

## Kontext

---

Jedna z klasických aplikací teorie grafů je v implementaci mapově orientovaných aplikací. Ty jsou, zvláště v dnešní době, velmi žádanými doplňky aplikací hlavně v oblasti mobilních zařízení. Mnohé aplikace doplňují své služby o geografické informace a umožňují uživatelům zobrazovat a vyhledávat dle těchto dat. Z tohoto důvodu si v následujícím projektu vyzkoušíme tvorbu jednoduchého navigačního systému pomocí otevřeného mapového systému OpenStreetMap, který mnohé z mobilních aplikací také využívají.

Samotná realizace projektu bude simulovat klasický běh projektu v reálném prostředí (samozřejmě s nutnými zjednodušeními umožňující projekt během semestru stihnout a soustředit se na grafové problémy). K dispozici bude výchozí stav projektu, který nám ušetří prvotní programování a poskytne základní strukturu.

## Zadání

---

Úkolem tohoto projektu je vytvoření jednoduchého navigačního systému, pro který budeme využívat datových podkladů z OpenStreetMap. Jedná se o crowd-sourcingový mapový systém, jehož obsah vytvářejí přímo jeho uživatelé.

- [www.openstreetmap.org](http://www.openstreetmap.org)

Tento systém poskytuje komplexní mapové podklady, ze kterého využijeme síť ulic. Ulice jsou definovány jednotlivými body, které určují geografickou polohu každé ulice. Důležité je, že lze z tohoto systému jednotlivé mapy stahovat, viz

- [https://wiki.openstreetmap.org/wiki/Downloading\\_data](https://wiki.openstreetmap.org/wiki/Downloading_data)

Konkrétní mapu by bylo možné zjednodušeně reprezentovat grafem tak, že za vrcholy budou odpovídat křižovatkám (nebo slepým koncům ulic) a hrany budou reprezentovat ulice tyto křižovatky spojující. Tato reprezentace ovšem nezahrnuje cenné informace o topologii sítě – takto by například geografická vzdálenost křižovatek nemusela odpovídat celkové délce ulice z důvodu značně velkého zakřivení jejího běhu. Navíc při této reprezentaci není možné definovat přesněji cílové místo pro náš navigační systém.

Z tohoto důvodu jsou ulice v systému OpenStreetMap reprezentovány posloupností bodů vedoucí od jejího jednoho konce na druhý. Některé body mohou být pouze definicí zakřivení, jiné mohou být křižovatkami s jinými ulicemi, a některé mohou mít ještě další speciální vlastnosti. Příklad takové ulice je zobrazen na následujícím obrázku.



**Obrázek 1:** Příklad části ulice Domažlická poblíž Parukářky, spojující šest bodů, z nichž jsou čtyři zároveň křižovatkami s jinými ulicemi (reprezentované body 21311329, 25973241, 25973242, 25973237) a dva body jsou přechody pro chodce (1131753606 a 1131753673).

Odpovídající struktura v OSM formátu vypadá následovně.

```
1 <way id="39120883" visible="true" version="2" changeset="7160314" timestamp="2011-02-02T00:06:24Z" user=
2 <nd ref="21311329"/>
3 <nd ref="1131753606"/>
4 <nd ref="25973241"/>
5 <nd ref="25973242"/>
6 <nd ref="1131753673"/>
7 <nd ref="25973237"/>
8 <tag k="highway" v="residential"/>
9 <tag k="name" v="Domažlická"/>
10 <tag k="oneway" v="yes"/>
11 </way>
12 <node id="21311329" visible="true" version="2" changeset="766878" timestamp="2009-03-08T23:54:24Z" use
13 <node id="1131753606" visible="true" version="1" changeset="7160314" timestamp="2011-02-02T00:06:12Z"
14 <tag k="highway" v="crossing"/>
15 <tag k="source" v="bing:ortofoto"/>
16 </node>
17 <node id="25973241" visible="true" version="1" changeset="218259" timestamp="2007-02-17T23:29:05Z" use
18 <node id="25973242" visible="true" version="2" changeset="766878" timestamp="2009-03-08T23:54:24Z" use
19 <node id="1131753673" visible="true" version="1" changeset="7160314" timestamp="2011-02-02T00:06:13Z"
20 <tag k="highway" v="crossing"/>
21 <tag k="source" v="bing:ortofoto"/>
22 </node>
23 <node id="25973237" visible="true" version="2" changeset="7160314" timestamp="2011-02-02T00:06:15Z" us
```

