Vysoké učení technické v Brně
Fakulta informačních technologií

Algoritmy
Rovinnost grafu 2022/2023

Technická zpráva

Rastislav Mazúr (xmazur09)
Jakub Kavka (xkavka02)
Ondřej Podroužek (xpodro03)
Vladimír Mečiar (xmecia00)

Brno, 7. prosince 2022

Obsah

1 Zadání varianty

Graf je rovinný, jestliže lze jeho vrcholy (jako body) a hrany (jako spojité křivky) umístit do roviny tak, aby se žádné dvě hrany nekřížily.

Vytvořte program, který rozhodne, zda je graf rovinný.
Součástí projektu bude načítání grafů ze souboru a vhodné testovací grafy. V dokumentaci uveďte teoretickou složitost úlohy a porovnejte ji s experimentálními výsledky. Nepředpokládá se, že by nalezení rozmístění vrcholů a hran v rovině bylo součástí řešení. Pozor! Otestování rovinnosti grafu je netriviální problém. Můžete použít algoritmus založený na tvrzení pana Kuratowského (1930): Rovinný graf neobsahuje podgraf izomorfní s grafy K(5) nebo K(3,3).

2 Práce v týmu

2.1 Rozdělení práce

První schůzku jsme měli v říjen. Touto dobou jsme se seznamovali a začali jsme rozebírat zakladní problematiku. Součástí bylo aj určení algoritmu na odhalení rovinnost grafu. Od této časti jsme naneštěstí dlouho nijak nepokročili, protože jsme měli na starosti jiné povinosti související se školou

Navrh a implementacia automatu pre prijmanie znakov

- Ondrej Podrouzek

Testovanie automatu

- Rastislav Mazur

Navrh, implementacia a testovanie datovej struktury pre ukladanie grafu

- Vladimír Mečiar

Implementacia finalnej logiky programu

- Jakub Kavka

Testovanie finalnej logiky programu

- Rastislav Mazur

Tvorba dokumentace

- Vladimír Mečiar

Revizia kodu a dokumentacie

- Rastislav Mazur, Ondrej Podrouzek

2.2 Verzování

Pro sdílení a verzování kódu jsme použili verzovací systém GitHub.

V projektu jsme pracovali na více částech současně. Pro každou větší část jsme vytvářeli vlastní větve, které jsme po dokončení mergovali do větve `develop`. Do finální verze repozitáře `main` jsme prepojili až finální otestovanou verzi a následně jsme přímo v ní prováděli jen poslední úpravy.

