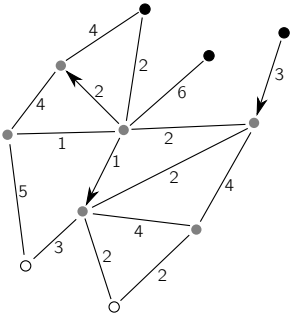
Toky v sítích

Ruský petrobaron vlastní ropná naleziště na Sibiři a trubky vedoucí do Evropy. Trubky vedou mezi nalezišti, uzlovými body a koncovými body, kde ropu přebírají odběratelé. Každá trubka může a nemusí mít definováno, kterým směrem jí má téci ropa. Pro každou trubku zvlášť víme, kolik nejvýše jí za hodinu protlačíme.



Naleziště jsou bezedná a mohou posílat neomezená množství ropy. Odběratelé také dokáží neomezená množství ropy z koncových bodů odebírat. Petrobaron čelí problému, jak protlačit danou distribuční sítí co nejvíce ropy za hodinu ze zdrojů k odběratelům.

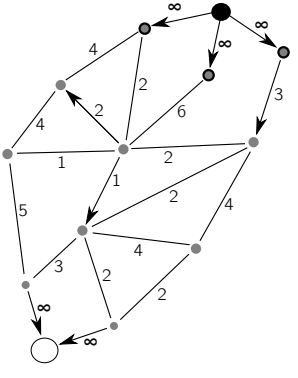
Zapeklité je to hlavně proto, že v uzlových bodech nelze ropu hromadit, ani pálit – rozhodně tedy nejde bez rozmyslu přikázat, ať každou trubkou teče maximum, protože bychom poškodili cenná zařízení a v hnusu labutě zahubili.

Zmatematizování

V zadání vidíme graf, který obsahuje orientované i neorientované hrany, kde je nějaká podmnožina vrcholů označená jako zdroje a jiná jako ... říkejme tomu třeba stoky.

Abychom měli situaci jednodušší, zbavíme se hned na úvod mnohočetnosti zdrojů a stoků. Přikreslíme si dva nové vrcholy – z nadzdroje budeme posílat ropu do všech zdrojů, do nadstoku budeme posílat ropu ze všech stoků. Kapacitu přikreslených hran pak nastavíme na nekonečno.

Teď nám stačí vymyslet algoritmus, který řeší problém s právě jedním zdrojem a právě jedním stokem. Každý vstup totiž popsaným způsobem převedeme, pošleme ho algoritmu a z výstupu prostě jen odstraníme dva přidané vrcholy a připojené hrany.



Podobně se zbavíme neorientovaných hran. Každou takovou hranu v každém zadání změníme na dvojici protisměrných orientovaných hran se stejnou kapacitou. V algoritmu pak už můžeme počítat jen s hranami orientovanými.

Dostáváme se nyní k nejdůležitějšímu – podmínkám na hledaný tok.

Na vstupu dostáváme ohodnocení hran nezápornými čísly a naším úkolem je sestavit jiné ohodnocení těch samých (všech) hran. Je důležité, aby se nám to nepletlo – ohodnocení ze vstupu se říká *kapacita* a značí se $c(e)$, konstruované ohodnocení se jmenuje *tok* a říkáme mu $f(e)$.

Konstruované ohodnocení maximalizujeme, ale omezuje nás kapacita a *Kirchhoffův zákon.* Tak budeme říkat podmínce na to, že součet toku na hranách, které do vrcholu

vstupují, musí být stejný jako součet toku na hranách, které z vrcholu vystupují. Máte-li rádi fyziku nebo, důvod k takovému pojmenování jistě chápete.

Formálně ony dvě podmínky vypadají takto:

$$\forall e \in E : f(e) \leq c(e)$$

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

Kirchhoffova podmínka se samozřejmě netýká ani zdroje, ani stoku – tam nám naopak jde o to ji co nejvíce porušit. Velikost toku je nejsnazší měřit na nich. Budeme ji definovat jako rozdíl mezi součtem odtoků a součtem přítoků ve zdroji.

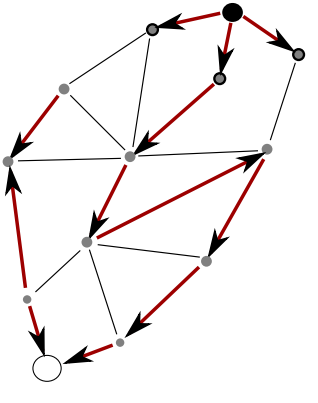
Cvičení

- Neorientované hrany, neboli obousměrné trubky, si zaslouží podrobnější rozbor, než jaký jsme jim věnovali v textu. Jak spolehlivě převedeme řešení algoritmu do původní sítě?
- Vymysleli jsme, jak vyřešit více zdrojů a stoků a jak ošetřit obousměrné trubky. Co kdyby bylo v zadání omezení na průtok vrcholy?
- Umíte dokázat, že je absolutní hodnota rozdílu přítoků a odtoků stejná na zdroji i na stoku? Tedy že bychom mohli velikost toku stejně tak dobře měřit i na stoku?

