Vysoké učení technické v Brně
Fakulta informačních technologií

Minimálna kostra grafu

Dokumentácia náhradného projektu IAL – zadanie č. 7

Autori:
xpospi95 - Michal Pospíšil (vedúci)
xtimko01 - Nikola Timková
xbalaz08 - František Balázsy

Brno, 3.12.2018

Obsah

Úvod

Táto dokumentácia popisuje riešenie náhradného projektu – zadania č. 7 – minimálna kostra grafu. Aplikáciu, ktorá je hlavným bodom tohto zadania sme sa snažili vytvoriť tak, aby bola jednoducho použiteľná.
V ďalších častiach dokumentu budeme popisovať aké stratégie sme zvolili pri riešení projektu, a prečo sme sa pre ne rozhodli. Popíšeme aj na aké problémy sme pri riešení projektu narazili.

Zadanie

- Kostra grafu je podgraf, ktorý je stromom, a ktorý obsahuje všetky vrcholy pôvodného grafu. Z tejto definície vyplývajú nasledujúce tvrdenia:
    - Kostra grafu je súvislý graf.
    - Kostra grafu má o jednu hranu menej než má vrcholov.
    - Nesúvislý graf nemá kostru.
    - Kostra grafu neobsahuje kružnicu.
- Vytvorte program pre nájdenie kostry grafu s minimálnym ohodnotením pre ohodnotené neorientované grafy.
- Ak existuje viac riešení, nájdite všetky. Výsledky prezentujte vhodným spôsobom. Súčasťou projektu bude načítanie grafov zo súboru a vhodné testovacie grafy. V dokumentácií uveďte teoretickú zložitosť úlohy a porovnajte ju s experimentálnymi výsledkami.

Výber vhodného algoritmu

Pre riešenie zadanej problematiky existuje rada algoritmov, preto sme museli vybrať najvhodnejší pre pokrytie všetkých požiadaviek vyplývajúcich zo zadania s prihliadnutím na ich technické parametre a jednoduchosť implementácie. Pre prehľad uvádzame krátky prehľad najznámejších algoritmov a zdôvodnením, prečo sme ich použili alebo nepoužili:

Borůvkov algoritmus

Pažravý algoritmus vynájdený pri riešení problematiky konštrukcie elektrickej siete na Morave. Značne obmedzujúci, pretože predpokladá grafy s hranami kladných a rôznych hodnôt. Môže však byť paralelizovaný, čo mu dáva výhodu oproti ostatným v prípade dostatku výpočtových jednotiek. Tvorí základ náhodnostného algoritmu bežiaceho v lineárnom čase, ktorý však funguje len pre grafy s neizolovanými vrcholmi.
Tento algoritmus sme si nevybrali z dôvodu absencie možnosti použitia hrán s rovnakou váhou a takisto pre jeho predpis len kladných hrán.

Primov algoritmus

Pažravý algoritmus pre neorientované grafy vynájdený Vojtechom Jarníkom. Svoje najväčšie využitie nachádza v prípadoch veľmi hustých grafov s veľa vrcholmi, kde beží s O(E + V log V). Táto časová zložitosť však platí len v prípade, že spracováva vstupné dáta v podobe Fibonacciho haldy a zoznamu susednosti. Pokiaľ by sme algoritmus spustili nad maticou susednosti v najhoršom prípade by dosiahol časovú zložitosť $O(|V|^2)$.
Tento algoritmus sme sa rozhodli nevybrať, pretože síce spĺňa naše požiadavky na použitie hrán s akoukoľvek váhou, avšak má vyššiu časovú zložitosť pre typické grafy, ktoré predpokladáme. Taktiež nie je efektívny v prípade spracovania dát v podobe matice susednosti, ako konvenčnej metóde reprezentácie grafu.

