Kapitola 1

Grafy, stromy

TODO: nejaky hezky obrazky? V této kapitole definujeme graf jakožto matematickou strukturu, popíšeme základní pojmy týkající se grafů a nastíníme možné vztahy mezi grafem a maticí. Dále definujeme strom, jakožto speciální případ grafu. Terminologie je převzata z [3]

1.1 Základní grafová terminologie

Definice 1. Mějme množinu V a množinu $E = \{\{u, v\} \,|u, v \in V\}$. Uspořádanou dvojici $G := (V, E)$, nazveme neorientovaný graf. Množinu V nazýváme množinou vrcholů grafu G, jejím prvkům říkáme vrcholy, množinu E nazýváme množinou hran grafu G, jejím prvkům říkáme hrany. Prvky hrany e označujeme jako vrcholy incidentní hraně e nebo koncové body hrany e. Říkáme, že hrana $e = \{v, w\}$ spojuje vrcholy v a w.

Pokud neuvažujeme hrany jako nejvýše dvouprvkové množiny vrcholů, ale jako uspořádané dvojice (u, v), nazýváme odpovídající graf **orientovaný**. Obvykle uvažujeme orientované a neorientované grafy zvlášť, ale je možné uvažovat i jejich kombinaci. Graf, v němž se vyskytují jak orientované tak neorientované hrany, nazýváme **smíšený**. Řekneme, že neorientovaný graf G je **úplný**, pokud $\forall u, v \in V$ $(\{u, v\} \in E)$.

Povšimněme si, že v definici grafu není vyloučen případ, kdy jsou oba koncové body hrany shodné. Hrana je pak jednoprvkovou množinou a nazýváme ji **smyčkou** v grafu.

Poznámka 1. V neorientovaném grafu $G = (V, E)$ platí, že jeho množina hran E je podmnožinou $\binom{V}{2} \cup V$, kde $\binom{V}{2}$ značí množinu všech dvouprvkových podmnožin množiny V. V orientovaném grafu $H = (W, F)$ je F podmnožinou množiny $W \times W$, tj. všech uspořádaných dvojic vrcholů z W.

Stupněm vrcholu $v \in V$ rozumíme počet vrcholů spojených s vrcholem v, značíme $d(v)$. Množinu všech vrcholů, které jsou v grafu G spojeny s vrcholem v značíme $\mathrm{adj}_G(v)$.

Definice 2. Podgrafem grafu G nazveme libovolný graf $H = (W, F)$ který splňuje: $W \subseteq V$, $F \subseteq E$ a všechny vrcholy incidentní hranám z F náleží do W. Úplný podgraf grafu G nazýváme **klikou** v grafu G. Podgrafem grafu G **indukovaným** množinou vrcholů W

nazveme takový podgraf G, který obsahuje všechny hrany grafu G, jejichž oba koncové body náleží do W, značíme $G(W)$.

TODO: Potrebuju ohodnoceny, vazeny?

Definice 3. Mějme graf $G = (V, E)$ a zobrazení $\omega : V \to \mathbb{R}$, resp. $c : E \to \mathbb{R}$. Přidáním zobrazení ω, resp. c ke grafu G dostaneme graf, který nazýváme **ohodnocený**, resp. **vážený** reálným ohodnocením.

Definice 4. TODO: doplnit definici cesty bez cyklu!, cyklu, delka cesty

Existuje-li mezi libovolnými dvěma vrcholy grafu cesta, řekneme, že graf je **souvislý**. **Vzdáleností** dvou vrcholů v souvislém grafu $G = (V, E)$ nazveme minimální délku cesty mezi těmito dvěma vrcholy. Vzdáleností vrcholu $v \in V$ od množiny vrcholů $W \subset V$ nazveme minimální vzdálenost mezi vrcholem v a libovolným vrcholem náležícím do W.
TODO: potrebuju bipartitni? TODO: potrebuju ctvercovou sit? TODO: potrebuju faktorgraf?

1.2 Strom

Nyní zaveďme základní pojmy týkající speciální třídy grafů nazývané stromy [3]

Definice 5. Stromem $T = (V, E)$ nazveme konečný souvislý neorientovaný graf bez cyklů. Pokud navíc v grafu T vyznačíme bod $r \in V$, nazýváme uspořádanou dvojici (T, r) **kořenovým stromem** a bod r nazveme kořenem tohoto stromu.