Řešení

Problém je velmi studovaný a k jeho řešení existují dva velké přístupy, které jsou humorně protikladné. Ten první vezme nulový tok a opatrně ho zlepšuje. Druhý si napíská veliké ohodnocení hran, které ani tokem není, a pak ho opravuje.

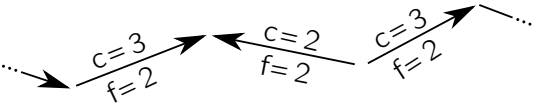
Předvedeme si onen první způsob a algoritmus, který se podle svých autorů jmenuje Fordův-Fulkersonův. Bude se nám odteď hodit tvářit se, že hrany mezi dvěma vrcholy buď vedou oběma směry, anebo žádným (tj. tam, kde hrana vedla jen jedním směrem, doplníme hranu druhým směrem). Přidaným hranám dáme nulovou kapacitu.



Představme si graf, na kterém počítáme tok a dejme tomu, že už nějaký tok máme – třeba prázdný. Představme si, že jsme ropný magnát a každý rozdíl mezi kapacitou potrubí a jejím využitím (tokem) nás stojí miliony dolarů. Už jsme se smířili s tím, že každá trubka nemůže být využita na maximum, ale … zkusme si vyznačit ty hrany, kde $c(e) \neq f(e)$.

Co když existuje cesta z nadzdroje do nadstoku, která vede pouze po takových hranách? Můžeme vzít minimum z rozdílů na každé hraně a o toto číslo zvýšit tok na každé z nich! Ani kapacitní, ani Kirchhoffovu podmínku to jistě nepoškodí.

Pokud žádnou takovou cestu nevidíme, znamená to, že tok vylepšit nejde? Ne úplně. Představte si následující situaci:



Copak nejde zlepšit? Jde! Není na to první pohled úplně jasné, ale můžeme zlepšovat výsledný tok i tím, že ho na protisměrné části cesty snížíme. Samozřejmě však nesmíme nastavovat tok záporný.

(Je smutné, že si teď trochu kazíme grafovou terminologii – co je to za cestu v orientovaném grafu, která nemusí respektovat orientaci hran?)

Takže jaká je přesně podmínka pro „vyznačení“ hrany $\overrightarrow{uv}$? Nastává $f(\overrightarrow{uv}) < c(\overrightarrow{uv})$ nebo $f(\overrightarrow{vu}) > 0$. Potom ji lze zlepšit o $c(\overrightarrow{uv}) - f(\overrightarrow{uv}) + f(\overrightarrow{vu})$.

Hledání všech vhodných („zlepšujících“) cest tedy můžeme dělat prostým prohledáváním do šířky přes vyznačené hrany. Budeme to dělat opakovaně znovu a znovu, až žádnou takovou nenajdeme, a pak vrátíme získaný tok jako výsledek.

Analýza algoritmu

Správnost

Zavolali jsme algoritmus na prázdný tok, ten ho zlepšil do situace, ve které neexistuje zlepšující cesta.

Znamená tato neexistence, že je výsledný tok maximální? Opačná implikace je jasná – maximální tok zlepšit žádným způsobem nepůjde, takže ani přes zlepšující cesty.

Když zkusíme algoritmus pustit na graf, kde už žádná taková cesta není, můžeme si poznamenat všechny vrcholy, kam jsme se pomocí prohledávání zlepšitelných hran ještě dostali. Tato množina bude jistě obsahovat zdroj (tam jsme začali) a jistě nebude obsahovat stok (to by existovala zlepšující cesta).

Na hranách mezi touto množinou a jejím doplňkem nemůžeme zlepšovat, jinak by se po nich náš program pustil dál a množinu vrcholů, kam se dostal, by rozšířil. Všechny hrany směřující ven tedy mají $f(e) = c(e)$, pro všechny hrany směřující dovnitř platí $f(e) = 0$.

Tyto hrany tvoří *řez* naším grafem. Dovolám se v tuto chvíli na vaši intuici (pro korektní důkaz viz [Skriptíčka]) – tok nemůže být větší než libovolný řez. Z toho už dostáváme, že náš algoritmus našel tok maximální, protože našel také řez, který zaručuje, že nemůže existovat tok větší.

