Univerzita Karlova

Přírodovědecká fakulta

GEOINFORMATIKA

Nejkratší cesta grafem

Kristýna Antošová, Eliška Pospěchová

1. ročník N-GKDPZ

Praha, 08.01.2025

Úloha 3 : Nejkratší cesta grafem

Implementujte Dijkstra algoritmus pro nalezení nejkratší cesty mezi dvěma uzly grafu. Vstupní data budou představována silniční sítí doplněnou vybranými sídly.

Otestujte různé varianty volby ohodnocení hran grafu tak, aby nalezená cesta měla:

- nejkratší Eukleidovskou vzdálenost,
- nejmenší transportní čas.

Ve vybraném GIS konvertujte podkladová data do grafové reprezentace představované neorientovaným grafem. Pro druhou variantu optimální cesty navrhněte vhodnou metriku, která zohledňuje rozdílnou dobu jízdy na různých typech komunikací. Výsledky (dvě různé cesty pro každou variantu) umístěte do tabulky, vlastní cesty vizualizujte. Dosažené výsledky porovnejte s vybraným navigačním SW.

Krok	**Hodnocení**
Dijsktra algoritmus.	20b
Řešení úlohy pro grafy se záporným ohodnocením.	*+10b*
Nalezení nejkratších cest mezi všemi dvojicemi uzlů.	*+10b*
Nalezení minimální kostry některou z metod.	*+15b*
Využití heuristiky Weighted Union	*+5b*
Využití heuristiky Path Compression	*+5b*
Max celkem:	**65b**

Obsah

1. Úvod

Cílem úlohy je nalezení nejméně nákladné cesty mezi dvěma uzly grafu, ve kterém jsou jednotlivé uzly provázány silniční sítí. Řešení je zpracováno použitím jazyka Python v prostředí Visual Studio Code. Nejprve bude použit Dijkstra algoritmus, který je momentálně nejpoužívanějším algoritmem pro hledání cesty z bodu A do bodu B. Naším cílem je nalézt jak nejkratší cestu, tak nejrychlejší cestu, která bude kromě maximálních rychlostí daných komunikací také zohledňovat jejich klikatost. Následně bude také nalezena nejkratší vzdálenost mezi všemi dvojicemi uzlů.

V rámci práce bude také řešena problematika nalezení minimální kostry grafu. Bylo vynalezeno již mnoho algoritmů řešící tuto úlohu, od Jarníkova algoritmu po Borůvkův algoritmus, který byl poprvé použit ve 20. letech minulého století pro minimalizaci nákladů energetické sítě v Česko-Slovensku. Naše řešení využívá Kruskalův algoritmus, který je modifikací Borůvkova algoritmu. Řešení bude doplněno o dvě heuristiky, které zlepší jeho efektivitu – Path Compression a Weighted Union.

