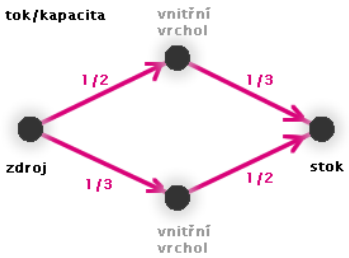
Toky v sítích

I. Úvod

Toky v sítích jakožto obor zkoumaný v rámci teorie grafů je vysoce prakticky využitelný. Zmíněné sítě totiž mohou být v reálném životě zastoupeny sítí silniční, železniční, vodovodní sítí a jakoukoliv jinou potrubní rozvodnou sítí, stejně tak rozvodnou energetickou sítí. Obecně bych to nazval distribučními sítěmi, kde dochází k nějakým pohybům mezi uzly, tedy řečeným tokům.

Takové sítě se výhodně reprezentují grafy (jakými a s kterými vlastnostmi viz II. Matematizace reálné situace), kde hrany budou intuitivně silnice, železnice a podobně, tedy vrcholy budou uzly, kde se schází více ramen sítě, například silniční křižovatka nebo velice konkrétní český železniční uzel Česká Třebová.



Obrázek č. 1

V těchto reálných situacích sítě nejčastěji podléhají zkoumání na maximální tok v síti. Zjednodušeně a názorně předvést, co je tím myšleno, je snadné. Například u vodovodní sítě je výhodné vyzkoumat, kolik nejvíce vody mohu z bodu A poslat do bodu B, aniž by byla síť přetížená nebo zbytečně nevyužitá.

II. Matematizace reálné situace

a) Síť

Síť je hranově ohodnocený orientovaný graf $G = (V, E)$, kde V je množina vrcholů rozdělených na zdroj, stok a vnitřní vrcholy a E je množina hran, které jsou ohodnoceny kapacitou a tokem. Daný graf G je navíc i symetrický, tedy pro každé dva vrcholy u, v z množiny vrcholů V platí, že hrana $uv \in E$ právě tehdy, když $vu \in E$.

b) Zdroj, stok a vnitřní vrcholy

Zdroj je vrchol z, ze kterého proudí tok do sítě, může jich být i více. Stok je naopak vrchol s, ve kterém se tok ze sítě setkává, opět jich může být více. Ostatní vrcholy mimo stoky a zdroje se nazývají vnitřní vrcholy.

c) Kapacita

Kapacita hrany je funkce $c: E \rightarrow R_0^+$, to znamená, že každá hrana je ohodnocena nezáporným reálným číslem. Takové ohodnocení získáváme společně s grafem. Pro převedení na reálnou situaci si můžeme pod kapacitou hrany představit u potrubní sítě maximální průtok nebo u datové sítě propustnost dat.

d) Tok

Tok v hraně je funkce $f: E \rightarrow R_0^+$, tedy i v tomto případě je každá hrana ohodnocena nezáporným reálným číslem. Zde se na rozdíl od kapacity více využívá definicí povolený nulový tok. Tok je v daném čase skutečné množství aut na silnici nebo skutečné množství ropy v ropovodu, a to nula bývá [1.][2.].

Tok nadále musí jakožto funkce splňovat několik podmínek. Pro každou hranu grafu G musí platit, že tok je menší nebo roven kapacitě hrany ($\forall e \in E : f(e) \leq c(e)$). Tok musí být v souladu s Prvním Kirchhoffovým zákonem ve znění „Součet vstupních toků do vnitřního vrcholu se rovná součtu výstupních toků z vnitřního vrcholu.“ [3.]. Vyloučeny jsou z téhle podmínky pochopitelně zdroje a stoky, pro které platí poupravené znění.

Pro zdroj musí tok splňovat, že vstupní tok je menší nebo roven výstupnímu toku a pro stok je to naopak, tedy že vstupní tok musí být větší nebo roven výstupnímu toku.

V případě toku lze také definovat velikost toku (míněno pro celý graf) jako součet toků ze zdroje. Neboť tok je ohodnocení hrany vznikající až v reálných procesech v kapacitně ohodnocené síti, snažíme se ho maximalizovat v souladu s kapacitou a definicí toku, především Kirchhoffovým zákonem. Takto dostáváme problém hledání maximálního toku v síti [1.] [2.].

e) Rezerva hrany

Rezerva hrany udává zbývající kapacitu hrany, tedy jaký tok ještě můžeme dodat k současnému toku, aniž bychom překročili kapacitu. Pracuje rovněž s tokem v opačném směru, tedy platí $r(uv) = c(uv) - f(uv) + f(vu)$, kde $r(uv)$ je rezerva hrany uv a vu je hrana opačná k uv, což nám umožňuje zadaná symetrie sítě [1.].

