Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Vyhľadávanie spojení mestskou hromadnou dopravou

Bakalárska práca

2018
Alojz Stúpal

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Vyhľadávanie spojení mestskou hromadnou dopravou

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	RNDr. Michal Forišek, PhD.

Bratislava, 2018
Alojz Stúpal

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Alojz Stúpal
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Vyhľadávanie spojení mestskou hromadnou dopravou
Finding connections using public transport

Anotácia: Práca sa zaoberá vyhľadávaním a zobrazovaním možností cesty MHD.

Cieľ: Práca má nasledujúce ciele:
1. Spracovať prehľad základných algoritmov vhodných na vyhľadávanie spojenia v grafikone MHD.
2. Implementovať softvér umožňujúci takéto vyhľadávanie a otestovať jeho funkčnosť na reálnych dátach.
3. Pokúsiť sa navrhnúť a implementovať možné vylepšenia, ako napr.: vyhľadávanie cesty bod-bod namiesto zastávka-zastávka; vizualizácia možných alternatívnych trás; výpočet dodatočných informácií, napr. očakávaného meškania v prípade nestihnutia prestupu.

Vedúci: RNDr. Michal Forišek, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: prof. RNDr. Martin Škoviera, PhD.

Dátum zadania: 07.11.2017

Dátum schválenia: 08.11.2017 doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie: Tu môžete poďakovať školiteľovi, prípadne ďalším osobám, ktoré vám s prácou nejako pomohli, poradili, poskytli dáta a podobne.

Abstrakt

Slovenský abstrakt v rozsahu 100-500 slov, jeden odstavec. Abstrakt stručne sumarizuje výsledky práce. Mal by byť pochopiteľný pre bežného informatika. Nemal by teda využívať skratky, termíny alebo označenie zavedené v práci, okrem tých, ktoré sú všeobecne známe.

Kľúčové slová: jedno, druhé, tretie (prípadne štvrté, piate)

Abstract

Abstract in the English language (translation of the abstract in the Slovak language).

Keywords:

Obsah

Úvod **1**

1 Grafy **2**

1.1 Definícia grafu 2

1.2 Základné pojmy týkajúce sa grafov 2

1.2.1 Typy grafov 3

1.2.2 Vzťahy medzi grafmi 3

1.3 Cesty a cykly 4

1.4 Súvislosť 4

1.5 Kostry, stromy a lesy 5

1.6 Špeciálne grafové štruktúry 5

1.6.1 Cesty a cykly pre orientované grafy 6

2 Algoritmy na grafoch **7**

2.1 Najlacnejšie cesty v grafe 7

2.1.1 Dijkstrov algoritmus 8

2.1.2 Bellman - Fordov algoritmus 10

2.1.3 Floyd - Warshallov algoritmus 10

2.1.4 Johnsonov algoritmus 10

2.1.5 Johnsonov algoritmus 10

3 Softvér **11**

4 Implementácia **12**

Záver **13**

Zoznam obrázkov

1 Orientovaný graf s ohodnotenými hranami 9

Zoznam tabuliek

Úvod

Cieľom tejto práce je

Kapitola 1

Grafy

V tejto kapitole uvedieme zopár definícií týkajúcich sa grafov, opíšeme niektoré grafové štruktúry, prípadne zavedieme rôzne názvy, ktoré budeme v ďalších častiach práce potrebovať.

1.1 Definícia grafu

Začneme so základnou defínicou - s definovaním pojmu *graf*. Tú prevezmeme od Reinharda Diestela [1, kapitola 0.1]:

Graf je dvojica $G = (V, E)$ disjunktných množín, kde prvky E sú dvojprvkové podmnožiny V. Prvky V sú *vrcholy* (prípadne *uzly* alebo *body*) grafu G, prvky E sú jeho hrany.

Vytvorený objekt by sa mal korektne nazývať *neorientovaný graf*. Avšak my, ako aj mnohí iní, od tohto pomenovania upustíme a budeme pre jednoduchosť používať len pojem *graf*.

