VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra aplikované matematiky

Aplikace magických ohodnocení grafů

Applications of magic-type graph labelings

2010 Lenka Šrubařová

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci na téma „Aplikace magických ohodnocení grafů“ vypracovala samostatně. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

V Ostravě dne: ..... Podpis: .....

Děkuji panu Mgr.Petru Kovářovi, PhD. za výborné vedení mé diplomové práce, rady a připomínky.

Rodičům, nejbližším přátelům, několika aikiďákům a Zlounovi ...

Abstrakt

Cílem práce je prozkoumat využití magického ohodnocení při ukládání řídkých matic. Součástí práce je shrnutí známých způsobů uložení řídké matice a porovnání klasických způsobů uložení řídkých matic a uložení, které využívá hranově magického totálního ohodnocení, z hlediska početní a časové náročnosti.

Abstract

The scope of the thesis is to investigate the use of magic labeling for storing sparse matrix. A part of this thesis is a overview of known forms of storing sparse matrix. This thesis investigates the time intensity of known methods and edge-magic total labeling approach.

Klíčová slova

hranově magické totální ohodnocení, řídká matice, ijv systém uložení, systém uložení pomocí komprimovaného sloupce, systém uložení pomocí magického ohodnocení

Keywords

edge-magic total labeling, sparse matrix, tripplet form, compressed-column form, edge-magic labeling form

Obsah

1 Úvod

Ve své práci se zaměřím na porovnání dvou nejznámějších způsobů ukládání řídkých matic s nově navrženým systémem využívajícím hranově magické ohodnocení.

Práce je rozdělena do čtyř částí. První část se týká pojmů z teorie grafů, které využiji k objasnění hranově magického ohodnocení grafu.

V druhé části své diplomové práce popíši systém ukládání řídkých matic a to systém ijv, systém komprimovaného sloupce a vysvětlím strukturu ukládání řídké matice pomocí hranově magického totálního ohodnocení. Provedu teoretický rozbor časových náročností operací s takto uloženými řídkými maticemi ve zkoumaných způsobech uložení.

Třetí část obsahuje vyhodnocení výsledků časové náročnosti operací mezi systémem ijv, systémem komprimovaného sloupce a uložení pomocí hranově magického totálního ohodnocení.

V poslední části své práce rozeberu některé možnosti urychlení nalezení hranově magického ohodnocení.

2 Základní pojmy

2.1 Graf

Graf G je uspořádaná dvojice (V, E), kde $V(G)$ je neprázdná množina vrcholů a $E(G)$ je množina (ne nutně všech) dvouprvkových podmnožin množiny V. Prvkům množiny $E(G)$ říkáme hrany, prvkům hrany říkáme koncové vrcholy. Počet vrcholů grafu G označíme jako $|V(G)|$, počet hran pak označíme jako $|E(G)|$.

Vrchol množiny $V(G)$ budeme značit v, i-tý vrchol označíme v_i.

Hranu množiny $E(G)$ budeme značit e, i-tou hranu označíme e_i. Ve své práci budu někdy využívat značení (v_i, v_j) (přestože je to neuspořádaná dvojice), které značí hranu mezi vrcholy v_i a v_j.

V tomto textu budu pracovat s konečnými jednoduchými neorientovanými grafy, tj. grafy, jejichž množiny vrcholů a hran jsou konečné, hrany existují pouze mezi dvěma různými vrcholy a hrany nejsou orientované.

2.2 Stupeň vrcholu

Stupeň vrcholu v je počet hran, se kterými je vrchol v incidentní, a značí se $\deg(v)$.

2.3 Cesta

Cesta v grafu G je posloupnost vrcholů a hran $(v_0, e_1, v_1, ..., e_t, v_t)$, kde vrcholy $v_0, ..., v_t$ jsou navzájem různé vrcholy grafu G a pro všechna $i = 1, ..., t$ je $e_i = (v_{i-1}, v_i) \in E(G)$. Cestu označíme P_t, kde t vyjadřuje počet vrcholů cesty P. V mé práci $t - 1$ vyjadřuje délku cesty.

2.4 Kartézský součin dvou grafů

Kartézský součin dvou grafů F a H je graf G, který má vrcholovu množinu $V(F) \times V(H)$, kde $\times$ znamená kartézský součin množin, a dva vrcholy (u_1, v_1) a $(u_2, v_2) \in V(G)$ jsou spojeny hranou právě tehdy, když platí $u_1 = u_2$ a $v_1v_2 \in E(H)$ nebo $v_1 = v_2$ a $u_1u_2 \in E(F)$. Pro všechna $u_1u_2 \in V(F)$, $v_1v_2 \in V(H)$. Kartézský součin grafů značíme $F \square H$.

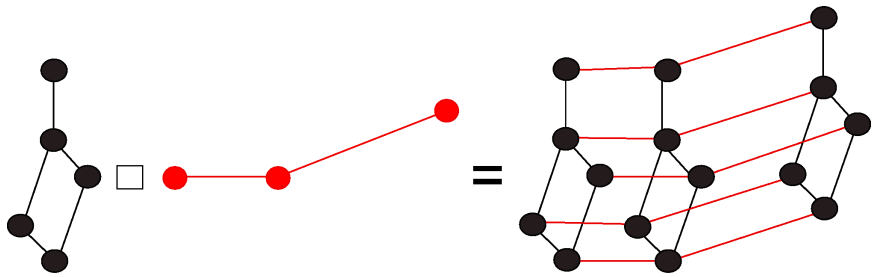
Počet vrcholů grafu $G = F \square H$ je

$$|V(G)| = |V(F)||V(H)|,$$

počet hran lze vyjádřit jako

$$|E(G)| = |E(F)||V(H)| + |E(H)||V(F)|.$$

Příklad kartézského součinu dvou grafů ilustruje následující obrázek.



2.5 Pojmy label, index, váha

Index bude celé číslo přiřazené určitému objektu podle potřeby. Ve většině případech budeme indexy začínat od nuly.

Váhou budeme nazývat hodnotu prvku a_{ij} matice sousednosti A, kde i označuje index řádku a j označuje index sloupce.

Labelem označujeme číslo přiřazované hranám a vrcholům v magickém ohodnocení, viz kapitola 3 *Ohodnocení grafu*.

Obecně váha a label jsou různá čísla, hraně budeme přiřazovat obě (label i váhu).

3 Ohodnocení grafu

Ohodnocení grafu G je zobrazení, které přiřazuje vrcholům nebo hranám grafu G číslo (obvykle kladné nebo nezáporné), které nazveme label. Slovo label počeštíme a budeme skloňovat podle vzoru hrad.

Zavedeme značení $\lambda(v_i)$ pro label vrcholu v_i a značení $\lambda(v_i, v_j)$ pro label hrany mezi vrcholy v_i a v_j, popřípadě značení $\lambda(e_i)$ pro label hrany e_i.

Vrcholové ohodnocení

Vrcholové ohodnocení grafu G je zobrazení, které přiřazuje labely pouze vrcholům grafu G.

Hranové ohodnocení

Hranové ohodnocení grafu G je zobrazení, které přiřazuje labely pouze hranám grafu G.

Totální ohodnocení

Totální ohodnocení je takové zobrazení, že labely jsou přiřazovány hranám i vrcholům grafu G.

V tomto textu pracujeme s totálním ohodnocením a využíváme jako množinu labelů čísla od 1 do $(|V(G)| + |E(G)|)$.

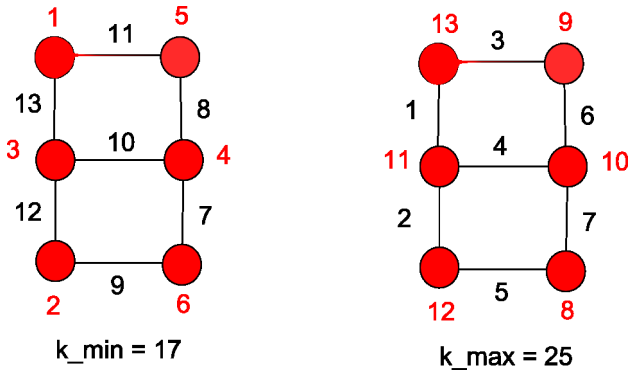
3.1 Hranově magické totální ohodnocení

Ve své práci budu pracovat s hranově magickým totálním ohodnocením, pro které platí, že součet labelů libovolné hrany grafu G a obou jejích koncových vrcholů je roven pevně zvolenému číslu k pro všechny hrany grafu G.

Jinými slovy řečeno hranově magické totální ohodnocení grafu G je prosté zobrazení λ $(V(G) \cup E(G))$ na celá čísla $1, 2, 3, ..., |V(G)| + |E(G)|$ s vlastností, že pro každou hranu (x, y) platí:

$$\lambda(x) + \lambda(x, y) + \lambda(y) = k,$$

kde k je nějaká pevně zvolená přirozená konstanta. Součet $\lambda(x) + \lambda(x, y) + \lambda(y)$ budeme nazývat součtem labelů hrany (x, y) a k magickou konstantou ohodnocení λ.
Příklad totálního magického ohodnocení téhož grafu pro dvě různé konstanty k je na obrázku:



4 Odhady magické konstanty k

Mějme graf $G = P_m \square P_n$. Kartézským součinem dvou cest vznikne mřížka o rozměrech $m \times n$, kde m je počet vrcholů v řádku a n počet vrcholů ve sloupci. Na tyto grafy se zaměříme v dalším textu. Abychom se vyhnuli triviálnímu případu, předpokládejme, že m, $n > 1$. Triviálním případem vznikne mřížka o jednom vrcholu nebo cesta. Vzhledem k tomu, že se zabýváme časovou náročností, je tento triviální případ příliš malý pro zahrnutí do testování jednotlivých operací.

Počet vrcholů grafu G spočítáme jako

$$|V(G)| = mn.$$

Počet hran grafu G spočítáme jako

$$|E(G)| = m(n-1) + n(m-1) = 2mn - m - n.$$

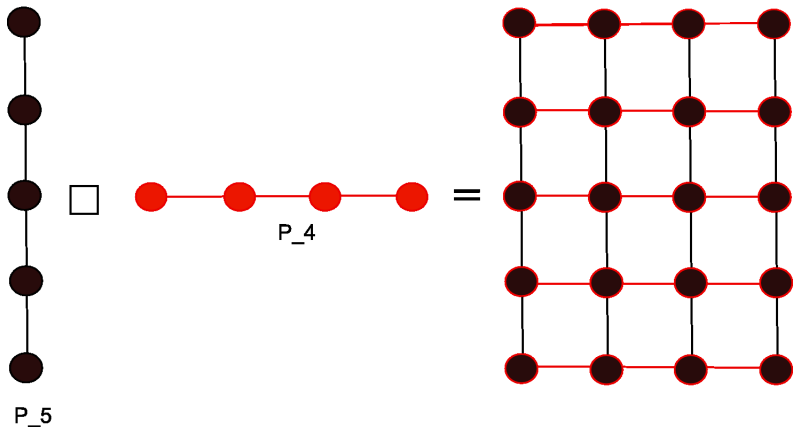
Zavedeme značení L pro množinu čísel od 1 do $(|V(G)| + |E(G)|)$. Pro dolní a horní odhad magické konstanty k musíme znát počet potřebných čísel v ohodnocení, který označíme $|L|$. Tato čísla budou přiřazena jednotlivým vrcholům a hranám daného grafu G, přičemž se žádná hodnota $1, 2, ...|L|$ nemůže objevit dvakrát. Je zřejmé, že platí

$$|L| = |V(G)| + |E(G)| = mn + m(n-1) + n(m-1),$$

po úpravě dostaneme

$$|L| = 3mn - m - n.$$

Znázornění grafu $G = P_4 \square P_5$:



4.1 Stupně vrcholů

Zavedeme označení V_i pro množinu vrcholů stupně i, kde $i = 2, 3, 4$. Vrcholy stupně 1 se v grafech, se kterými budu pracovat, nevyskytují, proto nejmenší $i = 2$. Taktéž se v těchto grafech nevyskytují vrcholy stupně většího než 4, proto největší $i = 4$. Dále zavedeme označení $|V_i|$ pro počet vrcholů stupně i. U grafů $P_m \square P_n$, kterými se zabývám ve své práci, víme, že počet vrcholů stupně 2 bude vždy čtyři, tedy $|V_2| = 4$. Pro počet vrcholů stupně 3 platí: $|V_3| = 2(n-2)+2(m-2)$. Pro počet vrcholů stupně 4 platí: $|V_4| = |V(G)| - 4 - [2(n-2)+2(m-2)]$. $|V_3|$ a $|V_4|$ mohou být nulové pro $m = n = 2$.