V OpenStreetMap se vyskytují tři typy objektů, z nichž objekty typu area pro nás nejsou zajímavé, zbývající dva typy však ano:

- **Node** <nd>
  - geometrický bod
  - má ID jednoznačné v rámci bodů
  - poloha je definovaná dvojicí Latitude (lat=) a Longitude (lon=)
  - může obsahovat další doplňující informace
- **Way** <way>
  - seřazený seznam 2 a více objektů typu Node - pořadí odpovídá směru ulice (mezi po sobě jdoucími body je silnice)
  - v mapě reprezentuje jakoukoliv lomenou čáru

- obsahuje další doplňující informace, zejména jakého typu odpovídající silnice je či jaká na ní platí omezení

## Vstupní data

Datový formát používaný pro zasílání map je označován jako OSM a jedná se o XML formát. Základem je tag `<osm>`, který obsahuje základní informace o licenci a typu generátor

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <osm version="0.6" generator="CGImap 0.4.0 (10746 thorn-05.openstreetmap.org)"
3   copyright="OpenStreetMap and contributors" attribution="http://www.openstreetmap.org/copyright"
4   license="http://opendatacommons.org/licenses/odbl/1-0/">
5 </osm>
```

Pokud byla mapa, kterou zpracováváte, stáhnuta jako podoblast definována obdélníkem, bude datový zdroj obsahovat také následující tag definující tyto hranice pomocí minima a maxima zeměpisných šířek a délek:

```
1 <bounds minlat="50.0864200" minlon="14.4607100" maxlat="50.0888400" maxlon="14.4655600"/>
```

Základní stavební jednotkou, jak je již zmíněno výše, je objekt typu `Node`. Příklad jsme již viděli:

```
1 <node id="21311329" visible="true" version="2" changeset="766878" timestamp="2009-03-08T23:54:24Z"
2   user="BiIbo" uid="3516" lat="50.0874984" lon="14.4641165"/>
```

Tento tag obsahuje následující informace:

- `id` ... jednoznačný identifikátor vzhledem k vrcholům
- `visible` ... nastavení zobrazitelnosti na mapě
- `version` ... informaci o verzi dat
- `changeset` ... označení změn - založeno kvůli možnostem editace
- `timestamp` ... informaci o zanesení poslední informace
- `user` ... informaci o uživateli, který zanesl danou informaci
- `uid` ... číselné označení uživatele
- `lat` ... zeměpisná šířka daného bodu
- `lon` ... zeměpisná délka daného bodu

Kromě této struktury může bod obsahovat i složitější informace v komplexním tagu. Viděli jsme například přechod pro chodce

```
1 <node id="1131753606" visible="true" version="1" changeset="7160314" timestamp="2011-02-02T00:06:12Z" us
2 <tag k="highway" v="crossing"/>
3 <tag k="source" v="bing:ortofoto"/>
4 </node>
```

nebo jakékoli jiné bodové informace, například

```
1 <node id="290558370" visible="true" version="5" changeset="20938887" timestamp="2014-03-05T21:20:23Z"
2 user="Christoph Lotz" uid="47978" lat="49.8346891" lon="9.8906709">
3 <tag k="addr:city" v="Veitshöchheim"/>
4 <tag k="addr:country" v="DE"/>
5 <tag k="addr:housenumber" v="3"/>
6 <tag k="addr:postcode" v="97209"/>
7 <tag k="addr:street" v="Danziger Straße"/>
8 <tag k="amenity" v="bank"/>
9 <tag k="atm" v="yes"/>
10 <tag k="email" v="VeitshoechheimII@Sparkasse-Mainfranken.de"/>
11 <tag k="fax" v="+49 (0)931 / 382-2780"/>
12 <tag k="name" v="Sparkasse Mainfranken"/>
13 <tag k="phone" v="+49 (0)931 / 382-0"/>
14 <tag k="website" v="https://www.sparkasse-mainfranken.de/module/ihre_sparkasse/filialfinder/index.php
15 ?n=%2Fmodule%2Fihre_sparkasse%2Ffilialfinder%2F#details/13093"/>
16 <tag k="wheelchair" v="limited"/>
17 </node>
```

Jak vidíme, tak tyto informace se týkají hlavně firem či zajímavých míst umístěných na tomto místě potažmo adrese.

Dále obsahuje mapa informace o samotných ulicích pomocí elementu [Way](#) reprezentovanému tagem <way>, jak už jsme viděli s ulicí Domažlická:

```
1 <way id="39120883" visible="true" version="2" changeset="7160314" timestamp="2011-02-02T00:06:24Z" user
2 <nd ref="21311329"/>
3 <nd ref="1131753606"/>
4 <nd ref="25973241"/>
5 <nd ref="25973242"/>
6 <nd ref="1131753673"/>
7 <nd ref="25973237"/>
8 <tag k="highway" v="residential"/>
9 <tag k="maxspeed" v="50"/>
10 <tag k="name" v="Domažlická"/>
11 <tag k="oneway" v="yes"/>
12 </way>
```



Ulice tak obsahuje jednak základní informace zahrnující

- `id` ... unikátní ID vzhledem k jednotlivým tagům `way`
- `visible` ... informaci o viditelnosti
- `changeset` ... označení změnové řady
- `timestamp` ... informaci o datumu změny
- `user` ... jméno uživatele
- `uid` ... id uživatele

Poté obsahuje každá ulice informaci o posloupnosti jednotlivých nodů v tagu `<nd>`. Atribut `ref` odkazuje na id specifického Node, který je součástí ulice.

Ulice výše se tak skládá ze 6 nodů (dvou počátečních a čtyř, které jsou součástí dané ulice). Následně jsou na ulici různé typy dalších vlastností jako sekvence tagů pojmenovaných `<tag>`. Výše zmíněné jsou

- `highway` -- jedná se o silnici nebo cestu  
**Pozor:** tag `Highway` může mít několik verzí - nahraďte pouze ty, které dávají smysl
- `residential` -- ulice v obydlené části
- maximální rychlost
- jméno dané ulice
- informace, že je ulice jednosměrná (pokud není, tento tag chybí)

Je potřeba dát pozor, že ne vše uložené v systému jako `way` je ve skutečnosti ulicí. Toto označují různé tagy jako `amenity`, `notroad` či `building`. Pro správné načtení sítě ulic prostudujte dokumentaci OpenStreetMap a zkuste si část mapy Prahy stáhnout (pozor, XML soubor větší oblasti je velmi rozsáhlý).

## Úkol

Úkolem tohoto projektu je vytvořit jednoduchou navigaci, která načte data ze systému OpenStreetMap, bude umět vyhledat nejkratší cestu mezi cílovými body a výsledek zobrazit v systému **Graphviz**. Samotné zpracování je rozděleno na dvě odevzdání, ve kterých budeme zpřesňovat konečnou podobu aplikace. K dispozici je šablona projektu pro jazyk Ruby, která se bude v rámci řešení rozšiřovat.