Hranu $\{x, y\}$, kde $x, y \in V$, budeme zasa zvyčajne označovať aj ako (x, y) alebo xy.

1.2 Základné pojmy týkajúce sa grafov

V nasledujúcej podkapitole sú uvedené definície rôznych pojmov, ktoré charakterizujú a popisujú vlastnosti grafových štruktúr.

Rád grafu definujeme ako počet vrcholov grafu G.

Budeme hovoriť, že vrchol v je *incidentný* s hranou e, ak $v \in e$. Čiže, ak je vrchol

v jedným z vrcholov, ktoré hrana e spája, nazveme ho incidentným s tou hranou. Ak sú dva rôzne vrcholy incidentné s tou istou hranou, budeme ich volať jej *koncovými vrcholmi*.

Dva vrcholy $x, y \in V$ sú *susedné*, ak xy je hrana v G. Dve hrany $e, f \in E; e \neq f$ sú *susedné*, ak majú spoločný vrchol.

Stupňom vrchola $v \in V$ nazveme počet hrán incidentných s v. Alebo, inak povedané, je to počet vrcholov, ktoré sú susedné s vrcholm v.

1.2.1 Typy grafov

Teraz uvedieme niektoré základné typy grafov.

Prázdny graf je graf $G = (\emptyset, \emptyset)$. Označenie zjednoduchšíme na $G = \emptyset$.

Pojmom *Triviálny graf* budeme označovať graf s rádom 0 alebo 1.

Kompletným grafom nazveme taký graf, ktorého všetky vrcholy sú navzájom susedné. Uvažujme n vrcholov, potom kompletný graf na týchto vrcholoch označíme K_n. Pre predstavu môže poslúžiť príiklad kompletného graf troch vrcholov K_3, čo je trojuholník.

Majme graf G. Ak každý jeho vrchol má rovnaký stupeň k, potom tento graf nazveme *k-regulárnym*. V definícií môžeme upustiť od počtu vrcholov a nazvať vzniknutý objekt iba *regulárnym*.

Nech $r \in N, r \geq 2$. Graf $G = (V, E)$ voláme *r-partitný*, ak V umožňuje rozklad na r tried (partícií), a to takých, že každá hrana má svoje konce v rôznych triedach. Alebo inak, vrcholy v tej istej triede rozkladu so sebou nesmú susediť. 2-partitným grafom budeme vravieť *bipartitné*. Ľahko si bipartitný graf predstaviť uvažujúc dve množiny vrcholov, pričom sa hrany grafu nachádzajú len a len medzi týmito množinami, nie v nich.

1.2.2 Vzťahy medzi grafmi

Nech $G = (V, E)$ a $G' = (V', E')$ sú grafy. Označíme $G \cap G' = (V \cap V', E \cap E')$ a $G \cup G' = (V \cup V', E \cup E')$. Ak prienik týchto dvoch grafov je prázdny graf $G \cap G' = \emptyset$, potom G a G' sú *disjunktné*. Ak $V' \subseteq V$ a $E' \subseteq E$, potom hovoríme, že G' je podgraf

G (alebo G je nadgraf G', či G obsahuje G') a píšeme $G' \subseteq G$.

1.3 Cesty a cykly

Nech $G = (V, E)$ je graf. Definujme *cestu*, ako neprázdny graf $P = (V', E')$, ktorý je podgrafom G, tvaru $V' = \{x_0, x_1, ..., x_k\}$ a $E' = \{x_0x_1, x_1x_2, ..., x_{k-1}x_k\}$, kde všetky x_i sú navzájom rôzne. Neformálne povedané, cesta spája (pomocou hrán) dva vrcholy grafu G, pričom sa v nej použité vrcholy nesmú opakovať. Počet hrán cesty sa nazýva aj *dĺžkou*. Pre zjednoduchšenie budeme cestu zapisovať ako $P = x_0x_1x_2...x_{k-1}x_k$.

