Datenstrukturen, Algorithmen und Programmierung 2

Amin Coja-Oghlan

June 10, 2022

Lehrstuhl Informatik 2
Fakultät für Informatik

Netzwerkflüsse

Worum geht es?

- ein Netzwerk ist ein gerichteter Graph mit Kantenkapazitäten
- welche maximale Ladung kann von einer Quelle zu einer Senke transportiert werden?
- *Anwendung:* Matchings in bipartiten Graphen

Netzwerkflüsse

Definition

Ein (endlicher) gerichteter Graph G besteht aus

- einer endlichen Menge $V(G)$ von Knoten und
- einer Menge $E(G) \subseteq V(G) \times V(G)$ von gerichteten Kanten

Eine Kante der Form (v, v) nennt man Schleife

Netzwerkflüsse

Notation

- wir bezeichnen mit

$$\partial^+ v = \partial_G^+ v = \{w \in V : (w, v) \in E(G)\}, \quad \partial^- v = \partial_G^- v = \{w \in V : (v, w) \in E(G)\}$$

  die eingehende und die ausgehende Nachbarschaft von v
- ferner definiere

$$d_G^+(v) = |\partial_G^+ v|, \qquad d_G^-(v) = |\partial_G^- v|$$

  als den eingehenden und den ausgehenden Grad von v

Netzwerkflüsse

Breitensuche

- wir haben `BFS` für ungerichtete Graphen kennengelernt
- der Algorithmus überträgt sich auf gerichtete Graphen, indem in der Hauptschleife (Schritt 8) nur $u \in \partial^- v$ in die Warteschlange eingefügt werden
- der Algorithmus bestimmt dann kürzeste *gerichtete* Pfade in G

Netzwerkflüsse

Definition

Ein Netzwerk $N = (G, c, s, t)$ besteht aus

- einem gerichteten Graphen G
- einer Kapazitätsfunktion $c : V \times V \to \mathbb{R}_{\geq 0}$, so daß

$$c(v, w) = 0 \quad \text{falls } (v, w) \notin E(G)$$

- einer Quelle $s \in V(G)$
- einer Senke $t \in V(G) \setminus \{s\}$

Netzwerkflüsse

Definition

Ein Fluß in einem Netzwerk N ist eine Funktion $f : V \times V \to \mathbb{R}$, so daß

- $f(v, w) \leq c(v, w)$ für alle $v, w \in V(G)$
- $f(v, w) + f(w, v) = 0$ für alle $v, w \in V(G)$
- $\sum_{w \in V(G)} f(v, w) = 0$ für alle $v \in V(G) \setminus \{s, t\}$

Der Wert von f ist definiert als

$$|f| = \sum_{w \in V(G)} f(s, w)$$

Netzwerkflüsse

Notation

Für eine Funktion $f : V \times V \to \mathbb{R}$, $v \in V$ und $A, B \subset V$ definieren wir

$$f(v, A) = \sum_{w \in A} f(v, w)$$

$$f(A, v) = \sum_{w \in A} f(w, v)$$

$$f(A, B) = \sum_{v \in A} \sum_{w \in B} f(v, w)$$

Netzwerkflüsse

Lemma

Sei N ein Netzwerk und f ein Fluß. Für alle $A, B, W \subseteq V$ gilt

$$f(A, A) = 0$$

$$f(A, B) + f(B, A) = 0$$

$$f(A \cup B, C) = f(A, C) + f(B, C) \quad \text{sofern } A \cap B = \emptyset$$

$$f(C, A \cup B) = f(C, A) + f(C, B) \quad \text{sofern } A \cap B = \emptyset$$

Netzwerkflüsse

Das Max Flow-Problem

- gegeben ist ein Netzwerk N
- das Ziel ist, einen Fluß mit maximalem Wert (einen “maximalen Fluß”) zu bestimmen
- die Idee ist, ausgehend vom Nullfluß den aktuellen Fluß immer weiter zu “augmentieren”

Netzwerkflüsse

Restflüsse

Sei N ein Netzwerk und f ein Fluß

- die Restkapazität von f in N ist definiert als

$$c_f(v,w) = c(v,w) - f(v,w) \qquad (v,w \in V(G))$$

- das Restnetzwerk von f in N ist das Netzwerk mit Kantenkapazitätsfunktion c_f und Kantenmenge

$$E_f = \{(v,w) \in V(G) : c_f(v,w) > 0\}$$

Netzwerkflüsse

Lemma

Angenommen N ist ein Netzwerk, f ist ein Fluß in N und g ist ein Fluß in N_f. Dann ist $f + g$ ein Fluß in N mit Wert $|f + g| = |f| + |g|$.

