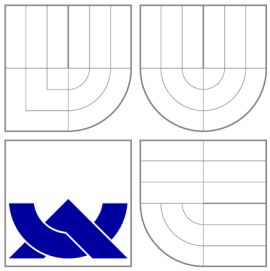
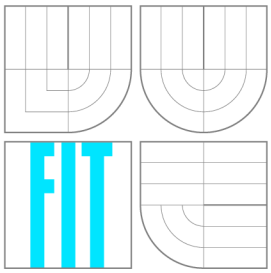


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

VYHLEDÁVÁNÍ CESTY VE VEKTOROVÝCH DATECH PROJEKTU OPENSTREETMAP

PATH FINDING IN THE VECTOR DATA IN OPENSTREETMAP PROJECT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ADAM ADAMČEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN VÁŇA
SUPERVISOR

BRNO 2012

Abstrakt

Vyhledávače cest patří mezi služby, které v dnešní době lidé často využívají při plánování přejezdu neznámým územím. Tato práce se zabývá návrhem a implementací aplikace, která bude schopna ve volně dostupných mapových datech projektu OpenStreetMap vyhledávat cesty a zobrazovat je. Při vyhledávání zohledňuje nejen typ cest, ale i možnost výběru dopravního prostředku a vlastnosti výsledné trasy. Součástí práce je i analýza formátu OSM a návrh vlastního kompaktnějšího formátu pro uložení předzpracovaných dat. V závěru je kromě otestování výkonu implementované aplikace také porovnání jejích výstupů s jinými běžně používanými vyhledávači cest.

Abstract

Pathfinding services are nowadays often used by people to plan a route through an unknown area. This bachelor's thesis deals with designing and implementation of an application which is able to search in free map data provided by the OpenStreetMap project and visualize resulting paths. During the search, not only the way types, but the selected transport type and the attributes of the eventual route will be taken into account as well. The work also contains an analysis of the OSM format and the proposal of a custom, more compact format useful for storing the preprocessed data. Eventually, the implemented application's performance is tested and its output is compared with other commonly used routing solutions.

Klíčová slova

OSM, OpenStreetMap, vyhledávání cest, vektorová data, mapa, dopravní prostředek, trasa

Keywords

OSM, OpenStreetMap, pathfinding, vector data, map, transport type, route

Citace

Adam Adamček: Vyhledávání cesty ve vektorových datech projektu OpenStreetMap, bakalářská práce, Brno, FIT VUT v Brně, 2012

Vyhledávání cesty ve vektorových datech projektu OpenStreetMap

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jana Váni.

.....
Adam Adamček
16. května 2012

Poděkování

Na tomto místě bych chtěl poděkovat svému vedoucímu Ing. Janu Váňovi za jeho čas, cenné připomínky a odbornou pomoc na konzultacích k této bakalářské práci. Dále bych rád poděkoval svým rodičům, bratrovi a přátelům za podporu při studiu.

Obsah

Úvod

Vďaka iniciatíve projektu OpenStreetMap zmapovať našu planétu a voľne poskytovať zaznamenané dáta vzniká v posledných rokoch veľké množstvo užitočných projektov uľahčujúcich orientáciu v tomto svete. Medzi ne patria aj vyhľadávače ciest, ktoré ľudia často využívajú na vyhľadanie najvhodnejšej trasy podľa určitých parametrov pri plánovaní prejazdu neznámym územím. Počas jej zostavovania tieto vyhľadávače zohľadňujú typ použitého dopravného prostriedku, dĺžku trasy alebo odhadovaný čas na jej prejdenie. Pri návrhu a vývoji takejto aplikácie je dôležitá správna analýza zdrojových mapových dát, aby sa zistilo, aké možnosti tieto dáta ponúkajú a vyhľadané cesty boli dostatočne relevantné. A keďže žijeme vo veľmi uponáhľanej dobe a používatelia sú často netrpezliví, dôležitými faktormi sú tiež rýchlosť a intuitívnosť aplikácie.

Vektorové dáta obsiahnuté v mapových súboroch je vhodné pretransformovať do inej formy reprezentácie, v ktorej bude možné použiť algoritmy na vyhľadávanie cesty. Takouto formou je graf zložený z uzlov a ohodnotených orientovaných hrán. Preto je úvod kapitoly 1 venovaný oboznámeniu sa s potrebnými pojmami z teórie grafov. Ďalej bude prebraný aktuálny stav v tejto oblasti a popis klasických algoritmov, ktoré sú na tieto účely bežne využívané, ale aj pokročilých algoritmov, ktoré vznikli až v posledných rokoch.

V kapitole 2 bude predstavený projekt OpenStreetMap, jeho ciele a štruktúra formátu poskytovaných dát. Súčasťou tejto kapitoly je tiež analýza, ktoré entity obsiahnuté v tomto formáte sú využiteľné pre vyhľadávanie ciest podľa bližšie špecifikovaných parametrov. V závere kapitoly uvádzam efektívny a presný spôsob, akým je možné vypočítať vzdialenosť medzi dvoma miestami na Zemi.

Kapitola 3 sa venuje návrhu aplikácie, ktorá využije vektorové dáta projektu OpenStreetMap na vyhľadanie a zvizualizovanie trasy medzi dvoma zvolenými bodmi na mape. Pri zostavovaní trasy je potrebné zohľadniť typy ciest a vybraný dopravný prostriedok tak, aby bol nájdený výsledok vzhľadom na tieto a ďalšie parametre čo najvhodnejší. Okrem toho tu bude popísaný formát dát, ktorý som navrhol s cieľom znížiť veľkosť uložených mapových dát potrebných pre vyhľadávanie a zrýchliť ich načítanie do pamäte aplikácie.

Na obsah predchádzajúcej kapitoly bude nadväzovať kapitola 4 popisujúca implementáciu navrhnutej aplikácie, ovládanie a problémy, s ktorými som sa v priebehu vývoja stretol a bolo ich potrebné riešiť.

Otestovanie funkčnosti implementovanej aplikácie je zdokumentované v kapitole 5. Testy sa zameriavajú najmä na rýchlosť vyhodnocovania používateľských požiadaviek. Obsahom tejto časti je porovnanie výkonov jednotlivých podporovaných algoritmov, parametrov vyhľadávania, ktoré je možné nastaviť, a zdôvodnenie ich vplyvu na dĺžku vyhľadávania. V druhej časti budú porovnané výstupy implementovanej aplikácie s trasami, ktoré pre rôzne parametre vyhľadávania našli konkurenčné bežne používané riešenia.

Kapitola 6 pojednáva o možných rozšíreniach aplikácie. Okrem návrhov na ďalšie zvýšenie rýchlosti a zníženie pamäťovej náročnosti tu budú diskutované rôzne vylepšenia, ktoré zatiaľ v našej aplikácii implementované neboli, avšak konkurenčné riešenia ich poskytujú.

Záverečná kapitola sumarizuje výsledky tejto práce.

Kapitola 1

Vyhľadávanie cesty

V tejto kapitole sa zoznámime s potrebnou teóriou a pojmami, ktoré budú v tejto práci ďalej využívané. V prvej časti kapitoly si zadefinujeme pojmy z oblasti teórie grafov a potom budú popísané algoritmy, ktoré sa využívajú pri vyhľadávaní ciest v grafoch. Zoznámime sa nielen s klasickými algoritmami, ktoré sa na túto úlohu bežne využívajú, ale budú predstavené aj pokročilé algoritmy, ktoré vznikli v priebehu posledných rokov výskumu v tejto oblasti. V poslednej časti tejto kapitoly si uvedieme krátky prehľad existujúcich riešení na vyhľadávanie trasy na mape.

1.1 Definície pojmov

Keďže užitočné dáta projektu OpenStreetMap sú reprezentované komponentami grafu a algoritmy v tejto práci tiež využívajú pojmy z teórie grafov, v nasledujúcej sekcii budú tieto pojmy zadefinované podľa [1].

1.1.1 Grafy

Jednoduchý neorientovaný **graf** $G = (V, E)$ je usporiadaná dvojica, kde V je množina vrcholov (uzlov) a E je množina hrán – množina vybraných dvojprvkových podmnožín pôvodnej množiny vrcholov. Hranu, ktorá spája vrcholy u a v, značíme $\{u, v\}$ a tieto vrcholy nazývame **susedné**. Ak medzi dvoma vrcholmi existuje viacero hrán, potom tieto hrany nazývame **násobné**.

Nech G je ľubovoľný graf. Potom definujeme nasledujúce pojmy:

- **Kružnica dĺžky n** spája n hranami $n \geq 3$ vrcholov do jedného cyklu.
- **Úplný graf** má všetky vrcholy navzájom prepojené hranami.
- **Stupeň vrcholu** v určuje počet hrán, ktoré vedú z alebo do tohoto vrcholu.
- **Podgraf grafu** G je ľubovoľný graf H na podmnožine vrcholov grafu G a jeho hrany sú podmnožinou grafu G, pričom oba vrcholy týchto hrán sa nachádzajú vo $V(H)$.
- **Sled dĺžky** n je striedavá postupnosť vrcholov a hrán medzi týmito vrcholmi v tvare $v_0, e_1, v_1, ..., e_n, v_n$.
- **Cesta dĺžky n** má $n + 1$ vrcholov spojených za sebou n hranami, pričom vrcholy sú rôzne.

- **Ťah** je sled v grafe bez opakovania hrán.
- **Súvislý graf** umožňuje vytvorenie cesty medzi ľubovoľnými dvoma vrcholmi tohoto grafu.
- **Vzdialenosť** je daná dĺžkou najkratšieho sledu medzi dvoma vrcholmi, pričom, ak sled neexistuje, vzdialenosť je ∞.
- **Ohodnotený (vážený) graf** má hrany alebo vrcholy ohodnotené číslom.
- **Orientovaný graf** má každú hranu e určenú usporiadanou dvojicou vrcholov $\{u, v\}$.

1.2 Algoritmy na vyhľadávanie ciest v grafoch

Spolu s rozvojom prenosnej elektroniky sa zvyšujú požiadavky spoločnosti na funkčnosť týchto technológií. Dostupnosť máp umožnila používateľom využívanie geograficky založených služieb, od ktorých očakávajú rýchlu rýchlu a relevantnú odozvu. V tejto časti kapitoly si uvedieme klasické algoritmy využívané na vyhľadávanie ciest v grafoch, ale budú predstavené aj niektoré novovzniknuté algoritmy zamerané na trasovanie v rozsiahlych grafoch, ktoré sú výsledkom posledných rokov výskumu v tejto oblasti.

1.2.1 Základné algoritmy

Nasledujúce algoritmy publikované prevažne v druhej polovici minulého storočia sú považované za základ a štandard v oblasti vyhľadávania ciest v grafoch. Vlastnosti týchto algoritmov môžu byť:

- **Úplnosť** – ak riešenie existuje, algoritmus ho nájde.
- **Konečnosť** – takýto algoritmus pri konečnom počte vstupných prvkov vždy skončí.
- **Optimálnosť** – výstupom tohoto algoritmu je najlepšie existujúce riešenie.

Prehľadávanie do hĺbky a do šírky. Spoločným znakom ďalších algoritmov je prehľadávanie grafu a ukladanie vrcholov, ktoré sa majú navštíviť, do dátovej štruktúry typu zásobník alebo fronta. Pri použití zásobníka nastane prehľadávanie grafu do hĺbky, v druhom prípade pôjde o prehľadávanie do šírky [1].

Bellman-Fordov algoritmus

Tento algoritmus vypočíta najkratšiu cestu medzi dvoma vrcholmi v ohodnotenom grafe, pričom tieto hrany môžu byť aj záporne ohodnotené. Keďže nižšie uvedené algoritmy sú pre nezáporné hrany efektívnejšie, práve tento nachádza svoje uplatnenie najmä v smerovacích protokoloch počítačových sietí a v tejto práci je uvedený iba informatívne. Časová náročnosť tohoto algoritmu je $O(V \cdot R)$ [2].

Dijkstrov algoritmus

V roku 1956 vytvoril a v roku 1959 publikoval holandský informatik Edgester W. Dijkstra algoritmus slúžiaci na vyhľadávanie najkratšej cesty z jedného počiatočného vrcholu v ľubovoľnom nezáporne hranovo ohodnotenom orientovanom grafe. Podstatou je prechádzanie nenavštívených vrcholov a relaxácia hrán vychádzajúcich z týchto vrcholov.

Pôvodný algoritmus nevyužíval prioritnú frontu a jeho časová náročnosť bola $O(V^2)$. Neskoršia implementácia využíva prioritnú frontu pomocou Fibonacciho haldy a s časovou náročnosťou $O(E + V \cdot logV)$ je považovaná za najrýchlejší algoritmus svojho druhu.

Dijkstrov algoritmus je *konečný*,*úplný* a *optimálny* a jeho priebeh je nasledovný:

1. Inicializuj ohodnotenie všetkých vrcholov v množine V na maximálnu hodnotu ∞ a počiatočný vrchol na hodnotu 0.
2. Vytvor množinu nenavštívených vrcholov $N = V$.
3. Kým množina N nie je prázdna opakuj:
   a Vyber vrchol v s najlepším ohodnotením z množiny N.
   b Ak bol vybraný cieľový vrchol, ukonči algoritmus.
   c Každý susedský uzol u vrcholu v, do ktorého vedie hrana z vrcholu v, ohodnoť funkciou $f(u)$ a tento výsledok ulož do uzla, ak je menší ako aktuálne uložené ohodnotenie – ide o **relaxáciu hrany**. Funkcia $f(u) = f(v) + \{v, u\}$ hodnotí podľa najkratšej prejdenej vzdialenosti od počiatočného vrcholu k u a využíva pri tom aktuálne ohodnotenie vrcholu v a ohodnotenie hrany $\{u, v\}$.

Po skončení predstavuje ohodnotenie cieľového uzla dĺžku cesty medzi začiatočným a cieľovým vrcholom. Ak je hodnota cieľového vrcholu rovná ∞, je zrejmé, že medzi týmito uzlami žiadna cesta neexistuje. Získanie výslednej postupnosti uzlov najkratšej cesty je možné tak, že si každý uzol u bude okrem ohodnotenia pamätať informáciu o vrchole v, od ktorého toto ohodnotenie pochádza. Počnúc cieľovým uzlom je potom možné rekurzívne zrekonštruovať cestu až k počiatočnému uzlu [2].

Algoritmus A*

Najznámejšou variantou Dijkstrovho algoritmu je A*, ktorý ohodnocuje uzol u funkciou $f(u) = g(u) + h(u)$, kde:

- $g(u)$ je funkcia určujúca prejdenú vzdialenosť od počiatočného do daného uzla u
- $h(u)$ je **heuristická funkcia** odhadujúca zvyšnú vzdialenosť od uzla u do cieľa

Tento algoritmus je pre vhodne určené heuristické funkcie nielen *konečný* a *úplný*, ale aj *optimálny* [3].

Algoritmus IDA*

Algoritmus IDA* (Iterative Deepening A*) je varianta predchádzajúceho algoritmu A* kladúca si za cieľ zníženie pamäťovej náročnosti potrebnej na vyhľadanie výslednej cesty. V skutočnosti ide o upravený algoritmus postupného zanorovania sa do hĺbky, ktorého ohodnocovacia funkcia využíva rovnakú metódu ohodnocovania ako aloritmus A* [4].

Algoritmus Shooting*

K tomuto algoritmu využívanom v projekte pgRouting [5] neexistuje žiadna štúdia, avšak ide o variantu predchádzajúcich dvoch algoritmov. Na rozdiel od nich nie je založený vrcholovo ale hranovo, čo umožňuje riešiť problémy spojené s modelovaním dopravných obmedzení na cestách, ako napríklad prikázaný smer jazdy na križovatkách. Princípom je transformácia pôvodného grafu G na *hranový graf*, v ktorom sú hrany pôvodného grafu G reprezentované ako vrcholy a novovzniknuté hrany medzi týmito vrcholmi predstavujú susednosť týchto hrán. Na vyhľadanie najkratšej cesty je potom možné použiť vyššie popísaný Dijkstrov algoritmus alebo A* [6].