## A Struktura šablony projektu

---

Programovacím jazykem je jazyk Ruby. Pro lepší představu o funkčnosti používá projekt export načítaných a zpracovaných grafů do formátu GraphViz. K tomu je potřeba instalace tohoto balíku do systému, viz

- [Graphviz download](#) ... stránky obsahují návod pro Windows/Linux/Mac

V rámci jazyka Ruby není k realizaci využít žádný grafově teoretický balík a ani to není v konečném odevzdání povoleno. Knihovny potřebné pro základní běh projektu jsou

- [nokogiri](#) ... knihovna pro načítání XML souborů potřebná pro načtení OSM dat.
- [ruby-graphviz](#) ... knihovna pro export grafů do formátu graphviz

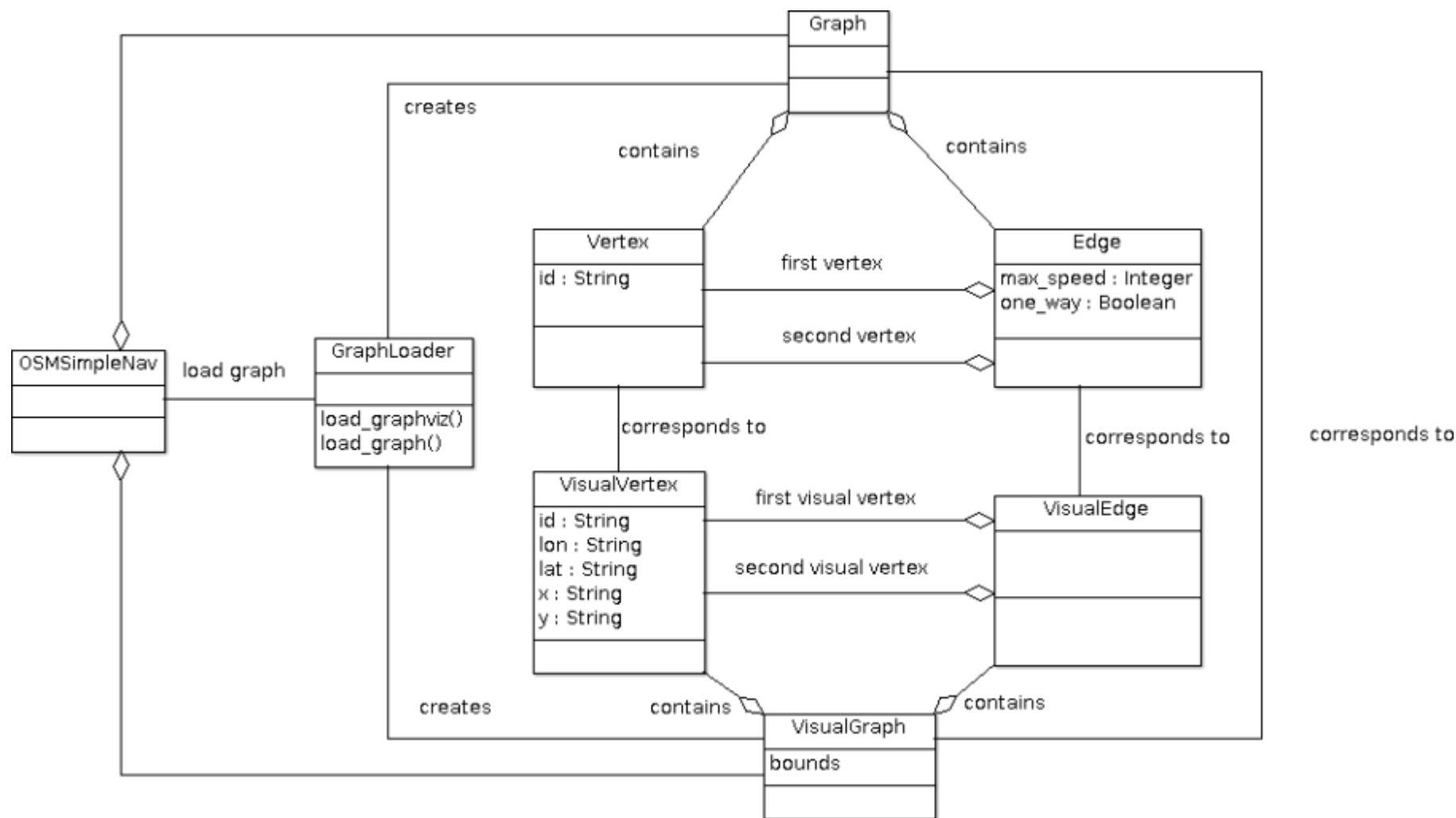
K dispozici je šablona pro odevzdání projektu, kterou dubou jednotlivá odevzdání rozšiřovat, viz příloha

-  [gal\\_template\\_project.zip](#)

### A.1 Struktura projektu

---

Projekt má jednoduchou strukturu tříd oddělující realizační a vizualizační část problému. Nahrubo nastíněný UML diagram projekt se nachází na následujícím obrázku.



Jak je zřejmé, tak základní třídou je třída `OSMSimpleNav`. Ta bude spravovat i příkazový řádek, který bude hlavním UI aplikace. Tato třída načte graf pomocí třídy `GraphLoader` a vytvoří dvě jeho reprezentace.

1. Samotný graf reprezentovaný třídou `Graph` obsahující vrcholy (třída `Vertex`) a hrany (třída `Edge`).
2. Třída `VisualGraph` reprezentuje grafické zobrazení samotného grafu. Obsahuje vizuální body vrcholů (třída `VisualVertex`) a Vizuální hrany (třída `VisualEdge`). Instance těchto tříd odpovídají objektům vrcholů a hran samotného grafu (třída `Graph`)

Zajímavé je dvojí uložení souřadnic v instancích třídy `VisualVertex`. Důvodem je přepočítání souřadnic pro `graphviz` a potřeba původních Latitude a Longitude při výpočtech.



## A.2 Uživatelské rozhraní projektu

Výchozí třídou je třída `OSMSimpleNav`, která realizuje ve své metodě `run()` zpracuje parametry příkazového řádku a dle jejich hodnot spustí dané příkazy. Příkazy jsou následující

- Načtení mapy a její export do jiného formátu

