Technische Universität Braunschweig

Decision Support
Institut für Wirtschaftsinformatik

Operations Research

Vorlesung 9

Graphen und Netzwerke: Maximale Flüsse & kantenorientierte Rundreisen

Wiederholung

- Neue Art der Modellierung: Graphentheorie
- Gerichtete und ungerichtete Graphen
- Gewichtete Graphen
- Minimale Spannbäume
- Kürzeste Wege in Graphen

Überblick

1. Maximale Flüsse in Netzwerken

2. Kantenorientierte Rundreisen

Überblick

1. **Maximale Flüsse in Netzwerken**

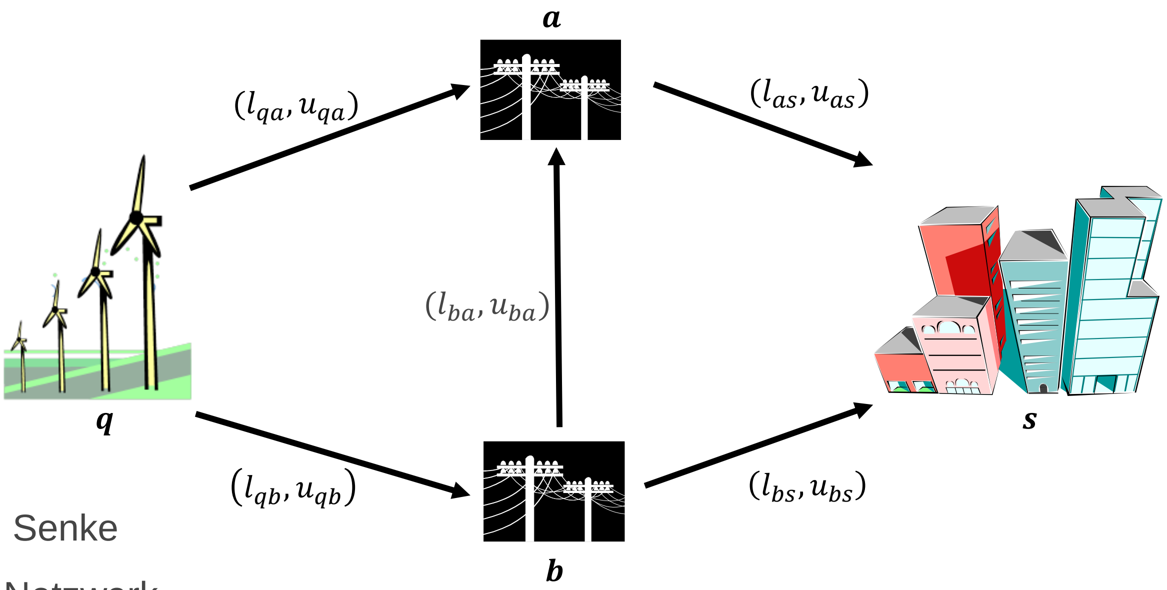
2. Kantenorientierte Rundreisen

Beispiel: Ausbau der Stromnetze

- Viele Windenergieanlagen in Nord-Deutschland
- Energie muss in andere Regionen von Deutschland transportiert werden
- Jedes Segment im Netz hat eine minimale und maximale Kapazität
- Wie können gegebene Netze maximal genutzt werden?
- Wie sollten Netze ausgebaut werden?

Windenergieanlagen in Deutschland 2012. Quelle: Bundesinstitut für Bau-, Stadt-, und Raumforschung BBSR-Analysen KOMPAKT 01/2014, Bonn, Mai 2014, http://www.bpb.de/politik/wirtschaft/energiepolitik/148524/ausbau-des-stromnetzes

Maximale Flüsse in Netzwerken



q, s: Quelle und Senke

a, b: Knoten im Netzwerk

c_{ij}: max. Kapazität der Leitung von i nach j

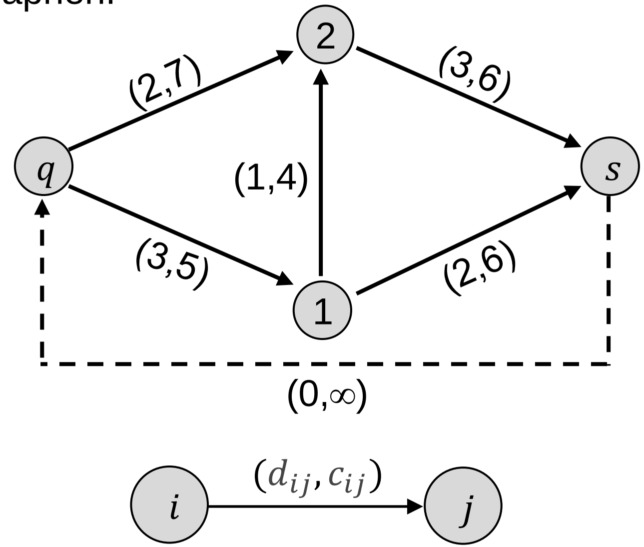
d_{ij}: min. Kapazität der Leitung von i nach j

Ziel: Max. den Fluss zwischen Quelle u. Senke

NB: Flusserhaltungssatz an den Knoten

Maximale Flüsse in Netzwerken

Abbildung in Form eines Graphen:



Einführung einer Zirkulationskante (s, q)
→ Fluss auf dieser Kante entspricht dem Gesamtfluss von q nach s!

→ Maximierung des Flusses auf (s, q) ist gleichbedeutend mit einer Maximierung des Gesamtflusses im Netzwerk

Definitionen:

c_{ij} : maximale Kapazität des Pfeils (i, j)

d_{ij} : minimale Kapazität des Pfeils (i, j)

x_{ij}: Fluss auf dem Pfeil (i, j)

Netzwerkflussproblem: Allgemeine Formulierung

- Gerichteter Graph mit Minimal- und Maximalkapazitäten
- Genau 1 Quelle
- Genau 1 Senke
- Zu jedem Knoten $i \in V$ sei definiert:
  - Menge $P(i)$ der Vorgänger von i
  - Menge $N(i)$ der Nachfolger von i

Max $\quad z = x_{sq} \quad \leftarrow$ Fluss

u.d.N. $\quad \sum_{k \in P(i)} x_{ki} - \sum_{j \in N(i)} x_{ij} = 0 \quad \forall\, i \in V$

$$x_{ij} \le c_{ij} \qquad \forall\, (i,j) \in E$$

$$x_{ij} \ge d_{ij} \Rightarrow -x_{ij} \le -d_{ij} \qquad \forall\, (i,j) \in E$$

$$x_{ij} \ge 0 \qquad \forall\, (i,j) \in E$$

Flussbilanz: Inflow = Outflow
“Kirchhoff'scher Knotensatz“
(analog elektrische Netzwerke)

Netzwerkflussproblem: Spezielle Formulierung

max $\quad x_{sq}$

u.d.N.

$$x_{sq} - x_{sq} - x_{q2} = 0$$
$$x_{q1} - x_{12} - x_{1s} = 0$$
$$x_{q2} + x_{12} - x_{2s} = 0$$
$$x_{1s} + x_{2s} - x_{sq} = 0$$
$$x_{q1} \geq 1$$
$$x_{q1} \leq 5$$
$$x_{q2} \geq 1$$
$$x_{q2} \leq 4$$
$$x_{12} \geq 2$$
$$x_{12} \leq 5$$
$$x_{1s} \geq 2$$
$$x_{1s} \leq 7$$
$$x_{2s} \geq 2$$
$$x_{2s} \leq 5$$

Lösung:

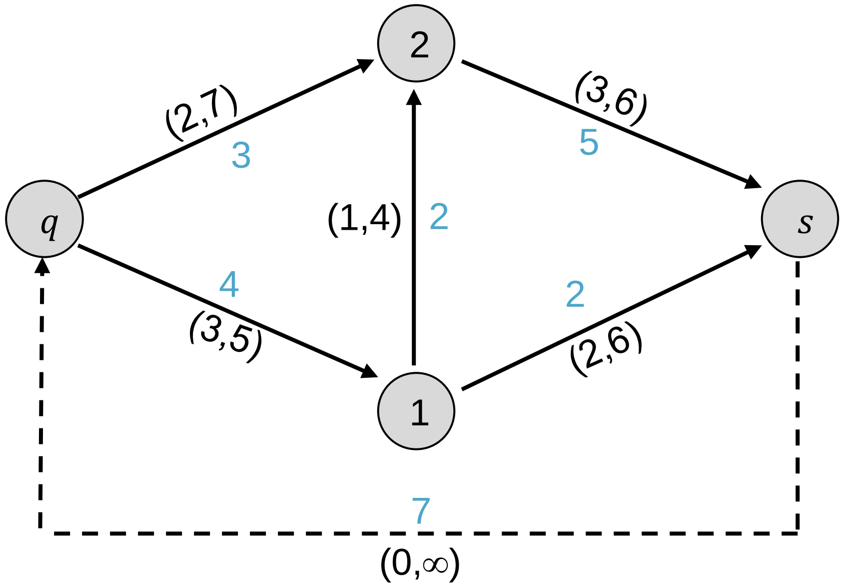
$$x_{sq} = 8$$
$$x_{q1} = 5$$
$$x_{q2} = 3$$
$$x_{12} = 2$$
$$x_{1s} = 3$$
$$x_{2s} = 5$$
$$ZF = 8$$

- Lineares Optimierungsproblem, d.h. grundsätzlich mit Simplex-Algorithmus lösbar,
- *ABER*: Spezieller "Netzwerk-Algorithmus" reduziert Problemkomplexität!

