VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

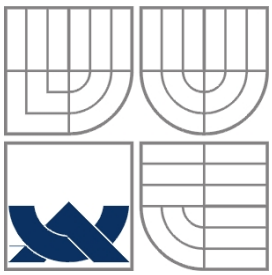
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

IMPLEMENTACE FORD-FULKERSONOVA ALGORITMU

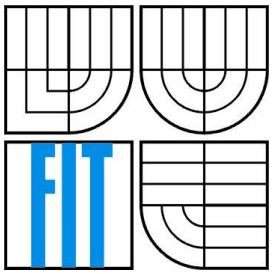
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE BENJAMIN MAKOVSKÝ
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

IMPLEMENTACE FORD-FULKERSONOVA ALGORITMU

IMPLEMENTATION OF THE FORD-FULKERSON ALGORITHM

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

BENJAMIN MAKOVSKÝ

VEDOUCÍ PRÁCE
SUPERVISOR

MGR. TOMÁŠ MASOPUST

BRNO 2007

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2006/2007

Zadání bakalářské práce

Řešitel: **Makovský Benjamin**
Obor: Informační technologie
Téma: **Implementace Ford-Fulkersonova algoritmu**
Kategorie: Alg. a datové struktury

Pokyny:

1. Seznamte se s teorií grafů, zejména s toky v sítích a Ford-Fulkersonovým algoritmem na hledání maximálního toku.
2. Navrhněte grafickou implementaci Ford-Fulkersonova algoritmu na hledání maximálního toku/minimálního řezu v síti jako Java appletu. Vstupem bude síť, výstupem bude zvýrazněný maximální tok a naznačen minimální řez.
3. Studujte možnosti užití různých rychlejších metod například pro rovinné sítě.
4. Zhodnoťte dosažené výsledky a diskutujte další možný vývoj projektu.

Literatura:

- Dememl, J.: Grafy a jejich aplikace (8.3.6 a 8.4.5)

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Masopust Tomáš, Mgr.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 15. května 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Jaroslav Zendulka, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Benjamin Makovský**
Id studenta: 43854
Bytem: Vyhlídka 865/15, 638 00 Brno
Narozen: 05. 06. 1973, Brno
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Implementace Ford-Fulkersonova algoritmu
Vedoucí/školitel VŠKP: Masopust Tomáš, Mgr.
Ústav: Ústav informačních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

.....	.....
Nabyvatel	Autor

Abstrakt

Tato práce navrhuje a řeší grafickou implementaci Ford-Fulkersonova algoritmu pro hledání maximálního toku a minimálního řezu v síti. Obsahuje stručné seznámení s teorií grafů a toků v sítích, popisuje princip Ford-Fulkersonova algoritmu. V práci je uveden objektový návrh reprezentující graf v programu, je popsáno řešení vykreslování grafu programem a vytvoření grafického uživatelského rozhraní aplikace. Výsledný program je zpracován jako Java applet, který je umístěn na veřejných internetových stránkách www.ffaplikace.php5.cz.

Klíčová slova

Graf, vrchol, hrana, síť, maximální tok, minimální řez, Ford-Fulkersonův algoritmus, Java applet.

Abstract

This work describes the design of the graphic implementation of the Ford-Fulkerson Algorithm for searching the maximum flow and the minimal cut in the network. It contains a brief introduction with the theory of the graphs and the flows in the networks, it describes the principle of the Ford-Fulkerson Algorithm. The object design representing the graph in the program is cited, the drawing of the graph by the program and the creation of the graphic user interface of the application is described. The final program is worked up as a Java applet which is placed at the public internet pages www.ffaplikace.php5.cz.

Keywords

Graph, node, edge, network, maximum flow, minimum cut, Ford-Fulkerson Algorithm, Java applet.

Citace

Benjamin Makovský: Implementace Ford-Fulkersonova algoritmu, bakalářská práce, Brno, FIT VUT v Brně, 2007

Implementace Ford-Fulkersonova algoritmu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Mgr. Tomáše Masopusta.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Benjamin Makovský
10. 05. 2007

Poděkování

Na tomto místě bych chtěl poděkovat vedoucímu mé bakalářské práce Mgr. Tomáši Masopustovi za zájem, připomínky a čas, který věnoval mé práci.

Obsah

Úvod

Cílem bakalářské práce bylo blíže se seznámit s teorií grafů, s toky v sítích a s Ford-Fulkersonovým algoritmem pro hledání maximálního toku a minimálního řezu v síti. Vlastní práce navrhuje a řeší grafickou implementaci Ford-Fulkersonova algoritmu jako Java appletu. Práce svou šíří zasahuje do oblasti diskrétní matematiky, programování a počítačové grafiky.

V první kapitole jsou definovány pojmy a výrazy z teorie grafů a sítí, které jsou důležité pro pochopení problematiky a jsou používány dále v textu. Kapitola směřuje k objasnění obecného principu algoritmu pro prohledávání grafu a algoritmu pro hledání maximálního toku v síti.

Druhá kapitola objasňuje princip Ford-Fulkersonova algoritmu, který je zde podrobně popsán pseudokódem. Závěrečná část kapitoly se věnuje podmínkám konečnosti algoritmu a časovému odhadu rychlosti jeho výpočtu.

Ve třetí kapitole se zabývám návrhem řešení programu. Je zde představen objektový návrh reprezentující graf v programu a uveden navržený diagram tříd.

Z provedené analýzy a návrhu řešení vyplývá implementace v konkrétním programovacím jazyce (Java), která je popsána ve čtvrté kapitole. Je zde uvedeno řešení vykreslení grafu na obrazovku a grafického uživatelského rozhraní aplikace.

V páté kapitole je stručně popsána metoda hledání maximálního toku v rovinné síti, která vede k rychlejšímu výpočtu.

Závěrečná šestá kapitola obsahuje zhodnocení práce a celkového řešení programu. Část kapitoly je věnována popisu a vyhodnocení testování hotové aplikace. Jsou zde nastíněny další možné vývoje projektu.

V přílohách na konci práce je uveden návod pro používání programu (příloha A), programová dokumentace, která je automaticky vygenerována nástrojem `javadoc.exe` (příloha B). V poslední příloze (C) je popsán obsah přiloženého datového nosiče s nahraným programem.

1 Analýza problému a princip řešení

1.1 Orientovaný graf

Graf se skládá z vrcholů a hran. Hrana vždy spojuje dva vrcholy a může být orientovaná nebo neorientovaná. Orientovaná hrana vede z počátečního vrcholu do koncového vrcholu. Spojuje-li hrana vrchol se sebou samým, jedná se o smyčku.

Orientovaný graf má všechny hrany orientované, neorientovaný graf má všechny hrany neorientované.

1.1.1 Definice orientovaného grafu

Orientovaný graf je trojice **G = (V, E, ε)**,
kde **V** je neprázdná konečná množina, jejíž prvky jsou vrcholy grafu,
E je konečná množina, jejíž prvky jsou orientované hrany
a **ε** je zobrazení $\varepsilon : E \rightarrow V^2$, které nazýváme vztahem incidence.

Toto zobrazení přiřazuje každé hraně $e \in E$ uspořádanou dvojici vrcholů (x, y), které nazýváme krajními vrcholy hrany e. Prvý z nich nazýváme počátečním vrcholem hrany a značíme jej PV(e), druhý nazýváme koncovým vrcholem hrany a značíme KV(e).

Jestliže PV(e) = KV(e), pak hranu e nazýváme smyčkou.

Je možné, aby několik hran mělo stejné počáteční a koncové vrcholy, tj. aby pro různé hrany e_1, e_2 platilo $PV(e_1) = PV(e_2)$ a $KV(e_1) = KV(e_2)$.

Množina hran grafu může být prázdná.

1.1.2 Důležité množiny hran a vrcholů

Množinu vrcholů grafu G značíme V(G), množinu hran grafu G značíme E(G). Počet prvků libovolné konečné množiny M značíme |M|.

1.1.2.1 Vybrané množiny hran a vrcholů v orientovaném grafu

Nechť je dán orientovaný graf G = (V, E, ε), *x* a *y* jeho dva libovolné vrcholy a $A \subseteq V$ je libovolná množina jeho vrcholů, pak zavedeme následující značení, které budu používat v dalším textu:

$V_G^+(x) = \{z \in V \mid (x,y) \in \varepsilon(E)\}$	množina vrcholů, do nichž vede hrana z *x*.
$V_G^-(x) = \{z \in V \mid (x,y) \in \varepsilon(E)\}$	množina vrcholů, z nichž vede hrana do *x*.
$V_G(x) = V_G^+(x) \cup V_G^-(x)$	množina vrcholů spojených hranou s vrcholem *x*.

$V_G(A) = \bigcup_{x \in A} V_G(x)$ množina vrcholů spojených hranou s některým vrcholem z A.

$E_G^+(x) = \{e \in E \mid Pv(e) = x\}$ množina hran s počátečním vrcholem *x*.

$E_G^-(x) = \{e \in E \mid Kv(e) = x\}$ množina hran s koncovým vrcholem *x*.

$E_G(x) = E_G^+(x) \cup E_G^-(x)$ množina hran incidentních s vrcholem *x*.

$W_G^+(x) = \{e \in E \mid Pv(e) \in A \ \& \ Kv(e) \notin A\}$ množina všech hran, jejichž počáteční vrchol leží v A a koncový vrchol neleží v A.

$W_G^-(x) = \{e \in E \mid Pv(e) \notin A \ \& \ Kv(e) \in A\}$ množina všech hran, jejichž počáteční vrchol neleží v A a koncový vrchol leží v A.

$W_G(x) = W_G^+(x) \cup W_G^-(x)$ množina všech hran, jejichž jeden vrchol leží v A a druhý v množině A neleží (řez určený množinou vrcholů A).