III. Řešení problému maximálního toku

a) Fordův – Fulkersonův algoritmus

Fordův – Fulkersonův algoritmus pracuje se sítí *G* tak, že zkouší všechny cesty ze zdroje do stoku a postupně zlepšuje tok na hranách těchto cest za využití rezervy hrany. V grafu s více zdroji a stoky lze využít následujícího zjednodušení. Zvolíme pomocný zdroj a stok, které hranami připojíme k původním zdrojům a stokům. Těmto hranám dáme kapacitní ohodnocení nekonečno. Na takový graf použijeme Fordův – Fulkersonův algoritmus a z výstupu poté pouze odebereme přidaný pomocný zdroj a stok [2.][4.].

Algoritmus obvykle začínáme s nastavenými nulovými toky na hranách, ale obecně lze mít na hranách na počátku algoritmu libovolný tok. Algoritmus probíhá, dokud ze zdroje do stoku lze najít cestu, kde každá hrana má nenulovou rezervu, tedy že na této cestě jde vylepšit (zvýšit) tok po celé její délce. V každé iteraci tedy cestu s nenulovou rezervou vybereme a zkoumáme ji [1.].

Ale jak takovou cestu vybrat? Musíme začít od zdroje. Všechny vrcholy v síti si označíme za nenavštívené. Budeme sledovat orientaci budované cesty pomocí +, - kvůli záznamu cesty P. Uložíme si i rezervy zkoumaných hran, to se bude hodit o něco později. Budeme rovněž postupně otevírat a zavírat vrcholy, jak jimi budeme procházet. Zdroj otevřeme a pak v cyklu, dokud bude existovat nějaký otevřený vrchol nebo zkoumaný vrchol nebude stok, budeme zkoumat nejdříve následníky a poté předchůdce zkoumaného uzlu. Nejprve však zavřeme zkoumaný uzel. Následníky zkoumáme tím způsobem, že pokud je následník nenavštíven a tok na spojující hraně je menší než kapacita hrany, tak ho otevřeme a zapíšeme orientaci hrany +. Předchůdce zkoumáme odlišným způsobem. Otevřeme takový vrchol, který nenavštíven a tok z uzlu do předchůdce je nenulový. Orientaci hrany označíme -. Pokud cyklus skončí a námi zkoumaný vrchol není stok, nešlo k němu dojít, neexistuje tudíž hledaná cesta P ze zdroje do stoku [5.].

Na každé nalezené cestě P následně zvolíme hranu s nejmenší rezervou a tuto rezervu si označme ε, kde $\varepsilon = min(r(uv), uv \in P)$. V algoritmu tedy pro každou cestu s nenulovou rezervou určíme ε a o tu zvýšíme tok na cestě, jde-li to

provést. Z toho je také patrné, proč hledáme na cestě nejmenší rezervu, abychom jí mohli zvednout tok na celé délce cesty *P*.

A další kroky musíme provést pro každou hranu *uv* na cestě *P* postupně. Zaprvé si pro hranu určíme hodnotu δ značící minimum z toku na opačné hraně *vu* a nejmenší rezervy ε na cestě *P*, $\delta = min(f(vu), \varepsilon)$. Tato hodnota nám pomáhá využít symetričnosti sítě a umožňuje nám použít opačnou hranu *vu* k úpravě našeho toku na hraně *uv* a nalezení maximálního toku, který primitivní algoritmus na postupné procházení všech cest bez znalosti pomocných hodnot ε a δ nemusí odhalit. Je třeba zdůraznit, že v orientovaném grafu bereme opačnou hranu *vu* tak, že má nulovou kapacitu i nulový tok. Ale podle definice rezervy hrany je dána rezerva *vu* tokem hrany *uv*. Hodnota δ nám tedy říká, o kolik snížit tok proti směru hrany, který podle symetričnosti sítě využít lze, ale mnohdy jen teoreticky, neboť máme většinou určenou orientaci hran jedním směrem.

