

Offline Review Form for Submission S-ID #129

Titulek vedeckeho prispevku

Review by Vojtěch Danišík

General instructions for the assessment: The better assessment, the higher mark, i.e. 0 = the poorest mark, 10 = the best mark (the only exception is the 'Amount of rewriting' field where 0 means 'no rewriting necessary' and 10 means 'the paper must be entirely rewritten'). Please, use your common sense or ask the organizers if unsure. **This form should be filled in using Adobe Acrobat – please, do not use the built-in PDF viewer in your browser.**

Originality - Rate how original the work is:

0	1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	---	----

Significance - Rate how significant the work is:

0	1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	---	----

Relevance - Rate how relevant the work is:

0	1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	---	----

Presentation - Rate the presentation of the work:

0	1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	---	----

Technical quality - Rate the technical quality of the work:

0	1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	---	----

Overall rating - Rate the work as a whole:

0	1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	---	----

Amount of rewriting - Express how much of the work should be rewritten:

0	1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	---	----

Reviewer's expertise - Rate how confident you are about the above rating:

0	1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	---	----

Main contributions - Summarise main contributions:



Positive aspects - Recapitulate the positive aspects:

Negative aspects - Recapitulate the negative aspects:

Comment (optional) - A message for the **author(s)**:

Internal comment (optional) - An internal message for the **organizers**:

After filling the form in, please, upload it to the TSD2019 web review application: Go to URL <https://www.kiv.zcu.cz/tsd2019> and after logging in, please, proceed to section 'My Reviews', select the corresponding submission and press the 'Review' button. There, you'll be able to upload this PDF file.



ZÁPADOČESKÁ UNIVERZITA V PLZNI

PROGRAMOVÁNÍ V JAZYCE C

KIV/PC

Dokumentace semestrální práce

Vojtěch DANIŠÍK
A16B0019P
danisik@students.zcu.cz

16. dubna 2018



Obsah

1	Zadání	2
2	Analýza úlohy	3
2.1	Reprezentace grafu	3
3	Popis implementace	6
3.1	Struktura pro vrcholy	6
3.2	Struktura pro hrany	6
3.3	Struktura pro graf	6
3.4	Struktura pro nalezené cesty	7
4	Uživatelská příručka	8
4.1	Překlad aplikace	8
4.2	Spuštění a ovládání aplikace	8
5	Závěr	10
6	Literatura	10

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 grafu.

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 spuštění s parametry načte zadaný vstupní soubor do vhodné struktury reprezentující graf a mezi zadanými uzly najde všechny cesty, jejichž délka nepřekročí konstantu nastavenou posledním parametrem.

Více informací naleznete zde.

2 Analýza úlohy

Ze zadání vyplývá, že hlavním problémem této úlohy je uložení grafu do vhodných struktur. Na zvolené struktuře závisí nejen časová náročnost při hledání všech cest v grafu mezi dvěma body, ale i paměťová náročnost struktur. Musí se použít co nejvhodnější datové typy pro uložení jednotlivých hodnot, aby nám aplikace nezabírala zbytečně moc paměti.

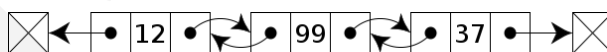
2.1 Reprezentace grafu

Graf lze reprezentovat několika způsoby:

První způsob reprezentace grafu je pomocí **spojového seznamu**. Spojový seznam (také lineární spojový seznam) je dynamická datová struktura, obsahující jednu nebo více datových položek stejného typu, které jsou navzájem lineárně provázány vzájemnými odkazy pomocí ukazatelů. Existují dva typy seznamů - jednosměrný a obousměrný. U každé položky v jednosměrném seznamu existuje pouze odkaz na následující položku, zatímco u obousměrného seznamu existuje navíc i odkaz na předcházející položku¹.



