VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Projekt IAL, 2018Z

Obarvení grafu

Projekt č.6

05. prosince 2018

Tým:
Adámek Josef, xadame42
Barnová Diana, xbarno00
Vanický Jozef, xvanic09
Weigel Filip, xweige01

Obsah

1 Zadanie

Vytvoriť program pre hľadanie **minimálneho zafarbenia neorientovaných grafov.** Ak existuje viacej riešení, stačí nájsť len jedno. Výsledky prezentujte vhodným spôsobom. Súčasťou projektu bude načítavanie grafov zo soúboru a vhodné testovacie grafy. V dokumentácii uveďte teoretickú zložitosť úlohy a porovnejte ju s experimentálnymi výsledkami.

2 Práca v týme

2.1 Príprava a plán

Pred začiatkom vývoja boli vziate do úvahy schopnosti každého člena týmu, na základe ktorých boli pridelené úlohy. Časové rámce jednotlivých častí boli len orientačné pre udržanie prehľadu nad postupom a zostávajúcim časom do ukončenia projektu. Byl vytvorený privátny repozitár na Githube, fungujúci na technológii Git pre ľahké verzovanie projektu. Možnosti vytvorit jednotlivé vetvy pro každú soúčasť a nezávislý vývoj a pre prípadnú orientaciu mezi verziami či vrátenia k predchádzjúcej verzii. Pro statickú analýzu kódu sme využili službu Codacy.

2.2 Postup a rozdelenie práce

Pri tomto projekte sme sa rozhodli využiť metódu Test-driven development a to z dôvodu zefektivnenia celeho vývoja. Najskôr bol napísaný test k súčasti, která bola až následně naprogramovaná. Tak sa zabránilo k zavedeniu chybu do už funkčnej časti.

2.3 Komunikácia

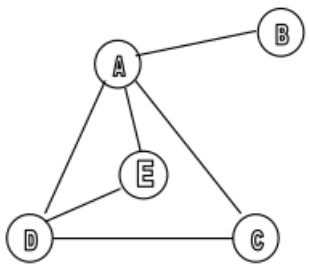
Komunikácia v týme prebiehala prostredníctvom služby Messenger ale aj osobne na pravidenlých stretnutiach, ktoré sme si dohodli už na začiatku riešenia projektu.

3 Teoretická časť

3.1 Priblíženie problematiky

Graf (všeobecne) je definovaný trojicou G=(N, E, I), kde

- N je množina uzlov, ktorým je možné priradiť hodnotu
- E je množina hrán, ktorým je možné priradiť hodnotu. Každá hrana spojuje dva uzly a môže byť orientovaná. Ak je hrana orientovaná tak sa hovorí o orientovanom grafe. V našom prípade sa teda jedná o **neorientovaný graf**, pretože hrany sú neorientované.
- I je množina spojenia, ktorá jednoznačne určuje dvojice uzlov daného grafu



	A	B	C	D	E
A	0	1	1	1	1
B	1	0	0	0	0
C	1	0	0	1	0
D	1	0	1	0	1
E	1	0	0	1	0

Obrázek 1: Príklad neorientovaného grafu a jeho matice

Implementácia neorientovaného grafu je pomocou matice koincidencie t.j spojenia. Táto matica je symetrická podľa hlavniej diagonály.

Zafarbením grafu rozumieme priradenie farieb uzlom grafu, pričom žiadne dve susedné uzly nesmú byť zafarbené rovnako. Minimálny počet použitých farieb se nazýva **chromatické číslo**. Práve toto chromatické číslo v našom projekte hľadáme.

[EDIT: Právě řešení s tímto chromatickým číslem v našem projektě hledáme.]

3.2 Popis projektu

Spúšťanie: ./main -f FILENAME [-h] [-b], kde

- -f FILENAME je názov súboru, ktorý obsahuje maticu grafu
- -b je volitelný parameter, EDIT: který spustí "brief mode"programu, takže program tiskne na standartní výstup pouze chromatické číslo zpracovaného grafu
- -h je volitelný parameter, ktorý má na starosti vytlačiť spravu pre používanie na standartní výstup

4 Testování projektu

Projekt sme najskôr testovani na menších testovacích grafoch, ktoré sme vytvorili a vedeli aj odkontrolovať. Neskôr sme na generovanie grafov vytvorili skript, ktory vytvoril aj väčšie grafy. Testovanie prebiehalo na platforme Ubuntu, MacOS a FreeBSD.

5 Implementácia

[TRANSLATE: Všechno pod tímto přeložit]

5.1 Algoritmus

Jako první metodu pro obarvování grafu jsme použili metodu "backtracking"(slepé prohledávání se zpětným navracením), jelikož pro náš typ problému (constraint satisfaction problem) je tato metoda úplná i optimální. Později jsme však zjistili, že je tato metoda pomalá i s našimi vylepšeními. Proto jsme jako druhou metodu použili "forward checking"(kontrola dopředu), která se dá chápat jako vylepšení metody backtracking, protože forward checking se jako metoda dívá dopředu a je schopná rychleji odhalit kombinace barev, které nevedou k řešení problému. Metoda je tak schopná prořezávat stavový strom a není proto potřeba používat tolik zpětného navracení. Tuto metodu jsme také konkrétně implementovali pomocí rekurze.

5.2 Struktury

Dvě primární struktury nabízející se pro reprezentaci neorientovaného grafu v programu jsou seznam sousedů a matice sousednosti. My jsme zvolili matici sousednosti a to jak pro formát načítání grafu do programu, tak i pro následnou práci s tímto grafem. V programu je matice sousednosti implementována jako jediné pole pro rychlejší přístup do ní.

Pro reprezentaci samotného uzlu jsme si vytvořili strukturu "Node", která zahrnuje id uzlu, jeho přidělenou barvu a množinu legálních barev, která je důležitá pro metodu forward checking. Tyto množiny barev jednotlivých uzlů jsme implementovali pomocí polí typu bool, ve kterých index představovuje barvu a hodnota True nebo False značí, jestli je tato barva pro tento uzel legální. Takto reprezentované uzly jsou poté seřazeny v poli typu Node, přičemž jejich index a id jsou stejné, aby bylo možné k nim rychle přistupovat.

5.3 Analýza složitosti algoritmu

Problém minimálního obarvení grafu (anglicku k-coloring) je takzvaný "NP-complete problem", což ve výsledku znamená, že hledání optimálního řešení tohoto problému má exponenciální složitost.

Metoda forward checking má časovou složitost $O(m^n)$, přičemž n v kontextu problému hledání chromatického čísla znamená počet uzlů v grafu a m počet barev. A dále jelikož při hledání chromatického čísla počet barev může být stejný jako počet uzlů, časová složitost je vlastně $O(n^n)$.

Níže přikládáme graf, kde porovnáváme teoretickou složitost algoritmu s našimi experimentálními výsledky:

6 Záver

Přestože jsme se snažili vytvořit co nejrychlejší program pro hledání optimálního řešení pro hledání chromatického čísla, jsme si vědomi dalších metod, které by program dále zrychlily. Pokusili jsme se například vylepšit metodu forward checking různými heuristikami, ale jelikož už se blížilo datum odevzdání, tak jsme to nestihli a odevzdali projekt bez nich.

7 Zdroje

- Opora a prezentace předmětu IAL
- Prezentace předmětu IZU
- Artificial Intelligence: A Modern Approach (3rd Edition), Peter Norvig, Stuart J. Russell
- Exponential time algorithms for graph coloring, Uriel Feige, Lecture notes, 2011