2.3 Komunikace

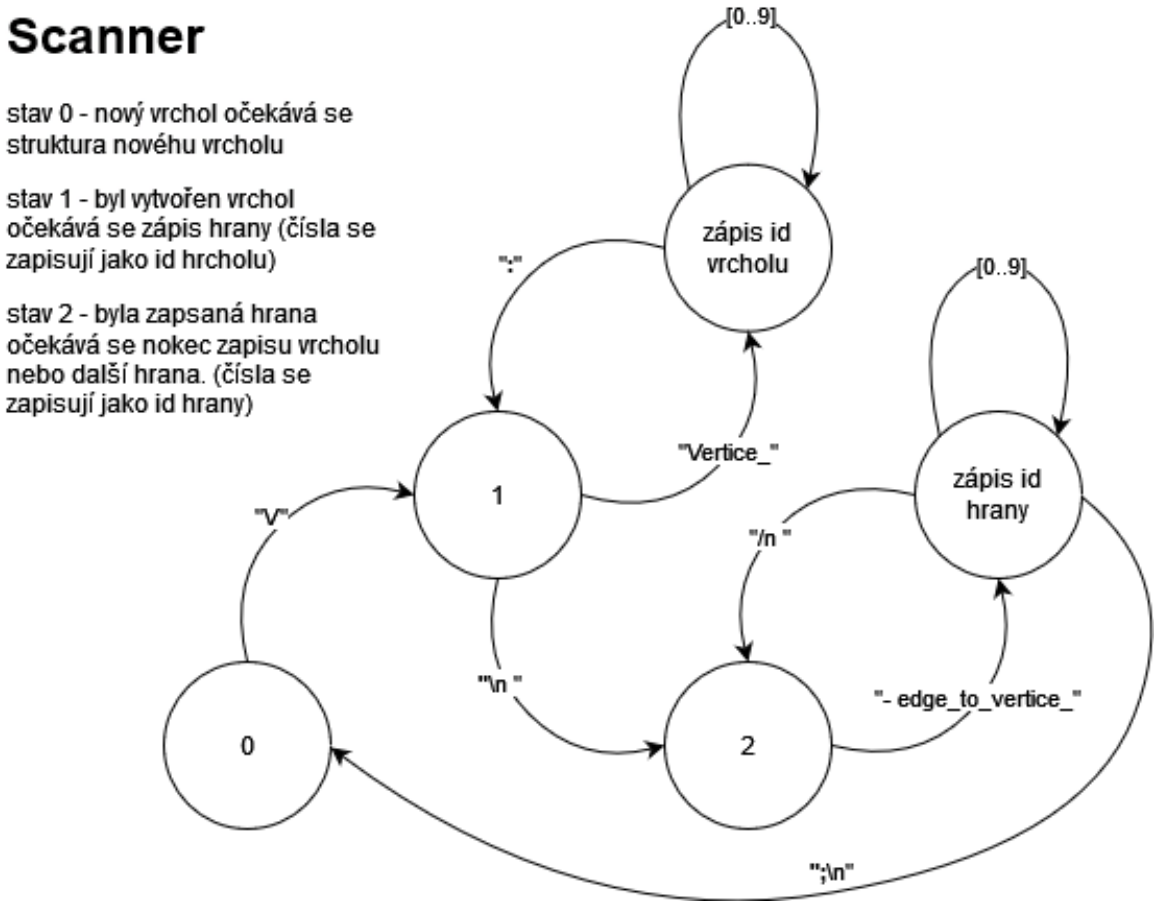
Jako komunikační platformy jsme využili *Discord*, který sloužil jako hlavní komunikační kanál.

3 Načítání dat

Struktůra kterou potřebujeme načítat (graf), je definovaná jako množina vrcholů a hran. Pro naše potřeby jsme použili uložení v textu s formátem.

```
Vertice_id:
  - edge_to_vertice_id
  - edge_to_vertice_id
  - edge_to_vertice_id
  - edge_to_vertice_id;
```

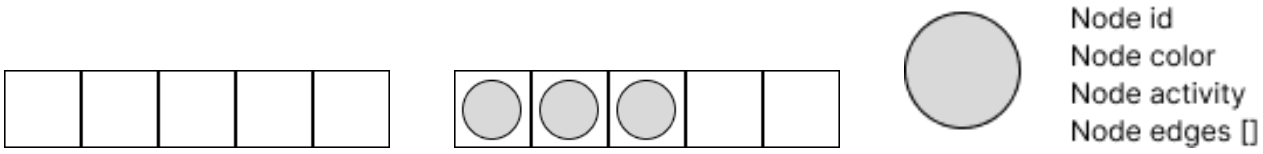
Scanner si nastaví buffer pro zapsání čísla a začíná na stavu 0. Přečte soubor znak po znaku funkcí a dívá se na první písmeno, které je na řádku. Scanner očekává soubor v přesném formátu viz. obrázek automatu scanneru, podle kterého bude první písmeno ‚V' nebo ‚ ' (mezera) a tyto dvě možnosti jsou vyjádřeny přes switch. Pokud řádek začíná znakem ‚V' předpokládá, že je ve stavu 0 a kontroluje, že Vrchol je v přesném formátu. Zapíše čísla do bufferu a po konci řádku vytvoří Vrchol z id, které má zapsané v bufferu a nastaví stav na 1. Stejně pracuje pokud řádek začíná ‚ ' mezerou, zde se jedná o hranu. Kde očekává stav 1 (hned po zápisu vrcholu), nebo stav 2 (po zápisu hrany). Na konci zápisu hrany se dívá, jestli tento řádek nekončí ‚/n' čímž nastaví stav 2 a pokračuje ve čtení hran pokud řádek končí ‚;', který značí konec struktury vrcholu. Nastaví stav na 0 a celý cyklus se opakuje, až do konce souboru.