Obrázek 1: Jednosměrný spojový seznam



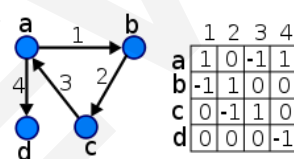
Obrázek 2: Obousměrný spojový seznam

Nástin implementace: V grafu bude uložen spojový seznam všech cest a spojový seznam všech bodů. Tyto body by obsahovaly jednosměrný spojový seznam všech hran, které vedou právě z těchto bodů. Implementace spojového seznamu je velice jednoduchá a rychlá, snadno se s ní pracuje. Paměťová náročnost je vcelku malá, pokud se zvolí vhodné datové typy. Místo spojového seznamu hran by bylo vhodné vytvořit strukturu pro uchovávání indexů bodů a pozici v poli, pro rychlejší vyhledávání.

Další způsob je **matice sousednosti**. Pro konečnou množinu vrcholů V grafu G , kterých je n , má podobu čtvercové matice $n \times n$, jejíž hodnota na místě a_{ij} je celé číslo odpovídající počtu hran vedoucích z vrcholu i do vrcholu j ². Pokud je graf neorientovaný, hodnota na místě $a_{ij} = a_{ji}$ z důvodu symetrie matice. Pro orientovaný graf platí, že pokud do bodu vede hrana, její hodnota bude záporná, zatímco pro hranu která vychází z bodu, bude hodnota kladná.



Obrázek 3: Matice sousednosti neorientovaného grafu



Obrázek 4: Matice sousednosti orientovaného grafu

Nástin implementace: Matici sousednosti je vhodné implementovat pouze v případě, platí-li $|H| \sim |V|^2$, kde H je počet hran v grafu a V je počet vrcholu v grafu.

Pro případ, kdy platí $|H| \ll |V|^2$, je paměťová náročnost velice vysoká, protože z většiny vrcholů bude vést malý počet hran a zbytečně se bude zabírat místo v paměti pro nulové hodnoty. Zároveň i časová náročnost bude velice vysoká, z důvodu projíždění celé matice a kontroly, zda na aktuální pozici existuje nějaká hrana mezi dvěma vrcholy. Tento případ je velice nevhodný.



Další vhodná reprezentace grafu je reprezentace pomocí **klasických polí**. Pomocí polí lze nacházet hodnoty rychleji, než je tomu u spojových seznamů, pokud zadáme index, na kterém se požadovaná hodnota nachází. Nástin implementace: Oproti implementaci spojového seznamu se do grafu uloží pole vrcholů a pole hran. U hran si stačí ukládat pouze jejich index a hodnotu, a pro vrcholy zase jejich index a pole ukazatelů na sousedy a hrany mezi nimi. Oproti implementaci spojového seznamu by tato reprezentace mohla být o malinko náročnější na paměť, ale rychlost vyhledávání v polích by se mohla několikanásobně zrychlit. Právě tuto reprezentaci jsem zvolil.

3 Popis implementace

Jak už bylo řečeno na konci analýzy, graf jsem se rozhodl reprezentovat pomocí několika polí.

3.1 Struktura pro vrcholy

Vrcholy jsou v programu reprezentovány strukturou **vertex**. Vertex obsahuje:

- index - číselný název vrcholu v grafu
- neighbours - číselný ukazatelé na název souseda (vrcholu)
- neighboursIndex - číselný ukazatelé na pozici souseda (vrcholu), jelikož vrcholy nejsou indexovány od 0 po maximální Index po 1, je třeba uchovávat pro rychlejší přístup i indexy v poli grafu
- edges - číselný ukazatelé na pozici hrany v poli hran

Např. zjištění prvního souseda a přes jakou hranu se k němu dostanu následovně:

1. **neighbours[0]** = název souseda
2. **neighboursIndex[0]** = hodnota, která obsahuje index do pole vertexů v grafu, kde tento bod najdu
3. **edges[0]** = hodnota, která obsahuje index do pole edge, kde tuto hranu najdu

3.2 Struktura pro hrany

Hrany jsou v programu reprezentovány strukturou **edge**. Edge obsahuje:

- value - struktura date, která obsahuje hodnoty rok, měsíc a den.

