Kombinatorika a grafy I — 6. cvičení*

21. listopadu 2022

1 Projektivní roviny

Příklad 1. *(a) Ukažte, že existuje graf s N vrcholy a s aspoň $\Omega(N^{3/2})$ hranami, který neobdsahuje $K_{2,2}$ jako podgraf.*

(b) Bonusový příklad (*): ukažte, že každý takový graf má nanejvýš $O(N^{3/2})$ hran. Hint: dvěma způsoby spočítejte počet kopií $K_{1,2}$.

Řešení. (a) řešením je incidenční graf konečné projektivní roviny řádu n. Jedná se o bipartitní graf s prtitami velikosti n^2+n+1, kde jednu partitu ztotožníme s X, druhou s $\mathcal{L}$ a položíme hranu (x, L) právě tehdy, když bod $x \in X$ leží na přímce $L \in \mathcal{L}$. Potom je počet hran roven $(n^2+n+1)(n+1)$, protože každým bodem prochází $n+1$ přímek, a počet vrcholů je $N := 2(n^2+n+1)$. Máme $(n+1)(n^2+n+1) \geq (n^2+n+1)^{3/2} = (N/2)^{3/2} \sim 0.35N^{3/2}$. Pokud by v grafu byl podgraf $K_{2,2}$, tak by odpovídal dvojici přímek, které sdílí dvojici bodů.

(b) Mějme graf $G = (V, E)$ bez $K_{2,2}$. Spočítáme dvěma způsoby počet M dvojic $(v, \{u, u'\})$, kde $\{u, u', v\} \in \binom{V}{3}$ a $\{v, u\}, \{v, u'\} \in E$. Pro fixní u, u' existuje nanejvýš jedno takové v (jinak $K_{2,2}$) a tedy $M \leq \binom{N}{2}$. Na druhou stranu každé fixní v přispívá $\binom{deg_G(v)}{2}$ páry $\{u, u'\}$. Dohromady máme $\sum_{v\in V} \binom{deg_G(v)}{2} \leq \binom{N}{2}$. Můžeme předpokládat, že všechny stupně jsou aspoň jedna, protože nemá smysl přidávat izolované vrcholy. Potom $\binom{deg_G(v)}{2} \geq (deg_G(v)-1)^2/2$ a $\sum_{v\in V}(deg_G(v)-1)^2 \leq n^2$. Z Cauchy-Schwartzovy nerovnosti

$$\sum_{i=1}^{N} x_i y_i \leq \sqrt{\sum_{i=1}^{n} y_i^2}\sqrt{\sum_{i=1}^{n} x_i^2}$$

pro $x_v = deg_G(v) - 1$ a $y_v = 1$ máme

$$\sum_{v\in V}(deg_G(v)-1)\cdot 1 \leq \sqrt{\sum_{v\in V}(deg_G(v)-1)^2}\sqrt{N} \leq N^{3/2}.$$

V poslední nerovnosti jsme použili $\sum_{v\in V}(deg_G(v)-1)^2 \leq \sum_{v\in V}\binom{deg_G(v)}{2} \leq \binom{N}{2}$. Protože $|E| = \frac{1}{2}\sum_{v\in V} deg_G(v)$, tak $|E| \leq \frac{1}{2}(N^{3/2}+N)$.

□

2 Latinské čtverce

Latinský čtverec řádu n je tabulka o rozměrech $n \times n$ nad prvky z $\{1, 2, \ldots, n\}$, kde se každé číslo v každém sloupci a řádku vyskytuje právě jednou. Dva latinské čtverce L a L' jsou *ortogonální*, pokud pro každou uspořádanou dvojici čísel (a, b) z $\{1, 2, \ldots, n\}$ existuje $i, j \in \{1, 2, \ldots, n\}$ takové, že $(a, b) = ((L)_{ij}, (L')_{ij})$. Množina $\{L_1, L_2, \ldots, L_t\}$ Latinských čtverců je množinou *navzájem ortogonálních Latinských čtverců*, pokud každé dva čtverce z této množiny jsou ortogonální. Tři navzájem ortogonální latinské čtverce řádu 4:

1	2	3	4
4	3	2	1
3	4	1	2
2	1	4	3

1	2	3	4
2	1	4	3
4	3	2	1
3	4	1	2

1	2	3	4
3	4	1	2
2	1	4	3
4	3	2	1

Příklad 2. *Rozhodněte, zda permutace řádků a sloupců zachovávají vlastnost být latinským čtvercem a zda tyto operace zachovávájí ortogonalitu (operace je provedena na jednom z čtverců, pro ortogonalitu).*

*Informace o cvičení naleznete na http://kam.mff.cuni.cz/~fila/

Řešení. Permutace řádků i sloupců vlastnost být latinským čtvercem zachovávají. Při prohození dvou řádků nevzniknou dva stejné prvky ve sloupci, protože hodnoty ve sloupcích mění jen pořadí, a ani v řádku, protože řádky se nemění vůbec. Analogicky se dá postupovat při prohazování sloupců.

Ortogonalita se může pokazit při obou operacích. Pro permutace řádků to je vidět na příkladě z obrázku, kde na prvních dvou latinských čtvercích stačí prohodit 2. a 3. řádek ve druhém čverci, čímž se 1. a 2. čtverec budou shodovat na 1. a 2. řádku. Pro permutace sloupců stačí uvážit stejnou permutaci pro transponované čtverce a romyslet si, že transpozice taky nepokazí vlastnost být latinským čtvercem. □

3 Toky v sítích

Síť je uspořádaná čtveřice (G, z, s, c), kde $G = (V, E)$ je orientovaný graf, neboli $E \subseteq V \times V$, z a s jsou dva různé vrcholy grafu G (zvané *zdroj* a *stok*) a *kapacita* $c\colon E \to \mathbb{R}_0^+$ je funkce ohodnocující hrany. *Tok v síti* je každá funkce $f\colon E \to \mathbb{R}$ splňující $0 \leq f(e) \leq c(e)$ pro každou hranu $e \in E$ a

$$\sum_{v:(u,v)\in E} f(u,v) = \sum_{v:(v,u)\in E} f(v,u)$$

pro každý vrchol $u \in V$ mimo stok a zdroj. *Velikost toku* je

$$w(f) = \sum_{v:(z,v)\in E} f(z,v) - \sum_{v:(v,z)\in E} f(v,z).$$

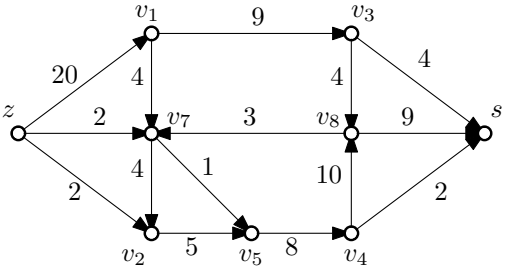
Řezem nazveme podmnožinu E, po jejímž odstranění neexistuje orientovaná cesta ze zdroje do stoku. *Kapacita řezu* R je $c(R) = \sum_{e\in R} c(e)$.

Jako *rezervu hrany* e na nějaké cestě P ze z do s označíme $r(e) = c(e) - f(e)$ pro hranu e orientovanou po směru P a $r(e) = f(e)$ pro hranu orientovanou proti směru P. Cesta P je pak *zlepšující*, pokud všechny její hrany mají kladnou rezervu.

