3. Dinicův algoritmus

(zapsali Jakub Melka, Petr Musil)[1]

Na minulé přednášce jsme si ukázali *Fordův-Fulkersonův* algoritmus. Víme o něm, že když se zastaví, tak vydá maximální tok. Jenže zastavit se nemusí (například pro sítě s reálnými kapacitami), nebo trvá příliš dlouho. Ukážeme si lepší algoritmus, *Dinicův*, který má výrazně menší složitost a zastaví se vždy.

Idea je následující: v algoritmu budeme používat *síť rezerv*, která bude obsahovat rezervy – kolik ještě po dané hraně můžeme pustit, aby to nepřekročilo její kapacitu. Síť rezerv pak budeme využívat k vylepšování toku.

Definice: *Síť rezerv* R k síti $S = (V, E, z, s, c)$ a toku f v S je síť $R = (V, E \cup \overleftarrow{E}, z, s, r)$, pro $\forall e \in E$:

- $r(e) = c(e) - f(e)$,
- $r(\overleftarrow{e}) = f(e)$,

kde hrana $\overleftarrow{e}$ vznikne z hrany e tak, že se zorientuje opačným směrem. V případě, že v síti rezerv už opačně orientovaná hrana[2] je, pak vznikne multigraf[3] s multiplicitou maximálně dva.

Síť rezerv budeme používat v algoritmu k hledání vylepšujících toků. K tomu nám bude sloužit následující věta:

Věta: Je-li f tok v síti S a g tok v příslušné síti rezerv, pak $\exists$ tok f' v S takový, že $|f'| = |f| + |g|$, což znamená $\forall e \in E : f'(e) = f(e) + g(e) - g(\overleftarrow{e})$.

Důkaz: Rozebereme si jednotlivé případy pro $\forall e \in E$:

a) $g(e) = g(\overleftarrow{e}) = 0 \Rightarrow f'(e) = f(e)$.
b) $g(e) > 0$ a zároveň $g(\overleftarrow{e}) = 0 \Rightarrow f'(e) = f(e) + g(e)$.
c) $g(e) = 0$ a zároveň $g(\overleftarrow{e}) > 0 \Rightarrow f'(e) = f(e) - g(\overleftarrow{e})$.
d) Nastává cirkulace, tu snadno odstraníme: odečteme ε od obou hran e a $\overleftarrow{e}$, kde $\varepsilon = \min(g(e), g(\overleftarrow{e}))$. Převedeme tím tento případ na jeden ze tří uvedených výše.

Je však f' tok? Víme, že f' určitě nemůže klesnout pod nulu, protože se odečítá jen v případě c), a tam je z definice vidět, že f' pod nulu klesnout nemůže. Kapacita také nemůže být překročena, přičítá se jen v případě b) a z definice se nepokazí, protože $g(e) = c(e) - f(e)$, tedy v nejhorším případě $f'(e) = c(e)$.

Dále dokážeme, že f' dodržuje Kirchhoffův zákon. V následujících sumách předpokládejme, že všechny vrcholy jsou rozdílné od zdroje a stoku. Musí platit, že:

$$\sum_{ab \in E} f'(ab) = \sum_{ba \in E} f'(ba).$$

[1] s díky Bernardovi Lidickému za obrázky
[2] vznikne tak, že v původní síti jsou dvě opačně orientované hrany mezi stejnými vrcholy.
[3] graf, který může mít mezi dvěma vrcholy více stejně orientovaných hran.

Rozepíšeme si tuto rovnici dle definice:

$$\sum_{ab \in E} f'(ab) - \sum_{ba \in E} f'(ba) = 0,$$

$$\sum_{uv \in E} (f(uv) + g(uv) - g(\overleftarrow{uv})) - \sum_{vu \in E} (f(vu) + g(vu) - g(\overleftarrow{vu})) = 0.$$

Roztrhneme si to na čtyři sumy a dostaneme:

$$\underbrace{\sum_{uv \in E} f(uv) - \sum_{vu \in E} f(vu)}_{0} +$$

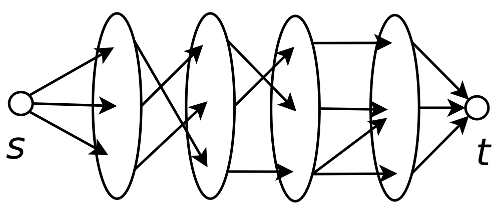
$$+ \underbrace{\sum_{uv \in E} g(uv) - g(\overleftarrow{uv}) - \sum_{vu \in E} g(vu) - g(\overleftarrow{vu})}_{0} = 0,$$

neboť f i g jsou toky a musí splňovat Kirchhoffův zákon. ♡

Tato věta nám říká, že pokud existuje nenulový tok v síti rezerv, pak lze tok v původní síti ještě zvětšit. Naopak pokud takový tok neexistuje, je tok v původní síti maximální.

Definice: f je *blokující tok*, pokud na každé orientované cestě P ze zdroje do spotřebiče $\exists e \in P : f(e) = c(e)$.

Definice: C je *pročištěná síť*, pokud obsahuje pouze vrcholy a hrany na nejkratších $z \to s$ cestách. Pročištěná síť nemá slepé uličky, ani hrany vedoucí ze stoku někam do dalšího vrcholu.



