

# ICPC

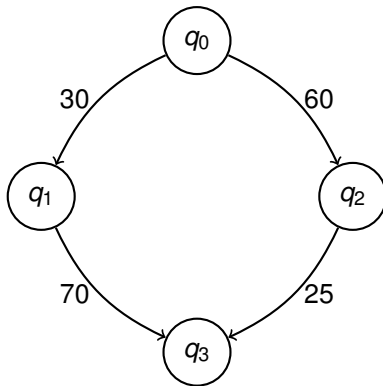
## Graphen 3

Tobias, Julian, Jakob, Tobias | 6. Juni 2018

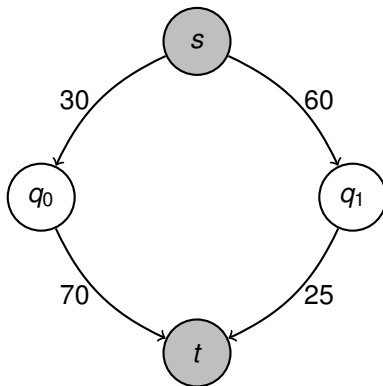
ITI WAGNER, IPD TICHY



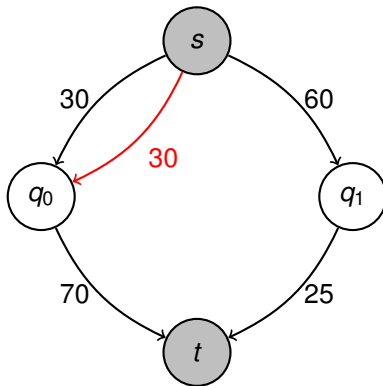
- 1 Einführung
- 2 Max-Flow Algorithmen
  - Ford-Fulkerson
  - Edmonds-Karp
- 3 Julian
- 4 Tobias T