$\Rightarrow$ **Algorithmus von Ford / Fulkerson**

Ford-Fulkerson: Motivation

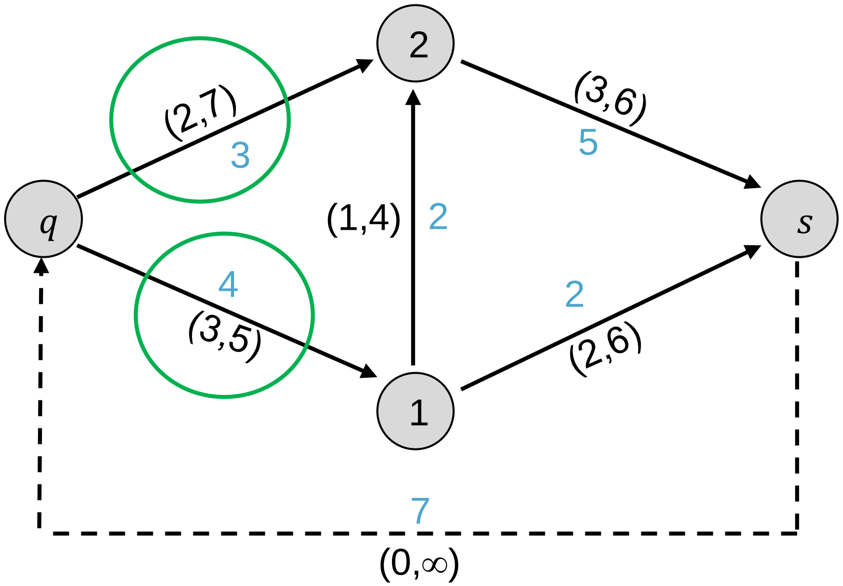
- Zunächst wird ein zulässiger Ausgangsfluss bestimmt.



- Ist der bestimmte Fluss optimal?

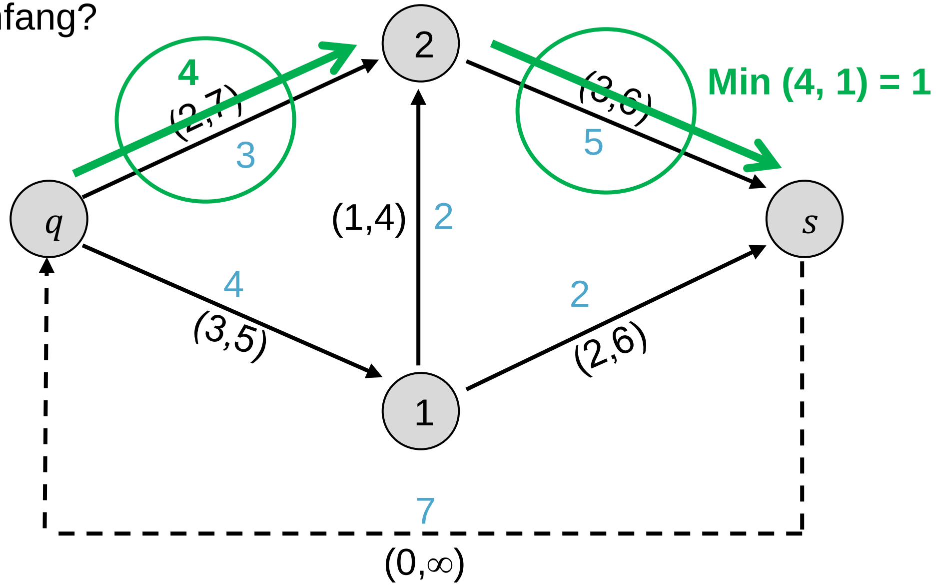
Ford-Fulkerson: Motivation

- Können wir den Fluss aus q erhöhen?



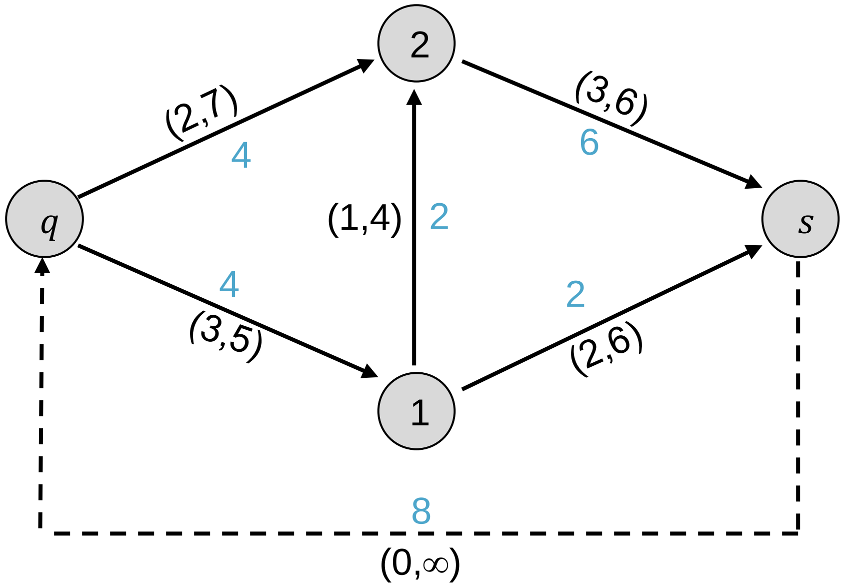
Ford-Fulkerson: Motivation

- Können wir die Flusserhöhung bis zu Knoten s fortsetzen?
- Wenn ja, in welchem Umfang?



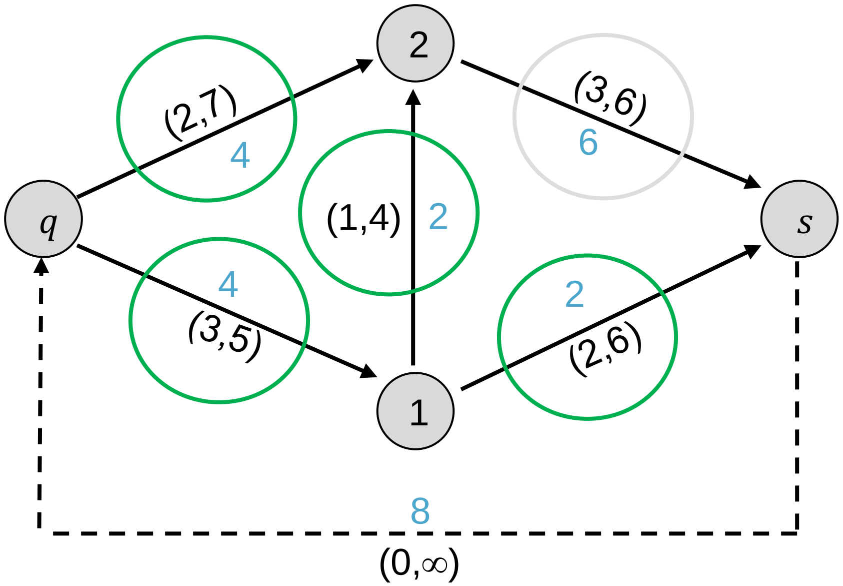
Ford-Fulkerson: Motivation

- Aktualisierter Fluss



Ford-Fulkerson: Motivation

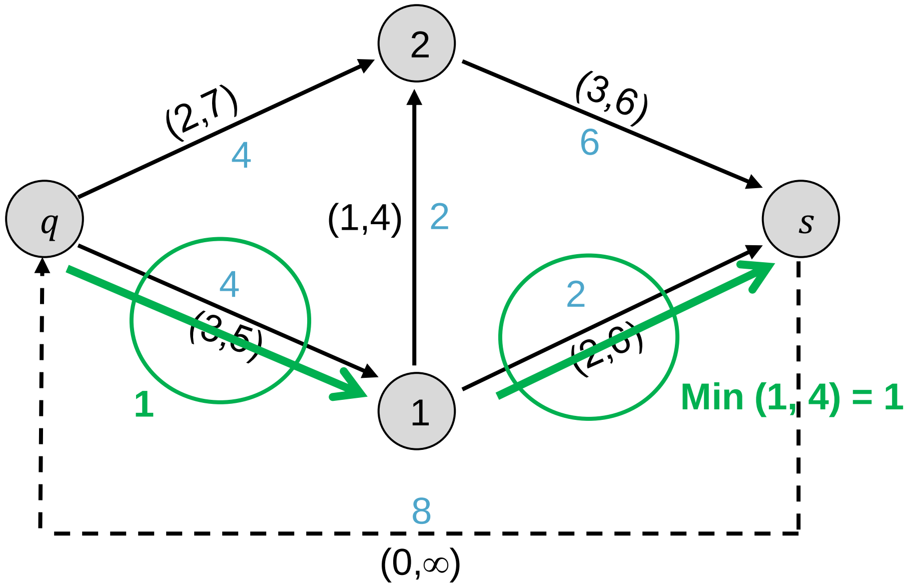
- Können wir den Fluss aus q erneut erhöhen?



- Es kann ein alternativer Weg bestimmt werden.

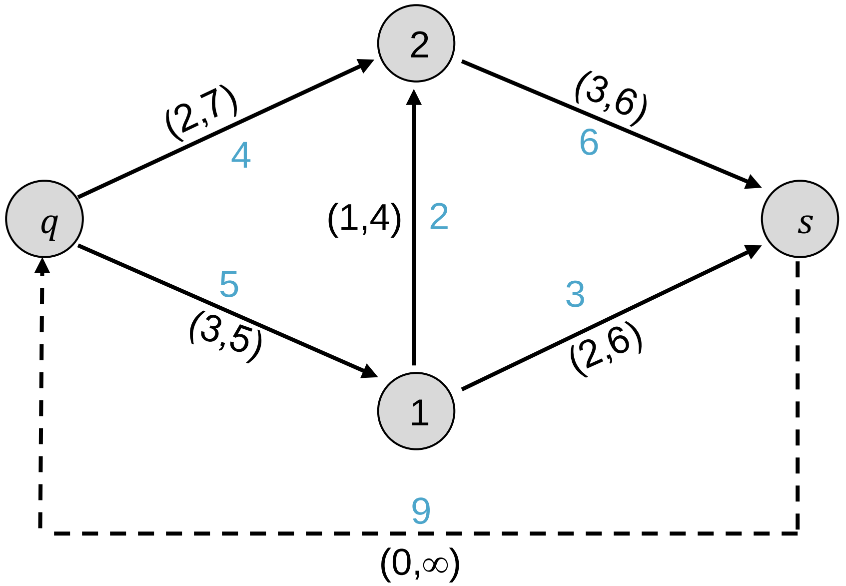
Ford-Fulkerson: Motivation

- Die Flusserhöhung bestimmt sich aus dem Minimum der freien Kapazitäten auf den Kanten.