Časová složitost

Je možné dobu běhu omezit počtem vrcholů a hran? Výše uvedeným postupem na grafu s celočíselnými kapacitami každou nalezenou cestou zvýšíme tok alespoň o jednotku, takže program nebude běžet déle, než je součet všech kapacit. Ale to není moc uspokojivý odhad, protože záleží na ohodnocení.

Pokud budeme hledat cesty skutečně prohledáváním do šířky, bude počet kroků v $\mathcal{O}(nm^2)$, protože se dá ukázat, že se hrany, které při zlepšování cesty tvoří minimum, postupně vzdalují od zdroje. Pak máme $\mathcal{O}(m)$ času k nalezení cesty a m hran,

které se nejvýše n-krát mohou vzdálit. Že to tak skutečně je, je lehce zdlouhavé intelektuální cvičení (viz [IntroAlg]).

O vylepšení daného postupu a rozbor jednoho alternativního si můžete přečíst v [ADS2].

Cvičení

- Důležitou vlastností algoritmu je, že když dostane celočíselné kapacity, vrátí celočíselný tok. Bude se nám to hodit v aplikacích. Umíte to dokázat?
- Rozdíl mezi Fordem-Fulkersonem, který hledá cesty obecným způsobem, a takovým, který to dělá prohledáváním do šířky, je ze složitostního hlediska docela velký, a proto se tomu druhému občas říká Edmondsův-Karpův. Najděte malý graf a nevhodnou posloupnost cest, která způsobí, že F-F poběží skutečně v závislosti na velikosti kapacit.
- Můžete dokonce zkusit využít zlatého řezu k nalezení grafu s reálnými kapacitami, na kterém F-F pro danou (nešikovnou) posloupnost cest nikdy neskončí.
- Skončí algoritmus v konečném čase, jsou-li kapacity čísla racionální?

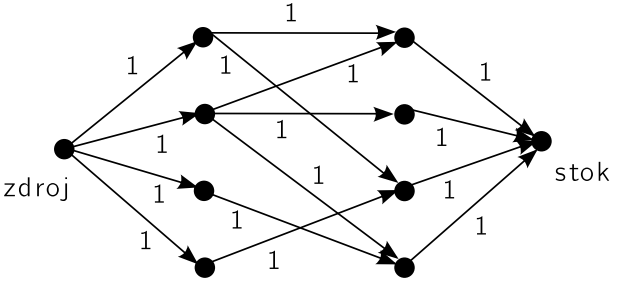
Užití

Párování v bipartitních grafech

Máme-li za úkol najít na plese co nejvíce tanečnicím tanečníka, kterého znají, stojíme před zásadním a nelehkým úkolem.

K tomu se nám bude hodit znát *bipartitní graf*, v němž jsou vrcholy rozděleny na dvě skupiny (mohou být i prázdné) a hrany vedou jen mezi těmito skupinami. Pokud jsou tedy vrcholy u, v ve stejné skupině, nikdy mezi nimi nevede hrana, jinak tam být může, ale nemusí. Skupinám vrcholů se někdy říká *partity*.

Na základě známosti postavíme bipartitní graf mezi partitou tanečníků a partitou tanečnic, přidáme zdroj za kluky a stok za holky. Oba nové vrcholy k nim připojíme hranami s jednotkovou kapacitou, hranám v bipartitním grafu také nastavíme jednotkové kapacity a nakonec všechno zorientujeme směrem do stoku.



Maximální celočíselný tok, který na tomto grafu získáme, nám hrany bipartitního grafu rozdělí na nevybrané s tokem 0 a vybrané s tokem 1. Můžou vybrané hrany sdílet tanečníka? Těžko, když do něj teče nejvýše jednotkový tok a musí platit Kirchhoffův zákon. Podobně s tanečnicemi.

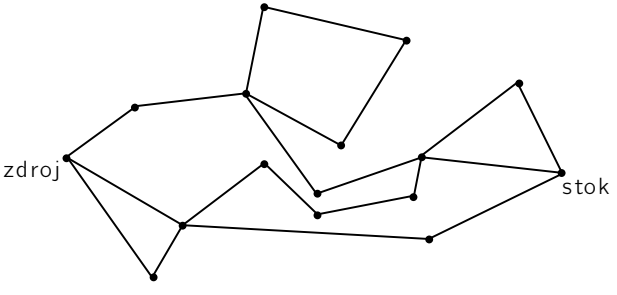
Vybrané hrany nám proto vytvoří párování. A protože jsme našli maximální tok, jde o párování největší. Kdyby existovalo párování větší, dokázali bychom podle něj zvětšit tok.

Hledání hranově a vrcholově disjunktních cest

Chceme-li se v grafu G dostat z vrcholu u do vrcholu v, může nás zajímat (třeba kvůli spolehlivosti, s jakou se umíme dopravit do cíle), kolik mezi nimi existuje cest, které nesdílí hrany, nebo nesdílí vrcholy. (Druhá podmínka je silnější. Když dvě cesty nesdílí vrcholy, nesdílí hrany.)

