# Toky v sítích
# (algoritmy Ford-Fulkerson a Dinic)

Prof. RNDr. Luděk Kučera, DrSc.

12. ledna 2004

## 1 Definice sítě

Síť je čtveřice $(G, c, z, s)$, kde $G$ je orientovaný graf, $c$ je funkce, která každé orientovaná hraně $h$ grafu $G$ přiřazuje nezáporné reálná číslo $c(h)$, nazývané kapacitou hrany $h$ a $z$, $s$ jsou dva různé vrcholy grafu $G$, nazývané zdroj a spotřebič. V celé kapitole bude uvažována jedna pevná síť $(G, c, z, s)$.

Je-li $t$ funkce z množiny hran $G$ do reálných čísel, pak divergencí funkce $t$ ve vrcholu $v$, označení $\delta(t, v)$, je součet toků $t(h)$ hranami $h$ vycházejícími z vrcholu $v$ minus součet toků $t(h)$ hranami $h$ vcházejícími do $v$. Je-li $A$ množina vrcholů, pak $\delta(t, A)$ je součet $\delta(t, v)$ pro všechny vrcholy $v \in A$.

Takováto funkce $t$ se nazývá tok, jestliže
pro každou hranu $h$ platí $0 \leq t(h) \leq c(h)$ a
pro každý vrchol $v$, různý od zdroje a spotřebiče, platí $\delta(t, v) = 0$.

Velikostí toku se nazývá číslo $\delta(t, z)$. úkolem je pro danou síť nalézt tok největší možná velikosti.

## 2 Ford-Fulkersonův algoritmus

Nejstarší a nejjednodušší algoritmus pro určení největšího toku je Ford-Fulkersonův algoritmus, který pracuje takto (je-li $h = (u, v)$ orientovaná hrana, pak často místo $t(h), c(h)$ budeme psát $t(x, y), c(x, y)$):

**Ford-Fulkersonův algoritmus**

**begin**
pro každou hranu $h$ položíme $t(h) = 0$;
*loop:*
určíme prostou posloupnost $z = v_0, v_1, \ldots, v_k = s$ vrcholů takovou,
že pro každé $i = 1, \ldots, k$ platí:
buď $(v_{i-1}, v_i)$ je hrana a $t(v_{i-1}, v_i) < c(v_{i-1}, v_i)$
nebo $(v_i, v_{i-1})$ je hrana a $t(v_i, v_{i-1}) > 0$;
pokud takováto posloupnost neexistuje, výpočet končí a $t$ je maximální tok.

určíme největší $\Delta > 0$ takové, že pro každé $i = 1, \ldots, k$ platí
bud' $(v_{i-1}, v_i)$ je hrana a $\Delta \leq c(v_{i-1}, v_i) - t(v_{i-1}, v_i)$
nebo $(v_i, v_{i-1})$ je hrana a $\Delta \leq t(v_i, v_{i-1})$.
pro každé $i = 1, \ldots, k$ položíme
je-li $(v_{i-1}, v_i)$ je hrana, pak $t(v_{i-1}, v_i) := t(v_{i-1}, v_i) + \Delta$,
nebo je-li $(v_i, v_{i-1})$ hrana, pak $t(v_i, v_{i-1}) := t(v_i, v_{i-1}) - \Delta$.
**goto** *loop*;
**end**;

Rychlost Ford-Fulkersonova algoritmu není jeho nejsilnější stránkou; jde-li nám při výpočtu o rychlost, je vhodné volit některý algoritmus, který z něho byl odvozen, ale je rychlejší. Proto se dobou výpočtu nebudeme zabývat. Je dokonce možné ukázat, že pokud kapacity hran nejsou celá čísla, pak se může stát, že se výpočet nikdy nezastaví.