Kruskalov algoritmus

Pažravý algoritmus vhodný na typické, redšie grafy. Vyznačuje sa veľkou implementačnou výhodou oproti Primovmu algoritmu pretože dokáže spracovať priamo maticu susednosti. Taktiež má nižšiu časovú zložitosť nad bežnými grafmi, a to O(E * log V).
Pre riešenie nášho zadania sme si vybrali Kruskalov algoritmus, pretože vyhovuje povahe našich vstupných dát, vyznačuje sa najjednoduchšou implementáciou a dosahuje najlepšiu časovú zložitosť nad bežnými grafmi. Taktiež je možné pomocou neho jednoducho nájsť viacero riešení, ako je to opísané v zdroji číslo 3.

Detaily implementácie

Reprezentácia grafu

Graf sme sa rozhodli reprezentovať maticou susednosti. Naše rozhodnutie ovplyvnilo, že jej veľkosť je ľahko vypočítateľná z počtu uzlov a na rozdiel od ukladanie samotných hrán sa nevyskytuje redundancia názvov uzlov ak vychádza viac hrán z jedného vrcholu. Navyše tento modul môže byť použitý s minimálnymi zmenami na čítanie súborov, ktoré by reprezentovali orientované grafy v inom programe.

Špecifikácia súborového formátu

Kódovanie súboru je kompatibilné s Unicode (napríklad UTF-8, ASCII). Program používa interne dátový typ *wchar_t*.
Formát súboru je veľmi jednoduchý. Pri jeho návrhu sme vychádzali z formátu CSV, ktorý je veľmi jednoduchý na spracovanie a kontrolu.
Riadok 1: Ako prvý reťazec musí súbor obsahovať reťazec ".xIALAdjMatrix". Nasledovať musí znak ukončenia riadka (EOL znak) v akomkoľvek štýle (LF/CRLF...). Tento reťazec umožňuje kontrolovať, že vstupom je naozaj súbor podporovaného typu.
Riadok 2: Počet uzlov zadaný číselne (číslicami 0-9), bez akýchkoľvek znamienok, ukončený EOL znakom. Maximálny počet uzlov je 255. Slúži na zjednodušenie implementácie syntaktickej kontroly matice susednosti - je ľahšie čítať zo vstupu neznáme dáta a overiť ich správnosť, keď vieme, koľko ich má byť.
Riadok 3: Názvy uzlov v takom poradí, v akom sú zapísané v matici vodorovne zľava doprava a zvislo zvrchu nadol. Názvy uzlov sú reťazce obsahujúce ľubovoľné znaky. Reťazce sú oddelené čiarkou. V prípade, že názov uzlu obsahuje čiarku, musí byť escapovaná znakom '\\' => (takto\,). To znamená, že aj samotný znak '\' musí byť escapovaný (takto: \\niečoZaLomítkom). Riadok je ukončený EOL znakom.
Ďalšie riadky (max. 255 - max. počet uzlov): Obsahujú maticou uložený graf - prvkami matice sú váhy hrán (min. $-(2^{64}/2)$ max. $2^{64}/2-1$) spájajúca vrchol reprezentovaný indexom v matici, ktorý je naviazaný na názvy uzlov. Ak uzly nie sú spojené, váha hrany je 0. Všetky váhy hrán sú oddelené čiarkami, nie sú tolerované žiadne biele znaky. Každý (aj posledný) riadok končí EOL znakom a súbor musí byť ukončený hneď za posledným EOL znakom..

Limity

Graf

Veľkosť grafu - max. 2^8-1 uzlov a $2^{16}-1$ hrán. Maximálna váha hrany je $2^{32}-1$. Tieto limity sú nezávislé na implementácii, boli použité dátové typy s presne vymedzenou veľkosťou.
Pri načítaní súboru je program ošetrený proti pretečeniu, ktoré hrozí pri väčších grafoch a proti syntaktickým a lexikálnym chybám v súbore.