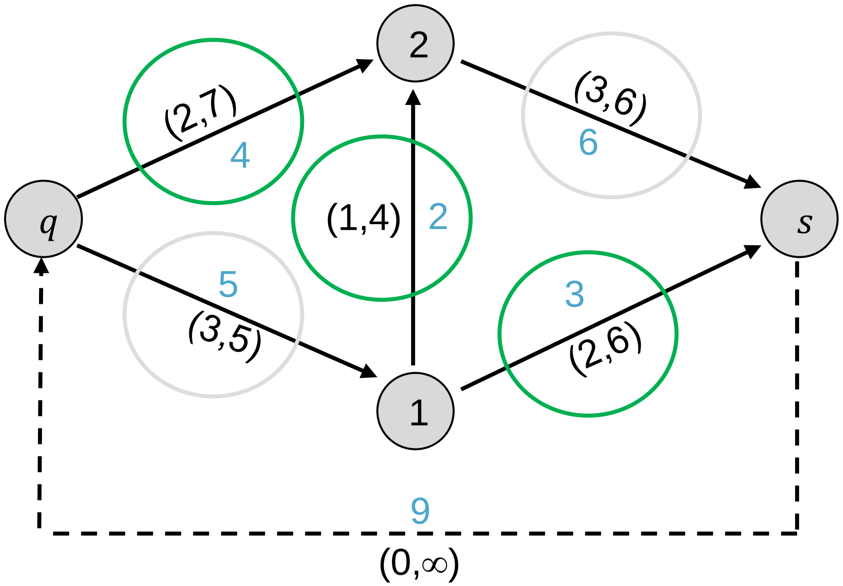
Ford-Fulkerson: Motivation

- Aktualisierter Fluss



Ford-Fulkerson: Motivation – Rückwärtskante

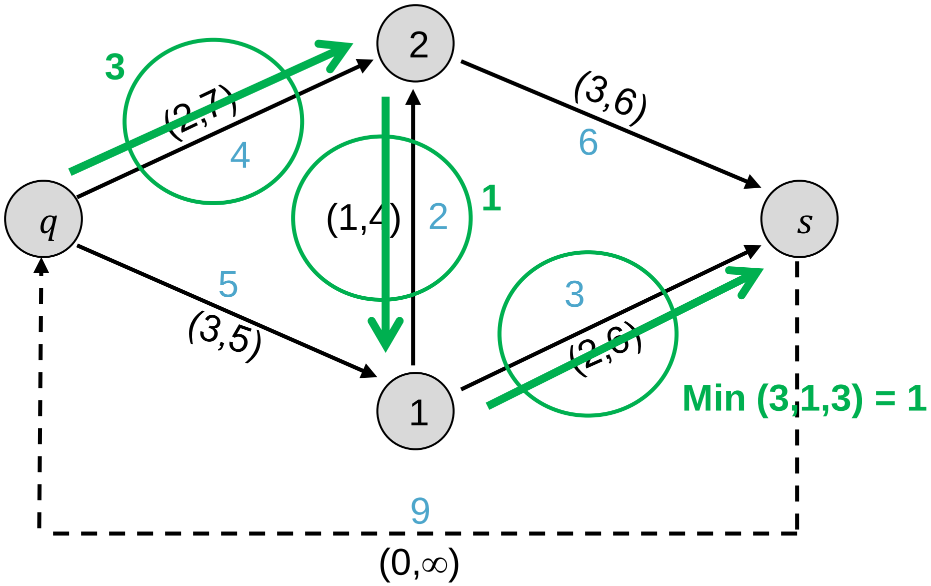
- Können wir den Fluss aus q erneut erhöhen?



- Es kann ein alternativer Weg bestimmt werden.

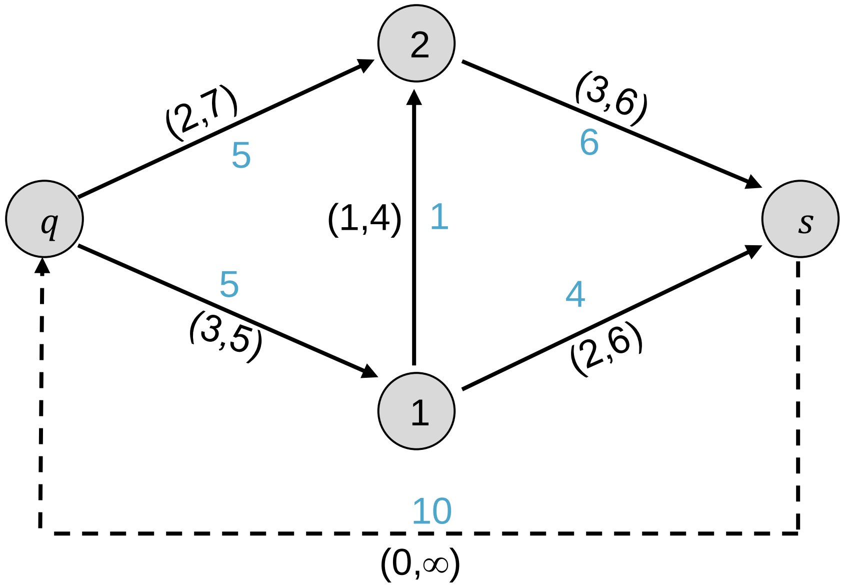
Ford-Fulkerson: Motivation – Rückwärtskante

- Wieviel können wir schicken?



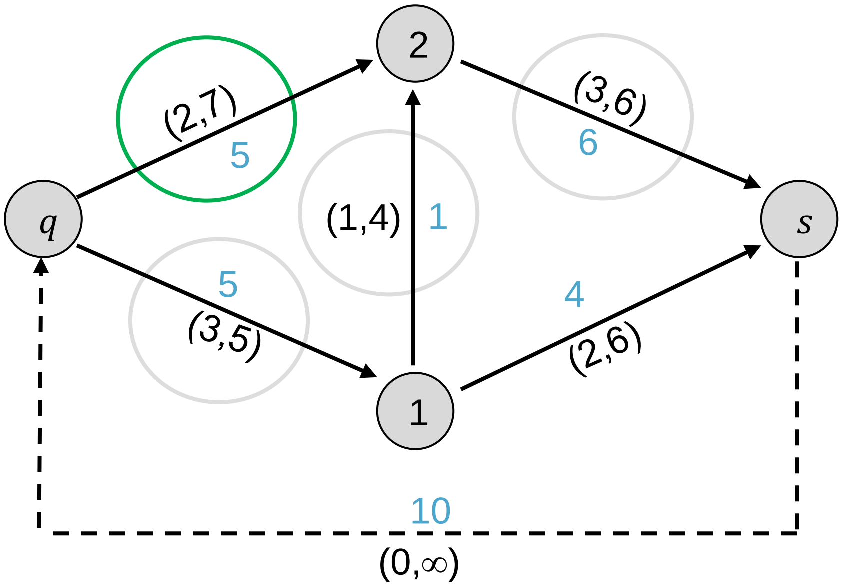
Ford-Fulkerson: Motivation – Rückwärtskante

- Neuer Fluss



Ford-Fulkerson: Motivation

- Können wir den Fluss aus q erneut erhöhen?



- Es kann kein weiterer Weg bestimmt werden.

Algorithmus von Ford / Fulkerson: Prinzip

- Ausgehend von einem *zulässigen Ausgangsfluss* werden sukzessive *zulässige Flüsse* wachsender Stärke ermittelt.

- Der Fluss entlang eines Weges von q nach s vergrößert sich durch
  - Vergrößerung des Flusses entlang Kanten in Richtung von s
  - Verkleinerung von Rückflüssen entlang Kanten in Richtung von q

- Flüsse und Rückflüsse werden simultan durch *Semiwege* modelliert.

Algorithmus von Ford / Fulkerson: Formulierung

- Es werden Iterationen durchgeführt, bis keine weitere Flusserhöhung erreicht werden kann.
- In einer Iteration wird Semiweg von q nach s konstruiert.
- Dies erfolgt durch Markierung besuchter Knoten entlang des Semiwegs.
- Ein Knoten $k \neq q$ wird genau dann *markiert,* wenn es einen flussvergrößernden (q, k)-Semiweg gibt, d.h. wenn von q nach k zusätzlich eine positive Menge transportiert werden kann.
- Existiert ein flussvergrößernder (q, s)-Semiweg (ausgehend von q wird s markiert), so kann der aktuelle Fluss erhöht werden.
- Die Flusserhöhung bestimmt sich aus dem Minimum der Erhöhung / Erniedrigung der Kanten des (q,s)-Semiwegs.

Algorithmus von Ford / Fulkerson: Formulierung

Von Knoten i kommend wird der nächste Knoten j bestimmt:

(a) Markierung am Ende einer Vorwärtskante (i, j): (i^+, ε_j)

- i^+: Vorgängerknoten auf dem Semiweg
- ε_j : Maximal mögliche Flussvergrößerung auf dem Semiweg von q nach j

  → Flussvergrößerung wird begrenzt durch

  - maximale Flussvergrößerung auf dem Weg von q nach i
  - maximaler zusätzlicher Fluss auf der Kante (i, j)

(b) Markierung am Start einer „Rückwärtskante" (j, i): (i^-, ε_j)

- i^-: Vorgängerknoten auf dem Semiweg
- ε_j : Maximal mögliche Flussvergrößerung auf dem Semiweg von q nach j

  → Flussvergrößerung wird begrenzt durch

  - maximale Flussvergrößerung auf dem Weg von q nach i
  - maximal mögliche Flussreduktion auf der Kante (i, j)

