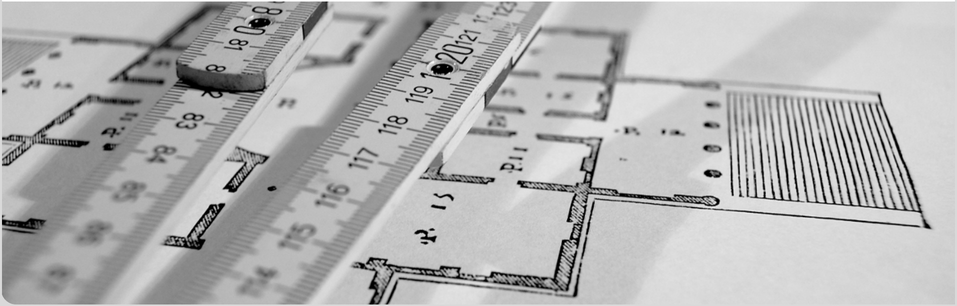
Karlsruher Institut für Technologie

ICPC

Graphen 3

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

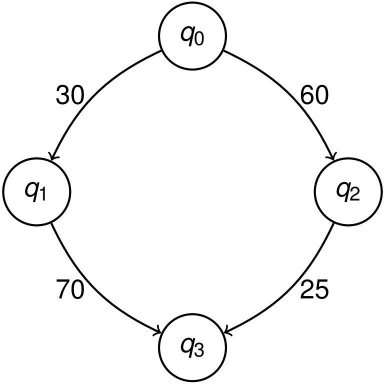
ITI WAGNER, IPD TICHY



Outline/Gliederung

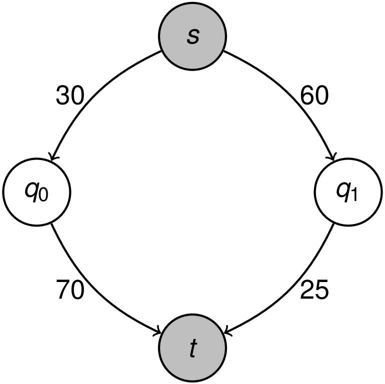
1. Einführung
2. Max-Flow Algorithmen
   - Ford-Fulkerson
   - Edmonds-Karp
3. Min-Cut
4. Sonderfälle
5. Max-Flow Modellierung
6. Bipartite Matching