Vstupný súbor

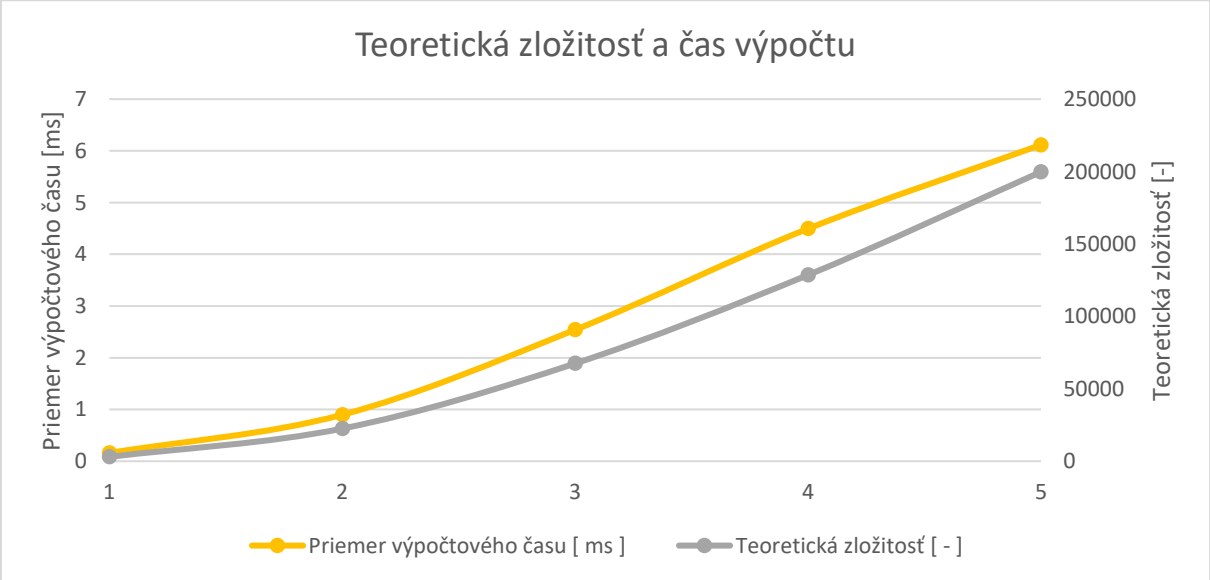
Súbor môže mať akúkoľvek dĺžku. Dĺžka jedného názvu by nemala byť dlhšia ako $2^{16}/2$ (neošetrené). Pri načítaní súboru je program ošetrený proti pretečeniu, ktoré hrozí pri grafoch nespĺňajúcich špecifikáciu a proti syntaktickým a lexikálnym chybám v súbore.

Porovnanie teoretickej zložitosti úlohy s experimentálnymi výsledkami

Teoretická zložitosť Kruskalovho algoritmu je O(E * log V), avšak naša implementácia algoritmu má časovú zložitosť $O(N * E * \log E + N^2)$, ako je možné vidieť v zdroji číslo 1. Na testovanie sme si pomocou VBA skriptu vygenerovali 5 matíc, ktoré sú odstupňované podľa zložitosti.

Matica	Počet uzlov - N	Počet hrán - E	Teoretická zložitosť [-]	Priemer výpočtového času [ms]
1	20	70	2983.137	0.159
2	50	178	22528.738	0.898
3	80	307	67484.119	2.541
4	110	406	128596.773	4.496
5	140	480	199779.411	6.116

Tabuľka č. 1 – Parametre testovacích matíc a výsledky testu



Graf č. 1 – Porovnanie teoretickej zložitosti s reálnym časom výpočtu

Na grafe číslo 1 môžeme vidieť trendy vypočítaných zložitostí a výpočtového času. Tvary oboch charakteristík sú značne podobné, preto môžeme usúdiť, že výpočtový čas koreluje s teoretickou zložitosťou nášho algoritmu.

Testovanie programu

Projekt sme testovali ručne za pomoci špeciálne vytvorených testovacích súborov, ktoré testovali limity alebo obsahovali syntaktické chyby v súboroch, aby sme overili funkčnosť programu za každých okolností.

V budúcnosti sa dá tento prístup vylepšiť napríklad vytvorením automatického bash skriptu, ktorý otestuje súbory a bude kontrolovať úspešnosť programu automaticky. Pre rozsah tohto projektu sme to však nepovažovali za potrebné.

Na syntakticky správnych vstupoch sme testovali, či program naozaj vypíše všetky nájdené minimálne kostry. Opäť sme vytvorili súbory tak, aby sme pokryli čo najviac možností, ktoré môžu nastať pri použití programu reálnym používateľom.