4.2 Dolní odhad magické konstanty k

Označíme $L(V_i)$ množinu labelů pro množinu vrcholů stupňů i, platí $|L(V_i)| = |V_i|$, l_{ij} označíme j-tý prvek množiny $L(V_i)$.

Ukážeme, že stejný graf může mít několik různých hranově magických totálních ohodnocení s různými magickými konstantami. Hodnoty magické konstanty nejsou libovolné, ale můžeme je odhadnout.

Pro dolní odhad magické konstanty k je požadováno, aby vrcholy s nejvyššími stupni měly co nejmenší možné labely, protože se do odhadu započítávají tolikrát, jaký je jejich stupeň.

Obecně můžeme vytvořit vzorec pro součet labelů vrcholů jednotlivých stupňů jako

$$i \sum_{j=1}^{|V_i|} l_{ij},$$

neboli

$$i|V_i|\frac{l_{i1}+l_{i|V_i|}}{2}.$$

Z výše uvedených vzorců dostaneme součty stupňů vrcholů stupně i a hran magické konstanty k:

Vrcholy stupně 4

Součet labelů na vrcholech stupně 4 je

$$\frac{4[mn-4-2(m-2)-2(n-2)][1+mn-4-2(m-2)-2(n-2)]}{2} = \cdots = 2[mn-2m-2n+4][mn-2m-2n+5]$$

Vrcholy stupně 3

Součet labelů na vrcholech stupně 3 je

$$\frac{3[[2(n-2)+2(m-2)][(1+mn-4-2(n-2)-2(m-2)+1-1)+(1+mn-4-2(n-2)-2(m-2)+2(m-2)+2(n-2)-1)]]}{2} =$$
$$\cdots = \frac{3[2(n-2)+2(m-2)][2mn-2m-2n+1]}{2}$$

Vrcholy stupně 2

Součet labelů na vrcholech stupně 2 je

$$\frac{2.4[[(1+mn-4-(2(n-2)+2(m-2)))+(2(m-2)+2(n-2)-1+1)]+1+mn-4-(2(n-2)+2(m-2))+2(n-2)+2(m-2)-1+1+3]}{2} =$$
$$\cdots = 4(2mn-3)$$

Hrany

Součet labelů hran je

$$\frac{[m(n-1)+n(m-1)]}{2}.$$
$$\frac{[(m(n-1)+n(m-1)+mn)+(m(n-1)+n(m-1)+mn-m(n-1)-n(m-1)+1)]}{2} =$$

$$\cdots = \frac{[2mn-m-n][4mn-m-n+1]}{2}$$

Dolní odhad magické konstanty k

Z výše uvedených mezisoučtů dostáváme odhad:

$$4(2mn-3)+3[2(n-2)+2(m-2)][2mn-2m-2n+1]+$$
$$+2[mn-2m-2n+4][mn-2m-2n+5]+\frac{[2mn-m-n][4mn-m-n+1]}{2}.$$

Vzhledem k tomu, že počítáme odhad pro jednu hranu grafu G, musíme celý výraz vydělit počtem hran:

$$\frac{4(2mn-3)+3[2(n-2)+2(m-2)][2mn-2m-2n+1]+2[mn-2m-2n+4][mn-2m-2n+5]+\frac{[2mn-m-n][4mn-m-n+1]}{2}}{2mn-m-n},$$

po úpravě dostaneme:

$$\frac{6m^2n^2+m^2n+mn^2-28mn-\frac{7m^2}{2}-\frac{7n^2}{2}+\frac{35m}{2}+\frac{35n}{2}+4}{2mn-m-n}$$

tedy:

$$k \geq \frac{6m^2n^2+m^2n+mn^2-28mn-\frac{7m^2}{2}-\frac{7n^2}{2}+\frac{35m}{2}+\frac{35n}{2}+4}{2mn-m-n}.$$

Dolní odhad magické konstanty označíme k_{min}. Pokud k_{min} není celé číslo, pak za k_{min} budeme považovat $\lceil k_{min} \rceil$.

4.3 Horní odhad magické konstanty k

Pro horní odhad konstanty k chceme, aby vrcholy nejvyššího stupně měly co nejvyšší labely, protože se do odhadu započítávají tolikrát, jaký je stupeň vrcholu.

Obecně můžeme vytvořit vzorec pro součet labelů vrcholů jednotlivých stupňů jako

$$i \sum_{j=1}^{|V_i|} l_{ij},$$

neboli

$$i|V_i|\frac{l_{i1}+l_{i|V_i|}}{2}.$$

Z výše uvedených vzorců dostaneme mezisoučty magické konstanty k pro hrany a jednotlivé množiny vrcholů příslušného stupně:

Vrcholy stupně 4

Součet labelů na vrcholech stupně 4 je

$$\frac{4[mn-4-2(m-2)-2(n-2)][[3mn-m-n]+(3mn-m-n-(mn-4-2(m-2)-2(n-2)-1)]}{2} =$$
$$\cdots = 2[mn-2m-2n+4][5mn-3].$$

Vrcholy stupně 3

Součet labelů na vrcholech stupně 3 je

$$\frac{3[[2(n-2)+2(m-2)]}{2}\cdot$$
$$\frac{[(3mn-m-n-(mn-4-2(m-2)-2(n-2)-1+1))}{2}+$$
$$+\frac{(3mn-m-n-(mn-4-2(m-2)-2(n-2)-1+1)-2(n-2)-2(m-2)+1)]]}{2} =$$
$$\cdots = \frac{3[(2m+2n-8)(4mn+1)]}{2}.$$

Vrcholy stupně 2

Součet labelů na vrcholech stupně 2 je

$$\frac{2\cdot 4[[(3mn-m-n-(mn-4-2(m-2)-2(n-2)-1+1)-2(n-2)-2(m-2)+1-1)]}{2}+$$
$$+\frac{3mn-m-n-(mn-4-2(m-2)-2(n-2)-1+1)-2(n-2)-2(m-2)+1-1-3}{2} = \cdots = 4[4mn-2m-2n+5].$$

Hrany

Součet labelů hran je

$$\frac{[m(n-1)+n(m-1)][1+(m(n-1)+n(m-1)]}{2}.$$

Horní odhad magické konstanty k

Z výše uvedených mezisoučtů dostáváme odhad:

$$2[mn-2m-2n+4][5mn-3]+\frac{3[(2m+2n-8)(4mn+1)]}{2}+4mn-2m-2n+5+\frac{[m(n-1)+n(m-1)][1+(m(n-1)+n(m-1)]}{2},$$

po úpravě dostaneme:

$$12m^2n^2-10m^2n-10mn^2-8mn+\frac{m^2}{2}+\frac{n^2}{2}+\frac{25m}{2}+\frac{25n}{2}-31,$$

což opět musíme vydělíme počtem hran:

$$k \leq \frac{12m^2n^2-10m^2n-10mn^2-8mn+\frac{m^2}{2}+\frac{n^2}{2}+\frac{25m}{2}+\frac{25n}{2}-31}{2mn-m-n},$$

Horní odhad magické konstanty označíme k_{max}. Pokud k_{max} není celé číslo, pak za k_{max} budeme považovat $\lfloor k_{max} \rfloor$.

4.4 Spektrum magické konstanty k

Spektrem magické konstanty k budeme nazývat interval celých čísel. Hranice tohoto intervalu bude dolní a horní odhad magické konstanty. k_{min}, resp. k_{max}. Zajímá nás, jak rychle velikost spektra roste v závislosti na rostoucích hodnotách m a n. Pro odvození spektra použijeme výše uvedené vzorce pro k_{min} a k_{max}. Po jednoduchých úpravách získáme pro k_{min} tento vzorec:

$$\frac{12m^2n^2-10m^2n-10mn^2+16mn+5m^2+5n^2-19m-19n+32}{2\cdot(2mn-m-n)}.$$

Podobně získáme vzorec pro k_{max}:

$$\frac{24m^2n^2-20m^2n-20mn^2+8mn+m^2+n^2+13m+13n-32}{2\cdot(2mn-m-n)}.$$

Spektrum magické konstanty k určíme jako rozdíl k_{max} a k_{min}. Po úpravě získáváme vzorec šířky spektra

$$\frac{6m^2n^2-5m^2n-5mn^2-4mn-2m^2-2n^2+16m+16n-32}{2mn-m-n}.$$

Z výše uvedených vzorců vidíme, že odhady v závislosti na m, n rostou následovně:
pro dolní odhad k_{min} řádově:

$$3mn,$$

pro horní odhad k_{max} řádově:

$$6mn,$$

pro odhad šířky spektra řádově:

$$3mn.$$

Tento odhad je zajímavý z toho důvodu, že s rostoucími hodnotami m a n roste i velikost spektra magické konstanty, čímž čekáme, že roste i počet možných ohodnocení grafu.

Je přirozené očekávat, že všechny grafy $m \times n$ budou mít hranově magické totální ohodnocení (grafy vyhovují nutným podmínkám, jsou skoro pravidelné a navíc známe výsledky ohodnocování grafů podle Josepha A. Galliana).

Příklady grafů s magickým ohodnocením
cyklus délky, která je dělitelná 4
cesta P_n
kompletní bipartitní graf $K_{m,n}$ pro jakákoli m, n
strom s méně než 16 vrcholy
hvězda
Petersenův graf
drak
slunce

5 Ukládání řídkých matic

Řídká matice je matice, která obsahuje řádově více nul než nenulových prvků. Řídká matice se dá uložit úsporněji než v poli o rozměrech $r \times s$, kde r je počet řádků a s je počet sloupců. Tato úspora se projeví, až když jsou počty řádků a sloupců větší než desítky, stovky. Pro názornost využívám v textu jen malé matice.

Příklad 1:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 2 & 0 \\ 3 & 0 & 0 & 6 \\ 0 & 7 & 0 & 9 \\ 5 & 0 & 0 & 4 \end{bmatrix}$$

5.1 ijv systém uložení

ijv systém uložení, tzv. *tripletová forma*, je nejjednodušší způsob uložení řídké matice. Tento systém využívá tři pole – dvě pole celých čísel $i[\,]$, $j[\,]$ a pole reálných čísel $v[\,]$. Hodnoty v poli $i[\,]$ označují řádek, v poli $j[\,]$ pak sloupec nenulového prvku, v poli $v[\,]$ ukládáme samotné hodnoty prvku. Číslování řádků i sloupců začínáme od nuly z důvodu přehlednosti, protože algoritmy programuji v programovacím jazyce $C++$, kde se pole indexují právě od nuly. Matici A z *Příkladu 1* typu $m \times n$ uložíme do polí $i[\,]$, $j[\,]$, $v[\,]$ takto

```
int i[]    = {0, 1, 3, 2, 0, 1, 2, 3};
int j[]    = {0, 0, 0, 1, 2, 3, 3, 3};
double v[] = {1, 3, 5, 7, 2, 6, 9, 4};
```

ijv systém lze jednoduše vytvořit, ovšem jeho použití pro rozsáhlé řídké matice není vhodné, viz kapitola 7.1 *Vyhledávání v systému* ijv. Pole nemusejí být uspořádána.