Einführung



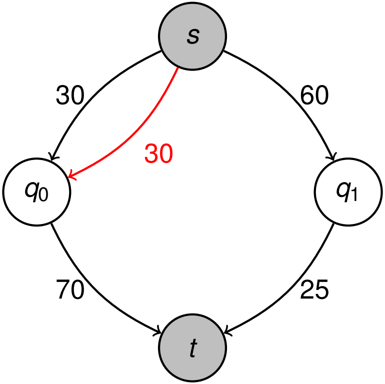
Gegeben gerichteter Graph

Einführung



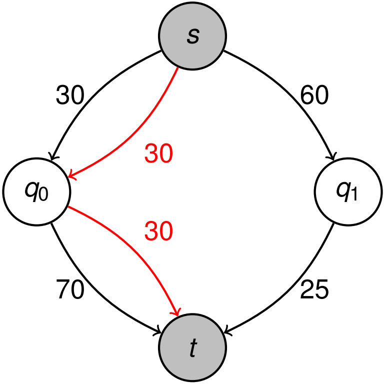
s: source, t:sink

Einführung



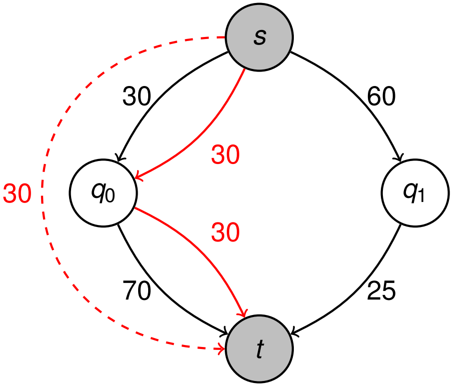
Fluss

Einführung



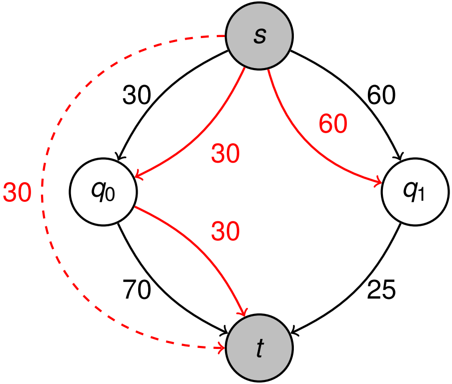
Flusserhaltung

Einführung

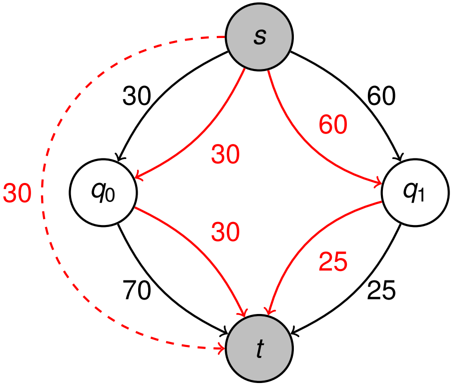


Wert eines s-t-Flusses

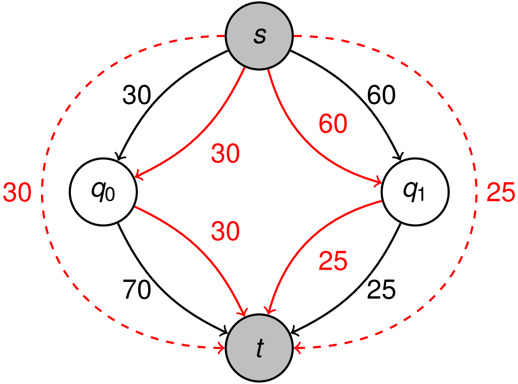
Einführung



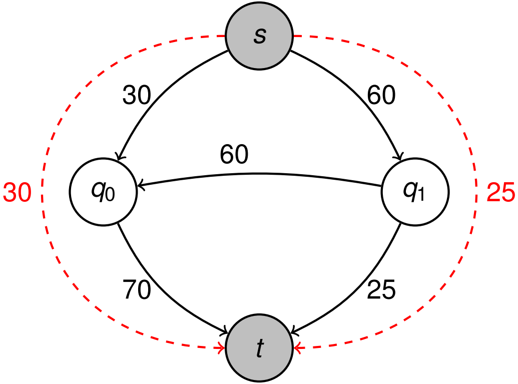
Einführung



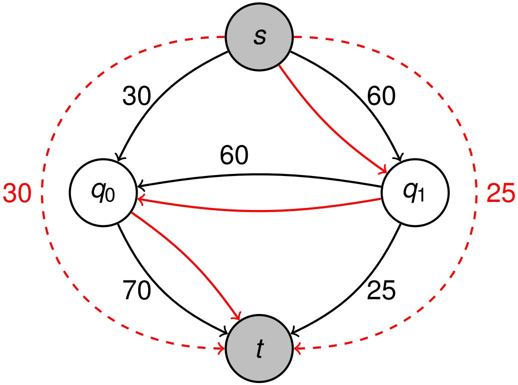
Einführung



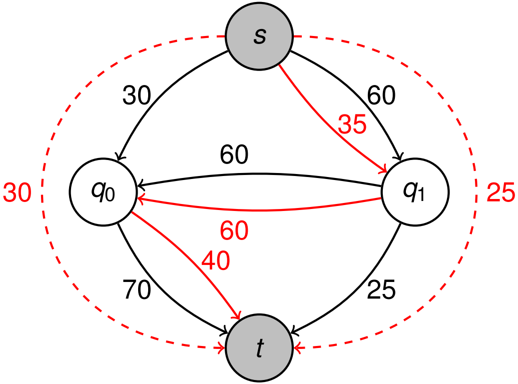
Einführung



Einführung

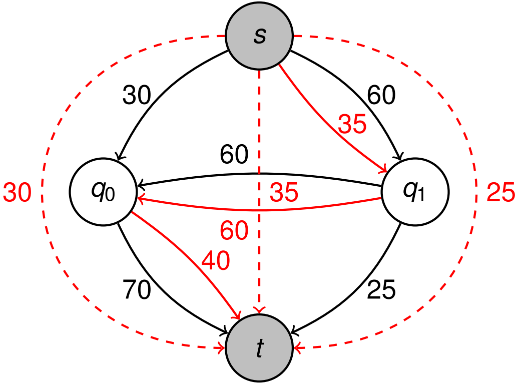


Einführung



Exzess: Werte entsprechend der Kantenkapazität abzüglich bereits vorhandener Flüsse

Einführung



Flussprobleme

- Schwierigeit im Erkennen der Aufgaben

Flussprobleme

- Schwierigeit im Erkennen der Aufgaben
- Seit 2013 vermehrtes vorkommen in contests

  decider Problem

Bestimmung des maximalen Flusses

Idee:

- Starte mit dem leeren Fluss
- Bestimme erweiternden Pfad (augmenting path) P
  - $\Rightarrow$ Ein erweiternder Pfad ist ein einfacher Pfad, der nur Kanten mit positiver Kapazität enthält
- Erweitere die Lösung um P