3.3 Struktura pro graf

Graf jsou v programu reprezentovány strukturou **graph**. Graph obsahuje:

- vertexes - pole všech bodů v grafu
- edges - pole všech hran v grafu



3.4 Struktura pro nalezené cesty

Zjištěné cesty jsou ukládány do struktury **path**. Path obsahuje:

- **metric** - rozdíl dní mezi nejstarším a nejnovějším datem (cenou hrany) v cestě
- **vertexes** - číselný ukazatelé na název vrcholu, přes který cesta vede
- **edges** - číselný ukazatelé na pozici hrany v poli hran

4 Uživatelská příručka

4.1 Překlad aplikace

Pro překlad je třeba použít **GCC (GNU Compiler Collection)**. Postup přeložení je následující:

1. Otevřít command line/terminal.
2. V terminalu/command line se dostat do složky, kde je uložen soubor makefile (kořenový adresář aplikace).
3. Napsat příkaz make/mingw32-make, podle typu OS a typu kompilátoru

```
C:\Users\Vojtěch\Desktop\PC\SEMESTRÁLKA>mingw32-make
gcc -c src/structures.c -o structures.o
gcc -c src/free_function.c -o free_function.o
gcc -c src/dfs.c -o dfs.o
gcc structures.o free_function.o dfs.o -o dfs
rm -f *.o
C:\Users\Vojtěch\Desktop\PC\SEMESTRÁLKA>_
```

Obrázek 5: Úspěšný překlad aplikace

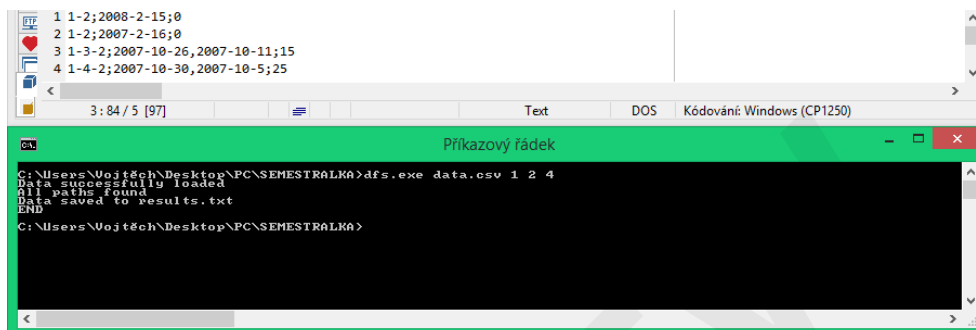
4.2 Spuštění a ovládání aplikace

Program se spustí následujícím příkazem:

dfs.exe <*soubor-grafu*> <*id1*> <*id2*> <*maxD*>

Formát souboru: **CSV (Comma-separated values)** - jednoduchý souborový formát pro výměnu tabulkových dat. Soubor ve formátu CSV sestává z řádků, ve kterých jsou jednotlivé položky odděleny znakem (v našem případě) středníkem (;) ³.

Aplikace se víceméně nijak neovládá, uživatel zadá příkaz, program následně otestuje parametry (zda je soubor formátu *csv*, *id1* / *id2* / *maxD* jsou číslíce a větší než 0, po načtení dat ještě jestli se *id1* a *id2* nachází v poli bodů) a následně vypisuje informační výstup (data načtena, cesty nalezeny, uložení do souboru a konec aplikace). Veškeré nalezené cesty nalezneme v kořenové složce v souboru *results.txt*.

