UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

L(2,1)-FARBENIA ŠPECIÁLNYCH GRAFOV

Diplomová práca

2016

Bc. Anna Dresslerová

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

L(2,1)-FARBENIA ŠPECIÁLNYCH GRAFOV

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Katedra informatiky
Vedúci práce:	RNDr. Michal Forišek, PhD.

Bratislava, 2016

Bc. Anna Dresslerová

67616643

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Anna Dresslerová
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: L(2,1) farbenia špeciálnych grafov
L(2,1) colorings of special graphs

Cieľ: Cieľom práce je skúmať algoritmické riešenia problému L(2,1) farbenia grafov. Súčasťou práce by mal byť prehľad najvýznamnejších výsledkov súvisiacich s touto oblasťou. Následne by sa autorka mala pokúsiť o zlepšenie známeho výsledku pre t-skoro stromy. Ďalej by sa práca mala zaoberať analýzou a efektívnym riešením ďalších špeciálnych prípadov tohto problému, prípadne aj jeho všeobecnou verziou.

Vedúci: RNDr. Michal Forišek, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: doc. RNDr. Daniel Olejár, PhD.

Dátum zadania: 09.12.2014

Dátum schválenia: 12.12.2014

prof. RNDr. Branislav Rovan, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie

Ďakujem môjmu školiteľovi RNDr. Michalovi Foriškovi, PhD. za odborné vedenie, ľudský prístup a cenné rady počas písania práce. Taktiež ďakujem svojej rodine za neustálu podporu počas celého štúdia.

Abstrakt

$L(2,1)$-farbenie je vzdialenosťou podmienené farbenie, ktoré priradzuje vrcholom prirodzené čísla s nulou. Susedné vrcholy musia mať čísla, ktoré sa líšia aspoň o dva a vrcholy, ktoré sú vo vzdialenosti dva, musia mať rôzne čísla. Chceme vedieť, aké je najmenšie číslo k, že existuje $L(2,1)$-farbenie grafu G, ktoré nepoužíva čísla väčšie ako k. Toto číslo potom označujeme $\lambda(G)$. Nájsť $\lambda(G)$ je vo všeobecnosti veľmi ťažké. Dokonca otestovať či $\lambda(G) \leq k$ je NP-úplný problém už pre sériovo-paralelné grafy. Existuje len málo tried grafov, kde je tento problém polynomiálne riešiteľný. Triedy, ktoré patria do tejto kategórie sú stromy a grafy s konštantným počtom cyklov. My sme tento poznatok rozšírili o triedu cyklových stromov (kaktusy s vrcholovo disjunktnými kružnicami). Ďalej sme určili tesné ohraničenia $\lambda(G)$ vzhľadom na maximálny stupeň grafu. Dokázali sme, že pre každý kaktus G s maximálnym stupňom Δ platí: $\Delta+1 \leq \lambda(G) \leq \Delta+3$. Ak navyše $\Delta \geq 5$, tak platí: $\Delta+1 \leq \lambda(G) \leq \Delta+2$.

KĽÚČOVÉ SLOVÁ: kaktus, cyklový strom, L(2,1)-farbenie

Abstract

An L(2,1)-labelling is a labelling of the vertex set of graph with non-negative integers such that the labels of adjacent vertices differ by at least two and the labels of vertices at distance 2 are distinct. It is required to determine, for a given graph G, the smallest integer k such that G admits an L(2,1)-labelling with integers not exceeding k; this invariant is denoted by $\lambda(G)$. Determining $\lambda(G)$ is known to be a hard problem. To test whether $\lambda(G) \leq k$ is NP-complete even for series-parallel graphs. On the other hand, there exist classes of graphs where this problem is polynomially solvable. Classes in this group are trees and graphs with constant number of cycles. In this thesis we show that cycle-trees (cacti with disjoint cycles) also belong to this group. We also derive tight upper and lower bounds for the λ-number of cacti. We prove that $\Delta + 1 \leq \lambda(G) \leq \Delta + 3$ for any cactus G of maximum degree Δ. Furthermore we prove that $\Delta + 1 \leq \lambda(G) \leq \Delta + 2$ for any cactus with $\Delta \geq 5$.

KEYWORDS: cactus, cycle-tree, $L(2,1)$-labelling

Obsah

Úvod

V tejto práci sa zaoberáme problémom $L(2,1)$-farbenia kaktusov a cyklových stromov. Najprv prezradíme niečo o $L(2,1)$-farbení. Ide o vzdialenosťou podmienené farbenie pomocou nezáporných celých čísel. Tento druh farbenia grafov je inšpirovaný problémom priradenia frekvencií (angl. Frequency Assignment Problem (FAP)). Koncept vzdialenosťou podmieneného farbenia grafov predstavil Hale v [13] a podľa [12] Roberts [19] bol prvý, kto navrhol model $L(2,1)$-farbenia.

V probléme FAP je na vstupe sieť vysielačov v broadcast-ovej sieti. Cieľom je priradiť vysielačom frekvencie tak, aby sa čo najmenej rušili. Ak dva vysielače môžu medzi sebou priamo komunikovať (ich oblasti pôsobenia sa vo veľkej miere prekrývajú), nesmú mať podobné frekvencie. Ďalšou podmienkou je, že ak dva vysielače vedia komunikovať s tretím vysielačom (ich oblasti sa čiastočne prekrývajú), tak tieto dva vysielače nemôžu mať rovnaké frekvencie.

Z týchto podmienok vznikli podmienky pre $L(2,1)$-farbenie grafu. Graf v tomto prípade predstavuje sieť vysielačov. Vrcholy sú vysielače a ak je medzi nimi hrana, znamená to, že vedia medzi sebou priamo komunikovať. $L(2,1)$-farbenie grafu je také priradenie celých nezáporných čísiel vrcholom, ktoré spĺňa dve podmienky:

- Ak sú dva vrcholy susedné, potom ich čísla sa musia líšiť aspoň o dva.
- Ak majú dva vrcholy spoločného suseda, potom ich čísla musia byť rôzne.

Samozrejme existujú aj zovšeobecnenia tohto pojmu. Napríklad $L(p,q)$-farbenie. My však zostaneme len pri $L(2,1)$-farbení.

Cieľom je pre daný graf G nájsť minimálne také číslo k, že existuje $L(2,1)$-farbenie grafu G, ktoré nepoužije číslo väčšie ako k. Inými slovami, zaujíma nás, aký najmenší rozsah čísiel musíme použiť. Toto číslo pre daný graf označujeme $\lambda(G)$. V tejto práci sa však zaoberáme rozhodovacou verziou tohto problému. V prípade rozhodovacej verzie chceme pre daný graf G a číslo k rozhodnúť, či $\lambda(G) \leq k$. Inými slovami, snažíme sa zistiť, či existuje $L(2,1)$-farbenie grafu G pomocou farieb $0, 1, \ldots, k$. Uvedieme niekoľko známych výsledkov v tejto oblasti.

Griggs a Yeh v [12] ukázali, že určiť $\lambda(G)$ pre ľubovoľný graf G je NP-ťažký problém. Dokonca vyslovili hypotézu, že tento problém je NP-úplný pre stromy. Neskôr však Chang a Kuo [6] našli polynomiálny algoritmus pre rozhodovaciu verziu problému $L(2,1)$-farbenia pre stromy s časovou zložitosťou $O(|V(G)|k^{4.5})$. Tento výsledok neskôr vylepšili Hasunuma a spol. [14] nájdením lineárneho algoritmu. Fiala a spol. [10] miernou úpravou prvého polynomiálneho algoritmu pre stromy získali polynomiálny algoritmus pre t-takmer stromy pre fixné t. t-takmer stromy sú súvislé grafy, ktoré majú n vrcholov a $n+t-1$ hrán. Ďalej v tomto článku ukázali, že pre všeobecné grafy je rozhodovací problém $L(2,1)$-farbenia NP-úplný pre každé fixné $k \geq 4$. Pre $k \leq 3$ je problém polynomiálny. NP-úplnosť pre planárne grafy dokázali Bodlaender a spol. [5] pre $k = 8$, neskôr Janczewski a spol. [16] pre $k = 4$, a nakoniec Eggeman a spol. [8] pre $k \geq 4$. V skutočnosti, vzdialenosťou podmienené farbenie je ťažší problém ako obyčajné farbenie. Tento fakt je najlepšie vidieť na tom, že rozhodnúť či $\lambda(G) \leq k$ je NP-úplné dokonca aj pre sériovo-paralelné grafy [9].

Ďalšou oblasťou, ktorou sa v práci zaoberáme, sú ohraničenia čísla λ pre kaktusy. Výskum v tejto oblasti podnietila hypotéza uverejnená v článku [12]. Griggs a Yeh v hypotéze tvrdia, že pre ľubovoľný graf G s maximálnym stupňom $\Delta \geq 2$ platí $\lambda(G) \leq \Delta^2$. Ďalej uviedli tesné ohraničenie čísla λ pre stromy. Pre ľubovoľný strom T platí $\Delta+1 \leq \lambda(T) \leq \Delta+2$, kde Δ je maximálny stupeň T. Pre všeobecné grafy ďalej ukázali horné ohraničenie $\lambda(G) \leq \Delta^2+2\Delta$. Znakom Δ vždy myslíme maximálny stupeň grafu G. Tento výsledok bol neskôr vylepšený autormi Chang a Kuo [6] na $\lambda(G) \leq \Delta^2+\Delta$. Momentálne najlepší známy výsledok uverejnil Conçalves [11] pre $L(p,1)$-farbenie. Konkrétne $\lambda(G,p,1) \leq \Delta^2+(p-1)\Delta-2$. Pre naše $L(2,1)$-farbenie to znamená $\lambda(G) \leq \Delta^2+\Delta-2$.

Spomenieme aj planárne grafy ako zástupcov menších tried. Heuvel a McGuinness [20] ukázali, že pre $L(p,q)$-farbenie a planárne grafy platí $\lambda(G,p,q) \leq (4q-2)\Delta+10p+38q-23$. Pre $L(2,1)$-farbenie to znamená $\lambda(G) \leq 2\Delta+35$. Podobný výsledok pre $L(p,q)$-farbenie mali aj autori Molloy a Salavatipour [18], ktorým vylepšili hornú hranicu pre $L(2,1)$-farbenie na $\lambda(G) \leq \lceil \frac{5}{3}\Delta \rceil + 95$. Spomenieme ešte jeden zaujímavý výsledok k planárnym grafom. Wang a Lih [21] skúmali ohraničenia $\lambda(G,p,q)$ pre $L(p,q)$-farbenie pre planárne grafy, ktorým zdola ohraničili dĺžku najmenšieho cyklu. Zaujímavé je, že čím bolo dolné ohraničenie dĺžky cyklu prísnejšie (boli povolené len dlhšie cykly), tak tým menšie bolo horné ohraničenie $\lambda(G,p,q)$.

Hypotéza, ktorá rozbehla tento výskum, ešte nie je dokázaná. Vieme však, že platí pre veľa tried grafov. Zaujímavosťou je, že jediné grafy, o ktorých vieme, že dosahujú hornú hranicu a majú maximálny stupeň väčší ako dva, sú Petersenov graf a Hoffmannov-Singletonov graf.

V našej práci sme sa rozhodli, že budeme skúmať triedy kaktusov a cyklových stromov. Kaktus je jednoduchý súvislý neorientovaný graf, pre ktorý platí, že každá jeho hrana leží na najviac jednej kružnici. Inými slovami, jediné jeho 2-súvislé komponenty (bloky) sú hrany

a kružnice. Cyklovými stromami nazývame také kaktusy, pre ktoré navyše platí, že každý vrchol leží na najviac jednej kružnici.

Tieto triedy sme si vybrali preto, lebo zatiaľ o nich nebolo známe, aký zložitý je pre ne problém $L(2,1)$-farbenia. Vieme, že sú podtriedami sériovo-paralelných grafov. Ďalej môžu obsahovať ľubovoľne veľa kružníc a zároveň majú príjemnú stromovitú štruktúru. Predpokladali sme, že pre obe triedy existuje polynomiálny algoritmus pre rozhodovaciu verziu skúmaného problému. Cieľom práce bolo tieto algoritmy nájsť. Ďalším cieľom bolo nájsť, pokiaľ možno, tesné ohraničenia čísla $\lambda(G)$ pre tieto triedy. Pri prieskume výsledkov z tejto oblasti sme nenašli žiaden korektný výsledok. Našli sme len jeden článok, ktorý sa venoval práve tomuto problému [7]. Po prečítaní článku sme zistili, že dôkazy, ktoré autorka predkladala, neboli korektné a nedali sa ani triviálnym zásahom opraviť.

Prvá kapitola tejto práce obsahuje definície, ktoré sú používané v práci a algoritmus na zakorenenie ľubovoľného kaktusu, ktorý bude použitý pri všetkých ďalších opísaných algoritmoch. Druhá kapitola obsahuje opis polynomiálneho algoritmu pre rozhodovaciu verziu problému $L(2,1)$-farbenia pre cyklové stromy. Je v nej uvedený aj dôkaz jeho korektnosti a odhad jeho časovej zložitosti. V poslednej kapitole sú uvedené dve ohraničenia $\lambda(G)$ pre kaktusy. Okrem dôkazov týchto ohraničení, táto kapitola obsahuje aj dôkaz tesnosti prvého z nich. V rámci tohto dôkazu je uvedená nekonečná trieda kaktusov, ktorých $\lambda(G)$ dosahuje hornú hranicu a aj spôsob, ako bola nájdená.

Kapitola 1

Definície a zakoreňovací algoritmus

Skôr ako sa budeme venovať našim výsledkom, uvedieme ešte pojmy, ktoré budeme ďalej používať. V každej ďalšej časti budeme pracovať so zakoreneným kaktusom. Okrem definícií spojených so zakoreneným kaktusom uvedieme aj algoritmus na efektívne zakorenenie ľubovoľného kaktusu. Pri stromoch by to znamenalo, pre každý vrchol zistiť množinu synov a otca. V našom prípade to je veľmi podobné.

1.1 Definície

Najprv definujeme dôležité pojmy, ktoré budeme používať. Ako prvé formálne definujeme k-$L(2,1)$-farbenie a triedy grafov, ktoré v práci skúmame.

Definícia 1 *k-$L(2,1)$-farbenie grafu G je funkcia, ktorá priradzuje vrcholom grafu G čísla z množiny $\{0,\ldots,k\}$, pričom dodržuje nasledujúce podmienky:*

- *ak $d(u,v)=1$, potom $|f(u)-f(v)|\geq 2$*
- *ak $d(u,v)=2$, potom $|f(u)-f(v)|\geq 1$*

kde u a v sú vrcholy grafu a $d(u,v)$ je ich vzdialenosť v grafe G

Definícia 2 *Kaktus je jednoduchý súvislý neorientovaný graf, pre ktorý platí, že každá jeho hrana leží na najviac jednej kružnici.*

Definícia 3 *Cyklový strom je kaktus, pre ktorý navyše platí, že každý jeho vrchol leží na najviac jednej kružnici.*

Inak povedané, kaktusy môžu obsahovať len hranovo disjunktné kružnice a cyklové stromy len vrcholovo disjuktné kružnice. V prípade cyklových stromov, medzi každými dvoma kružnicami existuje aspoň jedna hrana, ktorá neleží na žiadnej kružnici.

V práci budeme často používať pojem zakorenený kaktus a veľa ďalších pojmov, ktoré s tým súvisia. Kaktus si v prvom rade rozdelíme na bloky (2-súvislé komponenty). Už z definície kaktusu vidno, že každý jeho blok je buď hrana alebo kružnica.

Vyberieme vrchol kaktusu, ktorý budeme označovať ako *koreň*. Kaktus, ktorý má koreň nazývame *zakorenený kaktus*. Akonáhle má kaktus koreň, môžeme sa zaujímať o ďalšie vzťahy medzi jeho vrcholmi.

Definícia 4 *Vrchol v bloku B je* otcovským vrcholom *tohto bloku, ak je zo všetkých jeho vrcholov najbližšie ku koreňu. Nech je to vrchol v. Blok B budeme nazývať* synovským blokom *vrcholu v. Všetky vrcholy v bloku B, okrem vrcholu v, budeme nazývať* synmi *vrcholu v. Pre tieto vrcholy bude blok B ich* otcovským blokom *a vrchol v ich* otcom.

Pre úplnosť by sa patrilo poznamenať, že definícia otcovského vrcholu bloku je korektná. To znamená, že existuje práve jeden vrchol z každého bloku, ktorý je najbližšie ku koreňu.

V prípade, že by to neplatilo, tak existujú aspoň dva vrcholy, ktoré sú najbližšie. Nech sú to vrcholy u a v. Existujú dve cesty rovnakej dĺžky P_u a P_v vedúce z koreňa do u resp. v. Určite existuje taký vrchol x, od ktorého ďalej sa už tieto dve cesty nestretnú. V prípade, že blokom je hrana, vieme nájsť kružnicu, ktorej súčasťou je táto hrana. To je ale spor s tým, že táto hrana je 2-súvislý komponent. Ak je blokom kružnica, potom vieme nájsť inú kružnicu, ktorá s ňou nie je hranovo disjunktná. To je spor s definíciou kaktusu.

Budeme ešte často používať pojem *otcovské hrany*.

