8. Největší klika v neorientovaném grafu

Předmět: BALG/IALG-Algoritmy
Akademický rok vypracování: 2018/2019
Název zadání: 8. Největší klika v neorientovaném grafu
Autoři: Jiří Bekr (xbekrj00), Jiří Šrámek (xrame02), Matěj Kroulík (xkroul02), Tomáš Macko (xmacko11)

Anotace

Tato práce popisuje způsob hledání klik v grafu. Obsahuje popis řešení algoritmu vyhledávání klik, popis problematiky a popis našeho řešení vyhledávání největších klik.

Klíčová slova

Klika, graf, Bron-Kerbosch, algoritmus

Annotation

This work describes how cliques are searched for in a graph. It contains description of algorithm solution for clique search, it tells about clique algorithm problematics, and includes our solution of the biggest clique seeking problem.

Keywords

Clique, graph, Bron-Kerbosch, algorithm

Obsah

1. Zadání

Klika grafu je podgraf, který je úplným grafem. Kterýkoliv vrchol kliky je tedy spojen hranou se všemi ostatními vrcholy kliky.

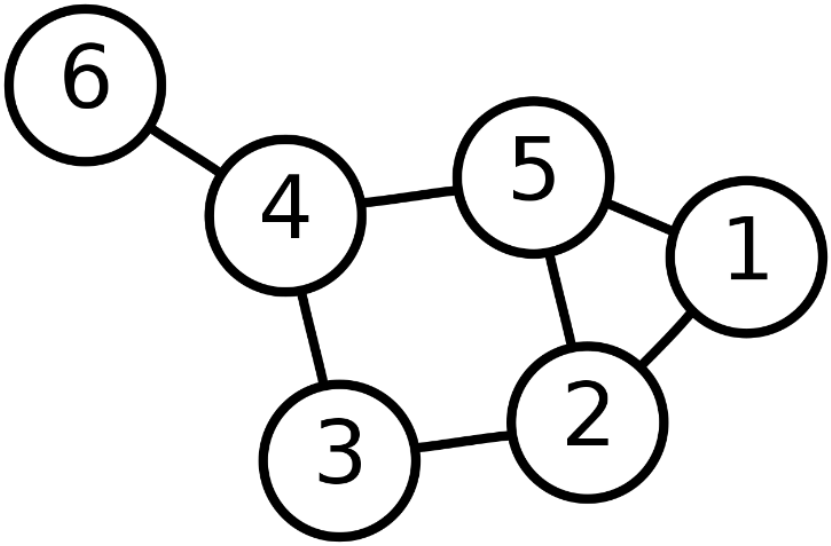
Vytvořte program pro hledání největší kliky v neorientovaném grafu. Pokud existuje více řešení, nalezněte všechna. Výsledky prezentujte vhodným způsobem. Součástí projektu bude načítání grafů ze souboru a vhodné testovací grafy. V dokumentaci uveďte teoretickou složitost úlohy a porovnejte ji s experimentálními výsledky.

2. Teoretická složitost úlohy

2.1. Teorie grafu

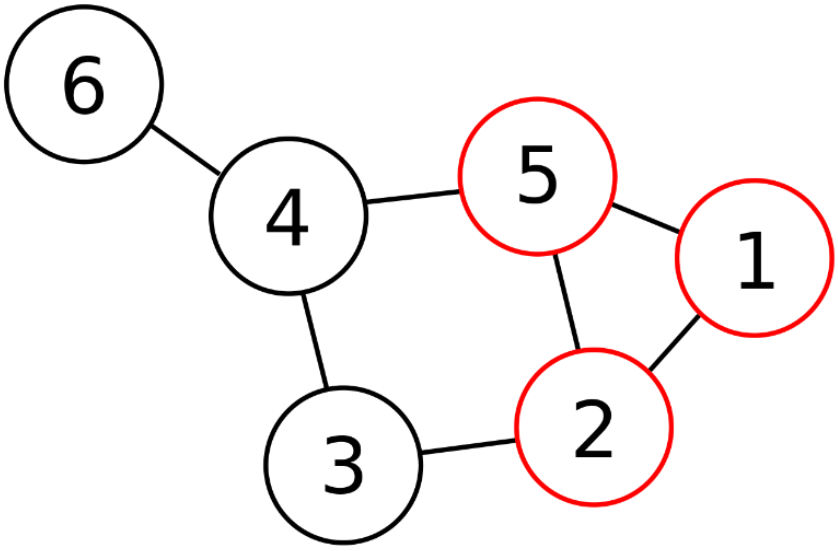
Neorientovaný graf se označuje takový graf, jehož hrany jsou dvouprvkové množiny. Oproti tomu hrany orientovaného grafu jsou uspořádané dvojice. Hrany neorientovaného grafu nemají danou orientaci.