Majme graf $G = (V, E)$. Nech $P = (V', E')$ je cesta v G, $P = x_0x_1, x_1x_2, ..., x_{k-1}x_k$, pričom $k \geq 3$ a v grafe G existuje hrana x_0x_k. Potom hovoríme, že graf G obsahuje *cyklus* alebo tiež *kružnicu* C, kde C je podgraf G a platí $C = (V', E' \cup \{x_0, x_k\})$.

Sled v grafe G je názov pre neprázdnu striedavú postupnosť $v_0e_0v_1e_1...e_{k-1}v_k$ vrcholov a hrán v G, pričom $e_i = v_iv_{i+1}$ pre všetky $i < k$. Sled teda, narozdiel od cesty, môže obsahovať rovnaké vrcholy viac krát. Preto sa dá povedať, že sled je cesta, v ktorej sa môžu vrcholy opakovať. Ale aj naopak: sled, v ktorom sú vrcholy navzájom rôzne, je cestou.

Sled, v ktorom sú hrany navzájom rôzne, sa nazýva *ťah*.

1.4 Súvislosť

Ak sú ľubovoľné dva vrcholy grafu G spojené cestou, tento graf nazveme *súvislým*.

Komponent grafu G je súvislý podgraf grafu G taký, že nie je obsiahnutý v žiadnom väčšom súvislom podgrafe grafu G. Napríklad, súvislý graf má práve jeden komponent, a to samého seba.

Hrana $e \in E$ grafu G sa volá *most*, ak graf G má menší počet komponentov v porovnaní s grafom G bez hrany e. Čiže, po odstránení hrany z grafu pribudne práve jeden komponent.

Vrchol $v \in V$ grafu G sa nazýva *artikulácia*, ak počet komponentov grafu G je menší ako ich počet po odstránení v z grafu G. Teda, ak po odstránení vrchola z grafu

pribudne aspoň jeden komponent.

1.5 Kostry, stromy a lesy

Nech $G = (V, E)$ je súvislý graf. *Kostrou* grafu nazveme taký graf $G' = (V', E'), G' \subseteq G$, pre ktorý platí $V' = V$ a navyše medzi každými dovma vrcholmi existuje práve jedna cesta. Ak by sme grafu G postupne odnímali hrany tak, aby sme nenarušili jeho súvislosť, tak po odstránení všetkých takýchto hrán by sme získali kostru grafu G. Z tejto konštrukčnej definície vyplýva, že všetky hrany v kostre grafu sú mostami. Je z nej taktiež zrejmé, že každá kostra grafu je súvislá.

Kostrový les je taký graf G, pre ktorý platí, že každý z jeho komponentov je kostrou.

Acyklický graf (to jest taký, ktorý neobsahuje cyklus), ktorý je navyše súvislý, nazveme *stromom*. Všetky vrcholy stromu so stupňom 1 sú jeho *listy*. Je vhodné si povšimnúť, že, ako pri kostre, všetky hrani sú mostami. Taktiež je zaujímavé uvažovať o rozdiele medzi stromom a kostrou grafu. Možno nahliadnuť, že jediným rozdielom medzi nimi je, že strom je grafom, zakiaľ čo kostra je len podgrafom nejakého daného grafu.

Les je acyklický graf. Takže každý jeho komponent je stromom.

1.6 Špeciálne grafové štruktúry

Graf, ktorého prvky sú ohodnotené číslom určujúcim cenu, prípadne výhodnosť prechodu cezeň, nazveme *ohodnotený graf*. *Hranovo ohodnotený graf* je graf s funkciou $w : E \to R$. Teda pre ľubovoľnú hranu e existuje číslo $w(e)$, ktoré nazveme *hodnotou hrany*, prípadne *cenou hrany*. Z praktického hľadiska je výhodné zadefinovať aj *kladne hranovo ohodnotený graf*. A to je taký, že $w(e) > 0$ pre všetky hrany. *Vrcholovo ohodnotený graf* je graf s funkciou $w : V \to R$. Teda pre ľubovoľný vrhol v existuje číslo $w(v)$.