Definícia 5 *Nech vrchol v má otcovský blok B.*

- *Ak B je hrana, potom má vrchol v len jednu otcovskú hranu a je to priamo hrana do otca bloku.*
- *Ak B je kružnica, potom má vrchol v dve otcovské hrany. Sú to hrany, ktoré ležia na jeho otcovskej kružnici a sú s ním incidentné.*

Potrebujeme ešte definovať pojmy, ktoré sú blízke pojmu list v stromoch. Definujeme *list* a *listový blok*. Naša nová definícia listu je rozšírením pojmu listu v stromoch.

Definícia 6 *List je vrchol, ktorý patrí len do jedného bloku. Buď je to vrchol so stupňom jeden, alebo vrchol so stupňom dva, ktorý leží na kružnici.*

Definícia 7 *Listovým blokom nazývame blok, ktorý obsahuje $m - 1$ listov, kde m je počet vrcholov v bloku. Ak je to hrana, potom jeden z jej vrcholov má stupeň jedna a ak je to kružnica, potom všetky jej vrcholy okrem jedného majú stupeň dva.*

V práci budeme pomerne často písať namiesto $L(2,1)$-farbenie iba farbenie. Vždy, keď budeme hovoriť o farbení, myslí sa pod tým $L(2,1)$-farbenie.

1.2 Zakoreňovací algoritmus

Pre úplnosť práce uvedieme v tejto kapitole aj algoritmus, pomocou ktorého efektívne zakoreníme ľubovoľný kaktus. Vo všetkých ďalších uvedených algoritmoch predpokladáme, že máme zakorenený kaktus. To znamená, že pre ľubovoľný vrchol kaktusu vieme, či jeho otcovským blokom je hrana alebo kružnica a ktoré vrcholy sú jeho synmi. Pre synov ďalej potrebujeme vedieť ich štruktúru. To znamená, že potrebujeme vedieť, ktoré vrcholy spolu ležia na synovských kružniciach a ktoré ležia na synovských hranách. V prípade kružníc, musíme vedieť rozlíšiť, ktorý vrchol patrí do ktorej kružnice a vrcholy patriace do jednej kružnice mať uložené v poradí, ako ležia na kružnici.

Tieto informácie sa dajú získať pomocou jednoduchej úpravy prehľadávania do hĺbky. Uvedieme najskôr algoritmus a potom ukážeme jeho korektnosť a efektívnosť.

1.2.1 Algoritmus

Predpokladáme, že máme kaktus daný zoznamami susedov pre každý vrchol. Vrcholy sú číslované od 0 po n. Ďalej máme vrchol r, v ktorom chceme tento kaktus zakoreniť.

Algoritmus bude mať niekoľko globálnych polí. V prvom poli B sa bude ukladať informácia, či je otcovským blokom hrana alebo kružnica. Na začiatku budú v tomto poli samé nuly. Koreňu tam zostane nula, lebo nebude mať žiaden otcovský blok. Ak po skončení behu programu bude v poli hodnota

- $B[v] = 1$, znamená to, že otcovským blokom vrcholu v je hrana a má jednu otcovskú hranu.
- $B[v] = 2$, znamená to, že otcovským blokom vrcholu v je kružnica a má dve otcovské hrany.

Ďalej bude mať pre každý vrchol v dvojrozmerné pole W_v, pre jeho synov, ktorí ležia na synovských kružniciach. Po skončení behu programu v ňom bude jeden riadok predstavovať jednu synovskú kružnicu. Vrcholy v jednom riadku budú uložené v poradí, v akom ležia na prislúchajúcej kružnici. Posledným poľom, ktoré bude algoritmus mať pre každý vrchol v, je jednorozmerné pole V_v. V tomto poli budú po skončení behu programu uložení synovia v, ktorý ležia na synovských hranách. Na ich poradí nám v tomto prípade nezáleží. Obe tieto polia sú na začiatku prázdne.

Keďže náš algoritmus je upravené prehľadávanie do hĺbky, potrebuje si ešte uchovávať informáciu o tom, či bol daný vrchol už navštívený alebo nie. Toto pole sa bude volať P a hodnota

- $P[v] = 0$ označuje vrchol, ktorý ešte nebol navštívený

- $P[v] = 1$ označuje vrchol, ktorý už bol navštívený, ale ešte sme neprezreli všetkých jeho susedov

- $P[v] = 2$ označuje vrchol, ktorý bol navštívený a už sme spracovali všetkých jeho susedov

Na začiatku je pole P iniciované samými nulami.

Zakoreňovanie kaktusu vo vrchole bude rekurzívna funkcia. Táto funkcia dostane ako parametre dve čísla vrcholov v a u. Tieto dve čísla budú znamenať, že ideme zakoreniť kaktus pod vrcholom v, pričom sme vrchol v objavili z vrcholu u. Táto funkcia môže vrátiť rôzne návratové hodnoty. Ak vráti -1, znamená to, že otcovským blokom vrcholu v je hrana. Ak vráti číslo $z \in \{0, 1, \ldots, n\}$, otcovským blokom vrcholu v je kružnica s otcovským vrcholom z.

Prehľadávanie začíname vo vrchole r. Hranu, ktorou vrchol bol naposledy vrchol opustený, voláme odchádzajúca hrana. Hranu, ktorou bol vrchol objavený, voláme objaviteľská hrana. Z každého objaveného a nespracovaného vrcholu ide práve jedna odchádzajúca hrana.

Ak algoritmus spracúva vrchol, ktorý je objavený a stále existujú hrany, ktoré ešte neboli preskúmané, vyberie si jednu z nich a určí ju za novú odchádzajúcu hranu. Môže sa stať, že takýmto spôsobom sa navštívi vrchol v, ktorý už bol objavený, ale nebol dokončený. V tomto prípade bola objavená kružnica, ktorá má otcovský vrchol v. V tomto prípade sa len vráti späť návratová hodnota v a v poli W_v sa vyrobí miesto na novú kružnicu. Musí sa posielať aj informácia o tom, ktorý vrchol je otcovským vrcholom kružnice, aby sa vedelo, kde túto správu zastaviť. Ak príde späť k vrcholu v, potom sú spracované všetky vrcholy na tejto kružnici. Ak sa objaví nový vrchol, len sa predĺži cesta a pokračuje sa ďalej v prehľadávaní.

Predpokladajme, že sa na proces pozeráme z pohľadu vrcholu v, ktorého objaviteľská hrana je (u, v). Postupne, ako sa vracajú výsledky od jeho jednotlivých susedov, mení si informáciu o otcovskom bloku. Táto informácia sa bude ukladať do premennej b. Najskôr predpokladáme, že jeho otcovský blok je hrana. Premenná b budeme mať hodnotu -1. Ak však príde od niektorého suseda informácia, že jeho sused leží na kružnici s otcovským vrcholom w taká, že $w \neq v$, tak hodnota premennej b sa zmení na w. Takáto informácia môže prísť iba jedna. Vrchol v teraz vie, že jeho otcovským blokom je kružnica s otcovským vrcholom w.

Keď sa prehľadajú všetci susedia, uložia sa do globálny polí všetky potrebné informácie.

- Ak $b = -1$, tak sa nenašiel sused ležiaci na kružnici, ktorá nemá otcovský vrchol v. v je teda synom u ležiaci na synovskej hrane. Vrchol v sa pridá do poľa V_u. V poli B bude hodnota $B[v] = 1$.

- Ak $b = w$, tak sa našiel sused ležiaci na kružnici s iným otcom ako v. v je synom

vrcholu w ležiaci na synovskej kružnici. Vrchol v sa pridá na koniec posledného riadku poľa W_w. V poli B bude hodnota $B[v] = 2$.

Následne sa po objaviteľskej hrane vráti informácia o otcovskom bloku práve dokončeného vrcholu. Je to číslo rovné hodnote premennej b. Keď sa takto prehľadá celý kaktus, budú vo vyššie popísaných globálnych poliach uložené správne informácie.

Pseudokód prehľadávania môžete vidieť nižšie. Funkciu ZAKOREŇ stačí zavolať napríklad s parametrami r a -1. Druhý parameter v tomto prípade nedáva zmysel, lebo vrchol r ako jediný nemá objaviteľskú hranu. Stačí tam teda dať číslo, ktoré určite nie je číslom vrcholu v kaktuse.

```
1:  function ZAKOREŇ(v, u)
2:      if P[v] = 1 then
3:          W_v.append([])
4:          return v
5:      P[v] ← 1
6:      b ← −1
7:      for každého suseda s vrcholu v, s ≠ u do
8:          if P[s] ≠ 2 then
9:              h ← ZAKOREŇ(s, v)
10:             if  h ≠ −1 and h ≠ v then
11:                 b ← h
12:     P[v] ← 2
13:     if b = −1 then
14:         V_u.append(v)
15:         B[v] = 1
16:     else
17:         W_b[posledná].append(v)
18:         B[v] = 2
19:     return b
```

1.2.2 Analýza algoritmu

Zostáva ešte ukázať, že vyššie popísaný algoritmus je korektný a efektívny. Skôr ako sa pustíme do tohto dôkazu, dokážeme si pomocnú lemu.

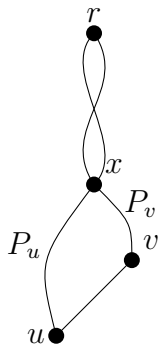
Lema 1.2.1 *Otcom bloku B je vrchol v práve vtedy, keď bol pri prehľadávaní do hĺbky objavený ako prvý zo všetkých vrcholov bloku B.*

Dôkaz. Musíme ukázať dve implikácie. Najprv ukážeme, že ak vrchol v je otcom bloku, potom bol navštívený ako prvý. Budeme to ukazovať sporom.

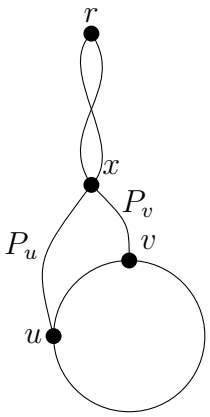
Nech vrchol v je otcom bloku B a nebol navštívený ako prvý. Nech bol navštívený ako prvý vrchol u. To znamená, že existujú dve cesty P_v a P_u vedúce z r do v resp. u. Cesta P_v je najkratšia cesta do v, lebo z definície je tento vrchol najbližšie ku koreňu. Zároveň táto cesta neobsahuje žiaden ďalší vrchol z tohto bloku. Inak by to bol spor s minimálnosťou vzdialenosti v od koreňa. Cesta P_u je cesta, ktorou sme objavili vrchol u. Táto cesta zjavne neobsahuje žiaden iný vrchol bloku B. Inak by u nebol prvý nájdený vrchol. Nech vrchol x je posledný spoločný vrchol týchto dvoch ciest. Taký určite existuje, lebo už vrchol r majú spoločný.

Ak blok B je hrana (u, v), potom hrany na ceste P_u z x do u, hrana (u, v) a hrany na ceste P_v z v do x tvoria kružnicu (Obr. 1.1a). To je ale v spore s tým, že hrana (u, v) je blok.

Ak blok B je kružnica, potom hrany na ceste P_u z x do u, hrany na kružnici B medzi vrcholmi u a v (vyberieme ľubovoľnú stranu kružnice) a hrany na ceste P_v z v do x tvoria kružnicu (Obr. 1.1b). Táto nový kružnica nie je hranovo disjuktná s kružnicou B. Čo je spor s definíciou kaktusu.



(a) Blok B je hrana



(b) Blok B je kružnica

Obr. 1.1

Zostáva dokázať opačnú implikáciu. Ak sme vrchol u objavili ako prvý, potom je vrchol u otcom bloku B. Opäť to dokážeme sporom. Predpokladáme, že platí, že u bol objavený ako prvý, ale skutočným otcom je v. Ďalší priebeh dôkazu je potom identický s dôkazom prvej implikácie. Aj tu sa dostaneme do sporu. Preto lema platí. □

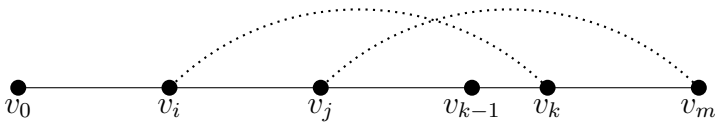
Teraz môžeme pristúpiť k dôkazu hlavnej vety.

Veta 1.2.1 *Vyššie popísaný algoritmus správne zakorení ľubovoľný kaktus. Jeho časová zložitosť je $O(n)$, kde n je počet vrcholov vstupného kaktusu.*

Dôkaz. Najprv ukážeme korektnosť. Môžeme si všimnúť, že vrcholy, ktoré sú objavené, ale nedokončené tvoria spolu so svojimi objaviteľskými hranami cestu. V tejto ceste sú všetky objavené a nedokončené vrcholy. Nech sú to vrcholy $v_0 = r, v_1, \ldots, v_m$ v takom poradí, že hrana (v_{i-1}, v_i) je objaviteľskou hranou v_i, pre $i \in 1, 2, \ldots, m$. Každý vrchol v_i má v sebe premennú b_i, v ktorej si uchováva informáciu o svojom otcovskom bloku. Hodnota premennej sa mení počas behu programu podľa výsledkov, ktoré prichádzajú od jeho susedov. Hodnota b_i môže byť -1, ak žiaden zo susedov zatiaľ nevrátil inú hodnotu ako -1 a v_i. Ďalšou možnosťou je, že je to ľubovoľné číslo od 0 po n. Premenná b_i mohla nadobudnúť takú hodnotu iba v prípade, že niektorý zo susedov ju vrátil. Nech vrchol v_k je posledný vrchol v poradí, ktorého $b_k \neq -1$. Poďme si rozobrať možnosti,ako sa táto situácia môže zmeniť pridaním ďalšej hrany do tejto postupnosti.

Najprv rozoberieme možnosť, že vrchol v_m ešte má susedov, ktorých môže prehľadať. Najjednoduchšou možnosťou v tomto prípade je, že navštívime suseda v_{m+1} vrcholu v_m, ktorý ešte nebol navštívený. V tomto prípade len vrchol v_{m+1} pribudne do postupnosti s hodnotou $b_{m+1} = -1$.

Zaujímavejšou možnosťou je, že vrchol v_{m+1} je niektorý z vrcholov $v_0, v_1, \ldots, v_{m-2}$. To znamená, že je jedným z objavených, ale ešte nie dokončených vrcholov. Nech je to vrchol v_j. Musí platiť, že $j \geq k$. Inými slovami, v_j je na ceste z nedokončených vrcholov z koreňa ďalej ako v_k. Ak by to tak nebolo, potom vrchol v_k dostane od dvoch rôznych susedov informáciu o tvorbe kružnice, ktorej otcovským vrcholom nie je v_k. V tomto prípade ale hrana (v_{k-1}, v_k) leží na viacerých kružniciach (Obr. 1.2). To je spor s definíciou kaktusu. Preto sa nikdy nemôže stať, že jeden vrchol dostane od dvoch rôznych susedov inú hodnotu ako -1 a jeho vlastné číslo.



Obr. 1.2

Zjavne vrcholy $v_j, v_{j+1}, \ldots, v_m$ tvoria kružnicu. Podľa predchádzajúcej lemy vieme, že otcovským vrcholom tejto kružnice je vrchol v_j. Algoritmus teda správne pošle vrcholu v_m informáciu, že v_m leží na kružnici s otcovským vrcholom v_j. Vo vrchole v_m sa táto informácia uloží do b_m. Nové k teraz bude m.

Zostávajú už len možnosti, keď vrchol v_m nemá suseda, ktorého ešte treba prehľadať. Podľa algoritmu nastal čas uložiť tento vrchol na správne miesto do globálnych polí a poslať o tom informáciu ďalej. Rozoberieme niekoľko možností podľa hodnoty b_m.

- $b_m = -1$: Táto možnosť mohla nastať iba v prípade, keď sa nenašiel žiaden sused, ktorý

by vrcholu v_m poslal inú informáciu ako -1 a v_m. Ak by vrchol v_m mal otcovský blok kružnicu a nie hranu, potom jeden z jeho susedov s ním buď leží na tejto kružnici alebo je jej otcom. V oboch prípadoch by tento vrchol o tom dostal informáciu, čo sa ale nestalo. Potom zjavne otcovským blokom vrcholu v_m je objaviteľská hrana a algoritmus správne zaradí tento vrchol medzi synov vrcholu v_{m-1}, ktorí ležia na synovských hranách.

- $b_m = v_j \neq -1$: Vo vrchole v_m môže byť uložená takáto informácia iba vtedy, keď jeden z jeho susedov vrátil inú hodnotu ako -1 a v_m. To znamená, že vrchol v_m spolu s jeho susedom majú otcovský blok kružnicu s otcom v_j. K vrcholu v_m sa táto informácia mohla dostať len tak, že vznikla pričinením vrcholu v_j a následne bola postupne posúvaná späť po objaviteľských hranách, ktoré tvoria túto kružnicu, až k vrcholu v_m. Všetky dokončené vrcholy sú už zapísané na správnom mieste, lebo v jednom čase môžeme tvoriť len jednu synovskú kružnicu vrcholu v_j. Vrchol v_m sa zapíše za tieto vrcholy a pošle informáciu ďalej po svojej objaviteľskej hrane, ktorá taktiež patrí do tejto kružnice. Tento postup zaručuje správne poradie vrcholov zapísaných v jednotlivých poliach.

Následne vrchol v_m bude medzi dokončenými vrcholmi. Preto už nebude patriť do cesty z objaviteľských hrán.