1.2.2 Pokročilé algoritmy na vyhľadávanie cesty v rozsiahlych grafoch

Spolu s rozrastaním dopravnej infraštruktúry rastie aj množstvo geografických dát zachytávajúcich tieto dopravné trasy, avšak používatelia stále očakávajú rýchlejšie spracovávanie svojich požiadaviek. Výskumu v tejto oblasti sa v predchádzajúcej dekáde venovalo niekoľko tímov a boli publikované viaceré algoritmy, ktoré si kladú za cieľ urýchliť vyhľadávanie cesty v rozsiahlych grafoch.

Transit-Node Routing

Algoritmus vyberie dôležité prístupové body (tranzitné uzly) na rôznych úrovniach, medzi ktorými predpočíta tabuľky vzdialeností. Taktiež predpočíta vstupné spojenia do týchto bodov z tranzitných uzlov na nižších úrovniach a vyhľadané cesty vedie cez tieto tranzitné uzly. Výhodou je nízky počet dotazov do databázy, a teda aj vyhodnotenie rozsiahlych trás na úrovni mikrosekúnd, nevýhodou mimoriadne pomalé predspracovanie a veľký objem predspracovaných informácií [7].

Contraction Hierarchies

Táto metóda je založená na princípe kontrakcií vrcholov – konečný počet vrcholov v danej oblasti je zoradený podľa dôležitosti a postupne po jednom sú tieto vrcholy nahradzované skrátenými cestami medzi ostatnými vrcholmi, až kým nevznikne jediný vrchol. Pri použití Dijkstrovho algoritmu je vyhodnotenie $5\times$ rýchlejšie a na uloženie predpočítaných informácií je potrebné menšie množstvo pamäte ako na uloženie pôvodných dát grafu. Okrem rýchleho predspracovania jej jednoduchosť a prispôsobiteľnosť ju predurčuje na použitie v kombinácii s ďalšími algoritmami a uplatnenie nájde nielen v mobilných technológiách [8].

Arc-Flags

Arc-Flags je modifikácia klasického Dijkstrovho algoritmu, ktorá má zamedziť preskúmavaniu nepotrebných ciest pre nájdenie najkratšej cesty počas prehľadávania grafu. Graf je rozdelený na menšie oblasti a pre každú kombináciu hrany a oblasti sa predpočíta informácia, či cez túto hranu vedie najkratšia cesta do nejakého vrcholu ležiaceho v tejto oblasti. Výhodou tohoto prístupu je znateľné urýchlenie vyhľadávania, nevýhodou zostáva pomalé predspracovanie [9].

1.3 Existujúce aplikačné riešenia

Od založenia projektu OpenStreetMap vzniklo značné množstvo aplikácií využívajúcich ponúkané voľne dostupné geografické údaje na vyhľadávanie trasy. Podľa toho, či pre svoju funkčnosť vyžadujú pripojenie k internetu sa rozdeľujú na dve základné skupiny – online a offline riešenia.

Sumarizáciu vlastností vybraných aplikácií zobrazuje nasledujúca tabuľka:

Tabuľka 1.1: Porovnanie vlastností voľne dostupných aplikácií využívajúcich OSM dáta na vyhľadanie trasy.

Názov	**Typ**	**Jazyk**	**Algoritmus**	**Funkcie**
MoNav	offline	C++/ Qt	Contraction Hierarchies	veľmi rýchle a presné vyhľadávanie vo veľkých oblastiach, nízke pamäťové nároky, portabilita
Navit	offline	C++	Dijkstra s Fibonacciho haldami	podpora hlasovej navigácie, 3D vizualizácia mapy
pgRouting	online	PostGIS/ pgSQL	Dijkstra/A*/ Shooting*	využíva databázu na uloženie dát, rýchly výpočet ohodnotení možný priamo za behu
OpenRoute-Service	online	Java	A*	veľké množstvo podporovaných typov dopravných prostriedkov a iných parametrov vyhľadávania
We-Travel	offline	Java	Dijkstra/A*	hlasová navigácia, funkčnosť na zariadeniach podporujúcich technológiu Java Mobile

Ako je možné vidieť v tabuľke 1.1, v rámci použitých algoritmov prevláda vo väčšine aplikácií varianta Dijkstrovho algoritmu a A*. Aplikácia MoNav podporuje nový algoritmus Contraction Hierarchies, vďaka ktorému trvá vyhľadávanie trasy aj vo veľkých krajinách na úrovni milisekúnd. Riešenie pgRouting sa zo svojej podstaty zameriava na prístup k vyhľadávaniu s využitím dostupných možností objektovo-relačného databázového systému PostgreSQL. Ostatné riešenia ponúkajú ďalšie navzájom rôzne vymoženosti, ktoré sa v iných aplikáciách v podobnej forme nenachádzajú, ako napríklad hlasovú navigáciu alebo trojrozmernú vizualizáciu mapy [10].

Kapitola 2

Projekt OpenStreetMap a vektorový formát OSM

V tejto kapitole sa stručne zoznámime s projektom OpenStreetMap a dátami, ktoré poskytuje. Ďalej budú uvedené formáty, ktoré využíva komunita projektu na uloženie vektorových dát a následne bude bližšie predstavený formát OSM založený na značkovacom jazyku XML. Zhrnieme si výhody aj nevýhody, ktoré používanie tohoto formátu prináša, a popíšeme jeho štruktúru. Pretože súbory s mapami obsahujú príliš veľké množstvo nadbytočných údajov, budeme ďalej analyzovať, ktoré entity sú využiteľné pre našu prácu. Na záver tejto kapitoly bude predstavený geodetický štandard, podľa ktorého je v mapových dátach zaznačovaná geografická poloha jednotlivých bodov a spôsob, akým je možné vypočítať vzdialenosť medzi nimi. Informácie uvedené v tejto kapitole sú preberané z oficiálnej dokumentácie k projektu OpenStreetMap nachádzajúcej sa na internetovej stránke [10].

2.1 Projekt OpenStreetMap

Projekt OpenStreetMap (OSM) [11] bol založený v roku 2004 s cieľom poskytnúť voľne dostupné geografické dáta a ich zvizualizovanú topografickú formu. Na vytváraní databázy sa podieľajú najmä dobrovoľní členovia komunity využívajúci GPS zariadenia a špecializované editory na spracovanie získaných údajov. Medzi ďalšie zdroje geodát alebo grafických podkladov patria aj dáta tretích strán, ak to umožňuje ich licencia. Dáta sú poskytované pod licenciou Creative Commons Attribution-ShareAlike 2.0 [12].

2.2 Vektorové dáta projektu OpenStreetMap

Ako už bolo spomínané, dáta, ktoré vytvára komunita projektu, sú prevažne vektorového charakteru. To znamená, že informácie sú zachytené ako súbor základných geometrických primitív – bodov, čiar a polygónov. Tie reprezentujú jednotlivé prvky mapy ako napríklad budovy, cesty alebo miesta záujmu. Pre každý z nich je evidovaných viacero údajov, ako napríklad používateľ, ktorý ich vytvoril alebo zmenil, dátum vytvorenia (zmeny) a podobne.

Na uloženie týchto dát využíva väčšina nástrojov komunity OpenStreetMap formát OSM popísaný pomocou značkovacieho jazyka XML. Z povahy tohoto formátu plynú nasledujúce výhody a nevýhody:

Výhody:

- jednoducho čitateľné z dôvodu prehľadnej štruktúry
- systémovo nezávislé kvôli presným definíciám formátu
- ľahko použiteľné s existujúcimi XML parsermi
- dobrý kompresný pomer

Nevýhody:

- veľmi veľké súbory
- potrebná dekompresia pred použitím (týka sa komprimovaných *.osm súborov)
- parsovanie je časovo náročné

Odstrániť nevýhody formátu OSM XML má formát PBF, ktorého veľkosť je asi polovičná a čítanie informácií je približne 6× a zapisovanie 5× rýchlejšie. Nevýhodou tohoto binárneho formátu môže byť neprehľadnosť a nízka podpora, keďže ide o relatívne nový formát.

Medzi ďalšie komunitné formáty patria JOSM, osmChange, Osmdiff a planetdiff, avšak ich štruktúra a obsah je optimalizovaná pre špecifické využitia.

2.2.1 Všeobecný formát

Keďže všeobecný formát projektu je popísaný jazykom XML, dokument sa skladá z koreňového `<osm>` elementu s atribútmi popisujúcimi generátor dokumentu a verziu API, s ktorou je obsah dokumentu kompatibilný.

V koreňovom elemente sa nachádza špecifikácia regiónu `<bounds>`, ktorý dokument popisuje, a zvyšný obsah je tvorený zoznamom dátových primitív – uzlami `<node>`, cestami `<way>` a vzájomnými vzťahmi `<relation>`. Každé z týchto primitív je jednoznačne označené identifikačným číslom a obsahuje informácie o autorovi, čase vytvorenia a verzii zmeny.

Element uzol je bod obsahujúci atribúty, ktoré určujú geografickú šírku (angl. latitude) a dĺžku (angl. longitude) podľa *Svetového geodetického systému 1984* [13].

Cesta je párový element reprezentujúci čiary alebo uzatvorené oblasti. Obsahuje elementy `<nd>` určujúce uzly, ktoré do tejto cesty patria, a `<tag>` špecifikujúci druh cesty ako napríklad diaľnica alebo budova.

Vzťah je rovnako ako cesta párový element a jeho úlohou je popísať logické alebo geografické vzťahy medzi predchádzajúcimi elementami. Taktiež obsahuje jeden alebo viac elementov `<tag>` popisujúcich typ vzťahu a minimálne jedno z prechádzajúcich primitív určené elementom `<member>`. Príkladom použitia je zoskupenie viacerých oblastí do koryta rieky alebo určenie dopravných obmedzení vzťahujúcich sa na cesty.

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.0.2">
 <bounds minlat="54.0889580" minlon="12.2487570" maxlat="54.0913900" maxlon="
     12.2524800"/>
 <node id="298884269" lat="54.0901746" lon="12.2482632" user="SvenHRO" uid="
     46882" visible="true" version="1" changeset="676636" timestamp="
     2008-09-21T21:37:45Z"/>
 ...
 <way id="26659127" user="Masch" uid="55988" visible="true" version="5"
     changeset="4142606" timestamp="2010-03-16T11:47:08Z">
  <nd ref="292403538"/>
  ...
```

