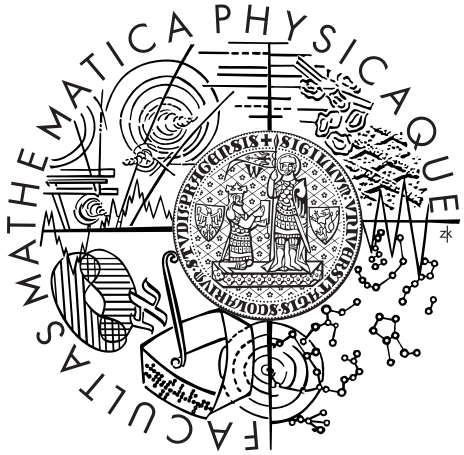
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Jana Rapavá

Algoritmy založené na omezené expanzi - implementace a vyhodnocení

Informatický ústav Univerzity Karlovy

Vedoucí bakalářské práce: doc. Mgr. Zdeněk Dvořák, Ph.D.
Studijní program: Informatika
Studijní obor: obecná informatika

Praha 2014

Ďakujem vedúcemu práce doc. Mgr. Zdeňkovi Dvořákovi, Ph.D. za odborné rady, ochotu a trpezlivosť pri konzultáciách, a Mgr. Tomášovi Gavenčiakovi za cenné diskusie. Poďakovanie tiež patrí mojej rodine a priateľom za podporu počas celého štúdia.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Název práce: Algoritmy založené na omezené expanzi - implementace a vyhodnocení

Autor: Jana Rapavá

Ústav: Informatický ústav Univerzity Karlovy

Vedoucí bakalářské práce: doc. Mgr. Zdeněk Dvořák, Ph.D, Informatický ústav Univerzity Karlovy

Abstrakt: Tato bakalářská práce navazuje na větu, která říká, že mnoho tříd grafů - konkrétně třídy s omezenou expanzí - má vlastnosti zjednodušující rozhodování grafových problémů definovatelných v logice prvního řádu. Důležitým příkladem takového problému je izomorfismus podgrafů. Cílem této práce je implementovat a otestovat navrhnutý algoritmus pro tento problém (který má lineární časovou složitost vzhledem k velikosti grafu, ve kterém hledáme podgraf).

Klíčová slova: izomorfismus podgrafů, omezená expanze, omezená stromová hloubka

Title: Algorithms based on bounded expansion - implementation and evaluation

Author: Jana Rapavá

Department: Computer Science Institute of Charles University

Supervisor: doc. Mgr. Zdeněk Dvořák, Ph.D, Computer Science Institute of Charles University

Abstract: This thesis builds upon the result that a lot of graph classes - namely classes with bounded expansion - have properties which make deciding graph problems definable in first-order logic easier. One very important example of such a problem is subgraph isomorphism. The purpose of this work is to implement and test proposed algorithm for this problem (which has a linear time complexity in relation to the size of graph we are looking for the subgraph in).

Keywords: subgraph isomorphism, bounded expansion, bounded treedepth

Názov práce: Algoritmy založené na obmedzenej expanzii - implementácia a vyhodnotenie

Autor: Jana Rapavá

Ústav: Ústav informatiky Karlovej univerzity

Vedúci bakalárskej práce: doc. Mgr. Zdeněk Dvořák, Ph.D, Ústav informatiky Karlovej univerzity

Abstrakt: Táto bakalárska práca stavia na výsledku, ktorý nám vraví, že veľa tried grafov - konkrétne triedy s obmedzenou expanziou - má vlastnosti zjednodušujúce rozhodovanie grafových problémov definovateľných v logike prvého rádu. Dôležitým príkladom takého problému je izomorfizmus podgrafov. Cieľom tejto práce je implementovať a otestovať navrhnutý algoritmus pro tento problém (ktorý má lineárnu časovú zložitosť vzhledem k veľkosti grafu, v ktorom hľadáme podgraf).

Kľúčové slová: izomorfizmus podgrafov, obmedzená expanzia, obmedzená stromová hĺbka

Obsah

Úvod

V praxi sa často stretávame s modelmi skutočnosti, ktoré používajú grafy. Konkrétny problém, ktorý potrebujeme vyriešiť, potom často môžme previesť na nejakú zo štandardných grafových úloh. Jednou z takýchto úloh je zistenie, či daný graf obsahuje daný podgraf - problém izomorfizmu podgrafov.

Tento problém má bohaté aplikácie v rozličných oblastiach, od modelovania molekúl, cez strojové učenie a rozpoznávanie vzorov, až po algoritmy počítačového videnia. [1]

Jeho formálna definícia je nasledovná:

Definícia. *[2]*
***Inštancia**: Grafy $G = (V_1, E_1)$, $H = (V_2, E_2)$.*
***Zadanie**: Obsahuje G podgraf izomorfný H? Inými slovami, existujú $V \subseteq V_1$, $E \subseteq E_1$ a bijekcia $f : V_2 \to V$, pre ktoré platí, že $|V| = |V_2|$, $|E| = |E_2|$ a $\{u, v\} \in E_2 \Leftrightarrow \{f(u), f(v)\} \in E$?*

Ako vidíme z [2], tento problém je NP-úplný; pravdepodobne teda pre jeho všeobecnú inštanciu neexistuje algoritmus s polynomiálnou časovou zložitosťou.

V súčasnosti známe algoritmy pre problém izomorfizmu podgrafov môžme rozdeliť do troch skupín.

Prvá skupina rieši problém v celej jeho všeobecnosti.

Patrí do nej napríklad triviálny algoritmus založený na backtrackingu [3], ktorý má časovú zložitosť $O(|V(G)|^{|V(H)|})$. Prerezávaním vyhľadávacieho stromu v tomto algoritme dostaneme efektívnejšie prístupy, ako napríklad [3] a [4].

