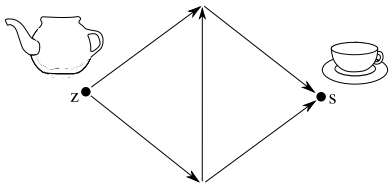
1. Toky v sítích

(zapsala Markéta Popelová)

První motivační úloha: Rozvod čajovodu do všech učeben.

Představme si, že by v budově fakulty na Malé Straně existoval čajovod, který by rozváděl čaj do každé učebny. Znázorněme si to orientovaným grafem, kde by jeden významný vrchol představoval čajovar a druhý učebnu, ve které sedíme. Hrany mezi vrcholy by představovaly větvící se trubky, které mají čaj rozvádět. Jak rozvést co nejefektivněji dostatek čaje do dané učebny?



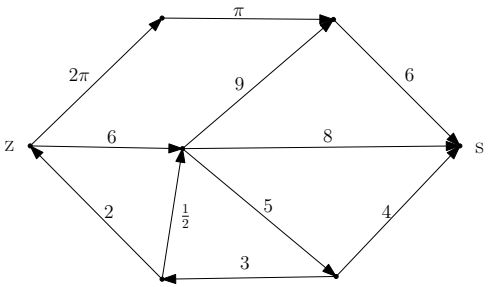
Čajovod

Druhá motivační úloha: Přenos dat.

Jiným příkladem může být počítačová síť na přenos dat, která se sestává z přenosových linek spojených pomocí routerů. Data se sice obvykle přenášejí po paketech, ale to můžeme při dnešních rychlostech přenosu zanedbat a považovat data za spojitá. Jak přenášet data mezi dvěma počítači v síti co nejrychleji?

Definice: *Síť* je uspořádaná pětice (V, E, z, s, c), kde platí:

- (V, E) je orientovaný graf.
- $c : E \to \mathbb{R}_0^+$ je *kapacita* hran.
- $z, s \in V$ jsou dva vrcholy grafu, kterým říkáme *zdroj* a *stok* (spotřebič).
- Graf je symetrický, tedy $\forall u, v \in V : uv \in E \Leftrightarrow vu \in E$ (tuto podmínku si můžeme zvolit bez újmy na obecnosti, neboť vždy můžeme do grafu přidat hranu, která v něm ještě nebyla, a dát jí nulovou kapacitu).



Příklad sítě. Čísla představují kapacity jednotlivých hran.

Definice: *Tok* je funkce $f : E \to \mathbb{R}_0^+$ taková, že platí:

1. Tok po každé hraně je omezen její kapacitou: $\forall e \in E : f(e) \leq c(e)$.
2. Kirchhoffův zákon:

$$\forall v \in V \setminus \{z, s\} : \sum_{u:uv \in E} f(uv) = \sum_{u:vu \in E} f(vu).$$

Neboli pro každý vrchol kromě zdroje a stoku platí, že to, co do něj přitéká, je stejně velké jako to, co z něj odtéká.

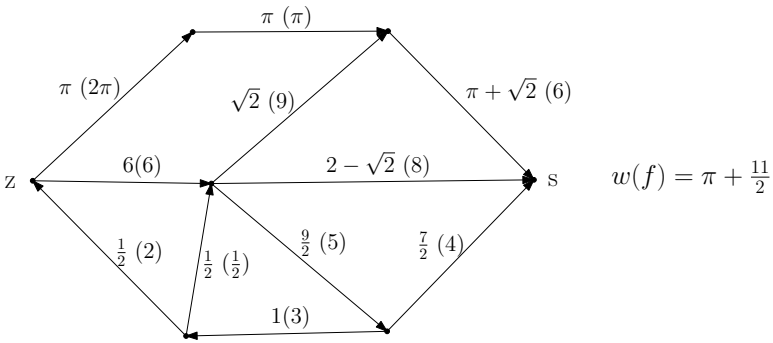
Poznámka: Pro zjednodušení zaveďme speciální značení:

- $f^+(v) = \sum_{u:uv \in E} f(uv)$ (to, co do vrcholu přitéká)
- $f^-(v) = \sum_{u:vu \in E} f(vu)$ (to, co z vrcholu odtéká)
- $f^\Delta(v) = f^+(v) - f^-(v)$ (rozdíl těchto hodnot)

Pak můžeme Krichhoffův zákon zapsat jednoduše jako:

$$\forall v \in V \setminus \{z, s\} : f^\Delta(v) = 0.$$

Poznámka: V angličtině se obvykle zdroj značí s a stok t jako source a target.



Příklad toku. Čísla představují toky po hranách, v závorkách jsou kapacity.

Pozorování: Nějaký tok vždy existuje. V libovolné síti můžeme vždy zvolit funkci nulovou (po žádné hraně nic nepoteče). Tato funkce splňuje podmínky toku, a tedy takovýto nulový tok je zcela korektní.

Definice: *Velikost toku* f je rozdíl součtu velikostí toku na hranách vedoucích do s a součtu velikostí toku na hranách vedoucích z s. Neboli od toho, co do stoku přitéká odečteme to, co ze stoku odtéká.

$$|f| := f^\Delta(s).$$

Cíl: Budeme chtít najít v zadané síti tok, jehož velikost je maximální.

Otázka: Má vůbec smysl mluvit o maximálním toku? Bude vždy existovat? Nevybíráme zde totiž z konečně mnoha případů a na první pohled není jasné, že supremum množiny všech toků bude zároveň i maximum této množiny.

Odpověď: Ano, pro každou síť existuje maximální tok. Toto poměrně překvapivé tvrzení můžeme nahlédnout za pomoci matematické analýzy. Nástin důkazu je takový, že množina toků je kompaktní a velikost toku je spojitá (dokonce lineární) funkce z množiny toků do $\mathbb{R}$. Proto nabývá velikost toku na množině všech toků svého maxima.

Poznámka: Pro naše případy předpokládejme, že kapacity jsou racionální. Poměrně nám to zjednoduší práci a příliš nám to neublíží, neboť práce s reálnými čísly je stejně pro informatika poměrně zapeklitá.

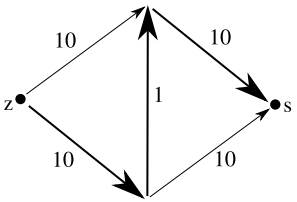
První řešení: Hledejme cestu P ze z do s takovou, že $\forall e \in P : f(e) < c(e)$ (po všech jejích hranách teče ostře méně, než jim dovolují jejich kapacity). Pak zjevně můžeme tok upravit tak, aby se jeho velikost zvětšila. Zvolme

$$\varepsilon := \min_{e \in P}(c(e) - f(e)).$$

Nový tok f' pak definujme jako $f'(e) := f(e) + \epsilon$. Kapacity nepřekročíme (ϵ je největší možná hodnota, abychom tok zvětšili, ale nepřekročili kapaicitu ani jedné z hran cesty P) a Kirchhoffovy zákony zůstanou neporušeny, neboť zdroj a stok nezahrnují a každému jinému vrcholu na cestě P se přítok $f^+(v)$ i odtok $f^-(v)$ zvětší přesně o ϵ.

