3. Dinicův algoritmus

V minulé kapitole jsme ukázali Fordův-Fulkersonův algoritmus. Tento algoritmus hledá maximální tok tak, že začne s tokem nulovým a postupně ho zvětšuje. Pro každé zvětšení potřebuje v síti najít *nenasycenou cestu,* tedy takovou, na níž mají všechny hrany kladnou rezervu. Podél takové cesty pak tok zlepší.

Ukázali jsme, že tato cesta existuje právě tehdy, když tok ještě není maximální. Také jsme dokázali, že pro racionální kapacity je algoritmus konečný a vždy najde maximální tok. V obecném případě to ovšem může trvat velice dlouho.

Nyní ukážeme trochu složitější, ale výrazně rychlejší Dinicův algoritmus založený na myšlence nezlepšovat toky pomocí cest, ale rovnou pomocí toků ...

Síť rezerv a zlepšující toky

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 při toku f.

Je důležité si uvědomit, že síť rezerv konstruujeme pro konkrétní tok v původní síti. Od původní sítě se liší pouze tím, že kapacity hran jsme nahradili jejich rezervami. Připomeňme ještě, že rezervu hrany uv jsme definovali vztahem

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

Nyní ukážeme, jak tok zlepšit pomocí toku v příslušné síti rezerv:

Lemma Z (o zlepšování toků): Pro libovolný tok f v síti S a libovolný 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: Nejdřív ukážeme, jak tok f' sestrojit. Poté dokážeme, že konstrukce opravdu dává korektní tok. Nakonec nahlédneme, že jeho velikost je taková, jakou lemma slibuje.

Zjednodušení toku g: Abychom si usnadnili práci, upravíme nejprve tok g tak, aby pro každou dvojici hran uv a vu byl tok g nenulový nejvýše na jedné z nich. Kdyby totiž jak $g(uv)$, tak $g(vu)$ byly kladné, můžeme od obou hodnot odečíst tu menší z nich. Tím jsme nezměnili přebytky vrcholů, tím pádem ani velikost toku, a tok po jedné z hran jsme vynulovali.

Konstrukce f': Pro každou dvojici hran uv a vu takovou, že $g(vu) = 0$, definujeme funkci f' následovně:

$$\begin{aligned} \delta &:= \min(f(vu), g(uv)) \\ f'(vu) &:= f(vu) - \delta \\ f'(uv) &:= f(uv) + g(uv) - \delta \end{aligned}$$

(To je vlastně totéž jako při zlepšování toku ve Fordově-Fulkersonově algoritmu.)

Proč je to tok: Funkce f' jistě není na žádné hraně záporná. Kapacity také nepřekročí: na hraně vu tok pouze snižujeme, na hraně uv ho zvýšíme jen tehdy, když $\delta < g(uv)$. Tehdy musí být $\delta = f(vu)$. Využijeme toho, že g je tok v síti rezerv, takže $g(uv) \leq r(uv) = c(uv) - f(uv) + f(vu)$. Proto $f'(uv) = f(uv) + g(uv) - \delta \leq f(uv) + c(uv) - f(uv) + f(vu) - f(vu) = c(uv)$.

Ještě ověříme, že f' splňuje Kirchhoffův zákon. Ukážeme, že pro každý vrchol v platí $f'^{\Delta}(v) = f^{\Delta}(v) + g^{\Delta}(v)$, takže pokud zákon platil pro f i g, musí platit i pro f'. Stačí si všimnout, že za hranu uv naše konstrukce zvýší $f^{\Delta}(u)$ o $-g(uv)$ a $f^{\Delta}(v)$ o $g(uv)$. To je přesně tolik, čím tato hrana přispívá k $g^{\Delta}(u)$ a $g^{\Delta}(v)$. Pro hranu vu, po které neteklo nic, platí triviálně totéž.

Velikost toku: Stačí využít toho, že přebytky se sečetly, abychom získali:

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

Časová složitost: Jak zjednodušení toku g, tak výpočet toku f' trvají $\mathcal{O}(1)$ pro každou hranu, celkově tedy $\mathcal{O}(m)$. ♡

Všimněte si, že zlepšení po nenasycené cestě je speciálním případem tohoto postupu – odpovídá totiž toku v síti rezerv, který je konstantní na oné cestě a všude jinde nulový.

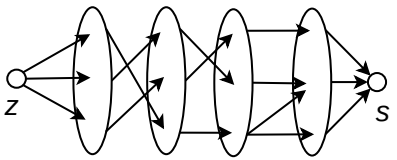
Dinicův algoritmus

Dinicův algoritmus bude postupně hledat nějaké pomocné toky v síti rezerv, původně nulový tok pomocí nich zlepšovat, až se dostane k maximálnímu toku. Počet potřebných iterací přitom bude záviset na tom, jak „kvalitní“ pomocné toky sežene – na jednu stranu bychom chtěli, aby byly podobné maximálnímu toku, na druhou stranu jejich výpočtem nechceme trávit příliš mnoho času. Vhodným kompromisem jsou tzv. blokující toky:

Definice: Tok je *blokující*, jestliže na každé orientované cestě ze zdroje do spotřebiče existuje alespoň jedna hrana, na níž je tok roven kapacitě.