```
ruby osm_simple_nav.rb --load <input_map.IN> --export <exported_map.OUT>
```

zpracování probíhá na základě hodnot přípon souborů `IN` a `OUT`. Je předpokládáno následující

Přípona	Povolený výskyt	Význam
osm	IN	XML formát map z projektu OpenStreetMap
dot   gv	IO/OUT	Graphviz formát grafu
pdf	OUT	Výstup mapy do formátu PDF
png	OUT	Výstup mapy do formátu PNG

**Důležité upozornění:** Načítání mapy přes formát Graphviz je pouze kontrolní a slouží k prvotnímu ladění úkolu načtení mapy přes formát OSM.

## A.3 Načtení map ze souborů

Načtení grafu a jeho grafické reprezentace pomocí metod:

1. `load_graph()` načítající mapu přímo z formátu OSM a
2. `load_graphviz()` načítající mapu z formátu graphviz (předpokládáme příponu `dot` a navíc se předpokládá, že soubory graphviz obsahují všechny potřebné informace).

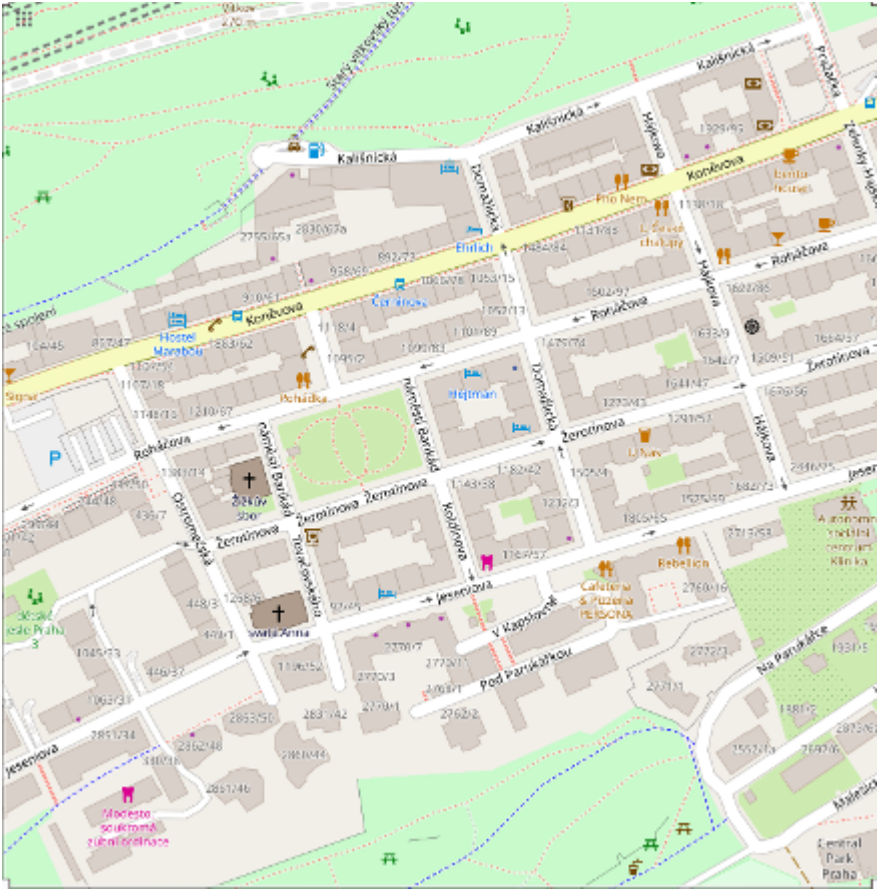
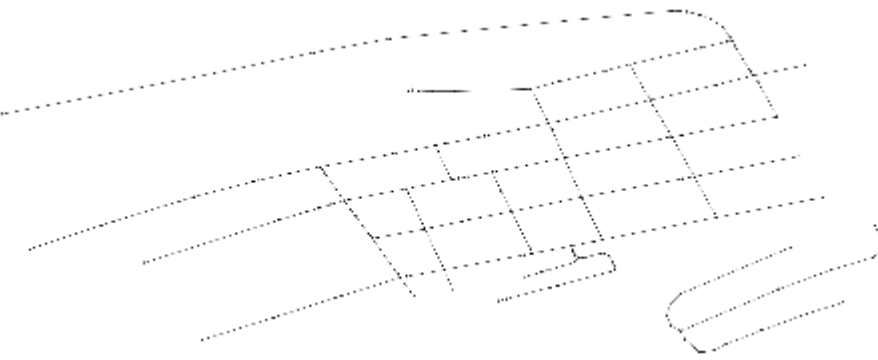
## A.4 Realizace exportu map

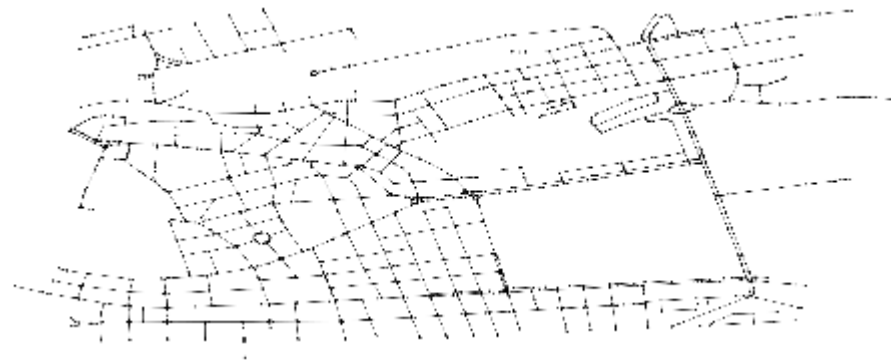
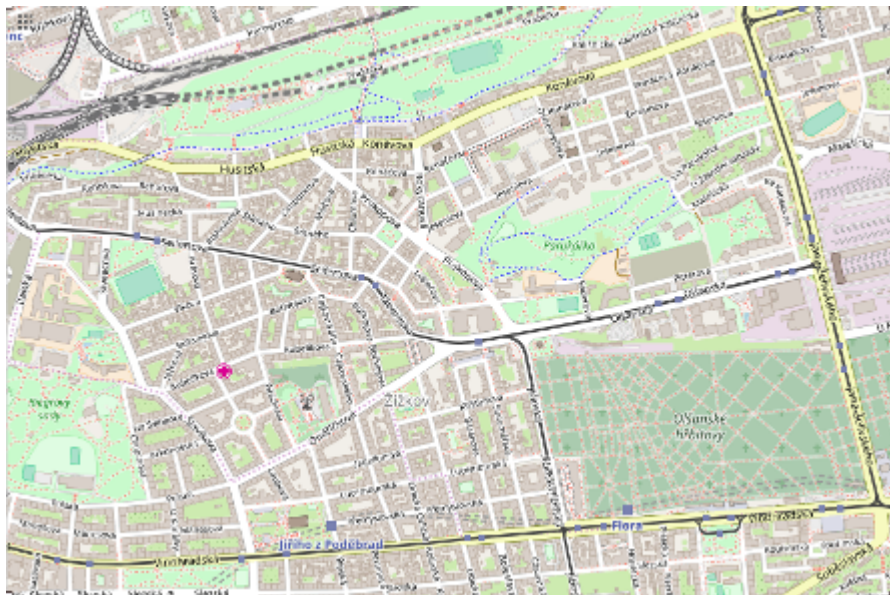
K exportu map se využívá přímo třídy `VisualGraph` skrze výše zmiňovanou knihovnu `ruby-graphviz`. Třída `VisualGraph` obsahuje metodu:

- `export_graphviz(export_filename)` ... která realizuje export celé mapy do formátu `graphviz`. Je snahou do exportovaného souboru uložit i informace, které nejsou přímo potřeba pro zobrazení (například povolenou rychlost), aby bylo případně možné tohoto souboru použít i jako vstup do aplikace (v konečném odevzdání toto není plně nutné).