Algorithmus von Ford / Fulkerson: Zusammenfassung

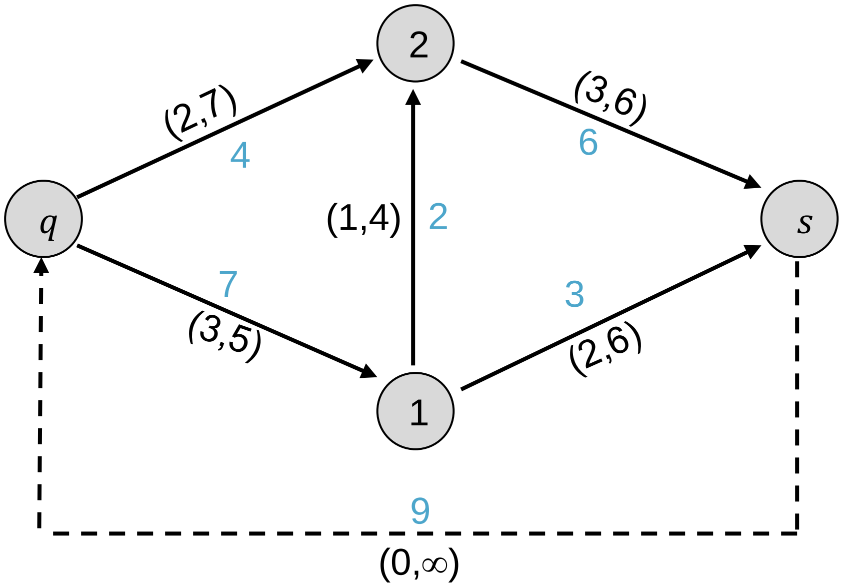
1. Markiere die Quelle q mit $(s^+, \varepsilon_s = \infty)$

2. Für jeden markierten Knoten i:
   Überprüfe alle Vorgänger und Nachfolger:
   - WENN für einen Nachfolger j gilt:
     a. j ist nicht markiert
     b. $x_{ij} < c_{ij}$
   - DANN markiere j mit $\left(i^+, \varepsilon_j = \min\left\{\varepsilon_i, c_{ij} - x_{ij}\right\}\right)$
   - WENN für einen Vorgänger k gilt:
     a. k ist nicht markiert
     b. $x_{ki} > d_{ki}$
   - DANN markiere k mit $(i^-, \varepsilon_k = \min\{\varepsilon_i, x_{ki} - d_{ki}\})$

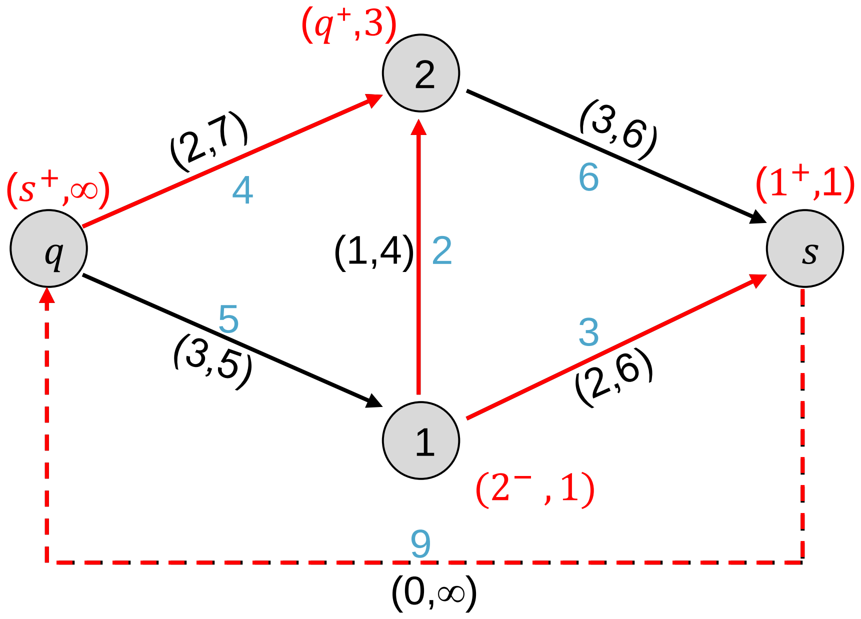
3. WENN Senke markiert ist DANN Flussvergrößerung gefunden, gehe zu 1.
   WENN keine Markierung mehr möglich DANN terminiere

Algorithmus von Ford / Fulkerson: Beispiel

- Zulässiger Ausgangsfluss:



Algorithmus von Ford / Fulkerson: Beispiel



$(q,2)$: Vorwärtskante,
maximale Flusserhöhung: 3

Von 2 aus: Keine Flusserhöhung entlang einer Vorwärtskante möglich

$(1{,}2)$: Rückwärtskante,
maximale Flussreduktion: 1

$$\min(2; 2-1) = 1 \rightarrow \varepsilon_1 = 1$$

$(1,s)$: Vorwärtskante,
maximale Flusserhöhung: 3

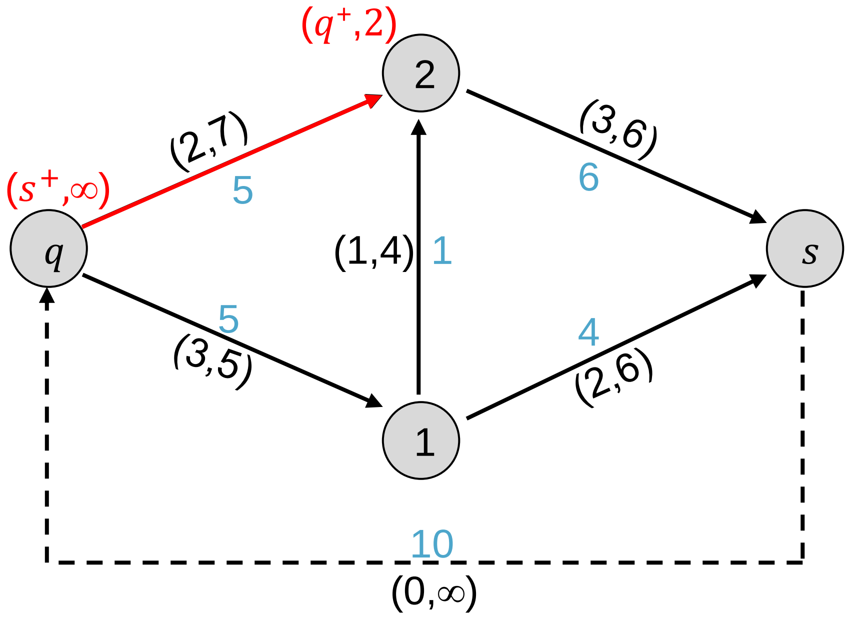
$$\min(1; 6-3) = 1 \rightarrow \varepsilon_s = 1$$

Senke markiert,

Semiweg: $q, 2, 1, s$

$\Rightarrow$ Flusserhöhung um $\varepsilon_s = 1$

Algorithmus von Ford / Fulkerson: Beispiel



$(q,2)$: Vorwärtskante, maximale Flusserhöhung: 2

Von 2 aus: Keine Flusserhöhung entlang einer Vorwärtskante möglich

Von 2 aus: Keine Verringerung entlang einer Rückwärtskante möglich

Von q aus: Keine Flusserhöhung entlang einer Vorwärtskante möglich

Es kann keine weitere Markierung mehr erfolgen, d.h. die Flusssenke kann nicht markiert werden

$\Rightarrow$ Abbruch, der Fluss $x_{sq} = 10$ ist maximal

Ford and Fulkerson 1954

In ihrem originären Beitrag “*Maximal Flow through a Network*” (publiziert in 1954), erwähnen Ford und Fulkerson, dass ihnen das Problem maximaler Flüsse von T.E. Harris wie folgt kommuniziert wurde:

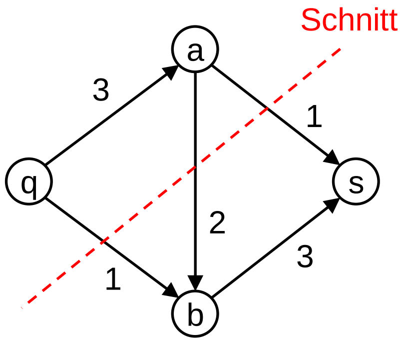
Consider a rail network connecting two cities by way of a number of intermediate cities, where each link of the network has a number assigned to it representing its capacity. Assuming a steady state condition, find a maximal flow from one given city to the other.

Max-Flow Min-Cut Theorem

- Die Beschreibung inspirierte Ford und Fulkerson zu ihrem berühmten Max-Flow Min-Cut Theorem: *“The maximum amount of flow is equal to the minimum capacity of the cuts of the network that separate all sources from all destinations.“*
- Untenstehendes Beispiel zeigt ein einfaches Netzwerk und einen möglichen Schnitt darin. Die Kapazität des Schnittes ist $c(q,b) + c(a,b) + c(a,s) = 1 + 2 + 1 = 4$.

- Abbildung zeigt maximale Kapazitäten
- Maximaler Fluss ist 4

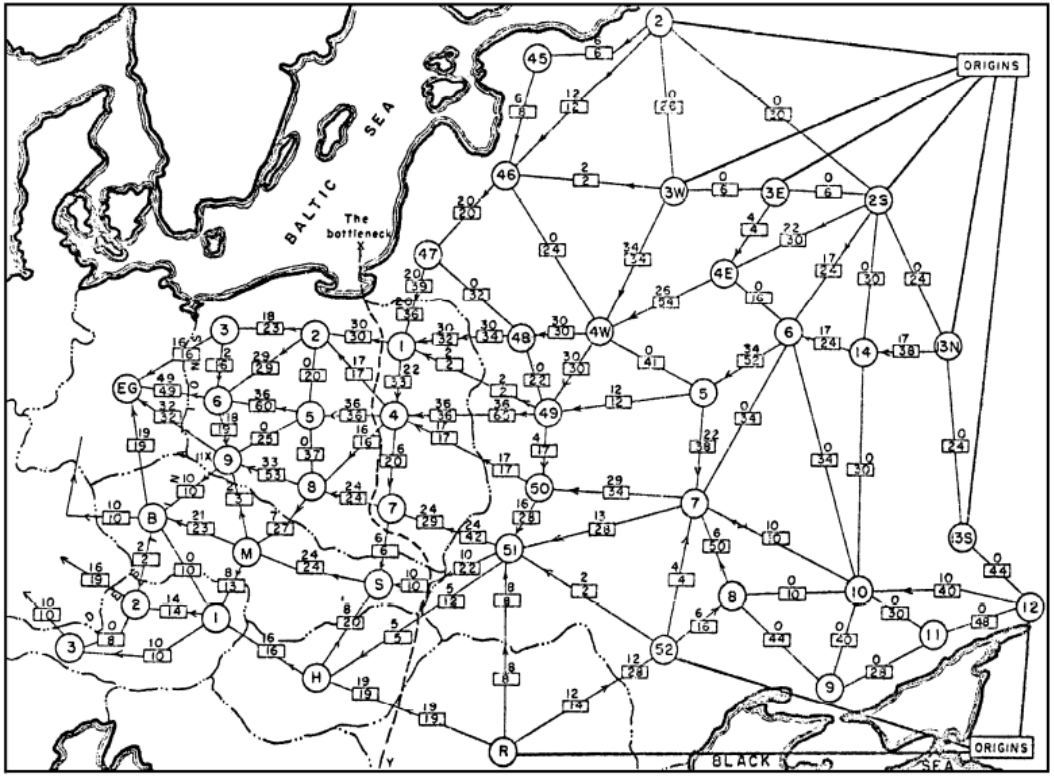
- Min-Cut ist das Duale Problem zum Max-Flow



