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

Vrcholová barvení grafů

Vertex colorings of graphs

2017

Jakub Přibylík

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

Zadání bakalářské práce

Student: **Jakub Přibylík**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 1103R031 Výpočetní matematika

Téma: Vrcholová barvení grafů
Vertex colorings of graphs

Jazyk vypracování: čeština

Zásady pro vypracování:

Vrcholová barvení grafů patří mezi klasická a dobře prozkoumaná témata teorie grafů. Chromatické číslo udává nejmenší počet barev, které jsou potřeba na dobré vrcholové barvení grafu tak, aby každé dva sousední vrcholy byly obarveny různou barvou. Je známo, že určení chromatického čísla obecného grafu je NP-těžká úloha.

Vrcholová barvení grafů nejsou jen samoúčelným či akademickým problémem, neboť barvení modeluje odlišnost a využije se při řešení řady praktických problémů. Typickým příkladem jsou přiřazovací a rozvrhová schémata.
Cílem bakalářské práce je zpracovat přehled základních typů algoritmů včetně jejich složitosti: algoritmy heuristické, přesné, paralelní, decetralizované, ...
Součástí práce by měla být i řada ilustrativních příkladů, na kterých bude demonstrován jednak princip algoritmu, ale třeba také jejich složitost.

Práci lze rozdělit do následujících částí:
- studium literatury a elektronických zdrojů
- zpracování přehledu různých typů algoritmů a jejich složitosti
- výběr a prezentace vhodných příkladů

V budoucnu mohou na tuto bakalářskou práci navázat další práce, které se budou věnovat řešení vybraných praktických úloh modelovaných pomocí vrcholového barvení grafu.

Seznam doporučené odborné literatury:

- D.B.West: Introduction to graph theory, Upper Saddle River NJ, (2001), ISBN 0-13-014400-2.
- K.H.Rosen: Discrete Mathematics and Its Applications, New York NY, (2007), ISBN-10 0-07-288008-2.
- odborné články a texty podle pokynů vedoucího

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Mgr. Tereza Kovářová, Ph.D.**

Datum zadání: 01.09.2016

Datum odevzdání: 29.04.2017

doc. RNDr. Jiří Bouchala, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 24. dubna 2017 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 24. dubna 2017 .....

Rád bych na tomto místě poděkoval Mgr. Tereze Kovářové, Ph.D., za vedení mé bakalářské práce, rady, připomínky a studijní materiály, které mi poskytla. Rovněž bych chtěl poděkovat doc. Mgr. Petru Kovářovi, Ph.D. za připomínky a čas, který mi věnoval. Dále bych chtěl poděkovat všem, kteří mi s prací pomohli, protože bez nich by tato práce nevznikla.

Abstrakt

V této práci se zabývám vrcholovým barvením grafů a především algoritmy k nalezení vrcholového barvení. Vrcholovým dobrým m - barvením grafu rozumíme takové přiřazení m barev vrcholům, že sousední vrcholy jsou obarveny různě. O vrcholovém m - barvení grafu je známo, že je to NP - kompletní problém, pro $m \geq 3$. Tedy zatím žádný algoritmus nedokáže tento problém vyřešit obecně v polynomiálním čase (předpokládáme, že $P \neq NP$). V práci ukazuji heuristický algoritmus využívající hladové barvení i s jeho optimalizovanou verzí a uvádím příklad deterministického algoritmu. Algoritmy porovnávám a na jednoduchých příkladech ukazuji jejich princip a funkčnost.

Klíčová slova: neorientovaný graf, třídy grafu, implementace grafu, složitost, barvení grafu, vrcholové barvení grafu, chromatické číslo, heuristický algoritmus, deterministický algoritmus

Abstract

In this bachelor thesis I am dealing with vertex colorings of simple graphs with focus on vertex coloring algorithms. A proper vertex m - coloring of a graph is an assignment of m colors to the vertices so that, no two adjacent vertices share a color. The problem of the proper vertex m - coloring of a graph is well known to be an NP - complete problem for $m \geq 3$. So far, no algorithm can solve this problem generaly in polynomial time (we assume that $P \neq NP$). I present a heuristic algorithm based on the greedy coloring and it's optimized version, then example of deterministic algorithm is examined. I compare these algorithms and on simple examples show their principles and functionality.

Key Words: non-oriented graph, graph classes, graph implementation, complexity, graph coloring, vertex coloring, chromatic number, heuristic algorithm, deterministic algorithm

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Hlavní problém, kterým se budeme v této práci zabývat je vrcholové barvení grafu. Důležité je zmínit, že v práci budeme pracovat pouze s neorientovanými grafy bez násobných hran a smyček.

U vrcholového barvení grafů se zabýváme přiřazením barev vrcholům grafu tak, aby vrcholy, jenž jsou spojeny hranou, nesdílely stejnou barvu. Přitom se, ale snažíme graf vrcholově obarvit za pomocí co nejmenšího počtu barev. Tento nejmenší počet barev vyjadřuje chromatické číslo.

Grafy se dělí do několika základních tříd, které si představíme v první kapitole s názvem Základní pojmy teorie grafů. Tyto třídy mají charakteristické znaky, strukturu, díky kterým je můžeme jednoduše určit a na základě této třídy určíme i hodnotu chromatického čísla, viz. kapitola 5 - Barvení grafů.

Je známo, že určení chromatického čísla grafů je NP - těžká úloha. Stále se hledají metody jak se pro libovolný graf chromatickému číslu co nejvíce přiblížit, ideálně získat přímo hodnotu chromatického čísla. K tomu se využívají různé druhy algoritmů. Zde, ale narážíme na další problém, algoritmy mohou být přesné, ale mají vysokou složitost (v reálném čase se nedočkáme výsledku) nebo použijeme méně přesné algoritmy a výsledek budeme mít v řádu hodin. To ovšem není vše, i když je algoritmus rychlejší, tak se často potýkáme s realizačními problémy. Tím je myšleno, že algoritmy mají na základě implementace do počítače, požadavky na výpočetní prostor, více o implementaci do počítače v kapitole 3.

Ve 4. kapitole s názvem Složitost se vysvětlují pojmy se složitostí spjaté, jako například Časová složitost. Časová složitost souvisí se složitostí algoritmů a to tak, že vyjadřuje počet operací, které algoritmus provede pro vstup o určité velikosti.

Vzhledem k tomu, že barvení grafu je NP - těžká úloha a k-barvení grafu je NP - kompletní, tak se pro velké grafy využívají převážně algoritmy heuristické. V této práci se věnujeme základním heuristickým algoritmům využívajícím hladové barvení. Tyto algoritmy najdou obarvení grafu pomocí k barev, ale nezaručují, že k bude rovno chromatickému číslu grafu. Algoritmus 1 pracuje s náhodným výběrem pořadí vrcholů, přičemž zajistí, že graf bude vždy dobře vrcholově obarven a počet barev nepřesáhne určenou horní mez pro hladové barvení. Počet použitých barev však silně závisí na zvoleném pořadí vrcholů. Proto jsme dále uvedli heuristický algoritmus, který opět využívá hladové barvení, ale s jednou podstatnou změnou. Změna spočívá v optimalizovaném pořadí výběru vrcholů. Algoritmus 2 je sice složitější, zato ve většině případů dosahuje znatelně přesnějších výsledků. Oba algoritmy jsme aplikovali na jednoduché příklady, kde ukazujeme princip a funkčnost. Dále v Kapitole 6 uvádíme příklad deterministického (přesný) algoritmu, což znamená, že vždy ze stejného vstupu vytvoří stejné výsledky. Tento algoritmus je uveden v [8]. Chybí však důkaz funkčnosti algoritmu pro $k = \chi(G)$. Autor dokazuje funkčnosti obecně pro $k \leq \Delta(G) + 1$, proto vybízí k hledání grafu, pro který by jeho algoritmus nenašel vrcholové barvení za pomocí k barev, pro k rovno chromatickému číslu daného grafu.

V poslední části práce uvádíme některé praktické problémy, které se dají řešit pomocí vrcholového barvení grafů.

2 Základní pojmy teorie grafů

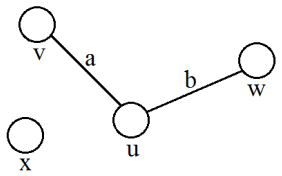
V současnosti se teorie grafů využívá při řešení mnoha praktických úloh. Její výhodou je snadná (intuitivní) představa hlavních pojmů a objektů, pomocí kterých modelujeme reálné situace. Obvykle při řešení reálného problému, množina objektů odpovídá vrcholům grafu a vztahy mezi nimi jsou modelovány hranami. Výhodou je poměrně snadná implementace objektů a postupů teorie grafů v počítači.

V této kapitole vytvoříme stručný přehled pojmů, se kterými budeme v textu dále pracovat.

2.1 Jednoduchý graf

Pojem grafu byl zaveden Leonhardem Eulerem roku 1736. Jde o model, který reprezentuje objekty (vrcholy) a vztahy (hrany) mezi nimi.

Definice 1 *Jednoduchý graf G je uspořádaná dvojice (V, E), kde V je neprázdná množina vrcholů a E je nějaká množina dvouprvkových podmnožin množiny V. Prvkům množiny E říkáme hrany.*



Obrázek 1: Jednoduchý graf G.

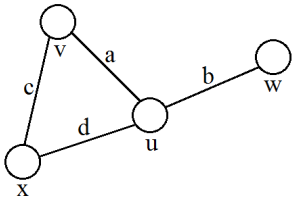
Na *Obrázku 1* je příklad jednoduchého grafu, který neobsahuje smyčky a násobné hrany. Graf může představovat mapa, na které vrcholy představují města a hrany silnice, přičemž každá z těchto silnic přímo spojuje dvě města.

Budeme pracovat s jednoduchými konečnými grafy. V takovém případě má graf $|V(G)| = n$ vrcholů a $|E(G)| = e$ hran, kde $n, e \in \mathbb{N}_0$. Vrcholy u a v, $u, v \in V(G)$, jsou koncové vrcholy hrany $e = \{u, v\}$, $e \in E(G)$. Hrany označujeme obvykle písmeny z první poloviny abecedy $(e, f, h, \ldots)$ a vrcholy písmeny z konce abecedy $(u, v, \ldots, z)$. Má-li hrana e koncové vrcholy u, v, tak místo zápisu $e = \{u, v\}$ používáme kratší zápis $e = uv$, nebo jen hrana uv. Je-li vrchol v koncovým vrcholem hrany e, můžeme psát $v \in e$ a říkáme, že vrchol v je incidentní s hranou e. Například v grafu G na *Obrázku 1* je vrchol u incidentní s hranou b.

Dva různé vrcholy u, v v grafu jsou sousední (závislé), jestliže v grafu existuje hrana uv. V opačném případě se vrcholy nazývají nesousední (nezávislé). Například v grafu G na *Obrázku 1* je vrchol w sousední s vrcholem u, vrcholy v a w jsou nesousední.

Definice 2 *Množina nezávislých vrcholů v grafu G je taková množina, ve které jsou každé dva vrcholy nezávislé.*

Například v grafu H na *Obrázku 2* - vrcholy u a x jsou sousední vrcholy vrcholu v. Hrany a, b, d jsou závislé. Vrcholy v, w tvoří množinu nezávislých vrcholů.



Obrázek 2: Jednoduchý graf H.

Podobně říkáme, že dvě hrany jsou nezávislé, jestliže nemají žádný společný koncový vrchol. Množina nezávislých hran pak obsahuje hrany, z nichž žádné dvě hrany nejsou závislé. Pokud jakékoliv dvě hrany mají společný koncový vrchol, říkáme, že jsou závislé (sousední). Například v grafu H na *Obrázku 2* jsou hrany c a b nezávislé, hrany a, b a d jsou hrany závislé.

Stupeň vrcholu v (značíme $deg(v)$) určuje počet hran incidentních s vrcholem. Největší stupeň vrcholu v grafu G se značí $\Delta(G)$, nejmenší stupeň vrcholu se značí $\delta(G)$. Pro graf H z *Obrázku 2* je $\Delta(H) = 3$ a $\delta(H) = 1$. Pro vrchol $v \in V(H)$ platí $deg(v) = 2$.

Podgraf grafu

V rámci grafu můžeme zkoumat různé podstruktury. Nejjednodušší obecnou podstrukturou je podgraf grafu. Podgraf grafu G získáme tak, že odebereme některou podmnožinu vrcholů, hran z grafu G.

Definice 3 *Mějme dán graf $G = (V, E)$. Řekneme, že graf $H = (V', E')$ je podgrafem grafu G, jestliže $V' \subseteq V$ a současně $E' \subseteq E$.*

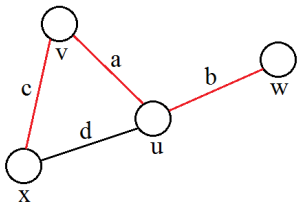
Při odebrání vrcholu je nutné odebrat zároveň i všechny hrany jdoucí z (do) tohoto vrcholu. Pokud odebereme vrchol (vrcholy) a pouze hrany s ním (s nimi) incidentní, dostaneme *indukovaný podgraf.*

2.2 Třídy grafů

Mezi jednoduchými grafy rozlišujeme některé speciální typy (struktury), které se často opakují a mají své ustálené značení a názvy. Jsou to třídy grafů, mezi které patří například:

Cesta P_n

Termínem cesta se označuje graf $P_n = (V, E)$, kde $V(P_n) = \{v_0, v_1, \ldots, v_n\}$, pro množinu hran platí $E(P_n) = \{v_1v_2, v_2v_3, \ldots, v_{n-1}v_n\}$. Je to tedy taková posloupnost vrcholů a hran, že v grafu existuje hrana z daného vrcholu do jeho následníka. Žádné dva vrcholy (ani hrany) se přitom neopakují.



Obrázek 3: Cesta P_4 jako podgraf.