```
  <tag k="highway" v="unclassified"/>
  <tag k="name" v="Pastower Straße"/>
 </way>
 <relation id="56688" user="kmvar" uid="56190" visible="true" version="28"
     changeset="6947637" timestamp="2011-01-12T14:23:49Z">
  <member type="node" ref="364933006" role=""/>
  <member type="way" ref="4579143" role=""/>
  ...
  <tag k="name" v="Küstenbus Linie 123"/>
  ...
 </relation>
 ...
</osm>
```

Kód 2.1: Ukážka obsahu OSM súboru.

2.2.2 Využiteľné entity

Ako bolo uvedené v sekcii 2.2.1, jednotlivé elementy typu cesta môžu reprezentovať čiary alebo uzatvorené oblasti. Význam tohoto prvku je určený značkou špecifikovanou v atribúte elementu `<tag>`. Pre účely tejto práce sú predmetom záujmu značky vymedzujúce dopravné trasy určené na pozemnú komunikáciu, teda tie, ktoré nesú označenie `highway`. Hodnota tejto značky bližšie špecifikuje, o aký druh trasy sa jedná a pre aký dopravný prostriedok je určená. Medzi štandardne používané patria nasledujúce hodnoty označení:

Motorové trasy:

- diaľnica (`motorway/motorway_link`)
- cesta pre motorové vozidlá (`trunk/trunk_link`)
- cesta I. triedy (`primary/primary_link`)
- cesta II. triedy (`secondary/secondary_link`)
- cesta III. triedy (`tertiary/tertiary_link`)
- cesta nižšej kategórie (`unclassified`)
- cesta v obytnej oblasti (`residential`)
- cesta pre prístup k pozemkom a nehnuteľnostiam (`service`)
- cesta pre agrikulturálne využitie (`track`)

Nemotorové trasy:

- cyklotrasa (`cycleway`)
- cesta pre nemotorové vozidlá (`path`)
- schody pre chodcov (`steps`)
- pešia zóna (`pedestrian`)
- trasa pre chodcov (`footway`)
- trasa pre chodcov, cyklistov a kone (`bridleway`)

2.2.3 Obmedzenia

Trasy popísané v predchádzajúcej sekcii môžu niesť rôzne obmedzenia bližšie špecifikujúce vlastnosti danej cesty, ktoré je potrebné zohľadniť pri plánovaní. Medzi významnejšie z nich patria:

- obmedzenie maximálnej rýchlosti cesty pre dopravné prostriedky (`maxspeed`)
- explicitné určenie, či je cestu možné využiť na cyklistické účely (`bicycleroute`)
- explicitné určenie, či je cesta vhodná pre chodcov (`footroute`)
- určenie, či sa pri ceste nachádza chodník pre chodcov (`sidewalk, footway`)
- obmedzenie pohybu pre dopravné prostriedky v jednom jazdnom smere (`oneway`)

2.3 Geodetický štandard WGS84

WGS84 je označenie pre geodetický štandard s názvom **Svetový geodetický systém 1984** (World Geodetic System 1984), ktorý bol vydaný v roku uvádzanom v jeho názve. Dôvodmi vzniku zjednoteného štandardu boli v päťdesiatych rokoch dvadsiateho storočia najmä počiatky vesmírneho výskumu a astronautiky a nejednotnosť dovtedy existujúcich geodetických štandardov. Štandard popisuje referenčný elipsoid a geoid, ktoré sú v dnešnej dobe využívané v oblastiach kartografie, geodézie a navigácie.

Súradnicový systém tohoto štandardu je reprezentovaný pravotočivým kartézskym súradnicovým systémom s počiatočným bodom tejto sústavy ležiacim v ťažisku Zeme. Parametre definujúce referenčný elipsoid a kompletnú definíciu štandardu je možné nájsť v literatúre [14].

2.3.1 Výpočet vzdialenosti medzi 2 bodmi

Aj keď formát WGS84 a s ním spojené geografické značenie predpokladá podklad v tvare elipsoidu pre ktorý vhodnejšia *Vincentyho metóda* využívajúca ako podklad rotačný elipsoid, z hľadiska výpočtovej zložitosti je príliš náročná. Ako vhodná alternatíva k tejto metóde sa javí použitie *Haversínovho vzorca* [15]. Ten pre svoje výpočty využíva guľový podklad a slúži na výpočet tzv. *ortodrómy* – najkratšej spojnice dvoch bodov na guľovej ploche. Výhodou tejto voľby je, že ponúka oveľa kratší počet krokov na zistenie výsledku za cenu veľmi malej odchýlky oproti skutočnému výsledku.

Výpočet ortodromatickej vzdialenosti medzi 2 bodmi na guli **Haversínovým vzorcom** podľa [15] uvádzajú nasledujúce rovnice:

$$\begin{aligned} \Delta lat &= lat_2 - lat_1 \\ \Delta lon &= lon_2 - lon_1 \\ a &= \sin^2(\frac{\Delta lat}{2}) + \cos(lat_1) \cdot \cos(lat_2) \cdot \sin^2(\frac{\Delta lat}{2}) \\ c &= 2 \cdot \arcsin(\sqrt{a}) \\ d &= R \cdot c \end{aligned} \tag{2.1}$$

kde lat_1 a lon_1 sú súradnice počiatočného bodu, lat_2 a lon_2 sú súradnice cieľového bodu, R je polomer gule, c je ortodróma týchto bodov v radiánoch a d je výsledná ortodromatická vzdialenosť medzi týmito bodmi v rovnakých jednotkách ako polomer gule R.

Kapitola 3

Návrh aplikácie

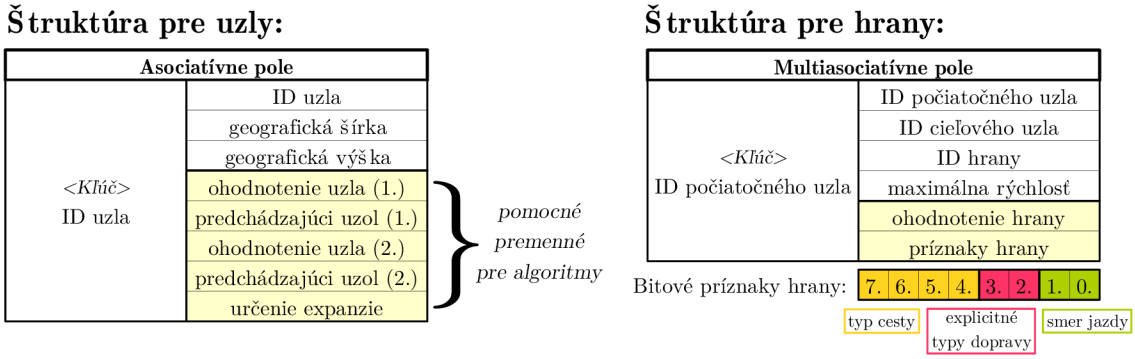
Obsahom tejto kapitoly je návrh aplikácie schopnej načítať užitočné informácie z formátu OSM popísaného v predchádzajúcej kapitole do vhodnej dátovej štruktúry pre reprezentáciu cestného grafu. V týchto dátach by mala byť schopná vyhľadávať cestu podľa zvolených parametrov a nájdenú výslednú trasu graficky zvizualizovať na zobrazenom mapovom podklade. Už raz predspracované dáta by bolo vhodné umožniť používateľovi uložiť v kompaktnom formáte, ktorý podporí ich opätovné rýchlejšie načítanie. Ako vyplynulo z predchádzajúcej kapitoly, cesty v mapových dátach obsahujú dodatočné rozširujúce informácie, ktoré je možné využiť na prispôsobovanie trasy potrebám používateľa. Tie je možné najjednoduchšie reprezentovať výberom vhodnej množiny dopravných prostriedkov, ktoré budú zachytávať dostatočne odlišné pravidlá pre vyhľadávanie. Kedže ľudia majú rôzne preferencie – jedni uprednostnia dĺžku trasy a pre iných je cennejší čas, bude tiež predstavená možnosť, ako zohľadniť aj tieto parametre pri vyhľadávaní výslednej trasy. Na záver kapitoly budeme diskutovať o vhodných textových formátoch pre export výslednej trasy.

3.1 Reprezentácia vektorových dát

Na to, aby vyhľadávací algoritmus mohol vykonávať svoju úlohu, je potrebné navrhnúť dátové štruktúry, ktoré budú po načítaní mapových dát reprezentovať cestný graf s ohodnotenými orientovanými hranami, v ktorom bude vyhľadávanie prebiehať. Taktiež je vhodné podporovať uloženie takto predspracovaných dát, pretože pri veľkých zdrojových OSM súboroch môže táto analýza a predspracovávanie zaberať neúmerne veľa času, pričom jej výsledok je pre daný súbor vždy rovnaký. Návrh vlastného formátu umožní načítavanie cestného grafu nepomerne rýchlejšie.

3.1.1 Dátové štruktúry

Pre uloženie nasledujúcich množín uzlov resp. hrán sú využité asociatívne polia. Ich výhodou je oproti obyčajným poliam možnosť (teoreticky) priameho prístupu k potrebným záznamom pomocou kľúča a nie pevne stanoveného indexu. Keďže záznamom asociatívneho poľa je pár *kľúč:hodnota*, hodnotami budú štruktúry obsahujúce údaje, ktoré je pre ne potrebné evidovať. Z neskorších implementačných dôvodov je vhodné ukladať informáciu obsiahnutú v kľúči aj do štruktúry v hodnote záznamu. Navrhované štruktúry sú zobrazené na obrázku 3.1.



Obr. 3.1: Navrhované štruktúry na reprezentáciu cestného grafu.

Dátová štruktúra uzla

Kľúčom k záznamom asociatívneho poľa reprezentujúceho množinu uzlov je **identifikačné číslo uzla**, ktoré pochádza z korešpondujúceho elementu uzol v OSM súbore. Ďalšie potrebné údaje pre úplné zadefinovanie uzla sú jeho **geografická šírka** a **výška**.

Ďalej je potrebné pre každý uzol evidovať dočasné informácie, ktoré využíva vyhľadávací algoritmus na ukladanie priebežných výsledkov. Prvou z týchto informácií je **ohodnotenie uzla**, do ktorého sa bude ukladať výsledok ohodnocovacej funkcie. Pre zostavenie výslednej cesty je potrebné evidovať aj informáciu o **predchádzajúcom uzle**, z ktorého algoritmus prešiel na aktuálny uzol. Pri použití obojsmernej varianty vyhľadávacích algoritmov je vhodné mať samostatne vyhradenú sadu týchto premenných pre každé z oboch vlákien obojsmerného algoritmu. Premenná **určujúca expanziu** uzla slúži na označenie ešte nespracovaných uzlov a určenie tých, ktoré už algoritmus navštívil a expandoval. Pri obojsmernom vyhľadávaní bude jeho hodnota obsahovať poradové číslo vlákna, ktoré daný uzol expandovalo.

Dátová štruktúra hrany

Keďže hrany grafovej štruktúry reprezentujúcej náš cestný graf sú orientovaného charakteru, závisí na poradí uzlov, ktoré prepájajú. Z hľadiska ďalšieho efektívneho používania v Dijkstrovom a jemu podobných algoritmoch je vhodné zoskupovať hrany podľa **počiatočného uzla**. Jeho identifikačné číslo je využité ako kľúč multiasociatívneho poľa, ktoré bude reprezentovať množinu hrán. Multiasociatívne z toho dôvodu, že môže existovať viacero záznamov s rovnakým kľúčom, pretože z jedného uzla môže vychádzať viacero hrán.

Po identifikačnom čísle **cieľového uzla** nasleduje **identifikačné číslo hrany** pochádzajúce z pôvodného elementu cesta v OSM súbore a všetky hrany, ktoré pochádzajú z rovnakého elementu cesta, majú túto premennú nastavenú na rovnakú hodnotu. Rovnaké identifikačné číslo dvoch rôznych hrán teda zároveň vyjadruje rovnaké ďalej uvedené rozširujúce vlastnosti týchto hrán.

Aj keď každá z typov ciest má určenú predvolenú **maximálnu rýchlosť**, ktorá sa musí na nej dodržiavať, značka `maxspeed` môže túto vlastnosť explicitne upraviť. V prípade, že explicitne určená nie je, nastavenie tejto premennej na hodnotu 0 vyjadruje použitie implicitnej rýchlosti.

V poradí predposledné miesto je v štruktúre hrany vyhradené na uloženie **ohodno-**

tenia hrany. Aj keď je túto informáciu možné vypočítať za behu algoritmu z ostatných evidovaných informácií, opakovaný zložitý výpočet konštantnej hodnoty by bol neefektívny a zvyšoval by výpočtovú náročnosť aplikácie oveľa viac v porovnaní s takto navýšenou pamäťovou náročnosťou.

Pre posledné 3 atribúty bližšie špecifikujúce vlastnosti cesty, ktoré je potrebné uchovávať spolu s cestou, je možné spraviť výčet konečného počtu možností. Z pohľadu efektívneho využívania pamäte by bolo nevhodné uchovávať každú z týchto vlastností v samostatnej premennej, pretože pri naozaj veľkých počtoch hrán, ktoré tvoria cestnú sieť, predstavuje pridanie každej niekoľkobytovej premennej nárast spotrebovanej operačnej pamäte o nezanedbateľné veľkosti. Z týchto dôvodov je vhodné pracovať s nasledujúcimi informáciami na bitovej úrovni a ukladať ich v jedinom 8-bitovom slove **príznakov hrany**.

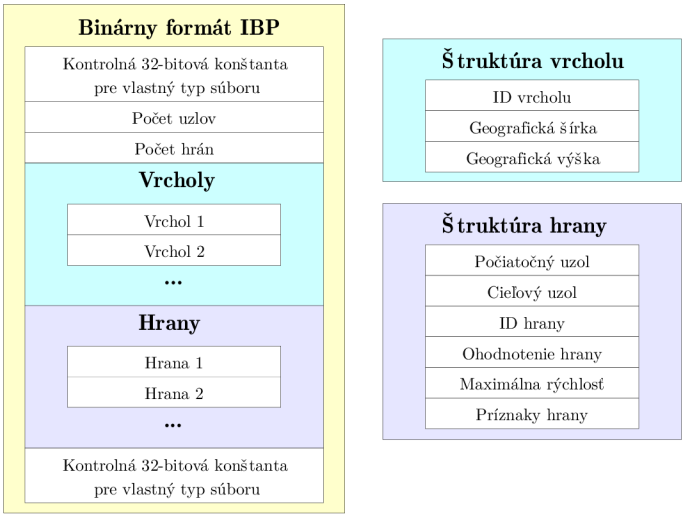
V podsekcii 2.2.2 bolo klasifikovaných 15 základných typov ciest, ktoré sú najčastejšie využívané projektom OpenStreetMap. Toto je ideálny počet pre využitie práve 4 bitov na uloženie tejto informácie, pričom šestnásta možnosť bude vyhradená pre *iný* typ cesty. Teda takej, ktorá nezapadá do ani jednej z predchádzajúcich kategórií. Takto označených ciest je však v týchto mapových dátach zanedbateľné množstvo. Pretože akákoľvek cesta môže mať explicitne povolené alebo zakázané používanie alternatívnych metód dopravy ako bicykel alebo peší pohyb, nasledujúce 2 bity sú vyhradené pre uloženie týchto atribútov. Posledné 2 bity slova príznakov sa dajú využiť na uloženie poslednej dôležitej vlastnosti cesty – smeru jazdy. Môžu určovať povolenie obojsmerného pohybu alebo pohybu iba v jednom smere jazdy, a to buď v rovnakom smere ako smeruje orientovaná hrana medzi uzlami alebo v smere opačnom.

Aby bolo možné obojsmerné vyhľadávanie popísané v podsekcii 3.2.2, každá hrana bude do multiasociatívneho pola vložená dvakrát – raz s kľučom určeným počiatočným uzlom, druhý raz bude jej kľúčom cieľový uzol. Ak hrana reprezentuje jednosmernú cestu, pri druhom vkladaní do pola je potrebné zmeniť príznak povoleného smeru jazdy na opačný.