Harris and Ross 1955

- Ford-Fulkerson’s Artikel zitiert einen geheimen Bericht von Harris and Ross betitelt: “*Fundamentals of a Method for Evaluating Rail Net Capacities”,* datiert auf den 24.10.1955 und geschrieben für die US Air Force.
- Der Bericht behandelt ein relativ großes Problem maximaler Flüsse, dass sich auf das Eisenbahnnetz im Westen der UDSSR und Osteuropa bezieht.
- Das Interesse der Autoren waren nicht maximale Flüsse sondern vielmehr minimale Schnitte (cut) in der sowjetischen Bahn-Infrastruktur
- Zitat: “Air power is an effective means of interdicting an enemy’s rail system, and such usage is a logical and important mission for this arm.”

Abbildung nach Harris and Ross (1955)
zitiert nach Alexander Schrijver, Uni Amsterdam



Überblick

1. Maximale Flüsse in Netzwerken
2. **Kantenorientierte Rundreisen**

Kantenorientierte Rundreisen

- Aufgabenstellungen

  1. Erstellen Sie eine distanzminimale Tour für einen Briefträger innerhalb eines Postzustellbezirkes!
  2. Legen Sie eine Sammeltour für ein Fahrzeug der Entsorgungswirtschaft fest!

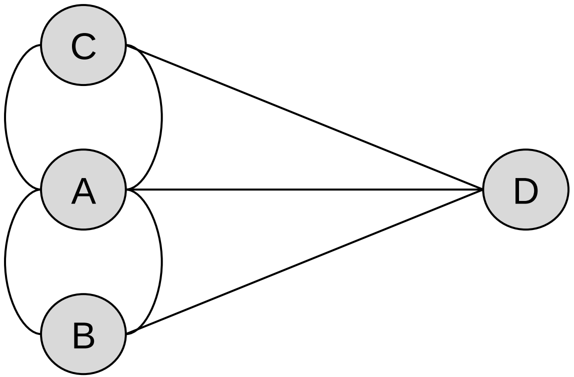
- Beide Aufgabenstellungen fragen nach einer Kosten-, bzw. Distanz-minimalen Rundtour, bei der alle Kanten eines zusammenhängenden Netzwerkes mindestens einmal durchlaufen werden müssen.

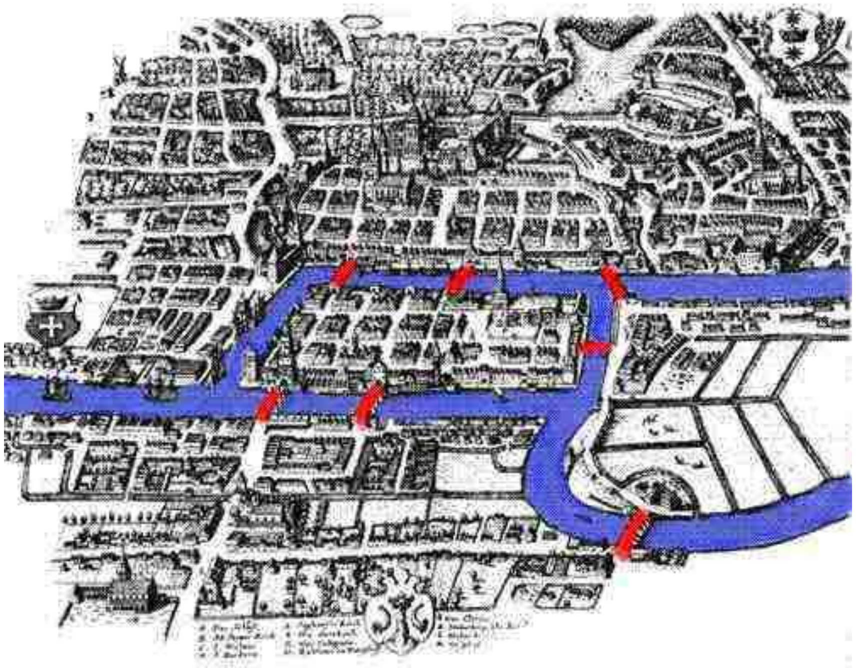
Zwei Fälle

1. Fall: Es ist möglich eine Tour zu finden, in der jede Kante genau einmal besucht wird
   → Euler-Netzwerk

2. Fall: Es müssen Kanten mehrfach besucht werden
   → Kein Euler-Netzwerk, Kostengünstige Erweiterung zum Euler-Netzwerk möglich

Euler-Netzwerk: Das Königsberger Brückenproblem





Aufgabe:

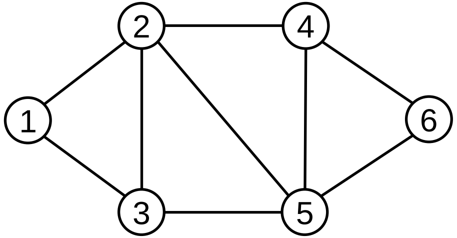
Gibt es einen Rundweg in Königsberg, auf dem man jede Brücke genau einmal überquert?

(Euler, 1736)

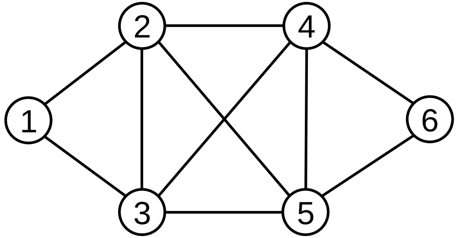
Euler-Netzwerke und Euler-Touren

- Ein Euler-Netzwerk ist ein zusammenhängendes Netzwerk, in dem jeder Knoten einen **geraden Knotengrad** von mindestens zwei besitzt.
- Eine Euler-Tour ist ein Zyklus in einem Netzwerk, der jede Kante genau einmal durchläuft. (Start- und Endknoten sind identisch bei einer Tour.)
- Aussage: In jedem Euler-Netzwerk gibt es eine Euler-Tour
- Euler-Touren in einem Euler-Netzwerk sind nicht eindeutig
- Beachte jedoch: Da alle Kanten im Netzwerk besucht werden müssen, sind alle Euler-Touren gleichwertig (→ jede Euler-Tour ist optimal)

→ Kantengewichte können vernachlässigt werden



Kein Euler-Netzwerk



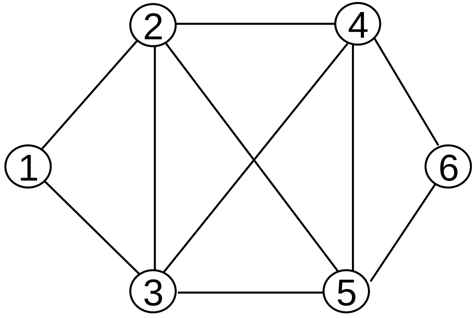
Euler-Netzwerk

Bestimmung einer Euler-Tour in einem Euler-Netzwerk

Gegeben: Ein Euler-Netzwerk mit eindeutig indizierten Knoten

- Ausgehend von einem (beliebig wählbaren) Startknoten *s*
  - Wähle aus den zum aktuellen Knoten *i* inzidenten und bisher unbesuchten Kanten (i, j) diejenige, welche zu dem Knoten mit dem kleinsten Index *j* führt
  - Wenn der Pfad an einem Knoten angekommen ist, welcher keine unbesuchte inzidente Kante hat (→ Zyklus), dann
    - Wähle aus den Knoten im Zyklus, welche noch unbesuchte inzidente Kanten haben, denjenigen mit dem kleinsten Index aus und setze von dort aus die Suche fort
    - Die Suche ist beendet, wenn alle Kanten besucht wurden
- Anschließend werden die Zyklen zu einer Gesamttour zusammengesetzt

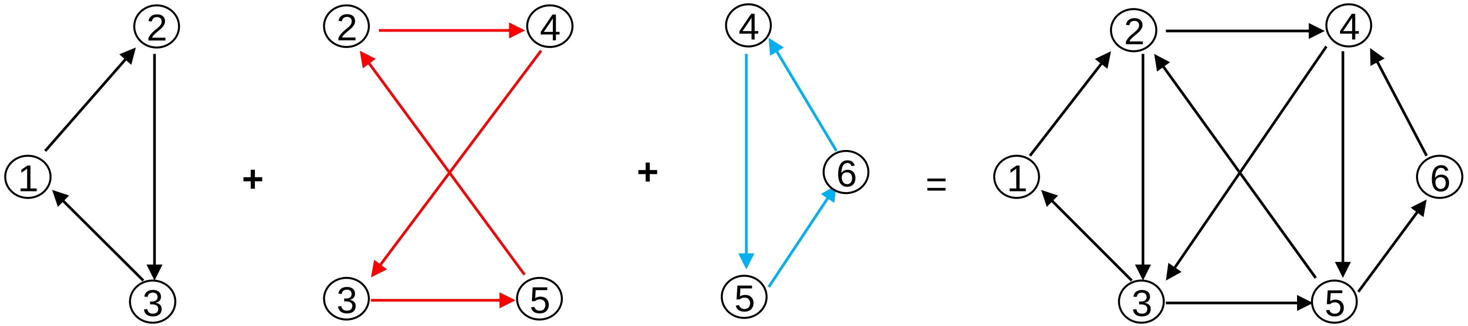
Beispiel für die Suche nach einer Euler-Tour (1)



