FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY A VÝPOČETNÍ TECHNIKY

Semestrální práce z předmětu KIV/PC

Vyhledávání cest v grafu technikou DFS

Filip Hácha – A16B0036P

hachaf@students.zcu.cz

1. ledna 2018

Obsah

1 Úvod

Tento dokument popisuje řešení semestrální práce z předmětu *Programování v jazyce C*. Popis zahrnuje rozbor zadání, návrh algoritmu, popis implementace i návod k překladu a spuštění hotové aplikace.

2 Zadání

Cílem práce bylo vytvořit přenositelnou konzolovou aplikaci, která bude procházet graf popsaný ve vstupním souboru technikou DFS (Depth-First Search), v zadaném grafu najde všechny cesty mezi dvěma vrcholy zadanými v parametrech při spuštění aplikace a cesty vypíše do konzole.

2.1 Vstupní soubor

Vstupní soubor aplikace je soubor ve formátu `CSV` a obsahuje tolik řádek, kolik je v grafu hran a každá řádka obsahuje informaci o tom, z mezi kterými vrcholy hrana leží a jaké má ohodnocení.

Ohodnocení hran je udáváno jako datum.

2.2 Výstup aplikace

Výstup aplikace se vypíše do konzole a obsahuje tolik řádek, kolik je v grafu cest mezi zadanými vrcholy a o maximální zadané délce. U cest je vypsána nejprve posloupnost vrcholů, dále jednotlivá ohodnocení hran a na konec celkové ohodnocení cesty, které odpovídá počtu dnů mezi maximálním a minimálním ohodnocením hran na cestě.

Řazení cest je vzestupné podle jejich délky a při shodné délce se aplikuje řazení vzestupně podle vypočteného rozdílu ohodnocení.

3 Analýza problému

Zadaný problém můžeme rozdělit na několik hlavních segmentů, u kterých je třeba zvážit možné způsoby řešení a pokusit se vybrat optimální.

3.1 Datové struktury

Hlavním problémem v oblasti reprezentaci dat v aplikaci je reprezentace zadaného grafu v paměti. Zde máme na výběr z několika možností. Tyto mož-

nosti můžeme rozdělit do dvou hlavních skupin - spojové struktury a maticové struktury.

Hlavní výhodou maticových struktur je možnost přímého přístupu ke konkrétní hraně či vrcholu, kdežto u spojových struktur obvykle musíme nejprve projít určité množství hran (vrcholů).

Spojové struktury jsou však efektivnější v oblasti paměťové náročnosti, pokud si nemůžeme být jisti, že budeme pracovat s grafem, jehož počet hran se blíží plnému grafu.

Další datovou strukturu, kterou při algoritmu potřebujeme, je struktura, která nám umožní ukládat mezivýsledky při rekurzivním volání funkce pro prohledávání grafu. Zde se nabízí abstraktní datový typ zásobník, neboť jeho způsob přístupu k uloženým prvkům se hodí pro rekurzivní algoritmy.

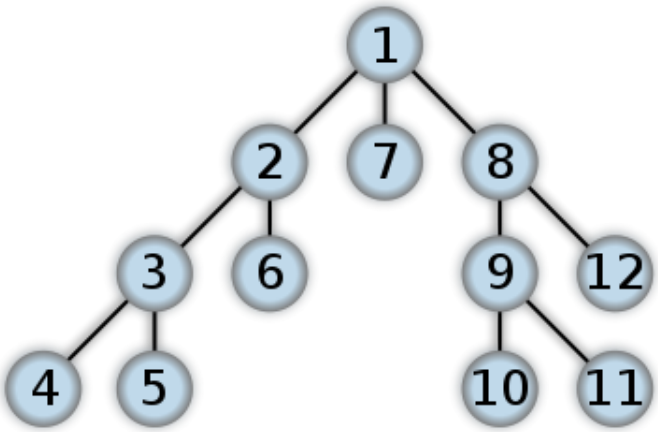
Další datové struktury jsou buďto primitivní výčtové typy, nebo pole.