Otázka: Najdeme takto ovšem opravdu maximální tok?

Odpověď: Nemusíme. Např. na obrázku je vidět, že když najdeme nejdříve cestu přes hranu s kapacitou 1 (na obrázku tučně) a už hodnotu toku na této hraně nesnížíme, tak dosáhneme velikost toku nejvýše 19. Ale maximální tok této sítě má velikost 20.



Čísla představují kapacity jednotlivých hran.

Zde by ovšem situaci zachránilo, kdybychom poslali tok velikosti 1 proti směru prostřední hrany – to můžeme udělat třeba odečtením jedničky od toku po směru hrany.

Někdy je tedy potřeba poslat něco i v protisměru. Definujme si *rezervu hrany*. Ta nám říká, kolik můžeme daným směrem ještě poslat. Využijeme zde, že síť je symetrická.

Definice: *Rezerva hrany* uv je $r(uv) := c(uv) - f(uv) + f(vu)$.

Ukažme si nyní algoritmus, který rezervy využívá, a dokažme, že je konečný a že najde maximální tok každé racionální sítě.

Algoritmus (Fordův-Fulkersonův)

1. $f \leftarrow$ libovolný tok, např. všude nulový ($\forall e \in E : f(e) \leftarrow 0$).
2. Dokud $\exists P$ cesta ze z do s taková, že $\forall e \in P : r(e) > 0$, opakujeme:
3. $\quad \varepsilon \leftarrow \min\{r(e) \mid e \in P\}$.
4. $\quad$ Pro všechny hrany $uv \in P$:
5. $\quad\quad \delta \leftarrow \min\{f(vu), \varepsilon\}$
6. $\quad\quad f(vu) \leftarrow f(vu) - \delta$
7. $\quad\quad f(uv) \leftarrow f(uv) + \varepsilon - \delta$
8. Prohlásíme f za maximální tok.

Problém: Zastaví se Fordův-Fulkersonův algoritmus?

- Pro celočíselné kapacity se v každém kroku zvětší velikost toku alespoň o 1. Algoritmus se tedy zastaví po nejvíce tolika krocích, jako je nějaká horní závora pro velikost maximálního toku – např. součet kapacit všech hran vedoucích do stoku
$$\sum_{u:us \in E} c(us).$$
- Pro racionální kapacity využijeme jednoduchý trik – kapacity vynásobíme společným jmenovatelem a převedeme na původní případ. Uvědomme si, že algoritmus nikde kapacity hran ale ani toky na hranách nedělí, takže už zůstanou celočíselné. A tak jsme převedli racionální kapacity na celočíselné, pro které už víme, že se algoritmus zastaví.
- Na síti s iracionálními kapacitami se algoritmus chová mnohdy divoce, nemusí se zastavit ale dokonce ani konvergovat ke správnému výsledku. **K zamyšlení:** Zkuste vymyslet příklad takové sítě.

Otázka: Vydá algoritmus maximální tok?

Odpověď: Vydá. Abychom si to dokázali, zaveďme si řezy a použijme je jako certifikát maximality nalezeného toku.

Definice: *Řez* je uspořádaná dvojice množin vrcholů (A, B) taková, že A a B jsou disjunktní, pokrývají všechny vrcholy, A obsahuje zdroj a B obsahuje tok. Neboli $A \cap B = \emptyset$, $A \cup B = V$, $z \in A$, $s \notin B$.

Definice: *Hrany řezu* $E(A, B) := E \cap A \times B$.

Poznámka: Řezy se dají definovat více způsoby, jedna z definic je, že řez je množina hran grafu takových, že po jejich odebrání se graf rozpadne na více komponent. Tuto definici splňuje i ta naše, ale ne naopak.

Definice: *Kapacita řezu* je

$$c(A, B) := \sum_{e \in E(A,B)} c(e).$$

Definice: *Tok přes řez* je

$$f(A,B) := \sum_{e \in E(A,B)} f(e) - \sum_{e \in E(B,A)} f(e).$$

Pozorování: Pro každý tok f a každý řez (A,B) platí, že $f(A,B) \leq c(A,B)$.
Důkaz:

$$f(A,B) = \sum_{e \in (A,B)} f(e) - \sum_{e \in E(B,A)} f(e) \leq \sum_{e \in E(A,B)} f(e) \leq \sum_{e \in E(A,B)} c(e) = c(A,B).$$

♡

Lemmátko: Pro každý tok f a pro každý řez (A,B) platí $f(A,B) = |f|$.

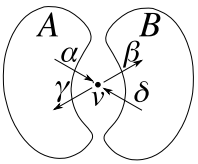
Důkaz: Indukcí a obrázkem.

Začneme s řezem $(V \setminus \{s\}, \{s\})$. Pro tento řez lemma platí z definice velikosti toku. Dále budu postupně přesouvat vrcholy z množiny B do množiny A. Libovolný řez může být takto vytvořen z toho triviálního.

Představme si, že máme již libovolný řez (A,B) a přesouváme vrchol v z A do B. Tedy $A' = A \setminus \{v\}$ a $B' = B \cup \{v\}$.

Uvědomme si, že všechny hrany jednoho typu (např. vedoucí z A do v) se chovají stejně, takže stačí uvažovat hrany pouze 4 typů (+ ostatní hrany (ty, které přesun neovlivní) označíme ε):

- α – hrany vedoucí z A do v
- β – hrany vedoucí z v do B
- γ – hrany vedoucí z v do A
- δ – hrany vedoucí z B do v



Přesun vrcholu v z A do B.

Před přesunem ($v \in A$) se $f(A,B)$ skládá z $\varepsilon + \beta - \delta$. Po přesunu ($v \in B$) se $f(A',B')$ skládá z $\varepsilon + \alpha - \gamma$. Rozdíl těchto hodnot je $\alpha + \delta - \beta - \gamma$.

Nicméně z Kirchhoffova zákonu o vrcholu v (což není ani zdroj ani stok) víme, že $\alpha + \delta - \beta - \gamma = f^{\Delta}(v) = 0$, neboť $\alpha + \delta$ je to, co do v přitéká, a $\beta + \gamma$ je to, co z v vytéká. Tedy tok přes řez před přesunem je stejně velký jako tok přes řez po přesunu. Pokud lemma platilo před přesunem, musí platit i po přesunu. ♡

Důsledek: Pro každý tok f a řez (A,B) platí, že $|f| = f(A,B) \leq c(A,B)$.

Pozorování: Pokud najdeme dvojici tok f a řez (A, B) takovou, že platí $|f| = c(A, B)$, pak tok f je maximální a řez (A, B) minimální.

Věta: Pokud se Fordův-Fulkersonův algoritmus zastaví, tak vydá maximální tok.

Důkaz:

Nechť se Fordův-Fulkersonův algoritmus zastaví. Definujme $A = \{v \in V; \exists$ cesta ze z do v jdoucí po hranách s $r > 0\}$ a $B = V \setminus A$.

Uvědomme si, že (A, B) je řez, neboť $z \in A$ (ze z do z existuje cesta délky 0) a $s \notin B$ (kdyby $s \in B$, tak by musela existovat cesta ze z do s s kladnou rezervou, tudíž by algoritmus neskončil, nýbrž tuto cestu vzal a stávající tok vylepšil).