Ďalší klasický algoritmus, vychádzajúci z detekcie trojuholníkov pomocou násobenia matíc, predstavili Nešetřil a Poljak [5]. Tento algoritmus má časovú zložitosť $O(|V(G)|^{\frac{\omega}{3} \cdot |V(H)|})$, kde ω je exponent maticového násobenia.

V druhej skupine sú stratégie poskytujúce približné, ale často dostatočne dobré riešenia - neurónové siete, simulované žíhanie, alebo genetické algoritmy [1].

V neposlednom rade vieme efektívne riešiť niektoré špeciálne triedy inštancií tohto problému.

Prípadom, kedy má H špecifický tvar, sa zaoberajú články [6] a [7].

Ak sú G a H rovinné grafy a H je pevne daný, existuje algoritmus s časovou zložitosťou $O(|V(G)|)$ [8] (kde multiplikatívna konštanta závisí exponenciálne na $|V(H)|$).

Pre triedy grafov uzavreté na minory je problém riešiteľný v polynomiálnom čase [9], ak je G z takejto triedy a H je pevne daný. Tento výsledok je možné rozšíriť na triedy grafov s lokálne zakázanými minormi [10].

S týmito triedami sú neporovnateľné triedy s obmedzenou stromovou šírkou, ktorými sa zaberá [11]. Z hlavného výsledku tohto článku plynie, že pre G s obmedzenou stromovou šírkou a H pevne daný je problém izomorfizmu podgrafov riešiteľný v lineárnom čase vzhľadom k veľkosti G.

Ak H zostane pevne dané, ale G má len lokálne obmedzenú stromovú šírku, dá sa na základe článku [12] dokázať, že pre problém existuje algoritmus s časovou zložitosťou $O(|V(G)|^{1+\frac{1}{k}})$, a to pre každé $k \geq 1$.

Do poslednej skupiny patrí aj algoritmus vychádzajúci z výsledkov Nešetřila a Mendeza [14], ktorého implementáciu a správanie v praxi skúma táto bakalárska práca.

Prvá kapitola sa zaoberá pojmami a výsledkami z teórie grafov, ktoré pre tento algoritmus potrebujeme.

Druhá kapitola popisuje samotnú implementáciu.

Záver obsahuje teoretické odhady parametrov algoritmu pre testované vstupy a ich porovnanie s reálnymi hodnotami.

1. Prehľad potrebnej teórie

V tejto kapitole sa nachádza stručný zoznam definícii a viet, ktoré budeme potrebovať v ďalších častiach práce.

Ako bolo spomenuté v úvode, pre triedy grafov uzavreté na minory je problém izomorfizmu podgrafov jednoduchší než vo všeobecnosti. Snaha o rozšírenie tohto (a niekoľkých ďalších) výsledkov viedla k zavedeniu pojmu triedy s obmedzenou expanziou. Tento pojem zaviedli Nešetřil a Mendez vo svojich článkoch [13] a [14], kde poskytli aj algoritmus pre hľadanie izomorfných podgrafov v grafoch, ktoré patria do takýchto tried.

Tento algoritmus využíva rozklad grafu G s obmedzenou stromovou hĺbkou. Aby sme sa mohli zaoberať touto problematikou, potrebujeme najskôr definovať nasledovné pojmy:

Definícia. Vzdialenosť $d(x, y)$ *medzi vrcholmi* x *a* y *v grafe je dĺžka najkratšej cesty spájajúcej* x *a* y*, alebo* ∞*, ak* x *a* y *patria do rozličných komponent súvislosti.*

Definícia ([14])**.** Polomer *súvislého grafu* G *je definovaný ako:*

$$\rho(G) = \min_{r \in V(G)} \max_{x \in V(G)} d(x, r).$$

Polomer nesúvislého grafu je maximum z polomerov jeho komponent súvislosti.

Definícia ([14])**.** H *je* minor *grafu* G*, ak je* H *možné získať z* G *postupnosťou kontrakcií hrán, odstránení hrán a odstránení vrcholov; značíme* $H < G$*.* Hĺbka minoru H *grafu* G $\operatorname{depth}(H, G)$ *je najmenší polomer časti* G*, ktorú musíme kontrahovať, aby sme dostali* H*.*

Definícia ([13])**.** *Buď* $f : \mathbb{N} \to \mathbb{R}$*. Graf* G *má* expanziu obmedzenú f*, ak pre každý* $G' < G$ *získaný kontrakciou disjunktného zjednotenia súvislých grafov s polomerom* $\leq r$ *a odstraňovaním vrcholov platí, že hranová hustota* G' *je zhora ohraničená* $f(r)$*.*

Definícia ([13])**.** *Trieda grafov* $\mathcal{C}$ *má* obmedzenú expanziu*, ak existuje funkcia* $f : \mathbb{N} \to \mathbb{R}$ *t.ž. každý graf z triedy* $\mathcal{C}$ *má expanziu obmedzenú* f*.*

Príklady tried s obmedzenou expanziou ([13]):

triedy grafov s obmedzeným stupňom Trieda grafov s maximálnym stupňom najviac k má expanziu obmedzenú funkciou $f(r) = k^{r+1}$.

rovinné grafy Rovinný graf s n vrcholmi má maximálne $3n - 6$ hrán a každý minor rovinného grafu je rovinný. Táto trieda má teda expanziu obmedzenú funkciou $f(r) = 3$.

vlastné triedy uzavreté na minory Každá vlastná trieda grafov uzavretá na minory má expanziu obmedzenú konštantou. Naopak, každá trieda grafov, ktorej expanzia je obmedzená konštantou, je súčasťou nejakej vlastnej triedy uzavretej na minory.

vlastné topologicky uzavreté triedy Tieto triedy sú definované (potenciálne nekonečnou) množinou S zakázaných konfigurácií: graf G patrí do triedy, ak nijaké delenie grafu z S nie je izomorfné podgrafu G. Tieto triedy majú expanziu obmedzenú funkciou $f(r) = 2^{r-1}(\min_{H \in S} |V(H)|)^{2^{r+1}}$.