## A.5 Ukázka výstupů

Pro ukázkou máme tři části Prahy, jejichž soubory jsou uvedeny i v ukázkovém projektu.

Obrázek z OSM	Vygenerovaný Graphviz
	





Detaily transformace jsou ukázány v projektu.

**POZOR:** Transformace není dělána na libovolné velikosti mapy. Pokud byste chtěli použít libovolnou velikost, potom je nutné pracovat s proměnnou scale jinak.

## B Úkoly k samostatné realizaci a odevzdání

Odevzdávejte jako zdrojové soubory v jazyce Ruby. Všechny potřebné soubory umístěte do adresáře `prijmeni_jmeno_N`, kde N označuje fázi odevzdávání (1 nebo 2). Tento adresář celý zabalte do souboru `prijmeni_jmeno_N.zip` nebo `prijmeni_jmeno_N.7z` a připojujte je postupně jako přílohy k artefaktu s vaším úkolem. Pokud budete používat jakoukoliv knihovnu mimo standardní, musíte ji připojit k odevzdání a popsat její instalaci. Jako **název odevzdávaného úkolu** vyplňte GAL DU1 či GAL DU2. (viz info níže).



Jenodlité kroky odpovídají dvěma postupným odevzdáním projektu. Úkoly jednotlivých kontrolních bodů jsou následující.

## B.1 První úkol

Úkolem bude **načíst soubor s mapou** ve formátu OSM (dokončit realizaci metody `load_graph`), vyfiltrovat pouze hrany potřebné pro konstrukci uliční sítě (tudiž například ne okraje budov), viz

- **Přehled hodnot** parametru highway v Open Street Map, viz pole všech možných hodnot, ze kterých budeme vybírat