Dále víme, že všechny hrany řezu mají nulovou rezervu, neboli $\forall uv \in E(A, B)$: $r(uv) = 0$ (kdyby měla hrana uv rezervu nenulovou, tedy kladnou, tak by vrchol v patřil do A). Proto po všech hranách řezu vedoucích z A do B teče tolik, kolik jsou kapacity těchto hran, a po hranách vedoucích z B do A neteče nic, tedy $f(uv) = c(uv)$ a $f(vu) = 0$. Máme řez (A, B) takový, že $f(A, B) = c(A, B)$. To znamená, že jsme našli maximální tok a minimální řez. ♡

Zformulujme si, co jsme zjistili a dokázali o algoritmu pánů Forda a Fulkersona.

Věta: Pro síť s racionálními kapacitami se Fordův-Fulkersonův algoritmus zastaví a vydá maximální tok a minimální řez.

Věta: (Fordova-Fulkersonova)

$$\min_{(A,B)\ \check{r}ez} c(A, B) = \max_{f\ tok} |f|.$$

Důkaz:

Již víme, že $\min_{(A,B)} c(A, B) \geq \max_f |f|$. Stačí tedy dokázat, že vždy existují tok f a řez (A, B) takové, že $c(A, B) = |f|$. Pro racionální kapacity nám Fordův-Fulkersonův algoritmus takový tok (maximální) a řez (minimální) vydá. Jak je to ale s reálnými kapacitami? Využijeme tvrzení, že maximální tok existuje vždy. Pak můžeme spustit Fordův-Fulkersonův algoritmus rovnou na tento maximální tok (místo nulového). Algoritmus se nutně ihned zastaví, neboť neexistuje cesta, která by měla alespoň jednu hranu s kladnou rezervou. A my víme, že pokud se algoritmus zastaví, tak vydá minimální řez. Proto i pro síť s reálnými kapacitami platí, že existuje maximální tok f a minimální řez (A, B) a $c(A, B) = |f|$. ♡

Věta: Síť s celočíselnými kapacitami má aspoň jeden z maximálních toků celočíselný a Fordův-Fulkersonův algoritmus takový tok najde.

Důkaz: Když dostane Fordův-Fulkersonův algoritmus celočíselnou síť, tak najde maximální tok a ten bude zase celočíselný (algoritmus nikde nedělí). ♡

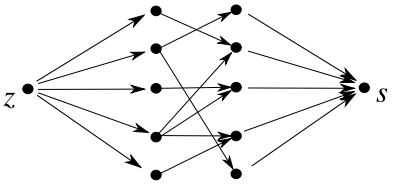
To, že umíme najít celočíselné řešení není úplně samozřejmé. (U jiných problémů takové štěstí mít nebudeme.) Ukažme si rovnou jednu aplikaci, která právě celočíselný tok využije.

Aplikace: Hledání maximálního párování v bipartitních grafech.

Definice: Množina hran $F \subseteq E$ se nazývá *párování*, jestliže žádné dvě hrany této množiny nemají společný ani jeden vrchol. Neboli $\forall e, f \in F : e \cap f = \emptyset$.

Definice: Párování je maximální, pokud obsahuje největší možný počet hran.

Mějme bipartitní graf $G = (V, E)$. V něm hledáme maximální párování. Sestrojme si síť takovou, že vezmeme vrcholy V grafu G a přidáme k nim dva speciální vrcholy z (zdroj) a s (stok) a ze zdroje přidáme hrany do všech vrcholů levé partity a ze všech vrcholů pravé partity povedeme hrany do stoku. Všechny kapacity nastavme na 1. Hrany bipartitního grafu zorientujme z levé partity do pravé. Nyní stačí jen na tuto síť spustit Fordův-Fulkersonův algoritmus (nebo libovolný jiný algoritmus, který najde maximální celočíselný tok) a až doběhne, tak prohlásit hrany s kapacitami 1 za maximální párování.



Hledání maximálního párování v bipartitním grafu.

Existuje totiž bijekce mezi párováním a celočíselnými toky při zachování velikosti. Z každého toku na výše zmíněném grafu (viz obrázek) lze sestrojit párování o stejné velikosti (velikost toku zde odpovídá počtu hran bipartitního grafu, po kterých poteče 1) a naopak. Důležité je si uvědomit, že definice toku (omezení toku kapacitou a Kirchhoffovy zákony) nám zaručují, že hrany s nenulovým tokem (tedy jedničkovým) budou tvořit párování (nestane se, že by dvě hrany začínaly nebo končily ve stejném vrcholu, neboť by se nutně porušila jedna ze dvou podmínek definice toku). Potom i maximální tok bude odpovídat maximálnímu párování a naopak.

V bipartitním grafu najdeme maximální párování v čase $\mathcal{O}(n \cdot (m+n))$. Forův-Fulkersonův algoritmus stráví jednou iterací čas $\mathcal{O}(m+n)$ (za prohledání do šířky) a při jednotkových kapacitách bude iterací nejvýše n, protože každou se tok zvětší alespoň o 1 a všechny toky jsou omezené řezem kolem zdroje, který má kapacitu nejvýše n. Výsledná časová složtost hledání maximálního párování bude tedy $\mathcal{O}(n \cdot (m+n))$.

2. Dinicův algoritmus

(zapsala Markéta Popelová)

Na minulé přednášce jsme si ukázali Fordův-Fulkersonův algoritmus. Tento algoritmus hledal maximální tok tak, že začal s tokem nulovým a postupně ho zvětšoval. Pro každé zvětšení potřeboval v síti najít cestu, na které mají všechny hrany kladnou rezervu (po takovéto cestě můžeme poslat více, než po ní aktuálně teče). Ukázali jsme, že pokud takováto cesta existuje, jde tok vylepšit (zvětšit). Zároveň pokud tok jde vylepšit, pak takováto cesta existuje. Dokázali jsme, že pro racionální kapacity je algoritmus konečný a najde maximální tok.

Forudův-Fulkersonův algoritmus má ovšem značné nevýhody. Funguje pouze pro racionální kapacity a je poměrně pomalý. Nyní si ukážeme jiný algoritmus, který nevylepšuje tok pomocí cest, ale pomocí toků... Budeme k tomi potřebovat síť rezerv.

Definice: Síť rezerv k toku f v síti $S = (V, E, z, s, c)$ je síť $R = (S, f) = (V, E, z, s, r)$, kde $r(e)$ je rezerva hrany e v toku f.

Konvence: Pro hranu e značí $\overleftarrow{e}$ hranu opačnou. Např. pokud $e = uv$, tak $\overleftarrow{e} = vu$.

Je důležité si uvědomit, že síť rezerv je závislá jak na původní síti S, tak na nějakém toku f v síti S. Síť rezerv R se pak od sítě S liší pouze kapacitami – síť R má jako kapacitu hrany rezervu hrany. Pro připomenutí: rezervu hrany e v síti $S = (V, E, z, s, c)$ s tokem f jsme si definovali jako $r(e) = c(e) - f(e) + f(\overleftarrow{e})$.