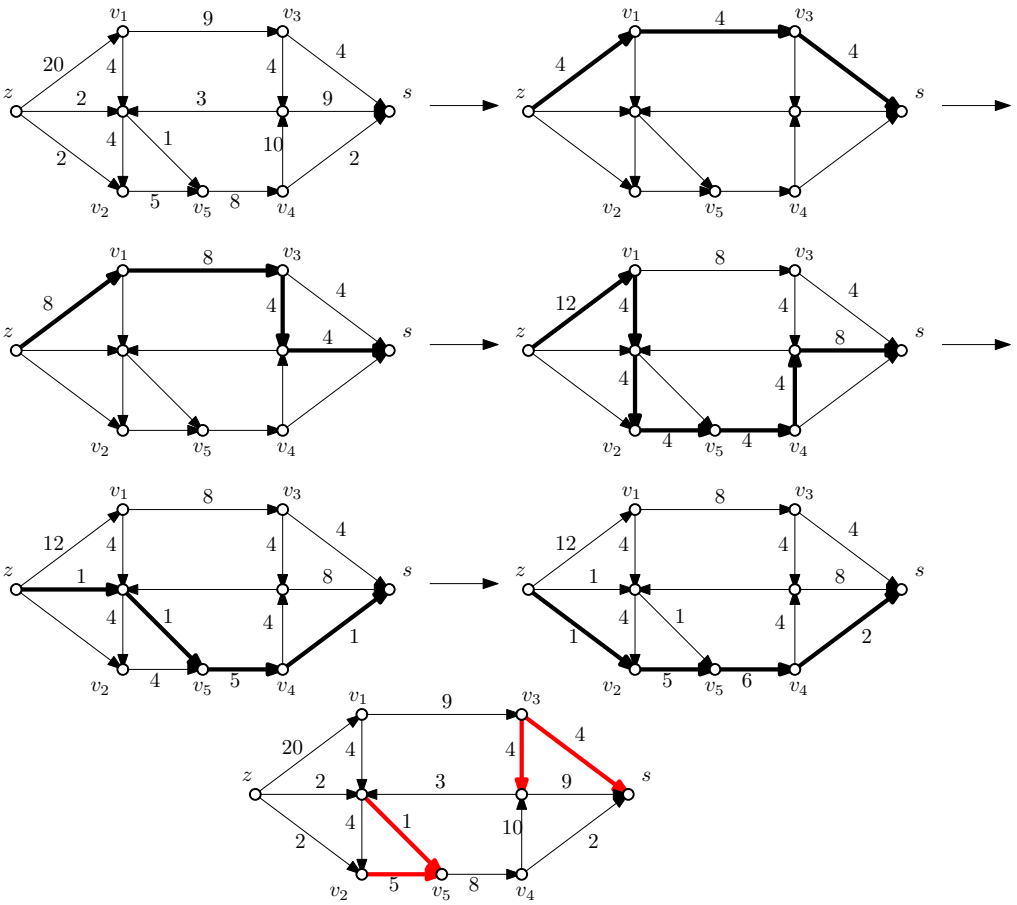
Algorithm 3.1: FORD–FULKERSON(G)

$f \leftarrow$ *nulový tok*
while existuje zlepšující cesta P ze z do s
do $\begin{cases} \epsilon_P \leftarrow \min_{e\in E(P)} r(e) \\ \text{Zvětšíme tok } f \text{ podél } P \text{ o } \epsilon_P \text{ (každé hraně } e \text{ po směru zvětšíme} \\ f(e) \text{ a hranám proti směru zmenšíme } f(e)). \end{cases}$
Vrať tok f.

Příklad 3. *Najděte Fordovým–Fulkersonovým algoritmem tok maximální velikosti v následující síti. Nalezněte také řez minimální kapacity a ověřte tak, že daný tok má skutečně maximální velikost.*



Řešení. Základní myšlenka algoritmu: používat i hrany v protisměru a tok ve směru simulovat odečtením toku v protisměru. Síť má celočíselné kapacity a tedy úloha má smysl. Jako rezervu hrany e označme $c(e) - f(e) + f(e')$, kde e' je hrana opačné orientace. Potom zlepšující cesta je cesta, ve které má každá hrana nenulovou rezervu (obsahuje-li síť cyklus délky 2, pak je třeba i uvážit tok opačnou hranou v definici toku). Pokud nemáme cykly délky 2, pak rezerva je $c(e)-f(e)$ pro hranu orientovanou po směru cesty a $f(e')$ pro hranu e orientovanou proti směru cesty (to proto, že hrany proti směru cesty mají nulové kapacity a nulové toky).



Nejprve nastavíme nulový tok. Poté hledáme zlepšující cesty (koukáme na rezervy, ne na kapacity!) a posíláme po nich toky. Vždy po zvětšení toku musíme přenastavit rezervy (ty jsou označené na obrázku) i u opačných hran. Takto dostaneme maximální tok velikosti 14. Viz obrázek.