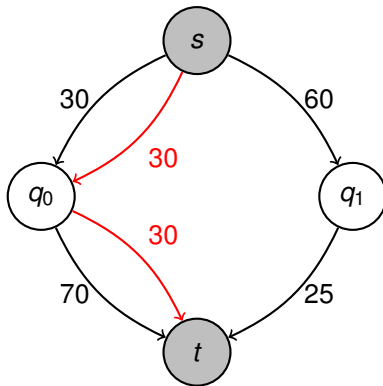
Gegeben gerichteter Graph



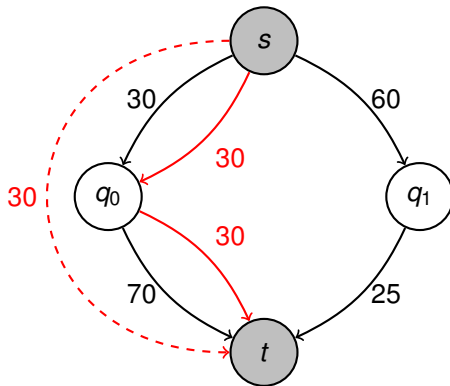
s: source, t:sink



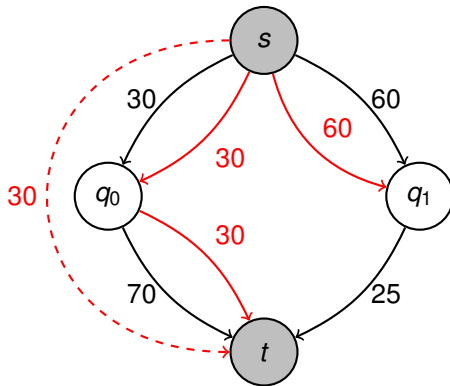
Fluss



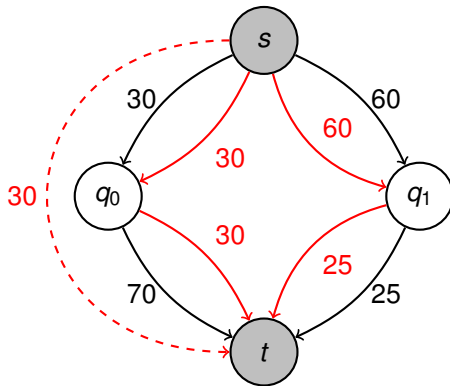
Flusserhaltung

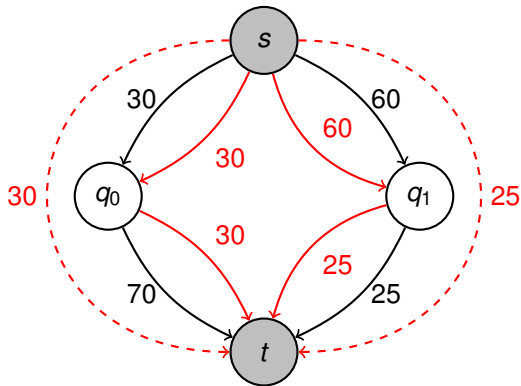


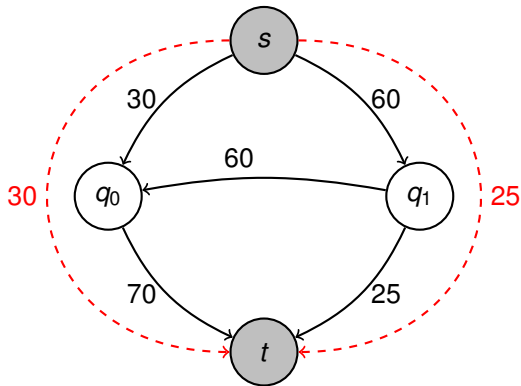
Wert eines s-t-Flusses

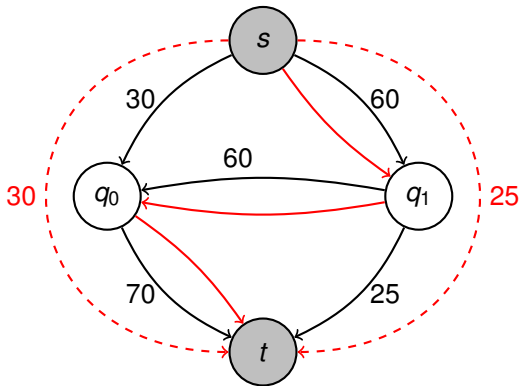


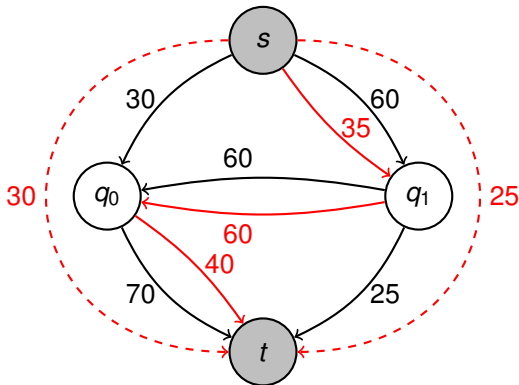




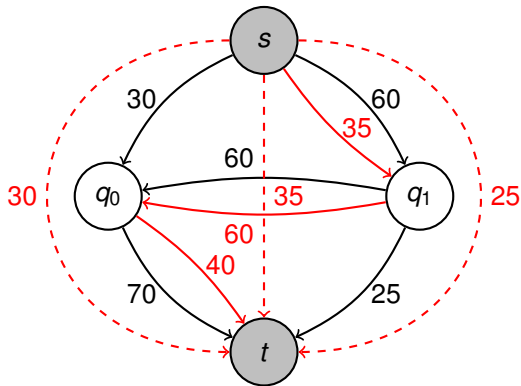








Exzess: Werte entsprechend der Kantenkapazität abzüglich bereits vorhandener Flüsse



- Schwierigkeit im Erkennen der Aufgaben  
Proble UVa xxx

- Schwierigkeit im Erkennen der Aufgaben
- Seit 2013 vermehrtes vorkommen in contests  
decider Problem