Než si ukážeme samotný algoritmus, dokážeme si následující lemma.

Lemma: Pro každý tok f v síti S a pro každý tok g v síti $R = (S, f)$ lze v čase $\mathcal{O}(m)$ nalézt tok f' v síti S takový, že $|f'| = |f| + |g|$.

Důkaz:

Důkaz rozdělíme do tří kroků. V prvním kroku si ukážeme, jak budeme tok f' v síti S konstruovat. V druhém kroku dokážeme, že takto zkonstruované f' je opravdu tok. A nakonec ukážeme, že splňuje požadovanou vlastnost, tedy že jeho velikost je součet velikostí toků f a g.

1. konstrukce f'

Pro každou dvojici hran $e, \overleftarrow{e}$ určíme $f'(e)$ a $f'(\overleftarrow{e})$.

- Pokud $g(e) = g(\overleftarrow{e}) = 0$, pak nastavme
  - $f'(e) := f(e)$
  - $f'(\overleftarrow{e}) := f(\overleftarrow{e})$.
- Pokud $g(e) > 0$ a $g(\overleftarrow{e}) = 0$, pak položme
  - $\varepsilon := \min(g(e), f(\overleftarrow{e}))$
  - $f'(e) := f(e) + g(e) - \varepsilon$
  - $f'(\overleftarrow{e}) := f(\overleftarrow{e}) - \varepsilon$.
- Pokud $g(e) > 0$ a $g(\overleftarrow{e}) > 0$, pak položme
  - $\delta := \min(g(e), g(\overleftarrow{e}))$
  - $g'(e) := g(e) - \delta$
  - $g'(\overleftarrow{e}) := g(\overleftarrow{e}) - \delta$.

A tímto jsme to převedli na předchozí případ, který už umíme vyřešit.

2. f' je tok

1. Nejdříve ověřme první podmínku: $\forall e \in E : 0 \leq f(e) \leq c(e)$. Vezměme lib. hranu $e \in E$. Podle toho, co teče po hranách e a $\overleftarrow{e}$ v toku g jsme rozdělili konstrukci toku na tři případy:

1. Pokud po hranách e a $\overleftarrow{e}$ netekl žádný tok g, pak jsme nastavili $f'(e) := f(e)$ a $f'(\overleftarrow{e}) := f(\overleftarrow{e})$. Tedy pokud f dodržoval kapacity, tak pro f' musí platit to samé.
2. Pokud po hraně e tekl tok g nenulový a po opačné nulový, tak jsme zvolili: $f'(e) := f(e) + g(e) - \varepsilon$. Víme, že jsme si ε vybrali tak, že $\varepsilon \leq g(e)$. Proto $f'(e) \geq 0$.
   Teď ověřme, že $f'(e) \leq c(e)$. V případě, že $\varepsilon = g(e)$, tak $f'(e) = f(e) \leq c(e)$. V opačném případě platí, že $\varepsilon = f(\overleftarrow{e})$. Pak ovšem

$$f'(e) = f(e) + g(e) - f(\overleftarrow{e}) \leq f(e) + [c(e) - f(e) + f(\overleftarrow{e})] - f(\overleftarrow{e}) = c(e).$$

   Využili jsme, že g je tok v síti rezerv, tedy $g(e) \leq c(e) - f(e) + f(\overleftarrow{e})$.
   Pro tok $f'(\overleftarrow{e})$ platí, že $\varepsilon \leq f(\overleftarrow{e})$. Proto $f'(\overleftarrow{e}) = f(\overleftarrow{e}) - \varepsilon \geq 0$. Zároveň $f'(\overleftarrow{e}) \leq f'(\overleftarrow{e}) \leq c(\overleftarrow{e})$.
   Tím jsme dokázali, že $f'(e)$ i $f'(\overleftarrow{e})$ dodržují kapacity.
3. V posledním případě tekl po obou hranách kladný tok g. Menší tok z $g(e)$ a $g(\overleftarrow{e})$ jsme vynulovali a od většího odečetli ten menší. Tok $g'(e)$ a $g'(\overleftarrow{e})$ tedy zůstal korektní a tok f' už konstruujeme podle předchozího případu.

2. Teď musíme ještě dokázat, že nový tok neporušuje Kirchhoffovy zákony:

$$\forall v \ \in V \setminus \{z, s\} : f'^{\Delta}(v) = 0.$$

   Vezměme si libovolnou hranu $e \in E$, že $e = uv$. Uvědomme si, že při přechodu z $f(e)$ na $f(\overleftarrow{e})$ a z $f'(e)$ na $f'(\overleftarrow{e})$ bylo:

   · $f^{\Delta}(u)$ sníženo o $g(e)$
   · $f^{\Delta}(v)$ zvýšeno o $g(e)$.

   Pak platí

$$f'^{\Delta}(v) = f^{\Delta}(v) + \sum_{u:uv \in E} g(uv) - \sum_{u:vu \in E} g(vu) =$$

$$= f^{\Delta}(v) + g^{+}(v) - g^{-}(v) = f^{\Delta}(v) + g^{\Delta}(v).$$

   Jelikož f byl tok, tak $f^{\Delta}(v) = 0$ a g byl tok, takže $g^{\Delta}(v) = 0$. Proto $f'^{\Delta}(v) = f^{\Delta}(v) + g^{\Delta}(v) = 0$.

   Tím jsme dokázali, že f' je tok v síti S.

3. $|f'| = |f| + |g|$

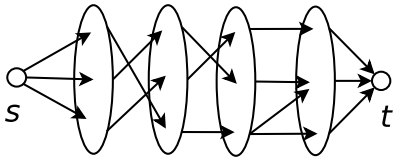
$|f'| = f'^{\Delta}(s) = f^{\Delta}(s) + g^{\Delta}(s) = |f| + |g|$. ♡

Pro algoritmus budeme potřebovat vybírat kvalitní toky g v síti rezerv. Pokud se nám to bude dařit, bude se tok f' rychle zvětšovat, až bychom mohli dojít k maximálnímu toku. Nejlépe by se nám hodily co největší toky v síti rezerv. Kdybychom si dali za cíl najít vždy maximální tok v síti rezerv, výsledek by byl sice krásný (dostali bychom tak rovnou i maximální tok v původní síti), ale problém hledání maximálního toku bychom pouze přenesli na jinou síť. Naše požadavky na tento tok budou tedy takové, aby byl dostatečně velký, ale abychom během jeho hledání nestrávili moc času. Podívejme se, jak se s tímto problémem vyrovná *Dinicův algoritmus*. Nejdříve si ale zadefinujme několik pojmů.

Definice: Tok f je *blokující*, jestliže pro každou orientovanou cestu P ze z do s existuje hrana $e \in P$ taková, že $f(e) = c(e)$.

Definice: Síť je *vrstevnatá (pročištěná)*, když všechny vrcholy a hrany leží na nejkratších cestách ze z do s.