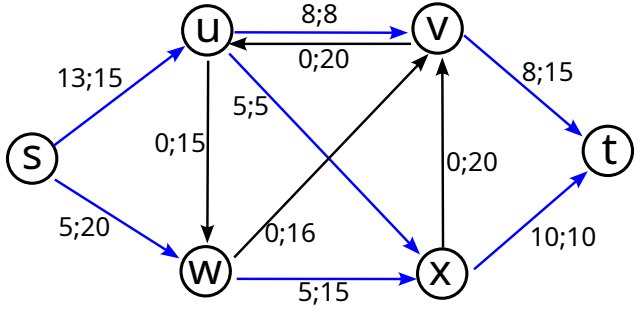
Netzwerkflüsse

Augmentierende Pfade

- ein f-augmentierender Pfad in N ist ein s-t-Pfad in N_f
- die Kapazität eines solchen Pfades p ist

$$c_f(p) = \min\{c_f(u,v) : (u,v) \text{ ist eine Kante auf } p\} > 0$$

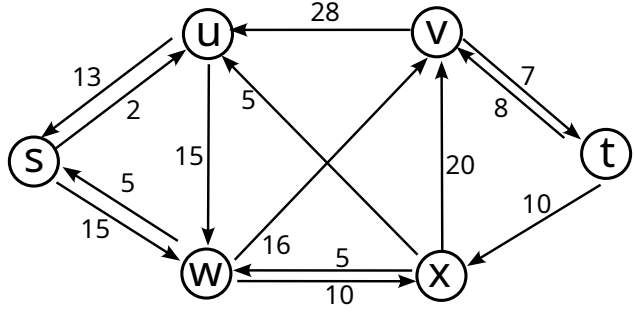
Netzwerkflüsse



Beispiel

- ein Netzwerkfluß f

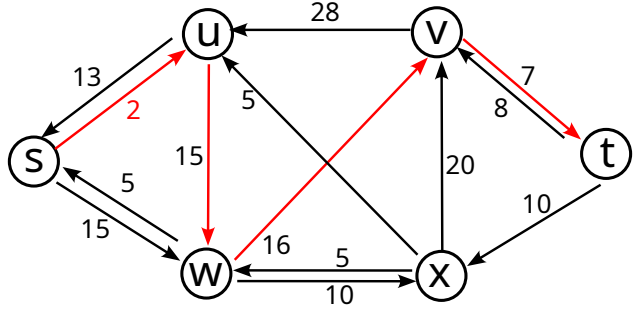
Netzwerkflüsse



Beispiel

- das Restnetzwerk N_f

Netzwerkflüsse



Beispiel

- ein augmentierender Pfad der Kapazität zwei

Netzwerkflüsse

Lemma

Angenommen N ist ein Netzwerk, f ein Fluß und p ein augmentierender Pfad. Dann ist $f_p : V \times V \to \mathbb{R}$ mit

$$f_p(v,w) = \left(\mathbb{1}\{(v,w) \text{ ist Kante von } p\} - \mathbb{1}\{(w,v) \text{ ist Kante von } p\}\right) c_f(p)$$

ein Fluß mit Wert $c_f(p)$ in N_f

Netzwerkflüsse

Korollar

Angenommen N ist ein Netzwerk, f ein Fluß und p ein augmentierender Pfad. Dann ist $f + f_p$ ein Fluß in N mit Wert $|f| + c_f(p) > |f|$.