Teraz predstavíme definície a výsledky z niekoľkých ďalších potrebných oblastí.

1.1 Acyklické orientácie grafov

Definícia. *Buď $G = (V, E)$ neorientovaný graf.* Orientácia *grafu G je orientovaný graf $G' = (V, E')$ t.ž. $(u, v) \in E \Leftrightarrow (u, v) \in E' \vee (v, u) \in E'$.*

Definícia ([13])**.** *Maximálny priemerný stupeň grafu G je definovaný ako*

$$\mathrm{mad}(G) = \max_{H \subseteq G} \frac{2|E(H)|}{|V(H)|}$$

Veta 1 ([14])**.** *Buď $G = (V, E)$, $|V| = n$, $|E| = m$. G má orientáciu s maximálnym vstupným stupňom k práve vtedy, keď $k \geq \frac{\mathrm{mad}(G)}{2}$. Navyše existuje algoritmus, ktorý nájde acyklickú orientáciu G s maximálnym vstupným stupňom $\lfloor \mathrm{mad}(G) \rfloor$ v čase $O(n+m)$.*

Algoritmus nájde v aktuálnom grafe vrchol so stupňom $\leq \mathrm{mad}(G)$, všetkým hranám, ktoré sú s ním incidentné, priradí orientáciu do tohto vrcholu, odstráni vrchol a zavolá sa rekurzívne na zvyšok grafu.

1.2 Tranzitívne fraternálne augmentácie

Definícia ([14])**.** *Buď $G = (V, E)$ orientovaný graf.* Striktná 1-tranzitívna fraternálna augmentácia *je orientovaný graf $H = (V, E')$, kde E' obsahuje všetky hrany z G a navyše spĺňa nasledovné podmienky:*

- $(x, z), (z, y) \in E \Rightarrow (x, y) \in E'$
- $(x, z), (y, z) \in E \Rightarrow (x, y), (y, x)$*, alebo oboje ležia v E'*
- *E' neobsahuje nijaké ďalšie hrany.*

Definícia ([14])**.** Striktná tranzitívna fraternálna augmentácia *orientovaného grafu G je postupnosť $G = G_1 \subseteq G_2 \subseteq \ldots G_i \subseteq G_{i+1} \subseteq \ldots$, pre ktorú platí, že G_{i+1} je striktná 1-tranzitívna fraternálna augmentácia G_i pre $i \geq 1$.*

Veta 2 ([14])**.** *Buď G orientovaný graf s maximálnym vstupným stupňom d. Potom je možné nájsť striktnú 1-tranzitívnu fraternálnu augmentáciu grafu G v čase $O(d^2|V(G)|)$.*

Z Dôsledku 3.6 v [14] plynie, že 1-tranzitívna fraternálna augmentácia grafu zachováva obmedzenú expanziu. Z toho a z predchádzajúcej vety dostávame:

Veta 3 ([14])**.** *Buď G graf z triedy s obmedzenou expanziou. Potom striktnú tranzitívnu fraternálnu augmentáciu grafu G je možné nájsť v čase $O(|V(G)|)$.*

1.3 Centrované ofarbenia

Definícia ([13]). *Buď $p \in \mathbb{N}$. Potom* p-centrované ofarbenie *grafu G je ofarbenie grafu G, v ktorom má každý súvislý podgraf H jednu z nasledovných vlastností:*

- *v H sa nejaká farba vyskytuje práve raz*
- *v H sa vyskytuje aspoň p rozličných farieb.*

Veta 4 ([14]). *Buď G graf z triedy s obmedzenou expanziou. Označme $R(p) = 1 + (p-1)(2 + \lceil \log_2 p \rceil)$. Buď G' striktná tranzitívna fraternálna augmentácia G. Potom každé dobré ofarbenie $G'_{R(p)}$ je p-centrované ofarbenie grafu G.*

Pri určovaní horného odhadu na potrebný počet farieb použijeme tvrdenie, že každý graf je $(\mathrm{mad}(G) + 1)$-ofarbiteľný [14]. Potrebujeme teda zhora odhadnúť $\mathrm{mad}(G'_{R(p)})$. Pretože každá striktná tranzitívna fraternálna augmentácia je orientovaným grafom, môžme z Vety 1 písať:

$$\mathrm{mad}(G'_{R(p)}) \leq 2\Delta^-(G'_{R(p)})$$

(kde $\Delta^-(G)$ značí maximálny vstupný stupeň orientovaného grafu G).

Graf $G'_{R(p)}$ je z triedy s obmedzenou expanziou; označme obmedzujúcu funkciu f. Z Dôsledku 5.3 v [13] potom plynie:

Veta 5.

$$\Delta^-(G'_{R(p)}) \leq \rho_{R(p)}(2f(0), f(2^{R(p)+1} - 1)).$$

Polynóm ρ zadefinujeme na základe niekoľkých pomocných polynómov [13]. Najprv si zadefinujeme $\Phi_i(x_0, \ldots, x_i)$ pre $i \in \mathbb{N}$:

$\Phi_0(x_0) = x_0$
$\Phi_i(x_0, \ldots, x_i) = \Phi_{i-1}(x_0, \ldots, x_i) + ((2\Phi_{i-1}(x_0, \ldots, x_{i-1} + 1)x_0)^{2i+1}x_i.$

Pomocou neho zadefinujeme polynóm $P_i(x, y) = \Phi_i(x + y, y, \ldots, y)$.