Dinicův algoritmus začíná s nulovým tokem. Potom vždy podle toku f sestrojí síť rezerv a v ní vymaže hrany s nulovou rezervou. Pokud v této promazané síti rezerv neexistuje cesta ze zdroje do stoku, tak skončí a prohlásí tok f za maximální. Jinak pročistí síť rezerv tak, aby se z ní stala vrstevnatá síť (rozdělí vrcholy do vrstev podle vzdálenosti od zdroje a odstraní přebytečné hrany). Ve vrstevnaté síti najde blokující tok, pomocí něhož zlepší tok f. Pak opět pokračuje sestrojením sítě rezerv na tomto vylepšeném toku f atd.



Pročištěná síť rozdělená do vrstev

Algoritmus (Dinicův)

1. $f \leftarrow$ nulový tok.
2. Sestrojíme síť rezerv R a smažeme $e : r(e) = 0$.
3. $l \leftarrow$ délka nejkratší cesty ze z do s v R.
4. Pokud $l = \infty$, zastavíme se a vrátíme f.
5. Pročistíme síť $R \rightarrow$ síť C.
6. $g \leftarrow$ blokující tok v C.
7. Zlepšíme tok f pomocí g.
8. GOTO 2.

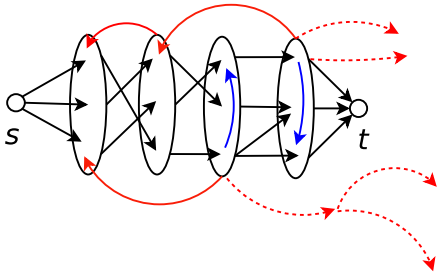
Pozorování: Pokud se algoritmus zastaví, vydá maximální tok.

Důkaz: Víme, že f je stále korektní tok (jediné, jak ho měníme je přičítání toku g, což je, jak jsme si dokázali, „neškodná operace“). Jakmile neexistuje cesta ze z do s v R, tak je f maximální tok, neboť v tuto dobu by se zastavil (a vydal maximální tok) i Fordův-Fulkersonův algoritmus, který je korektní. ♡

A teď ještě musíme ujasnit, jak budeme čistit síť rezerv a vybírat blokující tok g.

Algoritmus pročištění sítě rezerv

1. Rozdělíme vrcholy do vrstev podle vzdálenosti od z.
2. Odstraníme vrstvy za s, hrany do minulých vrstev a hrany uvnitř vrstev.
3. Odstraníme „slepé uličky“, tedy vrcholy s $\deg^{out}(v) = 0$, a to opakovaně pomocí fronty. (Nejdříve zařadíme do fronty všechny vrcholy s $\deg^{out}(v) = 0$. Pak dokud není fronta prázdná, vždy vybereme vrchol v z fronty, odstraníme v a všechny hrany uv. Pro každý takový vrchol u zkontrolujeme, zda se tím nesnížil výstupní stupeň vrcholu u na nulu ($\deg^{out}(u) = 0$). Pokud snížil, tak ho zařadíme do fronty.)



Nepročištěná síť. Obsahuje zpětné hrany, hrany uvnitř vrstvy a slepé uličky.

Hledání blokujícího toku začneme s tokem nulovým. Pak vezmeme vždy orientovanou cestu ze zdroje do stoku v síti C. V této cestě najdeme hranu s nejnižší hodnotou výrazu $r(e) - g(e)$ (neboli $c(e) - f(e)$ v původní síti). Tuto hodnotu označíme ε. Pak ke všem hranám přičteme ε. Pokud tok g na nějaké hraně dosáhne kapacity hrany, což je zde $r(e)$, tak hranu vymažeme. Následně síť dočistíme, aby splňovala podmínky vrstevnaté sítě. A pokud ještě existuje nějaká orientovaná cesta ze zdroje do stoku, tak opět pokračujeme s touto cestou.

Algoritmus hledání blokujícího toku

1. $g \leftarrow$ nulový tok.
2. Dokud existuje orientovaná cesta P ze z do s v C, opakuj:
3. $\quad \varepsilon \leftarrow \min_{e \in P}(r(e) - g(e))$.
4. $\quad$ Pro $\forall e \in P : g(e) \leftarrow g(e) + \varepsilon$.
5. $\quad\quad$ Pokud $g(e) = r(e)$, smažeme e z C.
6. $\quad$ Dočistíme síť zase pomocí fronty.

Časová složitost Rozeberme si jednotlivé kroky algoritmu.

1. Inicializace toku f ... $\mathcal{O}(1)$.
2. Sestrojení sítě rezerv a smazání hran s nulovou rezervou ... $\mathcal{O}(m + n)$.

3. Najití nejkratší cesty (prohledáváním do šířky) ... $\mathcal{O}(m+n)$.
4. Zkontrolování délky nejkratší cesty ... $\mathcal{O}(1)$.
5. Pročištění sítě ... $\mathcal{O}(m+n)$.
    1. Rozdělení vrcholů do vrstev – provedlo již prohledávání do šířky ... $\mathcal{O}(1)$.
    2. Odstranění některých hran ... $\mathcal{O}(m+n)$.
    3. Odstranění „slepých uliček“ pomocí fronty – každou hranu odstraníme nejvýše jedenkrát, každý vrchol se dostane do fronty nejvýše jedenkrát ... $\mathcal{O}(m+n)$.
6. Najití blokujícího toku g ... $\mathcal{O}(m \cdot n)$.
    1. Inicializace toku g ... $\mathcal{O}(1)$.
    2. Najití orientované cesty v pročištěné síti rezerv (stačí vzít libovolnou cestu ze zdroje, neboť každá z nich v této síti vede do stoku) ... $\mathcal{O}(n)$.
    3. Výběr minima z výrazu $r(e) - g(e)$ přes všechny hrany cesty – ta může být dlouhá nejvýše n ... $\mathcal{O}(n)$.
    4. Přepočítání všech hran cesty ... $\mathcal{O}(n)$.
    5. Smazání hran cesty, jejichž tok $g(e)$ se zvýšil na hodnotu $r(e)$ ... $\mathcal{O}(n)$.
    6. Dočišťování vyřešme zvlášť.

    Vnitřní cyklus (kroky 2 až 6) provedeme nejvýše m krát, neboť při každém průchodu vymažeme alespoň jednu hranu (tak jsme si volili ε).
    Čištění během celého hledání blokujícího toku g v pročištěné síti rezerv trvá dohromady $\mathcal{O}(m+n)$, neboť každou hranu a vrchol smažeme nejvýše jedenkrát.
    Najití blokujícího toku bude tedy trvat $\mathcal{O}(m \cdot n + (m+n)) = \mathcal{O}(m \cdot n)$.
7. Zlepšení toku f pomocí toku g ... $\mathcal{O}(m)$.
8. Skok na 2. krok ... $\mathcal{O}(1)$.

Zbývá nám jen určit, kolikrát projdeme vnějším cyklem (fází). Dokážeme si lemma, že hodnota l vzroste mezi průchody vnějším cyklem (fázemi) alespoň o 1. Z toho plyne, že vnějším cyklem můžeme projít nejvýše n-krát, neboť cesta v síti na n vrcholech může být dlouhá nejvýše n.

Uvědomme si, že uvnitř vnějšího cyklu převládá člen $\mathcal{O}(m \cdot n)$, takže celková časová složitost bude $\mathcal{O}(n^2 \cdot m)$.