Netzwerkflüsse

Algorithmus `FordFulkerson`

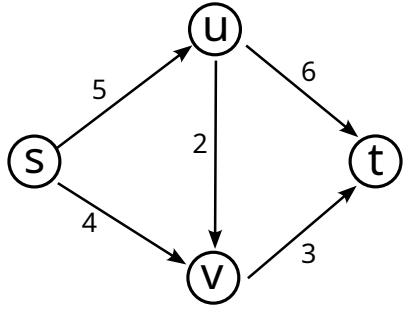
1. setze $f(v, w) = 0$ für alle $v, w \in V$
2. solange es einen augmentierenden Pfad p in N_f gibt
3. setze $f = f + f_p$
4. gib f aus

Netzwerkflüsse

Anmerkungen

- der Algorithmus spezifiziert nicht, wie/welcher Pfad p gefunden wird
- es ist nicht klar, daß der Algorithmus hält!
- sofern alle Kapazitäten ganz sind, ist das aber der Fall

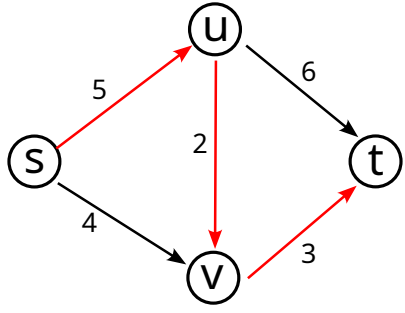
Netzwerkflüsse



Beispiel

- das ursprüngliche Netzwerk

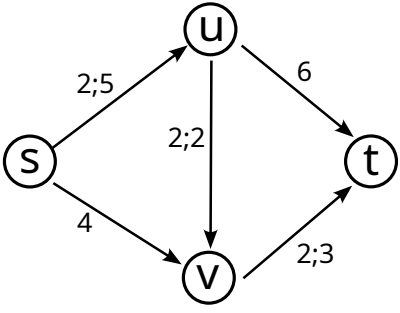
Netzwerkflüsse

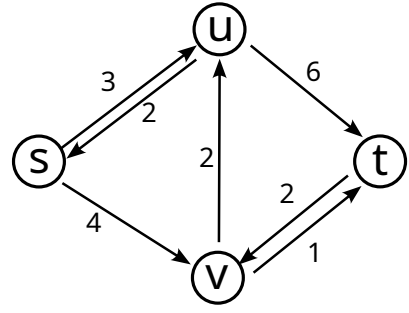


Beispiel

- ein augmentierender Pfad der Kapazität zwei

Netzwerkflüsse

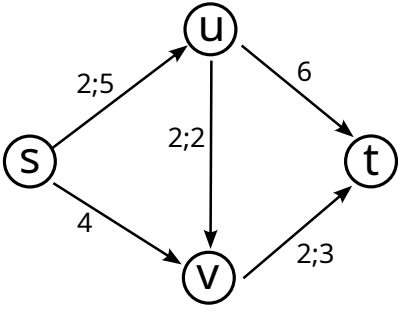


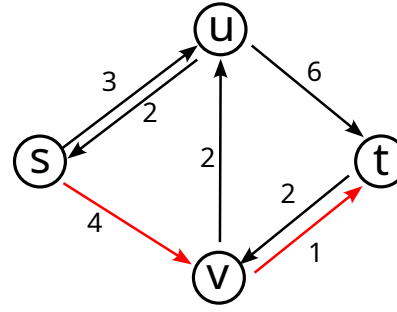


Beispiel

- links: der aktuelle Fluß im Ursprungsnetzwerk
- rechts: das Restnetzwerk

Netzwerkflüsse

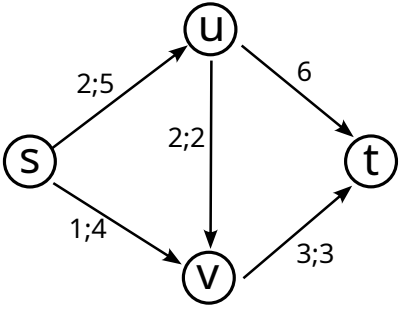


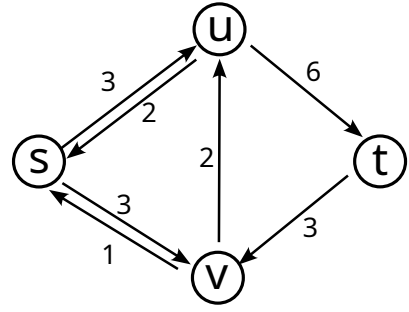


Beispiel

- links: der aktuelle Fluß im Ursprungsnetzwerk
- rechts: ein augmentierender Pfad mit Kapazität eins