Orientovaný graf je dvojica $G = (V, E)$, kde každý prvok $xy \in E$ vyjadruje usporiadanú dvojicu. Neformálne, ak je daná hrana xy, ale nie yx, existuje cesta dĺžky 1 z vrcholu x do y, ale z vrcholu y do x takáto cesta nejestvuje.

Rozdiel medzi orientovanými a neorientovanými grafmi je, ako názov napovedá v existencií orientácií hrán. Ak by sme túto skutočnosť chceli vyjadriť formálne, povedali by sme, že zatiaľ čo v neorientovanom grafe obsahuje množina E dvojprvkové množiny, v orientovanom sa skladá z usporiadaných dvojíc.

1.6.1 Cesty a cykly pre orientované grafy

Kapitola 2

Algoritmy na grafoch

Táto kapitola obsahuje popisy algoritmov, ktoré sú aplikovateľné na grafové štruktúry a navyše využiteľné pri vyhľadávaní spojení hromadnej dopravy. Formálnu charakterizáciu algoritmu zväčša doprevádza i jeho voľnejšia interpretácia, ktorá má za účel uľahčiť porozumenie čitateľom, ktorí sa s daným algoritmom doposiaľ nestretli. Naviac, pri algoritmoch sú uvedené výhody, respektíve nedostatky, pri jeho použití na nami vytýčený cieľ. V hojnom počte budeme využívať definície z kapitoly 1.

2.1 Najlacnejšie cesty v grafe

Pri vyhľadávaní spojení mestskou hromadnou dopravou sa zdá byť veľmi racionálne zaoberať sa nachádzaním najlacnejších ciest v grafikone MHD. Dovoľujeme si tak tvrdiť, pretože najväčší dôraz cestujúcich je kladený práve na čo najskorší príchod do požadovanej lokality. Pre úplnosť ešte dodávame, že cena cesty, ako je asi zrejmé, je súčet hodnôt hrán (respektíve vrcholov), ktoré daná cesta obsahuje. Z doposiaľ uvedeného vyplýva, že pri našich úvahách budeme používať ohodnotené grafy, kde pridelenou hodnotou bude čas medzi zastávkami. Navyše musíme nejako ošetriť i vrcholy grafu - zastávky, keďže na nich zvyčajne čakáme na prestup na ďalší spoj. Od tejto myšlienky však spočiatku upustíme a budeme sa jej venovať až neskôr. A posledná úvaha - hranami v našom grafe sú linky MHD, sme preto nútení použiť orientované grafy.

Ak uvažujeme vyhľadávanie najlacnejších ciest v grafe, musíme si najprv uvedomiť, čo presne je našim cieľom. Výledok, ktorý chceme dosiahnuť ako výstp algoritmu, má byť najlacnejšia cesta z počiatočného bodu do koncového. Avšak znalosť problematiky algoritmov na grafových štruktúrach nám ponúka riešenia iného, koplexnejšieho problému s asymptoticky rovnakou časovou zložitosťou. Týmto problémom je vyhľadávanie najlacnejších ciest z počiatočného vrcholu do všetkých ostatných vrcholov. Ľahko nahliadnuť, že riešeim tohto problému dostaneme odpoveď aj na našu počiatočnú otázku.

Preto sa v nasledujúcej časti budeme zaoberať algoritmami riešiacimi túto úlohu.

Ľahko spozorovať, že by mohlo byť v určitých situáciách výhodné vypočítať ceny a nájsť najlacnejšie cesty pre všetky dvojice vrcholov. Hlavne, keď graf obsahuje malé množstvo vrcholov - zastávok. Mnoho miest má len malú sieť mestskej hromadnej dopravy, a teda by bolo rozumné vypočítať všetky potrebné údaje naraz na začiatku, a potom, pri prijímaní dotazu na vyhľadanie spojenia, jednoducho vypísať už vypočítanú odpoveď. Z tohto dôvodu uvedieme i algoritmy riešiace tento problém.