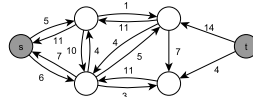
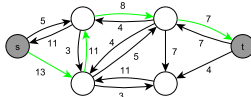
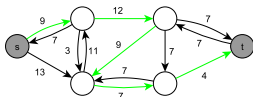
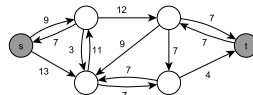
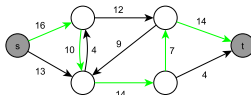
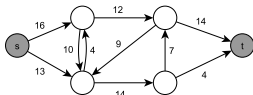
Idee:

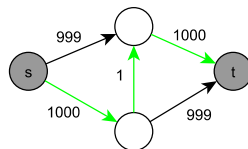
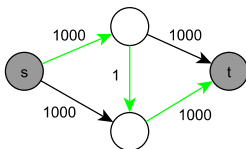
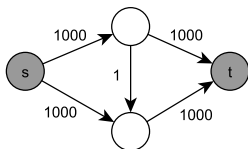
- Starte mit dem leeren Fluss
- Bestimme erweiternden Pfad (augmenting path)  $P$ , auf dem der Fluss von  $s$  nach  $t$  vergrößerbar ist
- Erweitere die Lösung um Pfad  $P$
- Wiederhole so oft, wie es einen passenden Pfad  $P$  gibt

Frage: Wie kann  $P$  gefunden werden?

- Greedy Algorithmus veröffentlicht in 1956 von L. R. Ford, Jr. und D. R. Fulkerson
- Verwendet Tiefensuche um den erweiternden Pfad  $P$  zu bestimmen
- Die Lösung wird um  $P$  erweitert indem,
  - die geringste Kapazität  $f$  einer Kante in  $P$  bestimmt wird
  - die Kapazitäten aller Kanten in  $P$  um  $f$  verringert werden
  - die Kapazitäten aller Gegenkanten um  $f$  erhöht werden
  - der maximale Fluss um  $f$  erhöht wird

# Ford-Fulkerson Algorithmus





- Im Worst-Case wird der maximale Fluss pro Iteration nur um 1 erhöht  
⇒ Laufzeit in  $\mathcal{O}(|f^*| \cdot |E|)$ , wobei  $|f^*|$  der Wert des maximalen Flusses beschreibt
- Deshalb **nicht** für ICPC-Aufgaben geeignet!

- 1972 von J. Edmonds und R. M. Karp veröffentlicht
  - Verwendet Breitensuche um den kürzesten erweiternden Pfad  $P$  zu bestimmen
  - Erweiterung der Lösung um  $P$  analog zu Ford-Fulkerson
  - Die Länge des erweiternden Pfades ist monoton steigend
  - Es sind maximal  $|V| \cdot |E|$  Iterationen notwendig
- ⇒ Laufzeit in  $\mathcal{O}(|V| \cdot |E|^2)$

---

## Algorithm 1: Edmonds-Karp

---

**Function** *Max-Flow* ( $G = (V, E)$ ,  $s$ ,  $t$ ,  $c$ )

$maxFlow = 0$

**do**

        find augmenting path  $P$  using BFS

$f = \min(c(u, v) | (u, v) \in P)$

**foreach**  $(u, v) \in P$  **do**

$c(u, v) -= f$

$c(v, u) += f$

**end**

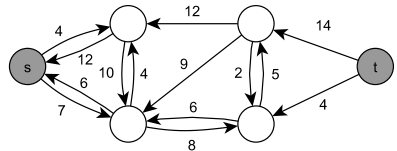
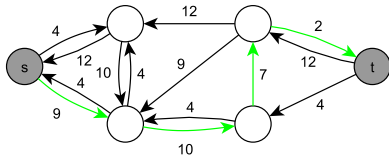
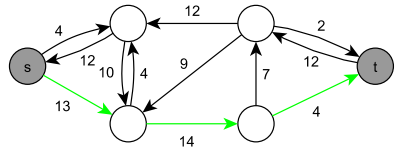
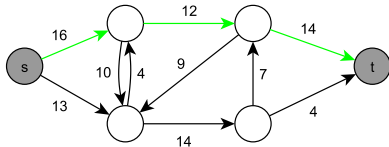
$maxFlow += f$