3.1.2 Formát predspracovaných dát

Ako už bolo spomínané, dáta v štandardnom formáte využívajú pre svoje uloženie súbory, ktorých obsah je reprezentovaný XML jazykom. Okrem toho, že spracovávanie takéhoto formátu je výpočtovo náročné, ide tiež o neefektívnu reprezentáciu údajov z hľadiska pamäťovej náročnosti. K motivácii o vytvorenie vlastného formátu prispel aj fakt, že spomedzi všetkých údajov, ktoré sú v pôvodnom OSM súbore obsiahnuté, je potrebných iba zopár špecifických na zostavenie výslednej grafovej štruktúry pre vyhľadávanie trás. Ďalej, niektoré informácie v pôvodnom súbore nie sú uložené vôbec a aj keď ich je možné odvodiť z existujúcich údajov, je to z praktických dôvodov mimoriadne nevhodné. Pre tieto príčiny vznikol návrh binárneho formátu **IBP**, ktorého cieľom je odstrániť vyššie spomínané nevýhody. Tento formát ukladá v pamäťovo efektívnej forme iba tie údaje, ktoré aplikácia pri práci s týmito súbormi naozaj využije. Jeho ďalšou prednosťou je fakt, že sekvenčne uložené predspracované informácie je možné bez komplikovaného parsovania priamo kopírovať do operačnej pamäte, čo predstavuje význačné zrýchlenie načítavania oproti pôvodnému OSM formátu.

Na začiatku súboru vo formáte IBP sa nachádza **32-bitová kontrolná konštanta**, ktorá býva nazývaná ako tzv. *magic number*. Jej hodnota označuje náš vlastný typ súboru a prípadne aj jeho verziu, vďaka ktorej by bolo možné v budúcnosti v prípade potreby určiť, aké dáta sú v súbore obsiahnuté. Po nej sa nachádzajú 2 číselné informácie udávajúce **počet uzlov** a **hrán** grafovej štruktúry, ktoré sú nasledované v rovnakom poradí sekvenčne



Obr. 3.2: Štruktúra binárneho formátu IBP ukladajúceho predspracované dáta.

uloženými štruktúrami týchto elementov. Pre **uzol** je to jeho identifikačné číslo, geografická šírka a výška. **Štruktúra hrany** obsahuje identifikačné čísla uzlov, medzi ktorými sa nachádza, v poradí počiatočný a cieľový uzol, ďalej identifikačné číslo hrany, jej ohodnotenie, explicitne obmedzujúcu maximálnu rýchlosť a príznaky hrany. Všetky z týchto informácií sú bližšie popísané v podsekcii 3.1.1. Na záver sa preventívne nachádza opäť 32-bitová kontrolná konštanta identická s tou na úplnom začiatku súboru. Keďže jej očakávaná hodnota je známa, slúži na overenie, či išlo naozaj o nami predpokladaný formát a jej úspešné načítanie a overenie zvyšuje pravdepodobnosť, že predchádzajúce dáta boli tiež korektne uložené a načítané.

3.2 Vyhľadávanie v dátach

Cestná sieť, ktorá je už načítaná v dátových štruktúrach predstavených v podsekcii 3.1.1, je pripravená na prehľadávanie algoritmami, ktoré budú uvedené v nasledujúcej časti kapitoly. Rozširujúce informácie hrany uložené v slove príznakov je možné využiť na prispôsobenie priebehu algoritmu tak, aby výsledná cesta reprezentovala trasu využiteľnú za špecifických podmienok.

3.2.1 Použité algoritmy

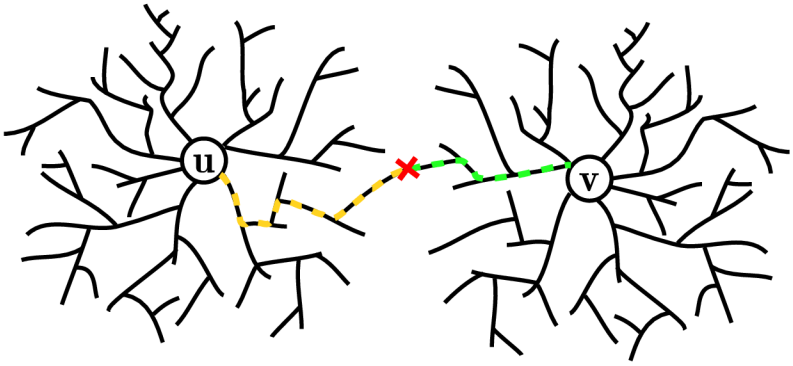
Za hlavný algoritmus využívaný v práci je zvolený Dijkstrov algoritmus, keďže je základom pre väčšinu ďalších pokročilých vyhľadávacích algoritmov, o ktoré je možné prácu v budúcnosti rozšíriť. Jeho ohodnocovaciu funkciu je možné ďalej upraviť pridaním heuristiky tak, aby spĺňal požiadavky algoritmu A*. Pre dosiahnutie heuristiky, vďaka ktorej by bol algoritmus A* optimálny, by bolo potrebné mať pre každý pár uzlov uloženú skutočnú vzdialenosť výslednej trasy vzhľadom na konkrétne nastavenia aplikácie. Z dôvodu pamäťovej nenáročnosti je ohodnocovanie vzdialenosti k cieľovému uzlu možné určiť vzdušnou vzdialenosťou. Táto heuristika je univerzálne a rýchlo aplikovateľná, aj keď za cenu odchýlok k optimálnemu nájdeniu trasy. Zakomponovanie takéhoto algoritmu A* predpokladá

zrýchlenie času vyhľadávania, keďže algoritmus by mal uprednostňovať uzly ležiace v smere k cieľovému uzlu. Oba zvolené algoritmy boli popísané v podkapitole 1.2.

Z nasledujúcich sekcií vyplynie, že nie všetky hrany vychádzajúce z aktuálne spracovávaného uzla predstavujú cesty, ktoré bude možné pri zadaných parametroch vyhľadávania použiť. Z tohoto dôvodu je potrebné do pôvodného Dijkstrovho aloritmu pridať pred ohodnocovaním ďalšieho uzla z aktuálneho uzla kontrolu, či hrana vedúca z aktuálneho uzla predstavuje cestu, ktorá môže byť do daného vyhľadávania zahrnutá.

3.2.2 Optimalizácia algoritmov

Zníženie výpočtového času je možné dosiahnuť súbežným (resp. striedavým) aplikovaním vyhľadávacieho algoritmu z oboch koncov. Tento prístup sa nazýva *obojsmerné vyhľadávanie* [3] a pri dvoch zadaných vrcholoch u a v, medzi ktorými sa vyhľadáva cesta, sa namiesto oblasti o polomere $\{u, v\}$ prehľadajú 2 oblasti s polomermi $\frac{\{u,v\}}{2}$, takže súčet navštívených uzlov je nižší ako pri pôvodnom prístupe. Vyhľadávanie prebieha dovtedy, kým sa tieto oblasti nestretnú, tzn. kým sa nenájde spoločný vrchol. Niekedy však existuje viacero spoločných vrcholov a v tom prípade aj keď ukončenie prehľadávania pri nájdení prvého spoločného vrcholu nezaručuje nájdenie optimálneho výsledku, predstavuje významné urýchlenie. Princíp tejto metódy ilustruje obrázok 3.3.



Obr. 3.3: Schéma obojsmerného vyhľadávania.

Pri prehľadávaní priestoru z cieľovej pozície je potrebné brať do úvahy opačný smer pohybu na jednosmerných cestách, a preto budú povolené práve cesty v protismere jazdy. Heuristikou z konečného bodu je pri algoritme A* analogicky zvolená vzdušná vzdialenosť k začiatočnému bodu.

3.2.3 Dopravné prostriedky

Vyhľadávanie najvhodnejšej cesty môže byť vo všeobecnosti ovplyvnené rôznymi parametrami. V prípade našej práce, kde grafová štruktúra reprezentuje cestnú sieť, je výber vhodných a povolených cestných úsekov, ktoré môžu byť považované za možné úseky výslednej trasy, závislý na tom, pre aké účely bude využitá výsledná trasa. Inými slovami, je potrebné vedieť určiť hrany reprezentujúce cesty, ktoré môže algoritmus zahrnúť do prehľadávania, a to podľa dopravného prostriedku, pre ktorý budeme hľadať výslednú trasu medzi dvoma miestami. V nasledujúcich riadkoch budú predstavené vybrané dopravné prostriedky, ktoré

sú podporované značením v OSM súboroch, a popis cestných pravidiel, ktoré sa s ich využitím spájajú.

Ľubovoľný

Táto možnosť pripúšťa do výslednej trasy všetky cestné úseky, ktoré boli počas načítania mapy zaradené do množiny hrán bez ohľadu na smer jazdy alebo iné obmedzenia. Nájdená výsledná trasa bude teda tou najvýhodnejšou podľa zvoleného algoritmu a parametrov vyhľadávania.

Auto

Pod týmto typom dopravného prostriedku je uvažované štandardné osobné motorové vozidlo bez akýchkoľvek veľkostných, váhových alebo rýchlostných obmedzení. Podporované sú všetky mestské cesty určené pre motorové vozidlá vrátane pomocných prístupových a obslužných ciest. Takisto aj normálne a vysokorýchlostné medzimestské trasy.

Bicykel

Tento ekologický spôsob dopravy je bežne využívaný vo všetkých končinách sveta. Podľa legislatívy je vo väčšine krajín považovaný za nemotorový dopravný prostriedok a má svoje špecifické pravidlá pre využívanie na cestách. V našej práci bude od cyklistov očakávané dodržiavanie maximálnej rýchlosti a povolený smer jazdy na týchto cestách. Okrem trás s implicitnou povahou pre využitie bicykla ako napríklad cyklotrasy, cesty v obytných oblastiach alebo bežné medzimestské cesty spájajúce obce, je možné využiť bicykel aj na cestách, ktoré budú explicitne povoľovať tento spôsob dopravy. Samozrejme, explicitne je možné použitie bicykla aj zakázať na cestách, ktoré jeho využitie implicitne predpokladajú.

Chodec

Cestami pre chodca sú považované komunikácie s implicitnou povahou určenou na peší pohyb ako napríklad pešia zóna, schody či turistické cesty. Ďalej sú tu zaradené komunikácie, ktoré sú explicitne označené, že ich chodci smú používať. Pri vyhľadávaní trasy sa neberú do úvahy obmedzenia určené smerom jazdy, keďže ani v skutočnom svete sa od chodcov nevyžaduje dodržiavanie takéhoto pravidla.

Chodec s bicyklom

Väčšina ľudí, ktorí žijú v mestách a svoj bicykel využívajú na dopravu medzi jeho jednotlivými časťami, si je vedomá toho, že existujú rôzne oblasti, v ktorých môže byť použitie tohoto dopravného prostriedku obmedzené. Príkladom je pešia zóna alebo schody. Na rozdiel od športových cyklistov je táto skupina ochotná zosadnúť z bicykla a v zmysle legislatívy sa pri tlačení bicykla správať ako chodec, ak existuje časovo alebo vzdialenosťou výhodnejšia trasa, ktorej úsek je určený práve pre chodcov. Z tohoto dôvodu je medzi uvažované spôsoby dopravy zahrnutý aj tento kombinovaný typ a nájdené trasy budú zložené z úsekov pre chodcov a cyklistov tak, aby bol výsledok čo najvýhodnejší.

3.2.4 Parametre vyhľadávania

Ako bolo uvedené v podsekcii 1.2, v Dijkstrovom algoritme zvolenom na vyhľadávanie optimálnej cesty ovplyvňujú ohodnotenia uzlov práve hrany, ktoré predchádzajú vstupu do týchto uzlov. Čím je ohodnotenie hrany nižšie, tým je táto hrana uprednostňovanejšia pri zaraďovaní do výslednej množiny hrán tvoriacich výslednú cestu. Toto ohodnotenie hrany je predstavované bezrozmernou veličinou, ktorej jedinou podmienkou je, že hrana s nižším ohodnotením musí byť z daného hľadiska „lepšia" ako hrana s vyšším ohodnotením.

V našej práci je možné nájsť dva takéto spôsoby ohodnocovania hrán, ktoré budú mať aj praktický význam. Prvým je ohodnocovanie hrán na základe dĺžok cestných úsekov, ktoré predstavujú. Na takéto určenie ohodnotenia hrany je možné použiť postup uvedený v podsekcii 2.3.1 a algoritmus bude v tomto prípade slúžiť na vyhľadanie trasy s **najkratšou vzdialenosťou** medzi dvoma miestami na mape.

Druhou možnosťou, ktorá sa naskytuje vďaka značeniu ciest v mapových dátach, je vyhľadávanie podľa **najkratšieho času** potrebného na presun z počiatočného do cieľového miesta. Ako ukazuje rovnica 3.1, pre jeden cestný úsek dĺžky Δl medzi dvoma uzlami je možné odhadovaný čas presunu Δt vyjadriť ako pomer vzdialenosti Δl a maximálnej povolenej rýchlosti v_{max} na tomto cestnom úseku.

$$\Delta t = \frac{\Delta l}{v_{max}} \tag{3.1}$$

3.3 Export výsledkov

Vyhľadávače ciest mnohokrát nemusia slúžiť iba na zobrazenie vyhľadanej trasy, ale môže byť záujem o ďalšie (často automatické) spracovanie tejto trasy. Preto je vhodné, aby naša aplikácia podporovala okrem vizualizácie výsledkov aj alternatívu vo forme textového výstupu. Navrhované sú nasledujúce 3 formáty, do ktorých by naša aplikácia mala byť schopná exportovať.

Prvým je zjednodušená verzia OSM formátu, ktorý bol popísaný v kapitole 2. Okrem uzlov a ciest bude využívať entity typu vzťah na zoskupenie ciest do výslednej trasy.

Z dôvodu uľahčenia automatického konzolového spracovávania skriptami je vhodné podporovať *plaintext* výstup. Ide o čistý text neformátovaných informácií oddelených iba medzerami, v našom prípade pôjde o šestice predstavujúce údaje pre jednotlivé hrany výslednej trasy v poradí od počiatočného miesta k miestu cieľovému. Po uvedení počtu týchto šestíc tvoriacich danú trasu budú sekvenčne nasledovať. Obsahom jednej šestice sú informácie v nasledujúcom poradí: geografická šírka a dĺžka počiatočného uzla hrany, geografická šírka a dĺžka cieľového uzla hrany, dĺžka hrany a odhadovaný čas prejazdu daným dopravným prostriedkom.

