Semestrální práce z KIV/PC

Vyhledávání cest v grafu technikou DFS

Petr Hlaváč
A15B0037P
petrh@students.zcu.cz

1. 12. 2017

Obsah

1 Zadání

Naprogramujte v ANSI C přenositelnou **konzolovou aplikaci**, která bude **procházet graf technikou DFS** (Depth–First–Search). Vstupem aplikace bude soubor s popisem grafu. Výstupem je pak odpovídající výčet všech cest mezi požadovanými uzly.

Program se bude spouštět příkazem **dfs.exe** <soubor-grafu> <id1> <id2> <maxD>

- Symbol <soubor-grafu> zastupuje parametr - název vstupního souboru se strukturou grafu.
- Následují identifikátory (dále jen id) dvou uzlů v grafu <id1> a <id2>, mezi kterými bude spouštěn proces hledání cest.
- <maxD> je parametr popisující maximální délku cest, které mají být nalezeny.

Vámi vyvinutý program tedy bude vykonávat následující činnosti.

1. Při spuštění bez potřebných parametrů vypíše nápovědu pro jeho správné spuštění a ukončí se.
2. Při spouštění s parametry načtě zadaný vstupní soubor do vhodné struktury reprezentující graf a mezi zadanými uzly najde všechny cesty, jejichž délka nepřekročí konstatnu nastavenou vstupním parametrem.

Váš program může být během testování spuštěn například takto:

```
dfs.exe graph.csv 1 2 4
```

Při tomto spuštění program vyhledá všechny cesty délky 1 až 4 mezi uzly 1 a 2.

Hotovou práci odevzdejte v jediném archivu typu ZIP prostřednictvím automatického odevzdávacího a validačního systému. Postupujte podle instrukcí uvedených na webu předmětu. Archiv nechť obsahuje všechny zdrojové soubory potřebné k přeložení programu, **makefile** pro Windows i Linux a dokumentaci ve formátu PDF vytvořenou v typografickém systému TEX, resp. LATEX.

1.1 Popis činnosti programu

Program načte vstup ve formátu CSV, který vhodným zbůsobem uloží do paměti pro pozdější prohledávání. Následně spustí algoritmus, jehož výstupem bude seznam všech cest mezi dvěma uzly, které bude vypisovat na standardní výstup ve formátu popsaném dále.

1.1.1 Specifikace výstupu programu

Program bude na standardní výstup vypisovat jednotlivé cesty. Vždy na jeden řádek právě jednu cestu. Cesty budou popsány posloupností id jenotlivých uzlů oddělených pomlčkou (tj. znakem minus, „**-**“). následovat bude středník a popisky jednotlicých hran v grafu oddělené čárkou. Za posledním středníkem bude uvedena hodnota sekundární metriky - relevance cesty, podle které budou cesty seřazeny (viz Řazení výstupu).

Například tedy pro hledání cest mezi uzly A a B:

A-B;h_1;m_1
A-F-B;h_2,h_3;$m_{2,3}$
A-W-B;h_4,h_5;$m_{4,5}$
A-F-O-B;h_2,h_6,h_7;$m_{2,6,7}$

Kde A, B, F, W, F, ... jsou popisky uzlů grafu, h_n jsou ohodnocení hran a $m_{c1,c2...cn}$ je hodnota metrika relevance nalezené cesty.

1.1.2 Řazení výstupu

Cesty budou na standardním výstupu seřazeny podle jejich délky. Pokud bude nalezeno více cest stejné dálky, budou seřezeny podle jejich relevance následujícím způsobem. Popisky grafu nesou informaci o kalendářním datu ve formátu YYYY–MM–DD. Každá cesta budde tedy ohodnocena celým číslem, které bude odpovídat rozdílu v počtu dní mezi nejstarším a nejnovějším datem, následně budou podle toho čísla cesty se shodnou délkou seřazeny vzestupně.