**while**  $P$  exists

**return**  $maxFlow$

---

# Edmonds-Karp Algorithmus



- In Adjazenzliste neben Zielknoten auch Kapazität und Verweis auf die Rückkante speichern
- In Breitensuche nur Kanten mit positiver Kapazität berücksichtigen
- Breitensuche abbrechen, sobald  $t$  erreicht wurde



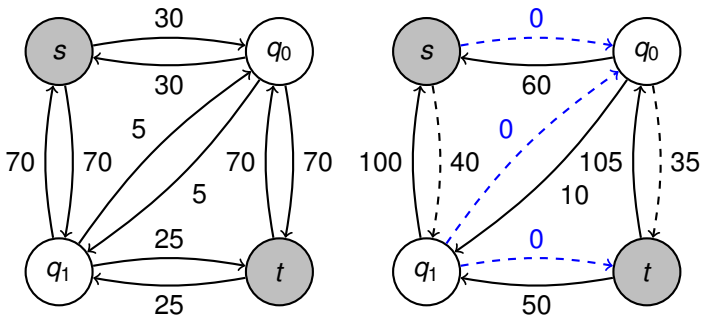
## Min-Cut

- Definiere Schnitt  $C = (S - \text{Komponente}, T - \text{Komponente})$  als Partition von  $V \in G$ , wobei  $s \in S - \text{Komponente}$  und  $t \in T - \text{Komponente}$
- Weiter sei die Schnittmenge  
 $c = \{(u, v) \in E \mid u \in S - \text{Komponente} \wedge v \in T - \text{Komponente}\}$
- Wähle  $c$  so, dass Max Flow von  $s$  nach  $t$  0 ist, für  $E' = E \setminus c$

## Max-Flow-Min-Cut-Theorem

- Ein maximaler Fluss im Netzwerk hat genau den Wert eines minimalen Schnitts.

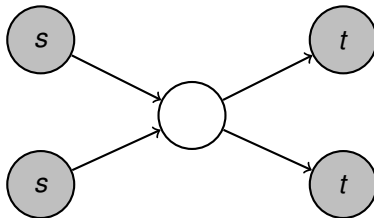
■ Bsp.:



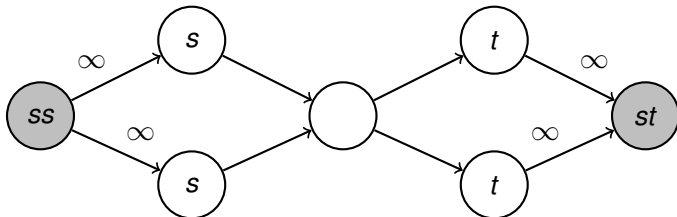
■ Hier

- $C = (\{s, q_1\}, \{t, q_0\})$
- $c = \{(s, q_0), (q_1, q_0), (q_1, t)\}$

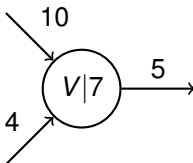
- Gegeben sei folgende Situation:



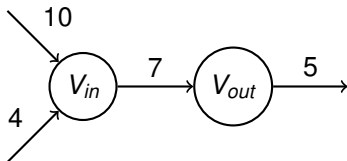
- Problem: Max-Flow Algorithmus kann nur mit einer Quelle und einer Senke arbeiten.
- Lösung: Erstelle Super-Quelle und Super-Senke und verbinde alle Quellen und Senken mit Kantengewicht  $\infty$



- Gegeben sind Knoten mit Kapazität.
- Bsp.:



- Gegeben sind Knoten mit Kapazität.
- Bsp.:



- Erkennen eines Netzwerkfluss-Problems nicht immer einfach
- Was hilft?
  - Übung
  - Übung
  - ...



- Erkennen eines Netzwerkfluss-Problems nicht immer einfach
- Was hilft?
  - Übung
  - Übung
  - ...

- Erkennen eines Netzwerkfluss-Problems nicht immer einfach
- Was hilft?
  - Übung
  - Übung
  - ...