Kompletní (úplný) graf K_n

Kompletní graf $K_n = (V, E)$ je takový graf, kde $V(K_n) = n$ a $E(K_n)$ obsahuje všechny možné hrany. Tj. $E(K_n) = \binom{n}{2}$.

Cyklus C_n

Cyklus je graf $C_n = (V, E)$ s vrcholovou množinou $V(C_n) = \{v_1, v_2, \ldots, v_n\}$, pro n alespoň 3 a množinou hran $E(C_n) = \{v_1v_2, v_2v_3, \ldots, v_{n-1}v_n, v_nv_1\}$.

Bipartitní graf $K_{m,n}$

Graf $K_{m,n} = (V, E)$, jehož vrcholová množina je sjednocením dvou neprázdných disjunktních množin U, W, $V(K_{m,n}) = U \cup W$ a množina hran je $E(K_{m,n}) = \{uw : u \in U \wedge w \in W\}$, se nazývá *kompletní bipartitní graf* (biklika) *s partitami U a W*. Kompletní bipartitní graf značíme $K_{m,n}$, kdy $m = |U|$, $n = |W|$.

Bipartitní graf je zobecněním kompletního bipartitního grafu. V bipartitním grafu nemusí být všechny hrany mezi partitami U a W, ale jen některé: $E \subseteq \{uw : u \in U, w \in W\}$.

Diskrétní graf D_n

Graf $G = (V, E)$ je diskrétní, pokud $E = \emptyset$. Diskrétní graf o n vrcholech je obvykle označován symbolem D_n.

k-regulární graf

Je to takový graf G, jehož všechny vrcholy mají stejný stupeň. $\exists k \in \mathbb{N}_0$ takové, že $\forall v \in V(G)$ platí $deg(v) = k$.

2.3 Další pojmy

Při barvení grafu potřebujeme další pojmy, které později využijeme.

Klikové číslo

Klikové číslo grafu je přirozené číslo udávající velikost největší kliky (úplného podgrafu) v daném grafu.

Definice 4 *Klikové číslo grafu G značíme $c(G)$. Jedná se o počet vrcholů největšího kompletního grafu (kliky), který je podgrafem G.*

Například klikové číslo úplného grafu K_n je n, klikové číslo diskrétního grafu je $D(n) = 1$.

Množina nezávislých vrcholů S

Nezávislá množina vrcholů v grafu je taková množina jeho vrcholů, že žádné dva z vrcholů této množiny nejsou spojeny hranou.

Definice 5 *Mějme graf $G = (V, E)$ a množinu $S \subseteq V(G)$, která je nezávislou množinou vrcholů, pokud platí:*

$$\forall x, y \in S : (x, y) \notin E(G).$$

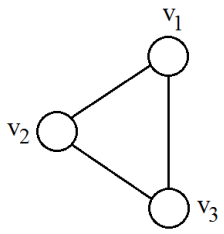
Kartézský součin grafů $G \square H$

Dále si nadefinujeme operaci na grafech, která patří mezi součiny grafů. Je to operace, která ze dvou grafů G_1 a G_2 vytvoří nový graf G.

Definice 6 *Řekneme, že graf G je kartézským součinem grafů G_1 a G_2, jestliže množina vrcholů je $V(G) = V(G_1) \times V(G_2)$. Jakékoliv dva vrcholy (u, u') a (v, v') jsou sousední právě tehdy když, vrchol u je roven vrcholu v a vrchol u' je sousední s vrcholem v' v grafu G_2 nebo když je vrchol u' roven vrcholu v' a vrcholy u, v jsou sousední vrcholy v grafu G_1. Výsledný graf G značíme $G_1 \square G_2$.*

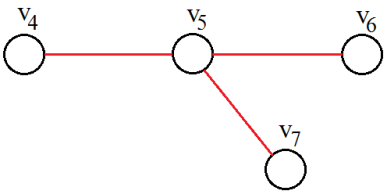
Příklad 1

Máme-li zadaný graf G_1, viz. *Obrázek 4*



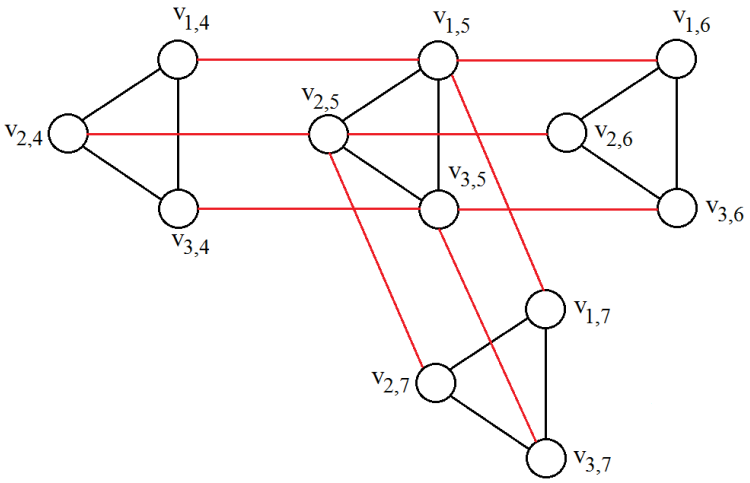
Obrázek 4: Graf G_1.

a graf G_2, viz. *Obrázek 5*,



Obrázek 5: Graf G_2.

pak kartézský součin grafů G_1 a G_2, $G_1 \square G_2$, vypadá následovně.



Obrázek 6: Kartézský součin grafů G_1 a G_2.

■

3 O implementaci grafů do počítače

Abychom mohli zkoumat grafy na počítači a řešit problémy s nimi spojené, je třeba počítači nějakým způsobem graf zadat - implementovat. Nejtypičtější implementací malých grafů do počítače je matice. V této části si ukážeme, jak sestrojit základní matice (sousednosti, incidentní) pro neorientované grafy. Rovněž zmíníme algoritmy na prohledávání grafů. Budeme vycházet především ze skript [1] Petra Kováře a publikace [3] Jakuba Černého.

Matice sousednosti $A(G)$

Matice sousednosti zachycuje, které vrcholy grafu spolu sousedí. Matice sousednosti $A(G)$, kde $G = (V, E)$, je čtvercová matice o rozměrech $n \times n$, kdy n je počet vrcholů grafu G, $|V(G)| = n$. To znamená, že počet řádků a počet sloupců odpovídá počtu vrcholů grafu G. Pro matici sousednosti tak platí $A(G) = (a_{i,j})$, kde prvek $a_{i,j} = 1$ právě tehdy, když jsou vrcholy v_i a v_j sousední. V opačném případě $a_{i,j} = 0$. Tj.

$$a_{i,j} = \begin{cases} 1 \Leftrightarrow v_i v_j \in E \\ 0 \Leftrightarrow v_i v_j \notin E \end{cases} .$$

Matice $A(G)$ je symetrická a součet prvků v i-tém řádku (i-tém sloupci) je roven stupni vrcholu v_i.

Pokud hustý graf zapíšeme pomocí matice sousednosti, tak získáme úspornější využití paměti. Než při použití incidentní matice. Matici sousednosti může modifikovat tak, že si do $a_{i,j}$ uložíme vzdálenosti vrcholu i od vrcholu j, pak dostáváme *matici vzdáleností* (metriku grafu G).

Implementace matice sousednosti do počítače se provádí přes dvourozměrné pole, $g[\,][\,]$.

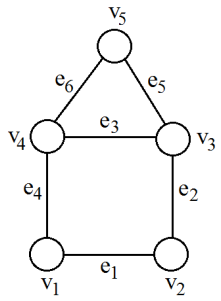
Vytvoření matice sousednosti

1) Očíslujeme si vrcholy grafu G.

2) Rozměr matice zvolíme rovem $|V(G)|$.

3) Začneme vrcholem v_1 na řádku 1, pokud daný vrchol sousedí s j-tým vrcholem, zapíšeme do j-tého sloupce 1, jinak 0. Posuneme se na další řádek (vrchol) a proces opakujeme, dokud matici sousednosti nenaplníme.

Příklad 2
Máme-li zadaný graf G, viz. *obr. 7*,



Obrázek 7: Graf G.

pak matice sousednosti vypadá následovně.

$$A(G) = \begin{pmatrix} 01010 \\ 10100 \\ 01011 \\ 10101 \\ 00110 \end{pmatrix}.$$

■

Incidentní matice $B(G)$

Incidentní matice $B(G)$, kde $G = (V, E)$, je obdélníková matice o rozměrech $(n \times m)$ s $n = |V(G)|$ řádky a $m = |E(G)|$ sloupci. Vrcholy označíme $v_1, \ldots, v_n$ a hrany $e_1, \ldots, e_m$. Pro incidentní matici $B(G) = b_{i,j}$ platí, že prvek matice $b_{i,j}$ nabývá hodnoty 1, pokud je vrchol v_i incidentní s hranou e_j, jinak $b_{i,j} = 0$. Tj.

$$b_{i,j} = \begin{cases} 1 \Leftrightarrow v_i \in e_j \\ 0 \Leftrightarrow v_i \notin e_j \end{cases} .$$

Pro jednoduché grafy dostáváme součtem čísel v j-tém sloupci vždy 2 (jednoduše pak vidíme, které 2 vrcholy hrana spojuje) a součtem čísel na i-tém řádku dostáváme stupeň vrcholu v_i.

Vzhledem k zápisu incidentní matice dostáváme řídkou a rozsáhlou matici, což způsobuje velké paměťové požadavky pro velké (husté) grafy. Incidentní matici v počítači naimplementujeme jako dvourozměrné pole, $h[\,][\,]$.

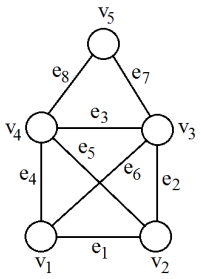
Vytvoření incidentní matice

1) Očíslujeme si vrcholy a hrany grafu G.

2) Rozměr matice zvolíme rovem $|V(G)| \times |E(G)|$.

3) Začneme vrcholem v_1 na řádku 1. Pokud daný vrchol inciduje s j-tou hranou, zapíšeme do j-tého sloupce 1, jinak 0. Posuneme se na další řádek (vrchol) a proces opakujeme, dokud incidentní matici nenaplníme.

Příklad 3
Máme-li zadaný graf H, viz *obr. 8*,



Obrázek 8: Graf H.

pak incidentní matice vypadá následovně.

$$B(H) = \begin{pmatrix} 10010100 \\ 11001000 \\ 01100110 \\ 00111001 \\ 00000011 \end{pmatrix}.$$

Všimněme si, že přidáním 2 hran oproti předešlému grafu G (*obr. 7*) dostáváme daleko řidší a větší matici. Při použití matice sousednosti bychom stále měli čtvercovou matici řádu 5. ■

Prohledávácí algoritmy

Základním stavebním kamenem většiny grafových algoritmů je nějaký způsob prohledávání grafu. Tím myslíme postupné procházení grafu po hranách od určitého počátečního vrcholu. Možných způsobů prohledávání je víc, ukážeme si základní: prohledávání do šířky a prohledávání do hloubky.

V těchto podkapitolách využijeme informací získaných z [4] *Knihy o algoritmech a datových strukturách* Martina Mareše a [5] *Populárně naučného portálu* ČVUT, fakulty biomedicínského inženýrství.

Prohledávání do šířky

Algoritmus prohledávání do šířky (BFS - breadth-first search) využívá datovou strukturu nazývanou *fronta* - prvek, který se zařadí jako první, bude jako první obsloužen. Hovoříme o tzv. FIFO principu z anglického *first in-first out*.

Algoritmus pracuje následujícím způsobem:

1) Implementujeme graf G a vybereme libovolný vrchol v, zvolíme jej za výchozí vrchol v_0.

2) Vložíme výchozí vrchol do fronty Q (Queue).

3) Vyjmeme vrchol z fronty a prozkoumáme ho.

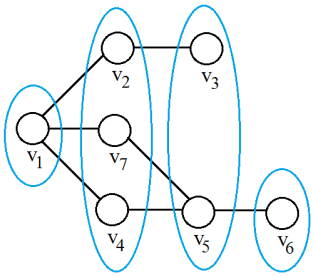
a) Pokud jsme nalezli řešení, ukončíme prohledávání a vrátíme výsledek.

b) Pokud řešení není nalezeno, zařadíme do fronty nalezené sousední vrcholy, které jsme ještě neprozkoumali.

4) Je-li fronta prázdná (všechny vrcholy jsme prozkoumali), ukončíme prohledávání a vrátíme výsledek „řešení nenalezeno“.

5) Není-li fronta prázdná, vracíme se zpět k bodu *3)*.

Proces bychom si mohli představit jako vodu, kterou „nalijeme“ do grafu a sledujeme, jak se šíří vlna, což je znázorněno na *Obrázku.9*.



Obrázek 9: Procházení grafu pomocí *prohledávání do šířky*.

Prohledávání do hloubky

Algoritmus prohledávání do hloubky (DFS - depth-first search) pracuje velice podobným způsobem, jako algoritmus prohledávání do šířky, akorát zde postupujeme rekurzivně a využíváme datovou strukturu zvanou *zásobník* - prvek, který vložíme jako první, bude obsloužen jako poslední. Hovoříme o tzv. LIFO principu z anglického *last in-first out*.

Algoritmus pracuje následujícím způsobem:

1) Implementujeme graf G a vybereme libovolný vrchol v, zvolíme jej za výchozí vrchol v_0.

2) Vložíme výchozí vrchol do zásobníku S (Stack).

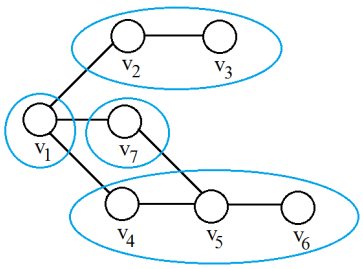
3) Vyjmeme vrchol ze zásobníku a prozkoumáme ho.

a) Pokud jsme nalezli řešení, ukončíme prohledávání a vrátíme výsledek.

b) Pokud řešení není nalezeno, zařadíme do zásobníku nalezené sousední vrcholy, které jsme ještě neprozkoumali.