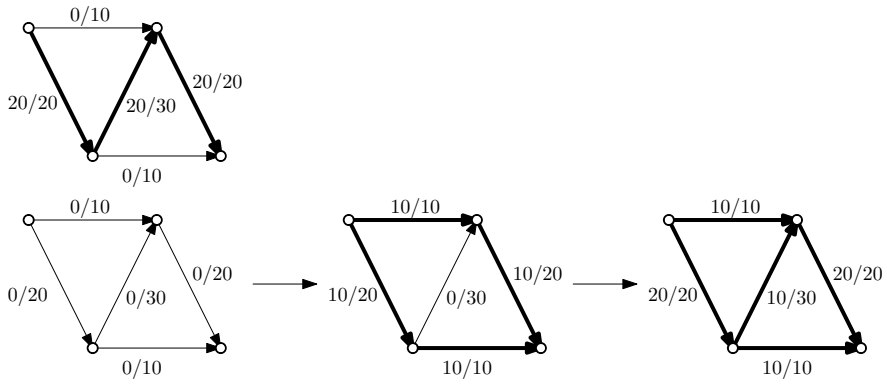
Pro rychlý běh je třeba strategie výběru cesty, jinak může být pomalé (např. podobný příklad jako u selhání hladové strategie s kapacitami). Cesty ze sítě rezerv jde vybírat i hladově, vybíráme-li je průchodem do šířky, je složitost $O(|V||E|^2)$ (Edmons-Karp).

řez minimální kapacity jde najít jako množinu hran $S(A, V \setminus A)$, které jdou z A do $V \setminus A$ (přesně v této orientaci!), kde A je množina vrcholů, do nichž se jde dostat ze zdroje po nenasycené cestě. V našem příkladu je takový řez červeně vyznačený. □

Příklad 4. *(a) Najděte síť (a posloupnost použitých zlepšujících cest), na které F.-F. algoritmus nedospěje ke správnému výsledku, pokud mu povolíme používat jen orientované zlepšující cesty.*

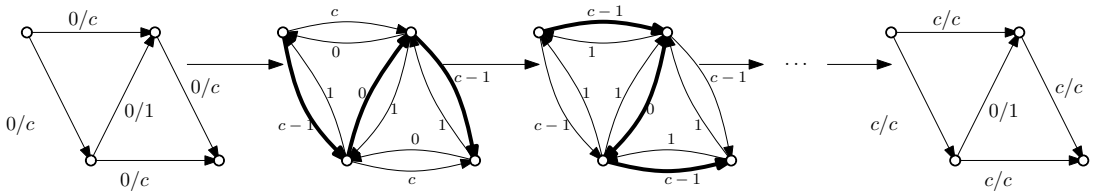
(b) Najděte posloupnost sítí (a posloupnosti použitých zlepšujících cest), na které má F.-F. algoritmus exponenciální časovou složitost (vzhledem k počtu bitů potřebných k uložení grafu a kapacit).

Řešení. (a) řešení je vidět z následujícího obrázku, kde hladový přístup najde tok velikosti 20, ale přitom existuje tok velikosti 30.



(b) Máme podobné řešení jako předtím. Velikost grafu je konstantní, kapacity potřebují k uložení logaritmicky mnoho bitů vzhledem k velikosti maximálního toku, zatímco časová složitost je

$O(|f|)$. Na prvním a posledním obrázku je u hran znázorněn tok a kapacity (síť), jinak rezervy (síť rezerv, c je nějaká velká konstanta).



□

Příklad 5. *Při hledání maximálního toku jsou kapacitami omezené hrany. Někdy se ale může stát, že budeme potřebovat nějaké kapacity přiřadit i vrcholům („vrcholem v nesmí protéct víc než x litrů tekutiny za jednotku času"). Jak najít maximální tok splňující i tuto podmínku?*

Řešení. Stačí uvážit rozdělení vrcholu v na dva v^- a v^+ a dát mezi ně hranu s kapacitou rovnou danému omezení na průtok vrcholem v. Potom stačí klasicky hledat maximální tok. Nalezený tok splňuje dodatečnou podmínku, protože jinak by daná vrcholová hrana byla přesycená. □

Příklad 6. *Ukažte, že problém hledání maximálního toku v síti, která má více zdrojů a více stoků, lze redukovat na případ s jedním zdrojem a jedním stokem.*

Řešení. Mějme síť $(G = (V, E), (z_1, \ldots, z_k), (s_1, \ldots, s_l), c)$ s k zdroji a l stoky. Definujme nový zdroj z a nový stok s a přidejme hrany $(z, z_1), \ldots, (z, z_k)$ a $(s_1, s), \ldots, (s_l, s)$ s velmi velkou kapacitou C (například součet všech kapacit v původní síti).