- Wiederhole so oft, wie es einen passenden Pfad P gibt

Frage: Wie kann P gefunden werden?

Bestimmung des maximalen Flusses

Idee:

- Starte mit dem leeren Fluss
- Bestimme erweiternden Pfad (augmenting path) P
  - $\Rightarrow$ Ein erweiternder Pfad ist ein einfacher Pfad, der nur Kanten mit positiver Kapazität enthält
- Erweitere die Lösung um P
- Wiederhole so oft, wie es einen passenden Pfad P gibt

Frage: Wie kann P gefunden werden?

Ford-Fulkerson Algorithmus

- 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 der Kanten 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

- 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 der Kanten 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

- 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 der Kanten 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

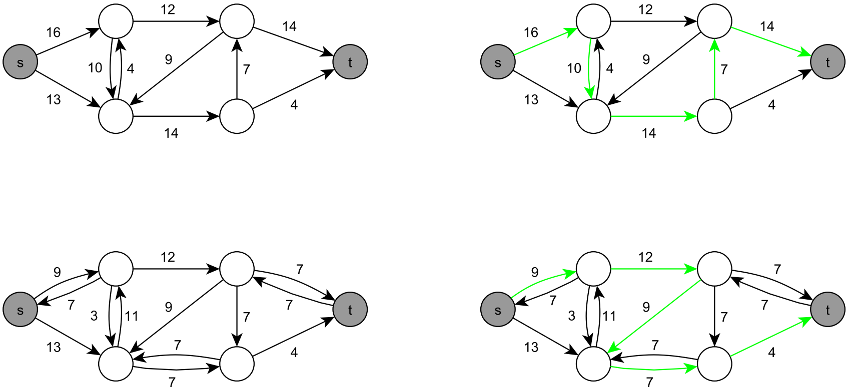
Ford-Fulkerson Algorithmus

- 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 der Kanten 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

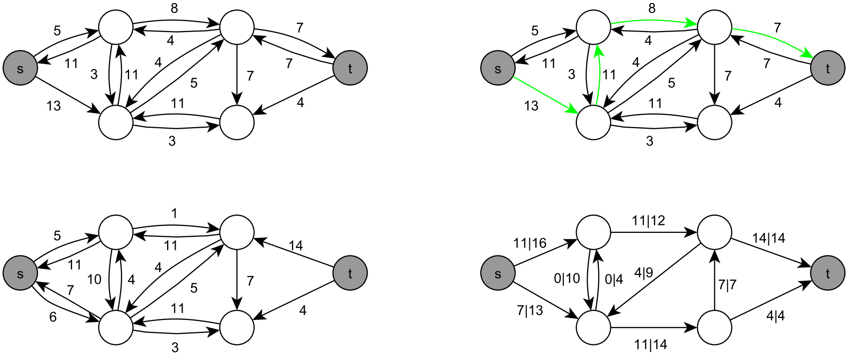
Ford-Fulkerson Algorithmus

- 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 der Kanten 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