2.1.1 Dijkstrov algoritmus

Holandský informatik, po ktorom je tento algoritmus pomenovaný, dokázal, okrem iného, vyriešiť aj nami nastolený problém. V jeho riešení je ale potrebné, aby boli ceny hrán grafu nezáporné reálne čísla, čo súhlasí s našou predstavou aplikovania algoritmu na grafikon MDH. Algoritmus je navrhnutý tak, že dostane ako vstup graf $G = (V, E)$, počiatočný vrchol v_0 a hodnotiacu funkciu $h : V \times V \to R_0^+$. Predpokladáme, že ak hrana $uv \notin E$, potom $h(u, v) = \infty$, ďalej, že $h(u, u) = 0$ a nakoniec, že funkciu h je možné vypočítať v konštantnom čase $O(1)$. Máme taktiež jeden predpoklad na vrcholy grafu, a to, že sú reprezentované celými číslami $1, 2, ..., k$ (kde k je počet vrcholov grafu). Nakoniec, výsledkom algoritmu bude pole čísel D, v ktorom bude pre každý vrchol $v \in V$ vypočítaná hodnota $D[v]$, čo je cena najlacnejšej cesty z počiatočného vrchola v_0 do vrchola v.

Tieto požiadavky, predpoklady, ba i vstup a výstup funkcie sú uvažované v horeuvedenom tvare len aby sme mohli predviesť implementáciu algoritmu od Pavla Ďuriša, ktorú možno nájsť aj v jeho knihe [7, kapitola 2.2.1].

begin

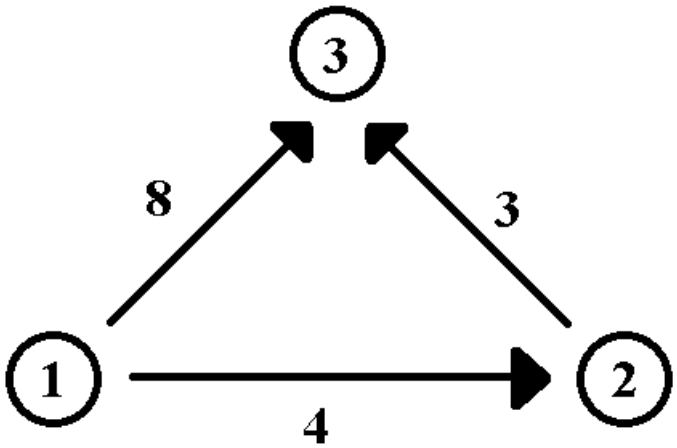
$S \leftarrow \{v_0\}$
$D[v_0] \leftarrow 0$
pre každý vrchol $v \in V \setminus \{v_0\}$: $D[v] \leftarrow h(v_0, v)$
while $S \neq V$ **do begin**
    vyber $w \in V \setminus S$ *taký, že hodnota* $D[w]$ *je minimálna*
    $S \leftarrow S \cup \{w\}$
    pre každý $v \in V \setminus S$:
        $D[v] \leftarrow min\{D[v], D[w] + h(w, v)\}$
end

end

algoritmus 1: Dijkstrov algoritmus

Časová zložitosť dijkstrovho algoritmu je $O(|V|^2)$, čo je dokázané spolu s korektnosťou algoritmu v už uvedenom zdroji od Pavla Ďuriša [7, kapitola 2.2.1].

Priebeh dijkstrovho algoritmu vysvetlíme na jednoduchom príklade.



Obr. 1: Orientovaný graf s ohodnotenými hranami

Keďže je našim cieľom nie ani tak zistiť cenu najlacnejšej cesty, ale skôr takéto cesty nájsť, dijkstrov algoritmus by sme potrebovali jemne modifikovať, a to tak, aby sme vedeli spätne zrekonštruovať nájdené cesty. Teda so znalosťou konečného vrchola by sme chceli vedieť vygenerovať postupnosť vrcholov, cez ktoré sme sa do finálneho

vrchola dostali. Budeme si preto pamätať pri každom vrchole informáciu, z ktorého vrchola sme sa doň dostali.

