf  $G$ ,  $\text{Rest}_\Phi = ((V, E'), r_\Phi)$  Restgraph dazu,  $P = (e_1, e_2, \dots, e_l)$  einfacher gerichteter Weg in  $\text{Rest}_\Phi$  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

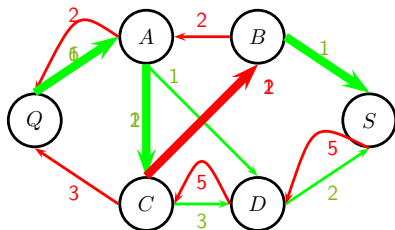
$$\Phi'(e) := \begin{cases} \Phi(e) + r & \text{falls } e \in P, \\ \Phi(e) - r & \text{falls } \text{rev}(e) \in P, \\ \Phi(e) & \text{sonst.} \end{cases}$$

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

# Beispiel Restgraph



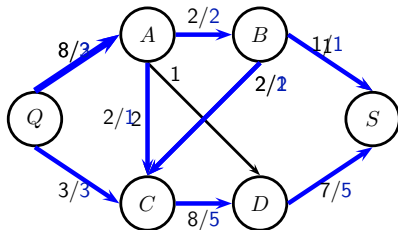
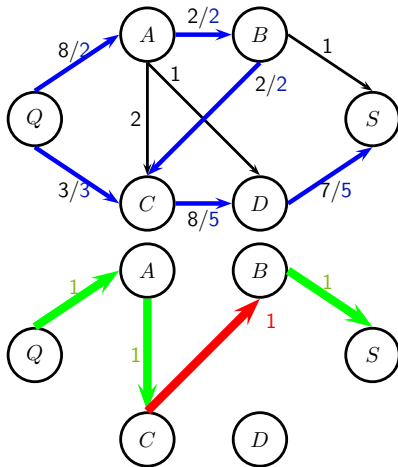
Netzwerk mit Fluss



Restgraph



## Beispiel Restgraph



# Über den Nutzen von Restgraphen

## Lemma 4.3

Sei  $(G = (V, E), c)$  Netzwerk,  $\Phi$  Fluss auf  $G$ ,  $\text{Rest}_\Phi = ((V, E'), r_\Phi)$  Restgraph dazu,  $P = (e_1, e_2, \dots, e_l)$  einfacher gerichteter Weg in  $\text{Rest}_\Phi$  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

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

$\Phi'$  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  $\Phi'$  wirklich ein Fluss? Kapazitäten, Kirchhoff-Regel

## Beweis von Lemma 4.3

Fluss  $\Phi$ ,  $\text{Rest}_\Phi = ((V, E'), r_\Phi)$  Restgraph,

$P = (e_1, e_2, \dots, e_l)$  einfacher gerichteter Weg in  $\text{Rest}_\Phi$

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

schon gesehen    Kirchhoff-Regel kritisch:  $\sum_{e=(\cdot, v)} \phi(e) = \sum_{e=(v, \cdot)} \phi(e)$

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  $\Phi$  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  $r$  ✓

## Beweis von Lemma 4.3 – 2. Teil

Fluss  $\Phi$ ,  $\text{Rest}_\Phi = ((V, E'), r_\Phi)$  Restgraph,

$P = (e_1, e_2, \dots, e_l)$  einfacher gerichteter Weg in  $\text{Rest}_\Phi$

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

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  $r$  ✓



## Auf dem Weg zum Algorithmus . . .

- Restgraph  $\text{Rest}_\Phi$  zu Fluss  $\Phi$
- $Q - S$ -Weg in  $\text{Rest}_\Phi \Rightarrow$  Fluss kann vergrößert werden:
  - minimale Restkapazität  $r$  auf dem Pfad berechnen
  - $r$  zu  $\Phi(e)$  für  $e \in E$  addieren
  - $r$  von  $\Phi(e)$  für  $e \notin E$  abziehen
  - Wert des Flusses erhöht sich um  $r$

### Definition

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

Wir sagen auch: Der Fluss wird entlang des Weges augmentiert.