Z definice stromu je patrné, že každý vrchol v kořenového stromu (T, r) spojuje s kořenem tohoto stromu právě jedna cesta. Vrcholy ležící na této cestě nazveme **předchůdci** vrcholu v. Předchůdce vrcholu v různé od v nazýváme **vlastními předchůdci** vrcholu v. Vrcholy, jejichž předchůdcem je vrchol v, nazýváme **následníky** vrcholu v. Vrcholy bez následníků nazýváme **listy stromu** T, vrcholy alespoň s jedním následníkem nazýváme **vnitřní vrcholy** stromu.

Definice 6. Podstromem stromu T určeným vrcholem v nazveme indukovaný podgraf stromu T tvořený vrcholem v a a všemi jeho následníky.

1.3 Vztah grafu a matice

Grafy a matice spolu úzce souvisí, což nám umožňuje převádět problémy na maticích na problémy na grafech a naopak. Nezanedbatelným praktickým důsledkem jejich vzájemného vztahu je i možnost používat grafové algoritmy při řešení některých maticových úloh, především může být tento přístup výhodný pro řídké matice. Dělení grafů může posloužit například při snaze o paralelizaci rozkladu matice.

Neorientovaný graf $G = (V, E)$ s vrcholy $V = v_1, \ldots, v_m$ a hranami $E = e_1, \ldots, e_n$ Tento graf lze reprezentovat pomocí matice dvěma základními způsoby. **Maticí sousednosti**, neboli adjacenční matici, nazveme matici A_G o rozměrech $m \times m$, jejíž prvek na pozici (i, j) je definován jako:

$$(A_G)_{i,j} := \begin{cases} 1 & \text{existuje-li hrana spojující vrcholy } v_i, v_j \\ 0 & \text{jinak} \end{cases}$$

Maticí incidence grafu G nazveme matici o rozměrech $m \times n$ definovanou následovně:

$$(\bar{A}_G)_{i,j} := \begin{cases} 1 & \text{je-li } v_i \text{ koncovým vrcholem hrany } e_j \\ 0 & \text{jinak} \end{cases}$$

V kapitole refspektral TODO: bude ref spektral? budeme potřebovat **Laplaceovu matici** Q grafu G, která je definována následovně:

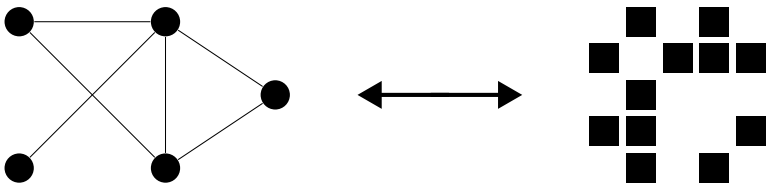
$$Q_{ij} := \begin{cases} -1 & \text{pro } i \neq j, (v_i, v_j) \in E \\ 0 & \text{pro } i \neq j, (v_i, v_j) \notin E \\ d(i) & \text{pro } i = j \end{cases}$$

Laplaceovu matici Q lze tedy vyjádřit jako $Q = D - A_G$, kde D značí diagonální matici se stupni jednotlivých vrcholů na diagonále.

Pokud chceme reprezentovat matici pomocí grafu, většinou nám stačí zachytit její strukturu. V takovém případě můžeme pro popis obecně nesymetrické matice A o rozměrech $n \times n$ použít orientovaný graf s množinou vrcholů $V = v_1, \ldots, v_n$ a množinou hran $E = \{(v_i, v_j) | a_{ij} \neq 0\}$. V případě, že je matice A symetrická, můžeme ji analogickým způsobem reprezentovat pomocí neorientovaného grafu. Pokud bychom chtěli do grafu zanést i numerické hodnoty jednotlivých prvků matice, museli bychom použít ohodnocený graf.

Poznámka 2. Pro jednoduchost zaveďme následující terminologii. Říkáme, že graf G **odpovídá** matici A právě tehdy, když matice A je maticí sousednosti grafu G.