Pokud tedy nalezete cestu, kterou budou tvořit tyto 4 hrany s hodnotami:

2000–01–15,
2000–01–04,
2000–01–19,

2000–01–24,

vypočte se hodnota metriky jako počet dnů mezi daty 2000–01–04 a 2000–01–24, což je 20.

1.2 Podrobné zadání

Originální nezkrácené zadání je možné nalézt na adrese:

http://www.kiv.zcu.cz/studies/predmety/pc/doc/work/sw2017-02.pdf

2 Analýza úlohy

Úloha se skládá z následujících kroků:

1. Načtení vstupních dat.
2. Vybrání vhodné datové struktury.
3. Modifikace algoritmu DFS pro hledání všech cest mezi dvěma body.

2.1 Načtení vstupních dat

CSV nebo-li Comma-separeted-values je jednoduchý souborový formát určený pro výměnu tabulkových dat.

Ukázka vstupního souboru ve formátu CSV

```
1;2;2007–02–16
1;2;2008–02–15
1;3;2007–10–26
1;4;2007–10–30
3;2;2007–10–11
3;6;2007–06–14
3;7;2007–08–14
3;9;2008–03–05
4;2;2007–10–05
5;1;2007–02–04
```

Data jsou formátovány ve stylu idUzlu1; idUzlu2; hodnotaHrany

Jedna řádka vstupního souboru představuje jednu hranu. Bude tedy vhodné načítat a zpracovávat data po řádcích. Formát CSV nám práci s daty velmi ulehčí, neboť nebude příliš náročné data v řádce rozdělit za pomoci znaku „**;**“.

2.2 Vybrání vhodné datové struktury

Pro reprezentaci grafu se využívají následující struktury:

1. Matice sousednosti
2. Spojový seznam

2.2.1 Matice sousednoti

Matice sousednosti má velikost $V \times V$ kde V je počet uzlů. Vzhledem k faktu, že máme ohodnocené hrany grafu vložíme na pozice v matici, kde se nachází hrana její hodnotu a na pozice kde hrana není nějakou specialní hodnotu v našem případě (NULL). Hrany v grafu jsou zadány jako neorientované, tudíž matice bude symetrická viz vzorová matice stavené pro výše zmíněná vstupní data.

0	1	2	3	4	5	6	7	8	9
1	$NULL$	$h1, h2$	$h3$	$h4$	$h10$	$NULL$	$NULL$	$NULL$	$NULL$
2	$h1, h2$	$NULL$	$h5$	$h9$	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$
3	$h3$	$h5$	$NULL$	$NULL$	$NULL$	$h6$	$h7$	$NULL$	$h8$
4	$h4$	$h9$	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$
5	$h10$	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$
6	$NULL$	$NULL$	$h6$	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$
7	$NULL$	$NULL$	$h7$	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$
8	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$
9	$NULL$	$NULL$	$h8$	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$

Neboť je matice pro orientované hrany simetrické, lze ji z paměťových důvodů polovinu zanedbat. Uložená matice by tedy vypadala takto.

0	1	2	3	4	5	6	7	8	9
1	$NULL$								
2	$h1, h2$	$NULL$							
3	$h3$	$h5$	$NULL$						
4	$h4$	$h9$	$NULL$	$NULL$					
5	$h10$	$NULL$	$NULL$	$NULL$	$NULL$				
6	$NULL$	$NULL$	$h6$	$NULL$	$NULL$	$NULL$			
7	$NULL$	$NULL$	$h7$	$NULL$	$NULL$	$NULL$	$NULL$		
8	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$	
9	$NULL$	$NULL$	$h8$	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$	$NULL$

Pokud se na tuto matici podíváme pozorně zjistíme, že jde v podstatě o reprezentaci Spojovým seznamem. Jedinou výhodou je přístup na dané hrany ve složitosti $O(1)$, neboť matice je reprezentována polem. Tato výhoda má ale ovšem i negativa. Spočívají v tom, že je třeba uložit pole, které je stejně veliké jako id největšího vrcholu v našem grafu. V případě, který nastal i