```
1 | @highway_attributes = ['residential', 'motorway', 'trunk', 'primary', 'secondary', 'tertiary', 'unclassified']
```

**Zkonstruovat neorientovaný graf** reprezentující danou síť (prozatím budeme ignorovat možnost jednoho směru). Uložte také informaci o délce každého segmentu ulice (pozor, jedná se o geografickou vzdálenost) a povolené rychlosti – pokud informace o povolené rychlosti nebude dostupná, pak ji nastavte jako 50 km/h. Pozor, že OpenStreetMap ukládá cesty typu way, ale my chceme graf a tedy pouze hrany obsahující dva body - je tedy nutné načtenou strukturu way převést na jednotlivé hrany.

Kontrola načtení může být provedena pomocí vygenerování grafického výstupu existující metodou `export_graphviz()`. Pokud při generování metody nastanou komplikace, pak je možné metodu upravit dle potřeb, ale pouze tak, aby fungovala i v jiných případech.

Úkolem pro samotný graf je vyfiltrování největší **maximální komponenty grafu**. V podstatě to znamená, že vezmete jen takovou část grafu, která je souvislá a ze všech takovýchto částí tu největší. Pro další zpracování potom budete využívat již jen tento podgraf (ostatní vrcholy a hrany lze tedy zahodit).

Program se bude spouštět následujícím způsobem:

```
ruby osm simple nav.rb --load-comp <input map.IN> --export <exported map.OUT>
```

Významy parametrů souborů jsou stejné jako v ukázkovém chování podaném v sekci A.2. Pouze první přepínač načítání označuje, že máme vybrat pouze souvislý graf. Pro kontrolu je možné použít původní generování a zobrazit si původní graf.

## B.2 Druhý úkol

V druhém odevzdání bude program realizovat samotnou navigaci. K určení nejkratší cesty poutřebujeme nejprve znát jednotlivé body, abychom mohli vybrat cíl a start bude nejprve nutné vypsát jednotlivé body v mapě

```
ruby osm_simple_nav.rb --load-comp <input_map.IN> --show-nodes
```

Druhý přepínač aktivity (a zároveň absence výstupního souboru) říká, že program má na obrazovku vypsát vrcholy sítě s uvedením ID a jeho souřadnic latitude a longitude. Tedy pokud máme postupně vrcholy vrchol\_1, vrchol\_2, ..., potom získáme následující výpis:

```
id_vrcholu_1 : lat_1, lon_1
```

```
id_vrcholu_2 : lat_1, lon_2
```

```
...
```

tedy například

```
247743: 50.0792640, 14.4300586
```

```
247745: 50.0795287, 14.4226679
```

```
...
```

Z tohoto seznamu si budeme moci přímo vybrat startovní a cílové body, mezi kterými budeme hledat nejkratší cestu. Ke kontrole bude sloužit speciální spuštění programu, které na mapě zvýrazní body, mezi kterými chceme počítat nejratčí cestu.

```
ruby osm_simple_nav.rb --load-comp <input_map.IN> --show-nodes <id_start> <id_stop> <exported_map.OUT>
```

kde hodnoty <id\_start> a <id\_stop> budou představovat id vrcholů startu a cíle. Alternativně bude možné také zadat hodnoty latitude a longitude a vykreslí se nejbližší body ze všech bodů mapy. Tuto možnost spustíme následovně.

```
ruby osm2graphviz.rb --load-comp <input_map.IN> --show-nodes <lat_start> <lon_start> <lat_stop> <lon_stop>  
<exported_map.OUT>
```

POZN: Zakreslení bodů proveďte vhodným způsobem, aby jej bylo možné dobře pozorovat.












Program bude mít dále schopnost určit pro dvě zadaná místa nejkratší cestu, pro níž také vypíše informaci o době, za jakou vozidlo cestu ujede (za použití maximální povolené rychlosti). Navíc je úkolem tuto cestu zakreslit do mapy odlišnou barvou a tloušťkou čáry a tak zvýraznit její průběh ve výstupu. Program bude mít následující rozhraní:

```
ruby osm2graphviz.rb --load-comp <input_map.IN> --midist<lat_start> <lon_start> <lat_stop> <lon_stop>  
<exported_map.OUT>
```

Výstupem bude tedy mapa, na které budou jednotlivé úseky nejkratší cesty vhodně zvýrazněny - například barvou a tloušťkou čáry.

## Potřebé zdroje

---

-   Základy objektově orientovaného programování
-   Formát pro výměnu dat XML
-   Grafové algoritmy
-   GAL Graf a jeho reprezentace
-   GAL Průchod grafem
-   GAL Hledání minimální kostry
- [www.openstreetmap.org](http://www.openstreetmap.org)
- Wiki stránky projektu  
[wiki.openstreetmap.org](http://wiki.openstreetmap.org)
- OSM XML format  
[wiki.openstreetmap.org/wiki/OSM\\_XML](http://wiki.openstreetmap.org/wiki/OSM_XML)
- [www.graphviz.org](http://www.graphviz.org)
- Příklad souboru pro Graphviz (pro příkaz neato) je v příloze tohoto artefaktu.
- Překlad zdrojového souboru do PDF se provede příkazem  
  
`neato -Tpdf -o VYSTUPNI_SOUBOR.pdf VSTUPNI_SOUBOR.gv`
- Tutorial  
[graphs.grevian.org](http://graphs.grevian.org)