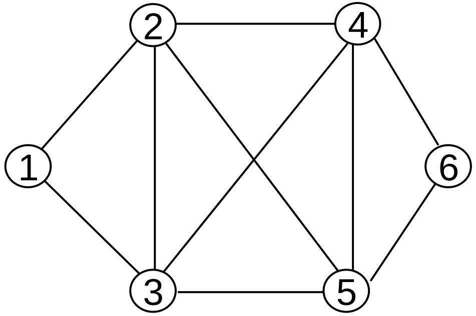
Bei Wahl von Startknoten 1:

- $Z_1 = (1,2,3,1)$,
- $Z_2 = (2,4,3,5,2)$,
- $Z_3 = (4,5,6,4)$

→ $Z = (1,2,4,5,6,4,3,5,2,3,1)$

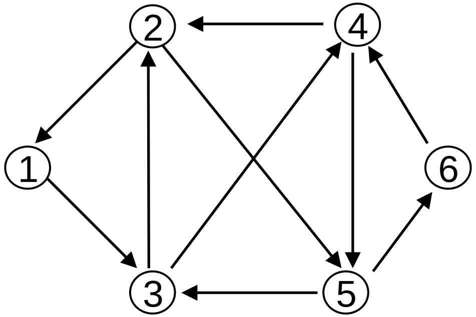


Beispiel für die Suche nach einer Euler-Tour (2)



Bei Wahl von Startknoten 6:

- $Z = (6,4,2,1,3,2,5,3,4,5,6)$



Allgemeiner Fall: Netzwerk ist kein Euler-Netzwerk

Es existiert keine Euler-Tour, wenn das Netzwerk kein Euler-Netzwerk ist!
→ Bei einer Rundreise müssen manche Kanten mehrfach durchlaufen werden

Gesucht: Die kürzeste Rundreise, bei welcher alle Kanten mindestens einmal besucht werden

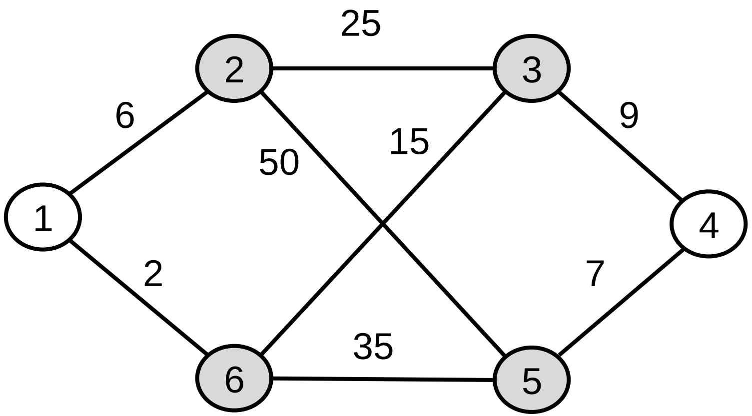
Lösungsidee:

- Bestimme die **distanzminimale Menge an Kanten,** welche mehrfach durchlaufen werden müssen und **erweitere** das bestehende Netzwerk N um eine **Menge** von Kanten, um ein Euler-Netzwerk N_e zu erhalten
- Bilde anschließend eine Eulertour auf dem Euler-Netzwerk N_e

Beachte:

→ Durch die distanzminimale Erweiterung ist jede Euler-Tour in dem erweiterten Netzwerk N_e eine optimale Lösung für das Ausgangsproblem!

Beispiel



Die Knoten 2, 3, 5, 6 haben jeweils den Knotengrad 3

→ es liegt **kein Euler-Netzwerk** vor!

Art und Weise der Netzwerkerweiterung

- Die Anzahl an Knoten mit ungeradem Knotengrad in einem zusammenhängenden Netzwerk ist immer gerade

- Durch die Verbindung von zwei Knoten mit ungeradem Grad durch eine Kante erhalten beide Knoten einen geraden Knotengrad
  → ein Netzwerk mit $2k$ Knoten mit ungeradem Grad kann durch Hinzunahme von k Kanten zu einem Euler-Netzwerk gemacht werden

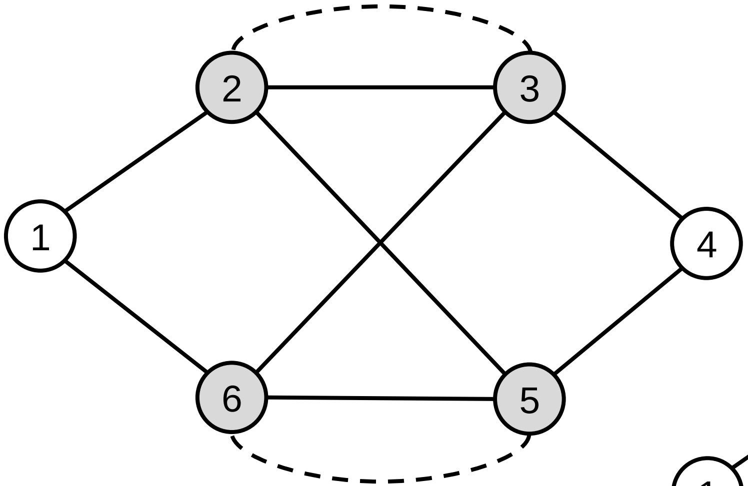
Ziel: Bestimmung der distanzminimalen Menge dieser Kanten!

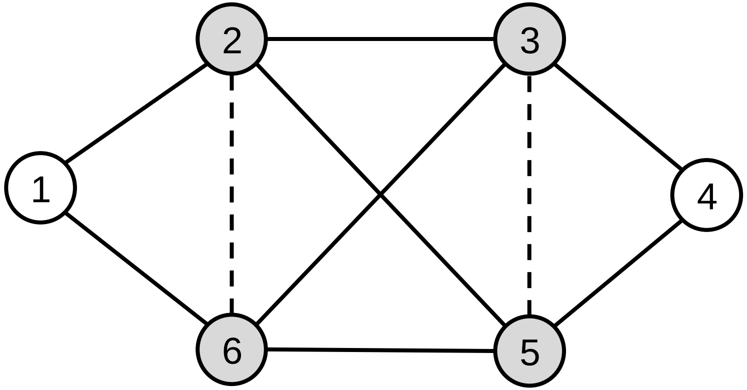
→ Gesucht sind also Knotenpaare, welche jeweils durch eine Kante verbunden sind

→ Ein solches Problem nennt man ein **Paarungs**- oder **Matching-Problem**

→ Gesucht ist ein **distanzminimales Matching**

Beispiel für mögliche Ergänzungen zu einem Euler-Netzwerk





Bildung eines Hilfsnetzwerk für das Matching

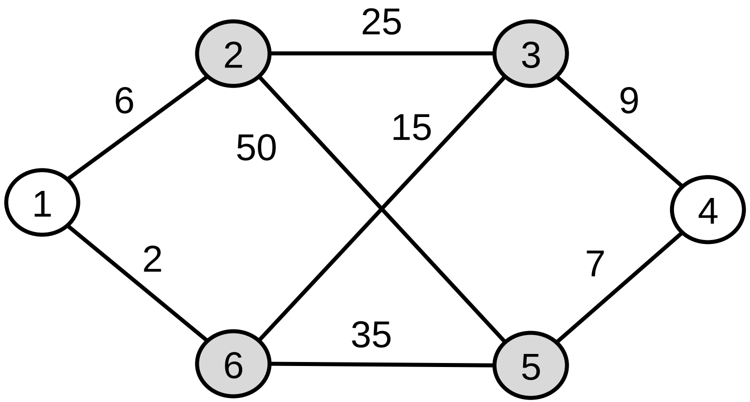
- Bilde ein Hilfsnetzwerk N_u wie folgt:
  - Die Menge V_u der Knoten in N_u entspricht der Menge aller Knoten im Ausgangsnetzwerk mit ungeradem Knotengrad
  - Füge zwischen allen Knoten in V_u je eine Kante ein (es entsteht ein vollständiges Netzwerk)

- Annahme: Die gegebene Infrastruktur kann nicht verändert werden, d.h. Verbindungen zwischen zwei Knoten können nur über bestehende Kanten erfolgen

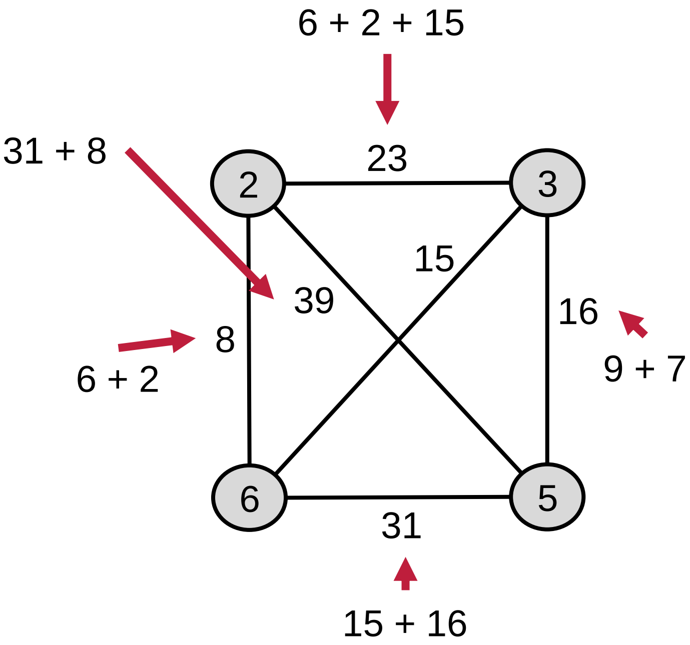
Daher:

- Bestimme die distanzminimale Verbindung (kürzester Weg) für jedes Knotenpaar (i, j) und verwende diese als Kantengewicht d_{ij} für das Hilfsnetzwerk N_u :
  - d_{ij} = Länge des kürzesten Wegs von i nach j im Ausgangsnetzwerk N

Bildung des Hilfsnetzwerks im Beispiel



Bestimme kürzeste Wege zwischen den Knotenpaaren



Formulierung des Matching-Problems als Lineares Programm

Gesucht: Ein minimales Matching im Hilfsnetzwerk $N_u = (V_u, E_u)$

→Formulierung als lineares Programm:

$$\min z = \sum_{(i,j)\in E_u} d_{ij} x_{ij}$$

u.d.N.

$$\sum_{(i,j)\in \delta_k} x_{ij} = 1 \quad \text{für alle } k \in V_u$$

$x_{ij} \in \{0; 1\}$ für alle $(i, j) \in E_u$

wobei:

$$x_{ij} = \begin{cases} 1, \text{wenn Kante } (i,j) \text{ gewählt wird} \\ 0, \text{sonst} \end{cases}$$

E_u : Menge aller Kanten in N_u
V_u : Menge aller Knoten in N_u
δ_k : Menge der zu Knoten k inzidenten Kanten

Matching-Problem am Beispiel

$\min z = 23\, x_{23} + 39\, x_{25} + 8\, x_{26} + 16\, x_{35} + 15\, x_{36} + 31\, x_{56}$

u.d.N.