TODO: rozhodnout, jestli tam tohle dávat Pro reprezentaci ne nutně čtvercové matice B o rozměrech $m \times n$ můžeme také použít bipartitní graf $G = (R, B, E)$ pro nějž platí $|R| = m$, $|B| = n$ a $E = \{(i, j') \mid i \in R, j' \in B, a_{ij} \neq 0\}$. TODO: end



Obrázek 1.1: Příklad grafu a jemu odpovídající struktury matice

Kapitola 2

Dělení grafů

TODO: doplnit

Kapitola 3

Rozklady matic

TODO: doplnit

Kapitola 4

Další parametry pro dělení grafu

Dělení grafu lze využít jako nástroj pro paralelizaci problémů, které lze převést na úlohy na grafech. Vzhledem ke vztahu matic a grafů popsanému výše v odstavci 1.3 je Choleského rozklad popsaný v kapitole TODO: ref,

Jak bylo zmíněno, při dělení grafu podle klasické definice bereme ohled pouze na dvě kritéria: na velikost separátoru a vyváženost jednotlivých částí rozdělení. V případě, že dělení grafu využíváme jako součást komplexnějšího algoritmu, například pro paralelizaci nějaké maticové úlohy, nemusí být rozdělení vytvořené s ohledem na tato kritéria optimální. V takovém případě je nutné brát ohled na to, jaké operace budeme na výsledných podgrafech provádět a v závislosti na tom specifikovat nová kritéria.

Pro dělení grafu s ohledem na více než dvě základní kritéria popsaná výše existují dva základní přístupy. Můžeme hledat rozdělení grafu optimalizující všechna kritéria zároveň (apriorní přístup), nebo můžeme graf rozdělit pomocí některého ze základních algoritmů pro dělení grafu a poté vzniklé rozdělení vylepšit tak, abychom optimalizovali vyváženost operací (aposteriorní přístup).

V této práci se zaměříme na situaci, kdy je dělení grafů použito jako nástroj pro vyvažování počtu operací na jednotlivých oblastech při paralelizaci Choleského rozkladu. Požadujeme tedy, aby počet operací potřebný pro Choleského rozklad na jednotlivých oblastech byl v optimálním případě stejný. Zřejmě nám takto vzniká kritérium, které není zahrnuto v kritériích podle klasické definice - ta totiž neberou v potaz počet hran v grafech odpovídajících jednotlivým oblastech, neboli v řeči matic počet nenulových prvků matic sousednosti jednotlivých podgrafů. Čas pro vypočítání rozkladu na jednotlivých oblastech může být kvůli tomu výrazně odlišný. Námi navržený algoritmus bude využívat aposteriorního přístupu, nejprve tedy nalezneme suboptimální řešení vzhledem k základním kritériím a poté budeme přesouvat vrcholy mezi jednotlivými částmi rozdělení s cílem vylepšit jej vzhledem k Choleského rozkladu.

Mějme graf G a jeho rozdělení na podgrafy $G_1, \ldots, G_k$ s maticemi sousednosti $A_{G_1}, \ldots, A_{G_k}$. Provedeme Choleského rozklad matic A_{G_1}, A_{G_2}, označme

$$A_{G_i} = L_{G_i} L_{G_i}^T \text{ pro } i = 1, 2.$$

Pokud existují $i, j \in \{1, \ldots, k\}$ tak, že matice $L_{G_i}^T$ a $L_{G_j}^T$ budou mít výrazně různý počet

nenulových prvků, je zřejmé, že toto rozdělení grafu G je nevhodné pro napočítávání Choleského rozkladu na oblastech vzhledem k vyváženosti na počet operací. Někdy pro vyvážení rozdělení za účelem Choleského rozkladu na oblastech stačí vhodně přečíslovat vrcholy grafu G a tím změnit matice sousednosti odpovídající jednotlivým podgrafům.TODO: Budeme to dělat takhle, nebo budeme číslovat vrcholy až po rozdělení?