4) Je-li zásobník prázdný (všechny vrcholy jsme prozkoumali), ukončíme prohledávání a vrátíme výsledek „řešení nenalezeno“.

5) Není-li zásobník prázdný, vracíme se zpět k bodu *3)*.



Obrázek 10: Procházení grafu pomocí *prohledávání do hloubky*.

Algoritmy se liší pouze použitou datovou strukturou. Při použití *fronty* - prvek, který vložíme jako poslední, bude obsloužen jako poslední. Naopak při použití *zásobníku* - prvek který vložíme jako poslední, bude obsloužen jako první. Na první pohled „zanedbatelný“ rozdíl, ale tento rozdíl způsobuje, že algoritmy fungují jinak, i když vypadají stejně.

4 Složitost

Velká-O notace

Velká-O notace se v matematice využívá více než sto let. V IT je široce využívána k analýze složitosti algoritmů a byla popularizována Donaldem Knuthem, který mimo jiné představil *velkou-*Θ *notaci* a *velkou-*Ω *notaci*. Německý matematik Paul Bachmann poprvé představil velkou-O notaci v roce 1892 v knize o teorii čísel [9]. Velká-O notace se často nazývá *Landauova notace* po Německém matematikovi Edmundovi Landau, který používal tuto notaci ve své práci.

Definice 7 ([7]) *Nechť $f(x)$ a $g(x)$ jsou dvě funkce definované na podmnožině reálných čísel. Potom tvrdíme, že*

$$f(x) \in O(g(x))$$

právě tehdy, když

$$\exists C : C \neq 0, \exists k \, \forall x > k \wedge k > 0 : |f(x)| \leq C|g(x)|,$$

kde C a k jsou konstanty.

Poznámka: *Zapsaný vztah $f(x) \in O(g(x))$ říká, že funkce $f(x)$ roste pomaleji, než nějaký násobek $g(x)$, pokud x roste neomezeně.*

Velká-Θ a velká-Ω notace

Velká-O notace se používá k popisu růstu funkcí, ale má své nevýhody. Například pokud máme funkci $f(x) \in O(x^2)$ víme, že násobek x^2 je horní mezí pro $f(x)$, platí pro větší hodnoty x. Velká-O notace nám však neposkytuje žádnou dolní mez pro $f(x)$. K určení dolní meze lze využít velkou-Ω notaci. V případě, že chceme jak horní, tak dolní mez, využíváme velkou-Θ notaci. Obě tyto notace představil Donald Knuth v 70. letech 20. století [10]. Důvodem proč byly tyto notace představeny, bylo špatné používání velké-O notace při určování dolní a horní nebo jen dolní meze.

Definice 8 ([7]) *Nechť $f(x)$ a $g(x)$ jsou dvě funkce definované na podmnožině reálných čísel. Potom tvrdíme, že*

$$f(x) \in \Omega(g(x))$$

právě tehdy, když

$$\exists C : C > 0, \exists k \, \forall x > k \wedge k > 0 : |f(x)| \geq C|g(x)|,$$

kde C a k jsou konstanty.

Poznámka: *Můžeme si všimnout vazby mezi velkou-O a velkou-Ω notací, $f(x) \in \Omega(g(x))$ pouze tehdy, když $g(x) \in O(f(x))$.*

Znalost stupně růstu funkce vyžaduje jak horní, tak dolní mez. To znamená, že pro funkci $f(x)$ potřebujeme funkci $g(x)$ takovou, že $f(x) \in O(g(x))$ a $f(x) \in \Omega(g(x))$. K tomu se využívá velká-Θ notace, která nám poskytne jak horní, tak dolní mez dané funkce.

Definice 9 ([7]) *Nechť $f(x)$ a $g(x)$ jsou dvě funkce definované na podmnožině reálných čísel. Potom tvrdíme, že*

$$f(x) \in \Theta(g(x))$$

právě tehdy, když

$$\exists C_1, C_2 > 0, \exists k \forall x > k : C_1|g(x)| \leq |f(x)| \leq C_2|g(x)|,$$

kde C_1, C_2 a k jsou konstanty.

Poznámka: *Existence konstant C_1, C_2 a k říká, že $f(x) \in \Omega(g(x)) \wedge f(x) \in O(g(x))$.*

Složitost algoritmů

Abychom mohli porovnávat různé algoritmy řešící stejný problém, zavádí se pojem složitost algoritmu. Složitost je jinak řečeno náročnost algoritmu z pohledu času, či paměti.

Časová složitost - může být vyjádřena z hlediska počtu operací, které musí algoritmus udělat pro zadaný vstup o určité velikosti. Jako operace pro měření složitosti můžeme brát například: násobení celých čísel, dělení, porovnávání, přičítání čísel nebo jakoukoliv jinou základní operaci. Časová složitost je popsána z hlediska počtu potřebných operací namísto času, který potřebuje počítač na provedení operace. Je tomu tak proto, že je poměrně složité převést na různých počítačích operace na základní bitové operace, které počítač používá. Čas k provedení operace na různých počítačích může dosahovat poměrně velkých rozdílů, například výkonný počítač provede jednoduchou bitovou operaci za 10 pikosekund, $10^{-11}s$, kdežto kancelářský počítač zvládne tu samou bitovou operaci za 10 nanosekund, $10^{-8}s$. Stejná operace by tak trvala na kancelářském počítači 1000-krát déle.

Prostorová složitost - sleduje nároky algoritmu na paměť. Jak velké paměťové místo je vyžadováno pro chod algoritmu a jak ovlivní větší paměť chod algoritmu.

Jelikož konkrétní čísla (čas, bity) se liší v závislosti na vstupních datech, množství zpracovávaných dat a na použitém programovacím jazyku, neudává se složitost čísly, nýbrž funkcí závislou na velikosti vstupních dat. Počítá se s nejhorším možným případem vstupu. To je důležité například u třídících algoritmů, kde hraje velkou roli to, jak moc už je vstupní pole setříděné (vstupuje-li do algoritmu už setříděná posloupnost čísel, algoritmus skončí okamžitě, zatímco s opačně seřazenými čísly bude algoritmus třídit dlouho), proto při počítání složitosti vkládáme do algoritmu ten nejhorší případ vstupu.

Jak snížit výpočetní složitost algoritmů? Pokud možno, zmenšením vstupních dat (řídké matice) nebo použitím jiných algoritmů schopných daný problém řešit rychleji. Pak se ale musíme často spokojit s přibližnými výsledky (možná i správnými výsledky).

Problémy, které jsou řešeny polynomiálními algoritmy patří do Třídy P. Pokud pro daný problém existuje polynomiálně omezený algoritmus, který ověří správnost řešení v polynomiálním čase, hovoříme o nedeterministicky polynomiálním problému. Tyto problémy pak tvoří Třídu NP.

NP kompletní problémy - Vrcholové barvení grafů, jak je zmíněno již v úvodu, se řadí do NP kompletních problémů. NP problémy jsou nedeterministicky polynomiální problémy, na které jsou polynomiálně redukovatelné všechny ostatní problémy z třídy *NP*. Třídu *NP - úplných* problémů tvoří *nejtěžší NP* problémy. Pokud nalezneme deterministický algoritmus, který řeší *NP - kompletní* problém, pak jsme schopni řešit všechny nedeterministicky polynomiální problémy v polynomiálním čase, což by znamenalo, že třída $P = NP$. Třída P obsahuje problémy řešitelné v polynomiálním čase. V současnosti se předpokládá nerovnost těchto tříd, tedy $P \neq NP$ (zřejmě je $P \subseteq NP$).

V roce 1972 vydal Richard Karp článek *Reducibility Among Combinatorial Problems* [11], ve kterém dokázal, že 21 výpočetních problémů patří do skupiny NP - kompletních problémů. Článek mimo jiné obsahuje důkaz o tom, že vrcholové barvení patří mezi NP - kompletní problémy.

Třída P - je obvykle považována za třídu problémů, které jsou efektivně řešitelné, přesto není v této třídě problém najít i efektivně neřešitelné problémy (se složitostí například $n^{1000000}$).

NP - úplnost definoval americký informatik Stephen Arthur Cook (* 14. prosince 1939 Buffalo, New York, USA) článkem v časopise *The Complexity of Theorem Proving Procedures* roku 1971 a v roce 2000 (přesněji 24.5.2000) byl vztah mezi P a NP zařazen *Clayovým matematickým ústavem* mezi sedm problémů tisíciletí, kdy za rozhodnutí vztahu mezi P a NP nabízí tento institut 1 milion dolarů.

5 Barvení grafů

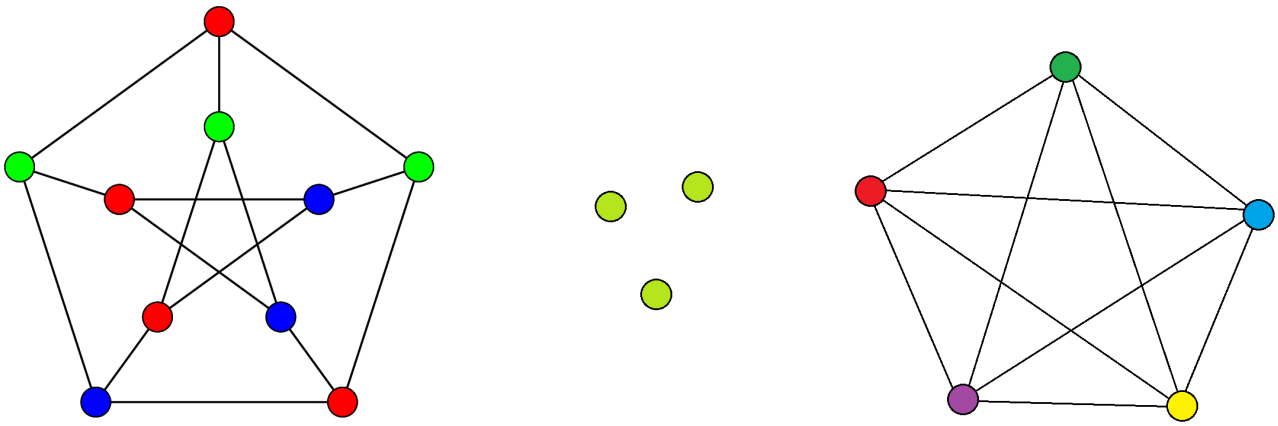
Barvení grafu je jednou z částí teorie grafů, která se zabývá přiřazením barev objektům v grafu, jako jsou například vrcholy a hrany nebo i další objekty grafu (stěny). V této části budeme vycházet především ze skript [1] Petra Kováře, odkud jsou převzaty definice hlavních pojmů a některé věty.

5.1 Vrcholové barvení grafu

Středem našeho zájmu je vrcholové barvení, které si můžeme nadefinovat následujícím způsobem.

Definice 10 *Vrcholové barvení grafu G je zobrazení c vrcholové množiny $V(G)$ do množiny barev B. Obvykle je $B = \{1, 2, \ldots, k\}$. Je-li $|B| = k$, budeme zobrazení c nazývat vrcholové k-barvení grafu G. Řekneme, že vrchol v je obarven barvou i jestliže $c(v) = i$. Číslu i říkáme barva vrcholu. Vrcholové barvení se nazývá dobré, jestliže žádné dva sousední vrcholy nejsou obarveny stejnou barvou.*

Cílem dobrého vrcholového barvení je obarvit vrcholy grafu G tak, aby sousední vrcholy neměly stejnou barvu a použitý počet barev byl co nejmenší. Stejný princip uplatňujeme i u stěnového nebo hranového barvení. Na rozdíl od hranového barvení není počet barev závislý na nejvyšším stupni vrcholu $\Delta(G)$ nebo nejnižším stupni $\delta(G)$, protože stupeň vrcholu zde neznamená nutně velký nebo malý počet barev pro obarvení grafu G.



Obrázek 11: Dobré vrcholové barvení *Petersenova grafu*, *diskrétního grafu* a *kompletního grafu*.

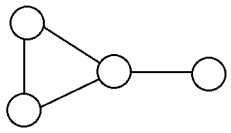
Dobré vrcholové barvení grafu G nastává, když obarvíme všechny vrcholy za pomocí k-barev a žádný z vrcholů nesdílí stejnou barvu se sousedním vrcholem.

Definice 11 *Řekneme, že graf G je vrcholově k-obarvitelný, jestliže existuje jeho dobré vrcholové barvení k barvami. Nejmenší číslo k takové, že graf G je vrcholově k-obarvitelný, se nazývá chromatické číslo grafu G a značí se $\chi(G)$. Graf G se nazývá vrcholově k-chromatický, je-li $k = \chi(G)$.*

Z této definice vyplývá, co je to chromatické číslo $\chi(G)$ - vyjadřuje kolik nejméně barev potřebujeme pro dobré vrcholové obarvení grafu G.

Další charakteristika grafu, která souvisí s dobrým vrcholovým barvením je chromatický polynom. Ten udává počet možností, jak můžeme graf G obarvit pomocí t-barev. Chromatický polynom grafu G je funkcí $P(G, t)$.

Příklad 4
Funkce $P(G,t) = t * (t-1)^2 * (t-2)$ je chromatickým polynomem grafu G z *Obrázku 12*. Kolika způsoby můžeme graf G obarvit, jestliže máme k dispozici 1,2,3, nebo 4 barvy?