Dále je třeba každé hraně na zkoumané cestě upravit tok a to v obou směrech. Tok hrany *vu* upravíme podle vztahu $f(vu) = f(vu) - \delta$ a tok hrany *uv* upravíme podle vztahu $f(uv) = f(uv) + \varepsilon - \delta$. V některých grafech nemusíme ani opačné hrany využívat, pak δ zůstává nulová a *f(uv)* upravujeme pouze o nejmenší rezervu na cestě *P*. V jiných grafech ale můžeme za využití opačné hrany najít cestu *P*, která má všechny hrany s nenulovou rezervou, a tudíž můžeme její tok zvýšit a upravit tak i dosud maximální tok této sítě.

Když není možnost zkoumat žádnou cestu *P*, neexistuje tedy další cesta s nenulovou rezervou na všech jejích hranách, nemůžeme žádné cestě navýšit tok a nemůžeme navýšit ani dosud maximální tok. Takový tok považujeme za maximální. Pro doplnění, myslíme tím součet toků ze zdroje [1.].

Ukázka pseudokódu Ford – Fulkersonova algoritmus

```
Deklarace[7.]  c[1..N, 1..N]  // matice kapacit
               f[1..N, 1..N]  // matice toků
               r[1..N, 1..N]  // matice rezerv
               E[1..N, 1..x]  // matice sousedů
               Akt[1..N]  // stav vrcholů

f := 0  na každé hraně sítě G
dokud existuje cesta P s nenulovou rezervou na každé hraně proveď
      ε := nejmenší rezerva na hraně cesty P
      f := f + ε  na každé hraně cesty P
      pro každou hranu cesty P proveď
            δ := min(f[v][u], ε)
            f[v][u] := f[v][u] - δ
            f[u][v] := f[u][v] + ε - δ
maximální tok f_max := součet toků na hranách ze zdroje


nalezení cesty P s hranami s nenulovou rezervou
Akt [u] := „nenavštíven“ pro všechny vrcholy sítě G
Akt [z] := „otevřen“ //otevřeli jsme zdroj
Opakuj Akt [u] := „zavřen“ //u je libovolný otevřený vrchol, který jsme
začali zkoumat
      Pro následníky v uzlu u
            Pokud Akt [v] := „nenavštíven“ a f[u][v] < c[u][v] proveď
                  Akt [v] := „otevřen“
                  Zapíšeme orientaci + a rezervu r[u][v]
      Pro předchůdce v uzlu u
            Pokud Akt [v] := „nenavštíven“ a f[v][u] > 0 proveď
                  Akt [v] := „otevřen“
                  Zapíšeme orientaci - a rezervu r[v][u]
Dokud existuje otevřený uzel u nebo zkoumaný uzel není stok[5.]
```

b) Edmondsův – Karpův algoritmus

Tento algoritmus je implementací Ford – Fulkersonova algoritmus, ale narozdíl od následujícího Dinitzova algoritmu nemění princip, ale zachovává postup, jen upravuje krok výběru cesty ze zdroje do stoku se všemi hranami s nenulovou rezervou. Nevybírá je totiž náhodně, ale bere takovou cestu co nejkratší. Provedeme to procházením sítě do šířky podobně jako v předchozím algoritmu. Budeme zkoumat následníky a předchůdce, zaznamenávat orientaci a také vzdálenost od zdroje.

Pomocí toho, ač je asymptoticky časově pomalejší než Dinitzův a Goldbergův algoritmus, je praktičtější a rychlejší pro řídké sítě [6.] [7.].

c) Dinitzův algoritmus

Dinitzův algoritmus je implementací Ford – Fulkersonova algoritmu stejně tak jako předchozí algoritmus, ale ve vylepšení asymptotické časové složitosti zachází ještě dále. Zatímco ve Ford – Fulkersonově algoritmu jsme vylepšovali tok pomocí cest ze zdroje ke stoku s hranami s nenulovou rezervou, zde budeme maximální tok vylepšovat vhodným tokem *g*.

Pro potřeby tohoto algoritmu si musíme objasnit některé nové pojmy:

i. Síť rezerv

Síť rezerv k toku *f*, kterým ohodnocujeme hrany v síti *G*, je síť *H*, tedy graf o množině vrcholů *V* včetně zdroje a stoku a o množině hran *E* ohodnocených rezervou hrany [8. – str. 1].

ii. Blokující tok

Tok *f* je blokující, jestliže pro každou orientovanou cestu *P* ze zdroje do stoku existuje $uv \in P$ taková, že $f(uv) = c(uv)$ [8. – str. 3].