# Algorithmus von Ford und Fulkerson

1. Start mit dem leeren Fluss  $\phi \equiv 0$ .
2. Berechne den Restgraphen.
3. Markiere  $Q$ .    *{\* Wir suchen FV-Weg. \*}*
4. Solange  $S$  nicht markiert ist
  5. Falls 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. Ausgabe  $\phi$ .
7. Betrachte den markierten Weg  $P$  von  $Q$  nach  $S$ .
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

- Schritte 4.-6. mit Breitensuche oder Tiefensuche erreichbar
- Minimumsuche geht in Zeit  $O(|P|) = O(V)$  .
- $\phi$  wird in Zeit  $O(|P|) = O(V)$  augmentiert (aktualisiert).
- Also: eine Runde läuft in Zeit  $O(|V| + |E|)$ .
- $\phi$  ist nach jeder Runde ein Fluss.
- Wenn  $S$  markiert wird, ist  $\Phi$  nicht maximal.
- Der berechnete Fluss ist ganzzahlig.

# Terminierung

Ist der Ford-Fulkerson-Algorithmus endlich?

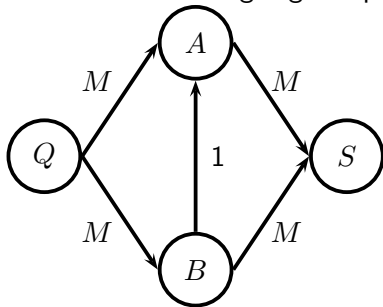
klar

- $B := \sum_{e=(Q,\cdot)} c(e)$  ist obere Schranke für  $\phi$
- $\phi$  wächst je Runde um  $\geq 1$
- Algorithmus stoppt nach  $\leq B$  Runden
- Laufzeit  $O(B \cdot (|V| + |E|))$



# Laufzeit des Ford-Fulkerson-Algorithmus

Unsere Abschätzung sagt „superpolynomiell“. Ist das realistisch?



also  $2M$  Flussvergrößerungen wenn man Pech hat  
 mehr dazu: später

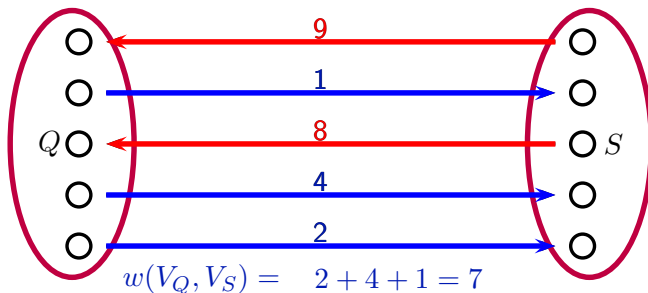
Ist der Fluss maximal? — Beweis im nächsten Abschnitt...

## 4.3 Das Max-Flow / Min-Cut Theorem

### Definition 4.5: $Q$ - $S$ -Schnitte

Eine Partitionierung der Knotenmenge von  $G$  in  $(V_Q, V_S)$  mit  $V_Q \uplus V_S = V$ ,  $Q \in V_Q$  und  $S \in V_S$  definiert einen  $Q$ - $S$ -Schnitt. Der **Wert** eines  $Q$ - $S$ -Schnitts ist definiert durch

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



# Max Flow = Min Cut

## Theorem 4.6 (Max Flow = Min Cut)

Der Wert eines maximalen Flusses in einem Netzwerk  $N$  ist gleich dem Wert eines minimalen  $Q$ - $S$ -Schnittes.