$x_{23} + x_{25} + x_{26} = 1$ (Knoten 2)

$x_{23} + x_{35} + x_{36} = 1$ (Knoten 3)

$x_{25} + x_{35} + x_{56} = 1$ (Knoten 5)

$x_{26} + x_{36} + x_{56} = 1$ (Knoten 6)

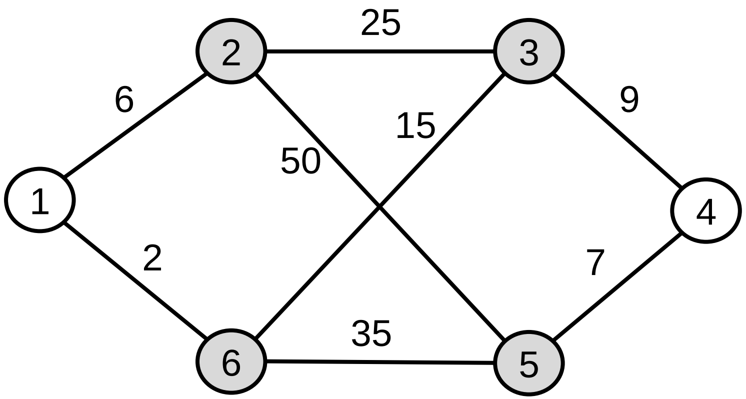
$x_{ij} \in \{0; 1\}$ für alle $(i, j) \in E_u$

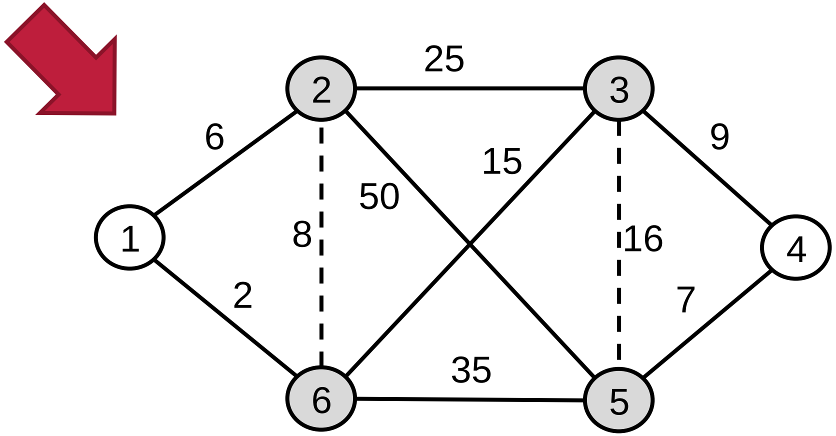
Lösung:

$z = 24$; $x_{26} = 1$; $x_{35} = 1$; alle anderen $x_{ij} = 0$

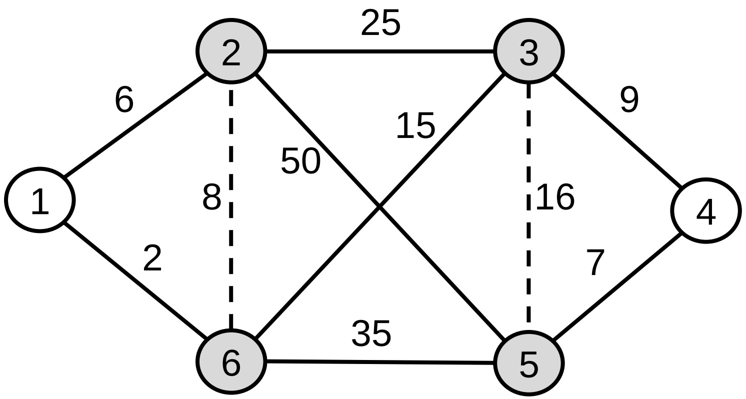
→ Kanten (2,6) und (3,5) werden zum Ausgangsnetzwerk hinzugefügt

Resultierendes Euler-Netzwerk

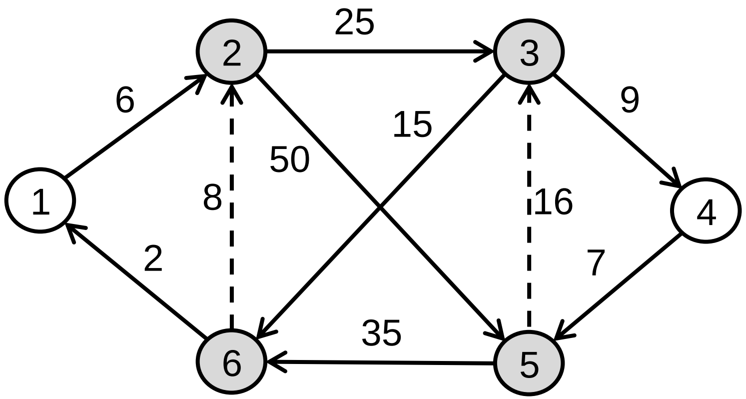




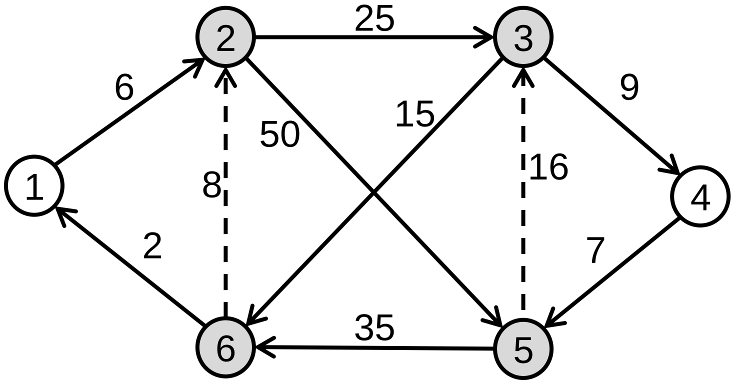
Eulerkreissuche im erweiterten Netzwerk



$$Z_1 = (1,2,3,4,5,3,6,2,5,6,1) = Z$$



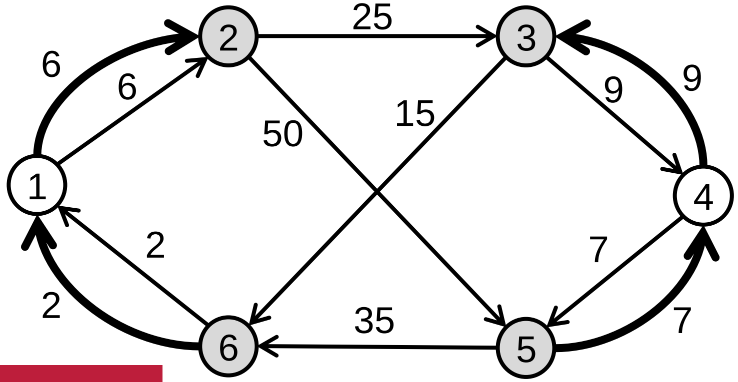
Interpretation des Eulerwegs im Ausgangsnetzwerk



Zyklus im erweiterten Netzwerk N_e

$$Z = (1,2,3,4,\mathbf{5},\mathbf{3},\mathbf{6},\mathbf{2},5,6,1)$$

Beachte: Die zusätzlichen Kanten im erweiterten Netzwerk N_e entsprechen den kürzesten Wegen zwischen den jeweiligen Knoten im Ausgangsnetzwerk N



Zyklus im Ausgangsnetzwerk N:

$$Z = (1,2,3,4,\mathbf{5},\mathbf{4},\mathbf{3},\mathbf{6},\mathbf{1},\mathbf{2},5,6,1)$$

→ Die Kanten (1,2) ; (1,6) ; (3,4) ; (4,5) werden doppelt besucht!

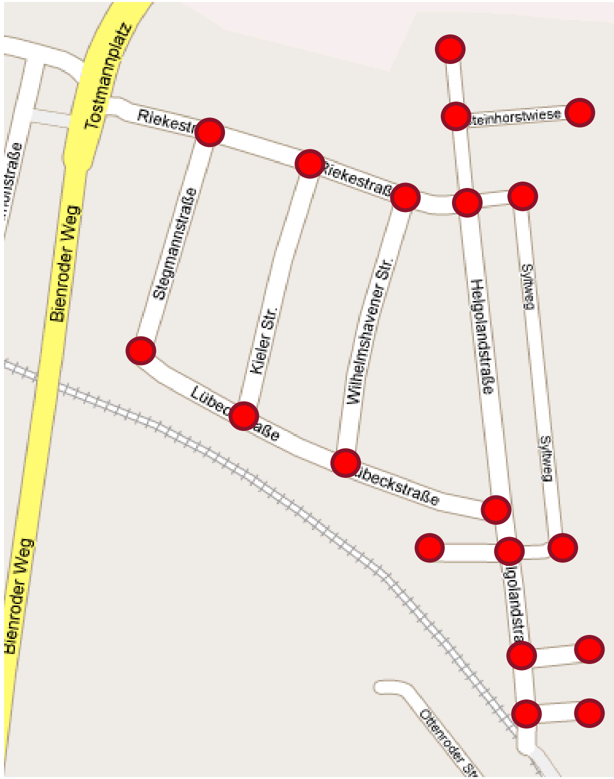
Zusammenfassung: Lösung des kantenorientierten Rundreise-problems auf allgemeinen Netzwerken (Briefträgerproblem)