Ďalej budeme potrebovať polynómy Q a Q_i:

$Q(x, y) = x(x + 1) + Q_0(x, y)$
$Q_i(x, y) = P_{i+1}(x + 1, y).$

A vzájomnou rekurziou definujeme dvojicu polynómov ρ a ρ':

$\rho_0(x, y) = x$
$\rho_i(x, y) = Q(\rho_{i-1}(x, y), \rho'_{i-1}(x, y))$

$\rho'_0(x, y) = x$
$\rho'_i(x, y) = Q_i(\rho_{i-1}(x, y), \rho'_{i-1}(x, y)).$

Potrebný počet farieb je teda (z Vety 5):

$$X(p) \leq 2\rho_{R(p)}(2f(0), f(2^{R(p)+1} - 1) + 1.$$

Štandardný žravý algoritmus nám tak umožní nájsť p-centrované ofarbenie grafu G v čase $O(|V(G)|)$.

1.4 Rozklady s obmedzenou stromovou hĺbkou

Definícia ([13]). Uzáver *clos(F) zakoreneného lesa F je graf s množinou vrcholov* $V(F)$ *a množinou hrán* $\{\{x, y\}$*: x je predok y v* $F \wedge x \neq y\}$

Definícia ([13]). Stromová hĺbka *súvislého grafu G je minimálna výška zakoreneného stromu, ktorého uzáver obsahuje G ako podgraf.*

Veta 6 ([14]). *Buď* $k \leq p - 1$*. Ak má graf G p-centrované ofarbenie k farbami, potom má stromovú hĺbku* $\leq k$*.*

Myšlienka, že centrované ofarbenie grafu sa dá použiť na konštrukciu jeho rozkladu s obmedzenou stromovou hĺbkou, pochádza z článku [14]. Algoritmus uvedený v tomto článku ale nefunguje napríklad v prípade cesty na troch vrcholoch. Tu však použijeme jeho základnú ideu: ak máme centrované ofarbenie grafu G p farbami, vieme z neho skonštruovať zakorenený les výšky p, ktorý obsahuje G vo svojom uzávere.

Algoritmus z každej komponenty súvislosti grafu G skonštruuje strom nasledovným spôsobom: nájde vrchol, ktorého farba sa v komponente vyskytuje práve raz (ten existuje z vety 4). Tento vrchol vyhlási za koreň stromu a zavolá sa rekurzívne na komponentu bez tohto vrcholu. Jeho odobratím sa komponenta mohla rozpadnúť - rekurzívne teda skonštruuje strom pre každú z nových komponent. Tieto stromy sa potom pridajú do výsledku ako podstromy aktuálneho koreňa.

1.5 Hľadanie izomorfných podgrafov

Jednou z možností využitia rozkladov s obmedzenou stromovou hĺbkou je zistenie, či je graf H izomorfný s nejakým podgrafom grafu G, a to v lineárnom čase vzhľadom k veľkosti grafu G. [14]

Podľa výsledkov popísaných v sekcii 1.2 existuje pre každú triedu s obmedzenou expanziou $\mathcal{C}$ funkcia $f(k)$ t.ž. pre každý graf G z $\mathcal{C}$ existuje jeho striktná tranzitívna fraternálna augmentácia G', pre ktorú platí: $\text{mad}(G'_k) \leq f(k)$.

Podľa výsledkov sekcie 1.3 má teda každý graf G z $\mathcal{C}$ p-centrované ofarbenie $f(R(p)) + 1$ farbami.

Nasledujúca veta plynie z vety 6 (a je slabšou verziou Dôsledku 3.8 v [14]):

Veta 7. *Nech G má p-centrované ofarbenie. Potom podgraf G indukovaný ľubovoľnými* $p - 1$ *farbami má stromovú hĺbku* $\leq p - 1$*.*

Ak sú G a H grafy a G je z triedy s obmedzenou expanziou, prípadný podgraf H v G má najviac $|V(H)|$ farieb. Ak teda nájdeme $(|V(H)| + 1)$-centrované ofarbenie grafu G, vieme, že H je podgrafom G práve vtedy, keď v G existuje nejakých $|V(H)|$ farieb, ktoré indukujú podgraf obsahujúci H [15].

Teraz sformulujeme algoritmus, ktorý určí, či sa v grafe G, pre ktorý máme rozklad s obmedzenou stromovou hĺbkou, nachádza podgraf izomorfný H [15].

Algoritmus prijíma ako vstup strom, ktorý je rozkladom s obmedzenou stromovou hĺbkou pre G, a graf H. Tento rozklad prechádzame do hĺbky, takže počas behu algoritmu máme G vždy rozdelené na tri časti: už prejdenú (spracovanú) časť, aktuálny vrchol a cestu z koreňa do neho, a ešte nespracovanú časť.

V každom vrchole stromu si algoritmus pamätá množinu funkcií $F = \{f | f : V(H) \to (P \cup \{S, N\})\}$, kde:

- P je množina vrcholov na aktuálnej ceste z koreňa
- S je značka, že vrchol je spracovaný
- N je značka, že vrchol je nespracovaný (nemá obraz)

a f nezobrazí žiadne dva rôzne vrcholy H na ten istý vrchol P.

Aby sme presne charakterizovali funkcie, ktoré patria do F, použijeme nasledovnú definíciu:

Definícia. *Funkcia $\varphi : V(F) \to V(G)$ určuje podgraf F v G, ak je prostá a hrany F zobrazuje na hrany G.*

Podgraf H grafu G indukovaný vrcholmi, ktoré f nezobrazí na N, si označme H_f. Potom f patrí do F práve vtedy, keď je f možné rozšíriť na funkciu φ určujúcu podgraf H_f v G t.ž. pre každý vrchol $v \in f^{-1}(S)$ patrí $\varphi(v)$ do už spracovanej časti grafu G.