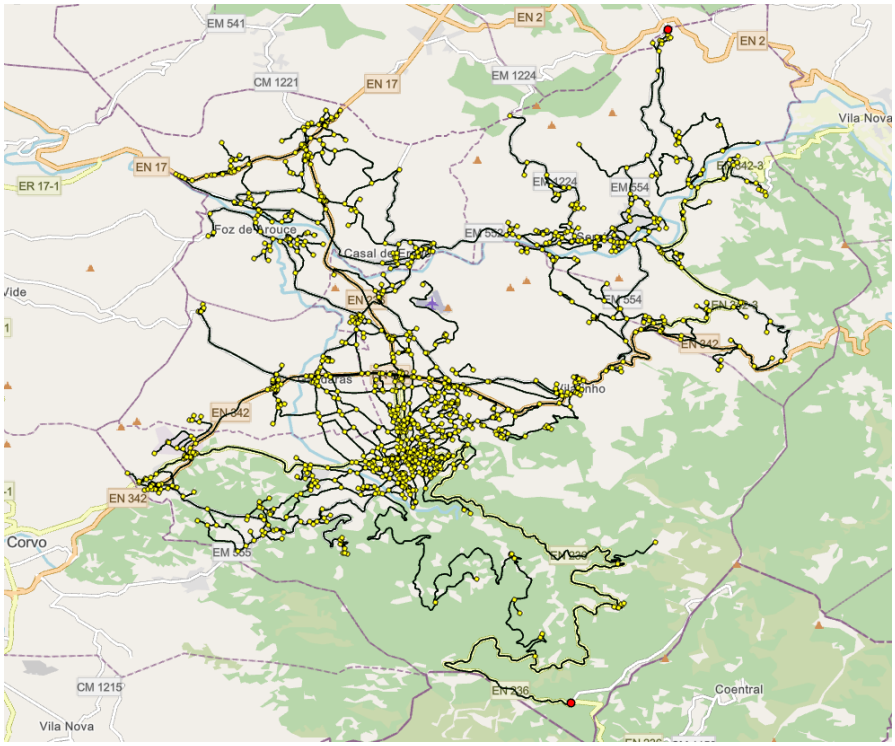
2. Zpracování

2.1. Nalezení nejkratší cesty

Pro úlohy byla použita knihovna *osmnx*, ve které jsou dostupná OpenStreetMap data již ve formátu grafu. Představují tak vhodná prostorová data pro naše analýzy. Hrany grafu jsou silniční sítí dané oblast a uzly grafu představují veškeré křižovatky. Jako zájmové území byla vybrána oblast portugalského města Lousã. Souřadnice počátečního a koncového bodu jsou popsány v tab. 1 a vizualizovány spolu s hranami grafu na obr. 1. Graf obsahuje 1234 uzlů a 2880 hran. Výsledné cesty byly vizualizovány s využitím knihovny *osmnx.*

Tab. 1: Souřadnice počátečního a koncového bodu grafu

	OSM ID	X	Y
Počáteční bod	252865248	-8,2133135	40,0608069
Koncový bod	1601656647	-8,1933466	40,199814



Obr. 1: Použitý graf (červeně koncový a počáteční bod)

Graf byl načten pomocí funkce *ox.graph_from_place* a vybrány byly silnice *network_type='drive'*. Orientovaný graf byl převeden na neorientovaný a data byla konvertována do následujícího formátu slovníků:

$G = \{V_1 : \{V_1:W_1, V_2:W_2, ..., V_k:W_k\},$
$V_2 : \{V_1:W_1, V_2:W_2, ..., V_k:W_k\},$
...
$V_k : \{V_1:W_1, V_2:W_2, ..., V_k:W_k\}\}$

kde G je graf, V jsou jednotlivé uzly a W váhy mezi nimi.

2.1.1. Dijkstra algoritmus

Algoritmus je určen k nalezení nejkratší cesty grafem, jehož hrany jsou ohodnoceny pouze kladnými hodnotami. Jedná se o strategii prohledávání grafu do šířky k čemuž je použita prioritní fronta. V průběhu iterativní fáze postupně dochází k zpřesňování odhadu nejkratší délky cesty od počátečního ke koncovému bodu. K tomu Dijkstra využívá princip relaxace hrany.

Princip algoritmu Dijkstra:

1. **Inicializace**:
   - Vytvoření prioritní fronty uzlů – uzly řazeny od aktuálně nejkratší vzdálenosti
   - Vytvoření slovníku vzdáleností
     - Iniciace všech vzdáleností na nekonečno
     - Nastavení výchozího uzlu na vzdálenost 0
   - Vytvoření slovníků předchůdců uzlů a nastavení na " none "
   - Vytvoření množiny navštívených uzlů
2. **Hlavní while cyklus**
   - Dokud není fronta prázdná:
     - Pro doposud nenavštívené uzly:
       - Vybrání nejbližšího uzlu
       - Označení uzlu jako navštívený
       - Relaxace – výpočet vzdáleností k sousedním uzlům s použitím ohodnocení hran
         - Pokud je tato vzdálenost kratší než známá – aktualizace
         - Aktualizace předchůdce
         - Přidání sousedního uzlu do prioritní fronty
3. **Ukončení iterativní fáze**
   - Pokud je prioritní fronta prázdná (neexistuje tedy žádný otevřený uzel) je iterativní fáze ukončena
   - Vrácení konečných vzdáleností a předchůdců

Ohodnocení hran

1. **Nejkratší trasa**

   Váhou je délka hrany – tedy délka silnice v m získaná z atributu OSM dat *length*.

2. **Nejrychlejší trasa**

V tomto případě byla jako metrika zvolena maximální povolená rychlost. Ta je u některých hran definovaná ve vstupních datech atributem *maxspeed*, který je řetězcem, proto musela být provedena konverze do číselné hodnoty a následně proběhlo převedení na m/s. Pokud hodnota tohoto atributu u určité hrany chybí, je pro definování minimální rychlosti použit atribut *highway,* ve kterém je uveden typ komunikace. Pro každou kategorii byl zvolen rychlostí limit podle pravidel silničního provozu Portugalska a uložen do slovníku *DEFAULT_MAXSPEED.*