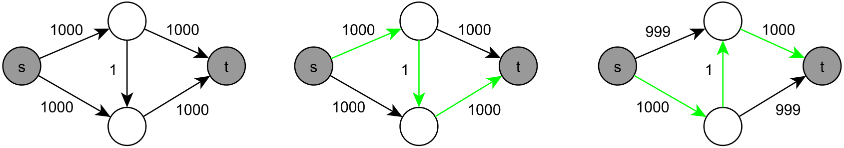
Ford-Fulkerson Algorithmus



Ford-Fulkerson Algorithmus



Ford-Fulkerson Algorithmus



- Im Worst-Case wird der maximale Fluss pro Iteration nur um 1 erhöht

$\Rightarrow$ Laufzeit in $\mathcal{O}(|f^*| \cdot |E|)$, wobei $|f^*|$ der Wert des maximalen Flusses beschreibt

- Deshalb **nicht** für ICPC-Aufgaben geeignet!

Edmonds-Karp Algorithmus

- 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

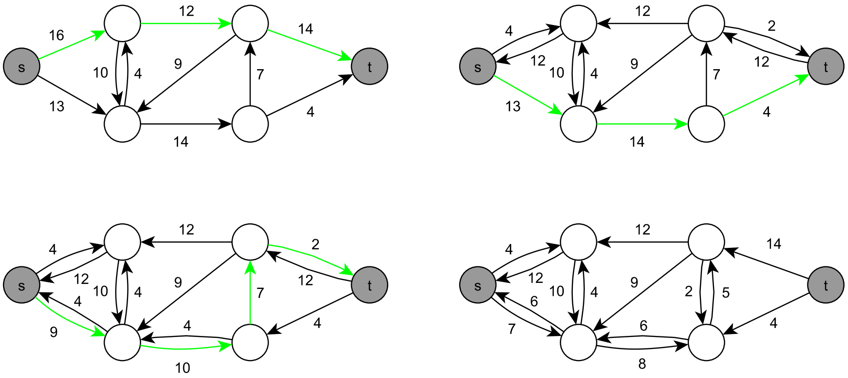
$\Rightarrow$ Laufzeit in $\mathcal{O}(|V| \cdot |E|^2)$

Edmonds-Karp Implementierung

Algorithm 1: Edmonds-Karp

```
Function Max-Flow (G = (V, E), s, t ∈ V, c : E → ℝ⁺)
    maxFlow = 0
    do
        find augmenting path P using BFS
        f = min{c(u, v)|(u, v) ∈ P}
        foreach (u, v) ∈ P do
            c(u,v) -= f
            c(v,u) += f
        end
        maxFlow += f
    while P exists
    return maxFlow
```

Edmonds-Karp Algorithmus



Edmonds-Karp Implementierungsdetails

- In Adjazenzliste neben Zielknoten auch Kapazität und Verweis auf die Rückkante speichern
- Nicht vorhandene Rückkanten mit 0 initialisieren und dem Graphen hinzufügen
- Bei der Breitensuche nur Kanten mit positiver Kapazität berücksichtigen
- Breitensuche abbrechen, sobald t erreicht wurde