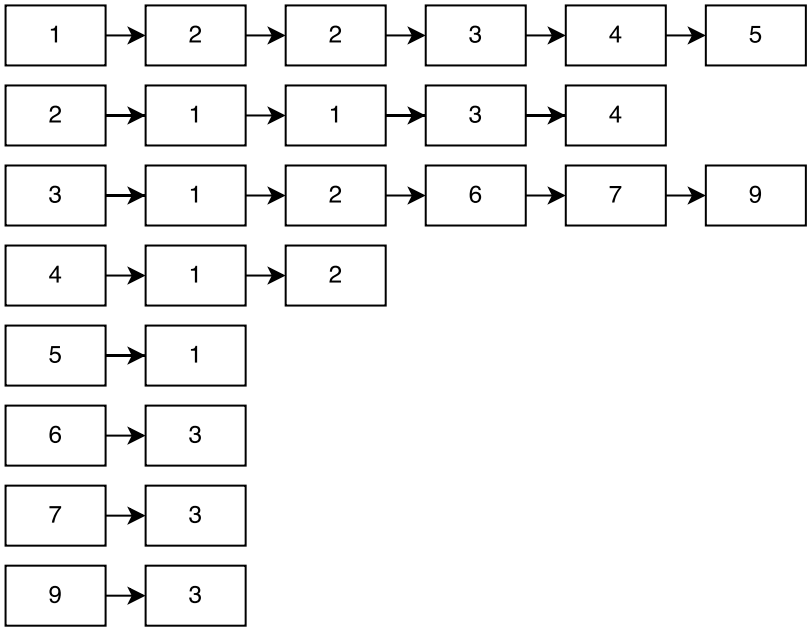
zde, je tedy možné že budeme mít v maticí řádky, které jsou uloženy zcela zbytečně například řádek číslo 8.

Další nevýhodou, které si můžeme všimnout je násobnost hran. Abychom v matici uchovali více hran mezi dvěma vrcholy. Musí být hrany v matici reprezentovány spojovým seznamem nebo polem. V případ volby pole by každé přidání nové hrany znamenalo zvětší pole o jedna a přepsání všech hodnot do nového pole, což je v při načítání velkého grafu velmi neefektivní. Vhodnější je tedy spojový seznam.

Výsledkem této reprezentace je tedy matice spojových seznamů. Velkou výhodou je přístup k hranám vrcholů v $O(1)$. Nevýhodou může být v některých případech grafů zbytečně velká paměťová složitost a také složitost implementace.

2.2.2 Spojový seznam

Struktura spojového seznamu znamené, že máme pro každý vrchol vytvořen spojový seznam hran, které z něho vycházejí a v kterých je uloženo id souseda a ohodnocení hrany. Spojový seznam pro výše zmíněna data by vypadal takto.



Obrázek 2.1: Reprezentace grafu spojovým seznamem

Neorientované hrany jsou vyřešené přidáním stejné hrany k obou vrcholům. Násobné hrany v této implementaci nedělají žádný problém.

Vrcholy grafu je možné mít uložené v poli nebo spojovém seznamu. Výhoda pole je přístup k hraném v $O(1)$. Nevýhodou je pamětová náročnost v případě nevyužití všech indexů pole. Spojový seznam má přístup k vrcholům v $O(n)$, což může program značně zpomalovat v případě velkého počtu vrcholů v grafu. Vhodná volba tedy záleží hlavně na znalosti vstupních dat.

2.2.3 Modifikace algoritmu DFS

Vyjdeme ze základního algoritmu DFS. Budeme potřebovat zásobník a jeden seznam. Ze zadání víme, že naše prohledávání bude omezeno hloubkou. Dále bylo také řečeno, že se jedná o cestu

3 Implementace

4 Uživatelská příručka

5 Závěr