3.2 Hledání cest

Způsob, kterým se prohledává graf pro nalezení cest je do značné míry dán zadáním, neboť musí jít o metodu prohledávání grafu do hloubky (DFS).

Prohledávání do hloubky (v angličtině označované jako depth-first search nebo zkratkou DFS) je grafový algoritmus pro procházení grafů metodou backtrackingu, viz Obrázek 1. Pracuje tak, že vždy expanduje prvního následníka každého vrcholu, pokud jej ještě nenavštívil. Pokud narazí na vrchol, z nějž už nelze dále pokračovat (nemá žádné následníky nebo byli všichni navštíveni), vrací se zpět backtrackingem. [1]

Jeden z problémů tkví v tom, že dopředu nevíme, kolik cest nalezneme. Můžeme tedy nalezené cesty ukládat do některé z dynamických datových struktur, nebo si nejprve zjistit počet cest a až teprve ve druhém průchodu cesty ukládat.



Obrázek 1: Pořadí v jakém je přistupováno k vrcholům při prohledávání do hloubky

3.3 Výpočet rozdílu extrémních ohodnocení hran

Po nalezení cest je potřeba pro každou z nich určit počet dnů mezi maximálním a minimálním ohodnocením hran, které cestu tvoří.

Pro tento účel stačí použít funkci, kterou popisuje Obrázek 2 a která vychází z přepočtu na číslo Juliánského dne. [2]

3.4 Řazení cest

Nalezené cesty je třeba seřadit v zadaném pořadí a jelikož kritéria řazení jsou dvě, můžeme buď to implementovat libovolný řadicí algoritmus, který při porovnávání dvou cest zohlední obě kritéria, nebo použít některý ze stabilních algoritmů pro řazení a nejprve cesty seřadit podle sekundárního kritéria a poté podle primárního - tedy nejprve podle počtu dní mezi mezními daty a poté podle jejich délky.

4 Popis datových struktur

4.1 Reprezentace grafu

Načtený graf je rezprezentován jako pole spojových seznamů, kde jeden spojový prvek odpovídá jedné hraně. V poli na indexu i tedy leží spojový seznam obsahující všechny hrany incidentní s vrcholem i.

```
int get_days_between(struct Date * date1, struct Date* date2)
    {
    int d1 = date1->day;
    int d2 = date2->day;
    int m1 = (date1->month + 9) % 12;
    int m2 = (date2->month + 9) % 12;
    int y1 = date1->year - (m1 / 10);
    int y2 = date2->year - (m2 / 10);
    int f1 = 365 * y1 + y1/4 - y1/100 + y1/400
      + (m1 * 306 + 5)/10 + (d1 - 1);
    int f2 = 365 * y2 + y2/4 - y2/100 + y2/400
      + (m2 * 306 + 5)/10 + (d2 - 1);
    int diff = f1 - f2;
    if (diff > 0) {
        return diff;
    } else {
        return -diff;
    }
}
```

Obrázek 2: Funkce pro výpočet počtu dnů mezi dvěma datumy.

Každá hrana tedy obsahuje informaci o svém druhém vrcholu (první je jasně definován indexem v poli) a o svém ohodnocení.

4.2 Reprezentace cest

Nalezené cesty jsou uloženy do dvourozměrného celočíselného pole, jehož rozměry jsou dány počtem nalezených cest v grafu a maximální zadanou délkou cesty. Při vytváření pole jsou jeho hodnoty inicializovány na hodnotu -1 a později při zápisu nalezených hran jsou tyto hodnoty nahrazovány identifikátory vrcholů.

5 Popis algoritmu

5.1 Načítání grafu

Načítání grafu ze vstupního souboru probíhá ve dvou průchodech. Při prvním průchodu algoritmus nalezne největší identifikátor vrcholu, který se v popisu grafu vyskytuje a podle jeho velikosti vytvoří v paměti příslušné pole pro spojové prvky hran.

Ve druhém průchodu řádek po řádku vytváří spojové prvky reprezentující hrany grafu a přidává je na konce spojových seznamů na příslušných indexech připraveného pole.