Pre akciu pri príchode do vrcholu si zadefinujeme pomocný pojem prípustného vrcholu.

Definícia. *Vrchol $v \in V(H)$ je* prípustný *pre zobrazenie f a vrchol $x \in V(G)$, ak je $f(v) = N$ a pre každé n suseda v platí buď $f(n) = N$, alebo $f(n) \in P$ a medzi x a $f(n)$ vedie v G hrana.*

Z definície stromovej hĺbky vieme, že x a vrchol n t.ž. $f(n) = S$ nemôžu nikdy byť spojené hranou.

Pri príchode do vrcholu x sa spočíta množina $F' = F \cup \{f \cup v \to x | f \in F$, v prípustný pre f a $x\}$, kde F je množina funkcií z rodiča x.

Synovia x sa spracujú rekurzívne.

Pri odchode z vrcholu x prejdeme všetky funkcie v jeho množine funkcií a zobrazenia $v \to x$ nahradíme zobrazeniami $v \to S$.

Po návrate do koreňa usúdime, že H je podgraf G práve vtedy, keď v koreni existuje funkcia f t.ž. $\forall v \in V(H)\ f(v) = S$.

Veta 8. *Časová zložitosť tohto algoritmu je*

$$O((d+2)^{|V(H)|}|V(H)|^2|V(G)|),$$

kde d je stromová hĺbka G.

Algoritmus budeme aplikovať na grafy so stromovou hĺbkou $|V(H)|$, čo dáva časovú zložitosť $O(|V(H)|^{|V(H)|+2}|V(G)|)$.

1.6 Zhrnutie

Celkový algoritmus pre hľadanie izomorfných podgrafov v grafoch s obmedzenou expanziou je nasledovný:

Sú zadané grafy G a H; buď $p = |V(H)| + 1$.

1. Nájdeme $R(p)$-tú striktnú 1-tranzitívnu fraternálnu augmentáciu grafu G.
2. Nájdeme jej ofarbenie; to nám dá p-centrované ofarbenie grafu G.

3. Pomocou uvedeného algoritmu pre hľadanie izomorfných podgrafov pre každú množinu $|V(H)|$ farieb C otestujeme, či je $H \leq G_C$ (podgrafu G indukovaného vrcholmi, ktorých farby sú z množiny C).

Časová zložitosť kroku 1 je $(X(p)|V(G)||V(H)| \log |V(H)|)$,
kroku 2 je $O(X(p)|V(G)|)$. [14]

Existuje najviac $\binom{X(p)}{|V(H)|}$ možných množín C, takže v kroku 3 spustíme $\binom{X(p)}{|V(H)|}$-krát algoritmus s časovou zložitosťou $O(|V(H)|^{|V(H)|+2}|V(G)|)$ (pozri Vetu 8).

To nám umožňuje vysloviť nasledovnú vetu:

Veta 9. *Celková časová zložitosť prezentovaného algoritmu je*

$$O\left(\binom{X(p)}{|V(H)|}|V(H)|^{|V(H)|+2}|V(G)|\right).$$

2. Implementácia

2.1 Užívateľské rozhranie

Pre beh tohto programu je potrebná Java 7. Program je distribuovaný ako JAR archív a spúšťa sa z príkazového riadka. Pri spustení prijíma štyri argumenty - súbor obsahujúci prvý graf, súbor obsahujúci druhý graf, parameter p pre p-centrované ofarbenie a počet farieb, ktoré je možné použiť pri hľadaní p-centrovaného ofarbenia.

Formát vstupných grafov je popísaný nasledovnou gramatikou:
Graph = Node\n |Node\n Graph
Node = Integer–List
List = [] |[Int(,Int)*]
Vrcholy grafu sú indexované od 0.

Pokiaľ vstupné grafy nezodpovedajú tomuto formátu, program vyhadzuje výnimku.

Výsledok výpočtu sa vypisuje na štandardný výstup.

2.2 Návrh programu

Program je možné intuitívne rozdeliť na tri časti.

Prvá obsahuje potrebné grafové dátové štruktúry, druhá implementuje algoritmus pre stromovú dekompozíciu, a tretia časť je tvorená samotným algoritmom pre detekciu izomorfných podgrafov.

2.2.1 Grafy

Do tejto časti patria zdrojové súbory *DegreeQueue.java*, *Edges.java*, *Graph.java*, *Node.java* a *NodeInterface.java*.

Dátová štruktúra *DegreeQueue* umožňuje v grafe rekurzívne vyberať vrchol s najnižším stupňom tak, aby jedna operácia výberu mala amortizovanú časovú zložitosť $O(1)$.

Edges.java obsahuje metódy pre prácu so zoznamami hrán a pre prevody medzi zoznamami hrán a grafmi.

Neorientovaný graf je reprezentovaný zoznamom objektov, ktoré predstavujú vrcholy. Každý takýto vrchol obsahuje svoj názov, stupeň, stav navštívenosti a odkaz na zoznam hrán, ktoré z neho vedú. Každá hrana sa zároveň odkazuje na tú istú hranu v opačnom smere.

Vzťah "byť hranou v opačnom smere" je reprezentovaný triedou *LinkEdge*. Tá obsahuje odkaz na vrchol na druhom konci hrany, aby sme vždy, keď označíme súčasný vrchol ako zmazaný, vedeli efektívne znížiť stupeň všetkých jeho susedov. To je tiež dôvod, prečo je tento odkaz reprezentovaný typom *LinkEdge* a nie typom *Edge*.

Táto dátová štruktúra má tú nevýhodu, že ju nie je možné úplne inicializovať už pri načítaní vstupu - na jej vytvorenie je potrebné prejsť vstupný graf dvakrát. Druhý prechod má na starosti metóda *postInitNodes*.

Orientované grafy sú reprezentované poľom zoznamov predchodcov, čo nám umožňuje nájsť striktnú tranzitívnu fraternálnu augmentáciu v lineárnom čase.