Obrázek 12: Graf G s chromatickým polynomem $P(G, t)$.

a) pro $t = 1$:

$$P(G,t) = t * (t-1)^2 * (t-2) = 1 * (1-1)^2 * (1-2) = 1 * 0 * (-1) = 0$$

b) pro $t = 2$:

$$P(G,t) = t * (t-1)^2 * (t-2) = 2 * (2-1)^2 * (2-2) = 2 * 1^2 * 0 = 0$$

Z dosazení do polynomu $P(G, t)$ vyšlo pro graf G (*Obrázek 12*), že neexistuje dobré vrcholové barvení pomocí 1 nebo 2 barev. Tento výsledek byl očekáván, protože graf G obsahuje K_3 podgraf a k dobrému vrcholovému barvení jsou potřeba alespoň 3 barvy.

c) pro $t = 3$:

$$P(G,t) = t * (t-1)^2 * (t-2) = 3 * (3-1)^2 * (3-2) = 3 * 4 * 1 = 12$$

d) pro $t = 4$:

$$P(G,t) = t * (t-1)^2 * (t-2) = 4 * (4-1)^2 * (4-2) = 4 * 9 * 2 = 72$$

Oproti *a*), *b*) můžeme graf G obarvit 3 nebo 4 barvami. V případě 3 barev máme 12 možností, jak graf G dobře obarvit, se 4 barvami máme 72 možností, jak graf G dobře obarvit. ■

Příklady chromatických polynomů základních tříd grafů lze najít například v [2] *Definition and terminology - Chromatic polynomial.*

5.2 Hodnota chromatického čísla

Hlavním problémem v souvislosti s dobrým vrcholovým barvením je určení chromatického čísla daného grafu. Tato charakteristika je pro základní třídy grafů známá a její hodnoty uvedeme v následující podkapitole.

5.2.1 Určení podle třídy grafu

Diskrétní graf D_n - Jediný graf, který může být dobře obarven jednou barvou, protože neobsahuje jedinou hranu, která by spojovala vrcholy.

$$\chi(G) = 1 \Leftrightarrow |E(G)| = 0.$$

Kompletní bipartitní graf $K_{m,n}$ - Označme U a W partity kompletního bipartitního grafu $K_{m,n}$. Obarvíme-li všechny vrcholy U první barvou a všechny vrcholy W druhou barvou, pak dostaneme dobré vrcholové barvení grafu $K_{m,n}$, protože žádné dva sousední vrcholy nejsou obarveny stejnou barvou.

$$\chi(K_{m,n}) = 2.$$

Sudý cyklus C_{2n} - Označíme vrcholy cyklu $v_1, v_2, \ldots, v_{2n}$ tak, že $v_i, v_{i+1} \in E(C_{2n})$ a $v_{2n}, v_1 \in E(C_{2n})$. Na obarvení stačí dvě barvy, protože obarvíme-li všechny vrcholy s lichým indexem $i = 1, 3, \ldots, 2n-1$ první barvou a vrcholy se sudým indexem $i = 2, 4, \ldots, 2n$ barvou druhou, pak dostáváme dobře vrcholově obarvený sudý cyklus C_{2n}. Každý vrchol se sudým indexem sousedí pouze s vrcholy s indexem lichým a naopak. Potom

$$\chi(C_{2n}) = 2.$$

Lichý cyklus C_{2n+1} - Jsou zapotřebí minimálně dvě barvy, protože cyklus obsahuje alespoň jednu hranu. Dvě barvy nám ale nevystačí, protože obarvíme-li vrcholy s indexy $i = 1, 3, \ldots, 2n-1$ první barvou a vrcholy s indexem $i = 2, 4, \ldots, 2n$ barvou druhou, tak nemůžeme obarvit vrchol v_{2n+1} první barvou. Vrchol v_{2n+1} totiž sousedí s vrcholem v_1 a nesmí být obarven druhou barvou. Zároveň sousedí s vrcholem v_{2n}, takže nám nezbývá jiná možnost, než jej obarvit barvou třetí.

$$\chi(C_{2n+1}) = 3 = \Delta(C_{2n+1}) + 1.$$

Kompletní (úplný) graf K_n - Jelikož jsou každé dva vrcholy spojeny hranou, tak nemohou mít stejnou barvu. Platí

$$\chi(K_n) = n = \Delta(K_n) + 1.$$

5.2.2 Dolní a horní odhad

Pokud neznáme $\chi(G)$ zadaného grafu G, můžeme na základě jiných charakteristik grafu G odhadovat v jakých číselných mezích by se $\chi(G)$ mohlo pohybovat.

Obecně pro graf G - V závislosti na počtu vrcholů grafu je možno provést následující nejhrubší odhad mezí pro $\chi(G)$. Pro graf G s n vrcholy, $|V(G)| = n$, zřejmě platí

$$1 \leq \chi(G) \leq n.$$

Graf G s podgrafem K_k - Pokud graf G obsahuje kompletní podgraf K_k, pak potřebujeme alespoň k-barev na dobré obarvení podgrafu K_k a tedy i grafu G. Potom zřejmě

$$\chi(G) \geq c(G).$$

kde $c(G)$ je klikové číslo G.

Hladové barvení - Pro graf G s největším stupněm vrcholu $\Delta(G)$, platí následující věta.

Věta 1 *Každý graf může být dobře obarven $\Delta(G) + 1$ barvami.*

$$\chi(G) \leq \Delta(G) + 1.$$

Tuto větu si uvedeme i s důkazem později v kapitole o heuristických algoritmech 6.1.1. Důkaz je konstruktivní a obsahuje "hladový"algoritmus k nalezení dobrého barvení grafu.

Pro velké grafy - Tento horní odhad se vztahuje pouze na grafy s velkým počtem vrcholů a to tak, že pokud graf G má n vrcholů a m hran. Potom platí

$$\chi(G) \leq 1 + \sqrt{2m * (n-1)/n}.$$

Věta 2 *[Brooksova věta] Nechť G je souvislý graf různý od kompletního grafu K_n takový, že $\Delta(G) \geq 3$. Potom platí*

$$\chi(G) \leq \Delta(G).$$

Tato věta, kterou dokázal Rowland Leonard Brooks (6.2.1916-18.6.1993) roku 1941, ukazuje vztah mezi chromatickým číslem a největším stupněm vrcholu grafu G. Přesněji řečeno, udává nejnižší horní mez pro chromatické číslo, obecně pro všechny grafy, až na dvě výjimky. Výjimky jsou liché cykly a kompletní grafy, které mají chromatické číslo o 1 vyšší než nejvyšší stupeň vrcholu v grafu. *Důkaz viz. např.[1] - Věta 8.7.*

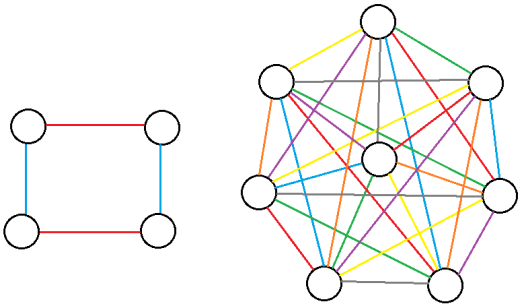
5.3 Jiné druhy barvení

Existují i jiné druhy barvení grafů, o kterých se v této podkapitole zmíníme.

5.3.1 Hranové barvení grafu

Obarvíme graf G tak, že žádná ze sousedních hran nesdílí stejnou barvu. Přesněji, každá hrana incidentní s vrcholem v je obarvena jinou barvou.

Definice 12 *Hranové barvení grafu G je zobrazení c hranové množiny $E(G)$ do množiny barev B. Obvykle je $B = \{1, 2, \ldots, k\}$. Je-li $|B| = k$, nazývá se zobrazení c hranové k-barvení grafu G. Řekneme, že hrana e je obarvena barvou i, jestliže $c(e) = i$. Číslu i říkáme barva hrany. Hranové barvení se nazývá dobré, jestliže žádné dvě závislé hrany nejsou obarveny stejnou barvou.*



Obrázek 13: Dobré hranové barvení *sudého cyklu* C_4 a *kompletního grafu* K_8.

Z ukázek jde vidět, že počet barev souvisí se stupni vrcholů (čím větší stupeň, tím více barev potřebujeme). Při řešení problému dobrého hranového barvení pro daný graf, hledáme optimální řešení s nejmenším počtem barev.

Definice 13 *Řekneme, že graf G je hranově k-obarvitelný, jestliže existuje jeho dobré hranové barvení k barvami. Nejmenší číslo k takové, že graf G je hranově k-obarvitelný, se nazývá chromatický index grafu G a značí se $\chi'(G)$. Graf G se nazývá hranově k-chromatický, je-li $k = \chi'(G)$.*

Pak by tato „souvislost se stupni vrcholů“ mohla vypadat následovně: $\chi'(G) \geq \Delta(G)$. Přesnější výsledek je obsahem *Vizingovy věty*, která tvrdí, že pokud nestačí $\Delta(G)$ barev, tak určitě bude stačit $\Delta(G) + 1$ barev.

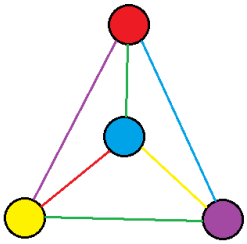
Věta 3 (Vizingova věta) *Pro libovolný graf G platí $\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1$.*

Důkaz viz.[1] - Věta 7.1.

5.3.2 Kompletní barvení grafu

Jedná se o kombinaci hranového a vrcholového barvení. Barví se vrcholy grafu G tak, aby vrchol nesdílel stejnou barvu s žádným sousedním vrcholem, ani s žádnou přilehlou hranou. Chromatické číslo $\chi''(G)$ je v tomto případě přinejmenším stejné jako chromatické číslo kompletního grafu. Platí

$$\chi''(G) \geq \Delta(G) + 1.$$



Obrázek 14: Kompletně obarvený graf G.

6 Algoritmy k nalezení dobrého vrcholového barvení

V této kapitole si představíme algoritmy k nalezení dobrého vrcholového barvení jednoduchého grafu G m barvami.

6.1 Heuristika

Volně řečeno, heuristika znamená zkusmé řešení problému založené na poučeném odhadu, intuici, zkušenostech či zdravém rozumu. Nezaručuje nalezení nejlepšího řešení, ale většinou je jednoduchá(univerzální) a rychle použitelná. V mnoha případech je schopna najít i optimální řešení, aniž by prošla všechny možnosti. Například metoda „pokus a omyl.“

6.1.1 Heuristické algoritmy

Heuristické algoritmy neposkytují matematickou kvalitu řešení, protože nevíme, kdy heuristika uspěje. Používají se nejčastěji jako metoda rychle poskytující většinou dostatečně přesné řešení. Nelze se spolehnout, že algoritmus obecně najde vždy optimální řešení. Heuristické algoritmy se používají, když není možné použít jiný lepší algoritmus. Lepší algoritmus ve smyslu, aby v reálném čase poskytl obecně přesné řešení ve všech případech řešeného problému.

Jednoduchý a klasický způsob, jak získat přijatelně dobré vrcholové barvení, je pomocí „hladových“ heuristik. Hladová barvící heuristika je založena na tzv. „step by step“ postupu, kdy v každém kroku přiřadíme vrcholu, který je na řadě barvu. Barva je vrcholu přiřazena tak, aby nesdílel stejnou barvu se sousedními vrcholy, kterým už byla nějaká barva přiřazena. U hladových heuristik je pořadí, ve kterém se vrcholy obarvují, stanoveno předem náhodně nebo podle určitého kritéria.

Dále si uvedeme dva příklady velmi jednoduchých heuristických algoritmů a srovnáme jejich funkčnost na příkladech.

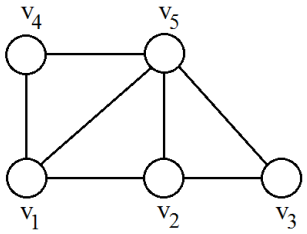
Algoritmus 1 - využívající hladové barvení

Popíšeme algoritmus dobrého vrcholového m-barvení libovolného grafu G, kde $m \leq \Delta(G) + 1$. Na začátku máme graf G a jeho libovolný vrchol obarvíme některou z $\Delta(G) + 1$ barev. Protože každý vrchol je sousední s nejvýše $\Delta(G)$ vrcholy, tak každý neobarvený vrchol v je sousední s vrcholy nejvýše $\Delta(G)$ různých barev. Vždy najdeme alespoň jednu barvu, která se na vrcholech v okolí vrcholu v nevyskytuje a tu můžeme použít na obarvení vrcholu v. Po $|V(G)|$ krocích máme graf G dobře vrcholově obarvený nejvýše $\Delta(G) + 1$ barvami.

Pro některé grafy, například pro liché cykly a kompletní grafy, dostaneme tímto jednoduchým hladovým algoritmem optimální vrcholové barvení. V obecném případě však uvedený algoritmus nemusí dát barvení s nejmenším možným počtem barev, což si ukážeme na následujícím *Příkladě 5.*

Příklad 5

Podle *Věty 1* (5.2.2 - *hladové barvení*) najdeme a sestrojíme vrcholové barvení pro graf H, *Obrázek 15.*



Obrázek 15: Graf H.

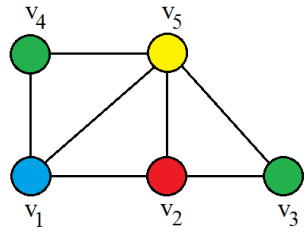
Z *věty 1* zjistíme odhad chromatického čísla

$$\chi(H) \leq \Delta(H) + 1$$

$$\chi(H) \leq 4 + 1$$

$$\chi(H) \leq 5$$

Podle vzorce budeme potřebovat nejvýše 5 barev. Náhodně si zvolíme počáteční vrchol, který obarvíme *barvou 1*, například vrchol v_4. Náhodně vybereme další vrchol, třeba v_3, jelikož tento vrchol nesousedí s vrcholy, které by byly obarveny, obarvíme jej *barvou 1.* Jako další vybereme vrchol v_1, ten sousedí s vrcholem v_4, proto musí být obarvený jinou barvou a to *barvou 2.* Dalším vrcholem, který obarvíme, bude vrchol v_2. Vrchol v_2 sousedí s již obarvenými vrcholy v_1 a v_3, obarvíme jej tedy *barvou 3.* Posledním vrcholem, který jsme neobarvili je vrchol v_5, ten sousedí s vrcholy v_1, v_2, v_3 a v_4, musíme zvolit jednu z doposud nepoužitých barev, například *barvu 4.* Výsledek můžeme vidět na *Obrázku 16.*



Obrázek 16: Graf H obarvený Algoritmem 1.