Prvním krokem se shoduje s Ford – Fulkersonovým algoritmem, a to inicializací hran nulovým tokem, od kterého algoritmus započneme. Následně pro každý tok *f* v průběhu algoritmu (včetně nulového) budeme ve druhém kroku vytvářet síť rezerv *H* podle definice a v ní poté smažeme při prohledávání do šířky všechny hrany s nulovou rezervou.

Třetí krok je zásadní pro ukončování algoritmu. V promazané síti rezerv je třeba nastavit *l* jako délku nejkratší vzdálenosti ze zdroje do stoku (nejmenší počet hran na cestě ze zdroje do stoku). Pokud *l* nabude konkrétní hodnoty, algoritmus dále pokračuje [8. – str. 4]. Ve chvíli, kdy *l* nabude hodnoty nekonečno, neexistuje mezi zdrojem a stokem cesta. Je potřeba se zamyslet, proč tomu tak je. Protože v další fázi algoritmu budeme hledat zlepšující tok *g* pomocí rezerv, tak zpráva, že v sítí rezerv se nelze dostat po žádné cestě ze zdroje do stoku, říká, že při toku *f* máme v síti hrany s nulovou rezervou, a tudíž nelze *f* pomocí *g* vylepšit a je nutné *f* nastavit jako maximální tok a algoritmus končí [8. – str. 3].

Ale teď se zaměřme na pokračování algoritmu, kdy máme konkrétní hodnotu *l* a cesta ze zdroje do stoku existuje. Nyní je potřeba pročistit síť *H*. Nejdříve si vrcholy rozvrstvíme podle vzdálenosti od zdroje. Takto uspořádané vrcholy využijeme k čištění. Zaprvé síť pročistíme od vrstev vrcholů za stokem. Ty nám nebudou jistě k užitku, protože za stokem ve vzdálenosti větší než *l* bychom

blokující tok *g* v pročištěné síti *J* a zároveň zlepšující tok v síti *G* nehledali. Stejně tak pročistíme i zpětné hrany do předchozích vrstev (tedy ve směru od stoku do zdroje) a hrany ve vrstvách. Takovými hranami jistě nepovede nejkratší cesta délky *l*.

Nakonec vyčistíme síť i od vrcholů, ze kterých dál nevede již žádná cesta, tedy že mají stupeň pro orientované výstupní hrany nulový. Odstraňujeme je postupně, dokud nějaký takový vrchol máme. Když odstraňujeme takový vrchol, musíme kontrolovat, abychom odstranili všechny hrany vedoucí do takového vrcholu, a zda vrchol, ze kterého vedla odstraňovaná hrana, nemá po našem zásahu nulový stupeň pro výstupní hrany. Pokud ano, musíme s ním počítat pro smazání ze sítě stejných způsobem [8. – str. 4].

V takto pročištěné síti *J* hledáme blokující tok *g*, o který zvýšíme tok *f* v síti G. Opět nastavíme tok *g* jako nulový. A dokud existuje orientovaná cesta *P* ze zdroje do stoku, tedy dokud nemáme blokující tok, tak opakujeme následující kroky. Musíme projít postupně všechny takové cesty. V každé cestě potřebujeme určit hodnotu ε, tedy minimum výrazu *r(e) – g(e)* ze všech hran cesty. Zmíněný výraz je v síti *J* kapacita hrany zmenšená o tok na hraně, tedy zjistili jsme nejmenší rezervu hrany na celé cestě. Logicky o ni zvýšíme tok na všech hranách cesty *P*.

Teď je potřeba zkontrolovat, zdali nějaká hrana svým novým tokem nedosáhla své kapacity. Pokud dosáhla, smažeme ji a nově upravíme a pročistíme síť *J* podle vzoru čištění, které už bylo v předchozí části algoritmu. Ve chvíli, kdy už po přerovnání sítě *J* nebude existovat orientovaná cesta *P*, je cesta mezi zdrojem a stokem blokována. V tu chvíli nastavený tok *g* označíme za blokující a zlepšíme pomocí něj tok *f* v původní síti *G*.

Pak opět pokračujeme krokem s tvorbou sítě rezerv *H* pro nový tok *f* a pokračujeme skrze tento cyklus, dokud (jak už bylo řečeno) nedojdeme ke zjištění, že nejkratší délka *l* ze zdroje do stoku v síti rezerv je nekonečno, tedy že neexistuje, a neoznačíme *f* za maximální tok v síti *G* [8. – str. 5].