1.1.3 Cesta v orientovaném grafu

1.1.3.1 Orientovaný sled

Orientovaným sledem nazýváme posloupnost vrcholů a hran $v_0, e_1, v_1, e_2, v_2, e_3, \ldots, e_k, v_k$, jestliže pro každou hranu e_i z této posloupnosti platí $Pv(e_i) = v_{i-1}$ a $Kv(e_i) = v_i$.

O sledu říkáme, že vede z vrcholu v_0 do vrcholu v_k (spojuje vrcholy v_0, v_k).

1.1.3.2 Orientovaná cesta

Orientovanou cestou nazýváme orientovaný sled, ve kterém se neopakuje žádný vrchol.

Protože se v cestě neopakují vrcholy, neopakují se v ní ani hrany.

1.1.4 Způsoby zadávání grafů

Existuje řada používaných způsobů zadávání grafů jako např.:

- seznamy vrcholů a hran,
- seznamy vrcholů a seznamy okolí vrcholů,
- matice sousednosti,
- matice incidence,
- nepřímý popis grafu.

Ve výsledné aplikaci jsem použil druhý uvedený způsob (seznamy vrcholů a seznamy okolí vrcholů), který je popsán dále.

Vrcholy a popřípadě i hrany se obvykle označují po sobě jdoucími přirozenými čísly.

1.1.4.1 Zadávání grafu seznamem vrcholů a seznamem okolí vrcholů

Množina vrcholů je popsána prostým výčtem (seznamem) prvků, hrany jsou popisovány po skupinách. Pro každý vrchol *x* je vždy uveden seznam množiny hran $E^+(x)$ (výstupní okolí vrcholu *x*).

Každá hrana je popsána svým jménem a koncovým vrcholem. Počáteční vrchol *x* je pro celou skupinu hran společný.

Příklad popisu grafu, který je v aplikaci použit jako referenční (obr. 4.1 v kap. 4.3):

v_0: (e_1, v_0), (e_2, v_1), (e_3, v_2)

v_1: (e_4, v_3)

v_2: (e_5, v_0), (e_6, v_1), (e_7, v_4)

v_3: (e_8, v_4), (e_9, v_5)

v_4: (e_{10}, v_1), (e_{11}, v_5)

v_5: Ø.

1.1.5 Kreslení grafů

Grafy se často znázorňují kreslením. Rovněž v navrhované aplikaci je nutné graf přehledně vykreslit. Vrcholy se znázorní kružnicemi, hrany jako čáry (úsečky, oblouky), které spojují příslušné dvojice vrcholů. Orientaci hrany označíme šipkou u koncového vrcholu.

Při kreslení grafu je dobré dodržovat zásadu, aby hrany byly kresleny pokud možno přímo s co nejmenším počtem průsečíků.

1.2 Toky v síti

Orientovaný graf zde představuje model sítě (např. dopravní, vodovodní, spojové, internetové), po níž se přepravuje určitá hmotná či nehmotná substance (zboží, voda, data). Přepravní kapacita každé hrany (přepravní cesty) je nějakým způsobem omezena, a tím jsou omezeny i celkové přepravní možnosti sítě.

Obvykle nás zajímá problém přenést z daného „zdroje“ do daného cíle – „spotřebiče“ co nejvíce této substance za omezujících podmínek kapacit jednotlivých hran.

1.2.1 Definice sítě

Graf, jehož hrany (případně vrcholy) jsou opatřeny nějakými hodnotami (obvykle číselnými), nazýváme ohodnoceným grafem nebo též sítí.

Síť je čtveřice **S = (G, z, s, c)**, kde

G je orientovaný graf,

vrcholy $z \in V(G)$ a $s \in V(G)$ jsou zdroj a spotřebič,

$c: E(G) \rightarrow \mathbf{R}$ je ohodnocení hran, zvané kapacita hran.

1.2.2 Definice toku v síti

Tokem v síti S = (G, z, s, c) nazýváme takové ohodnocení hran reálnými čísly f : E(G) → **R**, které pro každý vrchol *v* splňuje Kirchhoffův zákon

$$\sum_{e \in E^{+}(v)} f(e) = \sum_{e \in E^{-}(v)} f(e) .$$

Tok, který do vrcholu vtéká vstupními hranami je roven toku, který z vrcholu vytéká výstupními hranami. Orientace hrany určuje směr proudění. Záporná velikost toku znamená proudění proti směru hrany.

1.2.2.1 Tok od zdroje ke spotřebiči

Tok od zdroje ke spotřebiči splňuje Kirchhoffův zákon pro všechny vrcholy kromě zdroje a spotřebiče. Ve zdroji tok vzniká a ve spotřebiči stejné množství toku zaniká.

1.2.2.2 Velikost toku od zdroje ke spotřebiči

Velikost toku definujeme jako množství toku, které vzniká ve zdroji, tj.

$$F(f) = \sum_{e \in E^{+}(z)} f(e) - \sum_{e \in E^{-}(z)} f(e) .$$

1.2.3 Přípustný tok, transportní síť

Přípustným tokem nazýváme tok, který pro všechny hrany grafu splňuje nerovnosti

$$l(e) \le f(e) \le c(e) \text{ , kde}$$

číslo c(e) představuje kapacitu hrany *e* a nazýváme jej horním omezením toku v hraně *e*
a číslo l(e) nazýváme dolním omezením toku v hraně *e*.

Je-li v síti dán zdroj a spotřebič a jsou-li dolní omezení toku ve všech hranách nulová, nazýváme tuto síť transportní sítí. V takovém případě je nulový tok tokem přípustným.

Programovaná aplikace bude využívat právě transportní síť. Bude zadán zdroj (první vrchol grafu) a spotřebič (poslední vrchol grafu) a dolní omezení toku budou uvažována ve všech hranách nulová.

1.2.4 Úloha o maximálním toku

Je dána transportní síť S = (G, z, s, c(e)). Úkolem je najít maximální přípustný tok od zdroje *z* ke spotřebiči *s*, tj. přípustný tok, který má největší velikost F(f).

1.3 Minimální řez v síti

1.3.1 Definice řezu v síti

Řez v síti S = (G, z, s, c(e)), určený množinou vrcholů A, je množina hran, jejichž jeden vrchol leží v množině A a druhý nikoli. Uvedený řez značíme W(A).

Pokud množina A obsahuje zdroj a neobsahuje spotřebič, mluvíme o řezu oddělujícím zdroj a spotřebič.

1.3.2 Kapacita řezu

Kapacita řezu W(A) je číslo

$$C(A) = \sum_{e \in W^{+}(A)} c(e) - \sum_{e \in W^{-}(A)} l(e) .$$

1.3.3 Úloha o minimálním řezu

V zadané síti S = (G, z, s, c(e)) hledáme řez, který odděluje zdroj od spotřebiče a který má nejmenší kapacitu. Protože velikost maximálního toku je rovna kapacitě minimálního řezu (viz kapitola 1.4.5), lze minimální řez nalézt algoritmem pro maximální tok.

1.4 Hledání maximálního toku v síti

1.4.1 Historický vývoj významných algoritmů

Prvním algoritmem pro hledání maximálního toku byl Ford-Fulkersonův algoritmus (1957), který maximální tok stanovuje opakovaným hledáním zlepšujících cest ze zdroje ke spotřebiči, jejichž pořadí není specifikováno.

Tento nedostatek vyřešili ve svých modifikovaných algoritmech pánové Edmonds a Karp (1972), kteří hledají vždy nejkratší zlepšující cestu nebo zlepšující cestu s maximální kapacitou.

Významný pokrok přinesl Dinicův algoritmus (1972), který převádí problém hledání maximálního toku na opakované hledání nasyceného toku v síti speciálního tvaru.

Z Dinicova algoritmu potom vychází Algoritmus tří Indů (Malhotra, Pramodh Kumar a Maheshwari, 1978), kteří zefektivnili část algoritmu pro hledání nasyceného toku. Jejich algoritmus patří dodnes mezi nejrychlejší při hledání maximálního toku v hustých sítích.

S revolučním algoritmem přišel v roce 1988 pan Goldberg, jehož algoritmus dosahuje dobrých výsledků jak v řídkých tak i v hustých sítích.

1.4.2 Princip algoritmu pro hledání maximálního toku

Algoritmus je založen na postupném zvětšování toku při zachování jeho přípustnosti. Výchozím předpokladem je nějaký přípustný tok (např. nulový v případě transportní sítě).

Protože je třeba pro každý vrchol zachovat platnost Kirchhoffova zákona (kap. 1.2.2), nelze změny toku v hranách dělat jednotlivě. Zvýší-li se tok v hraně končící ve vrcholu *v*, musí se zvýšit tok v některé hraně z $E^+(v)$ anebo snížit tok v některé hraně z $E^-(v)$.

Změnu toku provádíme na hranách, které tvoří cestu od zdroje ke spotřebiči. Jedná se o tzv. zlepšující cestu. Velikost změny toku musí být ve všech hranách zlepšující cesty stejná. Zlepšující cesta je neorientovaná, může procházet i proti směru hran.

Algoritmus pro maximální tok bude vyhledávat zlepšující cesty a na hranách těchto cest měnit tok tak dlouho, dokud existuje zlepšující cesta od zdroje ke spotřebiči.

1.4.3 Zlepšující cesta

Protože hovoříme o neorientovaných cestách v orientovaných grafech, zavedeme následující pojmy: *hrana vpřed* je hrana orientovaná ve směru průchodu cestou a *hrana vzad* je orientovaná proti směru průchodu cestou.

V transportní síti S = (G, z, s, c(e)) máme přípustný tok f(e) na každé hraně. Zlepšující cesta vzhledem k toku f je neorientovaná cesta ze zdroje *z* ke spotřebiči *s*, jejíž každá hrana *e* splňuje:

- je-li hranou vpřed, pak $f(e) < c(e)$,
- je-li hranou vzad, pak $f(e) > 0$.

1.4.3.1 Kapacita zlepšující cesty

Na hranách vpřed lze tok zvýšit, na hranách vzad snížit o nějakou hodnotu $\delta > 0$, kterou nazveme kapacitou zlepšující cesty.