Lemma: Hodnota l vzroste mezi fázemi alespoň o 1.

Důkaz:

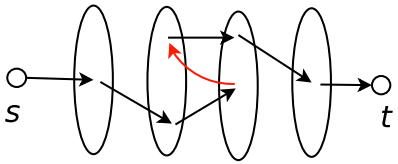
Podíváme se na průběh jednoho průchodu vnějším cyklem. Délku aktuálně nejkratší cesty ze zdroje do stoku označme l. Všechny původní cesty délky l se během průchodu zaručeně nasytí, protože tok g je blokující. Musíme však dokázat, že nemohou vzniknout žádné nové cesty délky l nebo menší. V síti rezerv totiž mohou

hrany nejen ubývat, ale i přibývat: pokud pošleme tok po hraně, po které ještě nic neteklo, tak v protisměru z dosud nulové rezervy vyrobíme nenulovou. Rozmysleme si tedy, jaké hrany mohou přibývat.

Hrany mohou přibývat jen tehdy, když jsme po opačné hraně něco poslali. Ale my něco posíláme po hranách pouze z vrstvy do té následující. Hrany tedy přibývají do minulé vrstvy.

Vznikem nových hran by proto mohly vzniknout nové cesty ze zdroje do stoku, které používají zpětné hrany. Jenže cesta ze zdroje do stoku, která použije zpětnou hranu, musí alespoň jednou skočit o vrstvu zpět a nikdy nemůže skočit o více než jednu vrstvu dopředu, a proto je její délka alespoň $l+2$. Pokud cesta novou zpětnou hranu nepoužije má buď délku $> l$, což je v pořádku, nebo má délku $= l$, pak je zablokovaná.

Tím je lemma dokázáno. ♡



Cesta užívající novou zpětnou hranu

Všechna dokázaná tvrzení můžeme shrnout do následující věty:

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

Poznámka: Algoritmus se chová hezky na sítích s malými racionálními čísly, ale kupodivu i na různých jiných sítích. Často se používá, neboť se chová efektivně. A je mnoho způsobů, jak ho ještě vylepšovat, či odhadovat nižší složitost na speciálních sítích.

Poznámka: Algoritmus nevyžaduje racionální kapacity! Další z důvodů, proč maximální tok existuje i v síti s iracionálními kapacitami.

3. Goldbergův algoritmus

(zapsala Markéta Popelová)

Představíme si nový algoritmus pro hledání maximálního toku v síti, který se ukáže být stejně dobrý jako *Dinicův algoritmus* ($\mathcal{O}(MN^2)$) a po několika vylepšeních bude i lepší. Nejdříve si připomeňme definice, které budeme potřebovat:

Definice: Mějme síť $S = (V, E, z, s, c)$, tok f a libovolný vrchol v. Pak $f^\Delta(v)$ nazýváme *přebytek* ve vrcholu v a definujeme ho:

$$f^\Delta(v) := \sum_{uv \in E} f(uv) - \sum_{vu \in E} f(vu).$$

Přebytek ve vrcholu v je tedy součet všeho, co do vrcholu v přiteče, minus součet všeho, co z v odteče.

Definice: Mějme síť $S = (V, E, z, s, c)$, tok f a libovolnou hranu uv. Pak $r(uv)$ je *rezerva* hrany uv a je definovaná:

$$r(uv) = c(uv) - f(uv) + f(vu).$$

Rezerva hrany značí, co ještě je možno po té hraně poslat.

Poznámka: Budeme používat značení, že N je počet vrcholů a M je počet hran, tedy $N = |V|$ a $M = |E|$.

Goldbergův algoritmus na rozdíl od Dinicova algoritmu začíná s něčím, co pravděpodobně tok není (budeme to nazývat *vlna*), a postupně to zmenšuje až na korektní tok.

Definice: Funkce $f : E \to \mathbb{R}_0^+$ je *vlna* v síti (V, E, z, s, c) tehdy, když jsou splněny následující dvě podmínky:

- $\forall e \in E : f(e) \leq c(e)$
- $\forall v \in V \setminus \{z, s\} : f^\Delta(v) \geq 0$.

Pozorování: Každý tok f je vlna. Neboť $\forall e \in E : f(e) \leq c(e)$ a $\forall v \neq z, s : f^\Delta(v) = 0$.

Operace: *Převedení přebytku*

Algoritmus bude potřebovat převádět přebytky z vrcholu u na sousední vrcholL v. Mějme hranu uv s kladnou rezervou: $r(uv) > 0$ a kladným přebytkem ve vrcholu u: $f^\Delta(u) > 0$. Část přebytku budeme chtít poslat z vrcholu u do vrcholu v. Vezmeme $\delta := \min(f^\Delta(u), r(uv))$ a po hraně uv pošleme tok o velikosti δ. Výsledná situace bude vypadat následovně:

- $f'^\Delta(u) = f^\Delta(u) - \delta$.
- $f'^\Delta(v) = f^\Delta(v) + \delta$.
- $r'(uv) = r(uv) - \delta$.
- $r'(vu) = r(vu) + \delta$.

Kdybychom ovšem nepřidali žádnou jinou podmínku, náš algoritmus by se mohl krásně zacyklit (např. posílat přebytek z u do v a zase zpátky). Abychom se tomu vyhnuli, zavedeme *výšku vrcholu* $h : V \to \mathbb{N}$ a dovolíme převádět přebytek pouze z vyššího vrcholu u na nižší v: $h(u) > h(v)$.

Shrnutí: Podímnky pro převedení přebytku po hraně $uv \in E$:

1. Ve vrcholu u je nenulový přebytek: $f^\Delta(u) > 0$.
2. Vrchol u je výš než vrchol v: $h(u) > h(v)$.
3. Hrana uv má nenulovou rezervu: $r(uv) > 0$.

Definice: Řekneme, že převedení je *nasycené*, pokud po převodu rezerva na hraně uv klesla na nulu, tedy $r(uv) = 0$. Naopak převedení je *nenasycené*, pokud po převodu klesl přebytek ve vrcholu u na nulu, tedy $f^\Delta(u) = 0$. Pokud $r(uv) = 0$ a $f^\Delta(u) = 0$, budeme převedení považovat za *nasycené*.

Operace: Pro vrchol $u \in V$ definujme *zvednutí vrcholu*: Pokud během výpočtu narazíme ve vrcholu u na přebytek, který nelze nikam převést, zvětšíme výšku vrcholu u o jedničku, tj. $h(u) \leftarrow h(u) + 1$.

Algoritmus (Goldbergův)

1. $\forall v \in V : h(v) \leftarrow 0$ (všem vrcholům nastavíme počáteční výšku nula) a $h(z) \leftarrow N$ (zdroj zvedneme do výšky N).
2. $\forall e \in E : f(e) \leftarrow 0$ (po hranách nejdříve nenecháme protékat nic) a $\forall zu \in E : f(zu) \leftarrow c(zu)$ (ze zdroje pustíme maximální možnou vlnu).
3. Dokud $\exists u \in V \setminus \{z, s\} : f^{\Delta}(u) > 0$:
4. Pokud $\exists v \in V : uv \in E,\ r(uv) > 0$ a $h(u) > h(v)$, pak převedeme přebytek po hraně z u do v.
5. V opačném případě zvedneme u: $h(u) \leftarrow h(u) + 1$.
6. Vrátíme tok f jako výsledek.