Zaujímavé pasáže

Aby sme mohli plne otestovať funkcie programu, rozmýšľali sme ako vytvárať matice susednosti, ktoré reprezentujú grafy. Keďže sme náš súborový formát založili na formáte CSV, obrátili sme sa na pomoc programu Microsoft Excel a jeho podporu makier v jazyku Visual Basic for Applications. V priečinku *examples* je teda možné nájsť hárok s makrom generujúcim náhodné grafy podľa počtu uzlov, hrán a intervalu, z ktorého sa majú vygenerovať váhy. Vďaka tomu sme boli schopní rýchlo vytvoriť mnoho testovacích príkladov, ktoré nám pomohli objaviť slabiny v implementácii, na ktoré by sme inak neprišli.

Zaujímavosťou je, že chybové ukončenie programu na Ubuntu 18.04 64-bit po vypísaní chybového hlásenia je zakončené signálom Zrušené (SIGABRT) a Chyba segmentácie (SIGSEGV). Pri vyšetrovaní tejto chyby sme prišli na to, že signál sa odosiela až po ukončení funkcie main - v debuggeri sme videli len disassembly kódu, ktorý vyzeral ako nejaká čistiaca rutina. Toto správanie bolo potvrdené len na systéme Ubuntu 18.04 64-bit.
Na referenčnom stroji eva, merlin, a na PC s macOS sa nám takáto chyba nepodarila zreprodukovať. Tiež sme skúmali podozrenie na prepisovanie nejakej pamäte a nástroj valgrind aj zobrazil nejaké nedostatky, ale opäť len po výpise chybovej hlášky - teda po ukončení funkcie main. Plné vyšetrenie tejto chyby by bolo časovo náročné, a keďže program funguje na referenčnom stroji a väčšine iných zariadení, ďalej sme tento problém nevyšetrovali.

Použitie programu

Preklad programu

Preklad prebieha pomocou nástroja make. Pre zostavenie finálneho programu stačí v hlavnom adresári projektu spustiť príkaz make bez parametrov, výsledkom je binárny súbor **sptree** (skratka zo spanning tree).

Spustenie programu

./sptree <filename>
./sptree -h
Program očakáva len jeden argument a tým je buď cesta k súboru <filename>, alebo prepínač -h, ktorý vypíše krátku nápovedu.

Zdroje

[1]Webstránka
Autori : Takeo Yamada, Seiji Kataoka and Kohtaro Watanabe
Názov: " Listing all the minimum spanning trees in an undirected graph"
Zdroj: International Journal of Computer Mathematics
Vydavateľ: Taylor & Francis
Dátum: 2010-11
URL: http://web.iitd.ac.in/~bspanda/mstgenpapers.pdf?fbclid=IwAR2r6Vc_937PF-urx-hcqUkzH-R6Khl4qSru_JXK5sw1plxts2pQGZ3x5SE

[2]Webstránka
Autori : Fórum
Názov: " Execution time of C program"
Zdroj: Stack overflow
Dátum: 2018-11-21
URL: https://stackoverflow.com/questions/5248915/execution-time-of-c-program?fbclid=IwAR0twzqPzRAW9wGpZi9HNcEOC1V1sbM215hh5g9308KPdE0db33YcQ9FbNQ

[3]Webstránka
Autori : João Guilherme Martinez, Rosiane de Freitas, Altigran Silva
Názov: " On the problem of finding all minimum trees "
Zdroj: VII Latin American Workshop on Cliques in Graphs
Vydavateľ: Universidade Federal do Amazonas
Dátum: 2016-11-8
URL:http://www.mate.unlp.edu.ar/~liliana/lawclique_2016/07.pdf?fbclid=IwAR3bzwzlMmfxfD_GMb-iBZ9bzpJJ1o6buifzgsqjrbOr-9CMOV09OZLfyXw

[4]Webstránka
Názov: " Minimum spanning tree"
Vydavateľ: Wikipedia
Dátum: 2018-12-3
URL: https://en.wikipedia.org/wiki/Minimum_spanning_tree