Tento algoritmus našel dobré barvení grafu H 4 barvami, ale neposkytl nám optimální barvení. Jaké je optimální řešení? Graf H obsahuje K_3 podgraf, takže by měl být obarvitelný 3 nebo více barvami. U následujícího algoritmu si ukážeme, že je graf H obarvitelný právě 3 barvami.

■

Algoritmus 2 - využívající hladové barvení s optimalizovaným výběrem pořadí vrcholů

Tento heuristický algoritmus využívá hladové barvení s optimalizovaným pořadím výběru vrcholů podle jejich stupně. Algoritmus je obsažen v důkazu následující *Věty 4*, proto větu uvedeme i s důkazem a na příkladech ukážeme funkčnost algoritmu. Algoritmus 2 je téměř stejný jako Algoritmus 1. Rozdíl spočívá v uspořádání vrcholů na začátku a procházení grafu dle tohoto pořadí vrcholů.

Věta 4 ([1]) *Pro libovolný graf G na n vrcholech s nerostoucí stupňovou posloupností*

$d_1 \geq d_2 \geq \ldots \geq d_n$ *platí*

$$\chi(G) \leq 1 + \max_{i=1,2,\ldots,n} \{min\{d_i, i-1\}\} \leq \Delta(G) + 1, \tag{1}$$

kde d_i je stupeň vrcholu a i je index vrcholu.

Důkaz Vrcholy obarvujeme v pořadí dle jejich stupně d_i, tj. od vrcholu s největším stupněm. Vrchol stupně d_i označíme v_i. Nejprve obarvíme vrchol v_1 stupně d_1 barvou 1. V obecném kroku pak obarvíme vrchol v_i, který je sousední s nejvýše $min\{d_i, i-1\}$ již obarvenými vrcholy a stačí proto vybrat některou nepoužitou barvu mezi barvami $1, 2, \ldots, 1 + min\{d_i, i-1\}$. Tak můžeme obarvit libovolný vrchol a proto stačí nejvýše $1 + \max_{i=1,2,\ldots,n}\{min\{d_i, i-1\}\}$ barev. ∎

Poznámka: Zřejmě tento heuristický algoritmus zaručuje nalezení dobrého vrcholového barvení, kde m je menší nebo rovno nejmenší horní známý odhad (Brooksova věta), $m \leq max\{min\{d_i, i-1\}\} \leq \Delta(G)$ kromě K_n a C_{2n+1}.

Funkčnost si ukážeme na příkladu barvení kompletního bipartitního grafu $K_{3,3}$. Dále si ukážeme na grafu H, z *Příkladu* 5, že Algoritmus 2 najde pro barvení vrcholů při jejich řazení dle stupně optimální řešení. Nakonec si na jiném grafu G ukážeme, že nám Algoritmus 2 (ani při seřazení vrcholů dle stupňů) neposkytne optimální řešení, tj. nalezené dobré vrcholové barvení m barvami bude pro $m > \chi(G)$.

Příklad 6

Použijeme Algoritmus 2 k nalezení dobrého vrcholového barvení kompletního bipartitního grafu $K_{3,3}$. Stupňová posloupnost je (3, 3, 3, 3, 3, 3). Dle Algoritmu 2 obsaženém v důkazu *Věty 4* sestrojíme dobré vrcholové barvení.

Využijeme vzorec 1 a zjistíme tak odhad pro chromatické číslo.

$$\chi(K_{3,3}) \leq 1 + \max_{i=1,2,3}\{min\{d_i, i-1\}\}.$$

1)Vypíšeme minima pro jednotlivé vrcholy:

$$v_1 : min\{3, 0\} = 0,$$

$$v_2 : min\{3, 1\} = 1,$$

$$v_3 : min\{3, 2\} = 2,$$

$$v_4 : min\{3, 3\} = 3,$$

$$v_5 : min\{3, 4\} = 3,$$

$$v_6 : min\{3, 5\} = 3.$$

2)Vybereme maximum z těchto minim:

$$\max_{i=1,\ldots,6}\{min\{d_i, i-1\}\} = 3.$$

3)Tedy pro odhad chromatického čísla dostáváme:

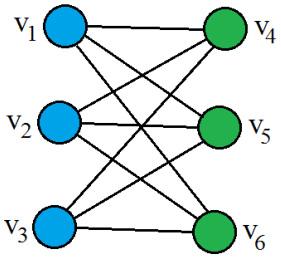
$$\chi(K_{3,3}) \leq 1 + 3,$$

$$\chi(K_{3,3}) \leq 4.$$

Podle vzorce 1 budeme potřebovat nejvýše 4 barvy. Všechny vrcholy mají stejný stupeň, proto je jedno, vzhledem k Algoritmu 2, v jakém pořadí budeme vrcholy grafu $K_{3,3}$ obarvovat. Jako počáteční zvolíme například v_1 a obarvíme ho *barvou 1.* Dále budeme obarvovat vrcholy v tomto pořadí (v_1, v_5, v_2, v_6, v_3, v_4). Vrchol v_5 obarvíme *barvou 2* a přesuneme se do dalšího vrcholu (v_2). Vrchol v_2 nesousedí s vrcholem v_1, takže ho můžeme obarvit *barvou 1.* Opět se přesuneme do dalšího vrcholu (v_6), tento vrchol nesmí být obarvený *barvou 1*, protože sousedí s vrcholy v_1 a v_2, které již jsou obarveny *barvou 1*, použijeme *barvu 2.* Z vrcholu v_6 pokračujeme vrcholem v_3, ten nesmí být obarven *barvou 2*, ale jelikož není sousední s vrcholy v_1 a v_2, můžeme ho obarvit *barvou 1.* Zbývá poslední neobarvený vrchol - v_4, který sousedí s vrcholy v_1, v_2 a v_3,

proto ho obarvíme *barvou 2* a získáváme tak dobře obarvený kompletní bipartitní graf pomocí 2 barev (*Obrázek 17*).

Víme, že na dobré vrcholové barvení $K_{3,3}$ grafu stačí pouze 2 barvy, viz. chromatická čísla pro známé třídy grafů, kapitola 5.2.1. Horní mez pro $\chi(K_{3,3})$ daná Algoritmem 2 neodpovídá optimálnímu barvení, nicméně pomocí postupu daného pro Algoritmus 2 dosáhneme při takto zvoleném pořadí vrcholů optimálního řešení.



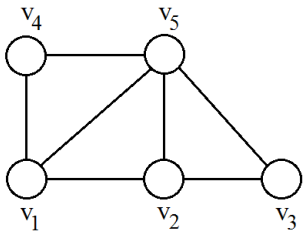
Obrázek 17: *Kompletní bipartitní graf* $K_{3,3}$ obarvený Algoritmem 2.

■

Nyní si otestujeme algoritmus na grafu H z *Příkladu* 5.

Příklad 7

Opět použijeme Algoritmus 2 k nalezení dobrého vrcholového barvení pro graf H, *Obrázek 18*. Dle Algoritmu 2 obsaženého v důkazu *Věty 4* sestrojíme dobré vrcholové barvení.



Obrázek 18: Graf H.

Využijeme vzorec 1 a zjistíme tak odhad pro chromatické číslo.

$$\chi(H) \leq 1 + \max_{i=1,\ldots,5}\{min\{d_i, i-1\}\}.$$

1)Vypíšeme minima pro jednotlivé vrcholy:

$$v_1 : min\{3, 0\} = 0,$$

$$v_2 : min\{3, 1\} = 1,$$

$$v_3 : min\{2, 2\} = 2,$$

$$v_4 : min\{2, 3\} = 2,$$

$$v_5 : min\{4, 4\} = 4.$$

2)Vybereme maximum z těchto minim:

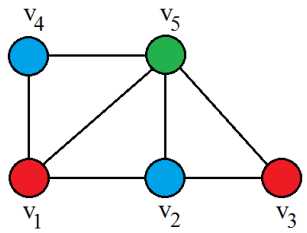
$$\max_{i=1,\dots,5}\{min\{d_i, i-1\}\} = 4.$$

3)Tedy pro odhad chromatického čísla dostáváme:

$$\chi(H) \leq 1 + 4,$$

$$\chi(H) \leq 5.$$

Podle vzorce budeme potřebovat nejvýše 5 barev. Vybereme vrchol s největším stupněm, tedy vrchol v_5 a obarvíme ho *barvou 1*. Vybereme další vrchol se stejným nebo nižším stupněm, například vrchol v_2. Vrcholy v_5 a v_2 jsou sousední, tak obarvíme vrchol v_2 *barvou 2*. Opětovně vybereme vrchol se stejným nebo nižším stupněm, vrchol v_1. Jelikož vrchol v_1 sousedí s vrcholy obarvenými *barvou 1* a *2*, musí být vrchol v_1 obarven *barvou 3*. Zbývá nám tak obarvit vrcholy v_3 a v_4, oba mají stupeň 2. Vrchol v_3 sousedí s obarvenými vrcholy v_5 a v_2 (*barvy 1* a *2*), obarvíme ho proto *barvou 3*. Posledním vrcholem je v_4 sousedící s vrcholy obarvenými *barvou 1* a *3* a obarvíme ho *barvou 2*. Dostáváme tak dobře obarvený graf H pomocí 3 barev (*Obrázek 19*). Výsledek barvení je optimální. Podgraf H, $\chi(H) = 3$.



Obrázek 19: Dobře obarvený graf H Algoritmem 2.

Přestože Algoritmus 2 pracuje téměř stejným způsobem jako Algoritmus 1 využívající hladové barvení v *Příkladu* 5, tak zprvu se mohlo zdát, že dosáhneme stejných výsledků. Ovšem díky tomu, že vrcholy seřadíme a začneme obarvovat od největšího stupně vrcholu po nejmenší,

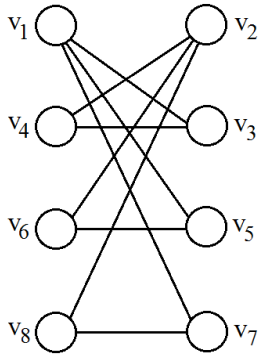
dosáhneme lepších výsledků, než u náhodného výběru vrcholů. Samozřejmě tohle pravidlo nemusí platit pro všechny grafy.

■

V následujícím příkladě nebude horní mez odpovídat optimálnímu barvení. Oproti *Příkladu 6* nedosáhneme v tomto příkladě pomocí postupu Algoritmu 2 optimálního řešení.

Příklad 8

Dle Algoritmu 2 obsaženého v důkazu *Věty 4* sestrojíme dobré vrcholové barvení pro graf G, *Obrázek 20*.



Obrázek 20: Graf G.

Využijeme vzorec 1 a zjistíme tak odhad pro chromatické číslo.

$$\chi(G) \leq 1 + \max_{i=1,\ldots,8} \{min\{d_i, i-1\}\}.$$

1)Vypíšeme minima pro jednotlivé vrcholy:

$$v_1 : min\{3, 0\} = 0,$$

$$v_2 : min\{3, 1\} = 1,$$

$$v_3 : min\{2, 2\} = 2,$$

$$v_4 : min\{2, 3\} = 2,$$

$$v_5 : min\{2, 4\} = 2,$$

$$v_6 : min\{2, 5\} = 2,$$

$$v_7 : min\{2, 6\} = 2,$$

$$v_8 : min\{2, 7\} = 2.$$

2)Vybereme maximum z těchto minim:

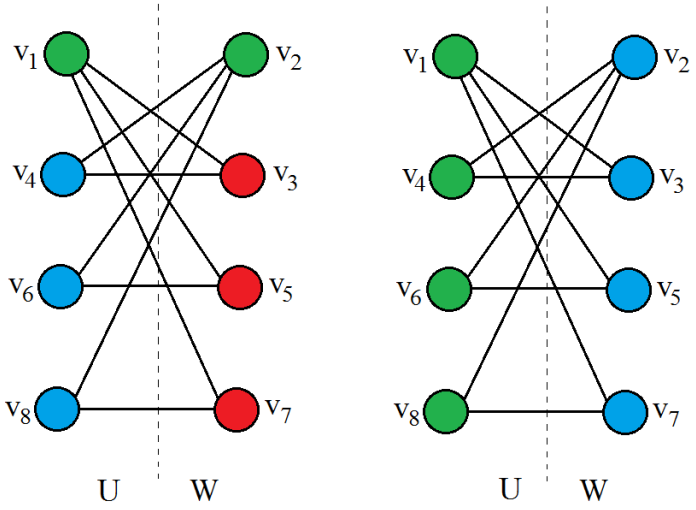
$$\max_{i=1,\dots,8}\{min\{d_i, i-1\}\} = 2.$$

3)Tedy pro odhad chromatického čísla dostáváme:

$$\chi(G) \leq 1 + 2,$$

$$\chi(G) \leq 3.$$

Podle vzorce budeme potřebovat nejvýše 3 barvy. Postupujeme podobně, jako v předchozích příkladech. Vybereme vrchol s největším stupněm například v_1 a obarvíme ho *barvou 1.* Následně vybereme další vrchol se stejným nebo nižším stupněm, tedy vrchol v_2. Jelikož vrcholy v_1 a v_2 nejsou závislé, obarvíme vrchol taktéž *barvou 1.* Zde ovšem nastává problém, protože se jedná o bipartitní graf a my jsme právě obarvili na každé partitě (U,W) jeden vrchol stejnou barvou. Zbývající vrcholy partity U (v_4, v_6, v_8) obarvíme *barvou 2*, protože jsou závislé s vrcholem v_2, ovšem vrcholy v_3, v_5, v_7 nemůžeme obarvit *barvou 1* ani *barvou 2*, protože jsou sousední s vrcholy, které již tuto barvu mají, musíme je tedy obarvit *barvou 3.* Dostáváme tak dobře obarvený bipartitní graf 3 barvami (*Obrázek 21*), ale z kapitoly 5.2 - *Kompletní bipartitní graf* víme, že jsme nepoužili optimální počet barev, což jsou 2 barvy. V tomto případě lze vidět, že výsledek získaný Algoritmem 2 nemusí vždy odpovídat optimálnímu barvení. Otázkou je, jaké je optimální pořadí vrcholů vzhledem k hladovému barvení?