Netzwerkflüsse

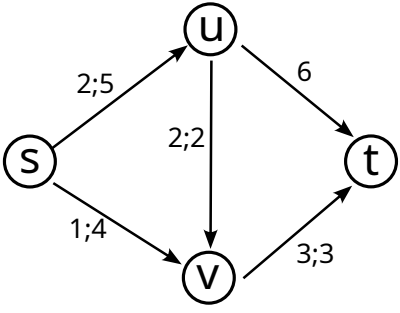


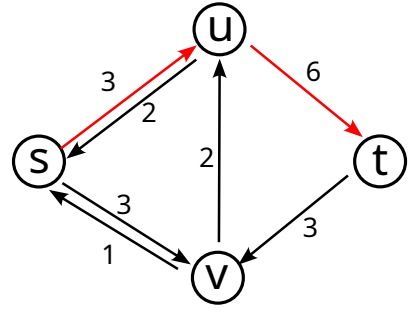


Beispiel

- links: der aktuelle Fluß im Ursprungsnetzwerk
- rechts: das Restnetzwerk

Netzwerkflüsse

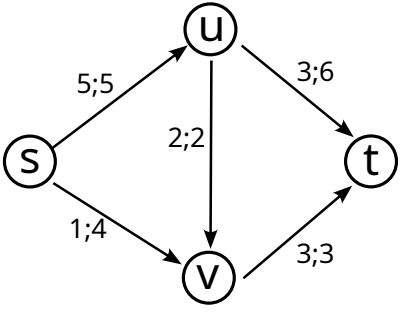


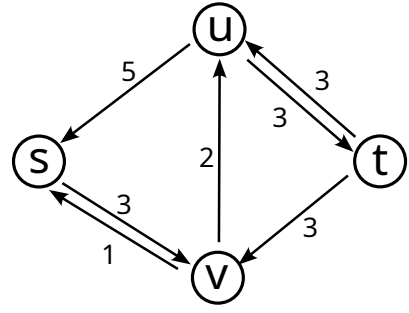


Beispiel

- links: der aktuelle Fluß im Ursprungsnetzwerk
- rechts: ein augmentierender Pfad mit Kapazität drei

Netzwerkflüsse

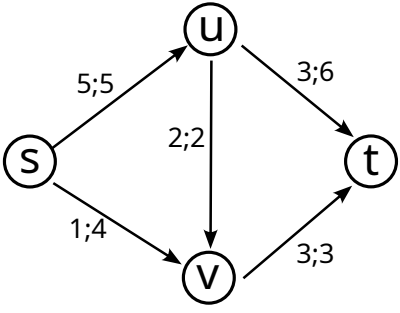


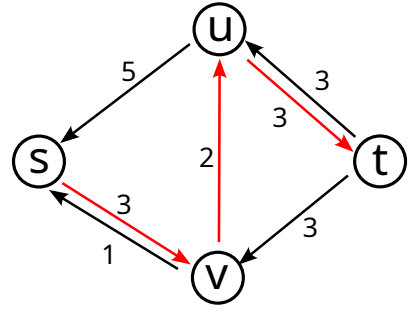


Beispiel

- links: der aktuelle Fluß im Ursprungsnetzwerk
- rechts: das Restnetzwerk

Netzwerkflüsse

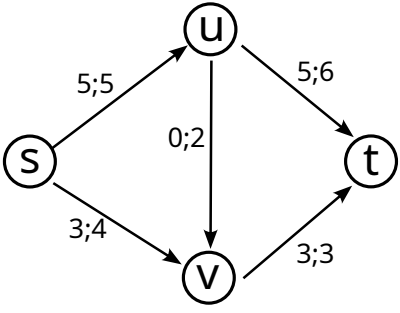


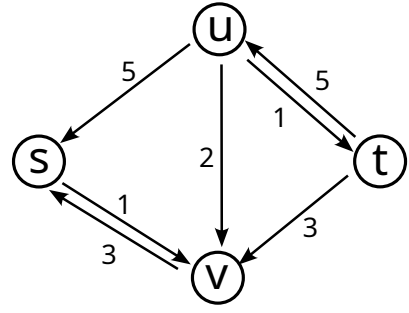


Beispiel

- links: der aktuelle Fluß im Ursprungsnetzwerk
- rechts: ein augmentierender Pfad mit Kapazität zwei

