Informatik II

Übung 13

FS 2019

Heutiges Programm

1. Feedback letzte Übung
2. Wiederholung Vorlesung
3. In-Class-Exercise (praktisch)

1. Feedback letzte Übung

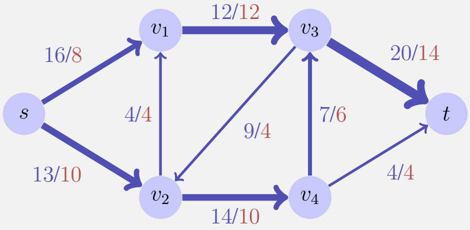
2. Wiederholung Vorlesung

Fluss

Ein *Fluss* $f : V \times V \to \mathbb{R}$ erfüllt folgende Bedingungen:

- *Kapazitätsbeschränkung*:
  Für alle $u, v \in V$: $f(u,v) \leq c(u,v)$.
- *Schiefsymmetrie*:
  Für alle $u, v \in V$: $f(u,v) = -f(v,u)$.
- *Flusserhaltung*:
  Für alle $u \in V \setminus \{s,t\}$:

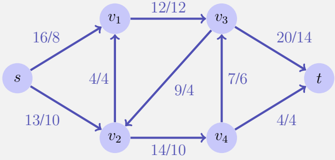
$$\sum_{v \in V} f(u,v) = 0.$$

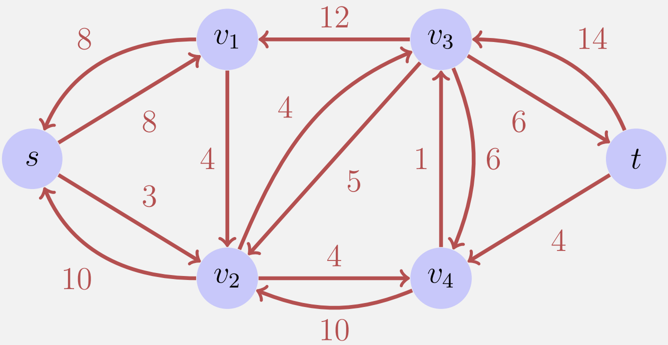


Wert w des Flusses:
$|f| = \sum_{v \in V} f(s,v)$.
Hier $|f| = 18$.

Restnetzwerk

Restnetzwerk G_f gegeben durch alle Kanten mit Restkapazität:





Restnetzwerke haben dieselben Eigenschaften wie Flussnetzwerke, ausser dass antiparallele Kapazitäten-Kanten zugelassen sind.

Erweiterungspfade

Erweiterungspfad p: einfacher Pfad von s nach t im Restnetzwerk G_f.

Restkapazität $c_f(p) = \min\{c_f(u,v) : (u,v) \text{ Kante in } p\}$

Max-Flow Min-Cut Theorem

Theorem

Wenn f ein Fluss in einem Flussnetzwerk $G = (V, E, c)$ mit Quelle s und Senke t is, dann sind folgende Aussagen äquivalent:

1. *f ist ein maximaler Fluss in G*
2. *Das Restnetzwerk G_f enthält keine Erweiterungspfade*
3. *Es gilt $|f| = c(S, T)$ für einen Schnitt (S, T) von G.*

Algorithmus Ford-Fulkerson(G, s, t)

Input: Flussnetzwerk $G = (V, E, c)$
Output: Maximaler Fluss f.

for $(u, v) \in E$ **do**
  $f(u, v) \leftarrow 0$

while Existiert Pfad $p : s \rightsquigarrow t$ im Restnetzwerk G_f **do**
  $c_f(p) \leftarrow \min\{c_f(u, v) : (u, v) \in p\}$
  foreach $(u, v) \in p$ **do**
    $f(u, v) \leftarrow f(u, v) + c_f(p)$
    $f(v, u) \leftarrow f(v, u) - c_f(p)$

Praktische Anmerkung

In einer Implementation des Ford-Fulkerson Algorithmus werden die negativen Flusskanten normalerweise nicht gespeichert, da ihr Wert sich stets als der negierter Wert der Gegenkante ergibt.

