Vysoké učení technické v Brně

Fakultka informačných technológií

# Dokumentácia k náhradnému projektu

### 

### Zadanie :

* *Kostra grafu* je podgraf, který je stromem a který obsahuje všechny vrcholy původního grafu. Z této definice vyplývají následující tvrzení. Kostra grafu je souvislý graf. Kostra grafu má o jednu hranu méně než má vrcholů. Nesouvislý graf nemá kostru. Kostra grafu neobsahuje kružnici.
* Vytvořte program pro hledání kostry grafu s minimálním ohodnocením pro ohodnocené neorientované grafy.
* 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.

## Obsah :

* Výber vhodného algoritmu
* Návod k programu
* Porovnanie teoretickej zložitosti úlohy s experimentom

### Autori :

1. xpospi95 - Michal Pospíšil
2. xtimko01 - Nikola Timková
3. xbalaz08 - František Balázsy

## Výber vhodného algoritmu :

Pre riešenie zadanej problematiky existuje rada algoritmov, našou úlohou bolo vybrať najvhodnejší pre pokrytie všetkých požiadaviek vyplývajúcich zo zadania s prihliadnutím na jeho technické parametre a jednoduchosť implementácie. Pre prehľad uvádzame krátku rešerš najznámejších algoritmov :

### 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 paralelizovateľný, čo mu dáva výhodu oproti ostatným v prípade dostatku výpočtových jednotiek. Tvorí základ probabilistického algoritmu bežiaceho v lineárnom čase, ktorý však funguje len pre grafy s neizolovanými vrchoľmi.

### 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).

### Kruskalov algoritmus

* Pažravý algoritmus vhodný na typické, redšie grafy. Vyznačuje sa veľkou implementačnou výhodou oproti Primsovmu 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 implementovateľnosťou a dosahuje najlepšiu časovú zložitosť nad bežnými grafmi. Taktiež je možné pomocou neho jednoducho nájsť viacero riešení.

## Návod k programu :

Preklad prebieha pomocou utility make. Pre zostavenie finálneho programu stačí v hlavnom adresári projektu spustiť príkaz make bez parametrov, výsledkom je binárny súbor sptree(skratka zo spanning tree). Program očakáva len jeden argument a tým je buď cesta k súboru, alebo prepínač -h, ktorý vypíše krátku nápovedu.

### *Spustenie programu :*

./sptree \<filename>

./sptree -h

### *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 UTF-8 alebo kódovania s ním kompatibilné (ASCII).

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. -(264/2) max. 264/2-1) 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..

### *Implementačné limity :*

Veľkosť grafu - max. 28-1 uzlov a 216-1 hrán. Maximálna váha hrany je 264/2-1, povolené sú aj záporné váhy (min. -(264/2)).

Pri načítaní súboru je program ošetrený proti pretečeniu, ktoré hrozí pri väčších grafoch a proti syntaktickým a lexikálnym chybám v súbore.

**Teoretická zložitosť Kruskalovho algoritmu :**