Nyní bude následovat několik lemmat a invariantů, jimiž dokážeme správnost a konečnost Goldbergova algoritmu.

Invariant A (základ):

1. Funkce f je v každém kroku algoritmu vlna.
2. $h(v)$ nikdy neklesá pro žádné v.
3. $h(z) = N$ a $h(s) = 0$ po celou dobu algoritmu.

Důkaz: Indukcí dle běhu algoritmu.

Na začátku je vše v pořádku (f je nulová funkce, přebytky všech vrcholů jsou nezáporné, tedy f je vlna, $h(z) = N$ a $h(s) = 0$. V průběhu se tyto hodnoty mění pouze při:

- Převedení po hraně uv: Po hraně uv se nepošle více než její rezerva. Přebytek u se sníží, ale nejméně na nulu. Přebytek v se zvýší. Tedy f zůstává vlnou. Výšky se nemění.
- Zvednutí vrcholu u: Mění pouze výšky – a to vrcholů různých od zdroje či stoku – a pouze je zvyšuje. ♡

Invariant S (o Spádu): Neexistuje hrana $uv \in E : r(uv) > 0$ & $h(u) > h(v) + 1$ (s kladnou rezervou a spádem větším než jedna).

Důkaz: Indukcí dle běhu algoritmu.

Na začátku mají všechny hrany ze zdroje rezervu nulovou a všechny ostatní vedou mezi vrcholy s výškou 0. V průběhu by se tento invariant mohl pokazit pouze dvěma způsoby:

- Zvednutím vrcholu u, ze kterého vede hrana uv s kladnou rezervou a spádem 1. Tento případ nemůže nastat, neboť hranu zvedáme pouze tehdy, když neexistuje vrchol v takový, že hrana uv má kladnou rezervu a spád alespoň 1. Takový vrchol v našem případě existuje, proto se místo zvednutí vrcholu u pošle přebytek po hraně uv.
- Zvětšením rezervy hrany se spádem větším než 1. Toto také nemůže nastat, neboť rezervu bychom mohli zvětšit jedině tak, že bychom poslali

něco v protisměru – a to nesmíme, jelikož bychom poslali přebytek z nižšího vrcholu na vyšší. ♡

Lemma K (o Korektnosti): Když se algoritmus zastaví, je f maximální tok.

Důkaz: Důkaz rozložme do dvou kroků. Nejdříve ukážeme, že f je tok, a pak jeho maximalitu.

1. Nechť se algoritmus zastavil. Pak nemohl existovat žádný vrchol v (kromě zdroje a stoku) s kladným přebytkem. Tedy $\forall v \in V \setminus \{z, s\} : f^{\Delta}(v) = 0$. (Víme již, že f je po celou dobu vlna, takže přebytek nemůže být nikdy záporný.) V tom případě splňuje f podmínky toku.
2. Pro spor předpokládejme, že tok f není maximální. Pak existuje zlepšující cesta, tedy cesta ze zdroje do stoku, jejíž všechny hrany mají kladnou rezervu. Vezměme si libovolnou takovou cestu. Zdroj je stále ve výšce N a spotřebič ve výšce 0 (viz invariant A). Tato cesta tedy překonává výšku N, ale může mít nejvýše $N - 1$ hran. Proto existuje alespoň jedna hrana se spádem alespoň 2. Tato hrana tedy nemůže mít kladnou rezervu (viz invariant S). Tato cesta proto nemůže být zlepšující, což je spor. Tím jsme dokázali, že f je nutně maximální tok. ♡

Definice: Cestu P nazveme *nenasycenou*, pokud všechny její hrany mají kladnou rezervu. Neboli $\forall e \in P : r(e) > 0$.

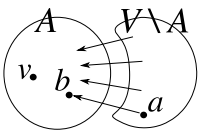
Lemma C (Cesta): Mějme vrchol $v \in V$. Pokud $f^{\Delta}(v) > 0$, pak existuje nenasycená cesta z vrcholu v do zdroje.

Důkaz: Mějme nějaký vrchol $v \in V$ takový, že $f^{\Delta}(v) > 0$. Potom definujme množinu $A := \{u \in V : \exists \text{ nenasycená cesta z } v \text{ do } u\}$.

Sečtěme přebytky ve všech vrcholech množiny A. Přebytek každého vrcholu se spočítá jako součet toků do něj vstupujících minus součet toků z něj vystupujících. Všechny hrany, jejichž oba vrcholy leží v A, se jednou přičtou a jednou odečtou. Proto nás budou zajímat pouze hrany mezi A a $V \setminus A$.

$$\sum_{u \in A} f^{\Delta}(u) = \underbrace{\sum_{ab \in E \cap (V \setminus A \times A)} f(ab)}_{=0} - \underbrace{\sum_{ab \in E \cap (A \times V \setminus A)} f(ab)}_{\geq 0} \leq 0.$$

Ukažme si, proč $\sum_{ab \in E \cap (V \setminus A \times A)} f(ab) = 0$. Mějme vrcholy $a \in V \setminus A$ a $b \in A$ takové, že $ab \in E$. O nich víme, že $r(ab) = 0$ (jinak by a patřilo do A) $\Rightarrow f(ab) = 0$. Proto do A nic nepřitéká.



Obrázek k důkazu lemmatu C

Druhý člen $\sum_{ab \in E \cap (A \times V \setminus A)} f(ab) \geq 0$ je zřejmý, neboť tok na hraně je vždy nezáporný a součet nezáporných čísel je nezáporné číslo.

Proto $\sum_{u \in A} f^{\Delta}(u) \leq 0$. Zároveň však v A je aspoň jeden vrchol s kladným přebytkem, totiž v, proto v A musí být také vrchol se záporným přebytkem – a jediný takový je zdroj. Tím je dokázáno, že $z \in A$, tedy že vede nenasycená cesta z vrcholu v do zdroje.

♡

Invariant V (Výška): $\forall v \in V$ platí $h(v) \leq 2N$.

Důkaz: Kdyby existoval vrchol v s výškou $h(v) > 2N$, tak by musel být někdy zvednut z výšky $2N$. Tehdy musel mít kladný přebytek $f^{\Delta}(v) > 0$ (jinak by nemohl být zvednut). Dle lemmatu C musela existovat nenasycená cesta z v do zdroje. Tato cesta měla spád alespoň N, ale mohla mít nejvýše $N - 1$ hran (jinak by to nebyla cesta v síti na N vrcholech). Tudíž musela na této cestě existovat hrana se spádem alespoň 2, což je spor s invariantem S (neboť všechny hrany této cesty mají z definice nenasycené cesty kladné rezervy). ♡

Lemma Z (počet Zvednutí): Počet všech zvednutí je maximálně $2N^2$.