Min-Cut

Min-Cut

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

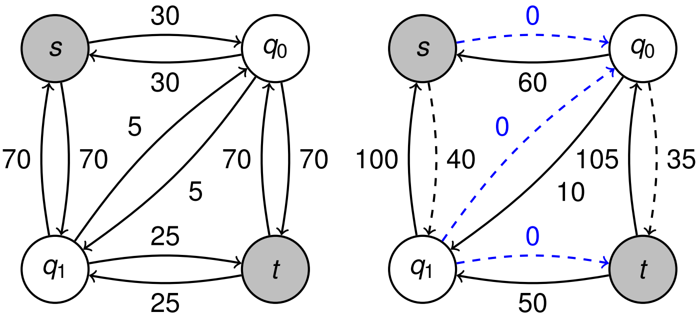
Max-Flow-Min-Cut-Theorem

Max-Flow-Min-Cut-Theorem

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

Max-Flow-Min-Cut

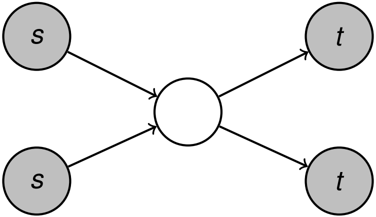
- Bsp.:



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

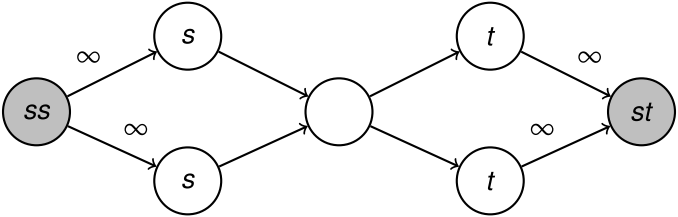
Multi-Quelle/Multi-Abfluss

- Gegeben sei folgende Situation:



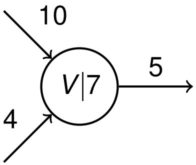
- Problem: Max-Flow Algorithmus kann nur mit einer Quelle und einer Senke arbeiten.
- Lösung: Ertelle Super-Quelle und Super-Senke und verbinde alle Quellen und Senken mit Kantengewicht ∞

Multi-Quelle/Multi-Abfluss Lösung



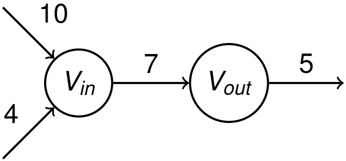
Knotenkapazität

- Gegeben sind Knoten mit Kapazität.
- Bsp.:



Knotenkapazität

- Gegeben sind Knoten mit Kapazität.
- Bsp.:



Modellierung

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

Modellierung

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

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

Modellierung - Beispiel

- 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
.	Treibeis
@	Großer Eisberg
#	Großes Holzbrett

Modellierung - Beispiel

- Gegeben sei nun folgende Eingabe:

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

- Wandle in Graphen um...

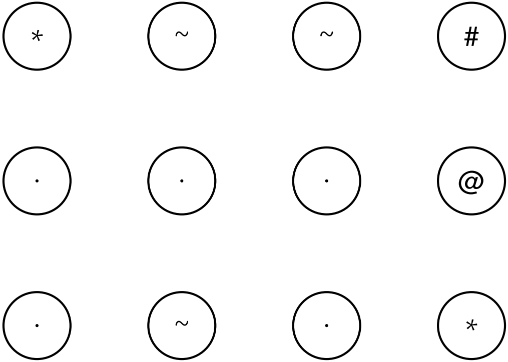
Modellierung - Beispiel

- Gegeben sei nun folgende Eingabe:

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

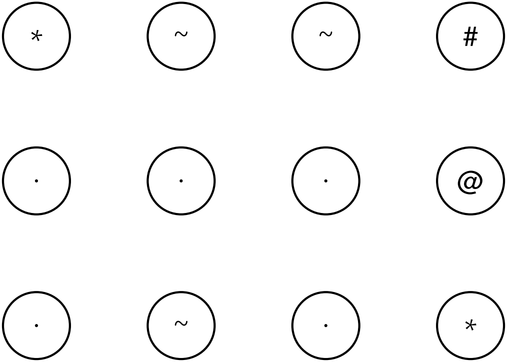
- Wandle in Graphen um...