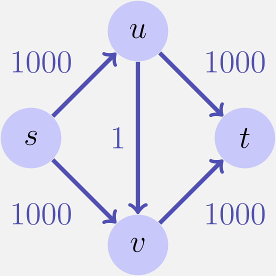
$$f(u, v) \leftarrow f(u, v) + c_f(p)$$
$$f(v, u) \leftarrow f(v, u) - c_f(p)$$

wird dann zu

if $(u, v) \in E$ **then**
  $f(u, v) \leftarrow f(u, v) + c_f(p)$
else
  $f(v, u) \leftarrow f(v, u) - c_f(p)$

Analyse

- Für ganzzahligen Fluss benötigt der Algorithmus maximal $|f_{\max}|$ Durchläufe der While-Schleife (denn der Fluss erhöht sich mindestens um 1). Suche einzelner zunehmender Weg (z.B. Tiefensuche oder Breitensuche) $\mathcal{O}(|E|)$. Also Gesamtlaufzeit in $\mathcal{O}(f_{\max}|E|)$.



Bei schlecht gewählter Strategie benötigt der Algorithmus hier bis zu 2000 Iterationen.

Edmonds-Karp Algorithmus

Wähle in der Ford-Fulkerson-Methode zum Finden eines Pfades in G_f jeweils einen Erweiterungspfad kürzester Länge (z.B. durch Breitensuche).

Edmonds-Karp Algorithmus

Theorem

Wenn der Edmonds-Karp Algorithmus auf ein ganzzahliges Flussnetzwerk $G = (V, E)$ mit Quelle s und Senke t angewendet wird, dann ist die Gesamtanzahl der durch den Algorithmus angewendete Flusserhöhungen in $\mathcal{O}(|V| \cdot |E|)$.

$\Rightarrow$ *Gesamte asymptotische Laufzeit:* $\mathcal{O}(|V| \cdot |E|^2)$

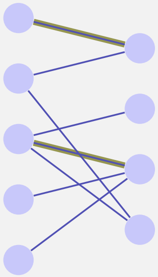
[Ohne Beweis]

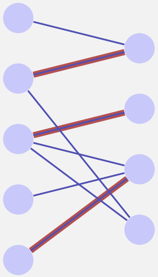
Anwendung: Maximales bipartites Matching

Gegeben: bipartiter ungerichteter Graph $G = (V, E)$.

Matching M: $M \subseteq E$ so dass $|\{m \in M : v \in m\}| \leq 1$ für alle $v \in V$.

Maximales Matching M: Matching M, so dass $|M| \geq |M'|$ für jedes Matching M'.

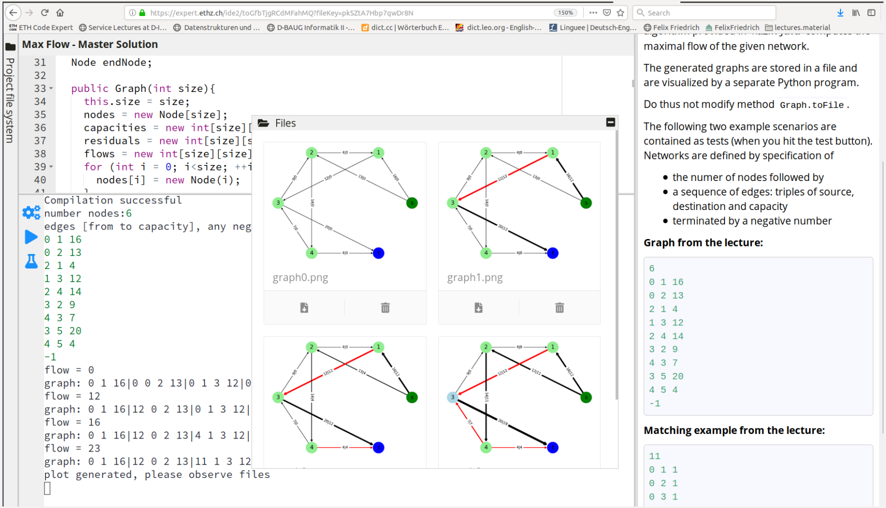




3. In-Class-Exercise (praktisch)

Maximale Flüsse – Implementation

Max-Flow Implementation



Fragen?