5.2 Systém uložení komprimovaného sloupce

Tato forma opět používá tři pole. Pole celých čísel $p[\,]$ je délky $n+1$, pole celých čísel $i[\,]$ délky $nzmax$ a pole reálných čísel $v[\,]$ délky $nzmax$, kde $nzmax$ je maximální možný počet nenulových prvků matice. Index řádku daného prvku matice v sloupci j je uložen v $i[p[j]]$ a příslušná hodnota je uložena na stejné pozici v poli $v[\,]$. První prvek $p[0]$ je vždy 0 a $p[n] \leq nzmax$. V $p[j+1]$ je uložen součet počtu nenulových prvků matice do j-tého sloupce včetně(sloupce jsme opět číslovali od 0). Matici A z předchozí strany uložíme metodou *komprimovaného sloupce* takto:

```
int p[]    = {0,          3, 4, 5,          8};
int i[]    = {0, 1, 3, 2, 0, 1, 2, 3};
double v[] = {1, 3, 5, 7, 2, 6, 9, 4};
```

5.3 Systém uložení komprimovaného řádku

Řídká matice se ukládá podobně jako u metody komprimovaného sloupce, ale v tomto případě komprimujeme řádek. Systém opět obsahuje tři pole. Pole celých čísel $p[\,]$ je délky $m+1$, pole celých čísel $i[\,]$ délky $nzmax$ a pole reálných čísel $v[\,]$ délky $nzmax$, kde $nzmax$ je maximální možný počet prvků matice. Index sloupce na daný prvek matice v sloupci j je uložen v $i[p[j]]$ a příslušná hodnota je uložena na stejné pozici v poli v.
Matici A uložíme metodou *komprimovaného řádku* takto:

```
int p[]    = {0,       2,       4,       6,       8};
int i[]    = {0, 2, 0, 3, 1, 3, 0, 3};
double v[] = {1, 2, 3, 6, 7, 9, 5, 4};
```

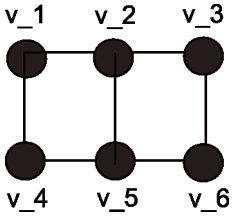
5.4 Ukládání pomocí magického ohodnocení

5.4.1 Specifický tvar řídké matice

Mějme graf $G = P_m \square P_n$. Každá hrana grafu G reprezentuje nenulový prvek matice sousednosti A. Zatím pro jednoduchost budeme předpokládat, že nenulové prvky jsou reprezentovány

hodnotou 1. Matice A bude čtvercová řídká, symetrická s nulami na diagonále – ve své práci budu pracovat s maticemi tohoto typu. Dále víme, že vzniklá matice sousednosti bude mít rozměry $mn \times mn$.
Máme-li graf $G = P_3 \square P_2$ na *Obrázku 1*:

Obrázek 1:



pak příslušná matice A bude vypadat takto

Příklad 2:

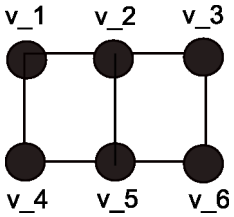
$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}.$$

5.4.2 Struktura uložení

K uložení řídké matice A z *Příkladu 2* pomocí magického ohodnocení musíme uložit magickou konstantu k, rozměry sítě m a n, pole reálných čísel *hodnoty*[ ], pro jednoduchost jsou to v tomto příkladu samé jedničky, velikost pole *hodnoty*[ ] je $(|V(G)| + |E(G)|)$, a dvourozměrné pole sousednosti s[ ][2] velikosti $(|V(G)| + |E(G)|)$ řádků a 2 sloupců, tzn. počet řádků pole s[ ][2] se bude měnit, ale počet sloupců bude vždy 2. Index řádku pole s[ ][2] označuje hodnotu labelu zvětšeného o jedna, jednička se přičítá z důvodu, že pole se v C++ indexuje od nuly, ale první číslo labelů je hodnota 1. V každém řádku pole s[ ][2] je uložena dvojice vrcholů, mezi kterými je hrana s labelem, která odpovídá indexu řádku pole s[ ][2] zvýšenému o jedna. Pokud label náleží vrcholu (není to tedy dvojice různých vrcholů), je v řádku s indexem (label -1) uložen tento vrchol na obou pozicích.

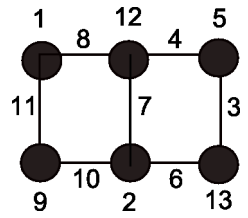
Pro ukládání do této struktury očíslujeme jednotlivé vrcholy grafu $G = P_m \square P_n$. Indexování vrcholů je od 1 do $|V(G)|$, kde v_1 označíme první vrchol vlevo nahoře, pokračujeme sousedy vpravo a dále po řádcích.

V tomto textu budeme pro jednoduchost pracovat s celými čísly.
Pro názornost mějme graf $G = P_3 \square P_2$. Vrcholy grafu G označíme:



Jedno z možných hranově totálních ohodnocení s $k = 21$ je:

Ohodnocení 1:



Pro tento graf G platí: $m = 3$, $n = 2$, $k = 21$, $|V(G)| = 6$, $|E(G)| = 7$. Počet labelů je tedy 13. Pole $s[\,][2]$ vypadá následovně:

Tabulka 1:

$s[\,][2]$

index	label	první vrchol	druhý vrchol
0	1	v_1	v_1
1	2	v_5	v_5
2	3	v_3	v_6
3	4	v_2	v_3
4	5	v_3	v_3
5	6	v_5	v_6
6	7	v_2	v_5
7	8	v_1	v_2
8	9	v_4	v_4
9	10	v_4	v_5
10	11	v_1	v_4
11	12	v_2	v_2
12	13	v_6	v_6

Vzhledem k tomu, že se při algoritmizaci ukládání jedná i o paměťovou náročnost uložení, nebudeme pole sousednosti $s[\,][2]$ ukládat včetně sloupců "index" a "label", tyto hodnoty získáme jednoduše z pozice v poli. Pole sousednosti $s[\,][2]$ bude uloženo v paměti počítače následujícím způsobem.

Tabulka 2:

v_1	v_1
v_5	v_5
v_3	v_6
v_2	v_3
v_3	v_3
v_5	v_6
v_2	v_5
v_1	v_2
v_4	v_4
v_4	v_5
v_1	v_4
v_2	v_2
v_6	v_6

V praxi však nemáme nenulové prvky matice pouze hodnoty jedna. Hodnoty matice sousednosti mohou vyjadřovat určitou fyzikální veličinu. V takovém případě musíme při ukládání matice sousednosti pomocí magického ohodnocení použít pole *hodnoty*[ ] velikosti $(|V(G)| + |E(G)|)$. Každý nenulový prvek matice je v magickém ohodnocení reprezentován hranou. Této hraně je v magickém ohodnocení přidělen určitý label. Abychom neztratili informaci o váze fyzikální veličiny (jaký nenulový prvek byl na této pozici uložen), uložíme váhu nenulového prvku matice sousednosti A do pole *hodnoty*[ ] na pozici *hodnoty*$[label - 1]$. Pole je velikosti $(|V(G)| + |E(G)|)$, protože v magickém ohodnocení jsou labely přiřazeny vrcholům i hranám a na začátku nevíme, které labely budou přiděleny hranám a které vrcholům.

Mějme matici A z *Příkladu 2* s konkrétními hodnotami, s ohodnocením viz *Ohodnocení 1* :

Příklad 3:

$$\mathbf{B} = \begin{bmatrix} 0 & 1 & 0 & 2 & 0 & 0 \\ 1 & 0 & 5 & 0 & 3 & 0 \\ 0 & 5 & 0 & 0 & 0 & 9 \\ 2 & 0 & 0 & 0 & 4 & 0 \\ 0 & 3 & 0 & 4 & 0 & 6 \\ 0 & 0 & 9 & 0 & 6 & 0 \end{bmatrix}.$$

Pro matici B z *Příkladu 3* pole s[ ][2] v tabulce *Tabulka 1* i *Tabulka 2* zůstane stejné, pole *hodnoty*[ ] vypadá následovně

hodnoty[ ]

index	0	1	2	3	4	5	6	7	8	9	10	11	12
label	1	2	3	4	5	6	7	8	9	10	11	12	13
hodnota	0	0	9	5	0	6	3	1	0	4	2	0	0

Opět z důvodu paměťové náročnosti uložení, uložíme pole w[ ] bez "zbytečných" řádků "index" a "label", pole *hodnoty*[ ] bude uloženo v paměti počítače takto:

0	0	9	5	0	6	3	1	0	4	2	0	0

5.5 Bloky a jiné speciální typy uložení

Jestliže je matice A tvořena z hustých čtvercových bloků nenulových čísel nějakého regulárního typu, můžeme modifikovat systém uložení komprimovaného sloupce nebo řádku, který využijeme k uložení blokového modelu. Blokové matice vznikají diskretizací parciálních diferenciálních rovnic, ve kterých jsou různé stupně volnosti asociovány s blokem mřížky. Pak rozdělíme matici na malé bloky velikosti rovnající se číslu stupňů volnosti, s každým blokem zacházíme jako s hustou maticí, přestože může obsahovat nulové prvky.

Dále n_b je dimenze bloku a $nnzb$ je počet nenulových bloků v matici A typu $n \times n$, pak konečné uložení potřebuje $nnz = nnzb \times n_b^2$. Dimenze bloku n_d matice A je pak definována jako $n_d = n/n_b$.

K uložení blokového systému potřebujeme(podobně jako u systému uložení komprimovaného řádku) 3 pole: obdelníkové pole pro čísla s plovoucí čárkou ($val(1 : nnzb, 1 : nb, 1 : nb)$), které ukládá nenulové bloky v (blokovém)řádkovém uspořádání. Pole celých čísel ($col_{ind}(1 : nnzb)$), ve kterém se ukládá aktuální sloupec matice A ukazující na první elementy nenulových bloků. Třetí pole je pole ukazatelů ($rowblk(1 : nd + 1)$), které obsahuje ukazatel na začátek každého bloku řádek v $val(:, :, :)$ a $col_{ind}(:)$.

Tento způsob uložení však není vhodný pro matice sousednosti grafu mřížky $P_m \square P_n$, proto se mu nebudeme dále věnovat. Existují i další způsoby uložení, ale nejsou tak rozšířené. [1]

6 Výhody a nevýhody jednotlivých systémů uložení

6.1 Systém ijv

Systém ijv není vhodný při ukládání velkých matic. Pole $i[\ ]$, pole $j[\ ]$ a pole $v[\ ]$ budou velká. Prvky nejsou v polích uspořádány, což znamená, že při přístupu k určitému hledanému prvku a_{ij} matice A, musíme prohledávat velká pole. Pokud prvky v polích budou uspořádány, vyhledávání bude rychlejší, ale je třeba použít navíc algoritmus k uspořádávání polí. Pole $i[\ ]$ se dá uspořádat, ale to zabere určitý čas – např. řadící algoritmus quick sort má časovou náročnost $O(n\ log\ n)$, avšak v nejhorším případě je tato časová náročnost $O(n^2)$. Navíc to po uspořádání nebude klasický systém ijv, ale jiný málo využívaný systém uložení matice.

Systém ukládání ijv také není optimální ani pro uložení matic, které jsou velmi řídké. Velikost polí budou menší, než při ukládání velkých matic s vysokým počtem nenulových prvků, avšak operace, při kterých musíme pracovat s jednotlivými prvky uložené matice A, budou zpomaleny vyhledáváním daného prvku v tomto systému uložení. Při rozhodování, zda-li je hledaný prvek a_{ij} nulový nebo ne, musíme procházet pole $i[\ ]$. V případě uložení velmi řídké matice A, tzn. matice A obsahuje přílíš mnoho nulových prvků, musíme pole $i[\ ]$ procházet mnohokrát zbytečně.

Pokud je pro nás důležitá časová náročnost při následné práci s takto uloženými maticemi, je systém ijv vhodné použít pro uložení menších řídkých matic.

V případech, kdy se nám nejedná o časovou náročnost vyhledávání jednotlivých prvků a_{ij} uložené matice A je tento systém uložení velmi jednoduše naprogramovatelný pro jakékoli řídké matice.

Výhodou je, že se pomocí systému ijv dají uložit jakékoli matice – obdélníkové i čtvercové, matice nemusejí mít žádný specifický tvar jako u uložení pomocí magického ohodnocení.

[1] *Davis, T. A.:* Direct methods for sparse linear systems

6.2 Systém komprimovaného sloupce

Mějme matici typu $m \times n$. Číslo m určuje počet řádků matice a číslo n počet jejich sloupců. Pro matici A bude platit $m < n$. Pro matici B pak bude platit $m > n$. Snadno tak rozlišíme, zda matice má více řádků (B) nebo sloupců (A).