Poznámka 3. Mějme soustavu lineárních algebraických rovnic $Ax = b$ a hledejme její řešení pomocí Choleského rozkladu této matice $A = LL^T$ za pomoci dělení grafů. Pro vyřešení soustavy potřebujeme vyřešit rovnice

$$\begin{aligned} Ly &= b \\ L^T x &= y \end{aligned}$$

Pokud rozdělíme graf příslušný matici A na k částí a provedeme rozklad matic $A_1 = L_1 L_1^T, \dots A_k = L_k L_k^T$ odpovídajících vzniklým podgrafům a matice $A_S = L_S L_S^T$ odpovídající vrcholovému separátoru, pak lze soustavu $Ly = b$ napsat ve tvaru

$$\begin{pmatrix} L_1 & & & & \\ & L_2 & & & \\ & & \ddots & & \\ & & & L_k & \\ S_1 & S_2 & & S_k & L_S \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \\ y_S \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \\ b_S \end{pmatrix}$$

a tedy

$$\begin{aligned} y_1 &= L_1^{-1} b_1 \\ y_2 &= L_2^{-1} b_2 \\ &\vdots \\ y_k &= L_k^{-1} b_k \\ y_S &= L_1^{-1}(b_S - S_1 y_1 - S_2 y_2 - \dots - S_k y_k) \end{aligned}$$

Je vidět, že zatímco výpočet $y_1, \dots, y_k$ může probíhat paralelně, výpočet y_S je závislý na $y_1, \dots, y_k$. Je tedy vhodné, aby separátor byl malý. Soustavu $L^T x = y$ poté vyřešíme obdobně.

Jak vidíme, vyvstávají nám při dělení grafu s cílem vypočítávat Choleského rozklad na oblastech dvě základní otázky:

1. Jak najít očíslování grafu takové, aby Choleského rozklad na jednotlivých podgrafech byl vyvážený? TODO: Přikláněl bych se k tomu rovnat až výsledné podgrafy
2. Jak určit počet nenulových prvků v Choleského rozkladu matice sousednosti jednotlivých podgrafů?

Způsob hledání odpovědí na tyto dvě otázky popíšeme v následujících dvou oddílech.

4.1 Algoritmy pro očíslování vrcholů grafu

V tomto oddíle popíšeme základní algoritmy pro očíslování vrcholů grafu. Zřejmě platí, že přečíslováním vrcholů grafu dojde k permutaci řádků a sloupů v jeho matici sousednosti. Hlavní motivací, proč je pro nás číslování vrcholů grafu důležité, tedy je, že permutací řádků matice můžeme výrazně redukovat počet operací potřebných pro její Choleského rozklad.

Na závěr kapitoly se zmíníme o topologickém číslování vrcholů stromu, které pro nás bude potřebné při popisu eliminačních stromů.

4.1.1 Číslování vrcholů grafu v závislosti na vzdálenosti od separátoru

V této podkapitole popíšeme nejjednodušší metodu číslování vrcholů podgrafu, který vznikl rozdělením původního grafu na n částí. Tuto metodu lze používat samostatně, ale vzhledem k její povaze ji lze využít i pro vylepšení ostatních metod očíslování grafu, například ji lze kombinovat s metodou minimálního stupně.

Mějme graf $G = (V, E)$ a jeho vrcholový separátor G_S, jehož odebráním se graf rozpadne na k podgrafů $G_1, \ldots, G_k$. Popišme číslování vrcholů podgrafu G_i:

1. Položme `j := 1`.
2. Nalezneme neočíslovaný vrchol v grafu G_i takový, že jeho vzdálenost od vrcholového separátoru v grafu G je maximální.
3. Tomuto vrcholu dáme číslo `j`, položíme `j := j + 1`.
4. Pokud jsou všechny vrcholy očíslovány, skončíme, jinak se vrátíme na krok 2

Z algoritmu je vidět, že výsledné očíslování vrcholů grafu nemusí být jednoznačné, protože pokud nalezneme dva nebo více vrcholů, jejichž vzdálenost od separátoru je shodná, můžeme je očíslovat v libovolném pořadí.