2.2.2 Rozklady s obmedzenou stromovou hĺbkou

Táto časť pozostáva zo zdrojových súborov *Tree.java* a *BoundedDepthDecomposition.java*.

V druhom z týchto súborov je implementovaný algoritmus pre hľadanie rozkladov s obmedzenou stromovou hĺbkou. Ten používa nasledovné pomocné metódy:

findOrientation()

Táto metóda implementuje algoritmus, ktorý nájde acyklickú orientáciu grafu s nízkym vstupným stupňom.

findAugmentation()

Táto metóda počíta striktnú tranzitívnu fraternálnu augmentáciu zadaného grafu tak, že v cykle hľadá a pridáva do grafu tranzitívne a fraternálne hrany.

Aby sme zabránili pridávaniu hrán, ktoré už existujú, vytvorený graf prevedieme na zoznam hrán, ten utriedime priehradkovým triedením, odstránime duplicity a výsledný zoznam prevedieme na graf.

Výpočet by bolo možné zrýchliť tak, že si každý vrchol bude pamätať poslednú iteráciu, v ktorej sa mu zvýšil stupeň, a dôjde k prepočítavaniu len v prípade, kedy sa v predchádzajúcej iterácii zmenil stupeň aspoň jedného zo zúčastnených vrcholov. Ďalej nie je potrebné, aby algoritmus, ktorý dopĺňa tranzitívne hrany, testoval vrcholy, ktoré už majú stupeň $n - 1$.

findColoring()

Táto metóda hľadá ofarbenie grafu zadaným počtom farieb. Štruktúra rekurzie je podobná ako vo funkcii *findOrientation()*.

V prípade, kedy graf nie je možné ofarbiť daným počtom farieb, metóda vyhadzuje výnimku.

findFullDecomposition

Táto metóda rozloží graf na komponenty súvislosti a potom nájde rozklad s obmedzenou stromovou hĺbkou pre každú z nich.

Rozklad grafu s obmedzenou stromovou hĺbkou sa konštruuje z centrovaného ofarbenia - vždy nájdeme v grafe vrchol, ktorého farba sa vyskytuje len raz, odstránime ho z grafu, zvyšok grafu rozložíme na komponenty súvislosti a zavoláme sa na každú z nich rekurzívne.

Mohlo by sa zdať, že tieto opakované rozklady na komponenty súvislosti počas celého behu algoritmu zaberú viac ako lineárny čas, ale to sa nestane, pretože uvažované grafy sú z tried s obmedzenou expanziou [13].

2.2.3 Hľadanie izomorfných podgrafov

Táto časť sa skladá zo zdrojových súborov *Function.java*, *State.java* a *SubgraphIsomorphism.java*.

Implementovaný algoritmus je presne ten popísaný v prvej kapitole práce.

Testovacie dáta pre tento algoritmus sa nachádzajú v adresári *tests* odovzdaného súboru *bakalarka.zip*.

Záver

Prezentovaný algoritmus pre hľadanie izomorfných podgrafov je parametrizovaný dvomi hodnotami: počtom iterácii R(p) procedúry, ktorá hľadá striktné 1-tranzitívne fraternálne augmentácie, a počtom farieb X(p), ktoré potrebujeme na nájdenie p-centrovaného ofarbenia pôvodného grafu.

V sekcii 1.3 sme sa tieto hodnoty pokúsili zhora odhadnúť.Teraz tieto odhady porovnáme so skutočnými hodnotami parametrov, získanými spustením algoritmu na vybrané grafy.

Pretože polynóm ρ rastie extrémne rýchlo, nie je možné použiť jeho presnú hodnotu. Keď si však uvedomíme, že ρ je v prvom argumente rastúci, môžme nájsť pre jeho hodnotu dolný odhad, ktorý bude pre porovnanie s experimentálnymi výsledkami dostatočne presný. (Na odhad pre rovinné grafy bolo použité $i = 2$, pre 4-regulárne grafy $i = 1$.)

V Tabuľke 2.1 sú zobrazené hodnoty parametrov pre vybrané rovinné grafy a ich porovnanie s teoretickými odhadmi.

Tieto testovacie dáta sa nachádzajú v adresári *tests-planar1* odovzdaného súboru *bakalarka.zip*. Testy majú tvar adresárov, ktoré obsahujú zvolený graf *graph1* a trojuholník *graph2*.

test1, test11 Bipartitný graf 2x100.

test2, test12 5-árny strom s 3 úrovňami.

test3, test13 Triangulácia na 7 vrcholoch.

test4, test14 Štvorcová mriežka 10x5.

test5, test15 Hviezda na 101 vrcholoch.

test6, test16 Cesta na 100 vrcholoch a 2 vrcholy spojené hranou práve s každým vrcholom cesty.

test7, test17 Zjednotenie 6 5-cyklov.

test8, test18 Trojuholníková mriežka 10x5.

test9, test19 Cesta na 100 vrcholoch.

test10, test20 5-cyklus, ktorý má každú hranu nahradenú trojuholníkom.

V Tabuľke 2.2 sa nachádzajú výsledky pre vstupy, kde je *graph1* zvolený graf z Tabuľky 2.1 a *graph2* je štvorcyklus. Tieto dáta sa tiež vyskytujú v adresári *tests-planar1* odovzdaného súboru *bakalarka.zip*.

V Tabuľke 2.3 sú hodnoty pre náhodné jednoduché rovinné grafy na 64 vrcholoch vygenerované programom **plantri**. Všetky vygenerované náhodné grafy boli prevedené na zoznam hrán príkazom **nauty-showg -e** a prekonvertované do používaného vstupného formátu pomocou skriptu **edges2input.sh**. Tieto testovacie dáta sa nachádzajú v adresári *tests-planar2* odovzdaného súboru *bakalarka.zip*. Testy majú tvar adresárov, ktoré obsahujú vygenerovaný graf *graph1* a trojuholník *graph2*.