Označme

$\delta_1 = \min (c(e) - f(e))$ pro hrany vpřed

a $\delta_2 = \min (f(e) - 0)$ pro hrany vzad.

Potom pro kapacitu zlepšující cesty platí $\delta = \min (\delta_1, \delta_2)$.

1.4.3.2 Změna toku podél zlepšující cesty

Pokud existuje zlepšující cesta o kapacitě δ, pak na hranách vpřed můžeme tok zvýšit o kapacitu zlepšující cesty δ a na hranách vzad tok snížit o δ. Tím se neporuší podmínky přípustnosti toku ani Kirchhoffův zákon, ale celková velikost toku stoupne o hodnotu δ.

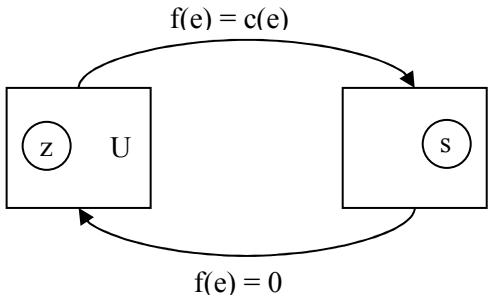
1.4.4 Nalezení minimálního řezu

Pokud v síti S = (G, z, s, c(e)) již nelze najít žádnou zlepšující cestu, potom označíme U množinu uzlů *v*, pro něž existuje cesta ze zdroje *z* do *v*, jejíž hrany splňují podmínky:

- je-li hranou vpřed, pak f(e) = c(e),
- je-li hranou vzad, pak f(e) = 0.

Je zřejmé, že spotřebič s $\notin$ U. Množina hran mezi vrcholy z množiny U a vrcholy z množiny V(G)- U tvoří minimální řez sítě.

Schematicky vypadá situace takto:



1.4.5 Fordova-Fulkersonova věta o maximálním toku a minimálním řezu

Velikost maximálního toku od zdroje ke spotřebiči je rovna kapacitě minimálního řezu oddělujícího zdroj a spotřebič.

1.4.5.1 Důkaz

Pro každý tok f a každý řez W v síti S platí, že velikost toku je menší nebo rovna kapacitě řezu (podle 1.3.2 a 1.2.2.2).

Podle 1.4.4 můžeme k maximálnímu toku sestrojit řez sítě s kapacitou rovnou velikosti tohoto toku

$$F(f) = \sum_{e \in E^{+}(U)} f(e) - \sum_{e \in E^{-}(U)} f(e) = \sum_{e \in E^{+}(U)} f(e) = \sum_{e \in W^{+}(U)} c(e) = C .$$

Takový řez bude tedy mít minimální kapacitu.

1.5 Prohledávání grafu

Z výše popsaného je zřejmé, že ve výsledné aplikaci bude nutné pro hledání cest v grafu použít metody pro prohledávání grafu.

1.5.1 Obecný postup lokálního prohledávání grafu

Vrcholům grafu přiřazujeme značky (*značkování vrcholů*). Má-li vrchol značku, znamená to, že do něj vede cesta z daného výchozího vrcholu.

Pro udržování nalezených a ještě nezpracovaných vrcholů používáme obecnou datovou strukturu – seznam *Úschovna*.

1.5.1.1 Algoritmus

1. Inicializace. Označkujeme vrchol *r*, ostatní vrcholy jsou beze značek. Do seznamu *Úschovna* vložíme vrchol *r*.
2. Test ukončení. Je-li seznam *Úschovna* prázdný, výpočet končí.
3. Volba vrcholu. Ze seznamu *Úschovna* odebereme vrchol *v*.
4. Postup z vrcholu *v*. Pro všechny hrany $E^+(v)$ označíme $w = K_V(e)$ a jestliže *w* nemá značku, označkujeme jej a vložíme do seznamu *Úschovna*. Po zpracování všech hran z množiny $E^+(v)$ pokračujeme bodem 2.

1.5.2 Způsoby implementace prohledávání grafu

Existují různé způsoby implementace obecného algoritmu pro procházení grafu:

1. Procházení do šířky – seznam *Úschovna* je implementován jako fronta, prohledáváme od prvních nalezených vrcholů.
2. Procházení do hloubky – seznam *Úschovna* je implementován jako zásobník, prohledáváme od posledních nalezených vrcholů.
3. Dijkstrův algoritmus pro nejkratší cestu – ze seznamu *Úschovna* vybíráme vždy vrchol nejbližší k počátečnímu vrcholu.

V programované aplikaci je pro hledání zlepšujících cest zvolen první způsob – procházení grafu do šířky, který zajistí nalezení cesty s co nejmenším počtem hran.

1.5.2.1 Časové nároky

Je-li graf zadán seznamem vrcholů a seznamem výstupních okolí vrcholů (kap. 1.1.4.1), proběhne hledání metodou do šířky i do hloubky v čase $O(|V|+|H|)$, kde $|V|$ je počet vrcholů a $|H|$ je počet hran.

2 Ford-Fulkersonův algoritmus

2.1 Popis algoritmu

V následujících kapitolách je uveden popis Ford-Fulkersonova algoritmu pro nalezení maximálního toku v síti tak, jak je použit ve výsledné aplikaci. Pro popis jsem použil vhodný pseudokód.

Vyjdu z nulového toku na začátku a pomocí označkování vrcholů hledám cestu, podél které bude možné tok zvětšit. Označkování vrcholů se provádí tak, aby se průběžně počítala hodnota δ, o kterou se tok zvětší. Po úpravě toku se označkování uzlů zruší a začíná se opět od začátku. To se opakuje tak dlouho, až se dosáhne maximálního toku.

Podstatnou část algoritmu tvoří hledání zlepšující cesty v síti. Je použito prohledávání od zdroje do šířky, takže cesta bude co do počtu hran nejkratší (Karpov Edmondsova úprava). Každý vrchol může být ve stavu neoznačkovaný nebo označkovaný. Protože se hledá neorientovaná cesta, procházejí se nejen následníci ale i předchůdci uzlu. V pomocných proměnách se uchovávají informace o označkování vrcholu, o předchozím vrcholu na zlepšující cestě, o orientaci příslušné hrany ve směru nebo proti směru cesty od zdroje ke spotřebiči a o jakou hodnotu by bylo možné zvýšit tok od zdroje k danému vrcholu.

Po nalezení zlepšující cesty se počínaje od spotřebiče upravují toky v hranách zvyšující cesty o hodnotu δ. Přitom se využívá informace o orientaci příslušné hrany.

2.1.1 Popis algoritmu pseudokódem

2.1.1.1 Vstup

Transportní síť S = (G, z, s, c), G je orientovaný graf, *z* zdroj, *s* spotřebič, *c* celočíselná hodnota kapacity každé hrany.

2.1.1.2 Výstup

Celočíselná hodnota toku každé hrany *f* a množina $A \subset V(G)$, která určuje minimální řez.

2.1.1.3 Pomocné proměnné

Značky vrcholů *stav*[u], předchozí uzly na cestě *pred*[u], směr orientace *smer*[u] a volná kapacita *kap*[u].

2.1.1.4 Algoritmus pro nalezení zlepšující cesty

1. Inicializace všech vrcholů $v \in V(G)$ na neoznačkované.
2. Označkuj zdroj *z*.

3. Zdroj *z* vlož do seznamu *fronta.*
4. Prováděj dokud není seznam *fronta* prázdný.
    5. Ze začátku seznamu *fronta* vyber vrchol *v*.
    6. Pro všechny hrany e $\in$ E^+(v) označ *u* := Kv(e) (následníci) proveď body 7. až 10.
        7. Pokud *u* nemá značku a zároveň platí f(e) < c(e), proveď operace 8. až 10.
            8. Označkuj vrchol *stav*[u], urči předchůdce *pred*[u] := *v*, zaznamenej kladný směr *smer*[u] a možný přírůstek toku *kap*[u] := minimum z hodnot (*kap*[v], c(e) – f(e)).
            9. Vrchol *u* vlož na konec seznamu *fronta*.
            10. Je-li *u* = *s* (spotřebič), značkování končí, zlepšující cesta je nalezena, jinak pokračuj.
    11. Pro všechny hrany e $\in$ E^-(v) označ *u* := Pv(e) (předchůdci) proveď body 12. až 15.
        12. Pokud *u* nemá značku a zároveň platí f(e) > 0, proveď operace 13. až 15.
            13. Označkuj vrchol *stav*[u], urči předchůdce *pred*[u] := *v*, zaznamenej záporný směr *smer*[u] a možný přírůstek toku *kap*[u] := minimum z hodnot (*kap*[v], f(e)).
            14. Vrchol *u* vlož na konec seznamu *fronta*.
            15. Je-li *u* = *s* (spotřebič) značkování končí, zlepšující cesta je nalezena, jinak pokračuj.
16. Zpět do bodu 4.
17. Je-li seznam *fronta* prázdný, značkování končí, zlepšující cesta není nalezena.

2.1.1.5 Algoritmus pro zvýšení toku

1. Začni od spotřebiče, označ *v* := *s*.
2. Kapacita zlepšující cesty *d* = kap(v).
3. Přiřaď vrcholu *u* := *pred*[v] (předchůdce vrcholu *v* na zlepšující cestě).
    4. Je-li *smer*[v] kladný, zvyš tok na hraně (u, v) o kapacitu zlepšující cesty *d*.
    5. Je-li *smer*[v] záporný, sniž tok na hraně (u, v) o kapacitu zlepšující cesty *d*.
    6. Přejdi na předchůdce *v* := *u*.
7. Když *v* = *z* (zdroj), algoritmus zvyšování toku končí, jinak pokračuj bodem 2.