V každom kroku prejde algoritmus po nejakej hrane. Po každej hrane prejde najviac dvakrát. Preto zaručene skončí, lebo hrán je len konečne veľa. Zároveň to znamená, že časová zložitosť tohto algoritmu je lineárne závislá od počtu hrán v kaktuse. Počet hrán v kaktuse s n vrcholmi môžeme z hora odhadnúť výrazom

$$|E(G)| \leq (n-1) + \frac{(n-1)}{2} = \frac{3n-3}{2}$$

Najväčší počet hrán má kaktus, ktorý je zložený z kružníc dĺžky tri. Takýto kaktus môžeme vytvoriť zo stromu tak, že dve susedné hrany spojíme novou hranou do trojuholníka a pokračujeme, kým sa dajú pridávať hrany. Na dve hrany stromu pridáme jednu novú hranu. Keď k počtu hrán stromu pridáme ešte polovicu tohto počtu dostaneme horné ohraničenie počtu hrán v kaktuse. Vidíme, že $|E(G)| = O(|V(G)|) = O(n)$. Preto celý zakoreňovací algoritmus má časovú zložitosť $O(n)$. □

Ukázali sme, že náš algoritmus na zakorenenie kaktusu pracuje správne a efektívne. Teraz sa môžeme venovať samotným výsledkom.

Kapitola 2

Polynomiálny algoritmus pre cyklové stromy

V tejto kapitole predstavíme polynomiálny algoritmus pre triedu cyklových stromov, ktorý sme vymysleli. Vstupom je cyklový strom G a nezáporné celé číslo k. Algoritmus v polynomiálnom čase vzhľadom na počet vrcholov grafu G a číslo k rozhodne, či existuje k-$L(2,1)$-ofarbenie cyklového stromu G.

Náš algoritmus je rozšírením prvého algoritmu pre stromy, ktorý publikovali autori Chang a Kuo v článku [6]. V tomto algoritme sa pracovalo so zakoreneným stromom. Každý vrchol okrem koreňa mal určenú hranu, ktorá viedla do otca. Hlavnou myšlienkou algoritmu bolo pre každý vrchol v vypočítať všetky prípustné farby pre vrcholy na otcovskej hrane (u, v). Prípustné sú také, že sa podstrom visiaci pod vrcholom v dá ofarbiť korešpondujúc s farbami hrany (u, v).

Menší technický problém bol iba s koreňom, lebo koreň nemá otca. To sa vyriešilo tak, že sa strom na vstupe zakorenil v niektorom liste. Tento list sa následne odstránil a koreňom sa stal jeho jediný syn. Tak sa docielilo, že každý vrchol v novom grafe má práve jedného otca.

Na konci už len stačilo pozrieť, či vypočítaná množina dvojíc farieb pre koreň nie je prázdna. Ak bola prázdna, tak sa strom nedal ofarbiť. A naopak, ak sa v nej niečo nachádzalo, tak sa dal ofarbiť.

V nasledujúcej časti popíšeme náš algoritmus, ktorý je založený na podobnej myšlienke.

2.1 Algoritmus

Bude sa postupovať rovnako ako v algoritme pre stromy. Cyklový strom G sa zakorení spôsobom popísaným v 1.2.1. Pre každý vrchol chceme vypočítať množinu prípustných farieb pre vrcholy na otcovských hranách. V našom prípade môže byť otcovských hrán viac.

Na začiatku sa otestuje, či je na vstupe graf s maximálnym stupňom najviac dva. Ak je to tak, tak to môže byť len cesta alebo kružnica. Pre tieto prípady platia nasledujúce výsledky, ktoré sa dajú jednoducho dokázať:

Graf G	$\lambda(G)$
cesta dĺžky 1	2
cesta dĺžky 2 alebo 3	3
cesta dĺžky aspoň 4	4
kružnica ľubovoľnej dĺžky	4

Pre tieto cyklové stromy sa dá odpoveď zistiť len na základe maximálneho stupňa, počtu hrán a počtu vrcholov.

Od teraz môžeme predpokladať, že maximálny stupeň cyklového stromu G je aspoň tri. V tomto prípade už treba použiť zložitejší algoritmus. Podobne ako pri stromoch, aj tu sa chceme správať ku všetkým vrcholom grafu rovnako, čo sa týka otcovských hrán. Musí sa preto spraviť jeden technický krok.

Keďže maximálny stupeň vstupného cyklového stromu je aspoň tri, tak existuje vrchol, ktorý leží aspoň v dvoch rôznych blokoch. Cyklový strom sa preto skladá z viacerých blokov. Dá sa ukázať, že v ňom existuje aspoň jeden listový blok. Jeden taký listový blok sa vyberie. Uprednostňuje sa hrana pred kružnicou, pretože kružnice na konci vyžadujú prácu navyše. Nech je vybratým listovým blokom blok B, ktorého jedinou artikuláciou je vrchol r. Z cyklového stromu G sa teraz vyrobí nový cyklový strom G' tak, že sa z G odstránia všetky vrcholy bloku B okrem vrcholu r. Následne sa nový cyklový strom G' zakorení v r.

V prípade, že blok B bola hrana, tak vrchol r bude mať jednu otcovskú hranu. V opačnom prípade bude mať dve otcovské hrany.

Teraz budeme zisťovať, či vieme ofarbiť graf G' spolu s otcovskými hranami vrcholu r. Vrcholy sa rozdelia do dvoch množín podľa počtu otcovských hrán.

$$\begin{aligned} V_1 &= \{v \in V(G') \mid v \text{ má jednu otcovskú hranu}\} \\ V_2 &= \{v \in V(G') \mid v \text{ má dve otcovské hrany}\} \end{aligned}$$

Zjavne platí $V_1 \cup V_2 = V(G')$ a $V_1 \cap V_2 = \emptyset$.

Pre každý vrchol v grafu G' chceme vypočítať množinu F_v možných ofarbení otcovských hrán. V množine F_v budú také dvojice (resp. trojice) farieb, ktoré môžme dať vrcholom na otcovskej hrane (resp. hranách) vrcholu v a podkaktus pod vrcholom v sa dá ofarbiť korešpondujúc s týmito farbami.

Zjavne pre list v je jednoduché určiť množinu F_v. Rozlišujú sa dva prípady:

- $v \in V_1$: $F_v = \{(a, b) \mid a, b \in \{0, \ldots, k\}, |a - b| \geq 2\}$
- $v \in V_2$: $F_v = \{(a, b, c) \mid a, b, c \in \{0, \ldots, k\}, |a - b| \geq 2, |b - c| \geq 2, |a - c| \geq 1\}$

Predpokladajme, že vrchol v nie je list a všetci jeho synovia majú už množiny F_{v_i} vypočítané. Rozlišujeme tri prípady, ako môže vyzerať okolie vrcholu v.

1. Vrchol v neleží na kružnici. To znamená, že patrí do množiny V_1 a rovnako do nej patria aj všetci jeho synovia.

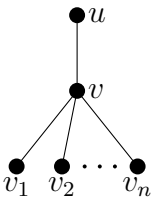
2. Vrchol v leží na kružnici a táto kružnica je jeho synovská kružnica. V tomto prípade $v \in V_1$. Niektorí jeho synovia patria do množiny V_2 a zvyšní synovia patria do množiny V_1.

3. Vrchol v leží na kružnici, ktorá je jeho otcovskou kružnicou. V tomto prípade $v \in V_2$ a všetci jeho synovia patria do V_1.

Ku každému prípadu pristupuje algoritmus trochu inak. V nasledujúcej časti ukážeme ako.

- $v \in V_1$ a všetci jeho synovia patria tiež do V_1:

  Vrchol v má len jednu otcovskú hranu (u, v) a všetky jeho synovské bloky sú hrany. Nech $v_1, v_2, \ldots, v_n$ sú jeho synovia (Obr. 2.1).



Obr. 2.1: v neleží na kružnici

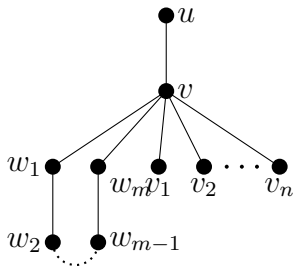
  Všetky množiny F_{v_i} obsahujú iba dvojice. Z týchto čiastočných výsledkov algoritmus vytvorí množinu F_v. F_v bude obsahovať iba dvojice farieb, a to konkrétne možné ofarbenia hrany (u, v). Každú dvojicu farieb $a, b \in \{0, 1, \ldots, k\}$ takú, že $|a - b| \geq 2$, treba otestovať. To znamená zistiť, či je možné priradiť vrcholu u farbu a a vrcholu v farbu b tak, aby sa podkaktus pod vrcholom v dal ofarbiť. Definujú sa množiny W_i pre $i \in \{1, 2, \ldots, n\}$:

$$W_i = \{c \mid (b, c) \in F_{v_i},\ c \neq a\}$$

  Teraz stačí otestovať, či existujú reprezentanti $c_1, c_2, \ldots, c_n$ týchto tried takí, že $c_i \in W_i$ a $c_i \neq c_j$, ak $i \neq j$. To sa dá overiť hľadaním maximálneho párenia na špeciálnom bipartitnom grafe, ktorého jedna partícia sú vrcholy $v_1, v_2, \ldots v_n$ a druhá partícia sú farby $0, 1, \ldots, k$. Medzi v_i a j je hrana ak $j \in W_i$. Ak v tomto grafe existuje párenie s mohutnosťou n, potom dvojica (a, b) patrí do množiny F_v.

- $v \in V_1$, časť jeho synov patrí do V_2 a časť do V_1:

  Vrchol v má len jednu otcovskú hranu (u, v). Jeden jeho synovský blok je kružnica a zvyšné sú hrany. Nech $w_1, w_2, \ldots w_m$ sú synovia v na kružnici v poradí, v akom sú umiestnené na kružnici. Vrcholy w_1 a w_m sú incidentné s v. Ďalej $v_1, v_2, \ldots, v_n$ sú synovia v na hranách (Obr. 2.2).



Obr. 2.2: v leží na kružnici a jeho otcovský blok je hrana

Najprv treba zistiť, aké farby môžu mať vrcholy w_1, v a w_m, aby sa podkaktus pod touto kružnicou dal ofarbiť. Množiny F_{w_i} pre $i \in \{1, 2, \ldots, m\}$ už sú vypočítané. Všetky tieto množiny obsahujú trojice farieb, ktoré predstavujú možné ofarbenia otcovských hrán vrcholov $w_1, w_2, \ldots, w_m$ také, že podkaktusy pod týmito vrcholmi sa dajú ofarbiť.

Vyrobí sa množinu:

$$X_1 = \{(a, b, b, c) \mid (a, b, c) \in F_{w_1}\}$$

Prvé dve čísla znamenajú ofarbenie hrany (v, w_1) a druhé dve znamenajú ofarbenie hrany (w_1, w_2). Postupne sa vyrábajú ďalšie X_i pre $i \in 2, 3, \ldots m$:

$$X_i = \{(a, b, d, e) \mid (a, b, c, d) \in X_{i-1}, (c, d, e) \in F_{w_i}\}$$

Jednoducho povedané, stále sa predlžuje už ofarbená časť kružnice a ukladajú sa do medzivýsledkov len farby začiatočných dvoch vrcholov a koncových dvoch vrcholov. Štvorica farieb $(a, b, c, d) \in X_i$ symbolizuje všetky $L(2, 1)$-ofarbenia vrcholov $v, w_1, w_2, \ldots, w_{i+1}$ také, že

$$\begin{aligned} f(v) &= a \\ f(w_1) &= b \\ f(w_i) &= c \\ f(w_{i+1}) &= d \end{aligned}$$

Ak $i = m$, potom $w_{i+1} = v$. Nie je podstatné, či sa k jednej štvorici dá prísť viacerými spôsobmi.

Hľadaná množina je zjavne:

$$Y = \{(b, a, c) \mid (a, b, c, a) \in X_d,\ b \neq c\}$$

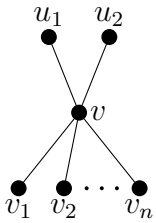
Aby sa zistilo, ktoré dvojice patria do množiny F_v, musia sa opäť otestovať všetky dvojice farieb $a, b \in \{0, 1, \ldots, k\}$ také, že $|a - b| \geq 2$. Aby sa zistilo, či (a, b) je vhodná dvojica farieb pre vrcholy u a v, musí sa pre všetky kompatibilné trojice farieb $(c, b, d) \in Y$ také, že $c \neq d \neq a$, zistiť, či sa dajú ofarbiť ostatní synovia v. Podobne ako v prvom prípade, sa vyrobia množiny W_i, $i \in \{1, 2, \ldots, n\}$ pre vrcholy $v_1, v_2, \ldots, v_n$.

$$W_i = \{c \mid (b, c) \in F_{v_i},\ c \neq a \neq b \neq d\}$$

Úplne rovnakým spôsobom ako predtým sa následne vyrobí bipartitný graf a zistí sa, či v ňom existuje maximálne párenie s mohutnosťou n. Ak pre aspoň jednu kompatibilnú trojicu z Y existuje také párenie, tak dvojica farieb (a, b) patrí do F_v.

- $v \in V_2$ a všetci jeho synovia patria do V_1:

  V tomto prípade má vrchol v dve otcovské hrany (u_1, v) a (u_2, v) a všetky jeho synovské bloky sú hrany. Nech jeho synovia sú $v_1, v_2, \ldots, v_n$ (Obr. 2.3).



Obr. 2.3: v leží na kružnici a jeho otcovský blok je kružnica

  Množina F_v bude teraz obsahovať trojice prípustných farieb (a, b, c), kde a je farba pre u_1, b je farba pre v a c je farba pre u_2. Pre každú skúšanú trojicu musí preto platiť: $|a - b| \geq 2$, $|b - c| \geq 2$ a $a \neq c$.

  Pri testovaní trojice farieb (a, b, c) sa bude postupovať rovnako ako v prvom prípade. Množiny W_i pre $i \in \{1, 2, \ldots, n\}$ sa vytvárajú nasledovne:

$$W_i = \{d \mid (b, d) \in F_{v_i},\ d \neq a \neq c\}$$

  Opäť sa vyrobí bipartitný graf rovnakým spôsobom ako v prvom prípade a zistí sa, či existuje maximálne párenie s mohutnosťou n. Ak áno, trojica (a, b, c) patrí do F_v.

Ak sa na začiatku z grafu G odstránila hrana, potom stačí pozrieť, či je F_r prázdna množina. Ak $F_r = \emptyset$, potom sa zjavne nedá kaktus G ofarbiť. Ak F_r obsahuje aspoň jednu dvojicu, potom sa dá odobratá hrana ofarbiť danou dvojicou a zvyšok grafu sa dá tiež ofarbiť, lebo tak sa vytvárali množiny F_v.

Ak sa z grafu G odstránila kružnica, musí sa ešte zistiť, či sa dá ofarbiť tak, že toto farbenie korešponduje s niektorým vypočítaným farbením otcovských hrán r. Nech sa odstránili

vrcholy $w_1, w_2, \ldots, w_m$. Určia sa množiny F_{w_i} pre $i \in \{1, 2, \ldots, m\}$ tak, ako v prípravnej fáze pre listy na kružniciach.

$$F_{w_i} = \{(a, b, c) \mid |a - b| \geq 2,\ |b - c| \geq 2,\ a \neq c\}$$

Následne sa vyrobí množina Y úplne rovnakým spôsobom ako v druhom prípade popísanom vyššie. Nakoniec sa zistí, či $Y \cap F_v = \emptyset$. Ak áno, potom neexistuje k-$L(2,1)$-farbenie cyklového stromu G. V opačnom prípade sa dá $L(2,1)$-ofarbiť celý graf G a to znamená, že $\lambda(G) \leq k$.

2.2 Analýza algoritmu

Už nám stačí len ukázať, že náš vymyslený algoritmus je korektný a polynomiálny. Formulujeme to ako dve vety.

Veta 2.2.1 *Algoritmus popísaný v sekcií 2.1 správne rozhodne pre daný cyklový strom G a číslo k, či platí $\lambda(G) \leq k$.*

Dôkaz. Najprv musíme dokázať, že sme množiny F_v vypočítali správne. To znamená, že sú to práve tie dvojice alebo trojice povolených farieb, ktoré môžeme priradiť otcovským hranám vrcholu v a podkaktus pod vrcholom týmto vrcholom sa dá ofarbiť. Označme si takú skutočnú množinu E_v pre vrchol v. Chceme teda ukázať, že pre každý $v \in V(G')$ platí

$$E_v = F_v$$

Ukážeme to matematickou indukciou vzhľadom na zložitosť podgrafu pod vrcholom.

V Báze indukcie ukážeme, že tvrdenie platí pre listy. Každý typ listu rozoberieme zvlášť. Nech v je list so stupňom jedna. Musíme ukázať dve inklúzie.

- $F_v \subseteq E_v$: Potrebujeme ukázať, že všetky dvojice v F_v sú aj v E_v. To je zjavne pravda. Akonáhle dvojica farieb (a, b) spĺňa, $|a - b| \geq 2$, tak potom otcovskú hranu (u, v) vrcholu v môžeme touto dvojicou ofarbiť, lebo pod vrcholom v už neležia žiadne vrcholy.

- $F_v \supseteq E_v$: Chceme ukázať, že všetky dvojice v E_v sú aj v F_v. Keď si zoberieme ľubovoľnú dvojicu farieb $(a, b) \in E_v$, tak musí spĺňať podmienky $L(2,1)$-farbenia, čiže $|a - b| \geq 2$. Keďže v F_v sa nachádzajú všetky dvojice farieb, ktoré spĺňajú túto podmienku, tak sa tam musí nachádzať aj dvojica (a, b).