Netzwerkflüsse





Beispiel

- links: der aktuelle Fluß im Ursprungsnetzwerk
- rechts: das Restnetzwerk; t ist von s nicht mehr erreichbar

Netzwerkflüsse

Schnitte

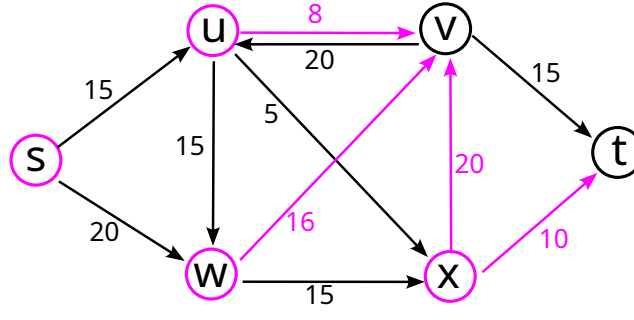
- angenommen N ist ein Netzwerk
- ein Schnitt in N ist eine Menge $S \subseteq V(G)$ mit

$$s \in S \qquad \text{und} \qquad t \notin S$$

- die Kapazität eines Schnittes S ist

$$c(S) = \sum_{(v,w) \in S \times (V \setminus S)} \mathbb{1}\{(v,w) \in E(G)\} c(v,w)$$

Netzwerkflüsse



Beispiel

- der Schnitt $S = \{s, u, w, x\}$ hat Kapazität 54

Netzwerkflüsse

Lemma

Angenommen N ist ein Netzwerk und f ist ein Fluß Für jeden Schnitt S gilt

$$|f| = f(S, V \setminus S)$$

Beweis

$$\begin{aligned}
|f| &= f(s, V) \\
&= f(s, V) + f(S \setminus \{s\}, V) && \text{[Flußerhaltung]} \\
&= f(S, V) = f(S, V \setminus S) + f(S, S) \\
&= f(S, V \setminus S) && \text{[weil } f(S, S) = 0\text{]}
\end{aligned}$$

Netzwerkflüsse

Korollar

Wenn N ein Netzwerk, f ein Fluß und S ein Schnitt ist, gilt $|f| \leq c(S)$

Beweis

Das Lemma zeigt $|f| = f(S, V \setminus S) \leq c(S)$.

Netzwerkflüsse

Theorem **["Max flow min cut theorem"]**

Für jedes Netzwerk N gilt

$$\max\{|f| : f \text{ ist ein Fluß in } N\} = \min\{c(S) : S \text{ ist ein Schnitt in } N\}$$

Beweis

- wir suchen einen Fluß f und einen Schnitt S mit $|f| = c(S)$
- sei f ist ein maximaler Fluß, d.h. $|f| = \max\{|\varphi| : \varphi \text{ ist ein Fluß in } N\}$
- angenommen es gäbe einen augmentierenden Pfad p
- dann wäre $|f + f_p| > |f|$; Widerspruch!

Netzwerkflüsse

Beweis

- also gibt es keinen Pfad von s nach t in N_f
- betrachte daher die Menge

$$S = \{v \in V(G) : \text{es gibt einen Pfad von } s \text{ nach } v \text{ in } N_f\}$$

- dann ist S ein Schnitt
- ferner gibt es keine Kante $(v, w) \in S \times (V \setminus S)$ in N_f
- folglich $f(v, w) = c(v, w)$ für alle $(v, w) \in S \times (V \setminus S)$ mit $(v, w) \in E(G)$
- also gilt $c(S) = |f|$