Definice: Síť je *vrstevnatá (pročištěná)*, pokud všechny její vrcholy a hrany leží na nejkratších cestách ze z do s. (Abychom vyhověli naší definici sítě, musíme ke každé takové hraně přidat hranu opačnou s nulovou kapacitou, ale ty algoritmus nebude používat a ani udržovat v paměti.)

Pozorování: Představme si rozdělení sítě na vrstvy, přičemž v i-té vrstvě leží ty vrcholy, jejichž vzdálenost od zdroje je rovna i. Z i-té vrstvy mohou hrany vést pouze do vrstev $0, 1, \ldots, i, i+1$ – tedy pouze uvnitř vrstvy, zpět a o právě jednu vrstvu dopředu. Po pročištění zbudou pouze hrany do následující vrstvy, proto se pročištěné síti také říká *vrstevnatá.*



Pročištěná síť rozdělená do vrstev

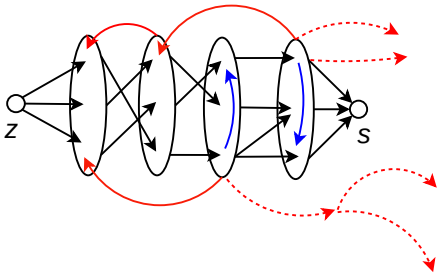
Algoritmus DINIC
Vstup: Síť (V, E, c, z, s).

1. $f \leftarrow$ nulový tok.
2. Opakujeme:
3. Sestrojíme síť rezerv R a smažeme hrany s nulovou rezervou.
4. $\ell \leftarrow$ délka nejkratší cesty ze z do s v R.
5. Pokud $\ell = \infty$, zastavíme se a vrátíme výsledek f.
6. Pročistíme síť R.
7. $g \leftarrow$ blokující tok v R.
8. Zlepšíme tok f pomocí g.

Výstup: Maximální tok f.

Procedura ČIŠTĚNÍSÍTĚ

1. Rozdělíme vrcholy do vrstev podle vzdálenosti od z.
2. Odstraníme vrstvy za s (tedy vrcholy ve vzdálenosti větší než ℓ).
3. Odstraníme hrany do předchozích vrstev a hrany uvnitř vrstev.
4. Odstraníme „slepé uličky“, tedy vrcholy s $\deg^-(v) = 0$:
5. $F \leftarrow \{v \neq s \mid \deg^+(v) = 0\}$. *(fronta vrcholů ke smazání)*
6. Dokud $F \neq \emptyset$, opakujeme:
7. Odebereme vrchol v z F.
8. Smažeme ze sítě vrchol v i všechny hrany, které do něj vedou.
9. Pokud nějakému vrcholu klesl $\deg^+$ na 0, přidáme ho do F.



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

Jak nalézt blokující tok: Začneme s nulovým tokem g a budeme ho postupně zlepšovat. Pokaždé najdeme nějakou orientovanou cestu ze zdroje do stoku – to se ve vrstevnaté síti dělá snadno, stačí vyrazit ze zdroje a pak následovat libovolnou hranu. Až cestu najdeme, tok g podél ní zlepšíme, jak nejvíce to půjde.

Pokud nyní tok na nějakých hranách dosáhl jejich rezervy, tyto hrany smažeme. Tím jsme mohli porušit pročištěnost – pakliže nějaký vrchol přišel o poslední odchozí nebo poslední příchozí hranu. Takových vrcholů se opět pomocí fronty zbavíme a síť dočistíme. Pokračujeme zlepšováním po dalších cestách, dokud nějaké existují.[1]

[1] Všimněte si, že algoritmus skončí tím, že smaže všechny vrcholy i hrany. Také si

Celé hledání blokujícího toku tedy vypadá následovně:

Procedura BlokujícíTok
Vstup: Vrstevnatá síť R s rezervami r.

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

Výstup: Blokující tok g.

Analýza Dinicova algoritmu

Lemma K (o korektnosti): Pokud se algoritmus zastaví, vydá maximální tok.
Důkaz: Z lemmatu o zlepšování toků plyne, že f je stále korektní tok. Algoritmus se zastaví tehdy, když už neexistuje cesta ze z do s po hranách s kladnou rezervou. Tehdy by se zastavil i Fordův-Fulkersonův algoritmus, a ten, jak už víme, je korektní. ♡

Nyní rozebereme časovou složitost. Rozdělíme si k tomu účelu algoritmus na *fáze* – tak budeme říkat jednotlivým průchodům vnějším cyklem. Také budeme předpokládat, že síť na vstupu neobsahuje izolované vrcholy, takže $\mathcal{O}(n + m) = \mathcal{O}(m)$.

Lemma S (o složitosti fází): Každá fáze trvá $\mathcal{O}(nm)$.
Důkaz: Sestrojení sítě rezerv, mazání hran s nulovou rezervou, hledání nejkratší cesty i konečné zlepšování toku trvají $\mathcal{O}(m)$.