Gegeben: Ein zusammenhängendes Netzwerk $N = (V, E)$

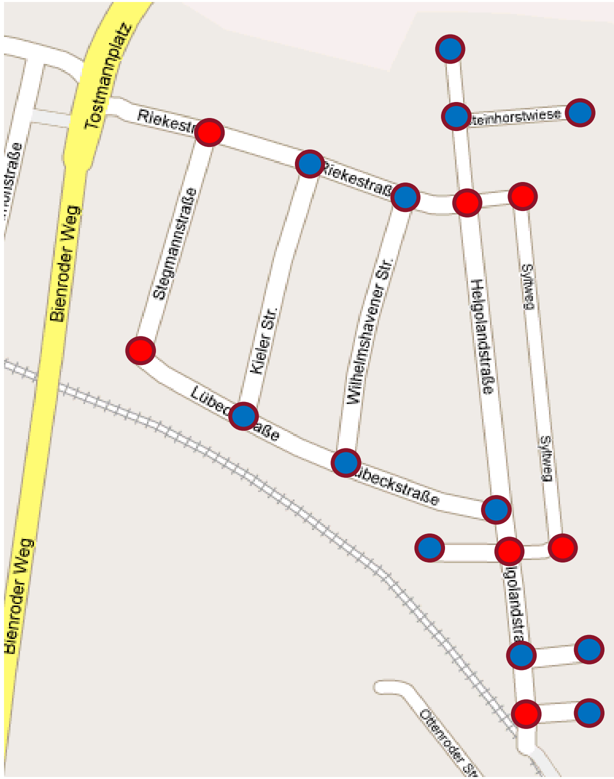
1. Bestimme die Menge V_u aller Knoten in V mit ungeradem Grad
   - Wenn $V_u = \emptyset$, so ist N ein Euler-Netzwerk

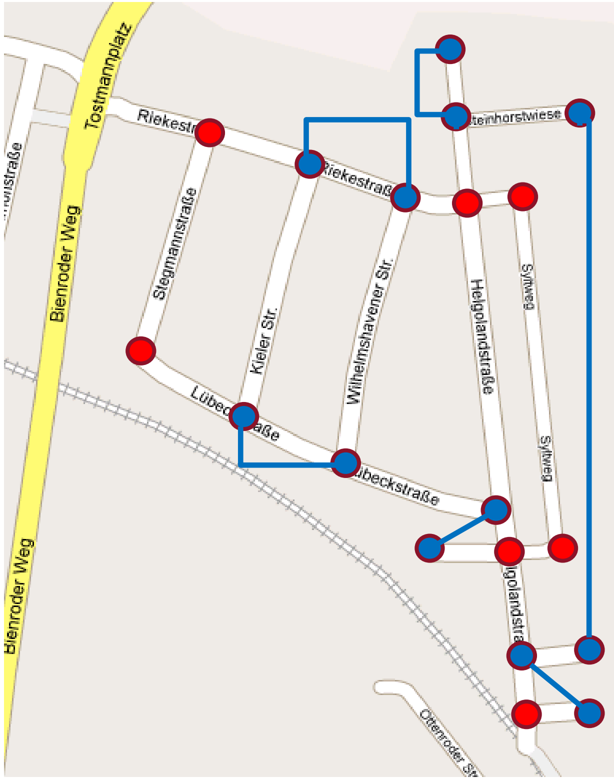
     → direkte Bestimmung einer Euler-Tour in N → Ende.
   - Sonst:
2. Bilde einen vollständiges Hilfsnetzwerk $N_u = (V_u, E_u)$
   - Wähle als Kantengewicht für jede Kante $(i, j) \in E_u$ die Länge des **kürzesten Wegs** zwischen *i* und *j* im Ausgangsnetzwerk *N*
3. Bestimme ein **distanzminimales Matching** (Paarung) in N_u (z.B. mit LP)
4. Bilde das erweiterte Netzwerk N_e durch **Hinzufügen der Matching-Kanten** zum Ausgangsnetzwerk N
5. Bestimme eine **Euler-Tour** im erweiterten Netzwerk N_e
6. Bilde eine Tour im Ausgangsnetzwerk N mittels **Ersetzung** der Matching-Kanten in N_e durch die **zu Grunde liegenden kürzesten Wege in** N

Entsorgungstour in der Schuntersiedlung

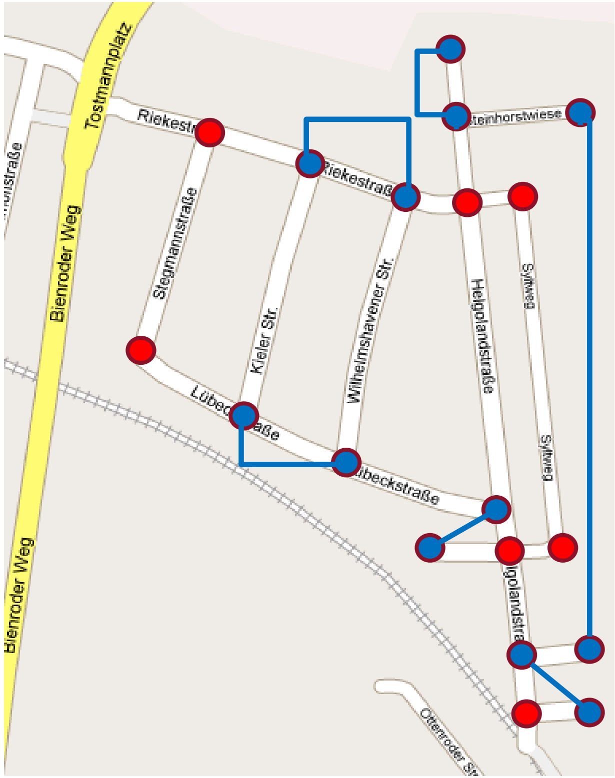


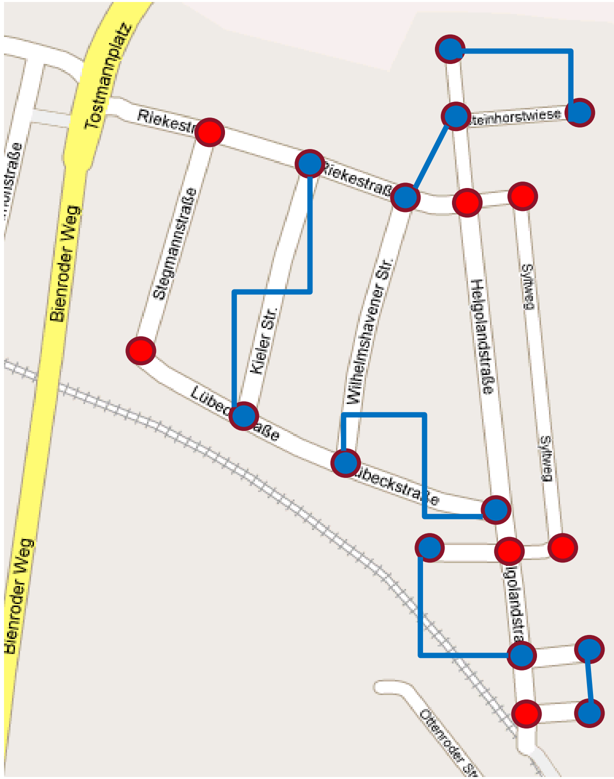
Zuordnung von Kreuzungen mit ungeradem KG





Alternative Euler-Netzwerke

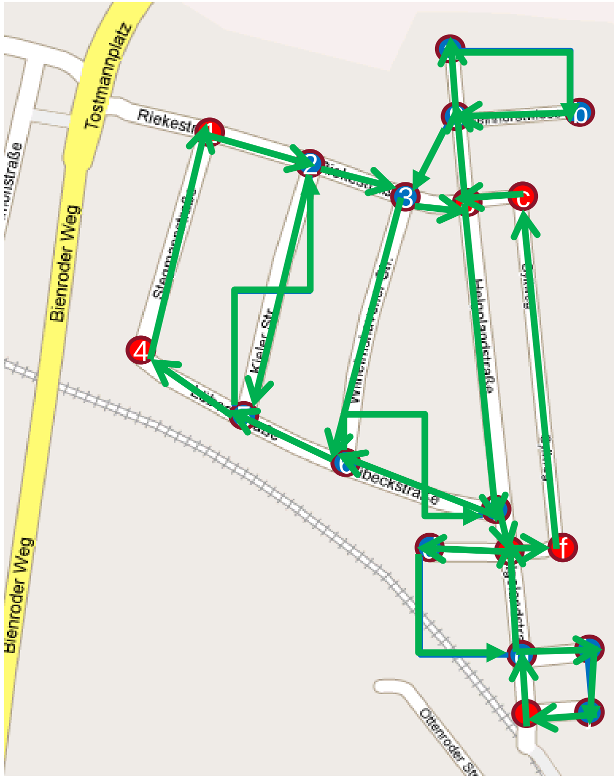




Bilden einer Euler-Tour

- Regel: Gehe entlang nicht markierter Kante zum benachbarten Knoten mit kleinstem Index.
- Zyklus 1: 1-2-3-6-5-2-5-4-1
- Zyklus 2: 3-8-7-6-7-e-d-g-e-
  f-c-8-9-3
- Zyklus 3: 9-a-b-9
- Zyklus 4: g-h-j-i-g

- Zyklus 2 → Zyklus 1: Stelle 3
- Zyklus 3 → Zyklus 2: Stelle 9
- Zyklus 4 → Zyklus 2: Stelle g



Zusammenfassung

- Optimierungsprobleme können in Graph-Modellen abgebildet werden.
- Deren Lösung bedingt einen prozedural beschriebenen Algorithmus.
- Kürzeste aufspannenden Bäume weisen lokale Entscheidungen auf.
- Kürzeste Wege benötigen Markierungen zur Strukturierung der Suche.
- Maximale Flüsse werden durch iterative Flusserhöhungen gelöst.
- Kantenorientierte Rundreisen werden durch Dekomposition beschrieben.
- Alle gezeigte Verfahren garantieren eine „schnelle“ optimale Lösung.