2.1.1.6 Algoritmus Ford-Fulkersonův

1. Po všechny hrany *e* $\in$ E(G) nastav tok f(e) = 0.
2. Dokud existuje zlepšující cesta (algoritmus 2.1.1.4) proveď:
    3. Zvyš tok na nalezené zlepšující cestě (algoritmus 2.1.1.5).
4. Označ množinu A $\subset$ V(G) označkovaných vrcholů, která určuje minimální řez. Výpočet končí, na všech hranách *e* $\in$ E(G) je nalezen maximální tok f(e).

2.2 Konečnost algoritmu

Algoritmus vždy skončí za předpokladu, že jsou kapacity c(e) všech hran $e \in E(G)$ sítě S celočíselné. Pro neceločíselné hodnoty kapacit hran se algoritmus nemusí vůbec zastavit, tok se bude zvětšovat po stále menších krocích a nikdy nedosáhne maxima.

2.3 Rychlost výpočtu algoritmu

Při používání metody hledání do šířky pro značkování vrcholů, tj. používání zlepšujících cest s co nejmenším počtem hran a při celočíselných hodnotách kapacit hran bude maximální tok nalezen v čase $O(|V| * |H|^2)$, kde |V| je počet vrcholů a |H| je počet hran. Časový odhad nezávisí na kapacitách hran ani na celkové velikosti maximálního toku.

3 Návrh řešení aplikace

3.1 Rozdělení na dílčí problémy

Implementaci výsledné aplikace jsem rozdělil na tyto části:

1. Návrh a implementace vytvoření grafu a jeho uložení v paměti.
2. Návrh a implementace vykreslování grafu.
3. Implementace Ford-Fulkersonova algoritmu pro zjištění maximálního toku.
4. Návrh a implementace uživatelského rozhraní a ovládání programu.

3.2 Uložení grafu v paměti

Vrcholy grafu budou uloženy jako objekty v datové struktuře seznam, kterou budu dále nazývat *úložiště vrcholů*. Identifikace vrcholů bude řešena jejich pozicí v tomto úložišti.

Hrany grafu budou uloženy jako objekty v datové struktuře seznam – *úložišti hran*, kterou bude obsahovat každý objekt vrcholu. Každý vrchol bude uchovávat hrany, které z něj vycházejí. Přístup k hraně bude řešen přes vrchol, ze kterého vychází a přes jeho pozici v úložišti hran.

Tento návrh uložení grafu v paměti odpovídá zadávání grafu seznamem vrcholů a seznamem okolí vrcholu podle kap. 1.1.4.1.

3.3 Objektový návrh

Graf se podle definice skládá z vrcholů a hran. Proto jsou v aplikaci navrženy třídy:

1. Třída *Vrchol*, jejíž objekty představují jednotlivé vrcholy grafu.
2. Třída *Hrana*, jejíž objekty představují hrany grafu.
3. Třída *Graf*, podle které se vytvoří konkrétní transportní síť obsahující graf.

Mezi třídami platí vztah kompozice. Graf představuje celek, jeho komponenty jsou vrcholy. Hrany jsou komponenty vrcholu.

3.3.1 Třída *Vrchol*

Třída *Vrchol* obsahuje atributy a metody pro objekty představující vrcholy grafu. Každý objekt této třídy si vytvoří úložiště (vector) pro ukládání objektů hran vycházejících z tohoto vrcholu.

Důležité atributy této třídy jsou proměnné uchovávající souřadnicovou polohu vrcholu na kreslícím plátně, barvu pro vykreslování vrcholu a dále pomocné proměnné pro výpočet Ford-Fulkersonova algoritmu.

Konstruktor této třídy vytvoří vrchol grafu, jeho souřadnicím na kreslícím plátně přiřadí hodnotu (0, 0), která bude změněna při vytváření celého grafu. Druhý - přetížený konstruktor - vytvoří vrchol grafu se zadanými souřadnicemi na plátně.

Třída obsahuje metody pro práci s úložištěm objektů hran: přidání a odstranění objektu hrany, nalezení konkrétního objektu hrany, počet hran, počet shodných hran se zadanou hranou. Dále je zde metoda zajišťující vykreslení vrcholu na kreslící plátno a metoda, která zjišťuje, zda byl vrchol zaměřen ukazatelem myši. Nechybí metoda pro uvolnění vrcholu včetně jemu příslušejících hran z paměti při jeho rušení.

3.3.2 Třída *Hrana*

Objekty vytvořené ze třídy *Hrana* představují jednotlivé hrany grafu. Objekty (hrany) jsou uchovávány v úložišti hran příslušného objektu vrcholu, ze kterého tyto hrany vycházejí.

Jedním atributem třídy je proměnná uchovávající koncový vrchol hrany, podle kterého je zřejmé kam hrana vede (index objektu tohoto vrcholu v úložišti vrcholů grafu). Dalšími atributy je kapacita a tok hrany. Vlastností každého objektu hrany je také barva pro vykreslování hrany na kreslící plátno.

Konstruktor vytvoří objekt hrany se zadanými parametry (koncový vrchol a kapacita) a s nulovým tokem.

Metody této třídy zajišťují vykreslení hrany nastavenou barvou na kreslící plátno včetně popisku hrany a šipky určující orientaci. Prochází-li touto hranou řez, bude rovněž vykreslen. Rozlišuje se kreslení úsečkou či obloukem mezi dvěma vrcholy nebo obloukem kolem jednoho vrcholu (smyčka). Jsou zde rovněž metody pro zjištění zaměření hrany ukazatelem myši.

3.3.3 Třída *Graf*

Instancí třídy *Graf* je transportní síť S = (G, z, s, c). Obsahuje úložiště (vector) objektů představujících vrcholy grafu. První objekt tohoto úložiště (na nulté pozici) představuje vrchol *z* (zdroj) a poslední objekt úložiště představuje vrchol *s* (spotřebič).

Konstruktor vytvoří zadaný počet vrcholů (zatím bez hran) a uloží je do úložiště vrcholů. Zároveň každému vrcholu přiřadí příslušné souřadnice pro kreslení vrcholů na kreslící plátno. To zajišťuje metoda, která souřadnice přiděluje podle počtu vrcholů jedním ze dvou způsobů. Hrany grafu budou doplňovány uživatelem.

Přetížený konstruktor vytvoří referenční graf, který bude použit při prvním spuštění aplikace. Tento graf bude již mít nadefinovány hrany.

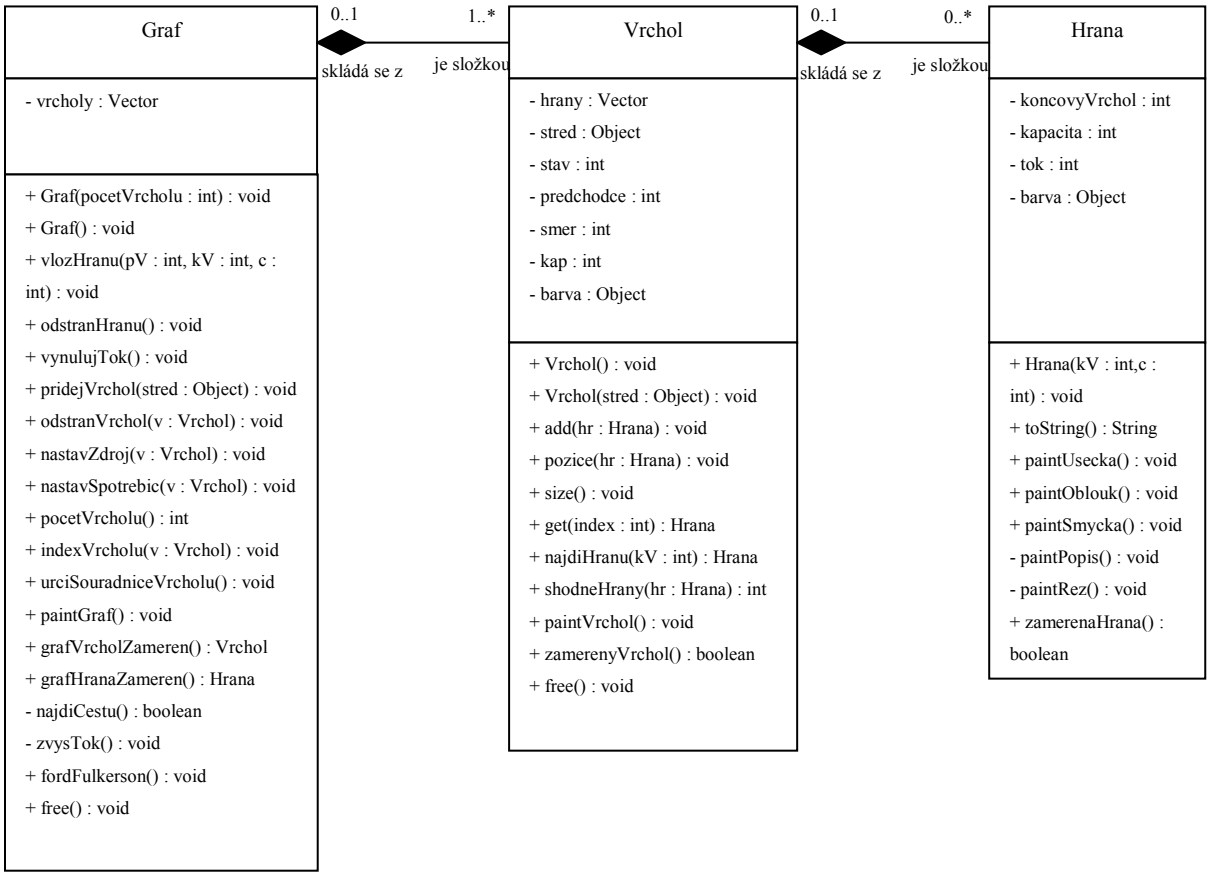
Základní metody této třídy provádějí:

- Přidávání nové hrany příslušnému vrcholu.
- Odstranění zaměřené hrany.