Dále předpokládejme, že počet nenulových prvků matice A i B je stejný a označme tento počet nz.

Systém komprimovaného sloupce využívá pro uložení řídké matice tři pole. Víme, že obě matice A i B mají stejný počet nenulových prvků, což znamená, že pole $v[\,]$ (pole pro uložení vah nenulových prvků) bude pro uložení obou matic A i B velikosti nz. Každý nenulový prvek těchto matic, má v poli $i[\,]$ uložen index svého řádku. To znamená, že pole $i[\,]$ je také velikosti nz pro obě matice A i B. Tato dvě pole $v[\,]$ a $i[\,]$ pro obě matice zaberou v paměti počítače stejné místo. Nezáleží tedy na tom, jestli má matice více řádků nebo více sloupců. U pole $p[\,]$ však nastane rozdíl. Matice A má více sloupců, proto bude pole $p[\,]$ (počet nenulových prvků matice j-tého sloupce včetně) pro tuto matici větší velikosti než pro matici B. Pokud máme matice A a B, pak menší paměťové nároky na své uložení bude mít matice B, u které platí $m > n$.

Pokud je důležitá časová náročnost operací s uloženými maticemi, systém uložení řídké matice pomocí komprimovaného sloupce se díky své struktuře hodí v případech, že potřebujeme častěji přistupovat k prvkům uložené matice a_{ij} po sloupcích.

Výhodou je, stejně jako u systému ukládání ijv, že se pomocí systému komprimovaného sloupce dají uložit jakékoli matice – obdélníkové i čtvercové, matice nemusejí mít žádný specifický tvar jako u uložení pomocí magického ohodnocení.

6.3 Magické ohodnocení

Nevýhodou popsaného ukládání pomocí magického ohodnocení je, že musíme mít matici určitého typu (viz kapitola 5.4.1 *Specifický tvar řídké matice*). Nemůžeme tedy ukládání pomocí magického ohodnocení použít na libovolnou řídkou matici. Tento fakt je velmi nepříjemný, protože matice ve specifickém tvaru příliš často z praxe nemáme.

Další obrovskou nevýhodou je, že s uloženými maticemi chceme pracovat. Pokude ale matici například vynásobíme vektorem nebo jinou maticí, změníme její strukturu a tudíž nemůžeme použít nalezené magické ohodnocení. Museli bychom zjišťovat magické ohodnocení jiného grafu.

Pokud však najdeme magické ohodnocení sítě, která nebude obdélníková (popřípadě čtvercová) a která bude vyjadřovat nějakou řídkou matici, pak je možné také tuto matici uložit pomocí magického ohodnocení.

Výhodou ukládání matice sousednosti pomocí magického ohodnocení je ta, že vyhledání jakéhokoli prvku je časově konstantní a tato konstanta je malá. Tzn. že nebude záležet na druhu operace, které chceme s takto uloženými maticemi provádět, prvky budou nacházeny přímým přístupem do pole sousednosti $s[\,][2]$.

7 Vyhledávání prvku a_{xy} matice v jednotlivých uloženích

S uloženými řídkými maticemi potřebujeme pracovat. Abychom mohli provádět různé operace (například násobení matice vektorem, násobení matice maticí apod.), potřebujeme přistupovat k jednotlivým nenulovým prvkům těchto matic. V této kapitole popíši, jak vyhledávání fun-

guje v jednotlivých systémech uložení. Prvek a_{xy} je prvkem matice A s indexem řádku x a s indexem sloupce y. Indexy x a y začínáme od nuly.

7.1 Vyhledávání v systému ijv

Máme k dispozici

- pole $i[\,]$ ... indexy řádků nenulových prvků matice A,
- pole $j[\,]$ ... indexy sloupců nenulových prvků matice A
- pole $v[\,]$ ... hodnoty nenulových prvků.

Pro přístup k určitému prvku matice a_{xy} se musí v ijv systému uložení procházet celé pole i, protože prvky nejsou nijak seřazeny. V momentě nalezení požadovaného indexu řádku x v poli $i[\,]$ musíme porovnat hodnotu indexu sloupce y s hodnotou v poli $j[\,]$ na pozici, na které je v poli $i[\,]$ uložen řádek. Pokud prvek a_{xy} byl v původní matici nulový, musíme projít celé pole $i[\,]$. Proto je tento ijv systém nevhodný pro ukládání větších řídkých matic. Vyhledávání jednotlivých prvků je poměrně časově náročné. V případě, že je pole neuspořádané, je časová náročnost $O(nzmax)$, v případě, že pole uspořádané je, je časová náročnost $O(log_2\ nzmax)$, kde $nzmax$ je počet nenulových prvků matice A.

7.2 Vyhledávání v systému komprimovaného sloupce

Máme k dispozici

- pole $p[\,]$ ... počet nenulových prvků matice do j-tého sloupce včetně,
- pole $i[\,]$ ... indexy řádků nenulových prvků matice A,
- pole $v[\,]$ ... hodnoty nenulových prvků.

Pro nalezení prvku a_{xy} v tomto systému uložení přistoupíme nejdříve do pole $p[y+1]$, kde je uložen počet nenulových prvků matice ve sloupcích předchozích včetně aktuálního sloupce y. Abychom získali počet nenulových prvků ve sloupci y, podíváme se na $p[y]$. Počet nenulových prvků v daném sloupci y je tedy $p[y+1] - p[y]$. Dalším krokem pro nalezení prvku a_{xy} je prohledání pole $i[\,]$ – pole, kde jsou uloženy indexy řádku nenulových prvků. Index x řádku hledaného prvku a_{xy} může být na pozici od $i[p[y]]$ až $i[p[y+1]-1]$. Pokud v poli $i[\,]$ na těchto pozicích nalezneme hodnotu x, prvek byl nenulový a jeho váhu nalezneme v poli $v[\,]$ na stejné pozici, kde byl nalezen řádek x v poli $i[\,]$.

7.3 Vyhledávání v magickém ohodnocení

Máme k dispozici

- konstantu k ... magická konstanta pro konkretní ohodnocení,
- rozměr m ... počet řádků mřížky,
- rozměr n ... počet sloupců mřížky,
- pole sousednosti $s[\,][2]$ ... dvojice vrcholů, mezi kterými je hrana a její label nebo label vrcholu.

Pro přístup k nenulovému prvku a_{xy} v uložení pomocí magického ohodnocení potřebujeme ještě pole vrcholů $w[\,]$, které na w[číslo vrcholu - 1] = $\lambda vrcholu$. Pole $w[\,]$ je velikosti mn. Toto pole $w[\,]$ dostaneme jednoduše z dvourozměrného pole $s[\,][2]$(*Tabulka 1*), kde label jednotlivých vrcholů získáme průchodem pole $s[\,][2]$ a zjištěním, ve kterých řádcích jsou dva stejně indexované vrcholy. Mohlo by se zdát, že toto pole je zbytečné tvořit, když jsme schopni najít label vrcholu v poli $s[\,][2]$, ale při algoritmizaci je to nezbytné z důvodu časové náročnosti. Pokud máme pole $w[\,]$ vytvořené, nemusíme už kvůli labelu vrcholu procházet pole sousednosti $s[\,][2]$.

Pole vrcholů $w[\,]$ pro *Příklad 2* s magickou konstantou $k = 21$, $m = 3$, $n = 2$:

$w[\,]$

označení vrcholu	v_1	v_2	v_3	v_4	v_5	v_6
label vrcholu	1	12	5	9	2	13

Vzhledem k paměťové náročnosti uložení, vypustíme celý řádek "označení vrcholu" a název "label vrcholu", pole $w[\,]$ bude uloženo v paměti počítače takto

1	12	5	9	2	13

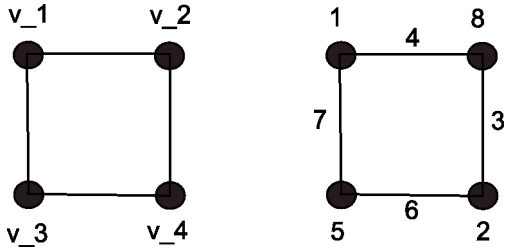
Máme rozhodnout, zda-li je prvek a_{xy} v původní matici nulový nebo nenulový. Pokud je a_{xy} nenulový, určíme jeho hodnotu. Pro toto rozhodování pracujeme s indexy řádku a sloupce prvku a_{xy}, které jsou v magickém uložení reprezentovány jako jednotlivé hrany (samozřejmě platí pouze pro nenulové prvky).

Zda-li je prvek a_{xy} nenulový, zjistíme přímým přístupem do pole sousednosti $s[\,][2]$ takto: x odpovídá vrcholu v_{x+1}, y odpovídá vrcholu v_{y+1} . Zajímá nás, jestli je mezi těmito dvěma vrcholy hrana. Jestliže hrana mezi těmito dvěma vrcholy neexistuje, pak prvek a_{xy} byl nulový, jestliže však hrana existuje, prvek a_{xy} byl nenulový a jeho hodnotu nalezneme v poli *hodnoty*$[\,]$ podle labelu hrany.

Nalezneme příslušné labely $\lambda(v_{x+1})$ pro v_{x+1} a $\lambda(v_{y+1})$ pro v_{y+1} v poli vrcholů $w[\,]$. Dále vypočteme label hrany, který byl měl být mezi v_{x+1} a v_{y+1}, tedy $\lambda(v_{x+1}, v_{y+1}) = k - \lambda(v_{x+1}) - \lambda(v_{y+1})$. Pokud v poli na $s[\lambda(v_{x+1}, v_{y+1}) - 1][0]$ je uložen v_{x+1} a na $s[\lambda(v_{x+1}, v_{y+1}) - 1][1]$ je uložen v_{y+1}(nebo naopak, na pořadí nezáleží z důvodu symetrie matice), pak můžeme říct, že prvek a_{xy} byl v původní matici A nenulový a jeho hodnotu najdeme v poli *hodnoty*$[\lambda(v_{x+1}, v_{y+1}) - 1]$. Pokud na těchto pozicích nejsou vrcholy v_{x+1} a v_{y+1}, pak vrcholy sousední nejsou (neexistuje hrana mezi vrcholy v_{x+1} a v_{y+1}).

7.3.1 Konkrétní příklad pro vyhledávání v magickém ohodnocení

Mějme graf $G = P_2 \square P_2$. V grafu G očíslujeme vrcholy. Použijeme jedno z možných hranově magických ohodnocení s magickou konstantou $k = 13$



Z grafu G vytvoříme matici sousednosti řádu 4

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Pro ukázku nahradíme hodnoty 1 jinými nenulovými hodnotami. Jak už bylo zmíněno v kapitole 5.4.2 *Struktura uložení*, hodnoty matice sousednosti mohou reprezentovat fyzikální veličinu, proto zvolíme i jiné hodnoty než hodnotu 1, přičemž musíme zachovat symetrii matice. Dostaneme matici B.

$$\mathbf{B} = \begin{bmatrix} 0 & 3 & 5 & 0 \\ 3 & 0 & 0 & 6 \\ 5 & 0 & 0 & 1 \\ 0 & 6 & 1 & 0 \end{bmatrix}$$

Pole sousednosti $s[\,][2]$ pro matici B (samozřejmě platí i pro matici A) je následující

$s[\,][2]$

v_1	v_1
v_4	v_4
v_2	v_4
v_1	v_2
v_3	v_3
v_3	v_4
v_1	v_3
v_2	v_2

Pole hodnot *hodnoty*[ ] pro matici B je následující

hodnoty[ ]

0	0	6	3	0	1	5	0

Z pole sousednosti $s[\,][2]$ vytvoříme pole vrcholů $w[\,]$

$w[\,]$

1	8	5	2

Při práci s poli je důležité nezapomenout, že indexy prvků se číslují od nuly.