Obrázek 21: *Bipartitní graf G* dobře obarvený podle *heuristického Algoritmu 2* a *bipartitní graf G* obarvený minimálním počtem barev.

■

Grundyho číslo

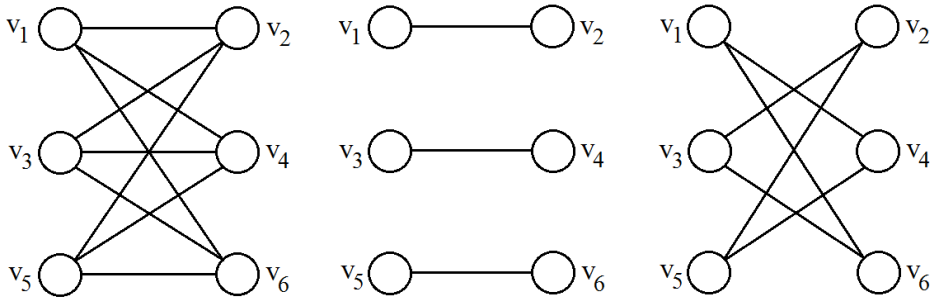
Existují grafy takové, že s vysokou pravděpodobností při náhodném výběru pořadí vrcholů povede heuristický Algoritmus 1 k mnohem většímu počtu barev, než je minimální počet barev nutný pro dobré obarvení daného grafu.

Obzvláště špatné výsledky hladového barvení můžeme dostat pro Crown graf. Crown graf vytvoříme obecně z grafu $K_{n,n}$ tak, že odebereme úplné párování, viz. *Příklad* 9. Vybereme-li pořadí vrcholů tak, aby vždy dva po sobě následující vrcholy tvořily hranu odebíraného párování, pak pomocí hladového barvení budeme potřebovat n barev, přičemž optimální počet barev je 2. Proto je u hladových barvení kladen důraz na vhodné seřazení vrcholů.

Počet barev získaný pomocí hladového barvení pro nejhorší možné uspořádání vrcholů grafu se nazývá *Grundyho číslo* [13], které roku 1939 představil Patrick Michael Grundy pro orientované grafy. Grundyho číslo pro neorientované grafy bylo představeno až roku 1979 Christenem a Selkowem. V tomto smyslu, najít nejhorší možné seřazení vrcholů grafu G, je NP - úplný problém.

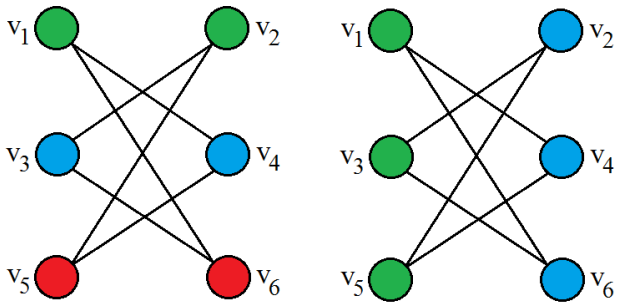
Příklad 9

V tomto příkladě si ukážeme, jak získat z $K_{3,3}$ tzv. Crown graf. Seřadíme vrcholy podle odebraného párování, pak graf obarvíme pomocí hladového barvení (Algoritmus 1). Jaká je hodnota Grundiho čísla?



Obrázek 22: Crown graf, který vznikne odebráním úplného párování z grafu $K_{3,3}$.

Vrcholy Crown grafu (*Obrázek 22*) seřadíme následovně: (v_1, v_2, v_3, v_4, v_5, v_6). Dle Algoritmu 1 sestrojíme dobré vrcholové barvení. Vezmeme vrchol v_1 a obarvíme ho *barvou 1*. Vrchol v_2 nesousedí s vrcholem v_1, proto ho také obarvíme *barvou 1*. Následuje vrchol v_3, ten ale sousedí s vrcholem v_2, který je obarven *barvou 1* - v_3 proto musí být obarven *barvou 2*. Vrchol v_4 také sousedí s vrcholem, který je obarvený *barvou 1* (v_1), avšak nesousedí s vrcholem v_3, takže ho obarvíme *barvou 2*. Předposledním vrcholem je vrchol v_5, jenž sousedí s vrcholy v_2, v_4 (barvy *1* a *2*), obarvíme ho *barvou 3*. U posledního vrcholu se dostáváme do stejné situace, v_6 sousedí s vrcholy v_1, v_3, které již mají barvu *1* a *2*, v_6 tak obarvíme *barvou 3*. Tímto máme dobře obarvený Crown graf pomocí 3 barev. Poté co z $K_{3,3}$ odebereme úplné párování, dostáváme sudý cyklus C_6 (zároveň můžeme říct, že výsledný graf je bipartitní), o kterém víme, že jeho chromatické číslo je $\chi(C_6) = 2$. Výsledek můžeme vidět na *Obrázku 23*.



Obrázek 23: Crown graf obarvený Algoritmem 1 a optimálně obarvený Crown graf.

Je zřejmé, že pro Algoritmus 1 nevymyslíme horší seřazení vrcholů C_6, takže počet barev potřebných na dobré vrcholové barvení je zároveň Grundyho číslem. Grundyho číslo pro Crown graf na 6 vrcholech je tedy 3. ■

Složitost Algoritmu 1 a Algoritmu 2

U Algoritmu 1 stačí projít všechny vrcholy grafu a každý vrchol porovnat s jeho sousedy (maximálně n sousedů), tedy výsledná složitost Algoritmu 1 je $O(n^2)$. Naopak u Algoritmu 2 musíme ještě na začátku seřadit vrcholy dle jejich stupně, složitost tedy závisí na použitém třídícím algoritmu. Složitost třídění však nebývá větší než $O(n^2)$, proto složitost celého Algoritmu 2 také nebude řádově vyšší než $O(n^2)$.

Pro grafy se stovkami vrcholů nebude na první pohled jasné, který z algoritmů je rychlejší. Pokud ale použijeme algoritmy na grafy o desetitisících vrcholech, tak bude rozdíl mezi Algoritmem 1 a Algoritmem 2 výraznější (silně záleží na tom, jaký použijeme třídící algoritmus u Algoritmu 2), Algoritmus 1 bude rychlejší. Nesmíme však zapomínat, přestože je Algoritmus 1 rychlejší, tak nemusíme dosáhnout optimálního počtu barev vzhledem k náhodně zvolenému pořadí vrcholů.

Aktuální heuristiky

Hladové heuristiky jsou obvykle rychlé (polynomiální složitost), ale mají tendenci obarvit graf více barvami, než kolik udává chromatické číslo daného grafu. Lepších výsledků dosáhneme použitím účinějších heuristik jako jsou například [14]:

• „Local search“ heuristiky - „local search“ je klíčovou složkou pro heuristická barvení grafů. Nejefektivnější doposud vytvořené barvící heuristiky využívají „local search“ přístup. Tento přístup však není dostatečně účinný pro grafy s 500 a více vrcholy.

• Genetický algoritmus - jedná se o heuristický postup založený na evolučních algoritmech. Genetický algoritmus se snaží pomocí evolučních algoritmů optimalizovat obtížné grafy, pro které neexistuje přesný algoritmus. Tyto algoritmy vypadají, jako slibná volba pro budoucí výzkumy.

• Současné metody pro barvení grafů jsou založeny na extrakci nezávislé množiny a progresivním barvení. Tyto metody dosáhly pozoruhodného výkonu na velmi těžkých a velkých grafech s více než 2000 vrcholy. Podařilo se zlepšit poměrně nedávno určené horní hranice chromatického čísla pro několik velkých srovnávacích („benchmark“) grafů.

Další informace o současných heuristikách nalezneme v publikaci Philippa Galiniera [14].

V další podkapitole také budeme využívat nezávislou množinu vrcholů k nalezení dobrého barvení grafu, nicméně půjde o zcela odlišný přístup. Především prezentovaný algoritmus nebude heuristický, ale deterministický.

6.2 Deterministický algoritmus

Deterministický algoritmus je v informatice označení pro algoritmus, který vždy ze stejných výchozích podmínek svým během vytvoří stejné výsledky. Každý krok v algoritmu je vždy jednoznačně definován, což je rozdíl oproti nedeterministickým algoritmům, kde následující krok nemusí být vždy jednoznačně určen.

Následující algoritmus se dá zařadit mezi deterministické algoritmy. Je složitější než třeba výše uvedené *heuristické algoritmy*, avšak výstupem je přesné řešení daného problému.

Algoritmus 3 - využívající kartézský součin grafů

Tento algoritmus je schopný v polynomiálním čase hledat dobré vrcholové barvení pomocí m barev. Tj. pro dané m a daný graf G by měl algoritmus najít dobré vrcholové barvení m barvami, nebo zjistit, že takové barvení neexistuje. Bylo dokázáno, že každý graf s n vrcholy a maximálním stupněm vrcholu $\Delta(G)$ musí mít $\chi(G) \leq \Delta(G) + 1$, algoritmus najde vždy dobré vrcholové m-barvení grafu G s $m \leq \Delta(G) + 1$. Autor tohoto algoritmu, Ashay Dharwadker [8], dokázal, že ve speciálních případech, kdy graf G je jednoduchým grafem a zároveň graf není kompletním grafem nebo cyklem liché délky, tak je algoritmus schopný najít dobré vrcholové barvení grafu G za pomocí m barev, kdy $m \leq \Delta(G)$. Během tohoto důkazu správnosti algoritmu tak získal autor nový konstruktivní důkaz pro Brooksovu větu (2) z roku 1941.

Pro všechny testované příklady grafů našel tento algoritmus dobré vrcholové m-barvení grafu G, kdy $m = \chi(G)$. Pokud by neexistoval graf, pro který by algoritmus nenalezl barvení $\chi(G)$ barvami, pak **třída NP = třídě P**. Z důvodu velkého významu problému zda $NP = P$, vybízí autor k hledání příkladu grafu, pro který by algoritmus dobré vrcholové barvení $\chi(G)$ barvami nenalezl.

Než uvedeme samotný algoritmus, potřebujeme dokázat tvrzení kartézského lemma dávající do souvislosti nezávislou množinu vrcholů kartézského součinu $G\Box K_m$ s existencí dobrého m-barvení grafu G. Z důkazu kartézského lemma pak bude zřejmé, jak sestrojit obarvení grafu G, máme-li k dispozici množinu nezávislých vrcholů $S \subseteq G\Box K_m$.

Lemma 6.1 (Kartézské lemma [8]) *Jednoduchý graf G s n vrcholy je m-obarvitelný právě tehdy, když kartézský součin $G\Box K_m$ má nezávislou množinu vrcholů o velikosti n.*

Důkaz pro kartézské lemma převezmeme od autora článku *The Vertex Coloring Algorithm*, Ashay Dharwadkera [8].

Důkaz Předpokládáme, že je graf G dobře obarven m barvami. Definujeme si podmnožinu S vrcholů z kartézského součinu $G\Box K_m$ následovně: vrchol $(u; v)$ z $G\Box K_m$ náleží podmnožině S právě tehdy, když k vrcholu u z grafu G je přiřazena barva v s ohledem na dobré m-barvení. Protože každý vrchol z grafu G má přiřazenou nějakou jednu barvu, pak $|S| = n$. Měli bychom dokázat, že S je nezávislá množina. Vezmeme $(u_1; v_1)$ a $(u_2; v_2)$ z S a předpokládáme, že existuje

hrana $\{(u_1; v_1), (u_2; v_2)\}$ v $G\square K_m$. Pak z definice kartézského součinu mohou nastat tyto dvě možnosti:

• $u_1 = u_2$ a $\{v_1; v_2\}$ je hranou v K_m. Z definice S, $u_1 = u_2$ implikuje, že $v_1 = v_2$, ale pak by $\{v_1; v_1\}$ musela být hranou v grafu K_m, tj. smyčka u vrcholu v_1. To je proti definici jednoduchého grafu - dochází ke sporu.

• $\{u_1; u_2\}$ je hranou v grafu G a $v_1 = v_2$. Tohle tvrzení popírá definici dobrého m-barvení grafu G, protože by to znamenalo, že jsme obarvili 2 sousední vrcholy v grafu G stejnou barvou (opět nastává spor).

Protože nemůže být hrana mezi dvěmi vrcholy z S, pak S musí být nezávislá množina vrcholů.

Naopak předpokládejme, že v kartézském součinu $G\square K_m$ máme nezávislou množina vrcholů S o velikosti n. Ukážeme, že graf G je dobře m-obarvitelný. Když m je **větší nebo rovno** n, pak graf G může být triviálně m-obarven, takže předpokládejme, že m je **menší** než n. Rozdělíme vrcholy množiny S do nejvýše m tříd ekvivalence $C_1, ..., C_m$, kde vrchol (u, v) z množiny S patří do třídy C_i právě tehdy, když $v = v_i$. Zřejmě třídy $C_1, C_2, \ldots, C_m$ tvoří rozklad nezávislé množiny S.