Netzwerkflüsse

Korollar

Wenn `FordFulkerson` terminiert, ist die Ausgabe ein maximaler Fluß

Beweis

Wenn es keinen augmentierenden Pfad mehr gibt, ist f ein maximaler Fluß

Anmerkung

- `FordFulkerson` terminiert, wenn die Kapazitäten ganzzahlig sind
- allerdings kann dies $\max\{|f| : f \text{ Fluß in } N\}$ Schritte erfordern
- `FordFulkerson` ist also nicht *effizient!*

Netzwerkflüsse

Algorithmus `EdmondsKarp`

- verwende in `FordFulkerson` Breitensuche, um den augmentierenden Pfad zu finden
- mit anderen Worte: finde jeweils einen *kürzesten* augmentierenden Pfad

Netzwerkflüsse

Monotonielemma

Seien $f_1, f_2, \ldots$ die Flüsse, die `EdmondsKarp` konstruiert. Sei ferner $\text{dist}_{N_{f_i}}(s, v)$ der Abstand von s, v im Restnetzwerk N_{f_i}. Dann gilt

$$\text{dist}_{N_{f_i}}(s, v) \leq \text{dist}_{N_{f_j}}(s, v) \qquad \text{für alle } i \leq j,\ v \in V(G).$$

Beweis

- angenommen nicht
- wähle i minimal mit $\text{dist}_{N_{f_i}}(s, v) > \text{dist}_{N_{f_{i+1}}}(s, v)$
- wähle außerdem v so, daß $\text{dist}_{N_{f_{i+1}}}(s, v)$ minimal ist
- sei ferner p ein kürzester Pfad von s nach v in $N_{f_{i+1}}$

Netzwerkflüsse

Beweis

- sei u der letzte Knoten vor v in p
- dann gilt $(u, v) \in N_{f_{i+1}}$ und $\text{dist}_{N_{f_{i+1}}}(s, u) + 1 = \text{dist}_{N_{f_{i+1}}}(s, v)$
- ferner gilt $\text{dist}_{N_{f_{i+1}}}(s, u) \geq \text{dist}_{N_{f_i}}(s, u)$ nach Wahl von v
- angenommen $(u, v) \in N_{f_i}$; dann gilt

$$\text{dist}_{N_{f_i}}(s, v) \leq \text{dist}_{N_{f_i}}(s, u) + 1 \leq \text{dist}_{N_{f_{i+1}}}(s, u) + 1 = \text{dist}_{N_{f_{i+1}}}(s, v),$$

im Widerspruch zur Wahl von v

Netzwerkflüsse

Beweis

- weil $(u, v) \in N_{f_{i+1}}$ aber $(u, v) \notin N_{f_i}$, muß beim Augmentieren der Fluß von v nach u erhöht worden sein
- weil `EdmondsKarp` entlang kürzester Pfade augmentiert, enthält der kürzeste Pfad von s nach u in N_{f_i} daher die Kante (v, u)
- daraus folgt aber

$$\text{dist}_{N_{f_i}}(s, v) = \text{dist}_{N_{f_i}}(s, u) - 1 \leq \text{dist}_{N_{f_{i+1}}}(s, u) - 1 = \text{dist}_{N_{f_{i+1}}}(s, v) - 2,$$

  im Widerspruch zur Wahl von v

Netzwerkflüsse

Satz

`EdmondsKarp` hat Laufzeit $O(|V(G)| \cdot |E(G)|^2)$

Beweis

- wir zeigen, daß `EdmondsKarp` nach höchstens $O(|V(G)| \cdot |E(G)|)$ Augmentierungen terminiert
- weil Breitensuche jeweils Zeit $O(|E(G)|)$ benötigt, folgt daraus der Satz
- eine Kante (v, w) auf einem augmentierenden Pfad p ist kritisch für einen Fluß f, falls $c_f(v, w) = c_f(p)$
- dann ist (v, w) nicht in N_{f+f_p} enthalten
- wir zeigen, daß keine Kante mehr als $|V(G)|/2$ mal kritisch werden kann