Modellierung - Beispiel



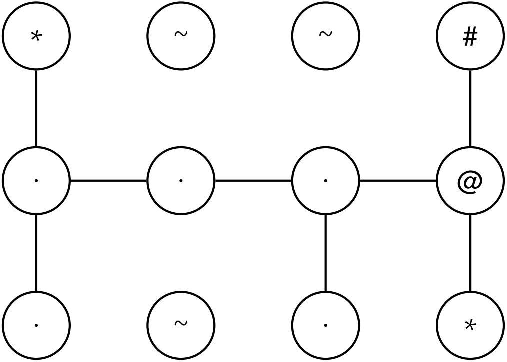
- Verbinde alle Knoten, über die ein Weg möglich ist...

Modellierung - Beispiel



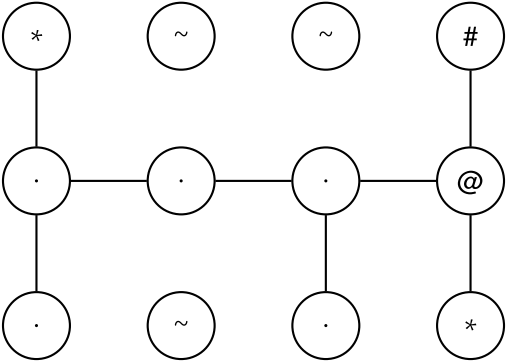
- Verbinde alle Knoten, über die ein Weg möglich ist...

Modellierung - Beispiel



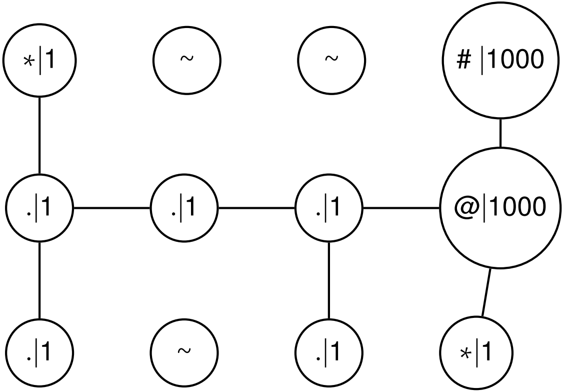
- Füge Knotengewichte hinzu...

Modellierung - Beispiel



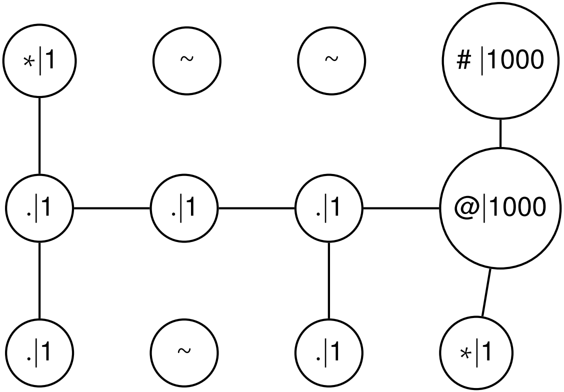
- Füge Knotengewichte hinzu...

Modellierung - Beispiel



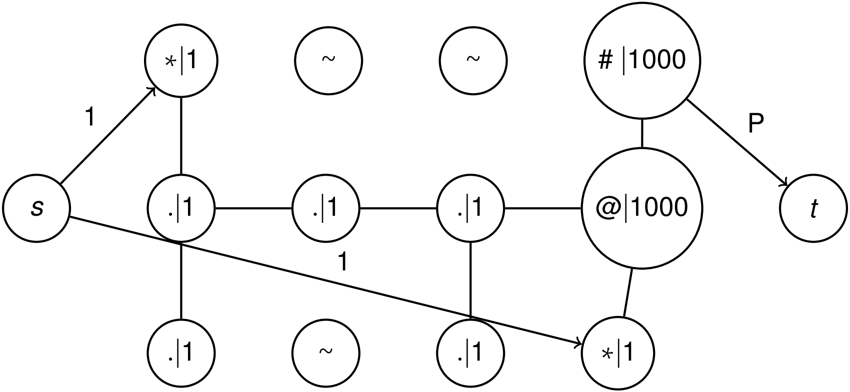
- Verbinde alle Menschen mit s und alle Holzbretter mit t...