Posledným a pravdepodobne najvyužiteľnejším formátom, do ktorého bude podporovaný výstup, je formát **GPX** (GPS Exchange Format). Ide o otvorený formát založený na jazyku XML, ktorý je v dnešnej dobe podporovaný väčšinou zariadení a aplikácií spracúvajúcich GPS dáta. Vďaka exportu do tohoto formátu tak bude možné vyhľadanú trasu napríklad načítať do navigačného zariadenia. Bližšie informácie o tomto formáte sú k dispozícii na oficiálnych stránkach [16].

Kapitola 4

Implementácia

V tejto kapitole si najprv uvedieme technológie zvolené na implementáciu výslednej aplikácie a dôvody, pre ktoré boli vybrané. Pri ich voľbe som kládol okrem záujmu o portabilitu súčasne oveľa väčší dôraz na to, ako rýchlo vďaka nim bude výsledná aplikácia pracovať v porovnaní s jednoduchosťou a kompaktnosťou, s akou by mohli byť naprogramované pri voľbe iných technológií. Toto sa týka najmä kľúčovej časti aplikácie – reprezentácie grafovej štruktúry a algoritmov, ktoré ju využívajú. Ďalej si predstavíme architektúru aplikácie a popíšeme jej jednotlivé časti. V predposlednej časti tejto kapitoly bude ukázaný výsledok implementácie a popísaný spôsob jej používania a úplne nakoniec budú spomenuté problémy, s ktorými som sa v priebehu implementácie stretol a bolo ich potrebné vyriešiť.

4.1 Použité technológie

Ako implementačný jazyk bol zvolený jazyk C++, a to najmä z dôvodu rýchlosti vykonávania skompilovaného natívneho kódu, a teda aj celkovej rýchlosti vyhľadávania najkratšej cesty vo vektorových dátach v porovnaní s interpretovaným kódom skriptovacích jazykov, čo je pri tejto problematike dôležitý faktor. Taktiež, využitie objektového paradigma je v súvislosti s riešením navrhovaného problému mimoriadne vhodné a zvyšuje prehľadnosť a pochopiteľnosť zdrojového kódu.

4.1.1 Qt framework

Na návrh a implementáciu grafického používateľského rozhrania bol vybraný Qt framework [17], prevažne z dôvodu predchádzajúcich pracovných skúseností s týmto prostredím. Vďaka využitiu poskytovaných knižných funkcií je tiež výsledné implementované riešenie systémovo nezávislé.

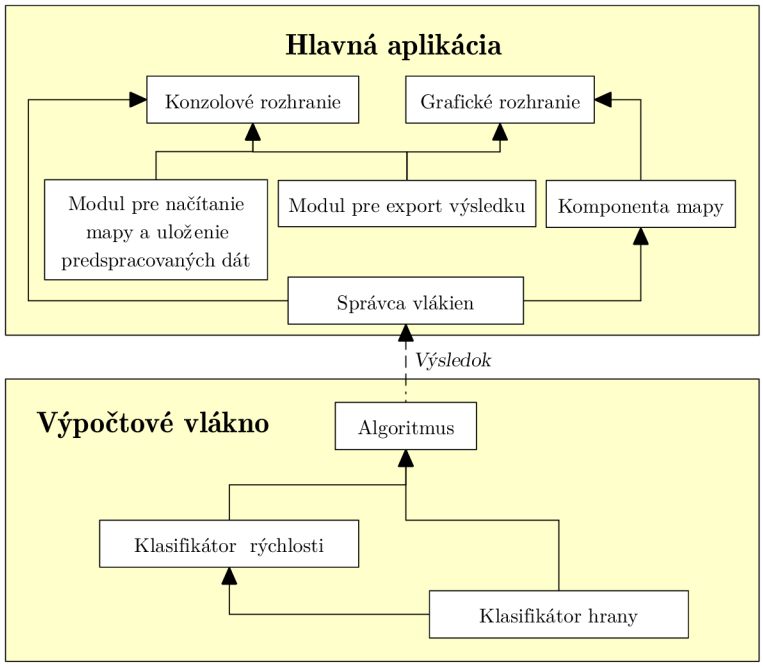
4.1.2 Vizualizácia mapy

Na vizualizáciu vyhľadanej cesty bola zvolená Qt komponenta QMapControl [18] šírená pod licenciou LGPL, ktorá umožňuje tzv. „slippy map"[1] zobrazenie, pričom je možné zvoliť zdrojovú službu grafického podkladu. Medzi bezosporné výhody tejto voľby patrí intuitívne ovládanie a podpora vektorovej vrstvy, vďaka ktorej je možné ľahko zobraziť výslednú vyhľadanú trasu.

[1] Ide o spôsob zobrazenia a ovládania mapových podkladov, pri ktorom je možné zorné pole posúvať ťahaním (príp. približovať) a mapové dlaždice sú dynamicky sťahované z webového serveru.

4.2 Architektúra aplikácie

Aplikácia bola podľa už spomínaného objektového prístupu rozčlenená a implementovaná ako niekoľko spolupracujúcich častí. Jednotlivé časti a ich hierarchické usporiadanie je zobrazené na obrázku 4.1. Vo vstupnej časti aplikácie prebehne kontrola, či bola aplikácia spustená s argumentami a v prípade, že áno, je aktivované konzolové rozhranie aplikácie spolupracujúce s používateľom výhradne cez štandardný vstup a výstup. Ak bola aplikácia spustená bez argumentov, zobrazí sa grafické používateľské rozhranie reprezentované hlavným oknom aplikácie. Oboje z týchto rozhraní využívajú pre svoju funkčnosť – načítanie zdrojových mapových dát a ich prípadné uloženie vo vlastnom binárnom formáte, vyhľadanie trasy podľa špecifikovaných parametrov a export výsledku – moduly popísané v nasledujúcej časti. Komunikácia medzi týmito jednotlivými spolupracujúcimi časťami prebieha s využitím špecifického komunikačného mechanizmu poskytovaného Qt frameworkom, a to tzv. **signálmi a slotmi**. Ďalej, ako je možné vidieť z obrázku ilustrujúceho architektúru aplikácie, výpočtový algoritmus bol implementovaný ako samostatné vlákno bežiace paralelne s hlavnou aplikáciou. Zdôvodnenie tohoto riešenia bude popísané v podsekcii 4.2.4.



Obr. 4.1: Architektúra aplikácie.

4.2.1 Modul pre načítanie mapy a uloženie predspracovaných dát

Úlohou tohoto modulu je podpora správeho načítania a spracovania podporovaných zdrojových súborov – štandardného OSM formátu obsahujúceho mapové dáta projektu OpenStreetMap vo vektorovej forme a vlastného binárneho IBP formátu s predspracovanými dátami. Obsahuje implementáciu XML parseru s pravidlami špecifickými pre analyzovanie elementov z OSM súboru potrebných pre zostavenie grafovej štruktúry, v ktorej sa bude neskôr vyhľadávať. Tento parser bol implementovaný ako jednoduchý sekvenčný parser pre XML (tzv. SAX) z dôvodov diskutovaných ďalej v sekcii 4.4. Zároveň je v priebehu načíta-

vania priebežne počítané podľa WGS84 ohodnotenie hrán určené ako vzdušná vzdialenosť dvoch uzlov, medzi ktorými hrana leží. Okrem podpory načítania tohoto formátu sú tiež implementované postupy pre načítanie binárneho formátu IBP a takisto aj možnosť uloženia grafovej štruktúry vo forme uzlov a hrán do rovnakého binárneho formátu. Špecifikácia a výhody tohoto formátu sú popísané v podsekcii 3.1.2.

4.2.2 Modul pre export výsledku

Tento modul zabezpečuje uloženie získaných informácií o vyhľadaných trasách do súborov, prípadne výpis priamo na štandardný výstup. V ponuke je niekoľko pravidiel pre formátovanie výstupných dát. Okrem momentálne najvyužívanejšieho formátu GPX na uloženie trás je ďalej podporovaný výstup v zjednodušenom OSM formáte. Pretože výstup v XML formátoch nie je najvhodnejší pre ďalšie konzolové skriptové spracovanie, je tiež podporovaný aj výstup v čistom texte.

4.2.3 Komponenta mapy

Okrem interakcie s používateľom pri výbere začiatočnej a cieľovej pozície táto komponenta komunikuje so správcom vlákien, ktorému zadáva požiadavky na zistenie potrebných trás a preberá výsledky vo forme usporiadaných zoznamov uzlov a k nim prislúchajúcich vyhodnotených štatistických údajov – vzdialeností a časových odhadov na prejdenie jednotlivých cestných úsekov. Z informácií o uzloch sú vykreslené cesty vo vektorovej vrstve mapy a štatistické údaje sú odovzdané hlavnému oknu a zobrazené v patričných informačných komponentách.

4.2.4 Správca vlákien

Vzhľadom na to, že z povahy práce ide o aplikáciu využívajúcu výpočtovo náročný algoritmus, je vhodné tento výpočet presunúť do samostatného vlákna. Bez tohoto prístupu by časť aplikácie slúžiaca na interakciu s používateľom počas celého prebiehajúceho hľadania trás prestala reagovať na používateľský vstup. Úlohou tohoto modulu hlavnej aplikácie je vytvorenie vlákna obsahujúceho výpočtový algoritmus vyhľadávajúci trasu a komunikácia s týmto vláknom zahrňujúca zadávanie úloh a preberanie výsledkov.

4.2.5 Vlákno algoritmu

Algoritmus je implementovaný tak, aby bol schopný behu v samostatnom vlákne. Po prijatí zadania úlohy obsahujúceho pre aký dopravný prostriedok a podľa čoho má byť uprednostňovaná vyhľadávaná trasa je spustené samotné vyhľadávanie. Jeho implementácia podporuje vyhľadávanie podľa 4 rôznych algoritmov – jednosmerný a obojsmerný Dijkstrov algoritmus a jeho jedno- a obojsmernú variantu A* s heuristikou vzdušnej vzdialenosti. Pri prehľadávaní využíva klasifikátor hrán na zistenie, či prehľadávaná hrana grafu je pri daných parametroch vyhľadávania povolená a môže byť považovaná za potenciálny úsek výslednej nájdenej trasy. Klasifikátor rýchlosti je využitý pri zisťovaní maximálnej povolenej rýchlosti pohybu po danej ceste určitým dopravným prostriedkom. Z tejto a informácie o dĺžke hrany je následne zisťovaný čas, za ktorý sa dá tento úsek prejsť.

4.2.6 Klasifikátor hrany

Táto časť aplikácie má za úlohu určiť, či je hrana predstavujúca úsek cesty medzi dvoma uzlami prejazdná v danom smere konkrétnym dopravným prostriedkom. Pri obojsmernom vyhľadávaní berie do úvahy, z ktorej vetvy prehľadávania požiadavka pochádza.

Implicitné pravidlá, podľa ktorých klasifikátor určuje povolené hrany, sú zobrazené v nasledujúcej tabuľke 4.1:

Tabuľka 4.1: Povoľujúce implicitné pravidlá implementované v klasifikátore hrán.

Typ prostriedku	**Smer jazdy**	**Povolený typ cesty**
Ľubovoľný	ľubovoľný	všetky
Auto	podľa pravidiel cesty	`primary, secondary, tertiary, trunk, motorway, residential, track, service, unclassified`
Bicykel	podľa pravidiel cesty	`cycleway, bridleway, path, residential, service, primary, secondary, tertiary, track, unclassified`
Chodec	ľubovoľný	`footway, steps, pedestrian, bridleway, path, residential, service, unclassified`
Chodec s bicyklom	ľubovoľný/podľa pravidiel cesty	`cycleway, footway, bridleway, path, steps, pedestrian, residential, service, primary, secondary, tertiary, track, unclassified`

Okrem týchto implicitne povolených ciest je povolený prejazd bicyklom resp. prechod chodcom aj po cestách, ktoré majú explicitne označené takéto povolenie v súbore s mapou.

4.2.7 Klasifikátor rýchlosti

Jeho úlohou je určiť odhadovanú priemernú rýchlosť, ktorou sa môže vybraný dopravný prostriedok pohybovať po danom cestnom úseku. Aj keď sa v praxi uplatňujú rôzne rýchlostné obmedzenia pre rovnaký typ cesty, ktorá prechádza obcou alebo sa nachádza mimo nej, implementovaný klasifikátor rýchlosti nerozlišuje medzi týmito dvoma oblasťami, a to prevažne z dôvodu komplikovaného určovania, ktoré z ciest patria do obce a ktoré už nie.

Algoritmus klasifikátora pri vyhodnocovaní rýchlosti je nasledovný:

1. Nastav rýchlosť na predvolenú rýchlosť 50 km/h
2. Uprav rýchlosť na maximálnu povolenú rýchlosť špecifickú pre danú cestu podľa tabuľky 4.2
3. Obmedz rýchlosť na rýchlosť najrýchlejšieho povoleného a použiteľného dopravného prostriedku pre daný typ dopravy na tejto ceste
    - Pre auto táto rýchlosť nie je ďalej upravovaná

- Priemerná rýchlosť bicykla bola empiricky zvolená na 20 km/h
- Preferovaná rýchlosť chodca je podľa [19] 5 km/h

Tabuľka 4.2: Maximálne rýchlosti pre jednotlivé cesty.

Typ cesty	**Maximálna rýchlosť [km/h]**	**Typ cesty**	**Maximálna rýchlosť [km/h]**
`motorway`	130	`cycleway`	bicykel (20)
`trunk`	110	`path`	bicykel (20)
`primary`	85	`steps`	60% chodca (3)
`secondary`	85	`pedestrian`	chodec (5)
`tertiary`	85	`footway`	chodec (5)
`residential`	východzia (50)	`bridleway`	bicykel (20)
`service`	20	`unclassified`	85
`track`	20		

Rýchlosti pre jednotlivé cesty vychádzajú z informácií poskytnutých v dokumentácii [10]. Pretože rôzne krajiny majú odlišne určené rýchlostné obmedzenia a implementovaná aplikácia pri vyhodnocovaní cesty neberie do úvahy oblasť, v ktorej sa cesta nachádza, bolo potrebné určiť jednotné rýchlosti za cenu určitých kompromisov. Ďalej, uvedené zdrojové údaje udávajú maximálnu povolenú rýchlosť na týchto cestách, avšak v skutočnosti vodiči túto rýchlosť dodržiavajú iba približne – niektorí ju prekračujú, iní si nechávajú rýchlostnú rezervu. Tento faktor bol využitý pri volení maximálnych rýchlostí využívaných aplikáciou, a to tak, aby približne vyjadrovali priemerné maximálne rýchlosti pre daný typ cesty spomedzi všetkých krajín (z dôvodov testovania aplikácie na európskych krajinách boli ich rýchlosti uprednostňované).