Problém

Vybereme si libovolný prvek b_{xy} matice sousednosti B. Prvek b_{xy} označuje hranu mezi vrcholy v_{x+1} a v_{y+1}. Chceme zjistit, zda je tento prvek v matici sousednosti B nenulový, a pokud ano, chceme znát jeho hodnotu.

a) určíme prvek b_{13} – hledaný prvek je číslo 6
b) určíme prvek b_{30} – hledaný prvek je číslo 0
c) určíme prvek b_{12} – hledaný prvek je číslo 0

Řešení a)

Prvek b_{13} označuje hranu mezi vrcholy $v_{1+1} = v_2$ a $v_{3+1}v_4$

1. krok:

- najdeme labely vrcholů v_2 a v_4 v poli $w[\,]$ - přistoupíme tedy do pole w na pozice $w[1]$ a $w[3]$, získáme $\lambda(v_2) = 8$, $\lambda(v_4) = 2$
- spočítáme label potenciální hrany $\lambda(hrany) = k - \lambda(v_2) - \lambda(v_4)$, tedy $\lambda(v_2, v_4) = 13 - 8 - 2 = 3$

2. krok:

- zkontrolujeme existenci pozice $\lambda(v_2, v_4) - 1$ v poli sousednosti $s[\,][2]$ – víme, že pole $s[\,][2]$ je velikosti 8 (tzn. nejvyšší možný index je pak 7), kontrolujeme pozici $s[2]$ – tato pozice v poli existuje, protože $2 \in \langle 0, 7\rangle$, můžeme přistoupit ke 3. kroku

3. krok:

- zkontrolujeme existenci hrany – přistoupíme do pole sousednosti $s[\,][2]$ na pozici $s[\lambda(v_2, v_4)-1]$, tedy na pozici řádku $s[2][i]$ $(i = 1, 2)$, $s[2][0]$ obsahuje v_2, $s[2][1]$ obsahuje v_4, můžeme říct, že hrana mezi vrcholy existuje, tzn. prvek b_{13} je nenulový

4. krok:

- zjistíme hodnotu nenulového prvku - přistoupíme do pole $hodnoty[\,]$ na pozici $hodnoty[\lambda(v_2, v_4)-1]$, tedy na $hodnoty[2]$, nenulový prvek má hodnotu 6.

Řešení b)

Prvek b_{30} označuje hranu mezi vrcholy $v_{3+1} = v_4$ a $v_{0+1} = v_1$

1. krok:

- najdeme labely vrcholů v_4 a v_1 v poli $w[\,]$ - přistoupíme tedy do pole w na pozice $w[3]$ a $w[0]$, získáme $\lambda(v_4) = 2$, $\lambda(v_1) = 1$

- spočítáme label potenciální hrany $\lambda(hrany) = k - \lambda(v_4) - \lambda(v_1)$, tedy $\lambda(v_4, v_1) = 13 - 2 - 1 = 10$

2. krok:

- zkontrolujeme existenci pozice $\lambda(v_4, v_1) - 1$ v poli sousednosti $s[\,][2]$ – víme, že pole $s[\,][2]$ je velikosti 8 (tzn. nejvyšší možný index je pak 7), kontrolujeme pozici $s[9]$ - tato pozice v poli neexistuje, protože $9 \notin \langle 0, 7\rangle$, tzn. neexistuje hrana mezi vrcholy v_4 a v_1, prvek b_{30} matice sousednosti B prohlásíme za nulový

Řešení c)

Prvek b_{12} označuje hranu mezi vrcholy v_2 a v_3

1. krok:

- najdeme labely vrcholů v_2 a v_3 v poli $w[\,]$ - přistoupíme tedy do pole w na pozice $w[1]$ a $w[2]$, získáme $\lambda(v_2) = 8$, $\lambda(v_3) = 5$
- spočítáme label potencionální hrany $\lambda(hrany) = k - \lambda(v_2) - \lambda(v_3)$, tedy $\lambda(v_2, v_3) = 13 - 8 - 5 = 0$

2. krok:

- zkontrolujeme existenci pozice $\lambda(v_2, v_3) - 1$ v poli sousednosti s – pole $s[\,]$ je velikosti 8 (tzn. nejvyšší možný index je pak 7), kontrolujeme pozici $s[-1]$ - tato pozice v poli neexistuje, protože $-1 \notin \langle 0, 7\rangle$, tzn. neexistuje hrana mezi vrcholy v_2 a v_3, proto prvek b_{30} matice sousednosti B prohlásíme za nulový

Shrnutí

Při algoritmizaci ukládání matice sousednosti B pomocí magického ohodnocení a následného vyhledávání prvku b_{xy} v tomto uložení je nutné si uvědomit, že mohou nastat případy *i*) a *ii*).

Případ *i*) demonstruje snahu přistoupit do pole na záporných pozicích, což samozřejmě technicky není možné, protože pole jsou indexována od nuly.

Případ *ii*) ukazuje situaci přístupu k prvku pole na pozici, která v poli neexistuje, resp. na pozici s indexem, který je větší než velikost pole zmenšená o jedna.

Popíšeme poslední možnou variantu, která může nastat. Zvolíme prvek matice, který označí dva vrcholy sítě. Vrcholy označme v_p a v_d. Opět vyhledáme $\lambda(v_p)$ a $\lambda(v_d)$ v poli $w[\,]$. Label $\lambda(v_p, v_d)$ hrany mezi v_p a v_d spočítáme už známým vzorcem $\lambda(v_p, v_d) = k - \lambda(v_p) - \lambda(v_d)$. Zjistíme, že pozice $\lambda(v_p, v_d)$ zmenšený o jedna existuje v poli $v[\,]$, avšak v řádku na této pozici jsou v sloupcích jiné vrcholy, než právě vrchol v_p a v_d. To znamená, že hrana mezi v_p a v_d neexistuje.

8 Operace s maticemi

8.1 Násobení matice vektorem zprava

Mějme matici A řádu $mn \times mn$ a sloupcový vektor délky mn. Výsledkem násobení matice A vektorem v zprava bude sloupcový vektor y délky mn.

$$\begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,mn} & a_{0,mn-1} \\ a_{1,0} & & \cdots & & a_{1,mn-1} \\ \vdots & & \ddots & & \vdots \\ a_{mn,0} & & \cdots & & a_{mn-2,mn-2} \\ a_{mn-1,0} & & \cdots & & a_{mn-1,mn-1} \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{mn-2} \\ v_{mn-1} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{mn-2} \\ y_{mn-1} \end{bmatrix}$$

Pro $y = Av$ platí

$$y_i = \sum_{j=0}^{mn-1} a_{i,j} v_j, \text{ kde } i = 0, 1, \cdots, mn-1.$$

8.1.1 Systém ijv

Jak už bylo řečeno v kapitole 7.1 *Vyhledávání v systému ijv*, indexy řádků prvků uložené matice sousednosti A v poli $i[\,]$ v uložení ijv nejsou uspořádány. Pokud chceme přistoupit k určitému prvku a_{ij} matice sousednosti A uložené pomocí systému ijv, musíme v nejhorším případě (v případě že byl prvek nulový) prohledat celé pole indexů řádků $i[\,]$.

Pro násobení matice A vektorem v zprava postupně procházíme pole indexů řádků $i[\,]$. Nenulovou hodnotu na dané pozici násobíme příslušnou hodnotou ve vektoru a přičteme do výsledného vektoru. Konkrétně $vysledek[i[k]] += vektor[j[k]] \cdot v[k]$.

8.1.2 Systém komprimovaného sloupce

Pro přístup k prvku a_{ij} matice A uložené pomocí systému komprimovaného sloupce nemusíme prohledávat celá pole, viz 7.2 *Vyhledávání v systému komprimovaného sloupce*. Pro nalezení daného prvku a_{ij} matice A musíme prohledat určitý počet pozic pole indexů řádků $i[\,]$, tento počet je závislý na počtu nenulových prvků v daném sloupci j prvku a_{ij}. Víme ovšem, že nejhorší možný počet pozic, které je nutno prohledat může být řád n matice A (počet jejich řádků). Ve skutečnosti pozic k prohledání bude méně, protože uložená matice A byla řídká.

Rychlejší variantou je postupně procházet všechny nenulové prvky a připočítávat je k prvkům výsledného vektoru.

```
for(int i=0;i<sloupec.size();i++){
  for(int j=sloupec[i];j<sloupec[i+1];j++){
    vysledek[radek[j]] += vektor[i]*hodnoty[j];
  }
}
```

Víme, že násobení matice A vektorem v zprava bude časově náročnější než násobení matice A vektorem v zleva, protože při násobení matice A vektorem v zprava potřebujeme nenulové prvky matice z jednotlivých řádků, ke kterým je v systému komprimovaného sloupce přístup komplikovanější než v přístupu ke sloupci.

8.1.3 Magické ohodnocení

V kapitole 7.3 *Vyhledávání v magickém ohodnocení* byl popsán postup pro vyhledání určitého prvku a_{ij} matice sousednosti A. Z popsaného postupu je zřejmé, že nalezení daného prvku a_{ij} matice sousednosti A zabere vždy stejný čas. Proto nezáleží, zda-li násobíme matici sousednosti A vektorem zleva nebo zprava. Tzn. nezáleží, zda potřebujeme vyhledat celý sloupec matice nebo její řádek.

8.2 Násobení matice vektorem zleva

Mějme matici A typu $mn \times mn$ a řádkový vektor řádu mn. Výsledkem násobení matice A vektorem v zleva bude řádkový vektor y řádu mn.

$$\begin{bmatrix} v_0 & v_1 & \cdots & v_{mn-2} & v_{mn-1} \end{bmatrix} \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,mn} & a_{0,mn-1} \\ a_{1,0} & & \cdots & & a_{1,mn-1} \\ \vdots & & \ddots & & \vdots \\ a_{mn,0} & & \cdots & & a_{mn-2,mn-2} \\ a_{mn-1,0} & & \cdots & & a_{mn-1,mn-1} \end{bmatrix} =$$

$$= \begin{bmatrix} y_0 & y_1 & \cdots & y_{mn-2} & y_{mn-1} \end{bmatrix}$$

Pro $y = vA$ platí:

$$y_i = \sum_{j=0}^{mn-1} a_{ji} v_j, \text{ kde } i = 0, 1, \cdots, mn-1$$

8.2.1 Systém ijv

Při násobení matice sousednosti A řádkovým vektorem v zleva násobíme celý sloupec matice vektorem. Opět projdeme nenulové hodnoty a přičteme na příslušné pozice ve výsledném vektoru. Časová náročnost vA je stejná jako Av.

8.2.2 Systém komprimovaného sloupce

Pro násobení matice sousednosti A vektorem v zleva v systému uložení pomocí komprimovaného sloupce potřebujeme sloupec uložené matice A. Obecně tato struktura uložení velmi rychle umožňuje přístup k prvkům matice po jednotlivých sloupcích.

Zrychlení však opět docílíme procházením nenulových prvků a jejich násobením prvkem vektoru v. Systém komprimovaného sloupce umožňuje rychlejší přístup ke sloupci než k řádku matice A, tudíž čekáme, že násobení matice A vektorem v zleva bude rychlejší než násobení matice A vektorem v zprava.

8.2.3 Magické ohodnocení

Časová náročnost násobení vA v uložení pomocí magického ohodnocení bude mít stejnou časovou náročnost jako násobení Av. V kapitole 7.3 *Vyhledávání v magickém ohodnocení* byl

popsán postup pro vyhledání určitého prvku a_{ij} matice sousednosti A. Z popsaného postupu je zřejmé, že nalezení daného prvku a_{ij} matice sousednosti A zabere vždy stejný čas. Proto nezáleží, zda-li násobíme matici sousednosti A vektorem zleva nebo zprava. Tzn. nezáleží, zda potřebujeme vyhledat celý sloupec matice nebo její řádek.

9 Shrnutí předpokládaných výsledků

9.1 Vyhledávání prvků

V tuto chvíli byla popsána náročnost vyhledávání prvků uložené matice sousednosti A v jednotlivých uloženích. Předpokládáme, že systém uložení ijv si vzhledem k časové náročnosti povede ze všech tří zkoumaných uložení nejhůř. Tento závěr je zřejmý z nutnosti prohledávání celého pole $i[\,]$. Vyhledání prvku a_{ij} v systému uložení pomocí komprimovaného sloupce bude rychlejší, protože nemusíme nikdy při hledání prohledat žádné jeho celé pole. Vyhledávání v prvku a_{ij} v uložení pomocí magického ohodnocení bude nejrychlejší – rozhodnutí o jeho nulovosti se provede přímým přístupem do pole sousednosti $s[\,][2]$, nemusíme žádné pole procházet.