Ukázka pseudokódu Dinicova Algoritmu

```
Deklarace c, r, f, E podle deklarace ve Ford – Fulkersonově algoritmu
          r je zároveň matice kapacit pro síť H, J
          E‘ je matice sousedů pro síť H
          E‘‘ je matice sousednosti pro síť J
          d[1..N]  // pole vzdáleností vrcholu od zdroje
          g[1..N, 1..N]  // matice toků v síti J

f := 0  na každé hraně sítě G
dokud není cyklus zrušen dodatečnou podmínkou o délce l proveď
      ze sítě G vytvoř síť rezerv H
      prohledej do šířky síť rezerv H
            pokud r[u][v] = 0 proveď
                  smaž uv
      prohledej do šířky promazanou síť rezerv H a
            d[v] := vzdálenost zkoumaného uzlu v od z
            pokud zkoumaný uzel = stok proveď
                  l := nejkratší cesta ze z do s, nejméně hran
      pokud skončilo prohledávání do šířky a zkoumaný uzel <> stok proveď
            l := ∞
      pokud l = ∞ proveď
            konec cyklu // v síti rezerv není cesta ze z do s, nelze
vylepšit maximální tok
      pročistíme síť rezerv H a vytvoříme pročištěnou síť J
            uspořádat vrcholy podle vzdálenosti od z
            pokud d[v] > l proveď  // vrcholy ve vrstvách za stokem
                  smaž v
            pokud d[v] < d[u] proveď  // zpětné hrany
                  smaž uv
            pokud d[v] = d[u] proveď  // hrany v jedné vrstvě
                  smaž uv
            pomocí fronty odstraníme vrcholy v: deg+(v) = 0 (stupeň pro
hrany vedoucí ven)
                  zařaď do fronty vrcholy v: deg+(v) = 0
                  dokud není prázdná fronta proveď
                        odeber v z fronty
                        smaž v
                        smaž všechny f[u][v]
                        pokud deg+(u) = 0 proveď
                              přidej u do fronty
      g := 0  na každé hraně sítě J // hledáme blokující tok g
      dokud existuje orientovaná cesta P ze z do s proveď // prohledáním
sítě do hloubky
            ε := min(r[u][v] – g[u][v], ze všech hran v cestě P)
            Pro každou hranu uv
                  g := g + ε
                  pokud r = g proveď
                        smaž hranu
                        vyčisti síť J  // podle vzoru čištění sítě rezerv H
      g := blokující tok
      f := f + g
maximální tok fmax := součet toků na hranách ze zdroje
```

d) Goldbergův algoritmus

Poslední zmiňovaný algoritmus využívá ještě jiného principu než modifikace Ford – Fulkersonova algoritmu a Dinitzův algoritmus. Základní verze Goldbergova algoritmu je stejně asymptoticky časově složitá jako Dinitzův algoritmus ($O(MN^2)$), kde M je počet hran a N je počet vrcholů. Nejvýraznější odlišení od předchozích algoritmů je ohodnocení hran tzv. vlnou, která se jeví nadsazeně oproti skutečnému toku a zmenšováním ho upravujeme na skutečný maximální tok. I zde ale musíme ještě představit nové definice pro pochopení tohoto algoritmu:

i) Přebytek

Máme-li síť G s tokem f a nějakým vrcholem v z množiny vrcholů V sítě G, pak $f^{\Delta}(v)$ nazýváme přebytek ve vrcholu v a je definován jako rozdíl součtu všech toků přitékajících do v a součtu toků odtékajících z v.

ii) Vlna

Funkce $f: E \rightarrow R^{+}_{0}$ je vlnou v síti G, pokud daná funkce splňuje dvě podmínky. Zaprvé že vlna na žádné hraně nepřekročí kapacitu hrany ($\forall e \in E : f(e) \leq c(e)$). Zadruhé že přebytek v každém vnitřním vrcholu je nezáporný ($\forall v \in V \setminus \{z, s\} : f^{\Delta}(v) \geq 0$).

iii) Výška vrcholu

Funkce $h: V \rightarrow N$ se nazývá výška vrcholu a přiřazuje podle algoritmu přirozená čísla vrcholům. Pohyb přebytku umožňuje pouze z vrcholu s větší výškou do vrcholu s nižší výškou. To aby se nám algoritmus nezacyklil [9. – str. 1].