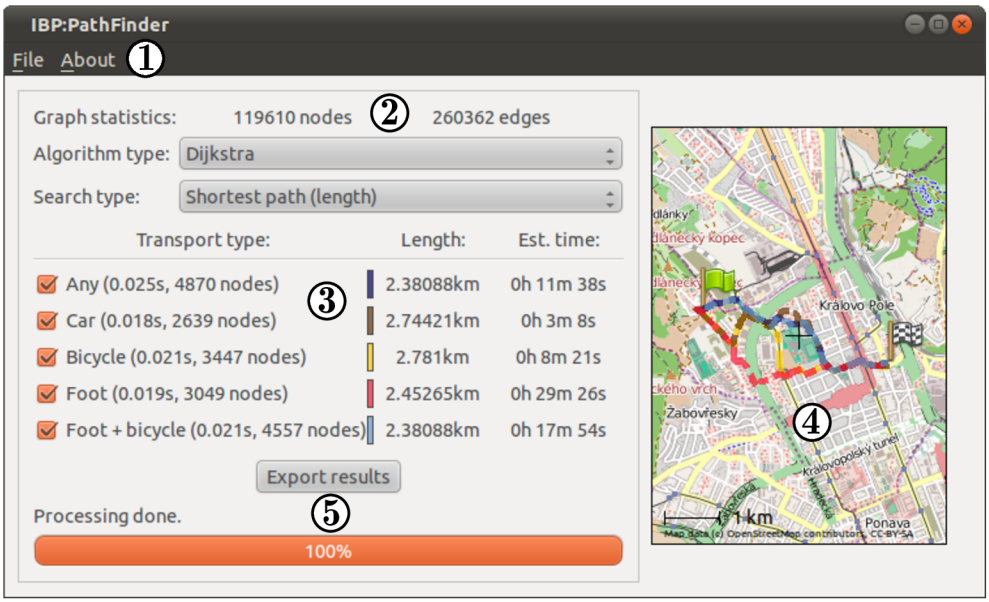
4.3 Používateľské rozhranie a ovládanie aplikácie

Pri vytváraní aplikácie bolo dbané na to, aby výsledný produkt bolo možné jednoducho používať graficky a aj z prostredia konzoly. Pri spustení aplikácie bez argumentov je aktivované grafické rozhranie aplikácie, v opačnom prípade komunikuje aplikácia s používateľom iba prostredníctvom štandardného vstupu a výstupu.

4.3.1 Grafické používateľské rozhranie aplikácie

Grafické rozhranie je reprezentované hlavným oknom aplikácie znázorneným na nasledujúcom obrázku 4.2, ktoré je tvorené nižšie popísanými časťami:

1. **Ponuka aplikácie** – umožňuje používateľovi otvoriť súbor s dátami projektu OpenStreetMap a uložiť načítané dáta v predspracovanej forme pre zníženie veľkosti a urýchlenie budúceho načítania
2. **Štatistika načítaných dát a nastavenie parametrov vyhľadávania** – informuje o počte načítaných uzlov a hrán, ktoré budú prehľadávané; ďalej ponúka možnosť zvoliť algoritmus, ktorý bude použitý na vyhľadanie, a pravidlo pre určenie najvýhodnejšej trasy (dĺžka resp. odhadovaný čas na jej prejdenie)

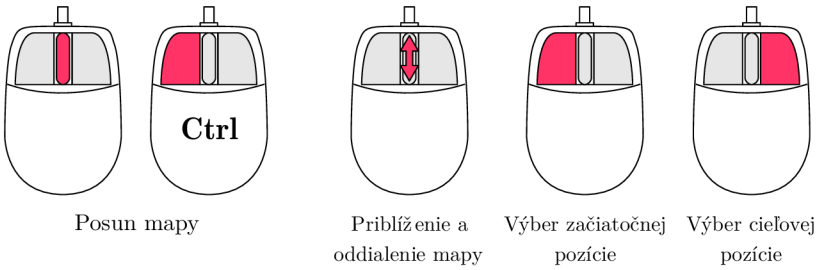


Obr. 4.2: Vzhľad grafického rozhrania implementovanej aplikácie.

3. **Tabuľka s výsledkami vyhľadávania** – pre jednotlivé typy dopravných prostriedkov informuje o uplynutom čase potrebnom na nájdenie výsledku, počte expandovaných uzlov, farbe zvizualizovanej trasy na mapovej komponente a podrobnostiach vyhľadanej trasy (dĺžke trasy a odhadovanom čase potrebnom na presun s využitím daného dopravného prostriedku)

4. **Komponenta mapy** – slúži na zobrazenie mapových podkladov, výber začiatočnej a cieľovej pozície vyhľadávanej trasy a jej následnú vizualizáciu; ovládanie tejto komponenty je znázornené na obrázku 4.3

5. **Indikátor priebehu vyhľadávania trasy a export výsledkov** – informuje o priebehu vyhľadávania a po jeho dokončení ponúkne možnosť exportovať výsledky do súboru



Obr. 4.3: Ovládanie mapovej komponenty.

4.3.2 Konzolové rozhranie aplikácie

Toto rozhranie bolo implementované z dôvodu podpory unifikovaného zadávania úloh bez nutnosti grafickej vizualizácie a interaktívneho ovládania. Jeho využitie je vhodné napríklad pri automatizovanom spracovávaní a vyhodnocovaní viacerých úloh pomocou skriptov. Argumenty, ktoré prijíma konzolové rozhranie sú uvedené v prílohe A.

4.4 Problémy pri implementácii

Počas vývoja tejto aplikácie som sa stretol s niekoľkými problémami, pre ktoré bolo potrebné nájsť riešenia. Pôvodne zvolený objektový prístup k parsovaniu XML súborov nazývaný DOM (Document Object Model) potrebuje mať pre svoju funkčnosť založenú na zostavení stromovej štruktúry XML dokumentu dostatok operačnej pamäte, aby mohol kompletne načítať celý vstupný XML súbor. Pri prvých testoch so zdrojovými OSM súbormi máp, ktoré mali veľkosť rádovo niekoľko stoviek kilobytov a zaberali rozlohu malých mestských štátov, nebol problém s funkčnosťou aplikácie, avšak mapu s dátami Českej republiky o veľkosti 6 gigabytov nebolo možné pri ďalšom testovaní načítať. Vhodná alternatíva, ktorá vyriešila tento problém, sa nazýva SAX (Simple API for XML) [20]. Táto metóda rieši spracovávanie súboru sekvenčne a o svojom priebehu informuje vyvolávaním udalostí, na ktoré môže program reagovať. Vďaka jej použitiu sú pamäťové nároky počas načítavania XML súboru minimálne a prišlo aj k samotnému urýchleniu načítavania, keďže zostavovanie grafu z OSM súboru prebieha súbežne s analyzovaním súboru počas čítania.

Ďalším problémom bola znížená používateľská odozva aplikácie keď výpočtový algoritmus prebiehal v rovnakom vlákne ako okno s grafickým používateľským rozhraním, a tak bol navrhnutý správca vlákien a výpočtový algoritmus presunutý do samostatného vlákna.

Iná záležitosť, ktorá čiastočne komplikovala programovanie a ladenie aplikácie, bolo správanie sa mapovej komponenty počas niektorých dní vývoja, a to tak, že nezobrazovala mapové podklady v priebehu dňa. Plne responsívna bola až v neskorých nočných hodinách. Po ladení mapovej komponenty a hľadaní príčiny, ktorá tento problém spôsobovala, som dospel k názoru, že servery OpenStreetMap v čase vysokej záťaže výrazne znevýhodňujú pri získavaní mapových dlaždíc iné požiadavky ako tie, ktoré pochádzajú z webového prehliadača navštevujúceho ich online aplikáciu. Tento problém som sa pokúšal vyriešiť zmenou zdrojových serverov, avšak neúspešne.

Kapitola 5

Testovanie a porovnanie výsledkov

Obsahom tejto kapitoly je testovanie implementovanej aplikácie a porovnanie jej výsledkov s konkurenčnými riešeniami. V prvej časti sa budeme venovať už spomínanému testovaniu, a to najprv analyzovaním funkčnosti implementovaných algoritmov. Ďalej sa zameriame na ich rýchlosť podľa voľby typu dopravného prostriedku. Zistíme, že voľba parametrov pri vyhľadávaní môže mať významný vplyv na rýchlosť aplikácie a objavené javy sa pokúsime zdôvodniť. V ďalšom teste odhalíme vzťah medzi počtom spracovaných uzlov a rýchlosťou vyhľadávania, po ktorom sa zameriame na analýzu efektivity implementácie. Druhá časť tejto kapitoly bude venovaná porovnaniu výstupov, ktoré naša aplikácia poskytuje, s výsledkami konkurenčných riešení využívajúcich rovnaký voľne dostupný ale aj proprietárne zdroje dát.

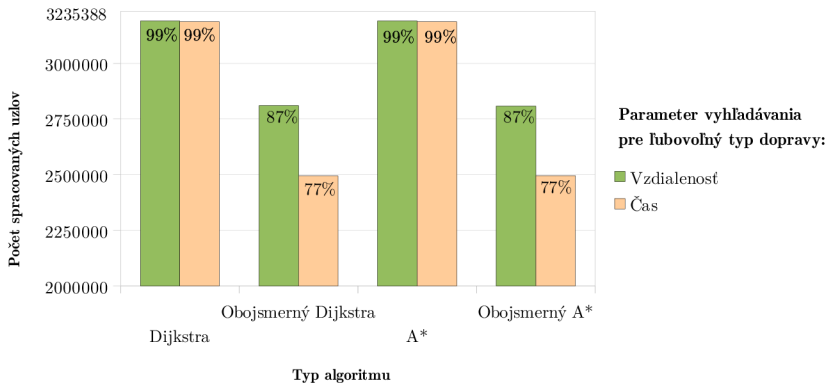
5.1 Testovanie aplikácie

Všetky nasledujúce výkonnostné testy prebiehali na notebookovej zostave zloženej z procesora Intel® Centrino® Core™2 Duo 2.1GHz, 4GB DDR2 800MHz operačnej pamäte a čipsetu Intel® GM45 Express s linuxovým operačným systémom Ubuntu 11.04. Implementovaná aplikácia bola skompilovaná v prostredí Qt SDK verzie 1.1.4 ako „release" verzia s optimalizáciou 3. stupňa.

5.1.1 Porovnanie algoritmov podľa počtu spracovaných uzlov

Pre tento test boli zvolené mapové dáta Českej republiky. Počet uzlov tejto cestnej siete predstavuje 3 235 388 a počet hrán je 6 880 682. Vyhľadávaná trasa z počiatočného bodu ležiaceho na úplnom západe Českej republiky v meste Hranice vedúca do cieľa na jej východe v meste Jablunkov naprieč celou Českou republikou bola zvolená s cieľom zistiť, koľko uzlov z celkového počtu tieto algoritmy spracujú pri takmer najvzdialenejšej trase, ktorú je možné medzi týmito uzlami nájsť. Ako parameter vyhľadávania boli použité obe možnosti – vyhľadanie trasy s najkratšou vzdialenosťou aj s najkratším odhadovaným časom prepravy. Za typ dopravného prostriedku bola zvolená možnosť ľubovoľný, ktorá pripúšťa do vyhľadávania všetky načítané hrany.

Výsledky tohoto testu boli prenesené do grafu 5.1 spolu s percentuálnym označením, koľko uzlov z celkového počtu v načítanej grafovej štruktúre algoritmus spracoval. Vidíme, že s takmer rovnakým počtom uzlov, ktoré po prepočte dosiahli skoro sto percent a spracovali skoro všetky uzly, boli jednoduché verzie *Dijkstrovho algoritmu* a *A**. Predpoklady, že



Obr. 5.1: Počet spracovaných uzlov pri vyhľadávaní trasy naprieč celou Českou republikou.

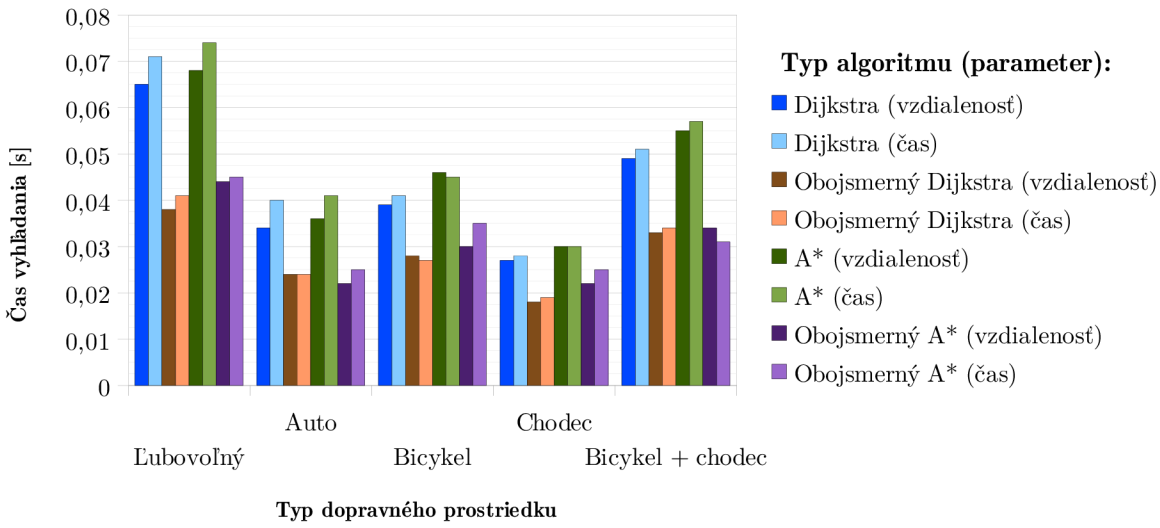
použitie heuristickej funkcie napomôže prehľadávaniu, sa z tohto hľadiska nepotvrdili. Optimalizácii v podobe obojsmerných variánt týchto algoritmov sa darilo lepšie, aj keď ani ich vzájomný výsledok sa nelíši pre daný parameter. Z celkového počtu uzlov sa pri vyhľadávaní najkratšej trasy počet spracovaných uzlov zredukoval o 12 %, pri vyhľadávaní podľa odhadovaného času je oproti tomuto výsledku možné pozorovať takmer dvojnásobné zlepšenie, a to celých 22 %. Znížený počet spracovaných uzlov pri obojsmernom vyhľadávaní vyplýva z jeho podstaty, ktorá bola vysvetlená v podsekcii 3.2.2, avšak rozdiel spôsobený zmenou parametra je oveľa zaujímavejším zistením. Tento jav je možné vysvetliť tým, že sieť uzlov grafu je rovnomerne rozložená, a tak ohodnotenia hrán, ktoré sa nachádzajú medzi nimi, sú podľa vzdialenosti veľmi podobné. Avšak, po zakomponovaní maximálnej rýchlosti, ktorou je možné sa po daných cestách pohybovať, vzniknú v ohodnoteniach hrán značné rozdiely a preferované je vyhľadávanie pozdĺž „chrbtovej sieti", ktorú tvoria práve vysokorýchlostné komunikácie.

5.1.2 Rýchlosť vyhľadávania

Pri ďalšom teste bola zvolená za zdrojové dáta mapa oblasti Brna s počtom uzlov 119 610 a počtom hrán 260 362. Na trase s počiatočným miestom pri Fakulte informačních technologií VUT v Brně na Božetechovej ulici a cieľovým miestom pred hlavnou vlakovou stanicou na Nádražní ulici boli zaznamenané časy vyhľadávania všetkých algoritmov pre obe bližšie špecifikujúce parametre. Tieto výsledky zobrazuje graf 5.2.

Pri pohľade na výsledných dvadsať zaznamenaných časových údajov vidíme, že na rýchlosť vyhľadávania má významný vplyv nielen typ algoritmu ale aj zvolený dopravný prostriedok. S najkratšími časmi vyhľadávania sa umiestnil *chodec* a najpomalšie vyhodnocovanie prebiehalo pre *ľubovoľný* typ dopravného prostriedku. Túto závislosť rýchlosti vyhľadávania na type dopravného prostriedku môžeme objasniť počtom uzlov, ktoré je potrebné spracovať. Pri *ľubovoľnom* type dopravy sa berú do úvahy všetky cesty. Spolu s nimi vzrastá exponenciálne aj počet navštívených a expandovaných uzlov. Naopak, ciest, na ktorých sa majú pohybovať chodci, je kvôli nedostatočnému značeniu v zdrojových dátach najmenej. Komunita, ktorá ich vytvára v tejto oblasti, teda uprednostňuje podrobnejšie mapovanie ciest pre motorové dopravné prostriedky.

Pri porovnaní rýchlosti vyhodnocovania podľa jednotlivých typov algoritmov sa objavuje súvis s výsledkami z predchádzajúceho testu s počtom expandovaných uzlov. Jednoduchý



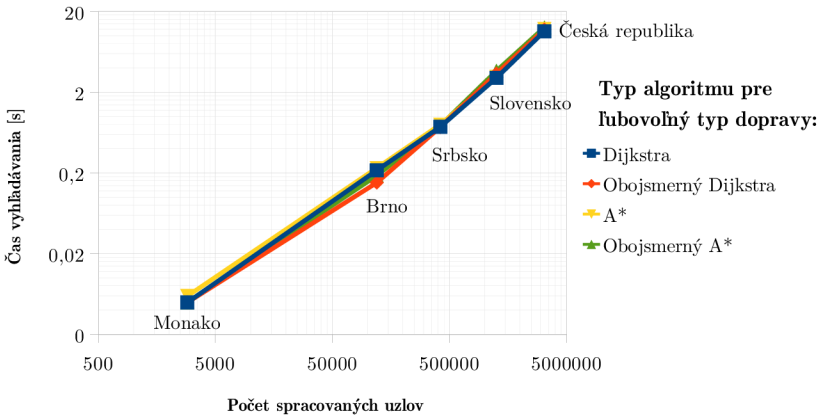
Obr. 5.2: Rýchlosť vyhľadávania pre jednotlivé dopravné prostriedky na trase z FIT VUT k hlavnej vlakovej stanici v Brne.

Dijkstrov algoritmus a *A** majú výsledné časy vzhľadom na konkrétny typ dopravného prostriedku veľmi podobné, pričom *A** je mierne pomalší. Dôvodom je pridaná heuristická funkcia, ktorá musí odhadovať zvyšnú cestu k cieľovému uzlu. Ďalej, vďaka nižšiemu počtu navštívených uzlov pri obojsmerných vyhľadávaniach vidíme badateľný rozdiel v časoch pri oboch obojsmerných algoritmoch a ich jednoduchých náprotivkoch. Obojsmerné varianty sú v tomto ohľade rýchlejšie približne o tretinu. Poslednou vecou, ktorú si môžeme všimnúť je, že vyhodnocovanie na základe parametra pre *najkratšiu vzdialenosť* je o niečo rýchlejšie ako vyhľadávanie trasy s parametrom *najkratšieho času*, ktorý je odhadovaný na prepravu medzi počiatočným a cieľovým uzlom. Toto je spôsobené spôsobom implementácie týchto parametrov, keďže prvá možnosť využíva ohodnotenie hrany obsahujúce ich vzdialenosti a druhá možnosť z týchto údajov vychádza a prepočítava ich podľa maximálnej povolenej rýchlosti na odhadovaný čas. Efekt „chrbtovej siete" pozorovaný v podsekcii 5.1.1 sa v mestskej cestnej sieti príliš neprejavil a jeho prípadné náznaky mohli byť vďaka vyššie popísanému spôsobu implementácie potlačené. Drobné odchýlky v meraniach mohli byť spôsobené nevhodným plánovaním úloh na úrovni operačného systému.

5.1.3 Rýchlosť vyhľadávania vzhľadom na veľkosť cestného grafu

