Vizualizace grafových algoritmů

Jakub Arnold

Tématem práce je vizualizace grafových algoritmů, konkrétně prohledávání do hloubky, do šířky, Dijkstrův algoritmus, hledání Eulerovské kružnice (nakreslení grafu jedním tahem) v neorientovaných grafech, a Jarníkův algoritmus.

Všechny algoritmy jsou krokovatelné uživatelem, který zadá libovolný graf (viz dále), a poté může krok po kroku sledovat, jak algoritmus probíhá, a případně ho může pustit znovu z jiného počátečního vrcholu.

1 Kompilace a spuštění

Pro kompilaci je potřeba překladač, který umí C++11 (testováno na GCC 5.x, Clang 3.6 a MSVC 14). Dále je potřeba Qt, ideálně verze 5, ale program funguje i pod verzí 4 (která je např. v labu). Kompilace a spuštění se pak provede následovně:

- `$ qmake .`
- `$ make`
- `$ ./build/debug/graphite`

Testováno na různých distribucích Linuxu, včetně labu, OS X a zlehka na Windows. U starších překladačů (které neumí `-std=c++11`, ale umí `-std=c++0x`) je možné tento flag změnit v souboru `graphite.pro`, viz. komentáře uvnitř souboru u položky `QMAKE_CXXFLAGS`.

Na konci dokumentu je případně odkaz na zkompilovanou 64bit verzi pro Linux.

2 Základní ovládání

Program se ovládá klávesnicí i myší, kde všechny příkazy jdou zadat buď přes hlavní menu a nebo pomocí klávesových zkratek.

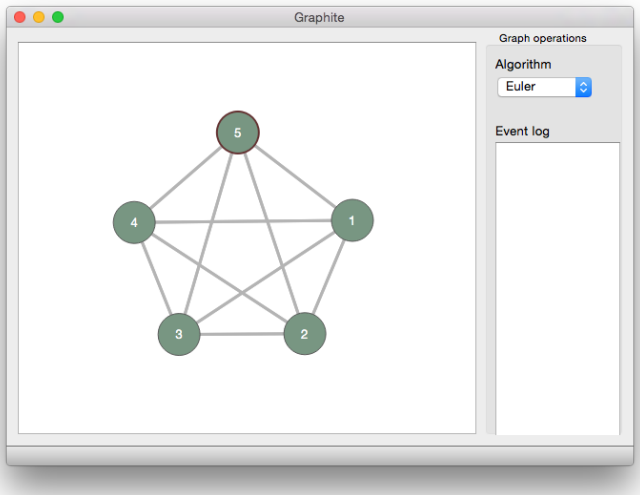
Zde je úplný seznam klávesových zkratek:

- Vygenerování náhodného grafu `Ctrl-R G`.
- Náhodná oritentace hran `Ctrl-R D`.
- Vygenerování náhodného Eulerovského grafu `Ctrl-R E`.
- Náhodné ohodnocení hran `Ctrl-R W`.
- Odstranění orientace hran `Ctrl-R U`.
- Přidání vrcholu `A`.
- Spojení dvou vrcholů hranou `C`. Je potřeba napřed vybrat první, zmáčknout `C`, vybrat druhý, a zmáčknout `C` znovu.
- Smazání vrcholu nebo hrany `D`. (nejprve je potřeba hranu nebo vrchol vybrat kliknutím myši.)
- Označení počátečního vrcholu `S`.
- Start/Restart algoritmu `R`.
- Krok algoritmu `N`.
- Změna orientace hrany `O` (nejprve je potřeba hranu vybrat kliknutím myši.)
- Nastavení ohodnocení hrany `1-9`, nejprve je ale potřeba mít vybraný Dijkstrův algoritmus, jinak se ohodnocení hran nezobrazí, a poté kliknout na vybrané ohodnocení (*ne na hranu*).