Klika grafu je podgraf, který je úplným grafem. Kterýkoliv vrchol kliky je tedy spojen hranou se všemi ostatními vrcholy kliky. Dvě kliky sousedící přes jeden prvek se nestávají jednou klikou.



Obrázek 1: Neorientovaný graf

Největší klika grafu je seznam všech prvků grafu, které na sebe mají navzájem vazby.



Obrázek 2: Největší klika v neorientovaném grafu

Teoretická časová složitost B-K algoritmu na rozdíl od jiných algoritmů pro hledání kliky není polynomická, ale jeho složitost pro n-uzlový graf lze vyjádřit jako $O(3^{n/3})$.

2.2. Popis algoritmu

Bron-Kerboschův algoritmus, dále jen B-K algoritmus, vyhledává nejvyšší počet klik v neorientovaném grafu.

Algoritmus vymysleli v roce 1973 nizozemští vědci Coenraad Bron a Joep Kebosch, odtud jeho název. Jejich algoritmus byl efektivnější než tehdy používané, díky čemuž nalezl uplatnění hlavně v chemických vědních oborech.

Současně s ním ještě vznikl algoritmus Akkoyunlu, který se sice sémanticky lišil, ale nakonec vedl ke shodnému rekurzivnímu postupu jako B-K algoritmus.

Jev si lze představit jako shluk lidí. B-K algoritmus vyhledá největší skupinu lidí, kteří se navzájem všichni znají, popřípadě kteří mají na sebe navzájem telefonní čísla.

Základní forma B-K algoritmu je rekurzivní funkce se zpětným vyhledáváním, která hledá všechny největší kliky v grafu. Algoritmus má tři parametry – R, P a X. R je výstup algoritmu. P je vstupní graf a X je seznam vyloučených uzlů.

B-K algoritmus je v obecném algoritmovacím jazyce realizován následovně:

```
BronKerbosch(R, P, X):
    if P and X are both empty:
        report R as a maximal clique
        return
    for each vertex v in P:
        BronKerbosch(R ⋃ {v}, P ⋂ N(v), X ⋂ N(v))
        P := P \ {v}
        X := X ⋃ {v}
```

Počáteční volání se provádí s prázdným grafem (seznamem uzlů) R a X. Parametr P obsahuje graf, ve kterém se mají hledat kliky. Při každém dalším rekurzivním volání projde algoritmus všechny uzly grafu {v} v P. Pokud je P prázdné a zároveň je prázdné i X, tak nahlásí kliku. V případě že je prázdné pouze P vrátíme se v algoritmu. Jinak proběhne rekurzivní volání, kdy je aktuální prvek {v} přidán k R, P a X jsou omezené na sousední uzly {v}. Tím se zajistí nalezení všech rozšířených klik R, které obsahují prvek {v}. Následně se přesune prvek {v} z P do X, čímž se vyloučí z dalších hledání. Algoritmus pokračuje dalším uzlem {v} z P.