Jestliže jsou kapacity celá čísla, pak je zřejmé, že vsechny proměnné algoritmu mají kdykoliv v průběhu výpočtu (a tedy i po ukončení) celočíselné hodnoty, protože jsou odvozeny z kapacit hran (celočíselných) a čísla 0 operacemi *min* a +. Protože hodnota $\Delta$ je vždy kladná, je proto vždy alespoň 1 a tedy každá iterace cyklu *loop* zvýší velikost toku o 1. Z lemma, které bude záhy uvedeno, tedy plyne, že se výpočet zastaví nejpozději po tolika iteracích cyklu, kolik je velikost libovolného řezu v síti (napr. součet kapacit hran vycházejících ze zdroje (řez {zdroj}) a, jak bylo řečeno, výsledný tok je v takovém případě celočíselný.

Nejprve ukážeme, že $t$ je vždy na začátku i konci cyklu *loop* tokem. Pro nulový tok na začátku to evidentně platí. Změny $t$, prováděné na konci cyklu přičítáním nebo odčítáním $\Delta$ jsou voleny tak, že pro každé $i = 1, \ldots, k-1$ se $\delta(t, v_i)$ nezmění, tedy zůstane nulové (zde mohou nastat čtyři možnosti, podle toho, zda jsou hrany, představované dvojicemi $u_{i-1}, u_i$ a $u_i, u_{i+1}$ orientovány směrem k vrcholu $v_i$ nebo od něj). Vrcholy $v_0$ a $v_k$ jsou zdroj a spotřebič a u ostatních vrcholů cesty divergence zůstává nezměněna, tedy také nulová.

Ukážeme nyní, že tento algoritmus nalezne vždy největší možný tok. Slovem *řez* budeme označovat každou množinu $C$ vrcholů, která obsahuje zdroj a neobsahuje spotřebič. Velikost řezu $C$ je definována jako součet kapacit všech hran $h = (v, w)$ takových, že $v \in C$ a $w \notin C$.

**Lemma 1** *Je-li $t$ libovolný tok a $C$ libovolný řez, pak velikost toku $t$ je menší nebo rovna velikosti řezu $C$.*

Důkaz: Velikost toku $t$ je rovna $\delta(t, z)$ a

$$\delta(t, z) = \delta(t, C) = \delta^+(t, C) - \delta^-(t, C),$$

kde $\delta^+(t, C)$ je součet toků všemi hranami $h = (v, w)$ takovými, že $v \in C$ a $w \notin C$ a $\delta^-(t, C)$ je součet toků hranami $h = (w, v)$ takovými, že $w \notin C$ a $v \in C$. $\delta^+(t, C)$ je ale evidentně menší nebo rovno velikosti řezu $C$ a $\delta^-(t, C) \geq 0$. Z toho již lemma plyne. ♣

Nyní již dokážeme

**Věta 2** *Ford-Fulkersonův algoritmus najde vždy maximální tok v síti.*

Důkaz: Označme si jako $C$ množinu vrcholů, zahrnující $z$ a všechny vrcholy $v$ takové, že exituje posloupnost vrcholů $z = v_0, \ldots, v_k = v$ takových, že pro každé $i = 1, \ldots, k$
buď $(v_{i-1}, v_i)$ je hrana a $t(v_{i-1}, v_i) < c(v_{i-1}, v_i)$
nebo $(v_i, v_{i-1})$ je hrana a $t(v_i, v_{i-1}) > 0$.

V okamžiku, kdy výpočet končí, je $C$ řez, neboť spotřebič v něm není (a zdroj je v něm vždy).

Ukážeme, že v okamžiku ukončení výpočtu je velikost toku $t$ rovna velikosti řezu $C$. Nechť $v \in C$ a $z = v_0, \ldots, v_k = v$ je posloupnost, která to dokazuje a nechť $w \notin C$. Nechť nejprve $h = (v, w)$ je hrana. Pak kdyby bylo $t(v, w) < c(v, w)$, posloupnost $v_0, \ldots, v_k = v, w$ by dokazovala, že $w \in C$, spor. Je-li $(w, v)$ hrana, pak kdyby bylo $t(w, v) > 0$, pak by výše uvedená posloupnost také dokazovala, že $w \in C$, což je spor. Tedy je-li $(v, w)$ hrana, pak $t(v, w) = c(v, w)$ a je-li $(w, v)$ hrana, je $t(w, v) = 0$, z čehož plyne rovnost velikostí toku $t$ a řezu $C$.