- Inicializace toku v grafu – nastavení na nulovou hodnotu.
- Přidání nového vrcholu.
- Odstranění zaměřeného vrcholu.
- Nastavení zaměřeného vrcholu na zdroj nebo spotřebič.
- Zjištění počtu vrcholů grafu – velikost úložiště vrcholů.
- Nalezení vrcholu v úložišti podle indexu.
- Nalezení indexu zadaného vrcholu v úložišti.
- Vykreslení grafu na kreslící plátno – jsou používány metody pro kreslení vrcholů a hran.
- Zjištění, zda je v grafu ukazatelem myši zaměřen vrchol nebo hrana.
- Ford-Fulkersonův algoritmus pro určení maximálního toku v síti, který používá privátní metody pro nalezení zlepšující cesty a pro zvýšení toku na zlepšující cestě.
- Uvolnění grafu z paměti.

3.4 Navržený diagram tříd

Na následujícím obrázku je uveden navržený diagram tříd představující graf.



3.5 Návrh grafického uživatelského rozhraní

Grafické uživatelské rozhraní je obrazové rozhraní aplikace. Usnadňuje používání programu tak, že poskytuje konzistentní vnější podobu programu. Mělo by se chovat předvídatelně, aby uživatel věděl, co může po provedení nějaké akce očekávat.

Pro snadnou práci s programem ze strany uživatele je třeba, aby v aplikaci bylo vyřešeno:

- Zadávání počtu vrcholů a spuštění vytvoření nového grafu (vstupní pole a tlačítko).
- Jednoduché a rychlé zadávání hran grafu včetně hodnoty kapacity těchto hran. Jsou navrženy dvě možnosti:
    1. Vyplněním vstupních polí pro počáteční a koncový vrchol a kapacitu, poté přidání hrany stiskem příslušného tlačítka.
    2. Postupným klepnutím myši na počáteční a koncový vrchol a zadáním kapacity v dialogovém okně.
- Spuštění výpočtu maximálního toku příslušným tlačítkem. Zřetelné zobrazení výsledku po ukončení výpočtu (hrany toku vykresleny odlišnou barvou, zobrazení hodnot toku na jednotlivých hranách) a zobrazení minimálního řezu v grafu přerušovanou čárou napříč hranami.
- Možnost doplnění nových vrcholů a nových hran, případně odstranění vrcholu nebo hrany grafu pomocí kontextových menu v obrázku.
- Přesouvání vrcholů s příslušnými hranami po kreslícím plátně pomocí myši, aby si uživatel mohl graf zobrazit podle své představy.
- Možnost definování uživatelem, který vrchol bude zdrojem a který spotřebičem pomocí kontextového menu příslušného vrcholu.
- Umožnění uživateli změnit kapacitu hrany pomocí kontextového menu hrany.

4 Popis řešení

4.1 Programovací jazyk

Aplikace je řešena v programovacím jazyku Java, který podporuje dva odlišné typy programů – applety a aplikace. Applet je zvláštní typ programu, který je součástí dokumentu HTTP a je možné s ním pracovat v prohlížeči internetových stránek, když je dotyčný dokument načten. Je vhodné, aby applety nebyly příliš rozsáhlé, aby je bylo možné stáhnout ze serveru v krátké době.

Výsledný program je Java appletem. Z bezpečnostních důvodů není appletům dovolen přístup k počítači, na kterém se provádějí. Z tohoto důvodu nemohou číst soubory z disku nebo je zapisovat na disk. Výsledná aplikace tudíž neumožňuje vyhotovené grafy uchovávat. Výhodou aplikace je její jednoduché zpřístupnění široké veřejnosti přes internetovou síť.

4.2 Volba datových typů a omezení

4.2.1 Datové typy důležitých proměnných

Aby byla splněna konečnost Ford-Fulkersonova algoritmu (kap. 2.2) jsou proměnné pro ukládání kapacity a toku hran deklarovány jako celočíselné hodnoty typu `integer`. Rovněž označení vrcholů grafu nabývají celočíselných hodnot typu `integer`. Umístění vrcholu na kreslícím plátně představují reálná čísla typu `double` ve směru osy x a ve směru osy y.

4.2.2 Omezení hodnot proměnných

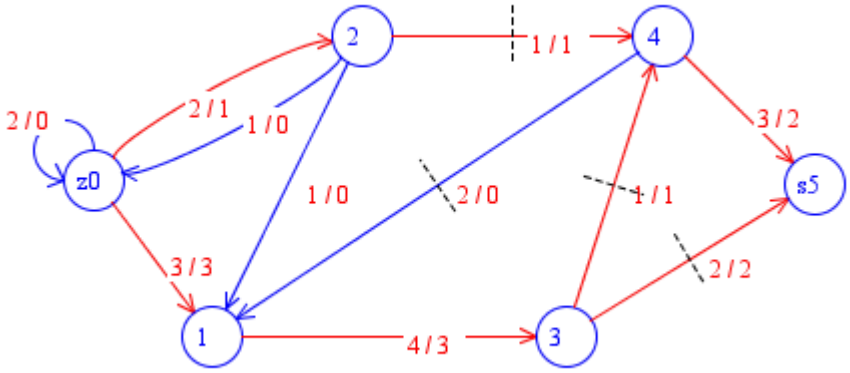
Z definice orientovaného grafu (kap. 1.1.1) vyplývá, že počet vrcholů musí být celé číslo větší nebo rovno jedné, tzn. že minimální počet vrcholů je omezen číslem 1. Nejvyšší počet vrcholů je omezen maximální hodnotou datového typu `integer`.

Rozsah hodnot, kterých mohou nabývat kapacita a tok hran, je dán možnostmi datového typu `integer`.

4.3 Řešení kreslení grafu

Pro vykreslení grafu je v aplikaci vytvořena instance třídy nazvané `GrafTab`, která je odvozena od třídy `JPanel` z balíku `javax.swing` a definuje fyzický panel, do kterého lze umísťovat komponenty. V této třídě přepíši metodu `paintComponent` tak, aby zobrazovala požadovaná data (vrcholy a hrany grafu). Metoda aplikuje metodu pro vykreslení grafu ze třídy `Graf` a tato dále volá

metody pro vykreslení vrcholů a hran, které jsou metodami tříd `Vrchol` a `Hrana`. Pro kreslení jednotlivých prvků (čáry, kružnice, oblouky a popisky) jsou použity nástroje balíku `java.awt.geom` pro zobrazování grafiky na obrazovku. Obrázek 4.1 je ukázkou vykresleného grafu.



Obr. 4.1 Ukázka grafu vykresleného řešenou aplikací.

4.3.1 Vykreslení vrcholů grafu

Vrcholy grafu jsou vykresleny pomocí kružnic se středem v definované poloze každého vrcholu na kreslícím plátně. Poloměr kružnice je zadán konstantou, která slouží zároveň jako měřítko pro všechny konstantní rozměry v obrázku grafu jako jsou vzdálenosti vrcholů, velikosti šipek na hranách, velikost písma v popiscích hran. Změnou této konstanty lze obrázek zvětšovat nebo zmenšovat.

Při posunování vrcholu tažením myší je přepisována hodnota polohy vrcholu na kreslícím plátně a obrázek je překreslen.

4.3.2 Vykreslení hran grafu

Hrany grafu jsou vykreslovány třemi způsoby:

1. Úsečkou ležící na přímce spojující dva středové body vrcholů grafu.
2. Obloukem spojujícím dvě kružnice vrcholů. Použije se v případě, kdy jsou dva vrcholy spojeny dvěma hranami, které jsou opačně orientované.
3. Obloukem, který začíná a končí na kružnici představující vrchol grafu. Tento způsob slouží k vykreslení smyčky (počáteční a koncový vrchol hrany je shodný).

4.3.2.1 První způsob vykreslení hrany - úsečka ležící na přímce spojující dva středové body vrcholů grafu

Čára představující hranu grafu je vykreslena jako úsečka spojující dva vrcholy. Počáteční a koncový bod úsečky leží vždy na obvodu kružnice představující vrchol. Tyto body je nutné určit posunutím ze

středu kružnice na její obvod ve směru přímky spojující oba středy vrcholů. Souřadnice počátečního vrcholu označíme $V_1 = (x_1, y_1)$ a koncového vrcholu $V_2 = (x_2, y_2)$. Způsob řešení v programu:

- Počáteční bod úsečky (označíme $A = (x_A, y_A)$) vznikne posunutím středu počátečního vrcholu (x_1, y_1) ve směru osy x o hodnotu poloměru kružnice vrcholu.
  Souřadnice počátku úsečky: $(x_A, y_A) = (x_1+R, y_1)$.
- Koncový bod úsečky (označíme $B = (x_B, y_B)$) prozatím leží vodorovně s osou x a je posunutý do polohy $(x_A + VZD – 2*R, y_A)$, kde VZD je vzdálenost středů kružnic představující spojované vrcholy. Vzdálenost se vypočte podle Pythagorovy věty:
  $$VZD = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} .$$
  Souřadnice konce úsečky: $(x_B, y_B) = (x_A + VZD – 2*R, y_A)$.
- Vytvoříme objekt úsečky z bodu A do bodu B: $u = AB = (x_A, y_A, x_B, y_B)$.
- Vytvoříme objekty dvou úseček, které vytvoří šipku na konci úsečky představující hranu.
  První úsečka $u_1 = (x_B – R/2, y_B – R/4, x_B, y_B)$.
  Druhá úsečka $u_2 = (x_B – R/2, y_B + R/4, x_B, y_B)$.
- Pokud hranou prochází minimální řez, vytvoříme objekt úsečky představující řez na hraně, která prochází středem úsečky hrany a je k ní kolmá. Označme střed $S_{AB} = (x_S, y_S)$.
  Úsečka řezu $u_R = (x_S - R, y_S, x_S + R, y_S)$.
- Vypočítáme úhel ALFA, který svírá přímka spojující středy vrcholů s osou x.
  $ALFA = arctan (|y_2 - y_1| / |x_2 – x_1|)$.
- Před vykreslením objektů hrany provedeme transformaci otáčení. Uživatelský souřadnicový systém otočíme o úhel $ALFA$ kolem bodu představujícího střed kružnice počátečního vrcholu. Tato transformace je ekvivalentní se třemi po sobě jdoucími transformačními operacemi – posunutí počátku, otáčení o úhel okolo nové polohy počátku a potom posunutí zpět k obnovení původního počátku.
- Provedeme vykreslení připravených grafických objektů hrany u, u_1, u_2, u_R v transformovaném uživatelském souřadnicovém systému a transformaci vrátíme zpět na původní hodnoty.