Tabulka 2.1: Parametre pre vybrané rovinné grafy
p=4, $R(p) = 13$, $X(p) > 10^{2486}$

Vstup	Hodnota R(p)	Hodnota X(p)
test1	3	5
test2	9	8
test3	3	6
test4	5	50
test5	3	3
test6	9	102
test7	5	15
test8	6	49
test9	7	100
test10	3	9

Tabulka 2.2: Parametre pre vybrané rovinné grafy
p=5, $R(p) = 21$, $X(p) > 10^{2486}$

Vstup	Hodnota R(p)	Hodnota X(p)
test11	3	5
test12	9	8
test13	3	6
test14	5	50
test15	3	3
test16	9	102
test17	5	15
test18	6	49
test19	7	100
test20	3	9

Tabulka 2.3: Parametre pre náhodné jednoduché rovinné grafy
$|V(G)|$=64, p=4, $R(p) = 13$, $X(p) > 10^{2486}$

Vstup	Hodnota R(p)	Hodnota X(p)
test1	13	46
test2	13	43
test3	13	43
test4	13	39
test5	13	46
test6	13	45
test7	13	45
test8	13	43
test9	9	46
test10	13	43

V Tabuľke 2.4 sa nachádzajú výsledky pre vstupy, kde je *graph1* náhodný graf z Tabuľky 2.3 a *graph2* je štvorcyklus. Tieto dáta sa tiež vyskytujú v adresári *tests-planar2* odovzdaného súboru *bakalarka.zip*.

Tabulka 2.4: Parametre pre náhodné jednoduché rovinné grafy
$|V(G)|$=64, p=5, $R(p) = 21$, $X(p) > 10^{2486}$

Vstup	Hodnota R(p)	Hodnota X(p)
test11	15	47
test12	16	44
test13	16	44
test14	18	41
test15	15	46
test16	14	45
test17	15	45
test18	20	46
test19	9	46
test20	17	44

V Tabuľke 2.5 sú hodnoty pre náhodné fullerény (rovinné kubické grafy, ktorých každá stena má veľkosť 5 alebo 6) na 200 vrcholoch vygenerované programom **fullgen**. Všetky vygenerované náhodné grafy boli prevedené na zoznam hrán príkazom **nauty-showg -e** a prekonvertované do používaného vstupného formátu pomocou skriptu **edges2input.sh**. Tieto testovacie dáta sa nachádzajú v adresári *tests-planar3* odovzdaného súboru *bakalarka.zip*. Testy majú tvar adresárov, ktoré obsahujú vygenerovaný graf *graph1* a trojuholník *graph2*.

Tabulka 2.5: Parametre pre náhodné fullerény
$|V(G)|$=200, p=4, $R(p) = 13$, $X(p) > 10^{2486}$

Vstup	Hodnota R(p)	Hodnota X(p)
test1	5	200
test2	5	200
test3	5	200
test4	5	200
test5	5	200
test6	5	200
test7	5	200
test8	5	200
test9	5	200
test10	5	200

V Tabuľke 2.6 sa nachádzajú výsledky pre vstupy, kde je *graph1* náhodný graf z Tabuľky 2.5 a *graph2* je štvorcyklus. Tieto dáta sa tiež vyskytujú v adresári *tests-planar3* odovzdaného súboru *bakalarka.zip*.

Tabulka 2.6: Parametre pre náhodné fullerény
$|V(G)|$=200, p=5, $R(p) = 21$, $X(p) > 10^{2486}$

Vstup	Hodnota R(p)	Hodnota X(p)
test11	5	200
test12	5	200
test13	5	200
test14	5	200
test15	5	200
test16	5	200
test17	5	200
test18	5	200
test19	5	200
test20	5	200

Tabulka 2.7: Parametre pre náhodné 4-regulárne grafy
$|V(G)|$=150, p=4, $R(p) = 13$, $X(p) > 10^{138095}$

Vstup	Hodnota R(p)	Hodnota X(p)
test1	4	150
test2	4	150
test3	4	150
test4	4	150
test5	4	150
test6	4	150
test7	4	150
test8	4	150
test9	4	150
test10	4	150

Druhá sada testov pozostáva z náhodných 4-regulárnych grafov generovaných programom **nauty-genrang**. Výstup tohto programu bol prevedený do používaného vstupného formátu pomocou príkazu **nauty-showg -e** a skriptu **edges2input.sh**.

V Tabuľke 2.7 sú zobrazené hodnoty parametrov pre náhodné 4-regulárne grafy na 150 vrcholoch. Tieto testovacie dáta sa nachádzajú v adresári *tests-4regular1* odovzdaného súboru *bakalarka.zip*. Testy majú tvar adresárov, ktoré obsahujú vygenerovaný graf *graph1* a trojuholník *graph2*.

V Tabuľke 2.8 sa nachádzajú výsledky pre vstupy, kde je *graph1* zvolený graf z Tabuľky 2.7 a *graph2* je štvorcyklus. Tieto dáta sa tiež vyskytujú v adresári *tests-4regular1* odovzdaného súboru *bakalarka.zip*.

V Tabuľke 2.9 sú zobrazené hodnoty parametrov pre náhodné 4-regulárne grafy na 200 vrcholoch. Tieto testovacie dáta sa nachádzajú v adresári *tests-4regular2* odovzdaného súboru *bakalarka.zip*. Testy majú tvar adresárov, ktoré obsahujú vygenerovaný graf *graph1* a trojuholník *graph2*.