Potom tok maximální velikosti v této nové síti má stejnou velikost jako tok maximální velikosti původní síti. Jistě každý tok v původní síti lze rozšířit na tok v nové síti stejné velikosti, stačí z nového zdroje poslat do každého původního zdroje poslat tolik toku, kolik z něj v původním toku vyplývá (to lze z volby C) a analogicky to udělat se stoky. čili tok maximální velikosti v nové síti je aspoň tak velký jako v té původní.

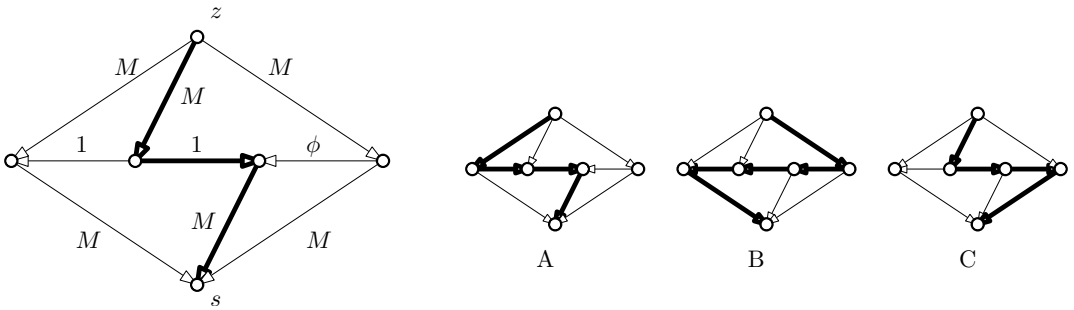
Naopak řez v původní síti je řezem v nové síti a tedy minimální kapacita řezu v nové síti je nanejvýš tak velká jako v původní síti. Podle Hlavní věty o tocích je tak velikost maximálního toku v obou sítích totožná. □

Příklad 7. *Dokažte, že počet toků maximální velikosti je v každé síti buď jedna nebo je nekonečný.*

Řešení. Uvažme síť, ve které jsou aspoň dva maximální toky f_1 a f_2. Potom libovolná jejich konvexní kombinace $\alpha f_1 + (1 - \alpha) f_2$, $\alpha \in [0, 1]$ je opět maximálním tokem. Především je to tok, protože pro každou hranu e máme $0 \leq \alpha f_1(e) + (1 - \alpha) f_2(e) \leq \alpha c(e) + (1 - \alpha) c(e) = c(e)$ a platí Kirchhoffův zákon, jelikož do každého vrcholu přitéká právě to co odtéká, což jde vidět, rozdělíme-li přítok na část přitékající přes f_1 a část přitékající přes f_2. Navíc nový tok je maximální, protože je snadno vidět, že jeho celková velikost je stejná jako u obou maximálních toků.

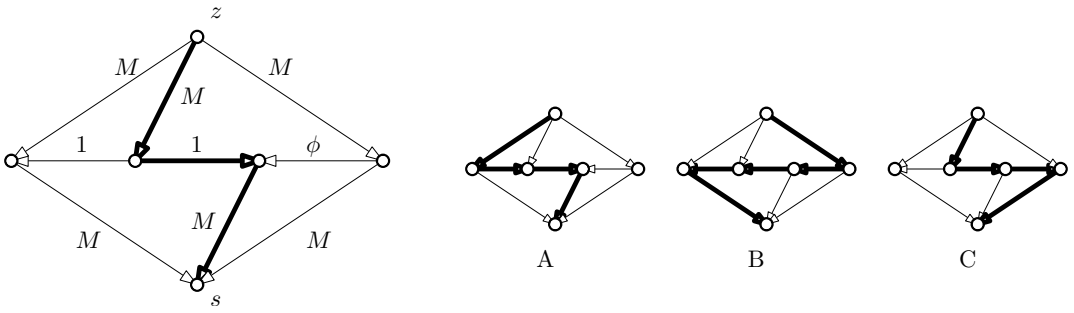
čili počet maximálních toků je jedna nebo nekonečně mnoho. Sítí s jediným maximálním tokem je třeba orientovaná cesta s jednotkovými kapacitami a zdrojem a stokem jako koncovými vrcholy. □

Příklad 8. *(*) Najděte síť s iracionálními váhami, na které F.-F. algoritmus nenajde maximální tok. (Hint:*



Kde $\phi^2 = 1 - \phi$.)