Pre tento test boli rovnakým spôsobom ako bol popísaný v podsekcii 5.1.1 zvolené uzly aj v ďalších zdrojoch mapových dát. Vyhľadané výsledky predstavovali takmer najdlhšie trasy, aké bolo možné v daných zdrojových dátach nájsť, a ich zdrojové súbory boli testované v poradí podľa počtu uzlov, ktoré zachytávali. Najmenším bola administratívna oblasť Monaka s počtom uzlov 2876 a hrán 6202, ďalej nasledovali oblasť Brna (119 610 uzlov, 260 362 hrán), Srbsko (417 860 uzlov, 887 340 hrán), Slovensko (1 260 414 uzlov, 2 657 556 hrán) a nakoniec Česká republika s grafom tvoreným 3 235 388 uzlami a 6 880 682 hranami. Cieľom testu bolo zistiť, aký vplyv má veľkosť štruktúry cestného grafu načítaného v pamäti na vyhľadávanie v nej. Výber trasy naprieč jednotlivými oblasťami a zvolenie parametra vyhľadávania podľa najkratšej cesty predpokladá, že pri vyhľadávaní bude spracovaná väčšina

uzlov grafovej štruktúry.



Obr. 5.3: Rýchlosť vyhľadávania vzhľadom na počet spracovaných uzlov pri vyhľadávaní trás naprieč jednotlivými oblasťami.

Aj keď sa o rýchlosti vyhľadávania v malej oblasti akou je Monako polemizovať nedá, s narastajúcou veľkosťou cestnej grafovej štruktúry je možné sledovať spomalenie, ktoré pri rozsiahlej a vcelku podrobne zmapovanej oblasti akou je Česká republika predstavuje vyše desať sekúnd. Mohlo by sa zdať, že doba tohoto spomaľovania má exponenciálny trend, avšak výsledky ukázali, že ide o lineárny trend, a teda čas hľadania pri všetkých algoritmoch je približne priamo úmerný počtu spracovaných uzlov. Z výsledkov ďalej vyplýva, že grafickú aplikáciu je možné tzv. *realtime* (v skutočnom čase, pri nízkej dobe odozvy) používať na vyhľadávanie trás ešte pri cca. stotisíc uzloch grafu a zhruba dvojnásobnom počte hrán.

5.1.4 Analýza implementácie

Posledným aspektom, na ktorý sme sa pri testovaní našej aplikácie zamerali, bolo analyzovanie výpočtového času stráveného v jednotlivých funkciách počas priebehu vyhľadávania vo výpočtovom vlákne. Na analýzu bol použitý program *Callgrind* a zvizualizovaný výsledok pre vyhľadávanie trasy s najkratším časom prepravy pomocou Dijkstrovho algoritmu je zobrazený v prílohe B.1. Ako je možné vidieť, takmer celý výpočtový čas vlákna algoritmu zaberá funkcia `ibp::Algorithm::Dijkstra()`, ktorá slúži na vyhľadanie trasy. V nej sa tento čas delí medzi ďalšie volané funkcie, z ktorých veľmi významnú časť – vyše štyri pätiny – spotrebúvajú práve funkcie slúžiace na prácu s asociatívnymi poliami uzlov a hrán. Naše vlastné pomocné funkcie slúžiace na určenie povolených hrán a zistenie ohodnotenia hrany spotrebujú iba približne $2,6$ % z celkového času. Zvyšná necelá jedna pätina je spotrebovaná priamo vo funkcii `ibp::Algorithm::Dijkstra()` a prípadne ďalších volaných funkciách, ktorých spotreba z celkového času je zanedbateľná.

5.2 Porovnanie výsledkov

V tejto časti kapitoly si zhrnieme výsledky porovnávajúce trasy našej aplikácie a konkurenčných riešení. Miesta, medzi ktorými bola vyhľadávaná trasa, boli zvolené na základe dostatočne rôznorodého výskytu odlišných typov ciest, ktoré sa medzi nimi nachádzajú a vďaka ktorému plynú nie celkom jednoznačné výsledky. V konečnom dôsledku závisí pri

vyhľadávaní na preferenciách jednotlivých riešení a iných príčinách, ktoré sa pokúsime zdôvodniť.

Za konkurenčné riešenia k našej aplikácii boli zvolené známe voľne prístupné online služby na vyhľadávanie ciest. Predstaviteľom služby, ktorá taktiež využíva mapové dáta projektu OpenStreetMap je služba OpenRouteService. Ďalšími zvolenými službami sú komerčné a pravdepodobne najvyužívanejšie riešenia v tejto oblasti s názvami Google Maps a Bing Maps, ktoré využívajú vlastné proprietárne zdroje dát. Výsledky porovnávania popísaného v nasledujúcich podsekciách sumarizuje tabuľka 5.1.

Tabuľka 5.1: Sumarizácia výsledkov vyhľadávačov ciest pre rozličné dopravné prostriedky.

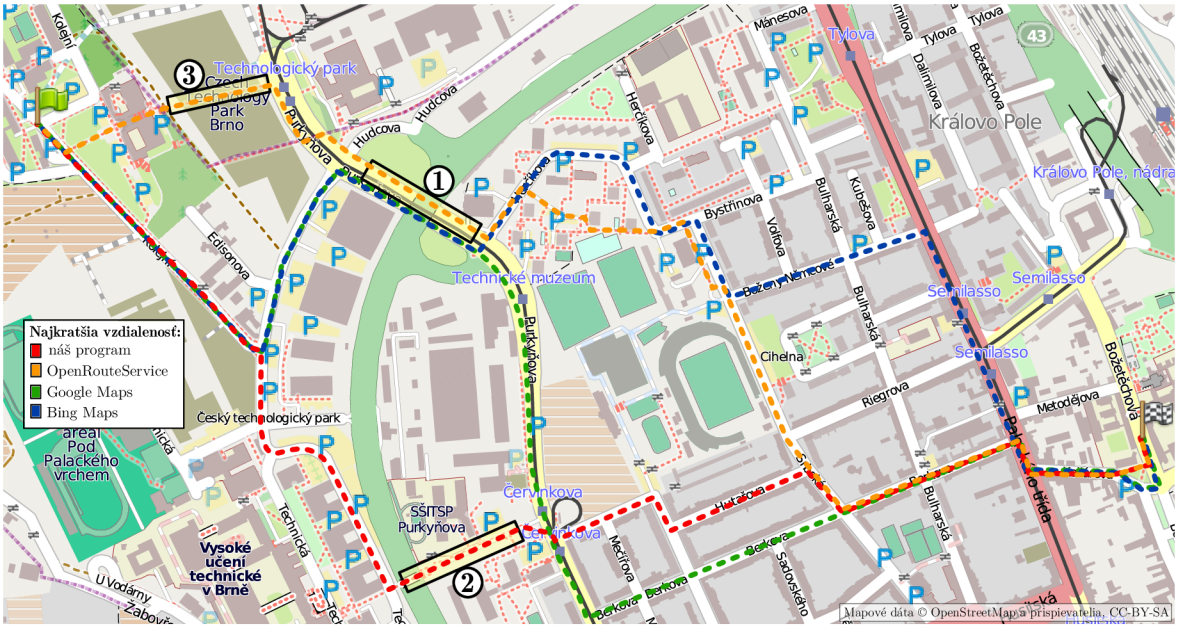
Aplikácia	**5.2.1: Trasa pre chodca**		**5.2.2: Trasa pre motorové vozidlo**		**5.2.3: Trasa pre cyklistu**	
	Dĺžka [km]	**Čas [min]**	**Dĺžka [km]**	**Čas [min]**	**Dĺžka [km]**	**Čas [min]**
naša aplikácia	2,43	29	4,8	5	2,86	9
OpenRouteService	2,4	26	4,9	5	2,6	10
Google Maps	2,8	31	5,9	12	—	—
Bing Maps	2,7	32	5,8	11	—	—

5.2.1 Najkratšia trasa pre chodca

Pre porovnanie trás zameraných na využitie pre chodcov bolo zvolené začiatočné miesto na Kolejní ulici kde sídlia Koleje pod Palackého vrchem a cieľovým miestom je opäť Fakulta informačních technologií VUT v Brně. Všetky štyri porovnávané aplikácie boli nastavené na vyhľadávanie najkratšej trasy a ich výsledky boli zaznačené do obrázku 5.4.

Ako môžeme vidieť, pravdepodobne najoptimálnejšiu trasu vyhľadala služba OpenRouteService. Zaujímavosťou je, že pri posune počiatočného miesta o pár desiatok metrov nižšie vyhľadá služba OpenRouteService rovnakú trasu ako naša aplikácia, a ďalej, pri pôvodnom počiatočnom mieste aj naša aplikácia vyhľadá trasu totožnú s odporúčanou trasou služby OpenRouteService, ale pre nastavenie ľubovoľného typu dopravy. Pri hľadaní príčiny tejto odchýlky bolo zistené, že mapové dáta služby OpenStreetMap neobsahujú v oblasti zaznačenej číslom 1 vhodné označenia ciest aj pre chodcov alebo aspoň prítomnosť chodníkov, z čoho vyplýva, že služba OpenRouteService predpokladá implicitne iné povolenia vyplývajúce z rôznych typov ciest.

Ďalšie 2 komerčné riešenia pri svojich výsledkoch upozorňujú, že na vyhľadaných trasách nemusia byť chodníky alebo cesty pre chodcov (tento typ upozornenia zobrazia napríklad aj pri vyhľadaní trasy pre chodca, ktorá vedie po diaľnici). Vyhľadané trasy týchto služieb sa značne líšia. Trasa služby Bing Maps je až na určité odchýlky podobná trase, ktorú vyhľadala služba OpenRouteService, avšak služba Google Maps vyhodnotila za najvhodnejšiu úplne inú trasu. Pri bližšom skúmaní bolo zistené, že dáta ani jednej z nich neevidujú existenciu ciest zaznačených oblasťami 2 a 3, čo je možnou príčinou týchto výsledkov. Takisto, tieto dáta nezachytávajú chodníky v obytných oblastiach v takej podrobnej miere ako dáta OpenStreetMap, ktoré služba OpenRouteService patrične využila.



Obr. 5.4: Výsledok testu popísaného v podsekcii 5.2.1 porovnávajúci nájdené trasy viacerých aplikácií pre chodca.

5.2.2 Trasa pre motorové vozidlo

Pri porovnávaní vyhľadanej trasy pre motorové vozidlo bol zvolený počiatočný bod opäť pri Fakulte informačních technologií VUT v Brně, avšak cieľ bol vybraný pred hlavnou vlakovou stanicou rovnako ako v teste popísanom v podsekcii 5.1.3. Naša aplikácia a služba OpenRouteService boli nastavené na vyhľadávanie najrýchlejšej trasy a pri zvyšných dvoch komerčných aplikáciách sa brali do úvahy prvé trasy automaticky vybrané týmito aplikáciami, keďže ponúkali aj alternatívne trasy. Výsledné trasy zobrazuje mapa v prílohe C.2.

Prvú približne polovicu vzdialenosti vedú všetky testované aplikácie po identickej trase. Ako prvá sa odkloní trasa služby Google Maps, ktorá odporučí prejazd cez ulicu Pionýrská označenú oblasťou 1, ktorá je v čase písania tejto práce už približne trištvrte roka neprejazdná z dôvodu rekonštrukcie. Aj keď výsledná trasa služby Bing Maps cez túto ulicu nevedie, taktiež ju považuje za prejazdnú. Tu je možné vidieť neaktuálnosť dát, ktoré tieto služby využívajú. Na rozdiel od nich je vďaka pohotovosti komunity v dátach služby OpenStreetMap táto cesta patrične označená ako neprejazdná.

V poslednej časti vzdialenosti sa ako ďalšia jemne vychýli trasa našej aplikácie, ktorá uprednostní menej významné ulice, a po hlavnom jazdnom ťahu ďalej vedú trasy služieb OpenRouteService a Bing Maps. Nakoniec však vo výhodný okamih služba OpenStreetMap využije odbočenie priamo k cieľu, pričom trasa služby Bing Maps obchádza celú oblasť vlakovej stanice a dostáva sa do cieľa z opačnej strany.

Pre zaujímavosť bola zaznačená aj vyhľadaná najkratšia trasa, keďže naša aplikácia aj služba OpenRouteService podporujú túto možnosť parametrizácie. Vyhľadané trasy sú úplne identické s výslednou dĺžkou 4.5 kilometra, pričom naša aplikácia odhaduje čas presunu na 5 minút a 26 sekúnd a konkurenčná online služba využívajúca rovnaký zdroj dát je trochu benevolentnejšia a odhaduje prejdenie tejto trasy počas približne 6 minút.

5.2.3 Najkratšia trasa pre cyklistu

Keďže služba Bing Maps vyhľadávanie cyklistických trás nepodporuje a služba Google Maps toto umožňuje iba experimentálne pre oblasť Severnej Ameriky, v tomto teste sme porovnávali iba našu aplikáciu s výsledkom služby OpenRouteService. Trasa bola zvolená rovnaká ako v podsekcii 5.2.1 a výsledné trasy sú zaznačené v grafickej prílohe C.1.

Nájdené trasy sa v druhej polovici zhodujú, avšak ich prvá polovica je odlišná, keďže naša aplikácia nenašla vhodné povolenie na prejazd po ceste zaznačenej v oblasti 1. Za zmienku stojí trasa, ktorá bola nájdená našou aplikáciou pre kombinovaný typ dopravy chodec s bicyklom. Tá využila cesty schodné chodcami na skrátenie výslednej trasy, avšak z časového hľadiska sa táto voľba neoplatí. Dĺžka tejto trasy je 2,4 kilometra pri odhadovanom čase presunu vyše dvojnásobnom, a to približne za 19 minút. Ako už bolo spomínané v teste s chodcom, v oblasti označenej číslom 1 je nevhodné značenie neumožňujúce odbočenie samostatne chodcovi a ani tentokrát bicyklu, ale ako vidíme, pri ich spoločnej kombinácii naša aplikácia tento problém prekonala.

5.3 Zhodnotenie výsledkov

Pri testovaní aplikácie v prvej časti kapitoly sme overili, že obojsmerné verzie implementovaných algoritmov sú rýchlejšie ako ich jednoduché náprotivky. Pri porovnávaní rýchlosti vyhľadávania vzdialenosťou najkratšej versus časovo najrýchlejšej trasy naprieč celou Českou republikou bol pozorovaný významný časový rozdiel v prospech vyhľadávania preferujúceho časové ohodnotenia. Tento fenomén, ktorý sme označili ako jav „chrbtovej siete" je spôsobený výraznejšími rozdielmi v ohodnoteniach hrán grafu cestnej siete pri zakomponovaní maximálnej povolenej rýchlosti prejazdu po daných cestách ako pri jednoduchom ohodnotení založenom iba na dĺžkach hrán predstavujúcich tieto cesty. Algoritmus prehľadávajúci takto ohodnotenú sieť je pri veľkých vzdialenostiach vedený pozdĺž vysokorýchlostnej komunikácie. Spomedzi parametrizácie dopravných prostriedkov obstáli rýchlejšie tie dopravné prostriedky, ktoré pripúšťajú do vyhľadávania menší počet hrán. Medzi spomaľovaním rýchlosti vyhľadávania a narastajúcej veľkosti cestného grafu sme zistili lineárny trend.

Výsledky porovnávania výstupov našej aplikácie a konkurenčných riešení pre rôzne dopravné prostriedky sú zvizualizované v podsekcii 5.2.1 a v prílohe C. Príčiny odlišností vyhľadaných trás sú viaceré. Prvým dôvodom, ktorý už bol vyššie uvedený, je rôzny zdroj mapových dát. Ďalej, samotné nastavenie ohodnocovacej funkcie má veľký vplyv na vzhľad výslednej trasy. Každé z riešení prikladá inú dôležitosť, prípade očakáva odlišnú rýchlosť pohybu na určitých typoch ciest. Okrem toho je viac než pravdepodobné, že tieto služby, ktoré zabezpečujú vyhľadávanie ciest v celosvetovom meradle, používajú pokročilé algoritmy, ktorých výsledky na úkor rýchlosti spracovania požiadaviek nie sú optimálne. Z porovnaní je však zrejmé, že výstupy našej aplikácie sú veľmi podobné ostatným výsledkom, a tak sa dajú považovať za dostatočne relevantné. V niektorých prípadoch sú dokonca aj zreteľne lepšie ako odporúčané trasy iných bežne používaných služieb.