- Situation: Die Titanic ist gesunken. Es soll ermittelt werden wie viele Menschen gerettet werden können.
- Eingabe:  $X, Y, P$  mit  $X, Y$  Dimension der Fläche ( $1 \leq X, Y \leq 30$ ) und  $P$  ( $P \leq 10$ ) die Anzahl von Personen, welche gleichzeitig auf ein Holzbrett können.

Symbol	Bedeutung
*	Menschen auf Treibeis
~	Eiskaltes Wasser
.	Trebbeis
@	Großer Eisberg
#	Großes Holzbrett

- Gegeben sei nun folgende Eingabe:

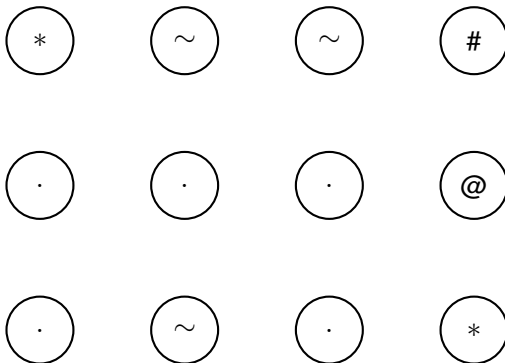
*	~	~	#
.	.	.	@
.	~	.	*

- Wandle in Graphen um...

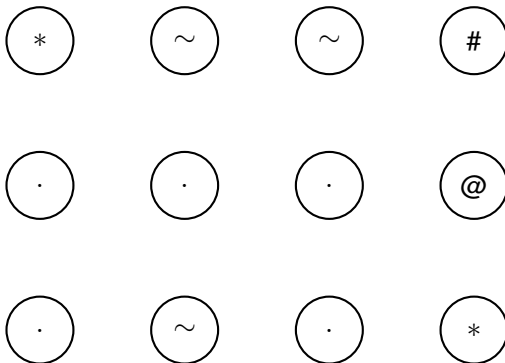
- Gegeben sei nun folgende Eingabe:

*	~	~	#
.	.	.	@
.	~	.	*

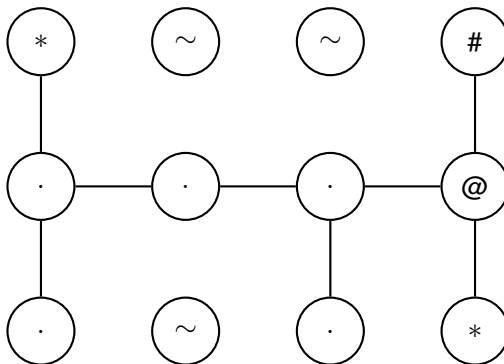
- Wandle in Graphen um...



■ Verbinde alle Knoten, über die ein Weg möglich ist...

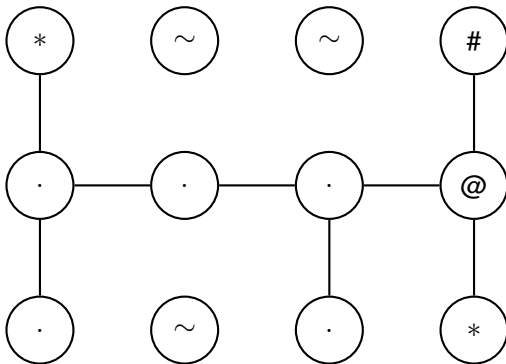


- Verbinde alle Knoten, über die ein Weg möglich ist...

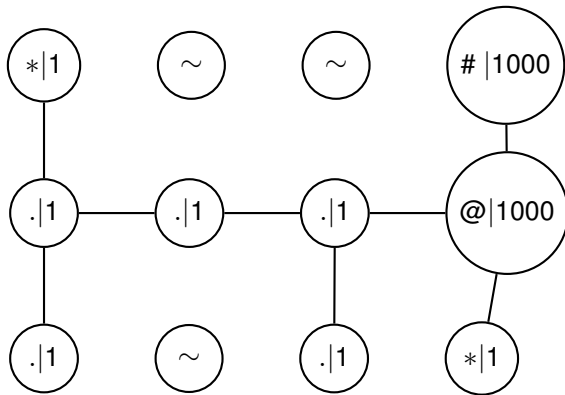


■ Füge Knotengewichte hinzu...