Síť, na které Ford-Fulkersonův algoritmus selže: F.-F. algoritmus funguje dobře na racionálních kapacitách. O celočíselných to víme, racionální obecně stačí vynásobit společným jmenovatelem a poté postupovat ve vzniklé celočíselné síti. Na iracionálních kapacitách obecně už algoritmus může selhat, což si ukážeme na příkladu od Uri Zwicka.



Uvažme síť z obrázku. Má 6 vrcholů a 9 hran, které jsou ohodnocené čísly 1, M (= velké celé číslo) a $\phi = \frac{\sqrt{5}-1}{2}$, které je zvolené tak, aby platilo $\phi^2 = 1 - \phi$. Při analýze selhání budeme sledovat rezervy tří horizontálních hran (rezervy zbylých šesti hran budou vždy alespoň $M - 3$).

F.-F. algoritmus začne posláním toku velikosti 1 po zlepšující cestě na největším obrázku. Rezervy tří horizontálních hran jsou (zleva doprava) $1, 0, \phi$. Nechť v indukci mají tyto hrany rezervy $\phi^{k-1}, 0, \phi^k$ pro nějaké $k \in \mathbb{N}$. Nyní postupně provedeme tyto 4 iterace F.-F. algoritmu:

1. Zlepšení o ϕ^k podle B – rezervy jsou poté $\phi^{k+1}, \phi^k, 0$.
2. Zlepšení o ϕ^k podle C – rezervy jsou poté $\phi^{k+1}, 0, \phi^k$.
3. Zlepšení o ϕ^{k+1} podle B – rezervy jsou poté $0, \phi^{k+1}, \phi^{k+2}$.
4. Zlepšení o ϕ^{k+1} podle A – rezervy jsou poté $\phi^{k+1}, 0, \phi^{k+2}$.

Tedy po $4n + 1$ iteracích algoritmu jsou rezervy daných hran $\phi^{2n}, 0, \phi^{2n+1}$. S rostoucím počtem kroků k nekonečnu velikost toku konverguje k

$$1 + 2\sum_{i=1}^{\infty} \phi^i = \frac{2}{1-\phi} - 1 = 2 + \sqrt{5} < 5,$$

což je málo, protože jde snadno vidět, že největší tok je $2M + 1$.

Příklad 9. *(*) Rozmyslete si analýzu Edmons-Karpova algoritmu:*

Algorithm 3.2: *EdmondsKarp(G)*

$f \leftarrow$ nulový tok
while *existuje zlepšující cesta P ze z do s*
do {
Buď P taková cesta s minimálním počtem hran.
$\epsilon \leftarrow \min_{e \in P} r(e)$
Zvětšíme tok f podél P o ϵ (každé hraně $e \in P$ zvětšíme $f(e)$, případně zmenšíme $f(e')$, podle toho, co jde).
}

Analýza složitosti Edmonds-Karpova algoritmu: Z příkladů uvedených výše je vidět, že složitost Ford-Fulkersonova algoritmu může být ošklivá. Ukážeme, ale, že vybíráme-li nejkratší z, s-cesty, tak dostaneme složitost polynomiální vzhledem k počtu hran m a vrcholů n (Edmonds-Karpův algoritmus). Algoritmus je založen na procházení grafu do šířky (BFS).

Algorithm 3.3: BFS(G, s)

$\forall v \in V : H(v) \leftarrow -1$
$H(s) \leftarrow 0$ a přidej s do fronty
while fronta je neprázdná
do { Odeber vrchol v z fronty.
for $\forall$ sousedy w vrcholu v
do { **if** $H(w) = -1$
then { Přidej w do fronty.
$H(w) \leftarrow H(v) + 1$

BFS prochází postupně všechny vrcholy grafu po vrstvách vzdálenosti od iniciálního vrcholu. Běží v čase $\Theta(m + n)$, protože testuje dvakrát každý vrchol pro každou jeho hranu a do fronty ho dává jednou. Je základem pro testování souvislosti a bipartitnosti, hledání minimální kostry nebo nejkratší cesty (prohledávání do hloubky, DFS, je základem pro topologické třídění, testování souvislosti, 2-souvislosti, silné souvislosti,...) Je-li graf souvislý, tak $n \leq m + 1$ a tedy máme čas $\Theta(m)$.