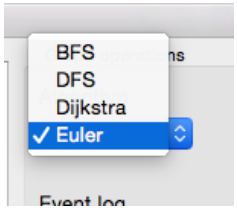
Grafy je také možné uložit do souboru `Ctrl-S` a znovu načíst `Ctrl-O`, přičemž se zachová i rozložení vrcholů v prostoru (pokud je uživatel přesunul.)

3 Používání programu

Nejjednodušší je vybrat jeden z přiložených grafů v souboru, a otevřít jej přes `File -> Open`, např. kompletní graf na 5 vrcholech v souboru `examples/k5.g`.



V seznamu algoritmů vybrat Eulerovský tah.



A kliknout na libovolný vrchol, vybrat ho jako počáteční (stisknutím `F`), inicializovat algoritmus (stisknutím `R`), a poté již krokovat stisknutím `N`. V libovolnou chvíli je možné znovu stiknout `R`, čímž se algoritmus resetuje a začne pracovat odznova. *Pokud uživatel graf jakkoliv změní v průběhu algoritmu, je nutné algoritmus resetovat stisktnutím* ***R****.*

Většina důležitých informací které program provádí jsou vypisovány do logu v pravé straně GUI (a některé navíc na STDOUT). Log v GUI je editovatelný text, a lze ho tedy označit myší a smazat.

4 Generování náhodných grafů

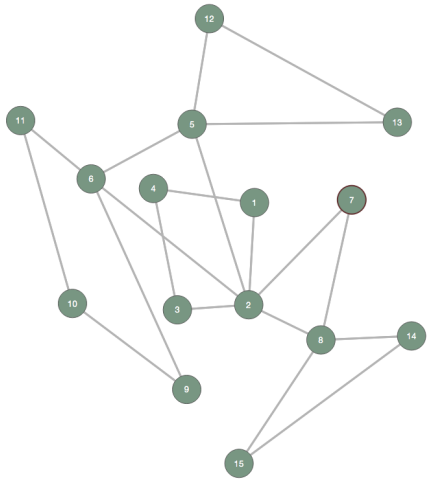
Aby bylo možné aplikaci jednoduše používat, obsahuje možnost vygenerování náhodného souvislého grafu (žádný ze zabudovaných algoritmů nedává smysl vizualizovat na nesouvislých grafech.) Graf je generován následujícím způsobem:

- vygeneruje se 10-15 vrcholů, které se postupně spojí hranami, dohromady tvořící jednu velkou cestu
- každému vrcholu se s pravděpodobností 2/5 přidělí jedna další náhodná hrana

Takto vygenerovaný graf bude vždy souvislý, a díky malému počtu hran i relativně přehledný. Graf je vždy generovaný jako neorientovaný. Pokud si uživatel přeje, může poté náhodně zorientovat hrany (`Ctrl-R, D`).

5 Generování Eulerovských grafů

Protože pro Eulerovské grafy musí platit, že každý vrchol má sudý stupeň, je také jednoduché nahlédnout, že musí ležet na nějaké kružnici. Generování grafu tedy probíhá tak, že se nejprve vytvoří jednovrcholový graf, a potom se 5-7krát vybere náhodný vrchol z grafu, a přilepí se na něj další kružnice délky 3-5 (pro přehlednost.) Výsledný graf pak vypadá např. takto



Obrázek 1: Náhodný Eulerovský graf

Takto vygenerovaný graf má opět výhodu, že je díky menšímu počtu hran přehledný.

5.1 Rozmístění vrcholů

Při generování náhodného grafu jsou vrcholy vždy rozmístěny na spirálu, která se rozvijí *zevnitř ven*. Pro grafy výše zmíněné náhodně generované grafy je toto rozložení relativně blízko tomu, co by si uživatel mohl předstatovat, a stačí zpravidla pouze přemístit pár vrcholů uvnitř spirály, aby se příliš mnoho hran nekřížilo.