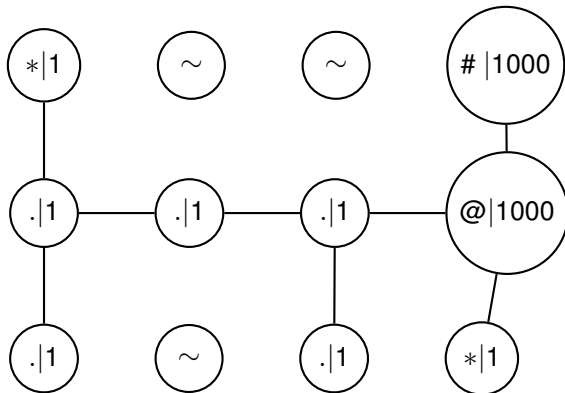




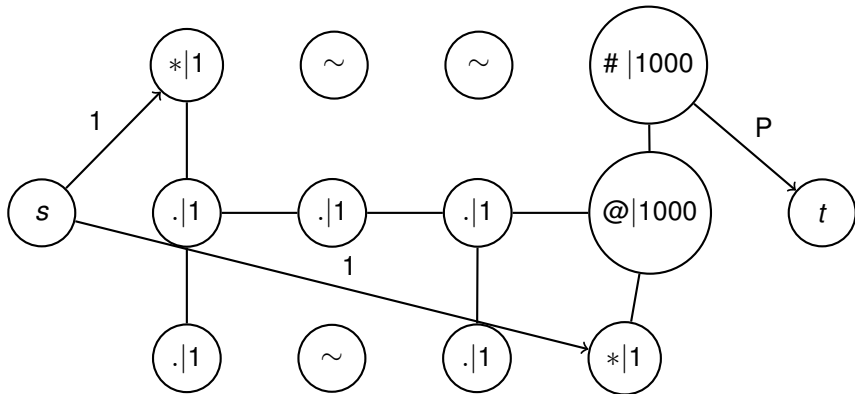
- Füge Knotengewichte hinzu...



■ Verbinde alle Menschen mit s und alle Holzbretter mit t...



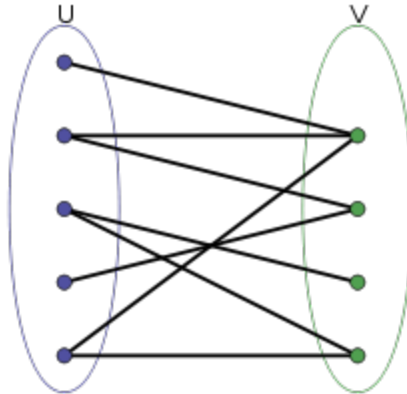
- Verbinde alle Menschen mit s und alle Holzbretter mit t...



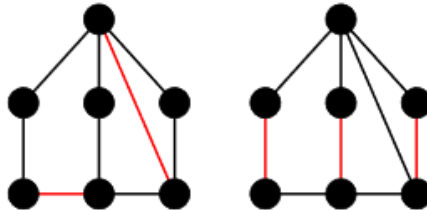
- Bem.: Knotengewichte müssen noch aufgelöst werden

# Bipartiter Graph

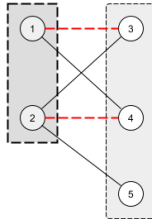
## ■ Bipartiter Graph



- Definitionen: Matching, maximales Matching, kardinalitätsmaximales Matching, perfektes Matching



- Kurz auf Laufzeit eingehen
- Beispiel: Primzahlen (Competitive Programming 3, Seite 180)
- Definitionen: Max Independent Set, Min Vertex Cover, Königs Theorem: —Min Vertex Cover— = —grtes Matching—



- Beispiel: Guardian of Decency (Competitive Programming 3, Seite 182)
- (Je nach verbleibender Zeit:) noch mehr Graphentheorie: bipartit  $\Leftrightarrow$  keine ungeraden Kreise, ...