Tabulka 2.8: Parametre pre náhodné 4-regulárne grafy
$|V(G)|$=150, p=5, $R(p) = 21$, $X(p) > 10^{138095}$

Vstup	Hodnota R(p)	Hodnota X(p)
test11	4	150
test12	4	150
test13	4	150
test14	4	150
test15	4	150
test16	4	150
test17	4	150
test18	4	150
test19	4	150
test20	4	150

Tabulka 2.9: Parametre pre náhodné 4-regulárne grafy
$V(G)|$=200, p=4, $R(p) = 13$, $X(p) > 10^{138095}$

Vstup	Hodnota R(p)	Hodnota X(p)
test1	4	200
test2	4	200
test3	4	200
test4	4	200
test5	4	200
test6	4	200
test7	4	200
test8	4	200
test9	4	200
test10	4	200

V Tabuľke 2.10 sa nachádzajú výsledky pre vstupy, kde je *graph1* náhodný graf z Tabuľky 2.9 a *graph2* je štvorcyklus. Tieto dáta sa tiež vyskytujú v adresári *tests-4regular2* odovzdaného súboru *bakalarka.zip*.

Namerané hodnoty $X(p)$ sú pre väčšinu testovaných grafov blízke $|V(G)|$. Pre dosiahnutie zmysluplných záverov by bolo potrebné uskutočniť testy na grafoch s vyšším počtom vrcholov. Žiaľ, pre $|V(G)| > 200$ som narazila na obmedzenia dostupného hardvéru, ktoré ďalšie testovanie znemožnili.

Zo súčasných výsledkov však môžme usúdiť, že popísaný algoritmus nie je pre reálne grafy prakticky použiteľný, pretože multiplikatívna konštanta $\binom{X(p)}{|V(H)|}$ je príliš veľká.

Tabulka 2.10: Parametre pre náhodné 4-regulárne grafy
$V(G)|$=200, p=5, $R(p) = 21$, $X(p) > 10^{138095}$

Vstup	Hodnota R(p)	Hodnota X(p)
test11	4	200
test12	4	200
test13	4	200
test14	4	200
test15	4	200
test16	4	200
test17	4	200
test18	4	200
test19	4	200
test20	4	200

Zoznam použitej literatúry

[1] MESSMER, Bruno T., a Horst BUNKE. *Efficient subgraph isomorphism detection: A decomposition approach*. IEEE Transactions on Knowledge and Data Engineering 12 (2) (2000) 307-323 [online]. Dostupné z: IEEExplore. DOI 10.1109/69.842269.

[2] GAREY, Michael R., a David S. JOHNSON. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979. ISBN 0-7167-1045-5.

[3] ULLMAN, J.R. *An Algorithm for Subgraph Isomorphism*. Journal of the ACM 23 (1) (1976) 31-42 [online]. Dostupné z: ACM DL. DOI 10.1145/321921.321925.

[4] HARALICK, R.M. a G.L. ELLIOT. *Increasing Tree Search Efficiency for Constraint Satisfaction Problems*. Artificial Intelligence 14 (3) (1980) 263-313 [online]. Dostupné z: ScienceDirect. DOI 10.1016/0004-3702(80)90051-X.

[5] NEŠETŘIL, J., a S. POLJAK. *On the complexity of the subgraph problem*. Commentationes Mathematicae Universitatis Carolinae 26 (2) (1985) 415-419 [online]. Dostupné z: The European Digital Mathematics Library. ISSN: 0010-2628.

[6] ALON, N., R. YUSTER a U. ZWICK. *Color-coding*. Journal of the ACM 42 (4) (1995) 844-856 [online]. Dostupné z: ACM DL. DOI 10.1145/210332.210337.

[7] ALON, N., R. YUSTER a U. ZWICK. *Finding and counting given length cycles*. Algorithmica 17 (3) (1997) 209-223 [online]. Dostupné z: SpringerLink. DOI 10.1007/BF02523189.

[8] EPPSTEIN, D. *Subgraph isomorphism in planar graphs and related problems*. Journal of Graph Algorithms and Applications 3 3 (1999) 1-27 [online]. Dostupné z: Journal of Graph Algorithms and Applications. DOI 10.7155/jgaa.00014.

[9] FLUM, J., a M. GROHE. *Fixed parameter tractability, definability, and model checking*. SIAM Journal on Computing 31 1 (2001) 113-145 [online]. Dostupné z: ACM DL. DOI 10.1137/S0097539799360768.

[10] DAWAR, A., M. GROHE a S. KREUTEZER. *Locally excluding a minor*. 22nd Annual IEEE Symposium on Logic in Computer Science (2007) 270-279 [online]. Dostupné z: IEEE Computer Society CS DL. DOI 10.1109/LICS.2007.31.

[11] COURCELLE, B. *The monadic second-order logic of graphs. I. recognizable sets of finite graphs*. Information and Computation 85 1 (1990) 12-75 [online]. Dostupné z: ScienceDirect. DOI 10.1016/0890-5401(90)90043-H.

[12] FRICK, M. a M. GROHE. *Deciding first-order properties of locally tree-decomposable structures*. Journal of the ACM 48 6 (2001) 1184-1206 [online]. Dostupné z: ACM DL. DOI 10.1145/504794.504798.

[13] NEŠETŘIL, J., a P. Ossona de MENDEZ. *Grad and classes with bounded expansion I. Decompositions*. European Journal of Combinatorics 29 (3) (2008) 760-776 [online]. Dostupné z: ScienceDirect. DOI 10.1016/j.ejc.2006.07.013.

[14] NEŠETŘIL, J., a P. Ossona de MENDEZ. *Grad and classes with bounded expansion II. Algorithmic Aspects*. European Journal of Combinatorics 29 (2008) 777-791 [online]. Dostupné z: ScienceDirect. DOI 10.1016/j.ejc.2006.07.014.

[15] DVOŘÁK, Z. 2014, osobná komunikácia.