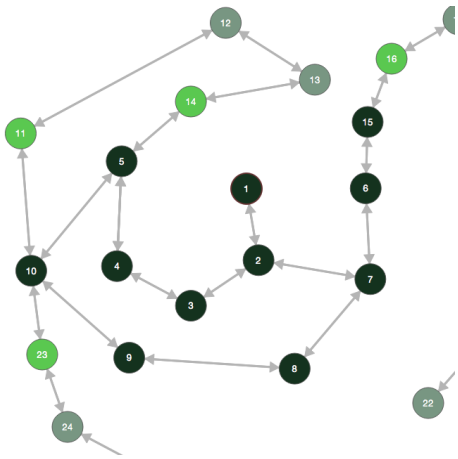
6 Algoritmy

Všechny algoritmy jsou implementované jako stavový automat, ktery se stiskem `R` přesune do počátečního stavu, a stiskem `N` postupně krokuje, až dojde do koncového stavu, kdy algoritmus doběhl. Proto jsem zvolil zásobníkovou variantu DFS místo rekurzivní, aby šlo jednoduše ovládat průběh algoritmu.

Všechny algoritmy pracují pouze na souvislých grafech. Pokud je graf nesouvislý, algoritmus proběhne pouze na komponentě souvislosti ve které leží počáteční vrchol. Toto řešení jsem zvolil převážně proto, že umožnuje uživateli větší graf rozpojit a sledovat průběh pouze na části grafu, aniž by ho musel celý smazat a pak znovu vytvářet.

6.1 DFS, BFS

Pro porovnání prohledávání do hloubky a do šířky je nejlepší zvolit stejný graf, a na něm pozorovat, jak se průběh jednotlivých algoritmů liší. Oba používají stejnou konvenci, a to že nenavštívený vrchol je tmavě zelený, otevřený je světle zelený a uzavřený je černý.



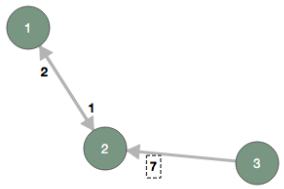
Obrázek 2: Barvy vrcholů

Jak DFS tak BFS umí pracovat s orientovanými grafy. Orientace hrany se změní označením hrany myší a stiskem `O`. Pro vrcholy `A` a `B` se postupně mění

typ hrany na `A -> B`, `A <- B`, a `A <-> B`.

6.2 Dijkstrův algoritmus

Dijkstrův algoritmus opět funguje i na orientovaných grafech, pričemž navíc zobrazuje i ohodnocení hran.

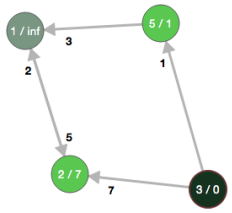


Obrázek 3: Ohodnocení hran

Váha hrany je vždy zobrazena ve směru kam hrana ukazuje, a tedy pro *obousměrné*, resp. *neorientované* hrany jsou zobrazena ohodnocení dvě, každé jedním směrem.

Hranám lze nastavovat hodnoty v rozsahu 1-9, což se provede kliknutím na ohodnocení hrany a stiskem patřičné klávesy 1-9. Označení se zobrazí tečkovaným čtvercem okolo ohodnocení hrany (viz. obrázek).

Samotný dijkstruv algoritmus se poté zobrazuje podobně jako u DFS/BFS. Nenavštívené vrcholy jsou tmavě zelené, otevřené jsou světle zelené a uzavřené jsou černé. Navíc se však u vrcholů zobrazuje jejich vzdálenost od počátečního vrcholu, a to tak, že se v popisku vrcholu zobrazí `číslo vrcholu / vzdálenost od zdroje`. Vrcholy zatím neobjevené mají nekonečnou vzdálenost, reprezentováno stringem `inf`, viz obrázek.