4.1.2 Číslování vrcholů pomocí metody minimálního stupně

Metoda minimálního stupně je jednoduchým algoritmem pro nalezení očíslování grafu. Algoritmus pro hledání očíslování grafu pomocí této metody je následující:

1. Mějme graf $G = (V, E)$ a položme $j := 1$.
2. Nalezneme neočíslovaný vrchol v grafu G s nejmenším stupněm a přiřadíme mu číslo j.
3. Přidáme hrany mezi vrcholy z $\mathrm{adj}_G(v)$ tak, aby $\mathrm{adj}_G(v)$ byla klika v grafu G.
4. Pokud nejsou všechny vrcholy očíslované, zvětšíme j o 1 a vrátíme se na krok 2.

Očíslování vrcholů grafu G pomocí tohoto algoritmu není jednoznačné, protože vrcholů s minimálním stupněm může být více.

4.1.3 Smíšené číslování

Vzhledem k povaze číslování popsaných v oddílech 4.1.2 a 4.1.1 můžeme tyto algoritmy zkombinovat. V tomto oddíle popisujeme dvě možnosti přístupu k tomuto problému.

Pokud máme rozdělení $G_1, \ldots, G_k$ grafu G s vrcholovým separátorem G_S, můžeme pro očíslování části G_i použít číslování vrcholů pomocí metody minimálního stupně, kde při výběru vrcholu ve 2. kroku přidáme kritérium vzdálenosti od separátoru popsané v 4.1.1. Nejprve tedy nalezneme množinu všech vrcholů grafu G, které mají minimální stupeň a poté mezi nimi zvolíme ten, který má nejmenší stupeň.

TODO: smíšené s různými koef.

4.1.4 Topologické číslování vrcholů stromu

Topologické číslování vrcholů stromu je intuitivní způsob pro očíslování vrcholů grafu, který je stromem.

Definice 7. Mějme graf $G = (V, E)$, který je stromem. Očíslování jeho vrcholů nazveme topologickým právě tehdy, když pro každý vrchol $v \in V$ platí, že libovolný následník vrcholu v ve stromu G má nižší číslo než vrchol v.

4.2 Eliminační stromy

V této kapitole se budeme zabývat eliminačními stromy a jejich významem pro rozklady řídkých matic. Eliminační stromy při rozkladu matic hrají důležitou roli, protože nám dávají informaci o zaplnění v Choleského faktoru matice bez toho, abychom museli počítat jednotlivé numerické hodnoty.Lze tedy díky nim jednoduše porovnávat vhodnost zvoleného uspořádání řádků a sloupců matice pro Choleského rozklad.

V této kapitole bez újmy na obecnosti předpokládáme, že matice, jejíž Choleského rozklad chceme napočítávat, je ireducibilní, a tedy přidružený graf této matice je souvislý.

4.2.1 Definice eliminačního stromu matice

Nejprve se omezme na ireducibilní, pozitivně definitní, symetrickou matici A_T o rozměrech $n \times n$, jejíž přidružený graf $G(A_T)$ je strom. V tomto případě je A_T tzv. perfektní eliminační matice, tj. existuje permutační matice P taková, že Choleského rozklad matice PA_TP^T nebude obsahovat žádné zaplnění [5] (Matici PA_TP^T můžeme vnímat pouze jako přečíslování řádků a sloupců matice A_T). Aby při choleského rozkladu matice A_T nedošlo k žádnému zaplnění, stačí když pomocí topologického číslování očíslujeme vrcholy jí přidruženého grafu (z předpokladu se jedná o strom) a řádky a sloupce matice A_T seřadíme odpovídajícím způsobem. Pak zjevně platí, že matice A_T má, s výjimkou posledního řádku, pod diagonálou vždy právě jeden nenulový prvek. Díky tomu můžeme definovat pro matici A_T funkci $\texttt{PARENT} : \{1, \ldots, n\} \to 1, \ldots, n$ následovně:

$$\forall j \in \{1, \ldots, n-1\} \quad \texttt{PARENT}[j] := p \quad \Leftrightarrow \quad a_{p,j} \neq 0 \wedge p > j$$
$$\text{a speciálně:} \quad \texttt{PARENT}[n] := 0.$$

Zřejmě ve stromu přidruženém k matici A_T platí, že předchůdcem vrcholu x_j je vrchol $x_{\texttt{PARENT}[j]}$.