9.2 Násobení vektorem

Pro násobení matice vektorem přistupujeme postupně ke všem nenulovým prvkům matice sousednosti, násobíme je s příslušným prvkem vektoru a tento výsledek pak přičítáme k výslednému prvku vektoru. Z tohoto důvodu očekáváme, že časová náročnost násobení matice vektorem zprava a zleva budou stejné v rámci jednoho uložení – přesněji v uložení pomocí systému ijv a systému pomocí magického ohodnocení, v uložení pomocí komprimovaného sloupce čekáme, že násobení matice A vektorem v zprava bude časově náročnější než násobení matice A vektorem v zleva.

Zajímavým výsledkem pak bude porovnání časové náročnosti násobení matice vektorem v systému uložení ijv, systému komprimovaného sloupce a v systému pomocí magického ohodnocení. Z návrhu algoritmu – pracování pouze s nenulovými prvky uložené matice, se dá předpokládat, že výrazné odlišnosti v časové náročnosti nebudou. Časová náročnost se bude lišit velmi málo taky z důvodu, že testované matice jsou velmi malinké.

10 Implementované algoritmy

Pro vytvoření všech algoritmů jsem využila programovací jazyk $C++$. Implementované algoritmy jsou na přiloženém CD.

10.1 Nalezení magického ohodnocení

10.1.1 Zdrojový soubor `Emt_global.cpp`

Tento program najde všechna hranově magická totální ohodnocení pro síť námi zadaných rozměrů m a n, kde m je počet řádků a n počet sloupců mřížky, pro všechny možné magické konstanty k ve zvoleném rozmezí k_{min} až k_{max}.

Program `Emt_global.cpp` vytvoří dva textové soubory – soubor s informacemi o jednotlivých ohodnoceních `vystup_info.txt`. V tomto souboru jsou uloženy rozměry sítě m a n, počet vrcholů a hran sítě, počet potřebných labelů pro ohodnocení, dále pak hodnoty k_{min}

a k_{max}. Pro jednotlivé hodnoty k je zde uveden počet nalezených ohodnocení. Posledním údajem je čas nalezení všech možných ohodnocení dané sítě o rozměrech m, n.

Druhý vytvořený textový soubor s názvem `vystup_ohodnoceni.txt` obsahuje jednotlivá konkretní ohodnocení mřížky daného rozměru m, n. U každého ohodnocení je uvedena hodnota magické konstanty k, pro kterou bylo ohodnocení nalezeno.

Následuje ukázka souboru `vystup_info.txt` pro mřížku 3×2.

```
Graf ma rozmery 3 x 2

Pocet vrcholu: 6
Pocet hran: 7
Pocet labelu: 13

Soubor obsahuje ohodnoceni z tohto intervalu:
k_min: 17
k_max: 25

--------------------------------------------------------------------------------

k=17 : 24 ohodnoceni
k=18 : 28 ohodnoceni
k=19 : 20 ohodnoceni
k=20 : 108 ohodnoceni
k=21 : 56 ohodnoceni
k=22 : 108 ohodnoceni
k=23 : 20 ohodnoceni
k=24 : 28 ohodnoceni
k=25 : 24 ohodnoceni

--------------------------------------------------------------------------------

vypocet trval 0 vterin
```

Ukázku celého textového souboru `vystup_ohodnoceni.txt` neuvádím, protože pro síť rozměrů 3×2 bylo nalezeno 416 různých ohodnocení. Následuje ukázka části výstupu textového souboru (první dvě nalezená ohodnocení pro konstantu $k = 17$ a poslední nalezené ohodnocení pro $k = 25$).

```
k = 17

ohodnoceny graf:
1    11    5    8    4
13        10         7
3    12    2    9    6

---------------------------------
```

```
k = 17

ohodnoceny graf:
1    13    3    12    2
11        10          9
5     8    4     7    6

---------------------------------
             .
             .
             .
---------------------------------

k = 25

ohodnoceny graf:
13    3    9    6    10
1          4          7
11    2   12    5     8

---------------------------------
```

Soubor `Emt_global.cpp` jsme naprogramovali z důvodu kontroly navržení algoritmu pro hledání magického ohodnocení grafu. Správnost algoritmu můžeme zkontrolovat tak, že počty nalezených ohodnocení pro jednotlivé konstanty k sítě rozměrů $m \times n$ musejí být dělitelné 4. Pro síť rozměrů $m \times n$, kde $m = n$, tedy jedná se o čtvercovou mřížku, pak tyto počty musejí být dělitelné 8. Obdélník má 4 symetrie – otočení o 0 stupňů a 180 stupňů, a dvě osové souměrnosti - podle svislé a vodorovné osy – tedy dělitelnost 4. Čtverec má symetrií 8, proto počty ohodnocení musejí být dělitelné 8.

10.1.2 Zdrojový soubor `Emt_global_jeden.cpp`

Program `Emt_global_jeden.cpp` je modifikací předešlého programu `Emt_global.cpp`. Program `EMT_global_jeden.cpp` nalezne jedno ohodnocení pro síť zadané velikosti a zadané konstanty. Vstupem programu jsou rozměry mřížky m, n a hodnota magické konstanty k, pro kterou se bude hledání provádět.

Výstupem jsou opět dva textové soubory – `vystup_info.txt` a `vystup_ohodnoceni.txt`, ve kterém v tomto případě bude uvedeno jen jedno ohodnocení mřížky rozměru $m \times n$.

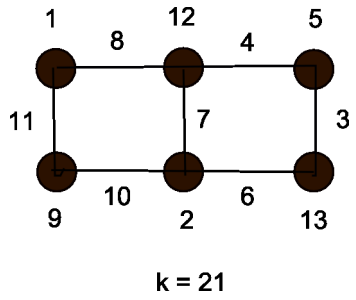
Následuje ukázka souboru `vystup_ohodnoceni.txt` pro mřížku 3×2 a námi zvolenou konstantu $k = 21$.

```
k=21

ohodnoceny graf:
1      8      12       4      5
11            7               3
9      10     2        6      13

---------------------------
```

Ukázku ohodnoceného grafu z textového souboru `vystup_ohodnoceni.txt` znázorníme graficky



Největší mřížka, která tímto programem byla ohodnocena, měla rozměry $m = 6$, $n = 5$. Při snaze ohodnotit mřížku větší už bylo nutno projít tolik možností, že jsme se výsledku nedočkali.

10.1.3 Zdrojový soubor `export_EMT.cpp`

Tento program je stejný program jako `Emt_global_jeden.cpp`, ale provádí výstup do textového souboru `export_ohodnoceni.txt`, s tímto textovým souborem pak pracují všechny programy, které provádějí operace s maticemi uloženými pomocí magického ohodnocení.

První řádek souboru `export_ohodnoceni.txt` obsahuje hodnotu magické konstanty k, na druhém řádku je uložen počet řádků mřížky m, na třetím řádku je pak uložen počet sloupců mřížky n. Od čtvrtého řádku následuje pole sousednosti $s[\ ][2]$. Každý řádek pak obsahuje dvě hodnoty a to indexy vrcholů.

Následuje ukázka textového souboru `export_ohodnoceni.txt` pro mřížku $m \times n$, $k = 21$.

```
21
3
2
1 1
5 5
3 6
2 3
3 3
```

```
5 6
2 5
1 2
4 4
4 5
1 4
2 2
6 6
```

10.2 Generování matice a vektoru

10.2.1 Zdrojový soubor `generuj_matici.cpp`

Jak už bylo napsáno v kapitole 5.4.1 *Specifický tvar řídké matice*, nemůžeme pomocí magického ohodnocení ukládat libovolnou řídkou matici. Matice musí být čtvercová a symetrická, viz 5.4.1 *Specifický tvar řídké matice*. Proto jsem vytvořila program `generuj_matici.cpp`, která vytvoří matici, kterou lze pomocí magického ohodnocení uložit.

Po spuštění programu `generuj_matici.cpp` je uživatel vyzván, aby zadal rozměry mřížky – rozměry m a n. Vzniklá matice bude typu $mn \times mn$. Dalším vstupem programu je zadání minimální a maximální váhy, která bude generována náhodně a bude uložena na pozici nenulového prvku matice sousednosti A. Náhodně vygenerované váhy prvků matice budou pro názornost celá čísla mezi těmito dvěma hodnotami. Např. zadá-li uživatel minimální váhu -3 a maximální 4, váhy prvků matice budou z množiny $\{-3, -2, -1, 1, 2, 3, 4\}$. Pokud je minimální váha zadána záporná a maximální kladná, mohlo by se stát, že se vygeneruje váha 0, to ale nechceme (narušení specifického tvaru matice sousednosti), proto je tato možnost v programu ošetřena tím, že v případě vygenerování nuly se zvýší tato váha na jedničku.

Program vygeneruje řídkou matici specifického tvaru typu $mn \times mn$ s váhami prvků v rozmezí zadaném uživatelem. Tato matice je uložena do textového souboru s názvem `matice.txt`. Jednotlivé prvky matice jsou v textovém souboru odděleny mezerou, řádek je ukončen enterem, avšak poslední řádek se enterem neukončuje.

Vytvořený soubor `matice.txt` umí načíst všechny tři programy pro uložení řídké matice:

`ulozeni_matice_tripplet.cpp`,
`ulozeni_matice_compressed.cpp`,
`ulozeni_matice_magicke_ohodnoceni.cpp`.

10.2.2 Zdrojový soubor `generuj_vektor.cpp`

Pro měření časové náročnosti jsem naprogramovala násobení uložené matice sousednosti vektorem zprava a zleva. Program `generuj_vektor.cpp` vytvoří vektor v zadané délky. Vstupem `generuj_vektor.cpp` je délka vektoru. Víme, že uložená matice sousednosti je čtvercová typu $mn \times mn$, proto délka vektoru bude mn. Opět uživatel musí zadat dolní a horní hranici čísel, mezi kterými se budou náhodně generovat váhy jednotlivých prvků vektoru v.

Program vygeneruje vektor požadované délky a uloží jej do textového souboru `vektor.txt`. Textový soubor pak načítají programy pro násobení matice sousednosti vektorem zprava a zleva uvedené v kapitole 8.1 *Násobení vektorem zprava a zleva*.

10.2.3 Zdrojový soubor `generuj_tvar.cpp`

Jedná se o stejný program jako `generuj_matici.cpp` s tím rozdílem, že místo konkrétních vah prvků máme na těchto pozicích matice jedničky.

Tento program vytvoří specifický tvar matice typu $mn \times mn$, kde váhy prvků budou jedničky. Vygenerovanou matici uloží také do textového souboru `typ_matice.txt`.

10.3 Ukládání řídkých matic

10.3.1 Zdrojový soubor `ulozeni_matice_tripplet.cpp`

Program `ulozeni_matice_tripplet.cpp` načte řídkou matici z textového souboru s názvem `matice.txt` a uloží ji systémem ijv do textového souboru s názvem `matice_tripplet.txt`.

Textový soubor `matice_tripplet.txt` obsahuje na prvních dvou řádcích rozměry ukládané matice m a n (i když víme, že naše ukládané matice budou čtvercové, ale program funguje pro libovolné matice). Na třetím řádku s polem indexů řádků nenulových prvků – pole $i[\ ]$, čtvrtý řádek obsahuje indexy sloupců nenulových prvků – pole $j[\ ]$, pátý řádek pak obsahuje pole vah $v[\ ]$.

Oddělovačem jednotlivých řádků v textovém souboru je enter, oddělovač jednotlivých hodnot je mezera.

10.3.2 Zdrojový soubor `ulozeni_matice_compressed.cpp`

Program `ulozeni_matice_compressed.cpp` načte řídkou matici z textového souboru s názvem `matice.txt` a uloží ji do textového souboru `matice_compressed.txt`.

Textový soubor `matice_compressed.txt` stejně jako `matice_tripplet.txt` obsahuje na prvních dvou řádcích rozměry ukládané matice m a n. Pole $p[\ ]$ – pole počtu nenulových prvků j-tého sloupce včetně nalezneme na třetím řádku souboru `matice_compressed.txt`. Čtvrtý řádek je pole $i[\ ]$ – indexy řádků nenulových prvků uložené matice sousednosti A. Poslední řádek obsahuje pole $v[\ ]$, ve kterém jsou uloženy váhy nenulových prvků uložené matice sousednosti A.