Kdyby existoval tok větší velikosti než má $t$ na konci výpočtu, musel by mít větší velikost než řez $C$ na konci výpočtu, což je podle předchozího lemmatu nemožné. ♣

Dinicův algoritmus je vlastně vylepšení Ford-Fulkersonova algoritmu, kdy pro zlepšení toku se volí vždy cesta, která má nejmenší možný počet hran. Pro zjednodušení výkladu budeme předpokládat, že síť má následující vlastnost: je-li $(v, w)$ hranou sítě, pak $(w, v)$ je také její hranou. Toto omezení není podstatné: pokud v původní síti hrana $(w, v)$ nebyla, přidáme ji s nulovou kapacitou. Tok takovou hranou musí být nulový a proto toky v původní a nové síti jsou ve vzájemně jednoznačné korespondenci, která zachovává i velikost toku.

Pro hranu $h = (u, v)$ si definujeme její *rezervu* $r(h) = r(u, v)$ předpisem $r(u, v) = c(u, v) - t(u, v) + t(v, u)$. Zlepšující cesta podle Ford-Fulkersonova algoritmu je proto v rámci našich konvencí a definicí orientovaná cesta tvořená hranami s kladnou rezervou, jako $\Delta$ se volí minimum rezerv jejích hran.

Bez újmy na obecnosti lze předpokládat, že tok v síti je takový, že pro každou hranu $(u, v)$ je buď $t(u, v) = 0$ nebo $t(v, u) = 0$. Pokud by totiž tato čísla byla obě kladná, lze je snížit o $\min(t(u, v), t(v, u))$. Tim jedno z nich poklesne na nulu, přitom se divergence vrcholů sítě nezmění, takže tok zůstane tokem stejné velikosti.

Jestliže přijmeme tento předpoklad, pak tok hranou $(u, v)$ sítě lze jednoznačné určit ze znalosti její rezervy a kapacity: pokud $r(u, v) \leq c(u, v)$, pak to znamená, že $t(v, u) = 0$ a tedy $t(u, v) = c(u, v) - r(u, v)$. Pokud $r(u, v) \geq c(u, v)$, pak musí být $t(u, v) = 0$. V průběhu výpočtu se tedy můžeme zabývat pouze rezervami hran a toky dopočítat až na konci. Operace, kterou opakovaně provádí Ford-Fulkersonův algoritmus je: nalézt cestu ze zdroje do spotřebiče, tvořenou hranami s kladnou rezervou, nalézt minimum $\Delta$ rezerv hran této cesty, o $\Delta$ snížit rezervy hran této cesty a o $\Delta$ zvýšit rezervy hran k nim opačných.

Je totiž zřejmé, že $c(u, v) + c(v, u) = r(u, v) + r(v, u)$ a tedy se tento součet nemění, takže pokles rezervy hrany $(u, v)$ o $\Delta$ jde současně se vzrůstem rezervy

opačné hrany o $\Delta$.

Algoritmus, který bude vyložen dále, byl prakticky současně popsán Edmondsem a Karpem a nezávisle na nich Dinicem. Edmonds-Karpův algoritmus je Ford-Fulkersonův algoritmus, ve kterém se vždy volí cesta s nejmenším počtem hran. Dinicova varianta ale zahrnuje některá další zlepšení, která o jeden řád zlepšují odhad doby výpočtu v nejhoršim případě.

Jako *síť rezerv* budeme označovat síť, která má stejnou množinu vrcholů a stejný zdroj a spotřebič jako výchozí síť a obsahuje právě ty hrany výchozí sítě, které mají kladnou rezervu. Vzdálenost vrcholu $v$ od zdroje v síti rezerv bude znamenat nejmenší $k$ takové, že existuje cesta ze zdroje do vrcholu $v$ v síti rezerv, která obsahuje $k$ hran. Pokud taková cesta neexistuje, je tato vzdálenost $+\infty$. Vzdálenosti do všech vrcholů v síti rezerv se snadno určí prohledáním sítě rezerv do šířky.