Vrcholy grafu G rozdělíme do nejvýše m tříd $C'_1, ..., C'_m$, kde vrchol u z grafu G patří do třídy C'_i právě tehdy, když (u, v_i) patří do třídy ekvivalence C_i. Ukážeme, že rozdělení vrcholů grafu G do tříd C'_i je dobře definovaný rozklad $V(G)$.

• Pokud vrchol u z grafu G patří do tříd C'_i a C'_j, pak $(u; v_i)$ patří do C_i a $(u; v_j)$ patří do C_j. Protože K_m je kompletní graf, pak $\{v_i; v_j\}$ tvoří hranu v K_m, $\{(u; v_i), (u; v_j)\}$ je hranou v kartézském součinu $G\square K_m$. Tohle je spor s tím, že S je nezávislá množina. Tedy množiny $C'_1, ..., C'_m$ jsou po dvou disjunktní.

• Vrcholy nezávislé množiny S zapíšeme následujícím způsobem:

$$
\begin{aligned}
&(u^1_1; v_1), (u^1_2; v_1), \ldots, (u^1_{i(1)}; v_1)\\
&(u^2_1; v_2), (u^2_2; v_2), \ldots, (u^2_{i(2)}; v_2)\\
&\ldots\\
&(u^m_1; v_m), (u^m_2; v_m), \ldots, (u^m_{i(m)}; v_m).
\end{aligned}
$$

Předpokládejme, že nějaký vrchol se v seznamu objeví dvakrát, tj. $u^i_j = u^k_l$. Pak, protože K_m je kompletní graf, $\{v_i; v_k\}$ je hranou v K_m, takže $\{(u^i_j; v_i); (u^k_l; v_k)\}$ je hranou v kartézském součinu $G\square K_m$. To je ale opět spor s tím, že S je nezávislá množina. Tedy, všechny vrcholy u^i_j, které se objeví na seznamu prvků jsou rozdílné a protože $|S| = n$, je zde n rozdílných vrcholů u^i_j. Proto každý vrchol grafu G je obsažený v nějaké třídě ekvivalence C_i.

Přiřadíme barvu i vrcholu u z grafu G, pokud u patří do ekvivalentní třídy C_i. Tím dostáváme dobré obarvení m barvami všech vrcholů grafu G. ∎

Definice potřebné k popsání algoritmu

Než uvedeme samotný algoritmus nadefinujeme si některé pojmy a procedury, které budeme potřebovat pro správné chápání chodu algoritmu.

Připojitelný vrchol

Přidáme-li do nezávislé množiny S vrchol v z okolí nezávislé množiny S, tvrdíme, že tento vrchol je připojitelný pokud $S \cup \{v\}$ je stále nezávislou množinou vrcholů.

Počet připojitelných vrcholů $\rho(S)$

Označujeme $\rho(S)$ počet připojitelných vrcholů k nezávislé množině vrcholů S.

Procedura A

Na vstupu procedury máme nezávislou množinu vrcholů S. Pokud množina nezávislých vrcholů S nemá připojitelný vrchol, vrať množinu nezávislých vrcholů S. Jinak pro každý připojitelný vrchol (u, v) k S najdi číslo $\rho(S \cup \{(u; v)\})$ k množině nezávislých vrcholů $S \cup \{(u; v)\}$. Nechť $(u; v)_{max}$ označuje připojitelný vrchol takový, že $\rho(S \cup \{(u; v)_{max}\})$ je největší. Pak $S \cup \{(u; v)_{max}\}$ je maximální nezávislou množinou. Tento postup opakujeme, dokud nezávislá množina vrcholů S nemá žádný další připojitelný vrchol.

Procedura B

Na vstupu procedury máme maximální nezávislou množinu vrcholů S. Pokud není žádný vrchol $(u_1; v_1)$ mimo množinu S takový, že $(u_1; v_1)$ má právě jeden sousední vrchol $(u_2; v_2)$ uvnitř množiny S, pak vrať S. Jinak najdeme vrchol $(u_1; v_1)$ vně množiny takový, že $(u_1; v_1)$ právě jeden sousední vrchol $(u_2; v_2)$ uvnitř S. Definujeme $S^{(u_1;v_1),(u_2;v_2)}$ přidáním vrcholu $(u_1; v_1)$ do množiny S a odebráním vrcholu $(u_2; v_2)$ z množiny S. Aplikujeme *proceduru A* na množinu $S^{(u_1;v_1),(u_2;v_2)}$ a výstupem procedury je výsledná množina nezávislých vrcholů.

Algoritmus

Na vstupu máme jednoduchý graf G s n vrcholy a hledáme dobré vrcholové barvení s použitím m barev. Označme si vrcholy grafu G jako $\{u_1; u_2; \ldots; u_n\}$ a vrcholy kompletního grafu K_m jako $\{v_1; v_2; \ldots; v_m\}$. Snažíme se najít maximální, nezávislou množinu vrcholů kartézského součinu grafů $G \square K_m$. Jestliže v některém z kroků má nezávislá množina vrcholů velikost alespoň n, přesuneme se na část III.

Část I.

- Pro $i = 1, 2, \ldots, n$ a $j = 1, 2, \ldots, m$ v každém kroce proveď:
- Inicializace nezávislé množiny $S_{i,j} = \{(u_i, v_j)\}$.
- Proveď *proceduru A* na nezávislé množině $S_{i,j}$.
- Pro $r = 1, 2, \ldots, n$ proveď na $S_{i,j}$ *proceduru B* r-krát.
- Výsledkem je maximální nezávislá množina vrcholů $S_{i,j}$.

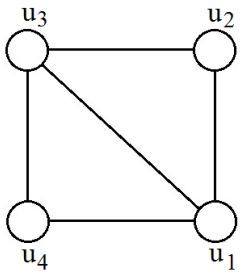
Část II.

- Pro každý pár z maximální nezávislé množiny $S_{i,j}$, $S_{k,l}$ nalezené v *části I*
- Inicializuj nezávislou množinu $S_{i,j,k,l} = S_{i,j} \cap S_{k,l}$.
- Proveď *proceduru A* na množině $S_{i,j,k,l}$.
- Pro $r = 1, 2, \ldots, n$ proveď na $S_{i,j,k,l}$ *proceduru B* r-krát.
- Výsledkem je maximální nezávislá množina vrcholů $S_{i,j,k,l}$.

Část III. - Jestliže byla nalezena nezávislá množina vrcholů S o velikosti n v kterékoliv části algoritmu, vypiš nezávislou množinu S a odpovídající dobré barvení vrcholů grafu G pomocí m barev (dle *kartézského lemma* 6.1). Jinak vypiš, že algoritmus nenalezl dobré vrcholové barvení grafu G pomocí m barev.

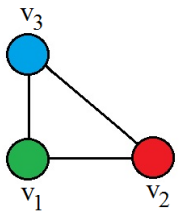
Příklad 10

Na tomto příkladě si ukážeme, jak algoritmus postupuje u hledání dobrého vrcholového barvení pomocí tří barev (zelená, červená, modrá) pro graf G (*Obrázek 18*).



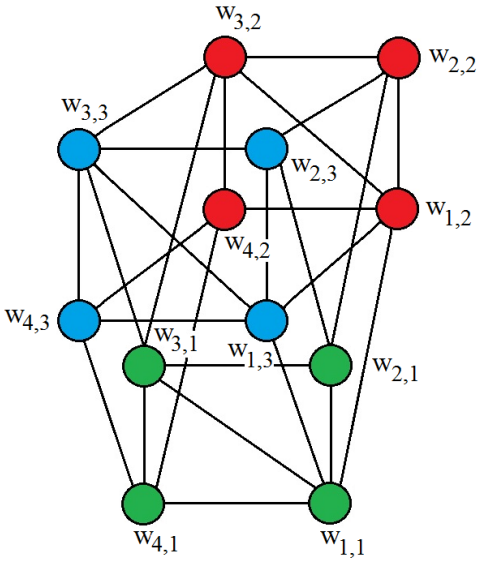
Obrázek 24: Graf G.

Jelikož máme graf G obarvit pomocí tří barev, budeme také potřebovat kompletní graf K_3 (*Obrázek 19*).



Obrázek 25: Graf K_3.

Algoritmus sestrojí kartézský součin grafů $G\square K_3$. Výsledkem kartézského součinu je graf $W = G\square K_3$ (*Obrázek 20*), pro jednoduchost jsme použili značení vrcholů $w_{i,j} = (u_i; v_j)$.



Obrázek 26: Graf W, který je kartézským součinem $G \square K_3$.

Algoritmus nyní prohledává graf W a hledá množinu nezávislých vrcholů S o velikosti 4. Část I. - pro $i = 1$ a $j = 1$ inicializuj nezávislou množinu vrcholů jako:

$$S_{1,1} = \{(w_{1,1})\}.$$

Protože $|S| = 1$ provede se *procedura A* a dostaneme tyto výsledky:

Připojitelný vrchol $w_{i,j}$ k $S_{1,1}$	Připojitelné vrcholy $S_{1,1} \cup \{(w_{i,j})\}$	$\rho(S_{1,1} \cup \{(w_{i,j})\})$
$(w_{2,2})$	$(w_{3,3}); (w_{4,2}); (w_{4,3})$	3
$(w_{2,3})$	$(w_{3,2}); (w_{4,2}); (w_{4,3})$	3
$(w_{3,2})$	$(w_{2,3}); (w_{4,3})$	2
$(w_{3,3})$	$(w_{2,2}); (w_{4,2})$	2
$(w_{4,2})$	$(w_{2,2}); (w_{2,3}); (w_{3,3})$	3
$(w_{4,3})$	$(w_{2,2}); (w_{2,3}); (w_{3,2})$	3

Tabulka 1: Připojitelné vrcholy k $S_{1,1} = \{(w_{1,1})\}$.

Z *Tabulky 1* vyplývá, že maximální $\rho(S_{1,1} \cup \{(w_{i,j})\}) = 3$ pro $(w_{i,j}) = (w_{2,2})$. Připojíme tedy vrchol $(w_{2,2})$ do množiny $S_{1,1}$.

Nezávislá množina vrcholů $S_{1,1} = \{(w_{1,1}); (w_{2,2})\}$ má nyní velikost $|S_{1,1}| = 2$.

Připojitelný vrchol $w_{i,j}$ k $S_{1,1}$	Připojitelné vrcholy $S_{1,1} \cup \{(w_{i,j})\}$	$\rho(S_{1,1} \cup \{(w_{i,j})\})$
$(w_{3,3})$	$(w_{4,2})$	1
$(w_{4,2})$	$(w_{3,3})$	1
$(w_{4,3})$	–	0

Tabulka 2: Připojitelné vrcholy k $S_{1,1} = \{(w_{1,1}); (w_{2,2})\}$.

Z *Tabulky 2* vyplývá, že maximální $\rho(S_{1,1} \cup \{(w_{i,j})\}) = 1$ pro $(w_{i,j}) = (w_{3,3})$. Připojíme tedy vrchol $(w_{3,3})$ do množiny $S_{1,1}$.

Nezávislá množina vrcholů $S_{1,1} = \{(w_{1,1}); (w_{2,2}); (w_{3,3})\}$ má nyní velikost $|S_{1,1}| = 3$.

Připojitelný vrchol $w_{i,j}$ k $S_{1,1}$	Připojitelné vrcholy $S_{1,1} \cup \{(w_{i,j})\}$	$\rho(S_{1,1} \cup \{(w_{i,j})\})$
$(w_{4,2})$	–	0

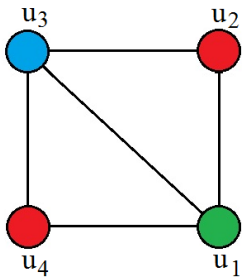
Tabulka 3: Připojitelné vrcholy k $S_{1,1} = \{(w_{1,1}); (w_{2,2}); (w_{3,3})\}$.

Z *Tabulky 3* vyplývá, že maximální $\rho(S_{1,1} \cup \{(w_{i,j})\}) = 0$ pro $(w_{i,j}) = (w_{4,2})$. Připojíme tedy vrchol $(w_{4,2})$ do množiny $S_{1,1}$.

Dostáváme tak maximální nezávislou množinu vrcholů

$$S_{1,1} = \{(w_{1,1}); (w_{2,2}); (w_{3,3}); (w_{4,2})\}.$$

Protože $|S_{1,1}| = 4$ přeskočíme <u>část II.</u> a pokračujeme <u>částí III.</u>, kde na výstupu algoritmus vypíše nezávislou množinu vrcholů $S_{1,1}$ jako požadované dobré barvení grafu G pomocí tří barev. Výstup chápeme tak, že pokud $S_{1,1}$ obsahuje vrchol $w_{1,1}$, vezmeme vrchol u_1 z grafu G a obarvíme ho barvou v_1, čili zelenou barvou. Tímhle způsobem obarvíme zbylé vrcholy grafu G, viz. *Obrázek 21.*



Obrázek 27: Dobře obarvený graf G pomocí 3 barev.

