

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



Projekt IAL, 2018Z

Zafarbenie grafu

Projekt č.6

5. decembra 2018

Tím:

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 viacero riešení, stačí nájsť iba jedno. Výsledky prezentujte vhodným spôsobom. Súčasťou projektu bude načítavanie grafov zo súboru a vhodné testovacie grafy. V dokumentácii uveďte teoretickú zložitosť úlohy a porovnejte ju s experimentálnymi výsledkami.

2 Práca v tíme

2.1 Príprava a plán

Pred začiatkom vývoja boli vzaté do úvahy schopnosti každého člena tímu, na základe ktorých boli pridelené úlohy. Časové rámce jednotlivých úloh boli len orientačné pre udržanie prehľadu nad postupom a zvyšným časom do ukončenia projektu. Bol vytvorený privátny repozitár na platforme Git-Hub, fungujúci na technológii Git pre ľahké verzovanie projektu a možnosti vytvoriť jednotlivé vetvy pre každú verziu a nezávislý vývoj a pre prípadnú orientáciu medzi verziami či vrátenia k predchádzajúcej verzii. Pro statickú analýzu kódu sme využili službu Codacy a službu CircleCI pre automatické spúšťanie testov.

2.2 Postup a rozdelenie práce

Pri tomto projekte sme sa rozhodli využiť metódu Test-driven development a to z dôvodu zefektívnenia celeho vývoja. Najskôr bol napísaný test k verzii, ktorá bola až následne naprogramovaná. Tak sa zabránilo k zavedeniu chyby do už funkčnej verzii nášho programu.

2.3 Komunikácia

Komunikácia v tíme prebiehala prostredníctvom služby Messenger, ale aj osobne na pravidelných stretnutiach, ktoré sme si naplánovali už na začiatku riešenia projektu.

3 Teoretická časť

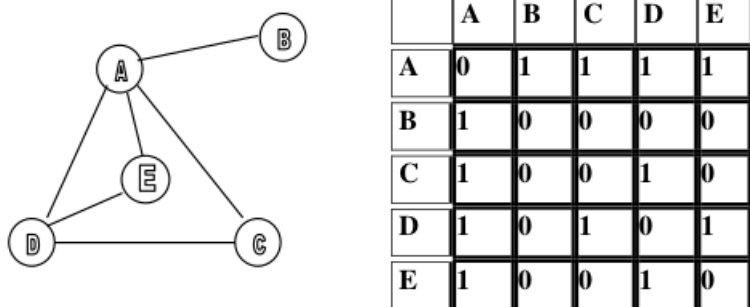
3.1 Priblíženie problematiky

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

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

Zafarbením grafu rozumieme priradenie farieb uzlom grafu, pričom žiadne dva susedné uzly nesmú byť zafarbené rovnako. Minimálny počet použitých farieb sa nazýva **chromatické číslo**. Práve riešenie s týmto chromatickým číslom v našom projekte hľadáme.



Obr. 1: Príklad neorientovaného grafu a jeho matice

4 Spustenie programu

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

- `-f FILENAME` určuje názov súboru, ktorý obsahuje maticu grafu
- `-b` je voliteľný parameter, ktorý spustí "brief mode" programu, takže program vypíše na štandardný výstup iba chromatické číslo spracovaného grafu
- `-h` je voliteľný parameter, ktorý má na starosti vypísať na štandardný výstup nápovedu pre užívateľa

5 Testovanie projektu

Pre projekt sme najprv vytvorili menšie testovacie grafy, ktorých chromatické číslo sme vedeli sami určiť. Tie sme neskôr použili pri testovaní správnosti každej novej verzie nášho programu, pomocou nami vytvoreného testovacieho skriptu. Neskôr sme na generovanie grafov vytvorili skript, ktorý dokáže vygenerovať ľubovoľný neorientovaný graf, podľa zadaných parametrov. Testovanie prebiehalo na platforme Ubuntu, MacOS a FreeBSD.

6 Implementácia

6.1 Algoritmus

Ako prvú metódu pre zafarbenie grafu sme použili metódu slepého prehľadávania so spätným navrátnom (angl: backtracking). Pre náš typ problému (constraint satisfaction problem) sme túto metódu vybrali, pretože je **úplná** a zároveň **optimálna**. Neskôr sme zistili, že je táto metóda pomalá aj napriek našim pokusom o jej vylepšenie. Z toho dôvodu sme ako druhú použili metódu doprednej kontroly (angl: forward checking), ktorá sa dá chápať ako vylepšená metóda slepého prehľadávania. Metóda doprednej kontroly sa ako metóda "pozerá" dopredu a tým je schopná rýchlejšie odhaliť kombináciu farieb, ktoré nevedú k riešeniu problému. Metóda je tak schopná redukovať stavový strom a vďaka tomu nieje potrebná používať prílišného spätného navrátenia. Túto metódu sme konkrétne implementovali pomocou rekurzcie.