Modellierung - Beispiel



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

Modellierung - Beispiel

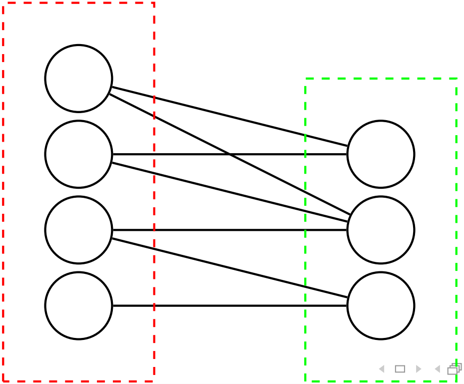


- Bem.: Knotengewichte müssen noch aufgelöst werden

Bipartiter Graph

Bipartiter Graph

- Ein Graph G = (V, E) heißt **bipartit**, wenn sich $V = A \cupdot B$ in 2 disjunkte Knotenmengen A und B aufteilen lässt, sodass zwischen den Knoten innerhalb der Teilmengen keine Kanten existieren.



Matching

Karlsruher Institut für Technologie

Matching

- Sei $G = (V, E)$ ein Graph. Ein **Matching** $M \subseteq E$ ist eine Menge paarweise knotendisjunkter Kanten, d.h.
  $\forall e_1 = \{u_1, v_1\}, e_2 = \{u_2, v_2\} \in M, e_1 \neq e_2 : e_1 \cap e_2 = \varnothing$
- Analog für gerichtete Graphen

Maximales Matching

- Ein Matching heißt **maximales Matching**, wenn nicht durch Hinzufügen einer Kante ein größeres Matching erstellt werden kann. (D.h. es gibt keine Kante $e = \{u, v\}$, wobei u und v nicht Teil des Matchings sind.)

Matching

Matching

- Sei $G = (V, E)$ ein Graph. Ein **Matching** $M \subseteq E$ ist eine Menge paarweise knotendisjunkter Kanten, d.h.
  $\forall e_1 = \{u_1, v_1\}, e_2 = \{u_2, v_2\} \in M, e_1 \neq e_2 : e_1 \cap e_2 = \varnothing$
- Analog für gerichtete Graphen

Maximales Matching

- Ein Matching heißt **maximales Matching**, wenn nicht durch Hinzufügen einer Kante ein größeres Matching erstellt werden kann. (D.h. es gibt keine Kante $e = \{u, v\}$, wobei u und v nicht Teil des Matchings sind.)

Perfektes Matching

Kardinalitätsmaximales Matching

- Ein Matching $M \subseteq E$ heißt **kardinalitätsmaximales Matching**, wenn es kein größeres Matching gibt. (D.h. $\forall$ Matchings M' : $|M| \geq |M'|$).

Perfektes Matching

- Ein Matching M heißt **perfekt**, falls $2 * |M| = |V|$, d.h. jeder Knoten $v \in V$ kommt in M vor.

Perfektes Matching

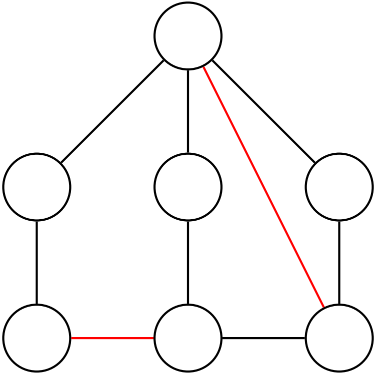
Kardinalitätsmaximales Matching

- Ein Matching $M \subseteq E$ heißt **kardinalitätsmaximales Matching**, wenn es kein größeres Matching gibt. (D.h. $\forall$ Matchings $M' : |M| \geq |M'|$).