Oddělovačem jednotlivých řádků v textovém souboru je nový řádek, oddělovač jednotlivých hodnot je mezera.

10.3.3 Zdrojový soubor `ulozeni_matice_magicke_ohodnoceni.cpp`

Program `ulozeni_matice_magicke_ohodnoceni.cpp` načte řídkou matici z textového souboru `matice.txt`. Pro uložení do textového souboru `matice_magicke_ohodnoceni.txt`, potřebuje ještě tento program načíst `export_ohodnoceni.txt`.

Textový soubor `matice_magicke_ohodnoceni.txt` obsahuje pouze jeden řádek, na kterém je uloženo pole *hodnoty*$[\ ]$.

10.4 Násobení vektorem zprava a zleva

10.4.1 Zdrojové soubory `nasobeni_vektorem_zprava_tripplet.cpp`, `nasobeni_vektorem_zleva_tripplet.cpp`

Programy pracují s uloženou maticí z textového souboru `matice_tripplet.txt` a vektorem načteným z textového souboru `vektor.txt`.

Výsledný vektor násobení matice a vektrou zprava (resp. zleva) je vypsán na konzoli.

10.4.2 Zdrojové soubory `nasobeni_vektorem_zprava_compressed.cpp`

Programy načtou uloženou matici z textového souboru `matice_compressed.txt` a vektor z textového souboru `vektor.txt`.

Výsledný vektor násobení matice a vektoru zprava (resp. zleva) je vypsán na konzoli.

10.4.3 Zdrojové soubory `nasobeni_vektorem_zprava_magicke_ohodnoceni.cpp`, `nasobeni_vektorem_zleva_magicke_ohodnoceni.cpp`

Uložená matice je načítána z textového souboru `matice_magicke_ohodnoceni.txt`. Dále oba programy potřebují export ohodnocení, ve kterém je uložena hodnota magické konstanty k, rozměry m a n a pole sousednosti $s[\ ][2]$. Z tohoto důvodu se načte i textový soubor `export_ohodnoceni.txt`.

Výsledný vektor násobení uložené matice a vektoru zprava (zleva) je vypsán na konzoli.

10.5 Přístup k prvkům matice

Ve této práci chceme také porovnat rychlost přístupu (rychlost vyhledání) ke všem prvkům uložené matice sousednosti A v systému uložení ijv, pomocí komprimovaného sloupce a pomocí magického ohodnocení.

Programy postupně přistupují ke všem prvkům uložené matice a zaznamenávají celkový čas.

10.6 Měření časové náročnosti algoritmů

Abychom mohli jednotlivé systémy uložení porovnat vzhledem k jejich časové náročnosti, byly vytvořeny algoritmy, které čas měří.

Programy ignorují režii načítání uložených matic z textových souborů. Čas se měří pouze při vykonávání dané operace. Čas násobení matice vektorem je tedy čas potřebný k vyhledání potřebných prvků matice a jejich vynásobení s prvky vektoru.

Všechny programy měřící čas vrací čas v sekundách. V těle algoritmu je v tuto chvíli nastaven for cyklus, který proběhne desettisíckrát, abychom mohli zaznamenat časy. Časy jsou velmi malé, protože testuji matice malých řádů (pro velkou matici se mi nepodařilo najít hranově magické totální ohodnocení), proto cyklus proběhne desettisíckrát, aby se výsledný čas dal zaznamenat.

Implementovali jsme následující algoritmy měřící čas:

`cas_pristup_tripplet.cpp`
`cas_pristup_compressed.cpp`
`cas_pristup_magicke_ohodnoceni.cpp`
`cas_nasobeni_vektorem_zprava_tripplet.cpp`
`cas_nasobeni_vektorem_zprava_compressed.cpp`
`cas_nasobeni_vektorem_zprava_magicke_ohodnoceni.cpp`
`cas_nasobeni_vektorem_zleva_tripplet.cpp`
`cas_nasobeni_vektorem_zleva_compressed.cpp`
`cas_nasobeni_vektorem_zleva_magicke_ohodnoceni.cpp`

Důležité je zdůraznit, že všechny operace probíhají v paměti, tudíž časy nejsou zvyšovány (zkreslovány) přístupem na disk.

10.7 Postup při použití implementovaných algoritmů

Chceme testovat matici sousednosti A typu $m \times n$, např. 20×20, tedy mřížku o rozměrech $m \times n$, kde $m = 5$, $n = 4$. Dále chceme porovnat časovou náročnost přístupu k jednotlivým prvkům a_{ij} matice sousednosti A v jednotlivých uloženích a časovou náročnost násobení této matice sousednosti A vektorem v zprava a zleva.

1. krok – generování matice, matice bude uložena v textovém souboru *matice.txt*

- spustíme program `generuj_matici.cpp`
- zadáme rozměry mřížky, v našem konkrétním případě bude $m = 5$, $n = 4$ (vznikne matice typu 20×20)
- zadáme rozsah vah prvků matice – např. dolní váhu zvolíme -5, horní 5

Pro práci s maticí v uložení pomocí magického ohodnocení, potřebujeme znát magickou konstantu k, rozměry mřížky m, n a pole sousednosti $s[\,][2]$, tyto údaje jsou uloženy v textovém souboru `export_ohodnoceni.txt`

2. krok – vytvoření exportu ohodnocení, výstup bude uložen v souboru `export_ohodnoceni.txt`

- spustíme program `export_EMT.cpp`
- zadáme rozměry mřížky, v našem konkrétním případě bude $m = 5$, $n = 4$

3. krok – uložení vygenerované matice

- spustíme program `ulozeni_matice_tripplet.cpp` – vznikne textový soubor s názvem `matice_tripplet.txt`
- spustíme program `ulozeni_matice_compressed.cpp` – vznikne textový soubor s názvem `matice_compressed.txt`
- spustíme program `ulozeni_matice_magicke_ohodnoceni.cpp` – vznikne textový soubor s názvem `matice_magicke_ohodnoceni.txt`

4. krok – změříme čas přístupu k jednotlivým prvkům matice A ve všech třech typech uložení

- spustíme program – `cas_pristup_tripplet.cpp` – vrátí čas v sekundách přístupu v systému uložení ijv
- spustíme program – `cas_pristup_compressed.cpp` – vrátí čas v sekundách přístupu v systému uložení komprimovaného sloupce
- spustíme program – `cas_pristup_magicke_ohodnoceni.cpp` – vrátí čas v sekundách přístupu v systému uložení pomocí magického ohodnocení

Po 4.kroku již známe dobu přístupu ke všem prvkům matice sousednosti A ve všech třech typech uložení. Výsledky toho měření jsou zaznamenány v tabulce v kapitole 11 *Výsledky*.

5. krok – generování vektoru, vektor bude uložen v textovém souboru `vektor.txt`

- spustíme program `generuj_vektor.cpp`
- zadáme délku vektoru – délka je mn, pro matici typu 20×20 je délka vektoru 20
- zadáme rozsah vah prvků vektoru – např. dolní váhu zvolíme -3, horní 5

Po 5. kroku máme k dispozici matici A uloženou třemi různými způsoby a vygenerovaný vektor potřebné délky.

6. krok – násobení uložené matice A vektorem v zprava

- spustíme program `cas_nasobeni_vektorem_zprava_tripplet.cpp` – vrátí čas v sekundách potřebný pro násobení v systému uložení ijv
- spustíme program `cas_nasobeni_vektorem_zprava_compressed.cpp` – vrátí čas v sekundách potřebný pro násobení v systému uložení pomocí komprimovaného sloupce
- spustíme program `cas_nasobeni_vektorem_zprava_magicke_ohodnoceni.cpp` – vrátí čas v sekundách potřebný pro násobení v systému uložení pomocí magického ohodnocení

11 Výsledky

V této práci chceme porovnat časovou náročnost při práci s uloženými maticemi sousednosti v jednotlivých systémech uložení.

Výsledky jsem zaznamenala do tabulek podle velikosti ukládané matice sousednosti A. V tabulkách pro jednotlivé matice nalezneme potřebný čas pro nalezení magického ohodnocení příslušné sítě, čas přístupu k prvkům matice a časy násobení matic sousednosti vektorem zprava a zleva.

Pro přehlednost je nejlepší dosažený čas v operacích přístup k prvkům matice, násobení matice vektorem zprava a násobení matice vektorem zleva v tabulce označen červeně.

11.1 Testovaná data

11.1.1 Matice typu 30×30, mřížka 6×5

Matice A typu 30×30, vektor v velikosti 30		
typ uložení	**čas Av v sekundách**	**čas vA v sekundách**
ijv	$6,565.10^{-5}$	$6,539.10^{-5}$
komprimovaný sloupec	$2,638.10^{-5}$	$1,560.10^{-5}$
magické ohodnocení	$2,250.10^{-5}$	$2,810.10^{-5}$

Čas přístupu ke všem prvkům matice A typu 30×30	
typ uložení	**čas přístupu v sekundách**
ijv	$6,636.10^{-3}$
komprimovaný sloupec	$2,640.10^{-4}$
magické ohodnocení	$2,500.10^{-5}$

Čas nalezení magického ohodnocení pro 6×5 v sekundách
1439

11.1.2 Matice typu 20×20, mřížka 5×4

Matice A typu 20×20, vektor v řádu 20		
typ uložení	**čas Av v sekundách**	**čas vA v sekundách**
ijv	$2,157.10^{-5}$	$2,157.10^{-5}$
komprimovaný sloupec	$1,397.10^{-5}$	$1,200.10^{-5}$
magické ohodnocení	$1,260.10^{-5}$	$1,410.10^{-5}$

Čas přístupu ke všem prvkům matice A typu 20×20	
typ uložení	**čas přístupu v sekundách**
ijv	$2,114^{-3}$
komprimovaný sloupec	$1,313.10^{-4}$
magické ohodnocení	$1,400.10^{-5}$

Čas nalezení magického ohodnocení pro 5×4 v sekundách
135

11.1.3 Matice typu 16×16, mřížka 4×4

Matice A typu 16×16, vektor v velikosti 16		
typ uložení	**čas Av v sekundách**	**čas vA v sekundách**
ijv	$8,125.10^{-6}$	$8,922.10^{-6}$
komprimovaný sloupec	$6,880.10^{-6}$	$6,300.10^{-6}$
magické ohodnocení	$6,800.10^{-6}$	$6,800.10^{-6}$

Čas přístupu ke všem prvkům matice A typu 16×16	
typ uložení	**čas přístupu v sekundách**
ijv	$8,985.10^{-4}$
komprimovaný sloupec	$7,500.10^{-5}$
magické ohodnocení	$7,800.10^{-6}$

Čas nalezení magického ohodnocení pro 4×4 v sekundách
10023

11.1.4 Matice typu 12×12

Matice A typu 12×12, vektor v velikosti 12		
typ uložení	**čas Av v sekundách**	**čas vA v sekundách**
ijv	$3,516.10^{-6}$	$3,325.10^{-6}$
komprimovaný sloupec	$3,700.10^{-6}$	$3,350.10^{-6}$
magické ohodnocení	$3,400.10^{-6}$	$3,400.10^{-6}$

Čas přístupu ke všem prvkům matice A typu 12×12	
typ uložení	**čas přístupu v sekundách**
ijv	$3,375.10^{-4}$
komprimovaný sloupec	$3,910.10^{-5}$
magické ohodnocení	$4,700.10^{-6}$

Čas nalezení magického ohodnocení pro 4×3 v sekundách
2

11.2 Shrnutí výsledků

Přístup ke všem a_{ij}

Časová náročnost přístupu k prvku a_{ij} matice sousednosti je v systému uložení ijv nejvyšší. Jak už bylo napsáno v kapitole 7 *Vyhledávání prvku* a_{ij} pro nalezení prvku a_{ij} v tomto systému uložení musíme procházet pole $i[\,]$, ve kterém nejsou obecně prvky uspořádané.