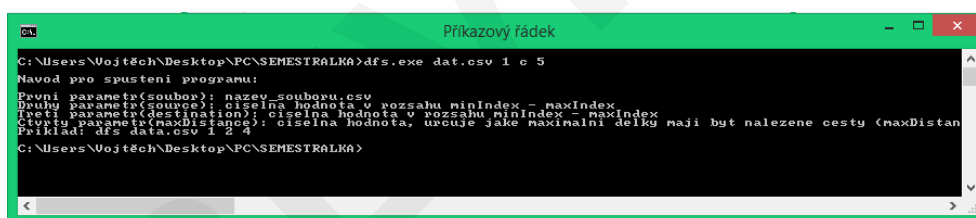


```
1 1-2;2008-2-15;0
2 1-2;2007-2-16;0
3 1-3-2;2007-10-26,2007-10-11;15
4 1-4-2;2007-10-30,2007-10-5;25

3: 84 / 5 [97] Text DOS Kódování: Windows (CP1250)

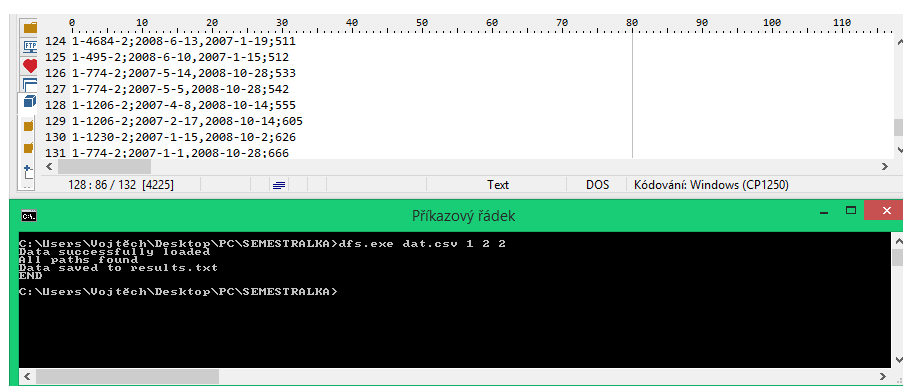
Příkazový řádek
C:\Users\Vojtěch\Desktop\PC\SEMESTRALKA>dfs.exe data.csv 1 2 4
Data successfully loaded
All paths found
Data saved to results.txt
END
C:\Users\Vojtěch\Desktop\PC\SEMESTRALKA>
```

Obrázek 6: Kontrolní výstup 1



```
Příkazový řádek
C:\Users\Vojtěch\Desktop\PC\SEMESTRALKA>dfs.exe dat.csv 1 c 5
Navod pro spusteni programu:
Prvni parametr(soubor): nazev souboru.csv
Druhy parametr(source): ciselna hodnota v rozsahu minIndex - maxIndex
Treti parametr(destination): ciselna hodnota v rozsahu minIndex - maxIndex
Ctyry parametry(maxDistances): ciselna hodnota, urcuje jak maximalni delky maji byt nalezene cesty (maxDistances)
Priklad: dfs data.csv 1 2 4
C:\Users\Vojtěch\Desktop\PC\SEMESTRALKA>
```

Obrázek 7: Kontrolní výstup 2



```
0 10 20 30 40 50 60 70 80 90 100 110
124 1-4684-2;2008-6-13,2007-1-19;511
125 1-495-2;2008-6-10,2007-1-15;512
126 1-774-2;2007-5-14,2008-10-28;533
127 1-774-2;2007-5-14,2008-10-28;542
128 1-1206-2;2007-4-8,2008-10-14;555
129 1-1206-2;2007-2-17,2008-10-14;605
130 1-1230-2;2007-1-15,2008-10-2;626
131 1-774-2;2007-1-1,2008-10-28;666

128: 86 / 132 [4225] Text DOS Kódování: Windows (CP1250)

Příkazový řádek
C:\Users\Vojtěch\Desktop\PC\SEMESTRALKA>dfs.exe dat.csv 1 2 2
Data successfully loaded
All paths found
Data saved to results.txt
END
C:\Users\Vojtěch\Desktop\PC\SEMESTRALKA>
```

Obrázek 8: Kontrolní výstup 3



5 Závěr

I přes různé komplikace (špatné přiřazování paměti atributům struktur, žádné zkušenosti s jazykem C, ...) si myslím, že jsem vytvořil program, který splňuje všechny požadavky, který by měl být dále rozšiřitelný o další funkce. Program by měl zaznamenat všechny špatně zadané parametry, kontrolovat, zda soubor s daty neobsahuje špatná data a další kontroly.

Bohužel při začátku psaní aplikace jsem nějak nedbal na kvalitu struktury a použití nejvhodnějších datových typů pro uložení dat, což mi dost zkomplikovalo programování. Až po nějaké době programování aplikace (víceméně při počítání všech cest) jsem zjistil, že na malý počet cest je potřeba hodně času a také aplikace zabírala poměrně dost paměti. Po této zkušenosti bude lepší si vždy před programováním udělat pořádný návrh, jak struktury a aplikace bude v budoucnu vypadat.

6 Literatura

¹ Wikipedia - Spojový seznam

² Wikipedia - Matice sousednosti

³ Wikipedia - CSV