Nech teraz v je list so stupňom dva. Opäť dokážeme dve inklúzie:

- $F_v \subseteq E_v$: Aby sme to dokázali ukážeme, že všetky trojice v F_v sú aj v E_v. Nech trojica $(a, b, c) \in F_v$. Z definície F_v musí platiť $|a - b| \geq 2$, $|b - c| \geq 2$ a $a \neq c$. Zjavne môžeme potom vrcholom u_1, v a u_2 ležiacim na otcovských hranách vrcholu v priradiť takéto farby, lebo pod vrcholom v už neležia žiadne ďalšie vrcholy. Preto je táto trojica farieb aj v množine E_v.

- $F_v \supseteq E_v$: Chceme ukázať, že všetky trojice v E_v sú aj v F_v. Podobne ako v prípade listu na hrane, si zoberieme ľubovoľnú trojicu farieb $(a, b, c) \in E_v$. Táto trojica musí spĺňať podmienky $L(2, 1)$-farbenia, čiže $|a - b| \geq 2$, $|b - c| \geq 2$ a $a \neq c$. Keďže v F_v sa nachádzajú všetky trojice farieb, ktoré spĺňajú túto podmienku, tak sa tam musí nachádzať aj trojica (a, b, c).

Zjavne tvrdenie platí pre listy. Tým sme ukončili dôkaz bázy indukcie.

Pokračujeme ďalej indukčným krokom. Nech vrchol v nie je list a pre všetkých jeho synov tvrdenie platí. Ukážeme, že potom platí aj pre v. Budeme opäť rozlišovať tri prípady ako môže vyzerať okolie vrcholu v.

Najprv rozoberieme prípad, keď vrchol v neleží na kružnici. Jeho otcovský blok aj synovské bloky sú hrany. Budeme používať rovnaké označenie ako v popise algoritmu. Vrcholy $v_1, v_2, \ldots, v_n$ sú synovia v a vrchol u je jeho otec. Z indukčného predpokladu vieme, že platí $F_{v_i} = E_{v_i}$ pre každé $i \in \{1, 2, \ldots, n\}$.

- $F_v \subseteq E_v$: Ukážeme, že ľubovoľná dvojica $(a, b) \in F_v$ patrí aj do E_v. Nech $c_1, c_2, \ldots, c_n$ sú farby, ktoré sme vybrali vrcholom $v_1, v_2, \ldots, v_n$ pomocou nájdenia maximálneho párenia pri pridávaní dvojice (a, b) do F_v. To bolo možné iba preto, lebo $(b, c_i) \in F_{v_i}$ a zároveň $a \neq c_i$ pre $i \in \{1, 2, \ldots, n\}$. Z predpokladu vieme, že $(b, c_i) \in E_{v_i}$. Keď teda dáme vrcholom $u, v, v_1, v_2, \ldots, v_n$ postupne farby $a, b, c_1, c_2, \ldots, c_n$, tak z indukčného predpokladu vieme, že zvyšok podkaktusu vieme ofarbiť tak, že toto farbenie korešponduje s už vybratými farbami. Preto dvojica (a, b) patrí aj do E_v.

- $F_v \supseteq E_v$: V tomto prípade ukážeme, že ľubovoľná dvojica $(a, b) \in E_v$ patrí aj do F_v. Zjavne musí platiť $|a - b| \geq 2$. Ďalej musia existovať farby $c_1, c_2, \ldots, c_n$ pre vrcholy $v_1, v_2, \ldots, v_n$, ktoré sú navzájom rôzne a pre každé $i \in \{1, 2, \ldots, n\}$ platí $|c_i - b| \geq 2$ a $c_i \neq a$, lebo podmienkou príslušnosti dvojice farieb do (a, b) do E_v je, že sa podkaktus pod v dá ofarbiť. Keďže podľa indukčného predpokladu $(b, c_i) \in F_{v_i}$, tak $c_i \in W_i$ pre každé $i \in \{1, 2, \ldots, n\}$. Potom ale platí, že vo vytvorenom bipartitnom grafe existuje maximálne párenie s mohutnosťou n. Preto aj $(a, b) \in F_v$.

Ďalším prípadom je, že otcovským blokom vrcholu v je hrana a jeden z jeho synovských blokov je kružnica. Nech u je jeho otec, $w_1, w_2, \ldots, w_m$ sú jeho synovia na synovskej kružnici a $v_1, v_2, \ldots v_n$ sú jeho synovia na synovských hranách. Pre všetkých synov tvrdenie platí,

čiže $F_{w_i} = E_{w_i}$ pre $i \in \{1, 2, \ldots, m\}$ a $F_{v_i} = E_{v_i}$ pre $i \in \{1, 2, \ldots, n\}$. Opäť ukážeme dôkaz dvoch inklúzií.

- $F_v \subseteq E_v$: Nech $(a, b) \in F_v$. Ukážeme, že aj $(a, b) \in E_v$. Aby sa (a, b) dostalo do F_v musela existovať taká trojica $(c, b, d) \in Y$, že $a \neq c \neq d$ a bipartitný graf, ktorý vznikol z množín W_i pre $i \in \{1, 2, \ldots, n\}$ mal maximálne párenie s mohutnosťou n. Poslednou podmienkou je, že $|a - b| \geq 2$.Najprv musíme ukázať, že keď $(c, b, d) \in Y$, potom existuje ofarbenie kružnice a celého podkaktusu pod touto kružnicou. Aby sa (c, b, d) mohlo dostať do Y, tak v X_m musela byť štvorica (b, c, d, b), lebo tak bola definovaná množina Y. Ukážeme že existujú farby $d_1, d_2, \ldots, d_m, d_{m+1}$, ktoré môžeme priradiť vrcholom $w_1, w_2, \ldots, w_m, v$ také, že $d_1 = c$, $d_m = d$ a $d_{m+1} = b$. Budeme postupovať spätne cez proces vytvárania množín X_i pre $i \in \{1, 2, \ldots, m\}$.

  Aby mohla vzniknúť štvorica farieb $(b, c, d_i, d_{i+1}) \in X_i$, musela existovať nejaká farba d_{i-1} taká,že $(d_{i-1}, d_i, d_{i+1}) \in F_{w_i}$ a $(b, c, d_{i-1}, d_i) \in X_{i-1}$ pre i postupne $m, m-1, \ldots, 2$. Z toho vyplýva, že v X_1 musela byť štvorica $(b, d_1, d_1, d_2) = (b, c, c, d_2)$. Táto štvorica mohla vzniknúť jedine tak, že existovala trojica $(b, c, d_2) \in F_{w_1}$. Z toho vyplýva, že hľadané farby pre vrcholy $w_1, w_2, \ldots, w_m$ a v korešpondujúce s trojicou farieb $(c, b, d) \in Y$ naozaj existujú a z indukčného predpokladu vyplýva, že sa dá ofarbiť celý podkaktus pod touto kružnicou tak, že sa rozšíri nájdené farbenie kružnice.

  Podobným argumentom ako v prvom prípade ukážeme, že aj pre vrcholy $v_1, v_2, \ldots, v_n$ vieme nájsť farby, $c_1, c_2, \ldots, c_n$ také, že korešpondujú s farbami a, b, c a d. Množiny možných farieb pre vrcholy $v_1, v_2, \ldots v_n$ sme vyberali tak, aby boli splnené podmienky $L(2, 1)$-farbenia. Vytvorený bipartitný graf mal párenie hodnosti n. Preto na základe indukčného predpokladu vieme, že celý podkaktus pod vrcholom v sa dá ofarbiť korešpondujúc s farbami (a, b). Preto platí $(a, b) \in E_v$.

- $F_v \supseteq E_v$: Nech teraz $(a, b) \in E_v$. Ukážeme, že aj $(a, b) \in F_v$. Keďže $(a, b) \in E_v$, potom existujú farby $d_1, d_2, \ldots, d_m$ pre vrcholy $w_1, w_2, \ldots, w_m$ a farby $c_1, c_2, \ldots, c_n$ pre vrcholy $v_1, v_2, \ldots, v_n$ také, že sa podkaktusy pod týmito vrcholmi dajú ofarbiť. Zjavne

$$\begin{aligned} (b, d_1, d_2) &\in F_{w_1} \\ (d_{i-1}, d_i, d_{i+1}) &\in F_{w_i}, \quad i \in \{2, 3, \ldots, m-1\} \\ (d_{m-1}, d_m, b) &\in F_{w_m} \\ (b, c_i) &\in F_{v_i}, \quad i \in \{1, 2 \ldots, n\} \end{aligned}$$

  Potom ale musí platiť

$$(b, d_1, d_1, d_2) \in X_1$$

$$\begin{aligned}(b, d_1, d_i, d_{i+1}) &\in X_i, \;\; i \in \{2, 3, \ldots, m-1\} \\ (b, d_1, d_m, b) &\in X_m\end{aligned}$$

Preto je v množine Y aj trojica (d_1, b, d_m). Keďže $(a, b) \in E_v$ tak zjavne platí $a \neq d_1 \neq d_m$. Tak isto d_1 je určite rôzne od všetkých farieb c_i a rovnaké tvrdenie platí aj pre d_m. Keď teda vytvoríme bipartitný graf spôsobom opísaným v algoritme, tak bude mať párenie s mohutnosťou n. Napríklad to môže byť párenie dané farbami $c_1, c_2, \ldots, c_n$. Preto náš algoritmus zaradí (a, b) do F_v.

Posledným prípadom zostáva, keď otcovským blokom vrcholu v je kružnica a všetky jeho synovské bloky sú hrany. Nech vrcholy na otcovských hranách sú u_1 a u_2 a jeho synmi sú $v_1, v_2, \ldots v_m$. Jedine v tomto prípade budeme pracovať s trojicami farieb. Opäť ukážeme dve inklúzie.

- $F_v \subseteq E_v$: Nech $(a, b, c) \in F_v$. Ukážeme, že $(a, b, c) \in E_v$. Zjavne musel existovať bipartitný graf vytvorený z výsledkov v F_{v_i} pre $i \in \{1, 2, \ldots, n\}$, ktorý mal párenie s mohutnosťou n. Toto párenie určuje farby $c_1, c_2, \ldots, c_n$ pre synov vrcholu v také, že spĺňaú podmienky $L(2,1)$-farbenia. Keďže F_{v_i} obsahuje dvojicu (b, c_i), tak z indukčného predpokladu vieme, že podgraf pod v_i sa dá ofarbiť. To platí pre všetky $i \in \{1, 2, \ldots, n\}$. Z toho vyplýva, že aj $(a, b, c) \in E_v$.

- $F_v \supseteq E_v$: Nech $(a, b, c) \in E_v$. Ukážeme, že $(a, b, c) \in F_v$. Z predpokladu vieme, že ak dáme vrcholom u_1, v, u_2 farby a, b, c, tak zvyšok grafu pod vrcholom v vieme ofarbiť. Vieme tiež, že farby spĺňajú podmienky $L(2,1)$-farbenia. Vyberme si jedno farbenie podkaktusu pod v korešpondujúce s farbami a, b a c. Nech priradilo synom vrcholu v farby $c_1, c_2, \ldots, c_n$. Z indukčného predpokladu vieme, že $(b, c_i) \in F_{v_i}$ a tým pádom aj $c_i \in W_i$, pre $i \in \{1, 2, \ldots, n\}$, lebo pre každé i platí $c_i \neq a \neq c$. Preto v bipratitnom grafe, vytvorenom v tejto fáze algoritmu, bude párenie s mohutnosťou n. Preto bude trojica (a, b, c) zaradená, do F_v.

Tým sme dokázali, že sme množiny F_v vypočítali korektne. Zostáva rozobrať posledný krok algoritmu.

Najprv rozoberieme možnosť, že sme graf G' vytvorili odstránením hrany. Ak existuje dvojice $(a, b) \in F_r$, potom odstránenému vrcholu môžeme dať farbu a a koreňu farbu b. Podľa predchádzajúcej časti dôkazu vieme, že zvyšok grafu sa dá $L(2,1)$-ofarbiť. Ak sa tam žiadna taká dvojica nenachádza, potom je zjavné, že ho nevieme ofarbiť. Náš algoritmus dá v tomto prípade správny výsledok.

Zostáva možnosť, že sme odstraňovali kružnicu. V tomto prípade sme si ešte vyrábali množinu Y. Ako sme ukázali v druhom prípade v indukčnom kroku, takto vytvorená množina

naozaj obsahuje práve tie trojice farieb, ktoré môžu mať vrcholy u_1, r, u_2, aby sa dala celá táto kružnica ofarbiť. Ak v prieniku F_r a Y nie je žiadna trojica, potom žiadna trojica z F_r sa nedala doplniť na ofarbenie celej kružnice. Preto sa žiadne farbenie grafu G' (plus hrán $(r, u_1), (r,_u 2)$) nedalo rozšíriť na farbenie kaktusu G. Ak je naopak tento prienik neprázdny, potom existuje trojica farieb (a, b, c), že vrcholy u_1, r a u_2 môžu mať tieto farby lebo celý podkaktus pod r sá dá ofarbiť a zároveň tieto farby umožňujú dofarbiť aj zvyšné vrcholy odstránenej kružnice, preto aj v tomto prípade dá náš algoritmus správny výsledok. □

Veta 2.2.2 *Algoritmus popísaný v sekcií 2.1 má časovú zložitosť'*$O(|V(G)|k^{7.5})$ *pre vstupný cyklový strom* G *a číslo* k*.*

Dôkaz. Aby sme nemuseli všade písať $V(G)$ a $E(G)$, budeme namiesto toho používať V a E. Pozrime sa, koľko času zaberajú jednotlivé kroky algoritmu.

Algoritmus najprv testuje, či je cyklový strom na vstupe cesta alebo kružnica. To sa dá otestovať v čase $O(|E|)$, lebo stačí zistiť maximálny stupeň, celkový počet hrán a počet vrcholov. Keďže máme na vstupe cyklový strom, tak platí $|E| = O(|V|)$. Dokázali sme to v dôkaze vety 1.2.1. Testovanie má teda časovú zložitosť $O(|V|)$. Zistené stupne vrcholov si môžeme zapamätať, aby sme ich nemuseli zisťovať znovu. Takisto si zapamätáme vrchol s maximálnym stupňom.

Ak je na vstupe zložitejší graf, potom algoritmus hľadá listový blok. V prvom rade hľadá, či existuje vrchol so stupňom jedna. To zaberie maximálne $O(|V|)$ času. Ak taký list neexistuje, musí sa hľadať listová kružnica. To sa dá spraviť napríklad tak, že sa kaktus zakorení vo vrchole, ktorý nemá stupeň dva. To znamená, že to je určite artikulácia a patrí do viacerých blokov. Môže to byť napríklad vrchol s maximálnym stupňom, ktorý sme našli v prvej fáze. Spustí sa teda algoritmus z časti 1.2.1. Potom už len pre každý vrchol, ktorý má nejakú synovskú kružnicu otestujeme, či všetci jeho synovia na tejto kružnici majú stupeň dva. Keďže v štruktúre, vytvorenej algoritmom na zakoreňovanie kaktusu, je každý vrchol uložený najviac raz, tak táto operácia spolu so zakoreňovaním kaktusu má časovú zložitosť $O(|V|)$.

Keď sa nájde listový blok, potom odstránime všetky vrcholy listového bloku okrem vrcholu r, ktorý ako jediný patril do viacerých blokov. Následne sa tento zmenšený kaktus zakorení v tomto vrchole. Toto celé opäť zaberie čas $O(|V|)$. Podkaktus, ktorý vznikne označíme $G' = (V', E')$.

V ďalšej fáze algoritmus počíta množiny F_v. Pozrime sa na množstvo práce v jednotlivých vrcholoch.

1. Vrchol v neleží na kružnici:

   V tomto prípade pre vrchol v počítame iba množinu F_v. Do F_v ukladáme dvojice farieb z množiny $\{0, 1, \ldots, k\}$. Takých dvojíc farieb je $O(k^2)$ a pre každú z nich spúšťame

hľadanie maximálneho párenia na grafe s veľkosťou $O(\Delta + k)$, kde Δ je maximálny stupeň kaktusu G. Z pravidla bude číslo $k > \Delta$. Keby to tak nebolo, tak môžeme rovno odpovedať, že sa tento graf nedá ofarbiť. Vysvetlenie nájdete v ďalšej kapitole. Veľkosť bipartitného grafu teda bude $O(k)$. Algoritmus na hľadanie párenia v grafe s n vrcholmi má časovú zložitosť $O(n^{2.5})$ [15]. Spolu teda dostávame časovú zložitosť $O(k^{4.5})$

2. Vrchol v leží na kružnici, ktorá je jeho synovská:

   V algoritme hľadáme pri tomto vrchole množinu Y. Túto prácu však započítame vrcholom, ktoré na nej ležia a nie vrcholu v. Predpokladajme, že už máme množinu trojíc farieb Y. Aj pre tento vrchol skúšame všetky vhodné dvojice farieb z množiny $\{0, 1, \ldots, k\}$, ktorých je $O(k^2)$. Pre každú dvojicu (a, b) skúšame všetky kompatibilné trojice $(c, b, d) \in Y$. Spolu máme $O(k^5)$ možností, ktoré musíme vyskúšať. Pre každú dvojicu farieb a každú k nej kompatibilnú trojicu z Y robíme opäť maximálne párenie na bipartitnom grafe s veľkosťou $O(k)$. Spolu teda dostávame časovú zložitosť $O(k^{7.5})$.