4.3.2.2 Druhý způsob vykreslení hrany - obloukem spojujícím dvě kružnice vrcholů

Počáteční bod oblouku bude mít hodnotu průsečíku kružnice vrcholu a pomyslné přímky vycházející ze středu kružnice pod úhlem 15° vůči ose x. Posunutí ve směru osy x je $\delta x = R * cos(\pi/12)$ a posunutí ve směru osy y je $\delta y = R * sin(\pi/12)$.

Souřadnice počátku oblouku (bod A) budou $(x_A, y_A) = (x_1 + \delta x, y_1 + \delta y)$ a souřadnice konce oblouku (bod B) budou $(x_B, y_B) = (x_A + VZD – 2* \delta x, y_A)$, kde VZD se určí stejně jako v předchozím bodě 4.3.2.1.

Objekt oblouku je definován pomocí obdélníku opsaného elipse, jíž je oblouk součástí. Zadá se počáteční úhel (*0°*), kde oblouk na elipse začíná a úhlová velikost (počet stupňů, které oblouk pokryje - *180°*). Šířka obdélníku (*w*) je rovna hodnotě *VZD – 2* δx*, kterou jsme vypočítali v předchozím odstavci. Výška obdélníku (*h*) je stanovena konstantně na hodnotu poloměru *R*. Horní levý roh opsaného obdélníku má souřadnice $(x_A, y_A - h/2)$, kde *h* je výška obdélníku.

Obdobným způsobem jako v předchozím bodě 4.3.2.1 jsou připraveny objekty šipky a řezu, je provedena transformace otáčení o úhel vypočítaný na základě polohy středů spojovaných vrcholů a všechny objekty jsou vykresleny.

4.3.2.3 Vykreslení hrany obloukem (smyčka)

Hrana, která představuje smyčku je vykreslována obloukem, který je částí kružnice. Oblouk je zadán pomocí obdélníku, do kterého se oblouk vepíše a pomocí počátečního úhlu (zadáno konstantně *0°*), ve kterém oblouk začíná a úhlové velikosti (počet stupňů, které oblouk pokryje, zadáno konstantně *270°*). Horní levý roh opsaného obdélníku má souřadnice konstantně posunuty vůči středovému bodu vrcholu do polohy $(x_1 - 2*R, y_1 - 2*R)$, kde *R* je opět poloměr kružnice vrcholu. Šířka a výška obdélníku je rovna hodnotě *2*R*.

4.3.2.4 Vykreslení popisku hrany

V popisku každé hrany je uvedena hodnota kapacity (c) a toku (f) ve formátu c/f. Popisek je vykreslen do bílého obdélníku, jehož levý horní roh je umístěn ve středovém bodě úsečky nebo oblouku představující hranu. Délka obdélníku závisí na velikosti délky popisku. Popisek se vykresluje zároveň s každou hranou.

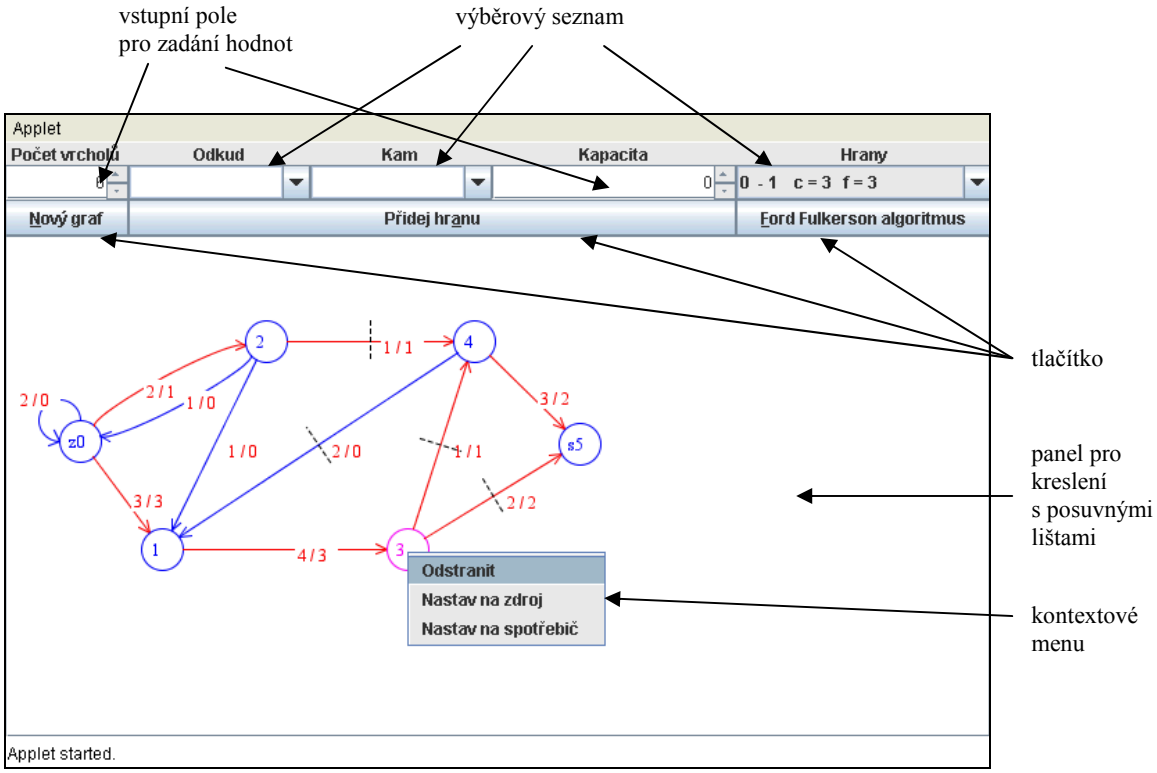
Obdélník pro umístění popisku hrany zároveň slouží k určení místa pro zaměření hrany kurzorem myši.

4.4 Řešení grafického uživatelského rozhraní

Postup vytvoření grafického uživatelského rozhraní programované aplikace je:

1. Vytvoření třídy kontejner, která bude obsahovat jednotlivé komponenty grafického uživatelského rozhraní. Jako základní kontejner slouží třída `FFaplikace`, která je podtřídou třídy `javax.swing.JApplet`. Třída `JApplet` navíc implementuje sadu metod, které tvoří rozhraní mezi appletem a webovým prohlížečem.
2. Volba správce rozvržení pro tento kontejner. V aplikaci je zvoleno použití správce rozmístění typu `GridBagLayout`, který prvky rozmístí do variabilní pravoúhlé mřížky, ve které se velikost každého prvku může lišit. Komponenta je umístěna v dané buňkové poloze mřížky určené souřadnicemi. Může zabírat více než jednu buňku řádku nebo sloupce, ale vždy zabírá pravoúhlou skupinu buněk.

3. Vytvoření komponent a jejich přidání do okna rámečku kontejneru `JApplet`. Vytvořené komponenty a jejich umístění jsou patrné z obrázku 4.2.
4. Vytvoření objektů pro „naslouchání“, které budou sledovat a reagovat na události očekávané každou komponentou. Je potřeba přidělit objekty pro naslouchání vhodným komponentám.



Obr. 4.2 Ukázka grafického uživatelského rozhraní aplikace.

4.5 Vlastní implementace

Výsledný applet obsahuje třídu `FFAplikace`, která vytvoří grafické uživatelské rozhraní, třídu `GrafTab`, která vytvoří kreslící plátno a vykreslí graf a třídu `HandlerMysi`, která sleduje a obsluhuje události myši.

Třída `FFAplikace` obsahuje metody `init()` a `actionPerformed()`. Metoda `init()` přepisuje metodu `init()` ve třídě `JApplet` a inicializuje grafické uživatelské rozhraní. Bude volána WWW prohlížečem ve chvíli, kdy applet začne svou činnost. Tato metoda vytvoří instanci třídy `Graf` - vytvoří referenční graf. Dále vytvoří objekt pro „naslouchání“ ze třídy `HandlerMysi`, aby sledoval a obsluhoval události generované myší v oblasti obrázku grafu.

Metoda `actionPerformed()` je metodou, kterou volá obsluha událostí pokaždé, když dojde ke klepnutí myší na některé tlačítko aplikace nebo k výběru některé položky z kontextového menu. Metoda provede akce podle vybrané komponenty.

Objekt třídy `GrafTab` je panel, do kterého se provede vykreslení obrázku grafu. Třída je zděděna ze třídy `javax.swing.JPanel`. Vykreslení zajistí metoda `paintComponent`, která volá metodu `paintGraf` objektu grafu. Panel je umístěn do posouvací plochy `scrollPane`, která umožní horizontální a vertikální posouvání v případě, že bude obrázek větší než zobrazené okno.

Vytvořený objekt třídy `HandlerMysi` kontroluje, zda je myší zaměřený vrchol nebo hrana grafu. Umožňuje přesouvání vrcholů grafu po kreslícím plátně pomocí držení levého tlačítka myši nebo přidávání hran „klikáním“ levým tlačítkem myši na vrcholy a otevírá příslušné příruční nabídky (kontextová menu) pro vrchol, hranu nebo mimo obrázek grafu.

Ve třídách `Graf`, `Vrchol` a `Hrana` jsou implementovány metody pro práci s grafem, které byly popsány v objektovém návrhu v kap. 3.3.