Obrázek 4: Průběh Dijkstrova algoritmu

6.3 Eulerovská kružnice

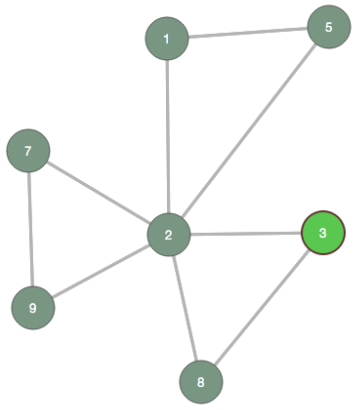
Algoritmus pro hledání Eulerovské kružnice se trochu liší od prvních tří, a to tím, že funguje pouze na neorientovaných grafech, kde stupeň všech vrcholů je sudý. Pokud je spuštěn na grafu který není Eulerovský, nemusí správně fungovat.

Pro implemntaci jsem zvolil Fleuryho algoritmus, který funguje následovně:

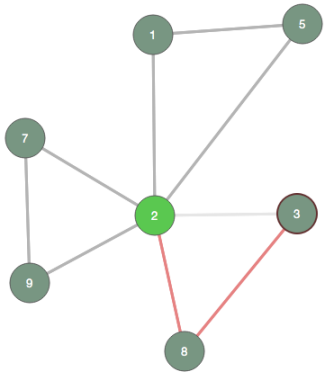
1. začni v libovolném vrcholu
2. vyber hranu která není most, pokud jsou všechny mosty tak vyber libovolnou
3. označ hranu jako smazanou, a přesuň se do vrcholu kam hrana ukazuje a opakuj krok 1.

Tento algoritmus je velmi přímočarý, a ze všech implementací které jsem zkoušel vede na nejnázornější vizualizaci, a to proto, že během svého průběhu může označovat hrany které jsou mosty, a uživatel tak snadno vidí, jak se algoritmus rozhoduje kudy půjde.

Na začátku algoritmu sice žádné mosty nemohou existovat, ale už po prvním kroku dojde ke *smazání* jedné hrany, a tím mohou nějaké mosty vzniknout. Viz obrázek



a po prvním kroku už máme nalezeny dva mosty.



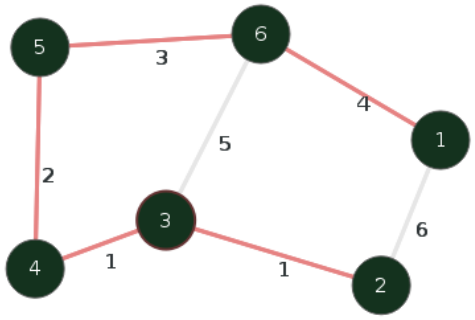
Hledání mostů je nutné provést po každém kroku algoritmu, a probíhá pomocí DFS klasifikace, konkrétně tak, že se prohledá celý graf pomocí DFS, najdou se vrcholy, které leží na nějaké kružnici (pomocí zpětných hran v DFS stromu), a ty co neleží jsou označený jako mosty.

Složitost algoritmu je tedy kvadratická, protože pro každý krok je nutné provést celé DFS na odhalení mostů. Existují sice alternativní algoritmy, které jsou ve výsledku rychlejší, ale z hlediska vizualizace mi přišel Fleuryho algoritmus nejlepší.

6.4 Jarníkův algoritmus

Poslední algoritmus který program implementuje je Jarníkův-Primův algoritmus pro hledání nejlehčí kostry neorientovaného grafu. Při spuštění algoritmus automaticky odstraní orientace všech hran, a není tedy nutné graf nijak dopředu opravovat.

Algoritmus lze spustit z libovolného vrcholu a vždy by se měl dobrat správného cíle, a to najít nejlehčí kostru.



Obrázek 5: Nalezená nejlehčí kostra

Samotná kostra je poté zobrazena červeně, a hrany které v kostře neleží jsou šedivé. Během průběhu algoritmu jsou otevřené vrcholy obarveny světle zelenou, a uzavřené vrcholy černé.

Odkazy

[1] `https://github.com/darthdeus/graphite-gui/releases/download/v0.1.0/graphite-x86_64`