3. Vrchol v leží na kružnici, ktorá je jeho otcovská:

   Spracovanie tohto typu vrcholu má dve fázy. Najprv sa vypočíta F_v a potom neskôr sa pozbierajú výsledky na tvorbu množiny Y.

   V prvej fáze skúšame všetky trojice farieb z množiny $\{0, 1, \ldots, k\}$. Tých je $O(k^3)$. Pre každú trojicu robíme bipartitný graf s veľkosťou $O(k)$ a hľadáme v ňom maximálne párenie. Táto fáza má teda časovú zložitosť $O(k^{5.5})$.

   V druhej fáze môžeme buď vytvárať prvé X, alebo tvoriť z predchádzajúceho X to nasledujúce. V prvom prípade je časová zložitosť $O(k^3)$, lebo musíme prejsť všetky trojice z F_v. V druhom prípade máme množinu X, ktorá obsahuje štvorice čísiel z množiny $\{0, 1, \ldots, k\}$. Preto $|X| = O(k^4)$. Pre každú štvoricu v X hľadáme všetky také trojice z F_v, že sa dajú skombinovať. Pri naivnej implementácií by sme skúsili skombinovať každú štvoricu z X s každou trojicou z F_v. Tak by sme dostali časovú zložitosť $O(k^7)$. Z posledného X vyrobíme ešte Y. To nám bude trvať $O(k^4)$, ale táto práca sa schová do $O(k^7)$.

   Čas na spracovanie vrcholu takéhoto typu je teda $O(k^7)$.

Nech vrcholov i-teho typu v grafe G' je n'_i. Zjavne $n'_1 + n'_2 + n'_3 = |V'|$. Celková časová zložitosť tejto fázy teda je $O(n'_1 k^{4.5} + n'_2 k^{7.5} + n'_3 k^7)$. Najhoršiu časovú zložitosť majú vrcholy druhého typu. Týchto vrcholov môže byť v cyklovom strome až jedna tretina. Preto celková časová zložitosť tejto fázy je $O(|V'|k^{7.5})$.

Zostáva rozobrať poslednú fázu. Najhoršie, čo sa nám v tejto fáze môže stať je, že sme odstránili kružnicu. Ak sme odstránili hranu, potom už nemusíme nič robiť, len sa pozrieť

do tabuľky. Nech počet odstránených vrcholov je m. Ku všetkým týmto vrcholom sa správame ako keby boli tretieho typu. To znamená, že im počítame možné trojice farieb a následne tieto výsledky zbierame. To znamená, že platí analýza popísaná vyššie pre tretí typ vrcholov, preto táto fáza má časovú zložitosť $O(mk^7)$.

Poslednú fázu môžeme pripočítať k predchádzajúcej fáze, lebo platí $|V'| + m = |V|$. Preto celková časová zložitosť posledných dvoch fáz je $O(|V'|k^{7.5} + mk^7) = O(|V|k^{7.5})$. Prvé fázy túto časovú zložitosť nezhoršia, lebo majú len lineárnu časovú zložitosť od počtu vrcholov grafu. □

Niektoré odhady v predchádzajúcom dôkaze vôbec neboli tesné, alebo by sa dali šikovnou implementáciou výrazne vylepšiť. Príkladom môže byť aj algoritmus, ktorý sme rozširovali. Z pôvodnej časovej zložitosti $O(|V|k^{4.5})$ sa len lepšími odhadmi a efektívnejšou implementáciou stala lineárna časová zložitosť. Pre naše potreby nám však zatiaľ tento výsledok stačí.

2.3 Rozšírenie algoritmu

Princíp algoritmu, ktorý sme opísali v časti 2.1, sa dá použiť aj na kaktusy, ktoré nie sú cyklové stromy.

Nech máme daný kaktus G a číslo k. Určí sa najväčšie také číslo t, že existuje vrchol grafu G, ktorý leží na t kružniciach. Myslí sa pod tým počet blokov, ktoré sú kružnice a tento vrchol na nich leží.

Bude sa používať ten istý postup ako v predchádzajúcom algoritme. Kaktus sa zakorení a vrcholy sa rozdelia do dvoch skupín podľa počtu otcovských hrán. V tomto prípade bude trochu väčší počet možností, ako môže vyzerať okolie spracúvaného vrcholu v. Nech vrchol v má m synovských kružníc. Pre každú takúto kružnicu sa vypočíta jej množina Y rovnako, ako v pôvodnom algoritme pre cyklové stromy. Označme si množinu Y i-tej kružnice Y_i.

Pri zisťovaní, či nejaká dvojica alebo trojica farieb patrí do F_v sa musí vyskúšať každá kombinácia trojíc farieb z množín Y_i. Ak kombinácia spĺňa podmienky $L(2,1)$-farbenia, tak sa ešte pre ostatné synovské vrcholy na synovských hranách zistí, či sa dajú ofarbiť. To sa urobí rovnakým spôsobom ako v predchádzajúcom algoritme. Všetkých kombinácií trojíc z množín Y_i pre $i \in \{1, 2, \ldots, m\}$ je $O(k^{3m})$. Na zistenie, či sa dajú aj zvyšné synovské vrcholy ofarbiť treba čas $O(k^{2.5})$.

Spolu dostávame, že pre vrcholy s jednou otcovskou hranou a m synovskými kružnicami bude treba čas $O(k^{2+3m+2.5})$. Pre vrcholy s dvoma otcovskými hranami a m synovskými kružnicami bude treba čas $O(k^{3+3m+2.5} + k^7)$. Do tohto odhadu sme pridali aj k^7 preto, lebo to je čas, ktorý algoritmus strávi pri vrchole v pri výpočte množiny Y pre jeho otcovskú kružnicu. Najhorší prípad nastane, keď vrchol v má t synovských kružníc a jednu otcovskú

hranu. To znamená, že časová zložitosť pre tento vrchol je $O(k^{2+3t+2.5}) = O(k^{3t+4.5})$. Celý algoritmus má v tomto prípade časovú zložitosť $O(|V(G)|k^{3t+4.5})$.

Dôsledkom tohto trvdenia je, že ak bude parameter t konštantný, tak výsledná časová zložitosť bude polynomiálna. To znamená, že pre kaktusy s konštantným počtom kružníc, na ktorých môže ležať jeden vrchol, je rozhodovacia verzia problému $L(2,1)$-farbenia tiež polynomiálne riešiteľná.

Kapitola 3

Ohraničenie $\lambda(G)$

Podobne ako pre stromy, tak aj pre iné triedy grafov existujú ohraničenia pre $\lambda(G)$ vzhľadom na maximálny stupeň grafu G z danej triedy. Preto sme aj my chceli nájsť, pokiaľ je to možné, tesné ohraničenie $\lambda(G)$ pre kaktusy a cyklové stromy. V tejto kapitole ukážeme dve ohraničenia. Prvé platí pre všetky kaktusy a je tesné pre kaktusy a aj pre cyklové stromy s malým maximálnym stupňom. Druhé, trochu lepšie ohraničenie, bude platiť pre kaktusy s väčším maximálnym stupňom.

V ďalších častiach sa budeme venovať hlavne horným ohraničeniam. Dolné ohraničenie je pre všetky grafy rovnaké a je to $\Delta + 1$, kde Δ je maximálny stupeň grafu. Tento fakt dokázali v článku [12]. Keďže je tento dôkaz jednoduchý, pre úplnosť ho uvedieme.

Lema 3.0.1 *Nech G je jednoduchý graf bez slučiek a násobných hrán s maximálnym stupňom Δ. Potom platí*

$$\lambda(G) \geq \Delta + 1$$

Dôkaz. Aby sme ukázali platnosť tohto tvrdenia, musíme ukázať, že na ofarbenie každého grafu potrebujeme aspoň $\Delta + 2$ farieb.

Všimnime si vrchol v, ktorý má maximálny stupeň. Tento vrchol má Δ susedov. Aby bola splnená druhá podmienka $L(2,1)$-farbenia, musia mať tieto vrcholy rôzne farby. Keď vrcholu v dáme niektorú z krajných farieb , potom jedna farba zo zvyšných farieb sa nebude dať použiť, aby sa neporušila prvá podmienka $L(2,1)$-farbenia. Keby sme dali vrcholu v niektorú zo stredných farieb, potom by sa nedali pre susedov v použiť dve ďalšie farby. Vrchol v teda obsadí aspoň dve farby a jeho pre susedov musí zostať aspoň Δ voľných farieb. Z toho dostávame, že potrebujeme aspoň $\Delta + 2$ farieb, aby bolo možné graf ofarbiť. Potrebujeme teda množinu farieb $\{0, 1, \ldots, \Delta + 1\}$. Preto $\lambda(G) \geq \Delta + 1$. □

Tento dolný odhad je tesný. Ako ukázali v článku [12], existuje nekonečne veľa grafov, pre ktoré platí $\lambda(G) = \Delta + 1$. Sú to napríklad stromy, ktoré majú jeden vrchol maximálneho stupňa a zvyšné vrcholy sú listy.

3.1 Ohraničenie pre všetky kaktusy

V tejto sekcií ukážeme, že pre ľubovoľný kaktus platí, že sa dá $L(2,1)$-ofarbiť pomocou $\Delta + 4$ farieb. Ako sme už spomínali, budeme sa venovať iba hornému odhadu, lebo dolný je jasný. Ukážeme nasledujúcu vetu

Veta 3.1.1 *Nech graf G je kaktus s maximálnym stupeň Δ. Potom platí:*

$$\Delta + 1 \leq \lambda(G) \leq \Delta + 3$$

Navyše existujú kaktusy, ktoré dosahujú aj dolnú a hornú hranicu.

Dôkaz ukážeme postupne. Ako prvý uvedieme dôkaz horného ohraničenia a potom ukážeme, že toto ohraničenie je tesné pre kaktusy a cyklové stromy.

3.1.1 Pažravý algoritmus

Aby sme ukázali platnosť horného odhadu, uvedieme pažravý algoritmus, ktorý dostane ľubovoľný kaktus a farby z množiny $\{0, 1, \ldots, \Delta+3\}$. Jeho výstupom bude $(\Delta+3)$-$L(2,1)$-farbenie f grafu G.

Zakoreníme kaktus v ľubovoľnom vrchole r pomocou algoritmu z časti 1.2.1 a dáme mu farbu 0. Tým, že tak spravíme, každému vrcholu určíme množinu jeho synov a ich rozdelenie do kružníc a hrán. To znamená, že budeme mať kaktus rozdelený na jednotlivé bloky.

V našom algoritme bude farbený kaktus po blokoch. Vždy sa bude farbiť blok iba v tom momente, keď práve jeden z jeho vrcholov bude už ofarbený. Bude to konkrétne vždy otec tohto bloku.

Predpokladajme, že existuje funkcia, ktorá dostane blok, čiastočné farbenie grafu G farbami v rozmedzí 0 až $\Delta + 3$ a jej výstupom je doplnené čiastočné farbenie zo vstupu na vrcholy zadaného bloku. Táto funkcia predpokladá, že práve jeden vrchol tohto bloku už bol ofarbený. Označme túto funkciu OFARBI_BLOK. Potom nasledujúca rekurzívna funkcia ofarbí celý kaktus, ak ju zavoláme pre vrchol r a čiastočné farbenie, ktoré priradzuje $f(r) = 0$.

```
1: function OFARBI_PODKAKTUS(v, f)
2:     if v je list then return f
3:     for každý synovský blok B vrcholu v do
4:         f ← OFARBI_BLOK(B, f)
5:     for každého syna u vrcholu v do
6:         OFARBI_PODKAKTUS(u, f)
7:     return f
```

Treba ale ukázať, že sa dá naozaj spraviť funkcia OFARBI_BLOK.

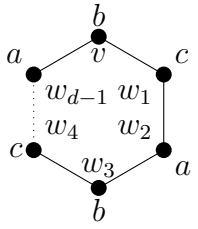
Pozrime sa na to, ako môže vyzerať blok B a aké sú možnosti jeho ofarbenia vzhľadom na už ofarbené vrcholy.

Prvým prípadom je, že blok B je hrana (v, u). Nech jej otcovský vrchol v má určenú farbu. V tomto prípade stačí nájsť farbu pre vrchol u, aby bol celý blok ofarbený. Vrchol v môže mať naviac $\Delta - 1$ už ofarbených susedov. Nepoužiteľných farieb z týchto vrcholov je teda najviac $\Delta - 1$. Nepoužiteľné farby vyplývajúce z farby vrcholu v sú najviac tri. Takže najviac $\Delta + 2$ farieb je už obsadených inými okolitými vrcholmi. Stále však pre vrchol u zostávajú aspoň dve farby, ktorými sa dá ofarbiť, lebo je k dispozícií aspoň $\Delta + 4$ farieb. Z možných farieb sa vyberie tá najmenšia.

Ďalším prípadom je, že blok B je kružnica. Ak B je celý graf G, tak platí $\lambda(G) = 4 = \Delta + 2$ a dá sa priamo ofarbiť. Predpokladajme, že $\Delta > 2$. Nech v je otcovský vrchol tejto kružnice a už má nejakú farbu b. Nech $w_1, w_2, \ldots, w_{d-1}$ sú vrcholy na kružnici v poradí, v akom ležia na kružnici a w_1 a w_{d-1} sú susedia vrcholu v. Môže existovať najviac $\Delta - 2$ už ofarbených susedov vrcholu v. To znamená, že pre w_1 a w_{d-1} je zakázaných najviac $\Delta + 1$ farieb. Čiže aspoň tri farby sú pre nich voľné. Vyberie sa taká dvojica (a, c) z týchto farieb, že $|a - c| \geq 2$. Keďže sa tieto farby vyberajú najmenej z troch možných farieb, určite taká dvojica existuje. Farby a, b a c majú takú vlastnosť, že sa dajú na kružnici dookola striedať. Algoritmus rozlišuje niekoľko prípadov podľa dĺžky d kružnice B.

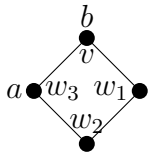
- $d \equiv 0 \pmod 3$: Farby a, b a c sa budú na kružnici stále striedať. Farba pre $w_i, i \in \{1, 2, \ldots, d-1\}$ sa teda určí takto (Obr. 3.1):

$$\begin{aligned} f(w_i) &= a\ ,\ i \equiv 2 \pmod 3 \\ f(w_i) &= b\ ,\ i \equiv 0 \pmod 3 \\ f(w_i) &= c\ ,\ i \equiv 1 \pmod 3 \end{aligned}$$



Obr. 3.1: $d \equiv 0 \pmod 3$

- $d = 4$: Vrcholu w_3 sa priradí farba a (Obr. 3.2). Zostáva určiť farby pre w_1 a w_2. Pozrime sa na počty farieb, ktoré sa dajú použiť pre vrcholy w_1 a w_2.



Obr. 3.2: $d = 4$

Vrchol w_1 má ofarbeného práve jedného suseda a najviac $\Delta - 1$ vrcholov vo vzdialenosti dva. Farieb, ktoré už nie sú preňho použiteľné, je najviac $\Delta + 2$. Preto má aspoň dve farby, ktoré sú preňho prípustné. Nazvime túto množinu farieb P_1.

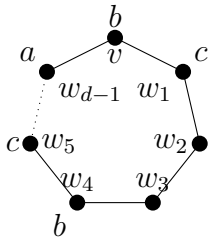
Vrchol w_2 má len jedného už ofarbeného suseda a práve jeden ofarbený vrchol vo vzdialenosti dva. Z toho dostávame, že najviac štyri farby sú pre vrchol w_2 už nepoužiteľné. Keďže je k dispozícií aspoň sedem farieb ($\Delta \geq 3$), tak vrchol w_2 má aspoň tri použiteľné farby. Nazvime túto množinu P_2.

Zjavne musia existovať farby e_1 a e_2 také, že $e_1 \in P_1$, $e_2 \in P_2$ a $|e_1 - e_2| \geq 2$. Vyberie sa zo všetkých takých dvojíc lexikograficky najmenšia. Potom

$$\begin{aligned} f(w_1) &= e_1 \\ f(w_2) &= e_2 \end{aligned}$$

- $d \equiv 1 \pmod 3 \wedge d \neq 4$: Farby a, b a c sa opäť budú striedať podobne ako v prvom prípade. Dva vrcholy však zostanú neofarbené. Konkrétne to budú vrcholy w_2 a w_3. Algoritmus priradí vrcholu w_1 farbu c a všetkým vrcholom w_i pre $i \in \{4, 5, \ldots, d-1\}$ priradí nasledujúce farby(Obr. 3.3):

$$\begin{aligned} f(w_i) &= a\ ,\ i \equiv 0 \pmod 3 \\ f(w_i) &= b\ ,\ i \equiv 1 \pmod 3 \\ f(w_i) &= c\ ,\ i \equiv 2 \pmod 3 \end{aligned}$$



Obr. 3.3: $d \equiv 1 \pmod 3 \wedge d \neq 4$

Pre oba vrcholy w_2 aj w_3 platí, že majú ofarbeného práve jedného suseda a práve dva vrcholy, ktoré sú od nich vo vzdialenosti dva. Tieto dva vrcholy majú vždy tú istú

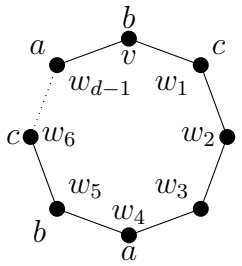
farbu. Preto oba vrcholy majú zakázané najviac štyri farby. To znamená, že aj vrcholy w_2 a w_3 majú k dispozícií aspoň tri farby. Čiže existujú také farby e_2 a e_3, že e_2 je prípustná farba pre w_2 a e_3 je prípustná farba pre w_3 a navyše $|e_2 - e_3| \geq 2$. Vyberie sa lexikograficky najmenšia taká dvojica. Potom:

$$\begin{aligned} f(w_2) &= e_2 \\ f(w_3) &= e_3 \end{aligned}$$

- $d \equiv 2 \pmod 3$: Aj v tomto prípade sa budú striedať farby a, b a c na vrcholoch $w_4, w_5, \ldots, w_{d-1}$:

$$\begin{aligned} f(w_i) &= a\ ,\ i \equiv 1 \pmod 3 \\ f(w_i) &= b\ ,\ i \equiv 2 \pmod 3 \\ f(w_i) &= c\ ,\ i \equiv 0 \pmod 3 \end{aligned}$$

Algorimus priradí $f(w_1) = c$, podobne ako v predchádzajúcich prípadoch (Obr. 3.4).