5 Maximální tok v rovinné síti

Velmi rychlé určení maximálního toku dovolují rovinné transportní sítě.

5.1 Rovinné sítě

Rovinnou sítí je taková síť, kterou lze do roviny zakreslit tak, že zdroj je zcela vlevo, spotřebič je zcela vpravo a všechny další uzly a hrany jsou uvnitř tohoto pásu. Libovolné dvě křivky přiřazené různým hranám mají společné nejvýše dva body, a to své krajní body.

Vlastnosti rovinného grafu se netýkají konkrétního nakreslení, ale pouze jeho existence nebo dokonce jen možnosti nějaké rovinné nakreslení vytvořit.

5.2 Hledání maximálního toku v rovinné síti

Před vlastním hledáním maximálního toku je třeba vhodným algoritmem pro zadaný graf rozhodnout, zda je to rovinný graf a případně nalézt rovinné nakreslení.

Výpočet maximálního toku zahájíme s nulovým tokem. Jako zlepšující se vždy hledá „nejhořejší“ cesta, která obsahuje jen hrany orientované ve směru od zdroje ke spotřebiči. Tok hranou se pak pouze zvyšuje a v každém průchodu hlavním cyklem se alespoň jedna hrana stane nasycenou.

K nalezení zlepšující cesty lze použít orientované hledání do hloubky, při kterém z každého vrcholu postupujeme nejprve hranou, která není nasycena a která vede nejvíce vlevo. Pro zjišťování kapacity zlepšující cesty a pro zvyšování toku lze použít stejný postup jako ve Ford-Fulkersonově algoritmu uvedeném v kap. 2.

Výpočet končí, když už neexistuje žádná orientovaná zlepšující cesta.

5.3 Časová složitost

Pro rychlost algoritmu je důležité, že v žádné hraně tok nikdy neklesne, neboť nepoužíváme hrany vzad. Každá zlepšující cesta způsobí nasycení alespoň jedné hrany. Z důvodu omezenosti počtu hran rovinných grafů (bez smyček) je časová složitost tohoto algoritmu $O(|V|^2)$, kde $|V|$ je počet vrcholů grafu.

6 Závěr

Důkladně jsem prostudoval literaturu týkající se grafů, sítí a jejich aplikací. Navrhl jsem objektový model grafu. Následnou implementací v programovacím jazyku Java vznikl nástroj, ve kterém lze zadat transportní síť (orientovaný graf s ohodnocenými hranami), která je graficky zobrazena na obrazovce. V aplikaci jsem implementoval Ford-Fulkersonův algoritmus pro výpočet maximálního toku v zadané síti. Po spuštění a provedení výpočtu je v obrázku graficky zobrazen maximální tok a minimální řez. Program je napsán jako Java applet a je přístupný na veřejných internetových stránkách www.ffaplikace.php5.cz. Zdrojový kód programu včetně programové dokumentace naleznete na přiloženém CD.

Ověření správnosti výpočtu maximálního toku jsem prováděl na grafech uváděných v literatuře včetně správného řešení, případně jsem správný výsledek ověřil výpočtem na papíře. V programu jsem nezjistil žádné nedostatky.

Program lze snadno doplnit o další různé algoritmy pracující s grafem, z nichž většinu je vhodné graficky zobrazovat (zadávání a vykreslování grafu je již vyřešeno). Mezi takové algoritmy mohou patřit různé varianty prohledávání grafu, úlohy o cestách a sledech, barvení grafu, nalezení minimální kostry souvislého grafu apod. Dále je také možné implementovat rychlejší algoritmy pro hledání maximálního toku. Jednou z možností je použití rovinné transportní sítě (kap. 5) po předchozím převedení zadaného grafu na rovinný.

Literatura

[1] Demel, J.: Grafy a jejich aplikace. Praha, Nakladatelství Akademie věd ČR, 2002.

[2] Hliněný, P.: Diskrétní matematika. Skripta VŠB – Technická univerzita Ostrava, 2004-2005.

[3] Horton, I.: Java 5. Praha, Neocortex, spol. s r. o., 2005.

Přílohy

A Návod pro používání programu

A1 Přístup k programu

Aplikace je přístupná na internetových stránkách www.ffaplikace.php5.cz. Je zpracována jako Java applet. Stránku lze prohlížet pomocí libovolného prohlížeče s rozšířením pro Java 2. Důležitou podmínkou správného zobrazení stránky je, aby prohlížeč spouštění appletů povoloval.

Pokud se applet na stránce správně nezobrazí, zkontrolujte, zda je spouštění appletů povoleno. Případně je nutné nainstaloval Java Plug-in (J2SE JRE nebo JDK) ze stránek http://java.sun.com/javase/downloads/index.jsp. Nainstalujte poslední verzi, případně 1.4.2 nebo vyšší.

A2 Popis zobrazení grafu

V hlavním kreslícím okně programu je vyobrazen graf. Vrcholy jsou zobrazeny kružnicemi s označením uvnitř. Zdrojový vrchol je označen písmenem „z“, spotřebič je označen „s“. Hrany představují úsečky nebo oblouky spojující vrcholy. U každé hrany se nachází její popis ve tvaru „c/f“, kde „c“ je hodnota kapacity hrany a „f“ je hodnota toku hranou.

Po spuštění Ford-Fulkersonova algoritmu je maximální tok zvýrazněn červenou barvou a minimální řez je vykreslen přerušovanou černou čárou na příslušných hranách.

A3 Ovládání programu

`Výpočet maximálního toku v grafu:`

- Stiskněte tlačítko Ford-Fulkerson algoritmus.

`Nový graf:`

- Zadejte pole Počet vrcholů a stiskněte tlačítko Nový graf.

`Přidat hranu:`

- *1. způsob*: Zadejte pole Odkud, Kam, Kapacita a stiskněte tlačítko Přidat hranu.
- *2. způsob*: Levým tlačítkem myši klikněte na počáteční vrchol hrany, dalším kliknutím levým tlačítkem myši označte koncový vrchol. V dialogu zadejte kapacitu.

`Přidat vrchol:`

- Klikněte na volné ploše obrázku pravým tlačítkem. V kontextové nabídce zvolte Přidat vrchol.

Zaměřit vrchol:

- Myší přejeďte dovnitř kružnice vrcholu. Zaměřený vrchol změní barvu. Pravým tlačítkem lze vyvolat kontextové menu.

Zaměřit hranu:

- Myší najeďte na popis hrany. Zaměřená hrana změní barvu. Pravým tlačítkem lze vyvolat kontextové menu.

Pohyb vrcholu:

- Zaměřte vrchol, při stisknutém levém tlačítku myši s vrcholem pohybujte.

Odstranit hranu:

- Zaměřte hranu, pravým tlačítkem vyvolejte kontextové menu, zvolte Smazat.

Změnit kapacitu hrany:

- Zaměřte hranu, pravým tlačítkem vyvolejte kontextové menu, zvolte Změnit kapacitu.

Odstranit vrchol:

- Zaměřte vrchol, pravým tlačítkem vyvolejte kontextové menu, zvolte Smazat.

Nastavit zdroj:

- Zaměřte vrchol, pravým tlačítkem vyvolejte kontextové menu, zvolte Nastavit na zdroj.

Nastavit spotřebič:

- Zaměřte vrchol, pravým tlačítkem vyvolejte kontextové menu, zvolte Nastavit na spotřebič.

B Programová dokumentace

Programová dokumentace je automaticky vytvořena nástrojem `javadoc.exe` ze zdrojových souborů programu. Zde je uvedena její část. Celý dokument se nachází na přiloženém CD a lze ho prohlížet v libovolném prohlížeči HTML souborů.

B.1 Package fordfulkerson

Class Summary	
FFaplikace	Hlavní třída aplikace, vytvoří grafické uživatelské rozhraní.
Graf	Třída Graf, její instance představuje graf.
Hrana	Třída Hrana, její instance představují hrany grafu.
Vrchol	Třída Vrchol, její instance představují vrcholy grafu.

B.2 fordfulkerson

Class FFaplikace

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └java.awt.Panel
              └java.applet.Applet
                  └javax.swing.JApplet
                      └fordfulkerson.FFaplikace
```

Hlavní třída aplikace, vytvoří grafické uživatelské rozhraní, vytvoří graf - instanci třídy Graf.

Nested Class Summary	
class	**FFaplikace.GrafTab** Třída vytvoří kreslící plátno a vykreslí graf.
class	**FFaplikace.HandlerMysi** Třída pro objekt, který sleduje a obsluhuje události generované myší v oblasti obrázku grafu.

Constructor Summary

FFaplikace()	

Method Summary

void	**actionPerformed**(java.awt.event.ActionEvent e) Obsluha tlačítek: "novyButton" - vytvoří nový graf se zadaným počtem vrcholů, "pridejButton" - přidá zadanou novou hranu, "fordFulkButton" - spustí výpočet pro nalezení maximalního toku, Obsluha kontextových menu: "vymazPol" - odstraní hranu, "zmenPol" - změní kapacitu hrany, "pridejVrchol" - přidá nový vrchol, "vymazVrchol" - odstraní vrchol, "zmenZdroj" - změní vrchol na zdroj, "zmenSpotrebic" - změní vrchol na spotřebič,
void	**init**() Metoda vytvoří graf. Inicializuje grafické uživatelské rozhraní.

B.3 fordfulkerson

Class FFaplikace.GrafTab

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └javax.swing.JComponent
              └javax.swing.JPanel
                  └fordfulkerson.FFaplikace.GrafTab
```

Třída vytvoří kreslící plátno a vykreslí graf.

Constructor Summary

FFaplikace.GrafTab() Konstruktor kreslícího plátna.	

Method Summary

void	**paintComponent**(java.awt.Graphics g) Metoda pro vykreslení grafu do kreslícího plátna.