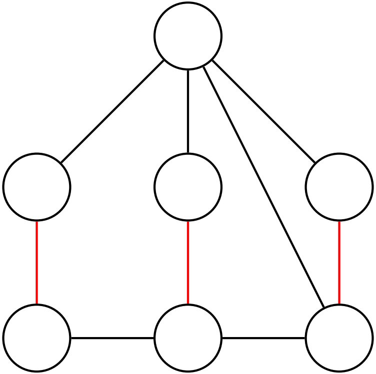
Perfektes Matching

- Ein Matching M heißt **perfekt**, falls $2 * |M| = |V|$, d.h. jeder Knoten $v \in V$ kommt in M vor.

Maximales Matching



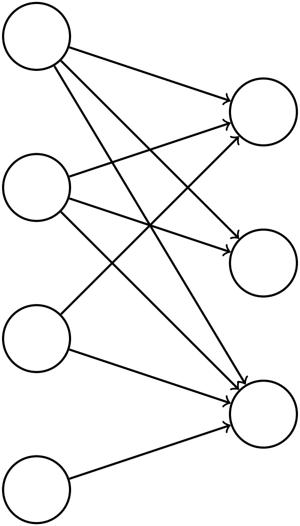
Perfektes Matching



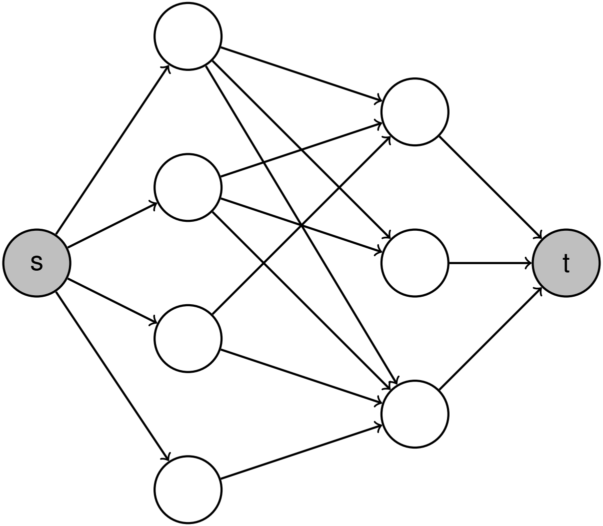
Modellierung

- Finden von kardinalitätsmaximalen Matchings in bipartiten Graphen $G = (V, E = A \uplus B)$:
  - Einfügen von neuen Knoten s und t
  - Einfügen von Kanten zwischen s und allen Knoten $v_A \in A$, und zwischen allen Knoten $v_B \in B$ und t.
  - Jede Kante im Graph (alte und neu eingefügte) hat Kapazität 1.
  - Berechnen des maximalen Flusses von s nach t.

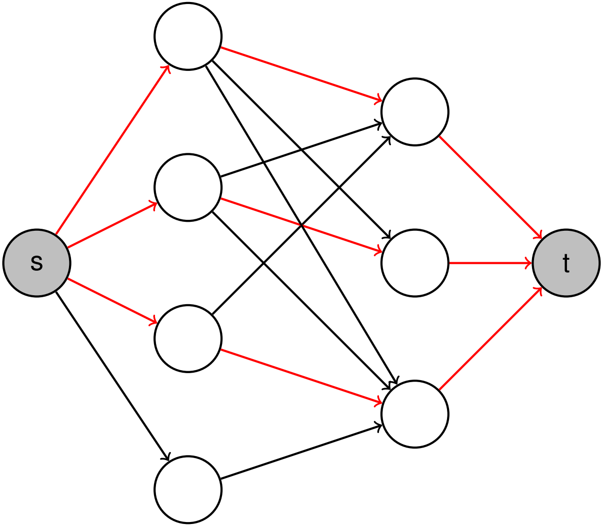
Modellierung



Modellierung

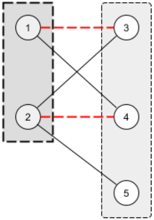


Modellierung



Beispiel

- 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—



Modelierung

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