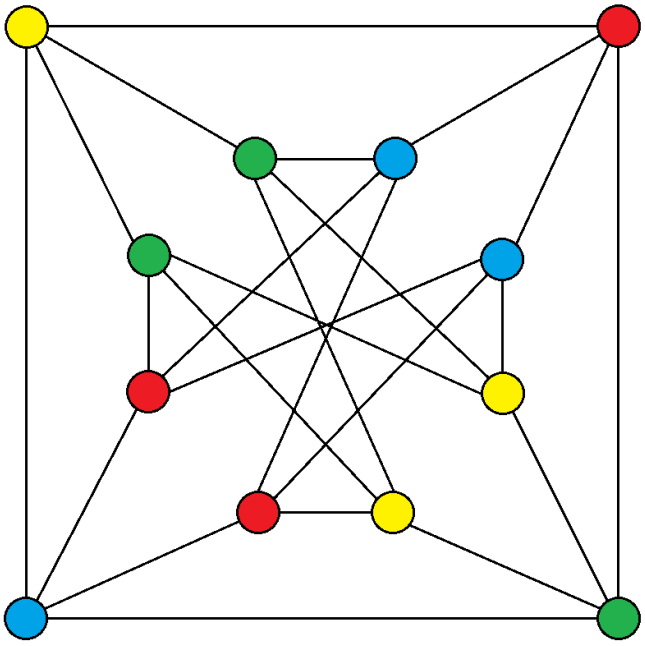
■

Složitost

Vyvstává otázka, jakou má vlastně Algoritmus 3 složitost? Pokud máme zadaný jednoduchý graf G s n vrcholy a m barvami, tak algoritmus vykoná maximálně $(mn)^8 + 2(mn)^7 + (mn)^6 + (mn)^5 + (mn)^4 + (mn)^3 + (mn)^2$ kroků při hledání dobrého vrcholového barvení grafu G pomocí m barev. Tedy složitost je $O((mn)^8)$. Celý postup výpočtu složitosti je uveden v [8].

Nezapomínejme však, že se jedná o počet kroků pro nejhorší možný případ. Skutečný počet kroků, které algoritmus vykonal, závisí na hodnotách m, n a na struktuře grafu G. Pro všechny testované případy grafů G se známým $\chi(G)$ algoritmus nalezl dobré vrcholové barvení pomocí $m = \chi(G)$ barev a skončil v mnohem kratším čase.

Algoritmus 3 byl testovaný na různých grafech, na kterých vždy našel dobré vrcholové m-barvení, kdy $m = \chi(G)$, například: Mycielskiho grafy, Bondy-Murtyho grafy, Petersonův graf, Grötzscheho graf nebo Chvátalův graf, o kterém si povíme víc, než jen to, že se jedná o graf s chromatickým číslem $\chi(G) = 4$ (*Obrázek 22*).



Obrázek 28: Chvátalův graf a jeho dobré vrcholové barvení pomocí 4 barev, které nalezl Algoritmus 3.

Chvátalův graf - Tento graf byl definován Václavem Chvátalem roku 1970 [12]. Jedná se o neorientovaný graf s 12 vrcholy a 24 hranami (viz. *Obrázek 22*), který je nejmenším možným 4-chromatickým, 4-regulárním ($\forall v \in V(G) : deg(v) = 4$) grafem bez trojúhelníků. Tj.: nejmenší cyklus, který Chvátalův graf obsahuje, je délky 4.

Jediným menším 4-chromatickým grafem je *Grötzscheho graf* s 11 vrcholy a 20 hranami. Oproti Chvátalovu grafu není regulární a $\Delta(G) = 5$.

U uvedeného Algoritmu 3 se zdá, že pěkně funguje a má polynomiální složitost, nicméně autor nedokázal, že skutečně pro každý graf dobré barvení pomocí $m = \chi(G)$ barev algoritmus nalezne. Publikace vyšla na Amazonu a byla citována v časopise [15].

Mohlo by být zajímavé testovat algoritmus na velkých „benchmark“ grafech, používaných k testování algoritmů pro dobré vrcholové barvení uvedených v [14], což provedeno nebylo.

7 Praktické využití vrcholového barvení

V podstatě existuje jen velmi málo reálných problému, které mohou být přímo řešeny pomocí vrcholového barvení. Některé z nich si v této kapitole představíme. Vycházíme z publikace Philippa Galiniera [14].

Časový rozvrh

Časový rozvrh zvaný též „timetabling“ je problém zabývající se přiřazením časového úseku k události (cvičení, zkouška, přednáška, ...) vzhledem k nějakým párovým omezením. Tato omezení například obsahují páry událostí, které nemohou být přiřazeny do stejného časového úseku - jeden učitel nemůže mít v daný okamžik zároveň přednášku a cvičení.

Barvení grafu se již dlouhou dobu využívá pro modelování těchto „timetabling“ problémů. Obvykle jsou události v grafu značeny jako vrcholy, hrany odpovídají omezením a časové úseky bývají barvy.

Alokace registrů

Alokace registrů je významný problém v oblasti kompilátorů. Vzhledem ke zdrojovému kódu, je totiž problém určit, které proměnné budou uloženy (přiřazeny) do cache paměti a které se uloží na RAMce (vyžadují pomalejší přístupový čas). Každá z těchto proměnných musí mít přidělenou jinou část paměti, aby nenastala situace, kdy zapíšeme 2 různé proměnné na stejný úsek paměti - došlo by k přemazání (ztrátě) dat.

Při modelování těchto problémů představují vrcholy proměnné, omezení jsou opět znázorněné pomocí hran a typ paměti rozlišujeme barvami.

Přiřazení frekvence

Problém s přiřazením frekvence hraje roli v telekomunikačních sítích po dobu více než 20 let. Snažíme se totiž přiřadit danou frekvenci k vysílačům (antény, routery, ...). Omezením je zde vzdálenost vysílačů. Snažíme se totiž zabránit tomu, aby geograficky blízké vysílače nesdílely stejnou frekvenci nebo si vzájemně nezasahovaly do vysílaného pásma - dochází pak k rušení signálu (přeslechy na linkách).

Proto lze chápat problém přiřazení frekvencí jako rozšířené barvení. Vrcholy reprezentují vysílače a barvami značíme frekvence.

Ve skutečnosti je tento problém komplikovaný tím, že máme pevně daný rozsah pro frekvence, ze kterých můžeme vybírat. Často tedy nemáme jinou možnost, než přistoupit na kompromis.

Směrování a přiřazení vlnové délky

S problémy směrování a přiřazovaní vlnové délky se setkáme v optických sítích. Optická síť je prezentována jako neorientovaný graf, kde uzly tvoří stanice a hrany jsou zde spoje (linky).

Jednu z dvojic stanic nazveme „požadavky“ (demands). K uspokojení požadavků, je potřeba vytvořit tzv. „světelnou cestu“ (lightpath), což je cesta v grafu s přidělenou frekvencí. Správným řešením je pak množina světelných cest, přičemž žádné dvě světelné cesty, které sdílí stejný spoj, nesmí mít stejnou frekvenci.

Při realizaci narážíme na problém jako u problému s přiřazením frekvence. Frekvence a vlnové délky, které zde využíváme jsou pevně dané, snažíme se tedy uspokojit co nejvíce požadavků.

8 Závěr

Cílem této bakalářské práce bylo zpracovat přehled základních typů algoritmů pro dobré vrcholové barvení včetně jejich složitosti. V práci jsem se zabýval vrcholovým barvením grafů za pomocí různých metod založených například na třídě grafu nebo stupních vrcholů. Dále jsem se zabýval algoritmy, které jsou schopné najít vrcholové barvení grafu. Zde jsem zjistil, že vrcholové m-barvení je dobře známý NP-kompletní problém a že existuje mnoho výsledků zabývajících se tímto problémem a mnoho různých algoritmů a jejich vylepšení. Viz. přehled heuristických algoritmů na straně 53 nebo v článku [14], samozřejmě takových článků existují stovky, protože barvením grafů se zabývají odborníci již století.

Nejlepší by bylo používat deterministické algoritmy, protože zaručují pro stejný vstup vždy stejný a správný výstup. Jenže tyto algoritmy jsou limitovány svou časovou složitostí a ne vždy se dají použít na velké grafy. V práci jsem uvedl příklad, stránky 53-62, deterministického algoritmu (Algoritmus 3) z roku 2006 s ukázkami funkčnosti. Autor tohoto algoritmu (Ashay Dharwaker) ukazuje, že jeho algoritmus dokázal najít vrcholové m-barvení na grafech jako jsou např.: Mycielskiho grafy (95 vrcholů), Bondy-Murtyho grafy (16 vrcholů), či Chvátalův graf (12 vrcholů) v polynomiálním čase, kde $m \leq \Delta(G)$. Což je velice zajímavé vzhledem k tomu, že dříve známé deterministické algoritmy jsou uváděny s exponenciální složitostí, viz. například článek [16]. Pokud by neexistoval graf, pro který by uvedený algoritmus nebyl schopen najít m-barvení pro $m = \chi(G)$, znamenalo by to $P = NP$. Předpokládáme totiž, že $P \neq NP$ a je tedy pravděpodobné, že takový graf existuje.

V praxi se ale hojně využívají heuristické algoritmy, protože poskytují relativně přesné výsledky za daleko kratší čas i pro velké grafy. V práci uvedené heuristické algoritmy (Algoritmus 1 a Algoritmus 2 z kapitoly 6) mají polynomiální složitost $O(n^2)$, kde n je počet vrcholů. Uvedl jsem heuristický Algoritmus 1 využívající hladové barvení, kde je vysvětlena jeho funkčnost a poukázáno na jeho nedostatky. Hlavním nedostatkem je náhodný výběr pořadí vrcholů, v kterém se budou vrcholy barvit. Poté jsem uvedl modifikovanou verzi tohoto algoritmu. Modifikace spočívala v tom, že algoritmus obarvuje vrcholy v pořadí dle stupně vrcholů (od největšího stupně po nejmenší). Tato „drobná“ změna zaručuje přesnější výsledky ve smyslu použití menšího nebo nejmenšího možného počtu barev. Přesto tento algoritmus není dokonalý. Jeho klady i zápory jsou pak demonstrované na příkladech, strany 44, 46-52.

Díky této práci jsem se dozvěděl několik zajímavých věcí ohledně vrcholového barvení grafů. Není to totiž tak jednoduchá věc, jak se zdálo na cvičeních diskrétní matematiky. Myslím si, že z velké části jsem splnil požadavky na mne kladené zadáním. Byl bych rád, kdyby se mi do práce podařilo zahrnout některý z dalších heuristických algoritmů zmíněných v [14]. Vzhledem k časové náročnosti bych nebyl schopen zpracovat další algoritmus na stejné úrovni, jako algoritmy v práci již uvedené.

Možností dalšího zkoumání v oblasti vrcholového barvení grafů je mnoho, například: nastudovat a zpracovat aktuálně nejlepší známé heuristiky [14] s cílem další optimalizace, dále testovat Algoritmus 3 na tzv. „benchmark“ grafech, popřípadě se pokusit najít graf, pro který by Algoritmus 3 nedokázal najít dobré vrcholové m-barvení, kde $m = \chi(G)$.

Literatura

[1] KOVÁŘ Petr. *Teorie grafů* [online]. Ostrava, 2012 [cit. 2017-04-21]. Dostupné z: http://mi21.vsb.cz/modul/teorie-grafu

[2] *Graph coloring* [online]. Poslední aktualizace 8. dubna 2017 23:05 [cit. 2017-04-21], Wikipedie. Dostupné z: https://en.wikipedia.org/wiki/Graph_coloring

[3] *Základní grafové algoritmy* [online]. Praha, 2010 [cit. 2017-04-21]. Dostupné z: http://kam.mff.cuni.cz/~kuba/ka/

[4] MAREŠ Martin. *Knížka o algoritmech a datových strukturách* [online]. Praha, 2017 [cit. 2017-04-21]. Dostupné z: http://mj.ucw.cz/vyuka/ads/

[5] *Grafy a grafové algoritmy 2: Prohledávání do šířky a do hloubky* [online]. Populárně naučný portál POPULAR [cit. 2017-04-21]. Dostupné z: http://popular.fbmi.cvut.cz/it/Stranky/grafy---prohledavani-do-sirky-a-do-hloubky.aspx

[6] ŽOLTÁ Lucie. *Složitost algoritmů* [online]. [cit. 2017-04-21]. Dostupné z: http://lucie.zolta.cz/index.php/zaklady-teoreticke-informatiky/15-turingovy-a-ram-stroje

[7] ROSEN H. Kenneth. *Discrete Mathematics and its Applications, seventh edition.* New York, 2012 [cit. 2017-04-21], ISBN 978-0-07-338309-5

[8] DHARWADKER Ashay. *The Vertex Coloring Algorithm* [online]. Haryana, 2006 [cit. 2017-04-21], ISBN 1466391324. Dostupné z: http://www.dharwadker.org/profile.html

[9] BACHMANN Paul. *Die Elemente der Zahlentheorie.* Lipsko, 1892 [cit. 2017-04-21], ISBN 9925023092

[10] KNUTH E. Donald. *The Art of Computer Programming: Sorting and Searching. Volume 3 (first edition).* Boston, 1973 [cit. 2017-04-21], ISBN 0-201-03803-X

[11] KARP M. Richard. *Reducibility Among Combinatorial Problems*, University of California at Barkeley, 1972 [cit. 2017-04-21], ISBN 978-1-4684-2001-2

[12] CHVÁTAL Václav. *Journal of Combinatorial Theory, Volume 9, Issue 1, July 1970, Pages 93-94* [online]. Kanada, 1970 [cit. 2017-04-21]. Dostupné z: http://www.sciencedirect.com/science/journal/00219800/9/1

[13] *Grundy number* [online]. Poslední aktualizace 25. prosince 2016 18:19 [cit. 2017-04-21], Wikipedie. Dostupné z: https://en.wikipedia.org/wiki/Grundy_number

[14] GALINIER Philippe, HAMIEZ Jean-Philippe, HAO Jin-Kao, PORUMBEL Daniel. *Recent Advances in Graph Vertex Coloring*. Heidelberg, 2013 [cit. 2017-04-21], ISBN 978-3-642-30504-7

[15] *Journal of the Korean Society for Industrial and Applied Mathematics* [online]. Vol.11 NO.4 str. 19-38, 2007 [cit. 2017-04-21]. Dostupné z: `http://mathnet.kaist.ac.kr/mathnet/kms_content.php?no=379610`

[16] *Operations Research Letters*. Volume 32, Issue 6, Stránky 547-556, 2004 [cit. 2017-04-21]. Dostupné z: `http://www.sciencedirect.com/science/journal/01676377/32/6`