Důkaz: Stačí si uvědomit, že každý vrchol můžeme zvednout maximálně $2N$-krát a vrcholů je N. ♡

Lemma S (naSycená převedení): Počet všech nasycených převedení je nejvýš NM.

Důkaz: Pro každou hranu uv spočítejme počet nasycených převedení (tedy takových převedení, že po nich klesne rezerva hrany na nulu). Abychom dvakrát nasyceně převedli přebytek (nebo jeho část) z vrcholu u do vrcholu v, tak jsme museli u mezitím alespoň dvakrát zvednout. Ale nejvýše můžeme hranu zvednout $2N$-krát. Všech hran je M, proto počet všech nasycených převedení je nejvýše NM. ♡

Lemma N (Nenasycená převedení): Počet všech nenasycených převedení je $\mathcal{O}(N^2 M)$.

Důkaz: Důkaz provedeme pomocí potenciálové metody – nadefinujme si následující funkci jako potenciál:

$$\Phi := \sum_{\substack{v : f^{\Delta}(v) > 0 \\ v \neq z, s}} h(v).$$

Nyní se podívejme, jak se náš potenciál během algoritmu vyvíjí a jaké má vlastnosti:

- Na počátku je $\Phi = 0$.
- Během celého algoritmu je $\Phi \geq 0$, neboť je součtem nezáporných členů.
- Zvednutí vrcholu zvýší Φ o jedničku (Aby byl vrchol zvednut, musel mít kladný přebytek $\Rightarrow$ vrchol do sumy již přispíval, teď jen přispěje číslem o 1 vyšším.). Již víme, že za celý průběh algoritmu je všech zvednutí maximálně $2N^2$, proto zvedáním vrcholů zvýšíme potenciál dohromady nejvýše o $2N^2$.
- Nasycené převedení zvýší Φ nejvýše o $2N$, protože buď po převodu hranou uv v u zůstal nějaký přebytek, takže se mohl potenciál zvýšit nejvýše

o $h(v) \leq 2N$, nebo je přebytek v u po převodu nulový a potenciál se dokonce o jedna snížil. Za celý průběh tak dojde k maximálně NM takovýmto převedením, díky nimž se potenciál zvýší maximálně o $2N^2M$.

- Konečně když převádíme po hraně uv nenasyceně, tak od potenciálu určitě odečteme výšku vrcholu u (neboť se vynuluje přebytek ve vrcholu u) a možná přičteme výšku vrcholu v. Jenže $h(v) = h(u) - 1$, a proto nenasycené převedení potenciál vždy sníží alespoň o jedna.

Z tohoto rozboru chování potenciálu Φ v průběhu algoritmu získáváme, že počet všech nenasycených převedení může být nejvýše $2N^2 + 2N^2M$, což je $\mathcal{O}(N^2M)$. ♡

Implementace:

Budeme si pamatovat seznam P všech vrcholů $v \neq z, s$ s kladným přebytkem. Neboli

$$P = \{v \in V \setminus \{z, s\} \mid f^{\Delta}(v) > 0\}.$$

Když měníme přebytek nějakého vrcholu, můžeme náš seznam v konstantním čase aktualizovat (např. tak, že si každý vrchol pamatuje pozici, na které v seznamu P je). V konstantním čase také umíme odpovědět, zda existuje nějaký vrchol s přebytkem.

Dále si pro každý vrchol $u \in V$ budeme pamatovat $L(u)$ seznam hran $uv \in E$ takových, které vedou dolů (mají spád alespoň 1) a kladnou rezervu. Neboli

$$L(u) = \{uv \in E \mid v \in V,\ r(uv) > 0,\ h(v) < h(u)\}.$$

Díky tomu můžeme přistupovat k patřičným sousedům u v čase $\mathcal{O}(1)$, stejně jako přidávat hrany do $L(u)$, resp. je mazat. Opět každá hrana si bude pamatovat pozici, na které se nachází v seznamu L.

Rozbor časové složitosti algoritmu:

1. Inicializace výšek ... $\mathcal{O}(1)$.
2. Inicializace vlny f ... $\mathcal{O}(M)$.
3. Výběr vrcholu u s kladným přebytkem – vezmeme první vrchol v P ... $\mathcal{O}(1)$.
4. Výběr vrcholu v, do kterého vede z u hrana s kladnou rezervou a který je níže než u – vezmeme první hranu z $L(u)$ ... $\mathcal{O}(1)$.
   Převedení přebytku: ... $\mathcal{O}(1)$.
   - Nasycené převedení ... $\mathcal{O}(1)$.
     - Rezerva hrany uv klesne na nulu ⇒ hrana uv vypadne z $L(u)$ ... $\mathcal{O}(1)$.
     - Přebytek vrcholu v se zvýší ⇒ pokud ještě nebyl v seznamu P, tak se tam přidá ... $\mathcal{O}(1)$.
     - Přebytek vrcholu u možná také klesne na nulu ⇒ pak by vrchol u vypadnul z P ... $\mathcal{O}(1)$.
   - Nenasycené převedení ... $\mathcal{O}(1)$.

· Rezerva hrany uv zůstane nezáporná $\Rightarrow$ hrana uv zůstane v $L(u)$ ... $\mathcal{O}(1)$.
· Vynuluje se přebytek vrcholu $u \Rightarrow$ vrchol u vypadne z P ... $\mathcal{O}(1)$.
· Přebytek vrcholu v se zvýší $\Rightarrow$ pokud ještě nebyl v seznamu P, tak se tam přidá ... $\mathcal{O}(1)$.

5. Zvednutí vrcholu u ... $\mathcal{O}(N)$.
   Musíme obejít všechny hrany do u a z u, kterých je nejvýše $2N - 2$, porovnat výšky a případně tyto hrany uv odebrat ze seznamu $L(u)$ resp. přidat do $L(v)$. Abychom pro odebrání hrany uv ze seznamu $L(u)$ nemuseli procházet celý seznam, budeme si $\forall v \in V$ pamatovat ještě $L^{-1}(v) :=$ seznam ukazatelů na hrany uv v seznamech $L(u)$.

Vidíme, že každé zvednutí je sice drahé, ale je jich zase poměrně málo. Naopak převádění přebytků je častá operace, takže je výhodné, že trvá konstantní čas.

Shrnutí:

- Všech zvednutí je $\mathcal{O}(N^2)$ (viz lemma Z), každé trvá $\mathcal{O}(N) \dots \mathcal{O}(N^3)$.
- Všech nasycených převedení je $\mathcal{O}(NM)$ (viz lemma S), každé trvá $\mathcal{O}(1) \dots \mathcal{O}(NM)$.
- Všech nenasycených převedení je $\mathcal{O}(N^2M)$ (viz lemma N), každé trvá $\mathcal{O}(1) \dots \mathcal{O}(N^2M)$.

Dohromady má tedy Goldbergův algoritmus časovou složitost $\mathcal{O}(N^2M)$. Vidíme, že už v tomto obecném případě to není horší než Dinicův algoritmus. Příště si ukážeme, že může mít i mnohem lepší. Nejdříve ale zformulujme všechna dokázaná tvrzení do následující věty:

Věta: Goldbergův algoritmus najde maximální tok v čase $\mathcal{O}(N^2M)$.