B.4 fordfulkerson

Class FFaplikace.HandlerMysi

```
java.lang.Object
  └javax.swing.event.MouseInputAdapter
      └fordfulkerson.FFaplikace.HandlerMysi
```

Třída pro objekt, který sleduje a obsluhuje události generované myší v oblasti obrázku grafu. Vytvořený objekt třídy HandlerMysi kontroluje, zda je myší zaměřený vrchol nebo hrana grafu. Umožňuje přesouvání vrcholů grafu po kreslícím plátně pomocí držení levého tlačítka myši nebo přidávání hran „klikáním“ levým tlačítkem myši na vrcholy a otevírá příslušné příruční nabídky (kontextová menu) pro vrchol, hranu nebo mimo obrázek grafu.

Constructor Summary

FFaplikace.HandlerMysi()	

Method Summary

void	**mouseClicked**(java.awt.event.MouseEvent e) Metoda reagující na kliknutí tlačítka myši.
void	**mouseDragged**(java.awt.event.MouseEvent e) Metoda kontrolující tažení se stlačeným tlačítkem myši.
void	**mouseMoved**(java.awt.event.MouseEvent e) Metoda kontrolující pohyb myši.
void	**mousePressed**(java.awt.event.MouseEvent e) Metoda kontrolující stlačení tlačítka myši.
void	**mouseReleased**(java.awt.event.MouseEvent e) Metoda kontrolující puštění tlačítka myši.

B.5 fordfulkerson

Class Graf

```
java.lang.Object
  └fordfulkerson.Graf
```

Třída Graf, její instance představuje graf. Objekty představující vrcholy grafu jsou instance třídy Vrchol a jsou uloženy ve vectoru vrcholy. Konstruktor vytvoří graf. Je implementován Ford-Fulkersonův algoritmus pro nalezení maximálního toku a minimálního řezu v síti. Jsou implementovány metody pro vykreslení grafu a pro zaměření jeho komponent myší.

Constructor Summary

Graf()	Přetížený konstruktor, vytvoří a načte referenční graf.
Graf(int pocetVrcholu)	Konstruktor, vytvoří nový graf.

Method Summary

void	**ford fulkerson**() Metoda vypočte maximální tok grafem.
void	**grafFree**() Uvolní graf z paměti, zničí všechny hrany a vrcholy.
Hrana	**grafHranaZamerena**(java.awt.event.MouseEvent e) Vrací hranu grafu, která je zaměřená myší.
Vrchol	**grafVrcholZameren**(java.awt.event.MouseEvent e) Vrací vrchol grafu, který je zaměřený myší.
void	**nactiGraf**() Naplní referenční graf hranami.
void	**nastavSpotrebic**(Vrchol zamereny) Zaměřený vrchol nastaví na Spotřebič.
void	**nastavZdroj**(Vrchol zamereny) Zaměřený vrchol nastaví na Zdroj.
void	**odstranHranu**() Odstraní zaměřenou hranu z grafu.
void	**odstranVrchol**(Vrchol odstranenyVrchol) Odstraní vrchol z grafu.
void	**paintGraf**(java.awt.Graphics2D g2D) Vykreslení grafu na plátno.
void	**pridejVrchol**(java.awt.geom.Point2D.Double stred)

	Do grafu přidá nový vrchol.
void	**souradniceVrcholu**() Určí souřadnice vrcholu grafu pro vykreslovaní na plátno.
boolean	**vlozHranu**(int pv, int kv, int c) Přidá novou hranu příslušnému vrcholu.
void	**vynulujTok**() Na všech hranách grafu nastaví tok na hodnotu 0 (inicializační).
void	**zobrazGraf**() Vypíše hrany grafu na konzoly.
void	**zvysKapacituHrany**(int pv, int kv, int c) Zvýší kapacitu hrany o zadanou hodnotu.

B.6 fordfulkerson

Class Vrchol

java.lang.Object
  └ **fordfulkerson.Vrchol**

Třída Vrchol, její instance představují vrcholy grafu. Objekty hran vycházejících z vrcholu jsou uchovávány ve vectoru hrany. Konstruktor vytvoří vrchol. Inicializuje vector hrany. Jsou implementovány metody pro práci s vectorem hrany. Je implementována metoda pro vykreslení vrcholu a pro jeho zaměření myší.

Constructor Summary

Vrchol(java.awt.Color barva) Konstruktor, vytvoří objekt vrcholu, inicializuje vector pro hrany, souřadnice středu vrcholu zatím (0,0) nastaví metoda grafu
Vrchol(java.awt.geom.Point2D.Double stred, java.awt.Color barva) Přetížený konstruktor, vytvoří objekt vrcholu, inicializuje vector pro hrany, nastaví souřadnice pro vykreslování na zadanou pozici

Method Summary

boolean	**add**(Hrana hr)

	Přidává novou hranu do vektoru hrany.
void	**del**(Hrana hr) Odstraní hranu z vectoru hrany.
int	**getKap**() Vrací hodnotu proměnné kap (volná kapacita hrany na zlepšující cestě).
int	**getPred**() Vrací hodnotu proměnné pred (předchůdce na zlepšující cestě).
int	**getSmer**() Vrací hodnotu proměnné smer (orientace na zlepšující cestě).
int	**getStav**() Vrací hodnotu
java.awt.geom.Point2D.Double	**getStred**() Vrací hodnotu souřadnice středu vrcholu na kreslícím plátně.
java.util.Iterator	**iterator**() Získá iterátor pro vektor hrany
Hrana	**najdiHranu**(int v) Nalezne hranu podle koncového vrcholu.
void	**nastavHodnoty**(int s, int p, int sm, int d) Metoda nastavuje hodnoty všech pomocných proměnných.
void	**nulujHodnoty**() Metoda vynuluje pomocné proměnné.
void	**paintVrchol**(java.awt.Graphics2D g2D, double polomer, java.lang.String str) Vykreslí vrchol na kreslící plátno.
void	**setBarva**(java.awt.Color c) Nastavuje barvu
void	**setStred**(double x, double y) Nastavuje hodnotu souřadnice středu vrcholu pro vykreslování na plátno
int	**shodneHrany**(int v) Metoda vyhledá počet hran ve vectoru hrany se shodným koncovým vrcholem.
int	**size**() Vrací velikost vectoru hrany - počet hran vycházejících z vrcholu.

void	**vrcholFree**() Zruší vector hrany.
boolean	**zamerenyVrchol**(double polomer, java.awt.event.MouseEvent e) Metoda zjistí, zda je vrchol zaměřený myší.

B.7 fordfulkerson

Class Hrana

java.lang.Object
  └ **fordfulkerson.Hrana**

Třída Hrana, její instance představují hrany grafu. Konstruktor vytvoří hranu. Jsou implementovány metody pro vykreslení hrany a pro zaměření hrany myší.

Constructor Summary

Hrana(int kv, int c, int poz, java.awt.Color barva) Konstruktor, vytvoří objekt hrany.

Method Summary

java.awt.Color	**getBarva**() Vrací nastavenou barvu hrany.
int	**getC**() Vrací hodnotu kapacity hrany.
int	**getF**() Vrací hodnotu toku hrany.
int	**getKv**() Vrací index koncového vrcholu hrany.
int	**getPoz**() Vrací hodnotu proměnné, která uchovává počet hran do stejného vrcholu.
void	**paintHrana**(java.awt.Graphics2D g2D, double polomer, java.awt.geom.Point2D.Double odkud, java.awt.geom.Point2D.Double kam, boolean rez) Metoda vykreslí hranu na kreslící plátno.

void	**setBarva**(java.awt.Color c) Nastavuje barvu hrany pro vykreslování.
void	**setC**(int x) Nastavuje hodnotu kapacity hrany.
void	**setF**(int x) Nastavuje hodnotu toku hrany.
void	**setKv**(int x) Nastavuje index koncového vrcholu hrany.
void	**setPoz**(int x) Nastavuje hodnotu proměnné, která uchovává počet hran do stejného vrcholu.
java.lang.String	**toString**() Převede hodnoty index koncového vrcholu, kapacity a toku hrany na řetězce a vytvoří z nich složený řetězec.
boolean	**zamerena**(double polomer, java.awt.geom.Point2D.Double odkud, java.awt.geom.Point2D.Double kam, java.awt.event.MouseEvent e) Metoda zjistí, zda je hrana zaměřena myší.

C Obsah přiloženého CD

K bakalářské práci je přiložen CD-ROM obsahující zdrojové kódy programu, soubory programové dokumentace, elektronickou verzi této bakalářské práce a zdrojový text HTML stránky, na kterou je umístěn naprogramovaný Java applet. V kořenovém adresáři je soubor `readme.txt`, který obsahuje popis obsahu CD.

C.1 Struktura adresářů

- `source` - V tomto adresáři jsou umístěny zdrojové kódy programu v jazyce Java.
- `doc` - Tento adresář obsahuje dokumentaci zdrojových kódů ve formátu Javadoc, tedy ve formě HTML stránek.
- `pdf` - V tomto adresáři je umístěna elektronická verze této bakalářské práce.
- `applet`- V tomto adresáři jsou umístěny soubory potřebné pro zobrazení HTML stránky s naprogramovaným Java appletem.

C.2 Spuštění programu

Aplikace je naprogramována jako Java applet. Pro spuštění zobrazte v HTML prohlížeči soubor `index.html` z adresáře `applet` umístěného na CD. Alternativou je zobrazení stránky www.ffaplikace.php5.cz z veřejné internetové sítě.

Stránku lze prohlížet pomocí libovolného prohlížeče s rozšířením pro Java 2. Důležitou podmínkou správného zobrazení stránky je, aby prohlížeč spouštění appletů povoloval.

Pokud se applet na stránce správně nezobrazí, zkontrolujte, zda je spouštění appletů povoleno. Případně je nutné nainstaloval Java Plug-in (J2SE JRE nebo JDK) ze stránek http://java.sun.com/javase/downloads/index.jsp. Nainstalujte poslední verzi, případně 1.4.2 nebo vyšší.