A teď k samotnému algoritmu. Nejdříve musíme inicializovat některé hodnoty. U všech vrcholů nastavíme výšku 0, ale zdroj ohodnotíme počtem vrcholů v síti, abychom zajistili, že bude vždy nejvýše. Stejně tak u všech hran nastavíme nulovou vlnu, ale pro všechny hrany ze zdroje nastavíme vlnu pro danou hranu jako nejvyšší možnou, tedy rovnou hodnotě kapacity hrany. Dále pak tuto vlnu povedeme skrz celou síť a budeme se snažit ustálit toky na hranách.

Následující část algoritmu probíhá v cyklu. Dokud existuje vnitřní vrchol u, který má nenulový přebytek, cyklus probíhá. Zjednodušeně, ve chvíli, kdy mají všechny vnitřní vrcholy nulový přebytek, jsou vlny v síti ustáleny, nemáme kam co

přesouvat a v závislosti na tom upravovat rezervy na hranách a tok v síti, tudíž součet toků ze zdroje pak tvoří maximální tok [9. – str. 2].

Ale pokud existuje popsaný vrchol *u*, pak zkoumáme, zda existuje vrchol *v* takový, že hrana *uv* má nenulovou rezervu a výška *h(u) > h(v)*, tedy splňuje podmínku pro přesun přebytku. Pak ho přesuneme z *u* do *v*. Jak to uděláme? Určíme tok δ jako minimum z přebytku vrcholu *u* a rezervy hrany *uv* (pokud je dostatečná rezerva hrany, pošleme celý přebytek, jinak nám nezbývá než jen využít co nejvíc dané rezervy). Pak tento určený tok převedeme, jak jsem zmiňoval. Tím také upravíme přebytky na obou vrcholech. Na *u* klesne o hodnotu δ a na *v* vzroste o hodnotu δ. Dále upravíme rezervu hrany *uv*. Samozřejmě ji snížíme o hodnotu δ, tedy zvýšíme tím tok na dané hraně. Těmito úpravami se právě nastavují a ustalují hodnoty toků na hranách v síti *G* [9. – str. 1]. Lze v tomto cyklu přidat i zrychlující kritérium pro výběr vrcholu s přebytkem. Nebudeme ho vybírat náhodně, ale budeme vždy zkoumat v daném cyklu ten vrchol *u* s přebytkem, který má největší výšku *h(u)*. Pak můžeme dosáhnout asymptotické časové složitosti $O(N^3)$ [9. – str. 7].

Mohlo by se ale stát, že najdeme vrchol s deklarovaným přebytkem, který ale nemůžeme do žádného vrcholu *v* poslat. Pak musíme jednu z podmínek upravit. Do rezervy hran zasahovat nebudeme, budeme upravovat výšku vrcholů. Když tedy máme vrchol *u* s přebytkem, který nemáme kam přesunout, jednoduše zvýšíme výšku *h(u)* o 1.

Až dojdeme do chvíle, kdy žádný vnitřní bod nemá nenulový přebytek, nemáme co přesouvat a upravovat a tok *f* jako součet toků ze zdroje nastavíme jako kýžený maximální tok [9. – str. 2].

Ukázka pseudokódu Goldbergova algoritmu

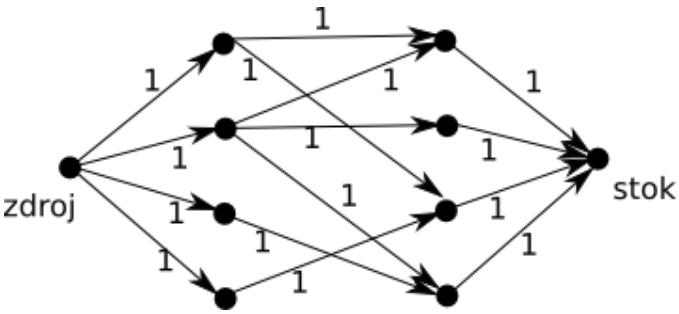
```
Deklarace c, r, f, E podle deklarace ve Ford – Fulkersonově algoritmu
          h[1..N]  // pole výšek vrcholů
          f Δ[1..N]  // pole přebytků ve vrcholech

h := 0 pro všechny vrcholy sítě G
h[z] := N
f := 0 pro všechny hrany sítě G
f[z][u] := c[z][u] pro všechny hrany ze zdroje
dokud existuje vnitřní vrchol u: f Δ[u] > 0 proveď
      pokud existuje v: r[u][v] > 0 a h[u] > h[v] proveď
```

```
            δ := min(f Δ[u], r[u][v])
            f Δ[u] := f Δ[u] - δ
            f Δ[v] := f Δ[v] + δ  // přesuneme přebytek z u do v
            r[u][v] := r[u][v] - δ
            f[u][v] := f[u][v] + δ
      jinak h[u] := h[u] + 1
maximální tok f_max := součet toků na hranách ze zdroje
```