V průběhu výpočtu budeme konstruovat další síť, která má také stejnou množinu vrcholů a stejný zdroj a spotřebič jako výchozí síť a kterou budeme nazývat *čistá síť*. Množina hran čisté sítě bude podmnožinou množiny hran sítě rezerv. Ponamenejme, že spotřebič je označován jako $s$.

**Dinicův algoritmus**

pro každou hranu $h$ položíme $t(h) := 0$ a $r := c(h)$;
*outer loop*
    polož množinu hran čisté sítě rovnou množině hran sítě rezerv;
    v síti rezerv najdi pro každý vrchol $v$ vzdálenost $L(v)$ od zdroje;
    pokud $L(s) = \infty$, **goto** *ukončení výpočtu*, jinak polož $L := L(s)$;
    pro každý vrchol $w$, pro nějž $w \neq s$ a přitom $L(w) \geq L$
        vynech všechny hrany začínající nebo končící ve $w$ z čisté sítě;
    vynech z čisté sítě všechny hrany $(v, w)$, pro které $L(v) \geq L(w)$;
    pro každý vrchol $W$ čisté sítě polož
        $d_{in}(w) :=$ počet hran čisté sítě, končících ve $w$;
        $d_{out}(w) :=$ počet hran čisté sítě, začínajících ve $w$;
        $Q_{in} :=$ množina vrcholů $v$, pro které $d_{in}(v) = 0$;
        $Q_{out} :=$ množina vrcholů $v$, pro které $d_{out}(v) = 0$;
    *inner loop*
        pročisti síť;
        nalezni cestu $C$ ze zdroje do spotřebiče v čisté síti;
        urči minimum $\Delta$ rezerv hran cesty $C$;
        pro každou hranu $h = (u, v)$ cesty $C$ proveď následující operace:
            sniž rezervu hrany $h$ o $\Delta$ a zvyš rezervu hrany $(v, u)$ o $\Delta$;
            pokud rezerva hrany $h$ poklesla na nulu, pak sniž $d_{out}(u)$ a $d_{in}(v)$ o 1;
            pokud $d_{out}(u)$ pokleslo na nulu, zařaď $u$ do $Q_{out}$;
            pokud $d_{in}(v)$ pokleslo na nulu, zařaď $v$ do $Q_{in}$;
        **goto** *inner loop*;
*ukončení výpočtu* z rezerv hran sítě vypočítej toky.

kde operace "pročisti síť se provádí následujícím způsobem:

dokud $Q_{in}$ je neprázdná, pak z ní vyjmi některý prvek $u$ a
pro všechny hrany $(u, v)$ vycházející z $u$ proveď následující:
    vynech $(u, v)$ z čisté sítě, sniž $d_{out}(u)$ a $d_{in}(v)$ o 1 a
    jestli $d_{in}(v)$ pokleslo na 0, pak zařaď $v$ do $Q_{in}$;
dokud $Q_{out}$ je neprázdná, pak z ní vyjmi některý prvek $v$ a
pro všechny hrany $(u, v)$ vstupující do $v$ proveď následující:
    vynech $(u, v)$ z čisté sítě, sniž $d_{out}(u)$ a $d_{in}(v)$ o 1 a
    jestli $d_{out}(u)$ pokleslo na 0, pak zařaď $u$ do $Q_{out}$;
jestliže v čisté síti nezbyly žádné hrany, **goto** *outer loop*;

Předně je možno ověřit, že operace s čistou sítí se provádějí tak, že vždy po ukončení operace pročistění sítě, pokud nenásleduje skok na *outer loop*, je čista síť představována těmi hranami sítě rezerv, přes které vede nějaká cesta ze zdroje do spotřebiče v síti rezerv, která obsahuje $L$ hran (t.j. nejmenší možný počet hran v daném okamžiku). Jediný netriviální krok v důkaze tohoto tvrzení je následující:

Při zpracování cesty $C$ se zvyšuje rezerva hran opačných k hranám cesty $C$ o $\Delta$. Pokud některá z těchto hran $h$ nebyla v síti rezerv (t.zn. pokud její rezerva byla rovna 0), pak se v síti rezerv po zpracování cesty $C$ ocitne. Tato hrana ale *není* přidána do čisté sítě, protože v daný okamžik neleží na žadné cestě ze zdroje do spotřebiče v síti rezerv, která by měla délku $L$ nebo menší:

uvažujme cestu $z = u_0, \ldots, u_L = s$ optimální délky $L$ ze zdroje do spotřebiče v síti rezerv a předpokládejme, že $h = (u_{i-1}, u_i)$. Je zřejmé, že nejkratší cesta z $u_{i-1}$ do spotřebiče v síti rezerv má v daný okamžik délku $L - i + 1$, protože jinak by složení této cesty a $u_{i-1}, \ldots, u_L = s$ dalo cestu ze zdroje do spotřebiče v síti rezerv, kratší než $L$, spor. Stejně tak nejkratší cesta ze zdroje do $u_i$ v síti rezerv má v uvažovaný okamžik délku alespoň $i$. Nejkratší cesta ze zdroje do spotřebiče, která v daný okamžik vede přes hranu opačnou k $(u_{i-1}, u_i)$, tedy přes hranu $(u_i, u_{i-1})$, má proto délku alespoň $(L - i + 1) + 1 + i = L + 2$.

Jelikož při provádění vnitřního cyklu algoritmu, daného opakovanými skoky na návěští *inner loop*, tedy nové cesty délky $L$ v síti rezerv nevznikají, ale naopak zpracováním cesty $C$ se alespoň jedna hrana cesty nasytí (t.j. její rezerva poklesne na nulu) a vypadá ze sítě rezerv a tedy cesta $C$ i případné další cesty vedoucí přes nasycenou hranu (hrany) přestávají být cestami v síti rezerv, dojde před skokem na návěští *outer loop* vnějšího cyklu algoritmu k tomu, že v síti rezerv nezůstane žadná cesta délky $L$ nebo menší.

Z toho plyne důležité tvrzení:

**Lemma 3** *Při každém skoku na* outer loop *je vzdálenost ze zdroje do spotřebiče v síti rezerv větší než při předchozím vstupu do tohoto místa algoritmu.*

Nazvěme *fází* výpočtu jeho část mezi dvěma po sobě následujícími vstupy do místa *outer loop*. Přímým důsledkem předchozího lemmatu je

**Věta 4** *V průběhu výpočtu se provede méně než $N$ fází, kde $N$ je počet vrcholů sítě.*

Důkaz: Vzdálenost ze zdroje do spotřebiče v síti rezerv je v každé fázi výpočtu alespoň o 1 vyšší než v předešlé fázi, ale je to číslo mezi 1 a N-1. ♣

Nyní dokážeme, že Dinicův algoritmus provede vždy nejvýše $O(N^2M)$ kroků, kde $N$ je počet vrcholů $M$ je počet hran sítě a to tím, že dokážeme, že jedna fáze trvá nejvýše $O(NM)$.

**Lemma 5** *Počet kroků výpočtu, provedených v jedné fázi ve všech operacích čistění sítě je* $O(M)$.

Důkaz: Je zřejmé, že celková doba strávená v rámci jedné fáze čistěním sítě je úměrná počtu hran, které byly v průběhu fáze při čistění vynechány z čisté sítě a ten je nejvýše roven počtu všech hran sítě. ♣

**Lemma 6** *Počet skoků na návěští* inner loop *během jedné fáze je nejvýše* $M$.

Důkaz: Při každé iterací vnitřního cyklu se zpracuje jedna cesta a alespoň jedna hrana (ta, jejíž rezerva je $\Delta$) z čisté sítě vypadne a v průběhu uvažované fáze se již zpátky nedostane. ♣

**Lemma 7** *Jedno nalezení a zpracování cesty ze zdroje do spotřebiče v čisté síti trvá nejvýše* $O(N)$ *kroků.*

Důkaz: Jelikož každá hrana v čisté síti leží na cestě ze zdroje do spotřebiče délky $L$, můžeme ze zdroje postupovat, vybírajíce si vždy libovolnou hranu čisté sítě, vycházející z právě dosaženého vrcholu a vždy dojdeme do spotřebiče a to cestou délky $L$. Zpracování cesty pak trvá čas úměrný její délce. Přitom $L \leq N$. ♣