Obr. 3.4: $d \equiv 2 \pmod 3$

Vrchol w_2 má zakázané farby z tejto množiny $Z_2 = \{c-1, c, c+1, a, b\}$. Ďalej vrchol w_3 má zakázané farby z množiny $Z_3 = \{a-1, a, a+1, b, c\}$. Samozrejme, nie všetky čísla v Z_2 a Z_3 musia byť povolené farby. Niektoré môžu byť záporné, alebo väčšie ako $\Delta + 3$.

Nech jedno z čísiel a alebo c je jedna z krajných farieb (0 alebo $\Delta + 3$). Ak je to číslo a, potom jedno z čísel $a-1$ a $a+1$ nie je povolená farba. Podobne, ak je to c, tak jedno z čísel $c-1$ a $c+1$ nie je povolená farba. Preto buď w_2 alebo w_3 má aspoň tri povolené farby. Z toho vyplýva, že existuje dvojica farieb, ktorá vyhovuje podmienkam $L(2,1)$ farbenia. Nech sú to farby e_2 a e_3. Potom

$$\begin{aligned} f(w_2) &= e_2 \\ f(w_3) &= e_3 \end{aligned}$$

Druhá možnosť je, že ani jedno z čísel a a c nie je krajná farba. Bez ujmy na všeobecnosti nech $a < c$. V tomto prípade algoritmus môže priradiť

$$f(w_2) = a - 1$$

$$f(w_3) \;=\; c+1$$

Zjavne budú stále zachované všetky podmienky $L(2,1)$-farbenia.

Keď sa teda ide farbiť kružnica, tak sa vyberú dve farby a a c, aby sa spolu s farbou b, ktorú má otec bloku, dali dookola striedať. V prípade, že farbená kružnica má dĺžku deliteľnú tromi, potom tieto farby sa budú stále striedať a ofarbí sa tak celá kružnica. V opačnom prípade sa takto ofarbí takmer celá kružnica. Nechajú sa neofarbené dva vrcholy. Ak sa dá, vyberú sa také vrcholy, ktoré nie sú susedné s vrcholom v. Ak sa nedá, nevadí. Pre každý neofarbený vrchol sa zistí množina farieb, ktoré môže mať vzhľadom na farby okolitých vrcholov. Z týchto množín sa vyberú také farby, že neporušujú žiadnu podmienku $L(2,1)$-farbenia.

Tým sme túto časť dôkazu ukončili. Každý blok sa dá ofarbiť, ak bol ofarbený len jeho otcovský vrchol. Ofarbenie bloku sa dá uskutočniť vyššie spomínaným postupom. Zjavne pri každom volaní funkcie OFARBI_BLOK sa dá daný blok B ofarbiť nejakým spôsobom.

3.1.2 Tesnosť horného ohraničenia

Keď sme našli vyššie spomenutý algoritmus, chceli sme vedieť, či existujú grafy, ktoré dosahujú hornú hranicu. Snažili sme sa preto také grafy nájsť. Na to sme ale potrebovali pomoc počítača. Naprogramovali sme si niekoľko funkcií na generovanie kaktusov, na hľadanie $\lambda(G)$ pre daný graf G a zisťovanie ďalších zaujímavých vlastností $L(2,1)$-farbenia grafov. Funkcionalitu sme rozdelili do dvoch skupín

- funkcie na generovanie kaktusov
- funkcie na zisťovanie vlastností $L(2,1)$-farbenia kaktusov

V nasledujúcich dvoch častiach opíšeme hlavné myšlienky použité pri implementácií.

Generovanie kaktusov

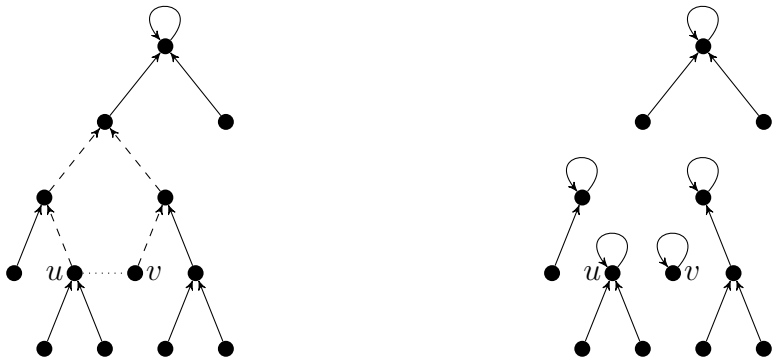
Začneme s generovaním. Všetky kaktusy budeme generovať zo stromov. Stromy sú tiež kaktusy a všetky ostatné sa z nich dajú vyrobiť správnym pridávaním hrán.

Najprv opíšeme, ako vyzerá správne pridávanie hrán. Máme dve kópie stromu T, ktorý tvorí kostru budúceho kaktusu. Jedna kópia T_1 sa používa na zisťovanie, či sa dá nejaká hrana pridať a druhá kópia T_2 slúži na uchovávanie čiastočne vyrobeného kaktusu. Nech chceme do vytváraného stromu pridať hranu h. Keď sa táto hrana pridá do stromu T_1, vyrobí kružnicu. Ku grafu T_2 sa pridá hrana h. Z T_1 sa následne odstránia všetky hrany, ktoré ležali na kružnici. Tým sa strom rozpadne na niekoľko kostrových komponentov a vznikne les T_1'. Ak chceme pridať ďalšiu hranu, musí to byť hrana, ktorá spája vrcholy v jednom komponente lesa T_1'. Keby sa spojili vrcholy v rôznych komponentoch, tak bude porušená definícia

kaktusu. Predstavme si, že ideme pridať hranu (u, v), ktorá spája dva rôzne komponenty T_1'. V pôvodnom strome T to vyrobí kružnicu. Keďže ale v T_1' to kružnicu nevyrobí, musí táto kružnica v T obsahovať hrany, ktoré už sú odstránené. Tie sa však odstraňovali preto, lebo už ležia na kružnici. To znamená, že pridaním hrany (u, v) do vytváraného kaktusu, by vznikli dve kružnice, ktoré ale nie sú hranovo disjunktné.

Kaktus teda vytvoríme tak, že vyberieme hranu, ktorá spája dva vrcholy v jednom kostrovom komponente a z kostry odstránime všetky hrany, ktoré ležia na kružnici. Tým sa tento kostrový komponent rozpadne na menšie komponenty. Hrany sa dajú pridávať až dovtedy, kým existuje kostrový komponent s aspoň tromi vrcholmi.

Rozpadnutie kostry sa dá urobiť pomerne pohodlne. Predstavme si, že máme kostru zadanú ako pole, kde na políčku v je zapísaný otec vrcholu v. V prípade, že v je koreň stromu, tak má na svojom políčku zapísané v. Čiže je si sám otcom. Predpokladajme teraz, že ideme pridať hranu (u, v). Najprv musíme nájsť hrany kostry, ktoré patria do kružnice. Môžeme si značiť vo vrcholoch, či ich otcovská hrana je v tejto kružnici. Pre u a v ideme postupne po otcovských hranách hore a po ceste zanechávame značky. Potom platí, že do kružnice patria hrany, ktoré sú otcovské hrany vrcholov s jednou značkou. Aby sa kostra rozpadla na správne komponenty, stačí, aby sa každému takému vrcholu nastavilo, že je si sám otcom (Obr. 3.5). Jeden komponent je reprezentovaný svojim koreňom. Každý vrchol na kružnici už bude mať iného reprezentanta komponentu.



Obr. 3.5: Pridávanie hrany. (a) Pôvodný stav pred pridaním hrany (u, v). Čiarkované hrany sú hrany na kružnici, ktorá vznikne pridaním tejto hrany. (b) Stav po pridaní hrany (u, v) a odstránení hrán na kružnici.

Predstavme si, že máme daný strom T. Z tohto stromu chceme vygenerovať všetky možné kaktusy, ktorým je tento strom kostrou. Na začiatku je možné pridať ľubovoľnú hranu, ktorá už nie je v strome. Vyberieme si jednu takú hranu h. Máme dve možnosti. Buď tam tú hranu dáme, alebo nie. V prípade, že nie, len z množiny prípustných hrán odstránime hranu h a môžeme pokračovať ďalej. V prípade, že ju pridáme do kaktusu,

kostra sa rozpadne na komponenty. Hrany, ktoré boli predtým použiteľné a spájajú dva rôzne komponenty sú už nepoužiteľné a preto ich odstránime z použiteľných hrán. V prípade, že množina použiteľných hrán je prázdna, potom už nemáme čo pridávať a graf, ktorý sme vyrobili je kaktus, ktorého kostrou je T. Stačí ho vrátiť na výstup.

Máme teda rekurzívnu funkciu GENERUJ_KAKTUSY, ktorá ako vstup dostane čiastočný kaktus $G = (V, E)$, jeho kostrové komponenty T a množinu použiteľných hrán H. Táto funkcia potom vráti množinu všetkých neizomorfných kaktusov, ktoré sa dajú vygenerovať pridaním hrán z H ku kaktusu G. V prípade, že H je neprázdna, robíme dve rekurzívne volania, ktorých výsledky spájame. Nech sa ide spracovať hrana h. Jedno rekurzívne volanie bude s parametrami G, T a $H - h$. Druhé volanie bude z parametrami $G' = (V, E \cup \{h\})$, T' a H', kde T' a H' vyrobíme vyššie popísaným spôsobom, rozpadom niektorého komponentu v T a odstránením príslušných nepoužiteľných hrán z H. Výsledkom budú dve množiny kaktusov G_1 a G_2. Vieme, že všetky grafy v G_1 sú navzájom neizomorfné, a rovnako aj všetky grafy v G_2 sú navzájom neizomorfné. Keď teda chceme spraviť zjednotenie týchto množín, musíme zabezpečiť, aby sa do zjednotenia nedostali dva izomorfné grafy. To sa spraví tak, že sa každý graf z G_2 otestuje, či nie je izomorfný s niektorým grafom z G_1. Ak nie, potom pôjde do výsledku. Pseudokód tejto funkcie môžete vidieť nižšie.

1: **function** GENERUJ_KAKTUSY(G,T,H)
2: **if** $H = \emptyset$ **then return** [G]
3: $h \leftarrow$ nejaká hrana z H
4: $G_1 \leftarrow$ GENERUJ_KAKTUSY$(G, T, H - h)$
5: $G' \leftarrow G$ s pridanou hranou h
6: $T' \leftarrow$ upravené T podľa pridanej hrany
7: $H' \leftarrow H$ bez nepoužiteľných hrán
8: $G_2 \leftarrow$ GENERUJ_KAKTUSY(G', T', H')
9: $vysledok = G_1$
10: $izomorfny = False$
11: **for** $g_2 \in G_2$ **do**
12: **for** $g_1 \in G_1$ **do**
13: **if** g_2 a g_1 sú izomorfné **then** $izomorfny = True$
14: **if** nie izomorfny **then** g_2 pridaj do $vysledok$
15: **return** $vysledok$

Už sme ukázali, ako sa dá jednoducho rozložiť graf na správne komponenty po pridaní nejakej hrany. Zostáva nám ešte ukázať, ako vieme identifikovať hranu, ktorá je už nepoužiteľná. Vieme, že táto hrana nesmie ležať medzi dvoma komponentmi. Nech sa ide testovať

hrana (u, v). Pre vrcholy u a v sa nájdu reprezentanti komponentov, v ktorých ležia. Ako sme vyššie spomínali, reprezentantom komponentu je jeho koreň. Pre u a v sa nájde koreň tak, že sa postupne posúva vyššie po otcovských hranách. Keď sa dosiahne vrchol, ktorého otec je on sám, tak tento vrchol je koreň a teda reprezentant komponentu. Na záver už len stačí otestovať, či takto vypočítaní reprezentanti sa zhodujú alebo nie. Ak sa nezhodujú, tak hrana (u, v) je už nepoužiteľná.

Teraz vieme z jedného stromu nerovať všetky kaktusy, ktoré tento strom majú ako kostru. Táto kostra však určite nie je kostrou všetkých kaktusov. Aby sme teda vygenerovali všetky kaktusy, musíme vyššie spomenutú procedúru použiť na všetky stromy. Keďže jeden kaktus má veľa neizomorfných kostier, takýmto spôsobom vygenerujeme ten istý kaktus viacerými spôsobmi. Musíme teda každý kaktus, ktorý takto vygenerujeme, testovať, či sme ho už neukladali do poľa s výslednými kaktusmi.

Aby sme generovanie čo najviac urýchlili, urobili sme ešte zopár úprav. Pri generovaní kaktusov z jedného stromu je zbytočné generovať aj tie, ktoré majú maximálny stupeň väčší ako pôvodný strom. Nech máme kaktus G s maximálnym stupňom Δ. Tento kaktus zjavne má aj kostru T s maximálnym stupňom Δ. Keď teda budeme generovať všetky kaktusy z kostry T, určite vygenerujeme aj kaktus G. T toho vyplýva, že ak budeme generovať kaktusy zo všetkých stromov, tak aspoň raz vygenerujeme každý kaktus. To sa dá implementovať tak, že pri určovaní, či je hrana použiteľná sa navyše urobí test, či jeden z jej vrcholov už nemá maximálny stupeň. Ak má, tak zjavne jej pridaním, by sme maximálny stupeň zväčšili a to nechceme. Ďalej sme už prvú množinu H, ktorú sme posielali pri prvom volaní, zbavili takých hrán, ktorých aspoň jeden vrchol mal maximálny stupeň.

Ďalšou úpravou bolo záverečné testovanie na maximalitu. Kaktus voláme maximálny, ak neexistuje hrana, ktorá by sa dala pridať, bez porušenia definície kaktusu a zväčšenia maximálneho stupňa. Túto úpravu sme si zvolili preto, lebo takéto grafy majú najväčšie čísla $\lambda(G)$. Zjavne platí, že ak graf G' je podgrafom grafu G, potom $\lambda(G') \leq \lambda(G)$. Keďže chceme nájsť kaktus dosahujúci hornú hranicu pre $\lambda(G)$, stačí, keď budeme hľadať medzi maximálnymi grafmi.

Maximalita grafu sa dá otestovať pomerne jednoducho. Kaktus je maximálny, keď žiaden z jeho kostrových komponentov neobsahuje dva vrcholy s menším stupňom ako maximálnym. Pre každý komponent teda stačí spočítať počet takýchto vrcholov. To sa dá spraviť tak, že sa každému vrcholu, ktorý nemá maximálny stupeň v G, vypočíta reprezentant jeho kostrového komponentu. Pre každého reprezentanta sa zaznamenáva, koľko takých vrcholov pod sebou má. Potom už len stačí skontrolovať, či niektorý z reprezentantov nemá pod sebou aspoň dva také vrcholy.

Naprogramovali sme teda jednu funkciu, ktorá dostane množinu stromov a informáciu, či chceme generovať všetky kaktusy alebo len maximálne. Následne jej výstupom je príslušná

množina kaktusov. Nech sa táto funkcia volá GENERUJ.

Farbenie kaktusov

Aby sme mohli pre vygenerované grafy zistiť ich čísla $\lambda(G)$, vyrobili sme si niekoľko pomocných funkcií na testovanie rôznych vlastností ich $L(2,1)$ farbenia. Najprv sme vyrobili funkciu, ktorá pre kaktus G zistí číslo $\lambda(G)$. Keďže nemáme pre všetky kaktusy polynomiálny algoritmus na jeho zisťovanie, pomohli sme si pomocou SAT-solvera. SAT-solver sme použili hlavne kvôli jeho rýchlosti. Keby sme si chceli implementovať vlastné hľadanie $\lambda(G)$, výsledná implementácia by bola príliš pomalá.

Keď chceme zistiť $\lambda(G)$ pre graf G, musíme zisťovať či $\lambda(G) \leq k$, pre k rovné $\Delta + 1$ a $\Delta + 2$. Stačí vyskúšať len tieto možnosti, ako sme dokázali v prvej časti dôkazu vety 3.1.1. Keď nájdeme prvé k, kde to platí, tak máme výsledok. Ak to neplatí ani pre jedno z nich, potom $\lambda(G) = \Delta + 3$.