Dijkstrov algoritmus sa zdá byť najvhodnejším kandidátom pre účely našej práce. Jeho časová zložitosť je príjemná, implementácia nenánorčná a je dosť ľahké sa v nej orientovať, takže si ju budeme môcť poľahky modifikovať, prispôsobiť našim potrebám. Budeme teda schopný vypísať dodatočné informácie pre používateľov aplikácie, prípadne vyhľadávanie spresniť či rozšíriť podľa jeho potrieb.

2.1.2 Bellman - Fordov algoritmus

2.1.3 Floyd - Warshallov algoritmus

2.1.4 Johnsonov algoritmus

2.1.5 Johnsonov algoritmus

Kapitola 3

Softvér

Táto kapitola obsahuje základné informácie o nami implementovanom softvére. Popisujeme v nej, ako program funguje, na čo slúži a ako sa ovláda. Uvádzame rôzne funkcie softvéru, prípadne aj príklady ich použitia.

Kapitola 4

Implementácia softvéru

V tejto kapitole uvedieme, ako sme postupovali pri implementácií softvéru. Odhalíme, aké postupy, štruktúry či algoritmy sme zvolili, pričom popíšeme aj dôvody nášho výberu. Neuvádzame celý kód, ale len tie časti, ktoré sa nám zdali najviac zaujímavé a zodpovedajúce obsahu našej práce.

Záver

Na záver už len odporúčania k samotnej kapitole Záver v bakalárskej práci podľa smernice [**?**]: „V závere je potrebné v stručnosti zhrnúť dosiahnuté výsledky vo vzťahu k stanoveným cieľom. Rozsah záveru je minimálne dve strany. Záver ako kapitola sa nečísluje.“

Všimnite si správne písanie slovenských úvodzoviek okolo predchádzajúceho citátu, ktoré sme dosiahli príkazmi `\glqq` a `\grqq`.

V informatických prácach niekedy býva záver kratší ako dve strany, ale stále by to mal byť rozumne dlhý text, v rozsahu aspoň jednej strany. Okrem dosiahnutých cieľov sa zvyknú rozoberať aj otvorené problémy a námety na ďalšiu prácu v oblasti.

Abstrakt, úvod a záver práce obsahujú podobné informácie. Abstrakt je kratší text, ktorý má pomôcť čitateľovi sa rozhodnúť, či vôbec prácu chce čítať. Úvod má umožniť zorientovať sa v práci skôr než ju začne čítať a záver sumarizuje najdôležitejšie veci po tom, ako prácu prečítal, môže sa teda viac zamerať na detaily a využívať pojmy zavedené v práci.

Literatúra

[1] Reinhard Diestel. Teória grafov, 2000. [Citované 2017-12-4] Dostupné z `http://www.dcs.fmph.uniba.sk/~haviarova/uktg/#materialy`.

[2] Shimon Even. *Graph algorithms*. Cambridge University Press, 2011.

[3] Jana Katreniaková. Vizualizácia a kreslenie pekných grafov, 2014. [Citované 2017-12-4] Dostupné z `http://sccg.sk/~ferko/VizualizaciaPreGrafikov.pdf`.

[4] S Meena Kumari and N Geethanjali. A survey on shortest path routing algorithms for public transport travel. *Global Journal of Computer Science and Technology*, 9(5):73–75, 2010.

[5] Jakub Novák. Dynamická navigácia osôb so spätnou väzbou v mestskej hromadnej doprave. Bakalárska práca, Univerzita Komenského v Bratislave, 2016.

[6] José Luis Santos. k-shortest path algorithms. 2007. [Citované 2017-12-4] Dostupné z `https://estudogeral.sib.uc.pt/jspui/bitstream/10316/11305/1/k-Shortest%20path%20algorithms.pdf`.

[7] Pavol Ďuriš. *Tvorba efektívnych algoritmov*. Knižničné a edičné centrum FMFI UK, 2009.