3. Experimentální výsledky

Výsledný program se spouští se jedním parametrem, kterým je cesta k souboru grafu. Návratové hodnoty signalizují chyby za běhu programu:

Návratová hodnota	Význam
0	Program proběhl v pořádku
1	Detekována chyba práce s dyn. pamětí (pouze pro ladění), ve stabilní verzi nenastane, knihovna check.c od Ing. Petyovského PhD
2	Chyba načtení vstupního souboru (nesprávný formát)
3	Nedostatečné množství paměti pro dynamická data
4	Nesprávné argumenty programu
5	Vstupní soubor se nezdařilo otevřít

Tabulka 1: Návratové hodnoty programu

Algoritmus jsme testovali na grafech různé složitosti a o různém počtu klik. Tyto grafy jsou uloženy ve složce Test-Graph, která je součástí odevzdávaného archivu. Soubor grafu je v textové podobě (enkódování UTF8 (bez BOM), Windows 1250, případně příbuzné, kde se neliší reprezentace čísel, čárky a dvojtečky).

Pro vytvoření vlastního grafu je vytvořena šablona ve výše zmíněné složce. Syntaxe souboru je velmi jednoduchá, vždy na začátku řádku uvedeme číslo uzlu, následuje dvojtečka a poté seznam čísel uzlů oddělených čárkou, které připojené uzly. Viz níže:

```
1   0:4,1
2   1:4,0,2
3   2:1,3
4   3:5,4,2
5   4:0,1,3
6   5:3
```

Obrázek 3: Formát vstupního souboru

Program v hlavní funkci main nejprve načte soubor s grafem a vypíše jeho data. Následuje volání funkce FindCliques, odkud se volá samotná funkce BronKerbosch, která vyhledá kliky v grafu. Pro práci s grafy jsme vytvořili funkce pro vkládání a odstranění prvku z grafu, vytvoření kopie grafu a zrušení grafu.

Náročnost algoritmu, co se týče počtu iterací, využití dynamické paměti a nalezení klik je v Tabulka 2: Náročnosti grafu:

Soubor grafu	Počet uzlů grafu	Počet iterací	Využití dynamické paměti		Nalezená klika
			V součtu	Špičkové	
Graf1.txt	6	15	4776 B	1084 B	Ano – 3 uzly
Graf2.txt	5	13	4072 B	940 B	Ano – 3 uzly
Graf3.txt	7	27	10192 B	1824 B	Ano – 4 uzly
Graf4.txt	13	46	21028 B	290 B	Ano – 3 uzly
Graf5.txt	26	104	54036 B	4848 B	Ano – 5 uzlů
Graf6.txt	6	16	4872 B	1152 B	Ano – 2x3 uzly
Graf7.txt	7	27	7708 B	1444 B	Ano – 4 uzly

Tabulka 2: Náročnosti grafu

4. Závěr

Povedlo se nám implementovat plně funkční rekurzivní algoritmus pro hledání klik v grafu. Tento algoritmus je schopen najít všechny kliky v grafu, pro zobrazení pouze těch největších bylo třeba výsledky filtrovat. Napsaný program je poměrně náročný na dynamickou paměť, z toho vyplývá i náročnost na chyby při alokaci/dealokaci, pro jejich kontrolu byl využit checker [2], což je daň za možnost prakticky nekonečného množství uzlů a propojů (omezeno velikostí systémovou RAM). Pro všechny testovací grafy bylo dosaženo očekávaného výsledku, všechny kliky v grafu byly správně nalezeny.

5. Seznam zkratek a pojmů

BOM – Byte Order Mask
B-K – Bron-Kerbosch

6. Seznam obrázků

7. Seznam tabulek

8. Použité zdroje a odkazy

[1] Bron–Kerbosch algorithm. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-12-03]. Dostupné z: https://en.wikipedia.org/wiki/Bron%E2%80%93Kerbosch_algorithm?fbclid=IwAR3qFBIr2hMzsY3idiGRCBAzOYgB9_ZY-gzPcl4aZHjf75ZkpLPkwQYS13Y

[2] SABATKA, Pavel a Petr PETYOVSKY. Checker. In: *Ústav automatizace a měřicí techniky* [online]. Brno: VUT Fekt Brno [cit. 2018-12-04]. Dostupné z: http://www.uamt.feec.vutbr.cz/~petyovsky/projects/checker/checker.zip