Většinou však nepracujeme s maticemi, jejichž přidružený graf by byl stromem. Zavedeme tedy konstrukci pro libovolnou řídkou, ireducibilní, pozitivně definitní, symetrickou matici A o rozměrech $n \times n$. Předpokládejme, že známe Choleského rozklad této matice, tj. $A = LL^T$. Maticí se zaplněním nazveme matici F definovanou jako $F = L + L^T$. Dále zavedeme matice L_t a F_t následovně. L_t je matice vzniklá z L tím, že v každém sloupci vynulujeme všechny prvky pod diagonálou kromě prvku s nejnižším řádkovým indexem a $F_t = L_t L_t^T$.

Z definice F_t vidíme, že se jedná o matici, jejíž přidružený graf $G(F_t)$ je strom.

Definice 8. Eliminačním stromem matice A nazveme graf $G(F_t)$ popsaný výše, značíme $T(A)$. Podstrom $T(A)$ s kořenem x_j značíme $T[x_j]$. Množinu vrcholů tohoto stromu značíme také $T[x_j]$.

Díky této definici můžeme definici funkce `PARENT` přirozeně rozšířit na matici A následovně:

$$\texttt{PARENT}[j] := \min\{i > j | l_{i,j} \neq 0\},$$

kde $l_{i,j}$ označuje i,j-tý prvek matice L.

Pozorování 1. Přímo z definice plyne, že $T(A)$ a $T(F)$ jsou identické.

Pozorování 2. Pokud x_i je vlastním předchůdcem x_j v eliminačním stromu, pak $i > j$.

Tvrzení 1. *Pro $i > j$ závisí numerické hodnoty sloupce $L_{\bullet i}$ na sloupci $L_{\bullet j}$ právě tehdy, když $l_{i,j} \neq 0$.*

Důkaz. Tvrzení plyne přímo ze sloupcového algoritmu [???] □

Kapitola 5

Různé

TODO: rozdistribuovat

5.1 Další témata

TODO: Řídké matice vs. husté matice

Kapitola 6

Algoritmizace a implementace

Cílem programu, který jsme implementovali, je ukázat, že rozdělení grafu získané za pomoci profesionální numerické knihovny pro dělení grafů METIS, nemusí být vyvážené vzhledem k výpočtu Choleského rozkladu na vzniklých oblastech. Dále jsme se pokusili navrhnout a otestovat metodu přerozdělení grafu založenou na přečíslování vrcholů grafu. Cílem navržené metody je zlepšit rozdělení grafu tak, aby počet operací potřebných pro Choleského faktorizaci na jednotlivých oblastem byl pokud možno stejný a nebyl vyšší než maximum z počtu operací při rozdělení počátečním.

TODO: Fortran, Fortran90, c++ TODO: dělení na kolik částí?

Naši implementaci můžeme rozdělit na několik logických celků. TODO: doplnit

6.1 Formát a uložení vstupní matice

V této podkapitole bude popsán vstupní formát matice a způsob její reprezentace v našem programu. TODO: Zajímá nás jen struktura

6.1.1 Vstupní formát matice

Jako základní matice pro testování výsledků našeho programu nám posloužily matice uložené ve formátu RSA v souborech s příponou `.rsa` nebo `.rb`. Jako zdroj pro tyto matice jsme použili kolekci [1].

TODO: popis HB formátu

Pro testování jsme dále používali matice vygenerované při jednoduché pětibodové diskretizaci dvojrozměrné Poissonovy rovnice (podprogram (poisson)) a několik jednoduchých testovacích matic, které jsme si ručně zapsali do souboru `testing.f90`

6.1.2 Reprezentace matice v programu

Matice je v našem programu uložena v **CSR formátu** (compressed sparse row format) [4, 6], který je běžně využívaným formátem sloužícím pro reprezentaci řídkých matic a grafů.

Mějme řídkou matici A o rozměrech $n \times n$, která obsahuje n_e nenulových prvků. Tuto matici v programu reprezentujeme pomocí tří polí: pole `ia` o délce $n+1$ a polí `ja`,`aa` o délce n_e. V poli `ja` na pozicích `ia(`i`)`,...,`ia(`$i+1$`)-1` jsou uloženy sloupcové indexy nenulových prvků na i-tém řádku matice A, na odpovídajících pozicích v poli `aa` jsou uloženy numerické hodnoty těchto prvků.