IV. Aplikace v praxi

Jak už jsem v úvodu zmiňoval, teorie toků v sítích se dá převést na jakoukoliv přenosovou síť, od obecně dopravní sítě, přes datovou a elektrickou, až po rozvodné potrubní sítě. Pomocí popsaných algoritmů dokážeme najít maximální tok v dané síti na základě znalosti kapacity hran i případné počáteční situace toků v síti. V části o Ford – Fulkersonově algoritmu jsem popsal, jak se vypořádat i s více zdroji a stoky. Můžeme tak snadno určit, kolik ropy lze vypustit z rafinérie do sítě k dalším uzlům, aby nedošlo k poškození ropovodů. Stejně tak lze určit, kolik maximálně dat lze přenést z jednoho počítače přes dostupnou síť do druhého. Také dokážeme ověřit, jestli kapacitně dostačuje silniční síť mezi dvěma určenými městy při očekávané hustotě dopravy.



Obrázek č. 2

Mimo tyto očekávané a intuitivní aplikace teorie toků v síti a hledání maximálního toku je dobré zmínit i jeden ne úplně zjevný. A to hledání maximálního párování v bipartitním grafu. Pro připomenutí, co je bipartitní graf, jen krátce shrnu, že bipartitní graf je takový graf, jehož množinu vrcholů lze rozdělit na dvě množiny, kdy žádné dva vrcholy z jedné množiny nejsou spojené hranou. A párování v takovém grafu je podmnožina hran taková, že žádné dvě hrany takového párování nesdílejí vrchol [10. – str. 14].

A nalezení maximálního párování provedeme následovně. Bipartitní graf upravíme na síť s jedním zdrojem a stokem. Za jednu množinu vrcholů v grafu vložíme zmíněný zdroj, za druhou množinu vrcholů v grafu vložíme zase stok. Zdroj a stok spojíme se všemi vrcholy příslušných množin [4.]. Všem hranám dáme kapacitu 1 (to vše viz obrázek č. 2). A všechny hrany zorientujeme od zdroje ke stoku. Pak na takovou síť vypustíme algoritmus na hledání maximálního toku. Když najdeme maximální celočíselný tok v síti, do párování vložíme ty

hrany, které mají jednotkový tok. Ty s nulovým tokem vynecháme. Je zřejmé, že hrany vybrané do párování nesdílejí vrcholy, neboť ze zdroje přitéká nejvýše jednotkový tok a do stoku odtéká nejvýše jednotkový tok. To podle Kirchhoffova zákonu ukazuje na to, že tyto dva vrcholy jsou pouze na jedné hraně s jednotkovým tokem vybrané do párování a na žádné jiné [4.] [10. – str. 14].

V. Zdroje

1) https://teorie-grafu.cz/vybrane-problemy/toky-v-sitich.php – obrázek č.1
2) https://cs.wikipedia.org/wiki/Tok_v_s%C3%ADti
3) https://cs.wikipedia.org/wiki/Kirchhoffovy_z%C3%A1kony#Prvn%C3%AD_Kirchhoff%C5%AFv_z%C3%A1kon_(o_proudech,_o_uzlech)
4) https://ksp.mff.cuni.cz/kucharky/toky-v-sitich/ – obrázek č. 2
5) https://cs.wikipedia.org/wiki/Ford%C5%AFv%E2%80%93Fulkerson%C5%AFv_algoritmus
6) https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=19941
7) https://cs.wikipedia.org/wiki/Edmonds%C5%AFv%E2%80%93Karp%C5%AFv_algoritmus
8) http://mj.ucw.cz/vyuka/0910/ads2/2-dinic.pdf
9) http://mj.ucw.cz/vyuka/0910/ads2/3-goldberg.pdf
10) https://is.muni.cz/el/1433/podzim2007/MA010/um/Grafy-lect--6.pdf