### Beweisidee

2 Schritte:

- ① „ $\forall$  Flüsse  $\phi$  und  $\forall$  Schnitte  $(V_Q, V_S): w(\phi) \leq w(V_Q, V_S)$ “
- ②  $\exists \Phi^*$  und  $\exists (V_Q^*, V_S^*): w(v_Q^*, V_S^*) = w(\Phi^*)$

Aus dem Beweis folgt:

- (a) Max Flow = Min Cut
- (b) Optimalität von Ford-Fulkerson, da  $\Phi^* :=$  Ausgabe von Ford-Fulkerson

# Beweis „Max Flow = Min Cut“

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

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)

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

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

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

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

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

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

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

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

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

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

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

also

$$w(\Phi) = \sum_{e \in E \cap (V_Q \times V_S)} \Phi(e) - \sum_{e \in E \cap (V_S \times V_Q)} \Phi(e) \leq w(V_Q, V_S) + 0$$

also  $\forall \Phi, (V_Q, V_S): w(\Phi) \leq w(V_Q, V_S)$  ✓

# Beweis „Max Flow = Min Cut“ (Fortsetzung)

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

**wollen** Max Flow = Min Cut

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

**Beobachtung** daraus folgt Behauptung

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

**Erinnerung** daraus folgt Optimalität des Algorithmus

**Definition** für  $(V_Q^*, V_S^*)$  betrachte markierten Restgraphen  $\text{Rest}_{\Phi^*}$

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

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

**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 $\Phi^*$ und $(V_Q^*, V_S^*)$

haben

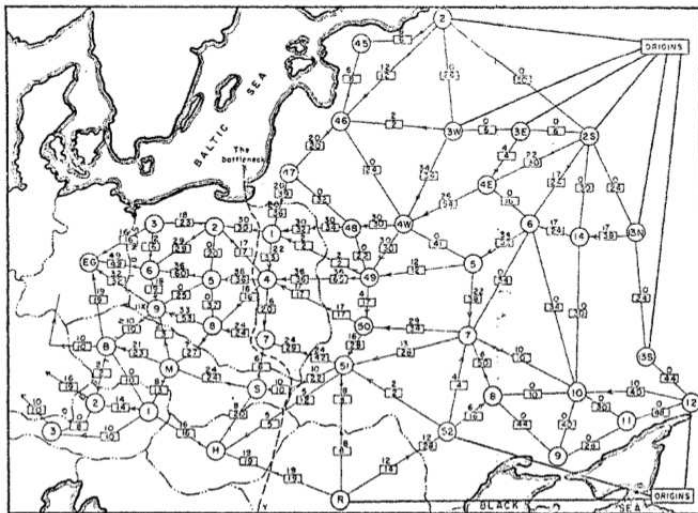
- $\forall e \in V_Q^* \times V_S^*: \Phi(e) = c(e)$
- $\forall e \in V_S^* \times V_Q^*: \Phi(e) = 0$

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





# Motivation für Max Flow vs. Min Cut



## Folgerungen aus Max Flow = Min Cut

- Wenn der Ford-Fulkerson-Algorithmus nicht die Senke  $S$  markiert, ist  $\phi$  maximal.
- Der Ford-Fulkerson-Algorithmus ist korrekt.
- Der maximale Fluss ist ganzzahlig.

### Theorem 4.7

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

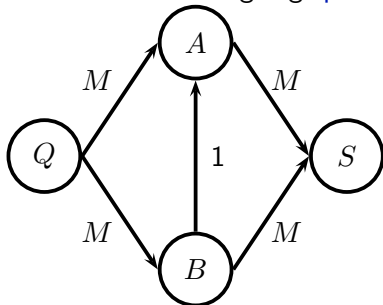
$$B = \min \left\{ \sum_{e=(Q,\cdot)} c(e), \sum_{e=(\cdot,S)} c(e) \right\}.$$

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

**also** Algorithmus von Ford und Fulkerson **nur** pseudopolynomiell

# Laufzeit des Ford-Fulkerson-Algorithmus

Unsere Abschätzung sagt **pseudopolynomiell**. Ist das **realistisch**?



**also**  $2M$  Flussvergrößerungen **wenn man Pech hat**

bei Eingabelänge  $\Theta(\log M)$

also Laufzeit  $O(B \cdot (|V| + |E|))$  **wirklich pseudopolynomiell**