Příklad 1. Mějme následující symetrickou matici

$$\begin{pmatrix} 0 & 5 & 0 & 3 & 0 \\ 5 & 0 & 9 & 4 & 2 \\ 0 & 9 & 0 & 0 & 0 \\ 3 & 4 & 0 & 0 & 4 \\ 0 & 2 & 0 & 4 & 0 \end{pmatrix}$$

Pak její reprezentace ve formátu CSR vypadá následovně:

```
ia = [ 1, 3, 7, 8, 11, 13 ]
ja = [ 2, 4, 1, 3, 4, 5, 2, 1, 2, 5, 2, 4 ]
aa = [ 5, 3, 5, 9, 4, 2, 9, 3, 4, 4, 2, 4 ]
```

Pokud budeme uvažovat pouze pole `ia`, `ja`, je podle teorie popsané v oddíle 1.3 výsledná reprezentace matice A zároveň reprezentací neorientovaného grafu, kterému matice A odpovídá. Pokud tedy nebudeme brát v úvahu hodnoty jednotlivých prvků matice, můžeme při popisu implementace bez újmy na obecnosti pojmy graf a matice zaměňovat.

Příklad 2. Vezměme matici z příkladu 1. Pak graf, který je reprezentován pomocí polí `ia`, `ja` je zobrazen na obrázku 1.3

6.2 Knihovna METIS pro dělení grafu

Pro samotné dělení grafu na k částí jsme využili knihovnu pro dělení grafů METIS [2] verze 5.1.0. Vzhledem k tomu, že jsme dělení grafů používali jako nástroj pro vyvažování počtu operací při Choleského rozkladu na jednotlivých oblastech, potřebovali jsme nalézt rozdělení grafu pomocí vrcholového separátoru. Takové rozdělení nám poskytne rutina `METIS_ComputeVertexSeparator`, kterou jsme ve Fortranu implementovali pomocí rozhraní `metis_interface.f95` a `metisinclude.c`.

Zmíněná rutina není zmíněna v oficiální dokumentaci knihovny METIS a není tedy ve své základní verzi kompatibilní s jazykem Fortran. Proto bylo nutné učinit několik změn ve formátu CSR tak, aby bylo možné rutinu použít. Nejprve bylo třeba odebrat z grafu smyčky a poté jej přečíslovat tak, aby bylo vrcholy indexovány od 0, jak je požadováno v jazycích C a C++.

Výstupem rutiny `METIS_ComputeVertexSeparator` je pole *part* o délce rovné počtu vrcholů grafu. Na i-té pozici tohoto pole je číslo od 1 do $k+1$ (po převedení do Fortranového zápisu), které určuje, ve které části rozdělení se daný vrchol nachází. $k+1$ značí, že daný vrchol je ve vrcholovém separátoru.

6.3 Tvorba podgrafů a přerozdělení

TODO: doplnit

6.3.1 Číslování

TODO: doplnit

6.3.2 Přerozdělení

TODO: doplnit

6.4 Výpočet Choleského rozkladu

TODO: doplnit

6.5 Softwarové požadavky pro běh

TODO: mám to tam vůbec dávat? Pro úspěšné slinkování a zkompilování TODO: nainstalovaný METIS

Závěr

TODO: doplnit TODO: motivace: nelineární problém viz Tumovy stranky

Literatura

[1] Harwell-boeing collection. `https://math.nist.gov/MatrixMarket/data/Harwell-Boeing/`. Accessed: 5. 10. 2018.

[2] G. Karypis. Metis - a software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices version 5.1.0. 2013.

[3] A. Koubková and V. Koubek. *Datové struktury 1.* Praha: Matfyzpress, 1 edition, 2011.

[4] S. Pissanetzky. *Sparse matrix technology.* Academic Press, Inc. [Harcourt Brace Jovanovich, Publishers], London, 1984.

[5] D. J. Rose. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. pages 183–217, 1972.

[6] Y. Saad. SPARSKIT: A basic tool kit for sparse matrix computations, version 2. Technical report, 1994.