Oba tyto problémy lze převést na hledání maximálního toku. V obou případech nastavíme u jako zdroj a v jako stok. V prvním případě nastavíme jednotkové kapacity všem hranám, v druhém navíc všem vrcholům (kapacitu vrcholu a přidělíme tak, že ho rozdělíme na dva vrcholy b a c, do b povedou všechny hrany vedoucí do a, z c budou vycházet všechny hrany původně vycházející z a a z b do c přidáme hranu o kapacitě vrcholu).

Ford-Fulkerson nastaví některým hranám jednotkový tok, některým nulový. Nulové nyní z grafu vyhodíme. Pokud jsme hledali hranově disjunktní cesty, můžeme nyní získat třeba takovýto graf:



Jak z něj vykřesat kýžený výsledek? Začneme procházet ze zdroje zbylé hrany. Vždy, když se dostaneme do vrcholu, ve kterém už jsme v tom samém průchodu byli, vyhodíme z grafu všechny hrany cyklu, který jsme tímto objevili. (Hodnota toku se tím nezmění.)

Průchodem grafu se vždy můžeme dostat až do stoku (všude jinde budeme moci podle Kirchhoffova zákona jít dál – dost to připomíná úvahu o eulerovských tazích) a protože jsme mezitím horlivě odstraňovali cykly, dostali jsme cestu. Vrátíme ji jako jeden výsledek, smažeme její hrany a pokud ještě tok není nulový, pokračujeme dál.

Počet cest je tedy velikost toku. Podle Mengerovy věty je navíc minimum počtu hranově/vrcholově disjunktních cest pro všechny dvojice vrcholů roven stupni hranové/vrcholové souvislosti grafu – pokud máme dost času zavolat tokový algoritmus pro každou dvojici vrcholů našeho grafu, získáváme tak použitelný (polynomiální) postup, jak vypočítat souvislost grafu (graf je hranově/vrcholově k-souvislý, když zůstane souvislý po odebrání libovolných $k-1$ hran/vrcholů).

Cvičení

- Úvaha nebyla naprosto přímočará kvůli cyklům v nalezeném toku. Říká se jim cirkulace. Je jasné, že v případě hledání hranově disjunktních cest vzniknout mohou. Co v případě vrcholově disjunktních, tedy v situaci, kdy jsme omezili tok vrcholy?
- Nepracuje náhodou Edmondsův-Karpův algoritmus rychleji, pokud je graf, jak jsme teď opakovaně viděli, ohodnocený toliko nulami a jedničkami?

Lukáš Lánský

Úloha 23-4-1: Studenti a profesoři

Studenti na Stanfordově univerzitě se chtějí prosadit a napsat co nejvíce článků, přičemž každý z nich má vytipováno několik profesorů, pod jejichž vedením by chtěl článek psát. S jinými profesory spolupracovat nechce a nebude.

Studenti jsou schopni psát maximálně K článků najednou. Leč čas profesorů je omezený, každý z nich je totiž ochoten spolupracovat maximálně s K studenty, přičemž je jim jedno, kteří to budou.

Vaším úkolem je najít algoritmus, který zjistí, jestli je možné, aby každý student psal právě K článků a každý profesor spolupracoval právě s K studenty, a pokud ano, tak vypsat, který student bude spolupracovat s kterým profesorem.

Můžete předpokládat, že profesorů i studentů je stejně, totiž N, a nemusíte uvažovat situaci, že by student chtěl psát u jednoho profesora více článků.

Příklady: pro vstup $N = 4$, $K = 2$, student S1 chce psát článek s profesory P1 a P2, student S2 s P1, P2, P3, P4, student S3 s P2, P3, P4 a student S4 s profesory P3, P4, jsou řešením tyto páry student–profesor: S1–P1, S1–P2, S2–P1, S2–P3, S3–P2, S3–P4, S4–P3, S4–P4.

Pro vstup $N = 5$, $K = 2$, studenti S1 a S2 chtějí psát u profesorů P1, P2, P3, student S3 u P3, P4, P5 a studenti S4, S5 u P4, P5, řešení neexistuje. Existovalo by, kdyby bylo $K = 1$, ale to už je zase jiný vstup.

Úloha 23-5-6: Limity a grafy

Dostanete na vstupu orientovaný graf s kladně celočíselně ohodnocenými hranami. Dále tam bude pro každý vrchol dvojice kýžených limitů – minimální součet vstupních hran a maximální součet výstupních hran.

Vaším úkolem je najít nové nezáporné celočíselné ohodnocení každé hrany, které nebude větší než to původní a které bude dohromady se všemi ostatními novými ohodnoceními respektovat dané limity.