Teraz sa môžeme zamerať len na zisťovanie, či platí $\lambda(G) \leq k$ pre jedno konkrétne k. Pokúsime sa zistiť, či existuje k-$L(2,1)$-farbenie grafu G. Na to používame SAT-solver. SAT-solver je program, ktorý vie pomerne rýchlo rozhodnúť, či nejaká formula v konjunktívnej normálnej forme je splniteľná alebo nie. Sformulujeme teda formulu v konjuktívnej normálnej forme, ktorá vyjadruje podmienky k-$L(2,1)$-farbenia grafu G. Následne túto formulu dáme ako vstup SAT-solveru, ktorý pomerne rýchlo určí, či je táto formula splniteľná alebo nie. Ak je formula splniteľná, potom existuje k-$L(2,1)$-farbenie grafu G a ak nie je, tak neexistuje.

Teraz popíšeme konštrukciu formuly. Zavedieme si premenné $X_{v,c}$ pre každý vrchol $v \in V(G)$ a farbu $c \in \{0, \ldots, k\}$. Premenná $X_{v,c}$ znamená, že vrchol v má farbu c. Aby platilo, že formula F je splniteľná práve vtedy, keď graf G má k-$L(2,1)$-farbenie, musí zahŕňať tieto tri podmienky:

1. Každý vrchol musí mať priradenú farbu.
2. Susedné vrcholy grafu G musia mať farby, ktoré sa líšia aspoň o dva.
3. Vrcholy, ktoré majú spoločného suseda, musia mať rôzne farby.

Aby sme zabezpečili prvú podmienku, pridáme do formuly F nasledujúcu klauzulu pre každý vrchol $v \in V(G)$:

$$\left(X_{v,0} \vee X_{v,1} \vee \ldots \vee X_{v,k}\right)$$

Ďalej potrebujeme zaručiť, že dva susedné vrcholy budú mať farbu aspoň o dva rôznu. Inak povedané, nesmú mať rovnakú farbu a ani nesmú mať susedné farby. To znamená, že pre každé dva susedné vrcholy u a v pridáme do formuly F nasledujú sadu klauzúl:

$$\forall c \in \{0, \ldots, k\}: \qquad \left(X_{u,c} \rightarrow \neg X_{v,c}\right) \equiv \left(\neg X_{u,c} \vee \neg X_{v,c}\right)$$

$$\forall c \in \{0, \ldots, k-1\}: \quad \Big(X_{u,c} \to \neg X_{v,c+1}\Big) \equiv \Big(\neg X_{u,c} \vee \neg X_{v,c+1}\Big)$$
$$\forall c \in \{0, \ldots, k-1\}: \quad \Big(X_{v,c} \to \neg X_{u,c+1}\Big) \equiv \Big(\neg X_{v,c} \vee \neg X_{u,c+1}\Big)$$

Poslednú, tretiu podmienku zaručíme tak, že do formuly pridáme ešte ďalšiu sadu formúl pre každú dvojicu vrcholov, ktoré majú vzdialenosť dva. Pre každé také dva vrcholy u a v do formuly F pridáme nasledujúcu sadu klauzúl:

$$\forall c \in \{0, \ldots, k\}: \quad \Big(X_{u,c} \to \neg X_{v,c}\Big) \equiv \Big(\neg X_{u,c} \vee \neg X_{v,c}\Big)$$

Takto zostrojená formula F bude mať požadované vlastnosti. Funkciu, ktorá nájde $\lambda(G)$ budeme volať NÁJDI_LAMBDA.

Zistili sme, že samotné zisťovanie čísla $\lambda(G)$ nestačí. Ako uvidíme vo výsledkoch, ani jeden z vygenerovaných grafov nebol príkladom grafu, ktorého číslo λ by dosahovalo hornú hranicu. Potrebovali sme teda zistiť viac informácií. Rozhodli sme sa, že pre každý vrchol grafu zistíme množinu farieb, ktoré môže mať, aby sa zvyšok grafu dal k-$L(2,1)$-ofarbiť. Zjavne stačí skúšať len polovicu farieb. Zvyšná polovica je totiž symetrická. Ak vieme ofarbiť vrchol farbou c, vieme ho ofarbiť aj farbou $k - c$ tak, že prvé farbenie otočíme.

Zisťovanie, či vrchol v môže mať farbu c, môžeme urobiť jednoducho tak, že k formule F na zisťovanie existencie k-$L(2,1)$-farbenia pridáme ešte klauzulu $X_{v,c}$. Tá zaručí, že vrchol v musí mať farbu c. Nech je táto nová formula F'. F' dáme na vstup SAT-solveru. Ak aj F' bude splniteľná, potom bude existovať k-$L(2,1)$-farbenie grafu, ktoré priradí vrcholu v farbu c.

Funkciu, ktorá pre každý vrchol grafu G nájde množinu prípustných farieb, budeme volať ZISTI_MOŽNÉ_FARBY.

Implementácia

Zostáva ešte uviesť, aké technológie sme pri implenentácií používali. Celý program sme písali v jazyku Python [3].

Pri generovaní stromov, sme používali okrem štandardných pythonovských knižníc aj ďalšie programy. Na generovanie stromov sme používali implementáciu FreeTree od autorov Gang Li and Frank Ruskey, ktorí v nej implementovali svoje algoritmy z článku [17]. Táto implementácia má tú výhodu, že generuje všetky neizomorfné stromy, v nami požadovanom formáte. Navyše vieme generovanie ovplyvniť tak, že nastavíme horné ohraničenie pre maximálny stupeň stromov, ktoré chceme generovať. Testovanie izomorfizmu grafov sme robili pomocou pythonovskej knižnice `igraph` na prácu s grafmi [4], ktorá je implementovaná v jazyku C. Z toho dôvodu je rýchlejšia ako bežná pythonovská implementácia.

Pri zisťovaní rôznych vlastností $L(2,1)$-farbení sme používali hlavne SAT-solver. Použili sme SAT-solver implementovaný v pythonovskej knižnici `pycosat` [2]. Aby sme vedeli vyhodnotiť výsledky, ktoré našla funkcia ZISTI_MOŽNÉ_FARBY, potrebovali sme si vypočítané

údaje vizualizovať. Na to sme používali program Graphviz [1]. Tento program z jednoduchého textového súboru vyrobí obrázok grafu. Stačilo teda pomocou našej funkcie vypočítať, aké farby môžu mať vrcholy a následne tieto údaje uložiť vo formáte vhodnom pre tento program. Potom vygenerovaný výstup posunúť do programu Graphviz.

Výsledky

Najprv sme skúsili vygenerovať všetky neizomorfné kaktusy. Generovali sme postupne podľa počtu vrcholov. Takto sa nám pomocou funkcie GENERUJ podarilo vygenerovať všetky neizomorfné kaktusy, ktoré mali od 2 do 11 vrcholov. Pre väčší počet vrcholov už generovanie trvalo príliš dlho. Keď sme mali tieto grafy vygenerované, pre každý z nich sme našli jeho $\lambda(G)$. Počty kaktusov rozdelené podľa počtu vrcholov a ich čísla λ môžete vidieť v nasledujúcej tabuľke.

počet vrcholov	všetky	$\Delta+1$	$\Delta+2$	$\Delta+3$
2	1	1	0	0
3	2	1	1	0
4	4	3	1	0
5	9	7	2	0
6	23	17	6	0
7	63	49	14	0
8	188	148	40	0
9	596	484	112	0
10	1979	1636	343	0
11	6804	5733	1071	0

Tabuľka 3.1: Počty všetkých kaktusov, rozdelené podľa $\lambda(G)$

Počas hľadania príkladu kaktusu, ktorého číslo λ dosahuje hornú hranicu, sa nám poradilo dokázať, že pre kaktusy s väčším maximálnym stupňom ako 4 platí ešte tesnejšie ohraničenie. Aby sme teda generovanie urýchlili, generovali sme len kaktusy zo stromov, ktoré majú maximálny stupeň najviac 4. Navyše sme generovali len maximálne kaktusy. To nám pomohlo tak, že sme vygenerovali všetky také kaktusy až do 15 vrcholov. Aj tam sme následne pre každý vygenerovaný kaktus vypočítali jeho $\lambda(G)$. V tabuľke 3.2 môžete vidieť, výsledné počty kaktusov rozdelené podľa počtu vrcholov a čísla λ.

V poslednom stĺpci oboch tabuliek sú samé nuly. To znamená, že takýmto spôsobom sa nám nepodarilo nájsť kaktus, ktorý by dosahoval hornú hranicu. Museli sme ho teda hľadať iným spôsobom.

počet vrcholov	všetky	$\Delta+1$	$\Delta+2$	$\Delta+3$
12	383	44	339	0
13	835	75	760	0
14	1907	140	1767	0
15	4321	237	4084	0

Tabuľka 3.2: Počty maximálnych kaktusov, rozdelené podľa $\lambda(G)$

Zjavne pre všetky kaktusy, ktoré majú najviac 15 vrcholov platí, že ich $\lambda(G) \leq \Delta + 2$. Rozhodli sme sa, že zistíme, ktoré z vygenerovaných kaktusov má nejakým spôsobom obmedzené $(\Delta + 2)$-$L(2,1)$-farbenie. To znamená, že niektorý z vrcholov nemôže mať všetky farby. Túto informáciu sme zistili tak, že sme pre každý vygenerovaný kaktus G zavolali funkciu ZISTI_MOŽNÉ_FARBY s parametrom $k = \Delta + 2$. Ak niektorý z vrcholov grafu nemohol mať všetky farby, tento kaktus sme si uložili aj spolu s množinami možných farieb pre jednotlivé vrcholy do súboru pre program Graphviz.

Tieto grafy sme si rozdelili na tie, ktoré majú maximálny stupeň 3 a 4. Najprv sme sa zamerali na kaktusy s maximálnym stupňom $\Delta = 3$. V tomto prípade je $k = 5$. Z nájdených kaktusov sme spravili výber, ktorý môžete vidieť na obrázku 3.6. Do výberu sme vybrali iba grafy, ktoré mali unikátny počet zaujímavých vrcholov (tie ktoré nemohli mať všetky farby) alebo neboli nadgrafom žiadneho už vybratého kaktusu. Postupovali sme od kaktusov s menším počtom vrcholov.

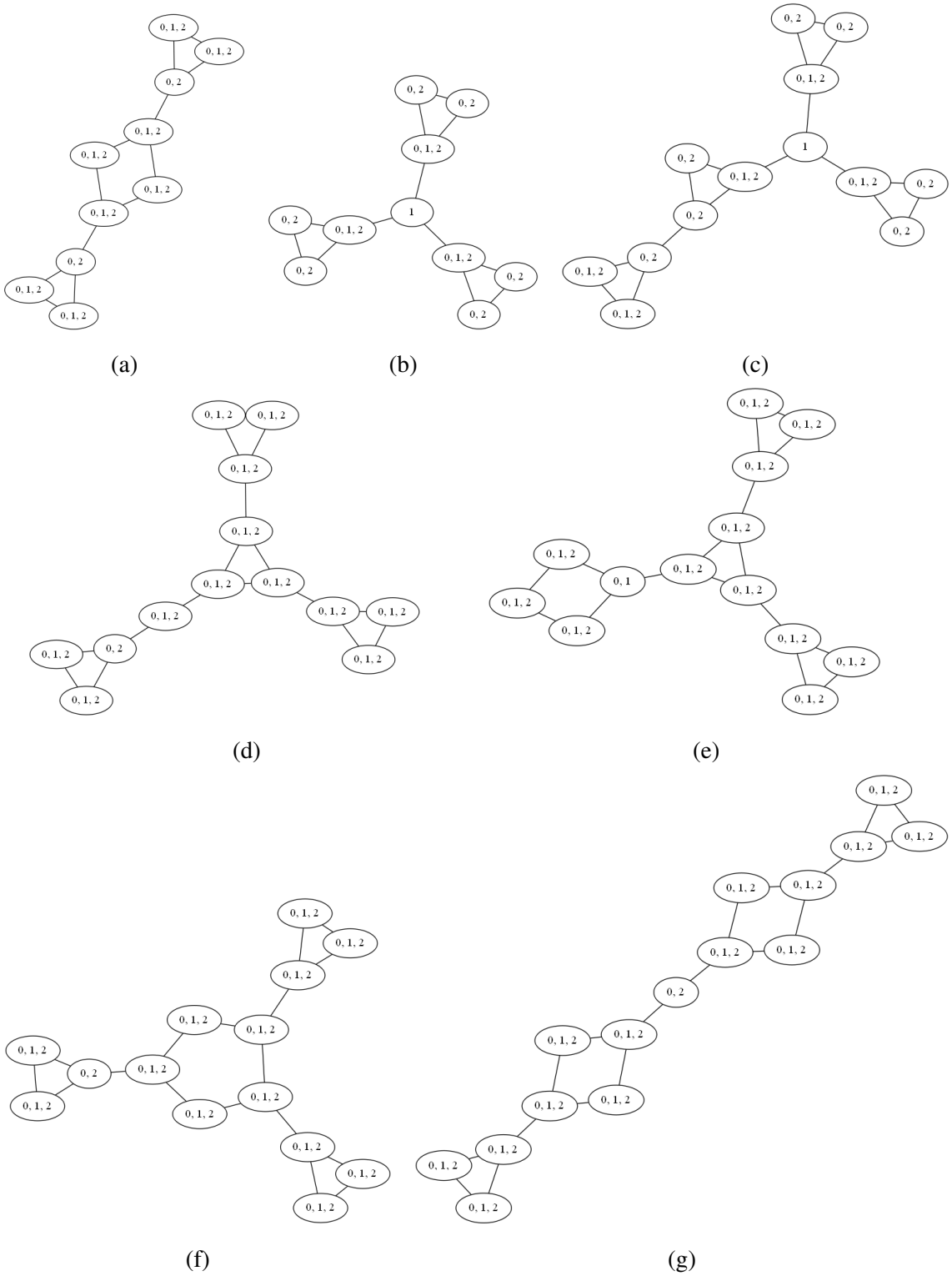
Z týchto grafov ja zaujímavý graf na obrázku 3.6b. Označme si ho G_0. Zaujímavé je, že okrem troch vrcholov, majú všetky jeho vrcholy nejakým spôsobom obmedzené farby. Navyše vrchol v strede má len dve možné farby a to 1 a 4. To značí, že má pravdepodobne málo rôznych $(\Delta + 2)$-$L(2,1)$-farbení.

Hľadali sme všetky jeho farbenia a zistili sme, že sú len štyri. Dve sú unikátne a zvyšné dve sú k nim symetrické. Tieto farbenia môžete vidieť na obrázku 3.7.

Graf G_0 sme sa následne snažili rozšíriť na väčší graf G_1, pre ktorý by platilo, že žiadne z farbení G_0 sa nedá rozšíriť na farbenie grafu G_1 bez pridania ďalšej farby. Malou nápovedou nám bol graf na obrázku 3.6c. Ak k listu grafu G_0 cez hranu napojíme kružnicu dĺžky tri, tak dostaneme ďalší vrchol, ktorý má obmedzené farby.

Skúsili sme preto graf G_0 rozšíriť tak, že sme pridali ku každému jeho listu hranu, ktorá má na druhej strane napojenú kružnicu dĺžky tri (Obr. 3.8). Tento graf sme následne otestovali pomocou funkcie NAJDI_LAMBDA, ktorá vrátila hodnotu $6 = \Delta + 3$.

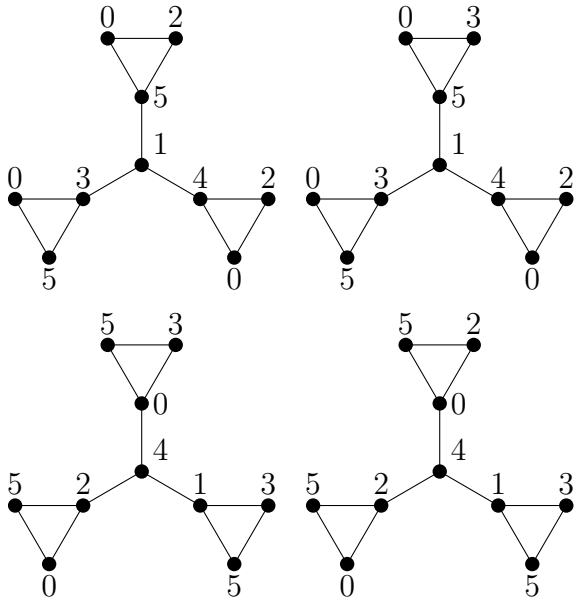
Spolu s týmto príkladom sme zároveň našli nekonečnú množinu kaktusov, ktorých číslo $\lambda(G)$ dosahuje hornú hranicu. Každý kaktus G s $\Delta = 3$, ktorý obsahuje G_1 ako podgraf, musí mať nutne $\lambda(G) = \Delta + 3$.



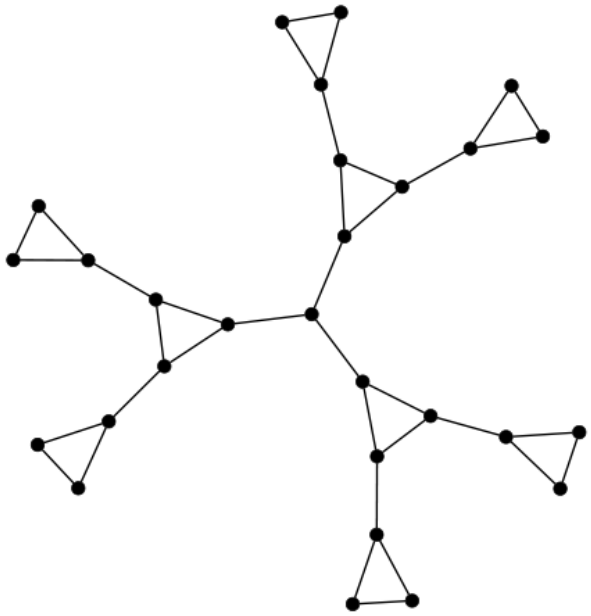
Obr. 3.6: Vybrané zaujímavé grafy. V každom vrchole je uvedená polovica farieb, ktoré tento vrchol môže mať. Druhá polovica sa dá doplniť symetricky

Môžete si tiež všimnúť, že graf G_1 je zároveň cyklový strom. Z toho dostávame, že horný odhad je tesný aj pre cyklové stromy.