5.2 Zjištění počtu cest

První aplikace metody prohledávání grafu DFS slouží pouze k zjištění počtu cest mezi zadanými vrcholy odpovídající maximální délce uvedené v parametrech.

Funkce zajišťující samotné prohledávání grafu je volána rekurzivně. Např. při hledání cest mezi vrcholy A a B algoritmus nalezne sousedy vrcholu A a opět pro ně zavolá stejnou funkci se stejným cílovým vrcholem.

Z nalezených úseků sestaví celkový počet cest, podle něhož se připraví struktura pro uložení cest samotných.

5.3 Nalezení cest

Druhá aplikace metody DFS je téměř identická s případem zjišťování počtu cest s tím rozdílem, že cesty se již ukládají do předpřipraveného dvourozměrného pole.

Dvojité prohledávání grafu nezvyšuje asymptotickou složitost algoritmu, proto nemá zásadní vliv na dobu běhu aplikace.

5.4 Nalezení extrémních ohodnocení hran a jejich rozdílu

Jako další krok algoritmus pro každou cestu nalezne hranu s nejnižším a hranu s nejvyšším ohodnocením, které uloží do polí o velikosti počtu cest. Dále se spočtou rozdíly mezi těmito hodnotami pro každou cestu ve dnech, které jsou stejným způsobem uloženy v poli.

5.5 Řazení cest

Řazení cest probíhá ve dvou krocích, nejprve se cesty seřadí vzestupně podle vypočtených rozdílů mezních hodnot hran. Poté se cesty seřadí vzestupně podle jejich délky.

Řazení cest je stabilní, díky čemuž u cest se shodnými délkymi zůstane zachováno původní řazení podle rozdílů ohodnocení hran.

5.6 Výpis výsledku

Nyní se již pouze vypíšou cesty v seřazeném poli jako posloupnost vrcholů, dále se vypíše ohodnocení všech hran, které je tvoří a jejich vypočtený rozdíl mezních ohodnocení.

6 Překlad a spuštění aplikace

6.1 Překlad

Odevzdávaný archiv se semestrální prací obsahuje kompletní soubory zdrojových kódů aplikace a soubor `Makefile`, jehož spuštění zajistí překlad aplikace s použitím překladače *GCC*. Pro překlad aplikace v prostředí systému Windows je třeba mít nainstalované vývojové prostředí, které obsahuje příslušný překladač, např. *MinGW*.

6.2 Spuštění aplikace

Přeloženou aplikaci je možné spustit příšlušným příkazem s názvem aplikace a správnými parametry v prostředí příkazového řádku nebo Unixového terminálu.

Obrázek 3 ukazuje spuštění aplikace v prostředí příkazového řádku na systému Windows a Obrázek 4 ukazuje spuštění v terminálu na Unixových systémech (konkrétně systém ElementaryOS). Místo výrazů v lomených závorkách je třeba uvést cestu ke vstupnímu CSV souboru s grafem, identifikátory vrcholů, mezi nimiž má algoritmus cesty hledat a maximální délku cesty.

```
dfs.exe <cesta vst. souboru> <id1> <id2> <max. delka cesty>
```

Obrázek 3: Spuštění na systému Windows.

```
$ ./dfs <cesta vst. souboru> <id1> <id2> <max. delka cesty>
```

Obrázek 4: Spuštění na systému Linux.

7 Závěr

Vytvořená aplikace implementuje zadaný algoritmus prohledávání grafu metodou DFS. Aplikace umožňuje nalezení všech cest mezi dvěma zadanými vrcholy splňující podmínku maximální délky uvedené v parametrech a to i na nesouvislých grafech.

Během testování aplikace na poskytnutých testovacích datech nedošlo k jejímu pádu ani k neúspěšnému pokusu o vyhledání cest.

Reference

[1] Prohledávání do hloubky - Wikipedia
https://en.wikipedia.org/wiki/Depth-first_search
Naposledy navštíveno 01. ledna 2018.

[2] Julian day - Wikipedia
https://en.wikipedia.org/wiki/Julian_day
Naposledy navštíveno 01. ledna 2018.