Čištění sítě (i se všemi dočišťováními během hledání blokujícího toku) pracuje taktéž v $\mathcal{O}(m)$: Smazání hrany trvá konstantní čas, smazání vrcholu po smazání všech incidentních hran taktéž. Každý vrchol i hrana jsou smazány nejvýše jednou za fázi.

Hledání blokujícího toku projde nejvýše m cest, protože pokaždé ze sítě vypadne alespoň jedna hrana (ta, na níž se v kroku 3 nabývalo minimum) a už se tam nevrátí. Nalezení cesty metodou „rovnou za nosem“ přitom trvá $\mathcal{O}(n)$. Celkem tedy $\mathcal{O}(nm)$ plus čištění, které jsme ale už započítali.

Celá jedna fáze proto doběhne v čase $\mathcal{O}(m + m + nm) = \mathcal{O}(nm)$. ♡

Zbývá nám jen určit, kolik proběhne fází. K tomu se bude hodit následující lemma:

Lemma C (o délce cest): Délka ℓ nejkratší cesty ze z do s vypočtená v kroku 4 Dinicova algoritmu po každé fázi vzroste alespoň o 1.
Důkaz: Označme R_i síť rezerv v i-té fázi poté, co jsme z ní smazali hrany s nulovou rezervou, ale ještě před pročištěním. Nechť nejkratší cesta ze z do s v R_i je dlouhá ℓ.

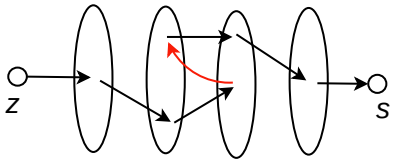
všimněte, že vrcholy s nulovým vstupním stupněm jsme ani nemuseli mazat, protože se do nich algoritmus při hledání cest nikdy nedostane.

Jak se liší R_{i+1} od R_i? Především jsme z každé cesty délky ℓ smazali alespoň jednu hranu: každá taková cesta totiž byla blokujícím tokem zablokována, takže alespoň jedné její hraně klesla rezerva na nulu, čímž hrana vypadla. Žádná z původních cest délky ℓ tedy již v R_{i+1} neexistuje.

To ovšem nestačí – hrany mohou také přibývat. Pokud nějaká hrana měla nulovou rezervu a během fáze jsme zvýšili tok v protisměru, rezerva se zvětšila a hrana se v R_{i+1} najednou objevila. Ukážeme ale, že všechny cesty, které tím nově vznikly, jsou příliš dlouhé.

Rozdělme vrcholy grafu do vrstev podle vzdáleností od zdroje v R_i. Tok jsme zvyšovali pouze na hranách vedoucích o jednu vrstvu dopředu, takže jediné hrany, které se mohou objevit, vedou o jednu vrstvu zpět. Ovšem každá cesta ze zdroje do spotřebiče, která se alespoň jednou vrátí o vrstvu zpět, musí mít délku alespoň $\ell + 2$ (protože spotřebič je v ℓ-té vrstvě a neexistují hrany, které by vedly o více než 1 vrstvu dopředu).

Tím je lemma dokázáno. ♡



Cesta užívající novou zpětnou hranu

Věta: Dinicův algoritmus najde maximální tok v čase $\mathcal{O}(n^2m)$.
Důkaz: Jelikož každá cesta obsahuje nejvýše n vrcholů, z lemmatu C plyne, že fází proběhne nejvýše n. Každá fáze podle lemmatu S trvá $\mathcal{O}(nm)$, což dává celkovou složitost $\mathcal{O}(n^2m)$. Speciálně se tedy algoritmus vždy zastaví, takže podle lemmatu K vydá maximální tok. ♡

Poznámka: Na rozdíl od Fordova-Fulkersonova algoritmu jsme tentokrát nikde nevyžadovali racionálnost kapacit – odhad časové složitosti se o kapacity vůbec neopírá. Nezávisle jsme tedy dokázali, že i v sítích s iracionálními kapacitami vždy existuje alespoň jeden maximální tok.

V sítích s malými celočíselnými kapacitami se navíc algoritmus chová daleko lépe, než říká náš odhad. Snadno se dá dokázat, že pro jednotkové kapacity doběhne v čase $\mathcal{O}(mn)$ (stejně jako Fordův-Fulkersonův). Uveďme bez důkazu ještě jeden silnější výsledek: v síti vzniklé při hledání největšího párování algoritmem z minulé kapitoly Dinicův algoritmus pracuje v čase $\mathcal{O}(\sqrt{n} \cdot m)$.

Cvičení

1. Dokažte, že pro jednotkové kapacity Dinicův algoritmus doběhne v čase $\mathcal{O}(mn)$.
2. Blokující tok lze také sestrojit pomocí prohledávání do hloubky. Pokaždé, když projdeme hranou, přepočítáme průběžné minimum. Pokud najdeme stok, vracíme se do kořene a upravujeme tok na hranách. Pokud narazíme na slepou uličku, vrátíme se o krok zpět a smažeme hranu, po níž jsme přišli. Doplňte detaily.