Obrázek 1: DKA

4 Ukládání dat

Pro Uložení dat jsme si potřebovali vybraly vhodnu datovou struktůru. V našem případě jsme použili nafukovací pole. Nadefinovaly jsme si ji jako strukturu s názvem Vektor. Pro uložení vrcholů jsme si nadefinovali zvlašť strukturu.

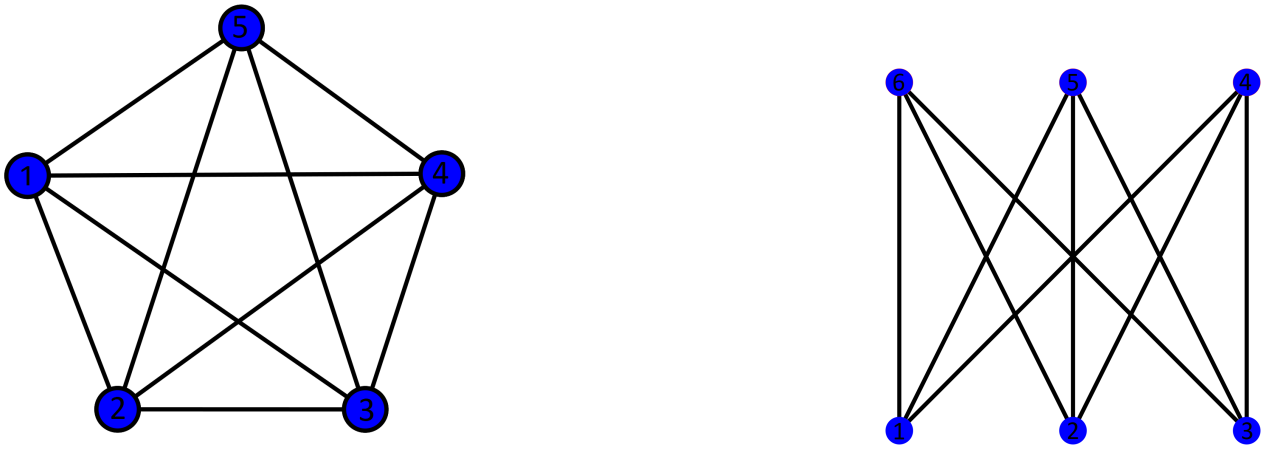


(a) Vektor (b) Graf (c) Uzol

Obrázek 2: Three simple graphs

5 Finální logika

Na Testování rovinnosti existuje několik podmínek. Bohužel většina z nich je len postačující, ne nutné. Jedinné co zůstalo použít je Kuratowského teorém. Jeho řešení spočívá v tom ze sa snažíme v grafu najít podgraf grafu 3a a 3b. Bohužel, nevýhoda tohoto algoritmu je neefektivita. Musime porovnat hodně vrcholů mezi sebou.



(a) Graf K_5

(b) Graf $K_{3,3}$

Obrázek 3: Nelinearne grafy

6 Testování

Na Testování jsme použili porovnání aktuálních vysledků programu s referenčními výsledkami. Rozdělení bylo použité jako v přibaleném Makefile.

- Testování datových struktur
- Testování Lexikálního analyzátoru
- Testování finální logiky algoritmu

Použily jsme skripty v adresaři test pro zjednodušení celého procesu testování.

7 Závěr

Při řešení jsme narazili na spoustu nejasností, které jsme museli řešit na Discordu, nebo některá na fóru projektu.

Celkově byl tento projekt přínosný. Naučili jsme se práci v týmu a organizování času. Velkým přínosem byly zejména zkušenosti s verzovacími systémy *Git*.

8 Časová složitost

Načítaní a ukladání do struktůry má složitost $O(n)$ Finalni algoritmus pracuje bohuzel takzvane brute-force. Teda algoritmus je neefektivni a porovnava kazdy vrcol s kazdim a ma tedy slozitost $O(n!)$