Příklad pročištěné sítě

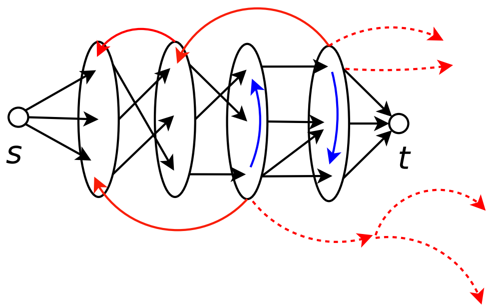
Algoritmus (hledání maximálního toku v síti, Dinicův)

1. $f \leftarrow$ nulový tok.
2. Sestrojíme síť rezerv R, vynecháme hrany s nulovou rezervou.
3. $l \leftarrow$ délka nejkratší cesty $z \to s$ v R.
4. Když $l = \infty$, tak skončíme.
5. Sestrojíme pročištěnou síť C, a to následujícím způsobem:

6. Spustíme BFS[4] algoritmus ze zdroje.
7. BFS nám rozdělí uzly do vrstev, vyhodíme hrany za spotřebičem a slepé uličky.
8. $g \leftarrow$ blokující tok v C.
9. Zlepšíme tok f podle g a jdeme na bod 3.

Postup tvorby pročištěné sítě podrobněji: Prohledáním do šířky vytvoříme vrstvy C_i, zahodíme ty za spotřebičem, ponecháme pouze hrany mezi C_i a C_{i+1}. Ještě musíme odstranit slepé uličky – cesty, které končí v $C_m : m < l$, protože ty určitě nejsou součástí nejkratší $z \to s$ cesty.

Pročištění zvládneme v lineárním čase $\mathcal{O}(N + M)$, v případě souvislého grafu pouze $\mathcal{O}(M)$.



Příklad nepročištěné sítě

Na obrázku nepročištěné sítě vidíme, co se má smazat. Černé hrany ponecháme, ty tam jsou správně. Červené hrany jsou zpětné, ty smažeme. Modré hrany jsou hrany ve stejné vrstvě, ty rovněž smažeme. Červené tečkované hrany nevedou vůbec do stoku, takže ty rovněž smažeme.

Definice: *Fází* algoritmu označíme jeden běh cyklu – kroky 3 až 9.

Provedeme podrobnou analýzu algoritmu z hlediska složitosti a uvidíme, že má složitost $\mathcal{O}(N^2 M)$. Nejprve analyzujeme hledání blokujícího toku, pak se podíváme, kolik fází maximálně může Dinicův algoritmus mít.

Algoritmus hledání blokujícího toku

1. $g \leftarrow$ nulový tok.
2. Dokud $\exists z \to s$ cesta P v pročištěné síti C:
3. $\varepsilon \leftarrow \min_{e \in P}(c(e) - f(e))$.
4. $\forall e \in P : g(e) \leftarrow g(e) + \varepsilon$, pokud $g(e)$ vzroste na $r(e)$, tak smažeme hranu e.
5. Dočistíme síť tím, že odstraníme slepé uličky, které mohly vzniknout smazáním hrany e.

[4] Breadth-First Search, standardní prohledávání do šířky.

Při každém průchodu se smaže vždy alespoň jedna hrana, tedy maximálně M-krát provádíme $\mathcal{O}(N)$ – právě tolik trvá nalezení cesty P, protože délka cesty bude kratší nebo rovna N. Čištění pak maximálně smaže celý graf, jedno mazání nás stojí konstantní čas, tedy celková složitost tohoto algoritmu bude $\mathcal{O}(MN)$.

Dokážeme si, že počet fází je menší nebo roven N. Algoritmus se ukončí, pokud $l > N$, protože pak už neexistuje nejkratší $z \to s$ cesta, prošli jsme všechny vrcholy.

Lemma: Při každé fázi vzroste l alespoň o jedna.

Důkaz: Uvažme síť R, rozdělenou na vrstvy, ještě před pročištěním. Po pročištění některé hrany zmizí. Přibýt[5] mohou jen zrcadlové protějšky již existujících hran.

Uvažme cestu P délky l nebo menší ze $z \to s$ a novou hranu e vzniklou při poslání toku po hraně s nulovým tokem:

a) Hrana $e \notin P \Rightarrow$ zablokování, taková cesta neexistuje.
b) Hrana $e \in P \Rightarrow$ délka $> l$, protože hrana e vede z nějakého vrcholu ve vrstvě C_i do vrcholu ve vrstvě C_{i-1}. ♡

Věta: Dinicův algoritmus najde maximální tok v čase $\mathcal{O}(MN^2)$.

Důkaz: Složitost plyne přímo z předchozího lemmatu a složitosti algoritmu hledání blokujícího toku.

Korektnost (najde vždy maximální tok) dokážeme takto: Nechť algoritmus proběhne. Skončit může jedině tehdy, když nenajde žádnou cestu ze zdroje do spotřebiče v síti rezerv, kterou by mohl být tok vylepšen. Tedy tok musí být nutně maximální. ♡

Takto napsaný algoritmus je ještě příliš pomalý, ale jde výrazně zrychlit. Například namísto přepočítávání toků na rezervy a naopak můžeme minimálně vnitřní cyklus počítat jenom v rezervách. Pročištění se dá dělat jednou namísto dvakrát, můžeme prohledávat do hloubky a mazat při vracení se z vrcholů. Dokonce i hledání blokujícího toku lze řešit v rámci prohledávání do hloubky. Cesta si pamatuje minimum rezerv a při vracení se rezerva snižuje.

Problematika toků v sítích má velké uplatnění v kombinatorice a teorii grafů. Zde uvedeme jeden příklad:

Hledání maximálního párování v bipartitních grafech: Zorientujeme všechny hrany zleva doprava a přidáme zdroj, z něhož vedou hrany do první partity, a stok, do něhož vedou hrany z druhé partity. Hrany mezi partitami mají jednotkovou kapacitu.

[5] Přibudou tak, že po hraně s nulovým tokem pošleme nějaký tok, v opačném směru v síti rezerv vytvoříme z nulové hrany nenulovou.