Časová náročnost vyhledání prvku matice sousednosti je v systému uložení pomocí komprimovaného sloupce menší než u systému ijv. Nejrychlejší přístup ke všem prvkům matice má systém uložení pomocí magického ohodnocení. Tyto výsledky nejsou překvapivé, časová náročnost je dána navrženou strukturou jednotlivých uložení. Za zmínění stojí fakt, že časové náročnosti jednotlivých systémů se v přístupu ke všem prvkům matice sousednosti A lišily o řád.

Násobení vektorem

Algoritmy jsou navrženy tímto způsobem: postupně se projdou všechny uložené nenulové prvky matice sousednosti A a jejich váha pak přispívá k jednotlivým prvkům výsledného vektoru.

Časové náročnosti násobení matice vektorem (zprava či zleva) vycházely řádově stejně u všech třech typů zkoumaných systémů uložení.

12 Nalezení magického ohodnocení

Počet potřebných labelů pro graf $G = P_m \square P_n$ je $|V(G)| + |E(G)|$. Labely využívané v této práci jsou čísla od 1 do $|V(G)| + |V(E)|$. Hledáme bijektivní zobrazení λ z $(V(G) \cup E(G))$ na celá čísla $1, 2, 3, ..., |V(G)| + |E(G)|$. Požadavek bijektivnosti zpomaluje vyhledání takového ohodnocení, které je magické. Při ukládání matice sousednosti nám nezáleží z jaké množiny jsou labely. Důležité je nalezení nějakého hranově magického ohodnocení pro danou mřížku. Proto je takto zvolená množina labelů příliš omezená.

12.1 Návrh na zrychlení nalezení magického ohodnocení

Zrychlení nalezení hranově magického ohodnocení se dá vyřešit rozšířením množiny labelů. Počet labelů zvýšíme o konstantu h, kde $h < |V(G)|$. Tímto řešením roste pravděpodobnost nalezení hranově magického ohodnocení. Na druhou stranu roste paměťová náročnost, protože musíme zvětšit pole sousednosti $s[\,][2]$. Rychlost přístupu k prvkům ukládané matice by však zůstal skoro stejný – zvětšením pole sousednosti $s[\,][\,]$ se rozšíří počet pozic, na které se může v průběhu hledání prvku a_{ij} přistupovat. Tzn. budou prováděny zbytečné operace porovnání prvků, které jsou v poli sousednosti uloženy, protože ne celé pole bude využito pro záznam labelů hran a vrcholů.

Dalším možným způsobem k urychlení hledání magického ohodnocení je nastavení časového limitu, který bude věnován nalezení vhodného ohodnocení pro jednotlivé prvky zadané mřížky. Tímto způsobem se program zbytečně nebude zdržovat a zkusí najít rychleji jiné ohodnocení. Problém ovšem bude odhadnout časový limit, během kterého se má jednotlivý prvek ohodnotit.

Hranově magické totální ohodnocení umíme v tuto chvíli hledat pouze počítačově tzv. hrubou silou. Zrychlení by nastalo, kdyby se povedlo najít obecný předpis pro nalezení magického ohodnocení grafu. V tomto případě by pak bylo možné ukládat i jiné matice, než se kterými jsme v celém textu pracovali.

12.1.1 Implementované algoritmy k urychlení nalezení magického ohodnocení

Rozšíření množiny labelů – zdrojový soubor `Emt_fast_vice_labelu.cpp`

Urychlení nalezení hranově magického ohodnocení v programu `Emt_fast_vice_labelu.cpp` spočívá v rozšíření množiny labelů. Jedná se o modifikaci programu `Emt_global_jeden.cpp` s tím rozdílem, že uživatel nevolí magickou konstantu k, ale může zadat konstantu $h \in \mathbb{N}$, o kterou se zvětší množina labelů používaná při ohodnocování grafu. Aby rozšíření množiny labelů mělo význam pro urychlení nalezení magického ohodnocení, hodnota $h < |V(G)|$, pokud tomu tak není, uživatel je na tuto skutečnost upozorněn výpisem na obrazovku. Před samotným zadáním h je uživateli oznámena maximální možná hodnota h. Program ovšem funguje i po zadání větší než vhodné konstanty h, protože tento program byl vytvořen pro testování. Program ukončí hledání při nalezení prvního vyhovujícího ohodnocení.

Program `Emt_fast_vice_labelu.cpp` vytvoří stejně jako `Emt_global_jeden.cpp` dva textové soubory – `vystup_info.txt`, kde jsou uloženy rozměry sítě m a n, počet vrcholů a hran, počet potřebných labelů pro ohodnocení, počet nových labelů a hodnoty magické konstanty k_{min} a k_{max}, a `vystup_ohodnoceni.txt`, který obsahuje konkrétní ohodnocení mřížky daného rozměru m a n. Program na konzoli vypíše čas v sekundách, který byl potřebný k nalezení ohodnocení.

Následují ukázky souborů `vystup_info.txt` a `vystup_ohodnoceni.txt` programu `Emt_fast_vice_labelu.cpp`. pro mřížku 3×3.

Soubor `vystup_info.txt`

```
Graf ma rozmery 3 x 3

Pocet vrcholu: 9
```

```
Pocet hran: 12
Pocet labelu: 21
Novy pocet labelu: 26

Soubor obsahuje ohodnoceni z tohto intervalu:
k_min: 25
k_max: 41

--------------------------------------------------------------------------------

k=25 : 1 ohodnoceni

--------------------------------------------------------------------------------

vypocet trval 0 vterin
```

Soubor `vystup_ohodnoceni.txt`

```
k = 25

ohodnoceny graf:
1    19    5    11    9
22        17         10
2    20    3    16    6
15        18         12
8    13    4    14    7

---------------------------------
```

Jak můžeme vidět ve výstupu souboru `vystup_info.txt`, počet labelů byl původně 21, při rozšíření množiny labelů o 5 se počet možných labelů zvýšil na 26. V hranově magickém totálním ohodnocení by nejvyšší přiřazený label byl 21, ale vidíme, že program nalezl jako první ohodnocení, kde použil label 22 (viz první sloupec, druhý řádek ohodnoceného grafu).

Srovnání časové náročnosti `Emt_global_jeden.cpp` **a** `Emt_fast_vice_labelu.cpp`

Mřížka 4×3			
Program	**počet labelů**	h **- počet labelů navíc**	**čas v sekundách**
`Emt_global_jeden.cpp`	29	0	12
`Emt_fast_vice_labelu.cpp`	39	10	15
`Emt_global_jeden.cpp`	29	0	12
`Emt_fast_vice_labelu.cpp`	34	5	15

Mřížka 4×4			
Program	**počet labelů**	h **- počet labelů navíc**	**čas v sekundách**
`Emt_global_jeden.cpp`	40	0	10023
`Emt_fast_vice_labelu.cpp`	40	15	$--$
`Emt_global_jeden.cpp`	40	0	10023
`Emt_fast_vice_labelu.cpp`	40	5	$--$

Pro hranově magické totální ohodnocení mřížky 4×4 potřebujeme 40 labelů. Pomocí programu Emt_global_jeden.cpp se podařilo tuto mřížku ohodnotit za 10 023 sekund. V programu Emt_fast_vice_labelu.cpp se hranově magické ohodnocení nepodařilo nalézt ani po 16 hodinách, množina labelů byla zvýšená o 15 labelů, tedy počet možných labelů bylo 55. Ani v případě, kdy byl počet labelů zvýšen jen o 5, nedočkali jsme se nalezení ohodnocení grafu.

Časový limit pro ohodnocení jednotlivých vrcholů a hran – zdrojový soubor `Emt_fast_random.cpp`

Urychlení nalezení hranově magického ohodnocení grafu v programu `Emt_fast_random.cpp` spočívá v zadání časového limitu pro ohodnocení jednotlivého vrcholu nebo hrany. Uživatel zadá čas v sekundách, který je potřebný k ohodnocení jednotlivé hrany nebo vrcholu. Pokud se v časovém limitu nepodaří vrchol nebo hranu ohodnotit, program se vrátí o krok zpět a pokusí se znova ohodnotit vrchol nebo hranu, ze které se nemohl "dostat" k dalšímu prvku. Program je ukončen při nalezení prvního vyhovujícího ohodnocení, nebo když projde všechna možná ohodnocení jednotlivých vrcholů a hran, ale nesplní časový limit. Program na konzoli vypíše čas v sekundách, který byl potřebný k nalezení ohodnocení.

Výsledky srovnání časové náročnosti nalezení hranově magického ohodnocení pomocí programu `Emt_global_jeden.cpp` a `Emt_fast_random.cpp` nedopadly podle očekávání. Testovala jsem mřížku o rozměrech 4×3. Program `Emt_global_jeden.cpp` dokázal nalézt hranově magické totální ohodnocení za 12 sekund. Program `Emt_fast_random.cpp` nalezl hranově magické totální ohodnocení za 16 sekund, zajímavé ale je, že tato doba, 16 sekund, nebyla závislá na časovém limitu, program byl spuštěn s časovým limitem od 10^{-1} po 10^{-20} sekundy.

Shrnutí

Algoritmy navržené pro urychlení nalezení magického ohodnocení nepřinesly očekáváné výsledky. Z tohoto důvodu soudím, že pro urychlení hledání magického ohodnocení grafu bude nutné nalézt obecný předpis.

13 Závěr

Cílem diplomové práce bylo srovnat časovou náročnost nejpoužívanějších způsobů uložení řídkých matic s nově navrženým systémem využívající hranově magického totálního ohodnocení. Konkrétně jsem tento nový způsob ukládání řídkých matic srovnávala se systémem uložení ijv a systémem uložení pomocí komprimovaného sloupce.

Hledání samotného hranově magického ohodnocení grafu je velmi časově náročné. Doposud nebyl nalezen obecný předpis pro hledání magického ohodnocení, proto se hledání provádí počítačově a to hrubou silou. Z důvodu "neznalosti" obecného předpisu pro hledání hranově magického totálního ohodnocení jsem testovala pouze řídké matice ve specifickém tvaru, u kterých jsem byla schopná v reálném čase najít jedno hranově magické totální ohodnocení. Přesto se hranově magické totální ohodnocení podařilo nalézt jen pro malé grafy (největší ohodnocená mřížka byla rozměru 6×5, tedy matice 30×30). Pokud by se uvažovalo dále pokračovat ve výzkumu přínosu ukládání řídkých matic pomocí hranově magického ohodnocení, pak by nalezení předpisu bylo nezbytné.

Snaha o zrychlení nalezení magického ohodnocení grafu nepřinesla uspokojivé výsledky. Ani jeden z navrhovaných způsobů (rozšíření množiny labelů, stanovení časového limitu pro ohodnocení jednotlivých prvků mřížky) podle provedeného testování neurychlí nalezení magického ohodnocení grafu.

Výsledky testování časové náročnosti jednoznačně ukázaly, že nový způsob ukládání řídkých matic pomocí hranově magického totálního ohodnocení je v přístupu ke všem prvkům uložené matice sousednosti A z porovnávaných systémů nejrychlejší.

Testy časové náročnosti násobení matice vektorem neprokázaly jednoznačně, který ze srovnávaných způsobů je nejrychlejší.

Ukládání matic pomocí magického ohodnocení grafu má smysl v případě, kdy pro nás není důležitý čas pro nalezení magického ohodnocení. Jak už ale bylo zmíněno výše, nepovedlo se ohodnotit mřížku větších rozměrů než 6×5, což je velmi malá matice, která má navíc specifický tvar. V případě dalšího využití ukládání matic pomocí magického ohodnocení bude opravdu nezbytné nalézt obecný předpis pro hledání magického ohodnocení grafu.

Reference

[1] *Davis, T. A.:* **Direct methods for sparse linear systems**
SIAM, Philadelphia 2006

[2] *Wallis, W. D.; Baskoro, E. T.; Miller, M.;, Slamin:* **Edge-magic total labelings**
Australasian Journal of Combinatorics 22 (2000), pp. 177–190

[3] *Barrett, R.; Berry, M.; Chan, T. F.; Demmel,J.; Donato,J. M.; Dongarra, J.; Eijkhout, V.; Pozo, R.; Romine, Ch. and Henk Van der Vorst:* **Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods**
dostupné z: `http://www.siam.org/books`

Seznam příloh

CD příloha