6.2 Štruktúry

Dve primárne štruktúry ideálne pre reprezentáciu neorientovaného grafu v programe sú zoznam susedov a matica susednosti. Zvolili sme maticu susednosti a to z dôvodu vhodného formátu načítania grafu zo vstupu do programu, ale aj kvôli následnej práci s týmto grafom. V programe je matica susednosti implementovaná ako jediné pole a to pre rýchlejší prístup do tejto matice.

Pre reprezentáciu samotného uzlu sme si vytvorili štruktúru "Node", ktorá zahŕňa id uzlu, jeho pridelenú farbu a množinu legálnych farieb. Táto množina je dôležitá pre metódu doprednej kontroly. Množiny farieb jednotlivých uzlov sme implementovali pomocou polí typu bool, v ktorých index predstavuje farbu a hodnota true alebo false

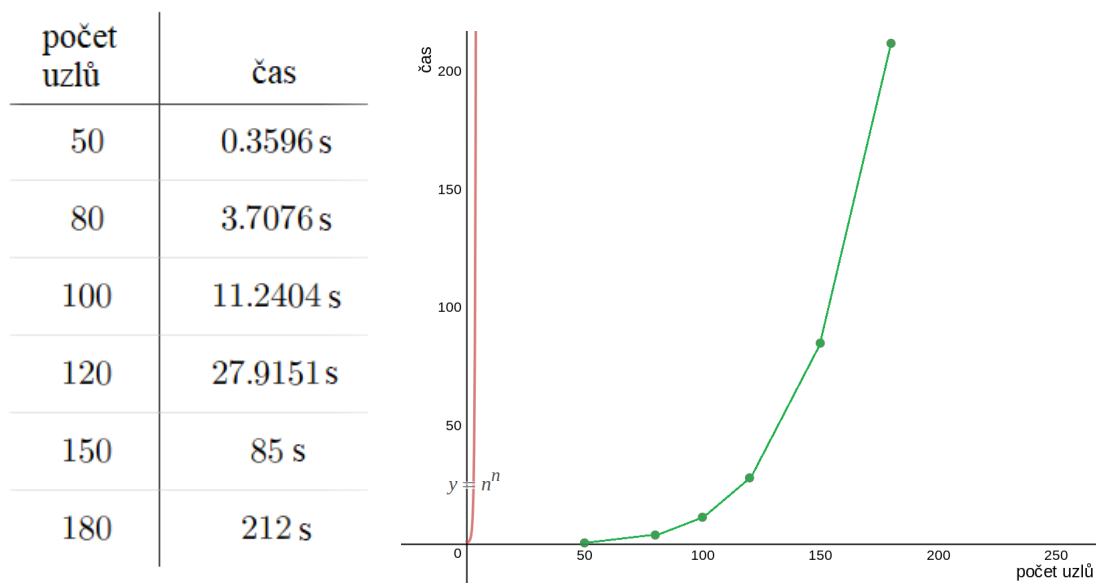
označuje, či je táto farba pre daný uzol legálna. Takto reprezentované uzly sú potom zoradené v poli typu "Node", pričom ich index a id sú rovnaké, aby bolo možné k uzlom rýchlejšie pristupovať.

6.3 Analýza zložitosti algoritmu

Problém minimálneho zafarbenia grafu (angl: k-coloring) je takzvaný "NP-úplný problém, čo vo výsledku znamená, že hľadanie optimálneho riešenia problému má exponenciálnu zložitosť".

Metóda doprednej kontroly má časovú zložitosť $O(m^n)$, pričom n v kontexte problému hľadania chromatického čísla vyjadruje počet uzlov v grafe a m vyjadruje počet farieb. Pretože pri hľadaní chromatického čísla môže byť počet farieb rovnaký ako počet uzlov, je vlastne časová zložitosť $O(n^n)$.

Priložený graf vyjadruje porovnanie teoretickej časovej zložitosti algoritmu s našimi experimentálnymi výsledkami.



Pre stabilitu výsledkov sme použili iba grafy, v ktorých sú spojené všetky uzly so všetkými.

7 Záver

Sme si vedomí, že aj napriek tomu, že sme sa snažili vytvoriť čo najrýchlejší program pre hľadanie optimálneho riešenia pre hľadanie chromatického čísla v neorientovaných grafoch, existujú ďalšie metódy, ktoré by mohli hľadanie zrýchliť. Jedným z našich pokusov bolo napríklad vylepšiť metódu doprednej kontroly rôznymi heuristikami, avšak kvôli blížiacemu sa termínu odovzdania projektu sme tieto heuristiky nestihli implementovať a odovzdali sme projekt bez nich.

8 Zdroje

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