Algorithm 3.4: EdmondsKarp(G)

$f \leftarrow$ *nulový tok*
while existuje zlepšující cesta P ze z do s
do { Buď P taková cesta s minimálním počtem hran.
$\epsilon \leftarrow \min_{e \in P} r(e)$
Zvětšíme tok f podél P o ϵ (každé hraně $e \in P$ zvětšíme $f(e)$, případně zmenšíme $f(e')$, podle toho, co jde).

Nechť v grafu nejsou cykly délky 2. Síť rezerv grafu $G = (V, E)$ a toku f je definována jako $G_f = (V_f, E_f)$, kde $V_f = V$ a E_f obsahuje dopředné hrany: je-li $e \in E(G)$, kde $f(e) < c(e)$ tak do E_f přidáme hranu e s kapacitou $c(e) - f(e)$, zpětné hrany: je-li $e = (u, v) \in E$ s $f(e) > 0$, přidáme do E_f hranu $e' = (v, u)$ s kapacitou $f(e)$.

Uvážíme počet iterací cyklu v Edmonds-Karpově algoritmu. Při BFS se vrcholy grafu roztřídí do vrstev BFS stromu $L_0, \ldots, L_i$ podle značek $H(v)$. Víme $L_0 = \{z\}$. Platí také to, že každá nejkratší cesta ze z do libovolného $v \in L_j$ obsahuje právě jeden vrchol z každé vrstvy $L_0, \ldots, L_j$ v tomto pořadí a žádné jiné vrcholy. čili každá nalezená z, s-cesta je tvaru $z = v_0, v_1, \ldots, v_j = s$, kde $v_k \in L_k$ pro každé k, $0 \leq k \leq j$.

Změny G_f po vylepšení toku po zlepšující cestě P:

1. Obsahuje-li P dopřednou hranu e, kterou vylepšení nasytí (nově bude $c(e) = f(e)$), tak e z G_f smažeme a můžeme přidat opačnou hranu e' (pokud ji G_f už před vylepšením neobsahovalo, což bylo, pokud tok přes e byl nulový).

2. Obsahuje-li P zpětnou hranu e', tak ji smažeme z G_f, pokud vylepšení sníží tok na e na nulu. Byla-li dopředná hrana e před vylepšením nasycená ($c(e) = f(e)$), tak ji poté přidáme do G_f.

žádné jiné hrany nemažeme ani nepřidáváme. Tedy přidáváme jen opačné hrany k hranám z P.

Každá hrana z P jde mezi i-tou a $i + 1$-ní vrstvou, tedy nové hrany jsou mezi $i + 1$-ní a i-tou vrstvou pro nějaké i. Tedy délka cesty ze z do libovolného v se při běhu algoritmu nikdy nesníží! (nikdy novou hranou nepřeskočíme vrstvu)

Je-li hrana $e = (u, v) \in P$ bottleneck (tj. $r(e) = \epsilon$), potom ji po vylepšení f z G_f odstraníme. Nechť $u \in L_i$ a $v \in L_{i+1}$ když toto nastane. Pokud se daná hrana znovu objeví v G_f, tak u musí poté být ve vyšší vrtvě než v, protože při objevení vždy hrana vede z jedné vrsty do právě té předcházející. Vrchol v zůstává v L_{i+1} či výš (tj. v úrovni s vyšším indexem), protože délka z, v-cesty se nikdy nesnižuje (otáčením hran nevytvoříme zkratku). Tedy u musí před dalším objevením hrany e postoupit do vrstvy L_{i+2} či výš, protože jinak by při znovuobjevení e nebylo ve vyšší vrstvě než v. BFS strom nemá víc než n vrstev, tedy e se jako bottleneck může obevit nanejvýš $n/2$-krát (stane-li se z e znovu bottleneck, tak už zase musí být v ve vyšší vrstvě než u). Celkem je $2m$ hran,

které se potenciálně mohou objevit v G_f, každá z nich se jako bottleneck objeví nejvýš $n/2$-krát, to celkem dělá nanejvýš mn bottlenecků. Každá zlepšující cesta má bottleneck a každý cyklus odpovídá zlepšující cestě, tedy máme nanejvýš mn iterací cyklu.

Každá iterace zabere čas $O(m)$, tedy celková složitost Edmonds-Karpova algoritmu je $O(m^2n)$.