Kapitola 6

Možné rozšírenia

Aj keď naša aplikácia preukázala pri testovaní značnú relevantnosť a použiteľnosť výsledkov na praktické využitie, nemožno ju považovať za konečný produkt.

Implementované jednosmerné a obojsmerné algoritmy založené na princípe Dijkstrovho algoritmu sú výborným základom pre rozšírenie o pokročilé metódy vyhľadávania ciest v rozsiahlych grafoch, ktoré boli zmienené v podsekcii 1.2.2. Aj keď za cenu ďalšieho zvýšenia veľkosti statických predspracovaných dát, umožnili by ďalšie niekoľkonásobné urýchlenie vyhľadávania a plynulé používanie aplikácie aj na oveľa väčších, pripadne hustejšie zmapovaných geografických územiach.

Ďalšou oblasťou, v ktorej by bolo vhodné spraviť optimalizácie, je správa operačnej pamäte používanej aplikáciou. V aktuálnej implementácii neboli okrem udržiavania minimálnej veľkosti grafových štruktúr robené žiadne iné opatrenia, a tak na úspešné fungovanie musí byť celá štruktúra prehľadávaného grafu načítaná v pamäti. Takéto riešenie je prijateľné pri trasovaní v menších oblastiach alebo pre nasadenie v serverovom prostredí pre pre dosiahnutie maximálnej rýchlosti spracovávania klientských požiadaviek, keďže veľkosť dostupnej operačnej pamäte je na serveroch rádovo niekoľkonásobne vyššia ako na bežne používaných prevažne mobilných zariadeniach. Pri cielení aplikácie na portabilné zariadenia by bolo vhodné vytvoriť mechanizmus konštantnej hornej hranice spotreby operačnej pamäte pri práci s príliš veľkými grafmi. Možným riešením by boli predspracované dáta uložené vo formáte podporujúcom náhodný prístup do súboru a počas vyhľadávania by boli priebežne načítavané menšie časti grafu.

Medzi funkcionalitu, ktorá je bežne poskytovaná inými trasovacími službami, je aj možnosť zvoliť viacero bodov, cez ktoré má vyhľadaná trasa prechádzať, prípadne poskytnutie slovných navigačných rád v textovej alebo zvukovej forme. Pri zameraní na offline mobilné trasovanie by bolo potrebné tiež zakomponovať vlastný vykresľovač mapových podkladov.

Záver

Úlohou tejto práce bolo zaoberať sa vyhľadávaním ciest vo vektorových mapových dátach, ktoré voľne poskytuje projekt OpenStreetMap, s cieľom vytvoriť aplikáciu, ktorej výstup bude dostatočne relevantný na využitie v praxi.

Prvým krokom pri riešení bolo zoznámenie sa s potrebnou teóriou a algoritmami využívanými v tejto oblasti. V rámci prípravy na návrh aplikácie som ďalej musel naštudovať dostupnú dokumentáciu projektu OpenStreetMap k obsahu poskytovaných dát, vyhľadať odpovede na nezdokumentované časti v rôznych diskusných skupinách, ktoré komunita tohoto projektu využíva na vzájomnú komunikáciu, a zvoliť vhodný prístup k výpočtu vzdialenosti medzi dvoma bodmi na Zemi. Súčasťou návrhu bolo aj vytvorenie účelných a pamäťovo nenáročných dátových štruktúr, s ktorými budú algoritmy pracovať. Okrem toho som predstavil vlastný formát určený na uchovanie užitočných dát potrebných na vyhľadávanie, ktorý umožňuje oveľa rýchlejšie načítanie a nezanedbateľne znižuje potrebné miesto na ich uloženie na pamäťovom médiu v porovnaní v pôvodným OSM formátom.

Výsledkom tejto práce je aplikácia, ktorá umožňuje používateľom vyhľadať trasu medzi dvoma zvolenými bodmi na mape, a to prostredníctvom grafického alebo konzolového rozhrania. Pri zostavovaní výslednej trasy umožňuje cieliť túto trasu na použitie pre chodca alebo pre jeden zo štandardne používaných typov dopravných prostriedkov, medzi ktoré patrí auto a bicykel. Medzi ďalšie možnosti parametrizácie vyhľadávania patrí voľba preferencie najkratšej vzdialenosti trasy alebo odhadovaného času prepravy. Aplikácia poskytuje na výber jeden zo štyroch rôznych implementovaných algoritmov, ktorými sú jednosmerná a obojsmerná varianta Dijkstrovho algoritmu a algoritmu A* s heuristikou vzdušnej vzdialenosti. Výsledok vyhľadávania je možné exportovať do jedného z troch formátov pre ďalšie využitie, napríklad v navigačných zariadeniach.

V prvej časti testovania som sa zameral na rýchlosť, s akou implementovaná aplikácia vyhľadáva trasy pri rôznych nastaveniach. Merania ukázali, že nielen voľba algoritmu ale aj zvolené parametre významne ovplyvňujú túto rýchlosť. Okrem zdôvodnenia príčin týchto javov ďalej vyplynulo, že už súčasná implementácia má pri vyhľadávaní trasy na úrovni mesta odozvu v rozsahu niekoľkých milisekúnd. Pri vyhľadávaní na rozsiahlejších územiach však jej rýchlosť klesá, a tak v rámci ďalšieho vývoja je vhodné zakomponovať pokročilejšie algoritmy, pre ktoré je už implementovaný Dijkstrov algoritmus výborným východiskom.

Porovnaním výstupov s konkurenčnými riešeniami som overil relevantnosť trás, ktoré implementovaná aplikácia vyhľadáva, a z výsledkov vyplynulo, že v niektorých prípadoch nájde vhodnejšie výsledky ako iné populárne a bežne používané vyhľadávače ciest. Ako sa ukázalo, vo všeobecnosti však neexistuje dokonalý výsledok a dôvody nájdených rozličných trás pre rovnako zvolené parametre vyhľadávania sú okrem rôznych zdrojov dát, z ktorých tieto riešenia vychádzajú, aj ich vlastné nastavenia a preferencie, s akými subjektívne ohodnocujú jednotlivé cestné úseky.

Literatúra

[1] Hliněný, P.: *Diskrétní matematika (456-533 DIM)*. Ostrava: FEI VŠB-TUO, 2005, s. 42–65.

[2] Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; aj.: *Introduction to Algorithms*. The MIT Press, třetí vydání, 2009, ISBN 0-262-03384-4, s. 651–664.

[3] Russell, S.; Norvig, P.: *Artificial Intelligence: A Modern Approach*. Prentice Hall, třetí vydání, 2010, ISBN 978-0-13-604259-4, s. 91–98.

[4] Korf, R. E.: Depth-first Iterative-Deepening: An Optimal Admissible Tree Search. *Artificial Intelligence*, č. 27, 1985: s. 97–109.

[5] PgRouting: PgRouting with Shooting-Star algorithm [online]. `http://www.pgrouting.org/docs/foss4g2008/ch09.html`, 2012, [rev. 2012-01-29].

[6] Patrushev, A.: Re: [Routing] Searching for ShootingStar -algorithm [online]. `http://www.mail-archive.com/routing@openstreetmap.org/msg00647.html`, 2009, [rev. 2012-01-29].

[7] Bast, H.; Funke, S.; Sanders, P.; aj.: Fast Routing in Road Networks with Transit Nodes. *Science*, č. 316, 2007: str. 566.

[8] Geisberger, R.; Sanders, P.; Schultes, D.; aj.: Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. In *Proceedings of the 7th Workshop on Experimental Algorithms (WEA'08)*, editace C. C. McGeoch, Lecture Notes in Computer Science, Springer, 2008, ISBN 978-3-540-68548-7, s. 319–333.

[9] Köhler, E.; Möhring, R. H.; Schilling, H.: Fast Point-to-Point Shortest Path Computations with Arc-Flags. In *9th DIMACS Implementation Challenge*, 2006.

[10] OpenStreetMap: OpenStreetMap Wiki [online]. `http://wiki.openstreetmap.org/`, 2012, [rev. 2012-01-25].

[11] OpenStreetMap Project [online]. `http://www.openstreetmap.org/`, 2012, [rev. 2012-01-05].

[12] Wikipedie, otevřená encyklopedie: Open Street Map [online]. `http://cs.wikipedia.org/wiki/OpenStreetMap`, 2012, [rev. 2012-01-25].

[13] Office of GEOINT Sciences: World Geodetic System 1984 (WGS 84) [online]. `http://earth-info.nga.mil/GandG/wgs84`, 2012, [rev. 2012-01-25].

[14] US National Imagery and Mapping Agency: *Department of Defense World Geodetic System 1984, Its Definition and Relationships With Local Geodetic Systems.* Washington: NIMA, třetí vydání, 2000, 175 s., NIMA TR8350.2.

[15] Sinnott, R. W.: Virtues of the Haversine. *Sky and Telescope*, č. 68, 1984: str. 159.

[16] Foster, D.: GPX: the GPS Exchange Format [online]. `http://www.topografix.com/gpx.asp`, 2012, [rev. 2012-04-21].

[17] Nokia Corporation: Qt [online]. `http://qt.nokia.com/`, 2012, [rev. 2012-01-25].

[18] Winter, K.: QMapControl [online]. `http://www.medieninf.de/qmapcontrol`, 2012, [rev. 2012-01-25].

[19] Browning, R. C.; Baker, E. A.; Herron, J. A.; aj.: Effects of obesity and sex on the energetic cost and preferred speed of walking. *Journal of Applied Physiology*, č. 100, 2006: s. 390–398.

[20] Brownell, D.; Megginson, D.: Events vs. Trees [online]. `http://www.saxproject.org/event.html`, 2012, [rev. 2012-04-24].

Zoznam príloh

Príloha A: Argumenty pri konzolovom použití aplikácie

Príloha B: Grafická príloha k analýze implementácie

Príloha C: Výsledky testov trasovacích aplikácií

- Bicykel
- Auto

Obsah CD

- Zdrojové kódy implementovanej aplikácie v adresári `/src/`
- Skompilovaná aplikácia pre operačné systémy Windows a Linux v adresári `/bin/`
- Mapové dáta použité pri testovaní v adresári `/maps/`
- Táto práca vo formáte PDF v adresári `/thesis/`
- Zdrojové kódy tejto práce vo formáte systému LaTeX v adresári `/thesis/latex/`

Príloha A

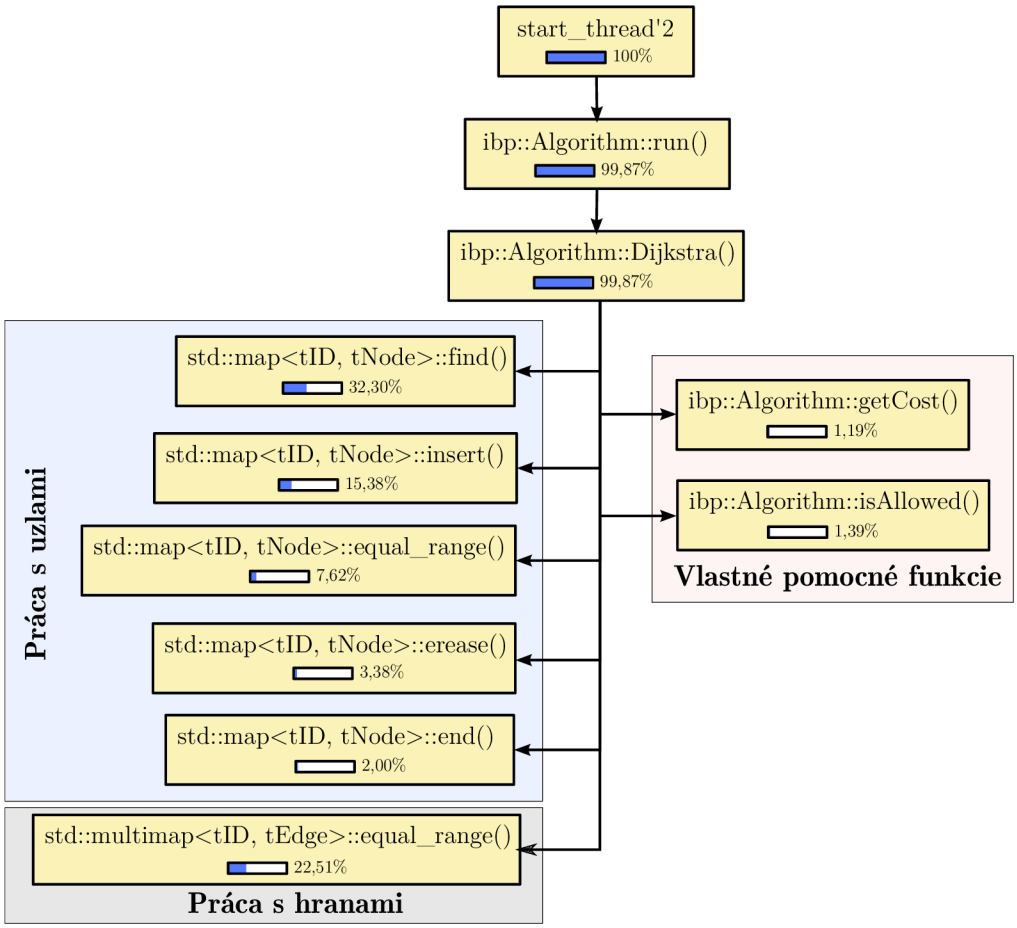
Argumenty pri konzolovom použití aplikácie

- `-h` , `--help` – zobrazenie nápovedy
- `--preproc` – načíta vstupný súbor a výstupom sú namiesto trasy predspracované mapové dáta
- `--input=cesta` – cesta k zdrojovému súboru, ktorú udáva *cesta*
- `--output=cesta` – cesta k cieľovému súboru, ktorú udáva *cesta*; pri vynechaní tejto možnosti je výpis na štandardný výstup
- `--outformat=` – jeden z nasledujúcich formátov výstupu; pri vynechaní je predvolený formát `GPX`
  - `gpx` – formát `GPX`
  - `osm` – zjednodušený formát `OSM`
  - `plaintext` – čistý text, v ktorom sú informácie oddelené medzerami
- `--srcpos=lat,lon` – začiatočná poloha, kde *lat* (geografická šírka) a *lon* (geografická dĺžka) sú reálne čísla
- `--destpos=lat,lon` – cieľová poloha, kde *lat* (geografická šírka) a *lon* (geografická dĺžka) sú reálne čísla
- `--algo=` – použitie jedného z nasledujúcich algoritmov:
  - `dijkstra` – Dijkstrov algoritmus
  - `bidijkstra` – obojsmerný Dijkstrov algoritmus (nie je optimálny)
  - `astar` – A* algoritmus s heuristikou vzdušnej vzdialenosti k cieľovému uzlu (nie je optimálny)
  - `biastar` – obojsmerný A* algoritmus s heuristikou vzdušnej vzdialenosti k cieľovému/začiatočnému uzlu (nie je optimálny)

- `--search=` – jeden z nasledujúcich parametrov vyhľadávania; pri vynechaní je predvolené vyhľadávanie najkratšej trasy
  - `shortest` – najkratšia trasa
  - `fastest` – najrýchlejšia trasa
- `--traveltype=` – použité typy dopravných prostriedkov, možné kombinovať čiarkami; pri vynechaní tejto možnosti sú použité všetky
  - `any` – ľubovoľný
  - `car` – auto
  - `bicycle` – bicykel
  - `foot` – chodec
  - `bicyclefoot` – chodec s bicyklom

Príloha B

Grafická príloha k analýze implementácie

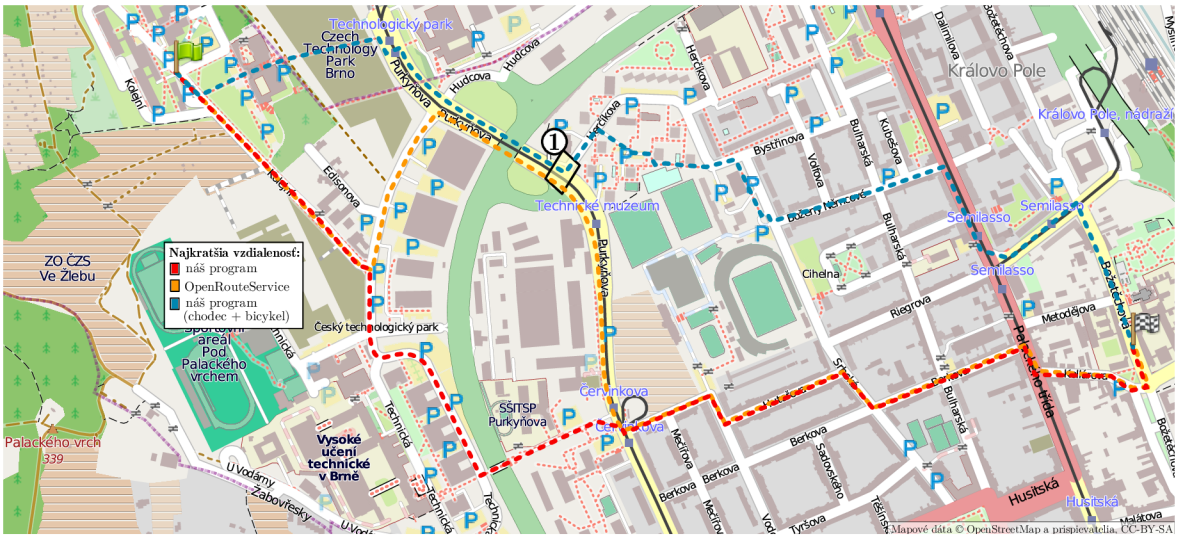


Obr. B.1: Grafická vizualizácia z programu Callgrind pre spotreby výpočtového času jednotlivých funkcií vo vlákne algoritmu našej aplikácie popísaná v podsekcii 5.1.4.

Príloha C

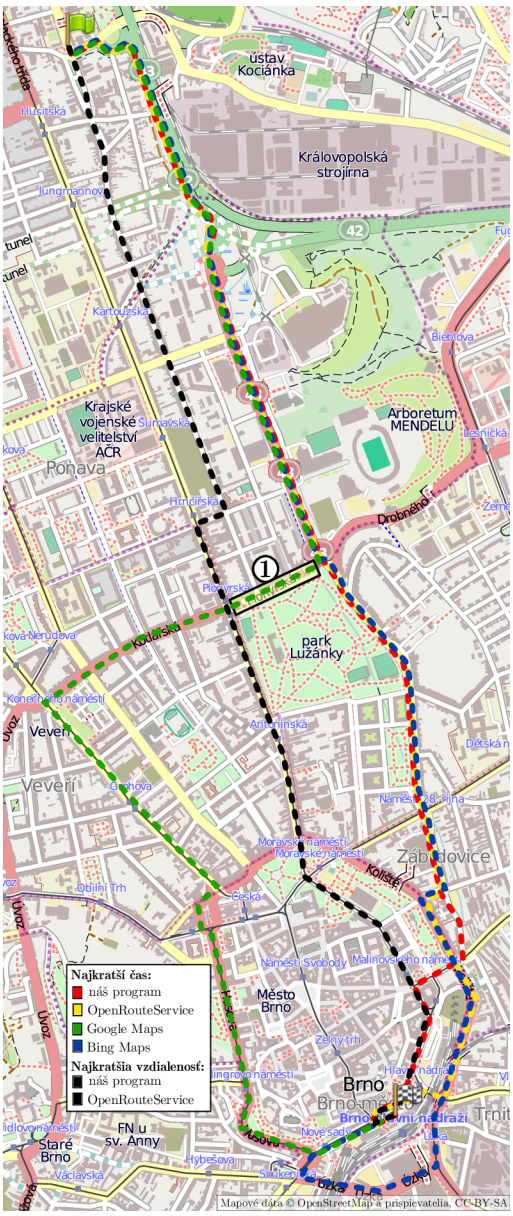
Výsledky testov trasovacích aplikácií

C.1 Bicykel



Obr. C.1: Výsledok testu popísaného v podsekcii 5.2.3 porovnávajúci nájdené trasy viacerých aplikácií pre bicykel.

C.2 Auto



Obr. C.2: Výsledok testu popísaného v podsekcii 5.2.2 porovnávajúci nájdené trasy viacerých aplikácií pre auto.