Podobná metóda sa nedala použiť pre kaktusy s $\Delta = 4$. Zaujímavé grafy, ktoré sme



Obr. 3.7: Všetky možné $(\Delta + 2)$-$L(2,1)$-farbenia grafu G_0.



Obr. 3.8: Kaktus G_1, ktorý dosahuje hranicu $\lambda(G_1) = \Delta + 3$.

dostali, mali stále veľa farbení a bolo príliš veľa možností ako ich rozšíriť. Pre túto triedu kaktusov sme teda nezistili, či je tento odhad tesný.

Týmto sme ukončili dôkaz vety 3.1.1.

3.2 Ohraničenie $\lambda(G)$ pre kaktusy s $\Delta \geq 5$

Pri hľadaní kaktusu, ktorého číslo λ dosahuje hornú hranicu ohraničenia z vety 3.1.1, sme našli ešte lepšie ohraničenie čísla $\lambda(G)$, ktoré platí len pre kaktusy s väčším maximálnym stupňom.

Ak by sa nám podarilo nejakým spôsobom vylepšiť pažravý algoritmus z časti 3.1.1, mohli by sme tým zároveň vylepšiť ohraničenie. V predchádzajúcom algoritme sme pri kružniciach hľadali tri farby, ktoré sa mohli na tejto kružnici stále striedať. To sme zabezpečili tak, že pre dva prvé vrcholy na kružnici zostali aspoň tri voľné farby. Keby sme však tieto farby vyberali naraz pre všetky kružnice pod nejakým vrcholom, nebolo by nutné mať o jednu farbu viac.

Musíme však dávať pozor na prípad, keď má vrchol v len jednu synovskú kružnicu. V tomto prípade, podobne ako predtým, potrebujeme mať na výber aspoň tri farby, aby vždy existovala vhodná dvojica. V najhoršom prípade má vrchol v otcovskú kružnicu a tým pádom už dvoch ofarbených susedov. V tom prípade je pre susedných synov v obsadených najviac 5 farieb. Keďže potrebujeme aspoň 3 voľné farby, musíme mať aspoň 8 farieb. Z toho vyplýva, že kaktus, ktorý takto chceme farbiť musí mať maximálny stupeň aspoň 5.

Z týchto úvah vznikol dôkaz nasledujúcej vety.

Veta 3.2.1 *Nech graf G je kaktus s maximálnym stupňom $\Delta \geq 5$. Potom platí:*

$$\Delta + 1 \leq \lambda(G) \leq \Delta + 2$$

Dôkaz. Opäť sa zameriame len na horné ohraničenie. Ukážeme pažravý algoritmus, ktorý každý takýto graf ofarbí v súlade s podmienkami $L(2,1)$-farbenia a použije farbu najviac $\Delta + 2$.

Podobne, ako v predchádzajúcom prípade, sa kaktus zakorení v ľubovoľnom vrchole r a položí sa $f(r) = 0$. Rozdiel oproti predchádzajúcemu algoritmu bude taký, že sa budú farbiť všetky synovské bloky spracúvaného vrcholu naraz. Algoritmus prechádzania kaktusu bude rovnaký ako v predchádzajúcom prípade, len budeme pre daný vrchol volať funkciu OFARBI_SYNOVSKÉ_BLOKY(v, f), ktorá naraz ofarbí všetky synovské bloky vrcholu v vzhľadom na čiastočné farbenie f:

```
1: function OFARBI_PODKAKTUS(v, f)
2:     if v je list then return f
3:     f ← OFARBI_SYNOVSKÉ_BLOKY(v, f)
4:     for každého syna u vrcholu v do
5:         OFARBI_PODKAKTUS(u, f)
6:     return f
```

Teraz už len treba ukázať, že funkcia OFARBI_SYNOVSKÉ_BLOKY existuje.

Nech teda farbíme synovské bloky vrcholu v, ktorý už má farbu b. Ak má vrchol jednu otcovskú hranu, tak má práve jedného suseda už ofarbeného a ak má dve otcovské hrany, tak má práve dvoch už ofarbených susedov. Nech $c_1, c_2, \ldots, c_k$ sú voľné farby v rastúcom poradí pre susedných synov vrcholu v. Nech má vrchol v m synovských kružníc a n synovských hrán. Z toho dostávame, že v má presne $2m + n$ susedných synov.

Vieme, že platí:

$$2m + n \leq k$$

Vychádza to preto, lebo máme k dispozícií $\Delta+3$ farieb. Z toho najviac tri farby obsadil vrchol v svojou farbou. Vrchol v má najviac stupeň Δ a tak zvyšných farieb je dosť na ofarbenie všetkých jeho susedov.

Označme si teraz všetkých synov vrcholu v. Vrcholy $w_{i,1}, w_{i,2}, \ldots, w_{i,d_i-1}$ sú vrcholy na i-tej synovskej kružnici a d_i je jej dĺžka, $i \in \{1, 2, \ldots, m\}$. Vrcholy $v_1, v_2, \ldots, v_n$ sú synovské vrcholy spojené s v synovskou hranou.

Ak je počet synovských kružníc aspoň dva, tak sa ofarbia susedné synovské vrcholy takto: Pre každé $i \in \{1, 2, \ldots, m\}$:

$$\begin{aligned} f(w_{i,1}) &= c_i \\ f(w_{i,d_i-1}) &= c_{i+m} \end{aligned}$$

a pre každé $i \in \{1, 2, \ldots, n\}$:

$$f(v_i) = c_{i+2m}$$

Rozoberme teraz prípad, keď v má len jednu synovskú kružnicu. Zjavne musí platiť, že $k \geq 3$. Určite je k dispozícií aspoň osem farieb a už doteraz ofarbené vrcholy mohli obsadiť maximálne päť farieb. Zvolí sa teda nasledujúce farbenie:

$$\begin{aligned} f(w_{1,1}) &= c_1 \\ f(w_{1,d_1-1}) &= c_3 \end{aligned}$$

Ak existuje nejaký vrchol v_i, tak:

$$f(v_1) = c_2$$

A ostatné vrcholy v_i pre $i \in \{2, 3, \ldots, n\}$ ofarbíme nasledovne:

$$f(v_i) = c_{i+2}$$

Ak neexistuje žiadna synovská kružnica, potom je to veľmi jednoduché. Vrcholy v_i pre $i \in \{1, 2, \ldots, n\}$ sa ofarbia nasledovne:

$$f(v_i) \;=\; c_i$$

Zjavne doterajšie doplnenie pôvodného čiastočného farbenia spĺňa podmienky $L(2,1)$-farbenia. Zostáva vyriešiť zvyšné vrcholy na kružniciach.

Pozrime sa na i-tu kružnicu. Zavedieme nasledujúce označenie:

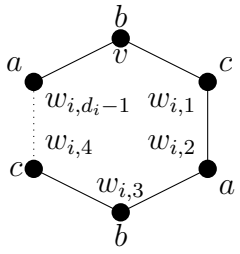
$$\begin{aligned} f(w_{i,d_i-1}) &= a \\ f(w_{i,1}) &= c \end{aligned}$$

Farby a a c sa vyberali tak, aby platilo $|a - c| \geq 2$. To znamená, že farby a, b a c sa dajú stále dookola opakovať. Rozoberme niekoľko prípadov.

- $d_i \equiv 0 \pmod 3$: V tomto prípade sa urobí presne to isté, čo v predchádzajúcom algoritme (Obr. 3.9):

$$\begin{aligned} f(w_{i,j}) &= a\ ,\ j \equiv 2 \pmod 3 \\ f(w_{i,j}) &= b\ ,\ j \equiv 0 \pmod 3 \\ f(w_{i,j}) &= c\ ,\ j \equiv 1 \pmod 3 \end{aligned}$$

  Pričom $j \in \{2, 3 \ldots, d_i - 2\}$.

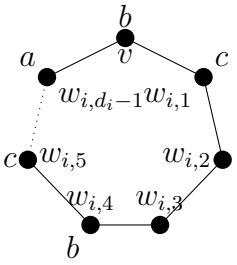


Obr. 3.9: $d_i \equiv 0 \pmod 3$

- $d_i \equiv 1 \pmod 3$: Aj tu sa budú farby a, b a c stále striedať, avšak vrchol $w_{i,2}$ zostane zatiaľ neofarbený (Obr. 3.10):

$$\begin{aligned} f(w_{i,j}) &= a\ ,\ j \equiv 0 \pmod 3 \\ f(w_{i,j}) &= b\ ,\ j \equiv 1 \pmod 3 \\ f(w_{i,j}) &= c\ ,\ j \equiv 2 \pmod 3 \end{aligned}$$

  Pričom $j \in \{3, 4, \ldots, d_i - 2\}$



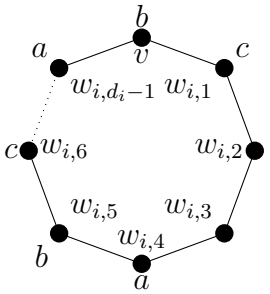
Obr. 3.10: $d_i \equiv 1 \pmod 3$

Vrchol $w_{i,2}$ má teraz práve dvoch ofarbených susedov. Každý z nich má inú farbu. Ďalej má najviac dva ofarbené vrcholy vo vzdialenosti dva, ale tieto vrcholy majú vždy rovnakú farbu. To znamená, že je pre vrchol $w_{i,2}$ obsadených najviac sedem farieb. Preto existuje farba, ktorá sa dá pre tento posledný vrchol použiť. Použije sa zo všetkých dostupných farieb tá najmenšia.

- $d_i \equiv 2 \pmod 3$: Opäť sa budú striedať farby a, b a c, ale zostanú neofarbené vrcholy $w_{i,2}$ a $w_{i,3}$ (Obr. 3.11):

$$\begin{aligned} f(w_{i,j}) &= a\ ,\ j \equiv 1 \pmod 3 \\ f(w_{i,j}) &= b\ ,\ j \equiv 2 \pmod 3 \\ f(w_{i,j}) &= c\ ,\ j \equiv 0 \pmod 3 \end{aligned}$$

Pričom $j \in \{4, 5, \ldots, d_i - 2\}$



Obr. 3.11: $d_i \equiv 2 \pmod 3$

Vrchol $w_{i,2}$ má práve jedného ofarbeného suseda a práve dva ofarbené vrcholy vo vzdialenosti dva. Z tohto dôvodu najviac päť farieb sa už pre tento vrchol nedá použiť, čo znamená, že existujú aspoň tri farby, ktoré sa dajú použiť. Rovnako to platí aj pre vrchol $w_{i,3}$. Z toho vyplýva, že určite existuje vhodná dvojica farieb pre tieto vrcholy. Spomedzi všetkých možných dvojíc sa vyberieme lexikograficky najmenšia dvojica.

Ukázali sme, že existuje funkcia OFARBI_SYNOVSKÉ_BLOKY, čím sme zároveň dokázali platnosť horného odhadu. □

Lepší výsledok pre túto skupinu kaktusov už neexistuje. Takéto ohraničenie platí aj pre stromy a stromy sú podtriedou kaktusov. Navyše vieme, že toto ohraničenie je pre stromy tesné.

Vieme, že ak si zvolíme ľubovoľný maximálny stupeň grafu, vždy existuje strom, pre ktorý platí $\lambda(G) = \Delta + 2$. Stačí zobrať taký, ktorý má vrchol s maximálnym stupňom a aspoň dvaja jeho susedia majú tiež maximálny stupeň. Ak by sa takýto graf dal ofarbiť len farbami $0, 1 \ldots, \Delta + 1$, potom nutne každý jeho vrchol s maximálnym stupňom musí mať niektorú z krajných farieb. V našom prípade ale máme tri vrcholy s maximálnym stupňom, ktoré musia mať rôzne farby, avšak krajné farby sú len dve. Preto sa taký graf nedá $(\Delta + 1)$-$L(2, 1)$-ofarbiť.

Záver

Cieľom práce bolo nájsť polynomiálne algoritmy pre rozhodovaciu verziu problému $L(2,1)$-farbenia pre cyklové stromy a kaktusy. Ďalším cieľom bolo nájsť tesné ohraničenie $\lambda(G)$ pre grafy z týchto tried.

V zadaní práce bolo medzi cieľmi spomenuté aj zlepšenie výsledku pre t-takmer stromy. V tomto prípade zamýšľaný prístup zlyhal a nič ďalšie sa nám k tomu neporadilo vymyslieť.

Prvý cieľ sme naplnili len čiastočne. Podarilo sa nám nájsť polynomiálny algoritmus pre triedu cyklových stromov. Ukázali sme však, že algoritmus, ktorý sme našli, sa dal triviálnym spôsobom rozšíriť pre všetky kaktusy. Ak mali tieto kaktusy konštantou ohraničený počet kružníc, na ktorých môže ležať jeden vrchol, tak bol tento algoritmus polynomiálny. Nepodarilo sa nám však nájsť polynomiálny algoritmus pre všetky kaktusy.

Pri ohraničeniach sme boli úspešnejší. Podarilo sa nám dokázať až dve ohraničenia. Prvé z nich bolo tesné pre kaktusy s maximálnym stupňom 3 a teda aj pre cyklové stromy. Druhé bolo tesné pre všetky kaktusy s maximálnym stupňom aspoň 5. Pre kaktusy s menším maximálnym stupňom ako tri bol tento problém vyriešený už predtým. Jedine pre kaktusy s maximálnym stupňom 4 nevieme, či máme tesné ohraničenie alebo nie.

Zostáva teda otvorená otázka, či existuje polynomiálny algoritmus pre problém $L(2,1)$-farbenia pre kaktusy. Taktiež zostáva otvorené, či existujú kaktusy s maximálnym stupňom 4, pre ktoré $\lambda(G) = \Delta + 3$.

Literatúra

[1] Graphviz - Graph Visualization Software. `http://www.graphviz.org/`. 2016.

[2] pycosat 0.6.1. `https://pypi.python.org/pypi/pycosat`. 2016.

[3] Python. `https://www.python.org/`. 2016.

[4] python-igraph. `http://igraph.org/python/`. 2016.

[5] H.L. Bodlaeder, T. Kloks, R.B Tan, and J. van Leeuwen. Approximations for lambda-colorings of graphs. *Computer Journal*, 47:193–204, 2004.

[6] Gerard J Chang and David Kuo. The L(2,1)-labeling problem on graphs. *SIAM Journal on Discrete Mathematics*, 9(2):309–316, 1996.

[7] Kalyani Das. Cactus graphs and some algorithms. *CoRR*, abs/1408.4005, 2014.

[8] Nicole Eggemann, Frédéric Havet, and Steven D Noble. k-L(2, 1)-labelling for planar graphs is NP-complete for k $\leq$ 4. *Discrete Applied Mathematics*, 158(16):1777–1788, 2010.

[9] Jiří Fiala, Petr A Golovach, and Jan Kratochvíl. *Distance constrained labelings of graphs of bounded treewidth*. Springer, 2005.

[10] Jiří Fiala, Ton Kloks, and Jan Kratochvíl. Fixed-parameter complexity of λ-labelings. *Discrete Applied Mathematics*, 113(1):59–72, 2001.

[11] Daniel Gonçalves. On the L(p,1)-labelling of graphs. *Discrete Mathematics*, 308(8):1405–1414, April 2008.

[12] Jerrold R Griggs and Roger K Yeh. Labelling graphs with a condition at distance 2. *SIAM Journal on Discrete Mathematics*, 5(4):586–595, 1992.

[13] William K Hale. Frequency assignment: Theory and applications. *Proceedings of the IEEE*, 68(12):1497–1514, 1980.

[14] Toru Hasunuma, Toshimasa Ishii, Hirotaka Ono, and Yushi Uno. A linear time algorithm for L(2, 1)-labeling of trees. *Algorithmica*, 66(3):654–681, 2013.

[15] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.

[16] Robert Janczewski, Adrian Kosowski, and Micha l Ma lafiejski. The complexity of the L(p, q)-labeling problem for bipartite planar graphs of small degree. *Discrete Mathematics*, 309(10):3270–3279, 2009.

[17] Gang Li and Frank Ruskey. The advantages of forward thinking in generating rooted and free trees. pages 939–940, 1999.

[18] Michael Molloy and Mohammad R. Salavatipour. A bound on the chromatic number of the square of a planar graph. *J. Comb. Theory, Ser. B*, 94(2):189–213, 2005.

[19] F.S. Roberts. Private Communication to J. Griggs.

[20] Jan van den Heuvel and Sean McGuinness. Coloring the square of a planar graph. *Journal of Graph Theory*, 42(2):110–124, 2003.

[21] Wei-Fan Wang and Ko-Wei Lih. Labeling Planar Graphs with Conditions on Girth and Distance Two. *SIAM J. Discret. Math.*, 17(2):264–275, February 2004.