Netzwerkflüsse

Beweis

- angenommen (v, w) wird kritisch für eine Fluß f für den Pfad p, den `EdmondsKarp` auswählt
- weil p ein kürzester Pfad ist, gilt für die Abstände in dem Netzwerk N_f:

$$\text{dist}_{N_f}(s, w) > \text{dist}_{N_f}(s, v)$$

- angenommen (v, w) wird später noch einmal kritisch
- dann muß die Kante (v, w) zunächst wieder in das Restnetzwerk eingefügt worden sein
- dies ist nur möglich, wenn die Kante (w, v) auf dem augmentierenden Pfad für einen Fluß f' gelegen hat
- für den Fluß f' gilt dann $\text{dist}_{N_{f'}}(s, v) > \text{dist}_{N_{f'}}(s, w)$

Netzwerkflüsse

Beweis

- das Monotonielemma zeigt also

$$\text{dist}_{N_{f'}}(s, v) \geq \text{dist}_{N_f}(s, v) + 2$$

- weil $\text{dist}_{N_{f'}}(s, v) \leq |V(G)|$, kann (v, w) also nur $|V(G)|/2$ mal kritisch werden

Netzwerkflüsse

Matchings

- ein Matching in einem ungerichteten Graphen $G = (V, E)$ ist eine Menge $M \subseteq E$ von Kanten, so daß

$$e \cap f = \emptyset \qquad \text{für alle } e, f \in M,\ e \neq f$$

- mit $\nu(G)$ wird die maximale Größe eines Matchings in G bezeichnet ("Matchingzahl von G")
- G heißt bipartit, falls $\chi(G) = 2$
- eine Bipartition von G ist ein Paar (S, T) von stabilen Mengen, so daß $S \cup T = V(G)$ und $S \cap T = \emptyset$

Netzwerkflüsse

Satz von Hall

Sei G ein bipartiter Graph mit Bipartition (S, T). Es gilt $\nu(G) = |S|$ genau dann, wenn

$$|\partial U| \geq |U| \qquad \text{für alle } U \subseteq S.$$

Erinnerung: $\partial U = \{v \in V(G) : \exists u \in U : v \in \partial u\}$

Netzwerkflüsse

Beweis

- wenn $\nu(G) = |S|$, dann gilt $|\partial U| \geq |U|$ für alle $U \subseteq S$
- nehme nun umgekehrt an, daß $|\partial U| \geq |U|$ für alle $U \subseteq S$
- konstruiere ein Netzwerk $N = (\Gamma, c, s, t)$, so daß

$$V(\Gamma) = V(G) \cup \{s, t\} \qquad \text{mit} \qquad s, t \notin V(G)$$
$$E(\Gamma) = \{(v, w) \in S \times T : vw \in E(G)\} \cup \{s\} \times S \cup T \times \{t\}$$

- ferner ist $c(v, w) = 1$ für alle $(v, w) \in E(\Gamma)$

Netzwerkflüsse

Beweis

- sei nun $f : V(\Gamma) \times V(\Gamma) \to \{0, 1\}$ ein maximaler Fluß
- dann ist

$$M = \{vw \in E(G) : f(v, w) = 1\}$$

  ein Matching von G
- zu zeigen ist also, daß $|f| = |S|$
- nach dem max flow min cut theorem genügt es zu zeigen, daß für jeden Schnitt Y von N gilt

$$c(Y) \geq |S|$$

Netzwerkflüsse

Beweis

- definiere daher $A = Y \cap S$ und $B = Y \cap T$
- nach Konstruktion von N gilt dann

$$c(Y) \geq |S \setminus A| + |\partial A|$$

- weil ferner $|\partial A| \geq |A|$, folgt $c(Y) \geq |S|$

Netzwerkflüsse

Matchings via Flüsse

- mit der Konstruktion eines Netzwerkes aus dem Beweis erhalten wir ein effizientes Verfahren, um die Matchingzahl bipartiter Graphen zu berechnen
- wir wenden auf dieses Netzwerk einfach `FordFulkerson` an
- da alle Kapazitäten $\{0, 1\}$ sind, ist der optimale Fluß auch $\{0, 1\}$-wertig
- *es gibt effizientere Algorithmen für dieses Problem* $\rightsquigarrow$ *VL Effiziente Algorithmen*

Netzwerkflüsse

Zusammenfassung

- Flüsse in Netzwerken
- `FordFulkerson` und `EdmondsKarp`-Algorithmen
- max flow min cut
- Matchings in bipartiten Graphen