Při výpočtu transportního času byla zohledněna klikatost silniční sítě. Byl zaveden index klikatosti, a to následovně:

sinusoidality = shapely_length / euclidean_distance

kde *shapely_length je* vzdálenost získaná z geometrie OSM vrstvy, euklidovská vzdálenost byla vypočtena pomocí funkce *distance*. *shapely_length* byla využita, protože má jednotky stupně, stejně tak jako euklidovská vzdálenost. Obě proměnné tedy vychází ze souřadnic WGS. Vzdálenost opravena o klikatost byla vypočtena jako:

enhanced_distance = sinusoidality * length_attr

Protože *enhanced_distance* bude v následujícím kroku vstupovat do výpočtu času (proměnné time), chceme ji v metrech. Proto byla využita *length_attr,* což je délka silnic v m získaná atributu OSM dat *length*. Výsledný čas byl definován v sekundách jako:

time = enhanced_distance / max_speed

V řešení úlohy byl vypočítán také transportní čas bez zohlednění klikatosti.

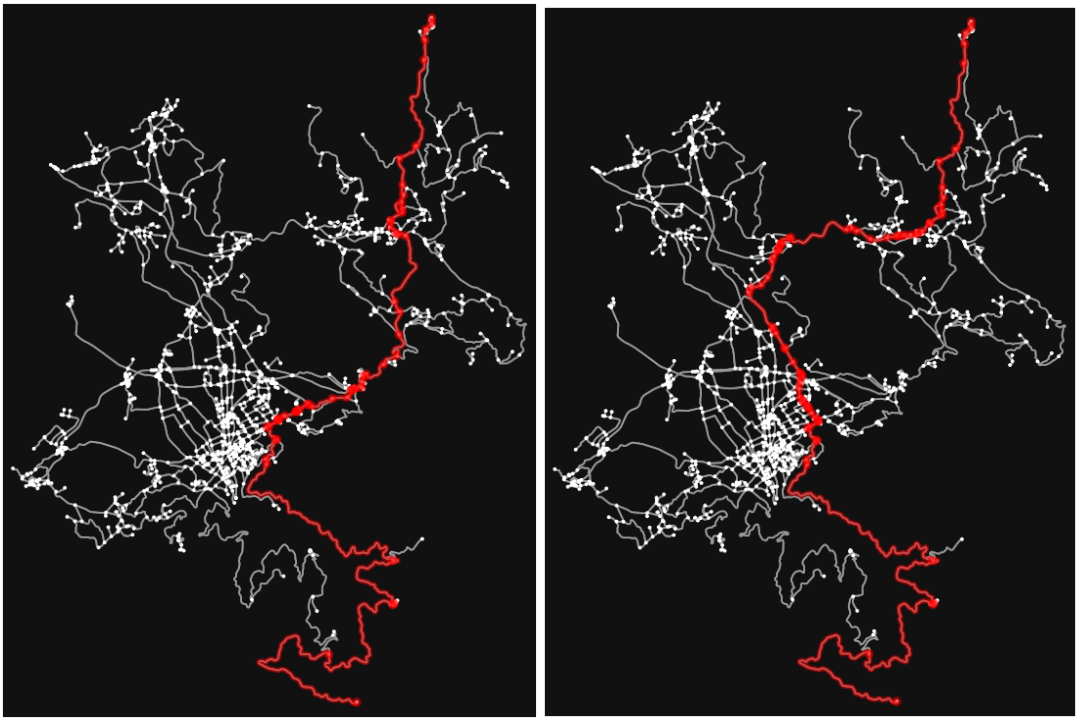
Výsledky

Na obr. 2 je červenou linií vyobrazena nejkratší (vlevo) a nejrychlejší (vpravo) cesta grafem. Ze stejného startovního do koncového bodu byl zadán výpočet trasy na stránkách Mapy.cz a Google Maps. Algoritmus ve dvou použitých verzích ohodnocení hran použije geograficky odlišné cesty grafem. Námi vytvořené cesty se začnou lišit přibližně v polovině cesty, kde se nejkratší trasa stává více klikatá, a tak zde algoritmus zohledňující klikatost volí sice delší ale ne tolik klikatou západnější trasu. Nejkratší trasa vygenerovaná Dijkstra algoritmem je graficky poměrně podobná trasám z navigačních systémů. Je o 1,6 km kratší než trasa Map.cz a o 2,5 km delší než trasa vytvořená Google Maps. Jak můžeme pozorovat na obr. 3. výsledky ze stránek Mapy.cz a Google Maps jsou prostorově identické, systémy však kvůli použití různých metodik výpočtu času trasy hlásí různé doby průjezdu danou trasou – časy se liší o 21 minut – viz tab. 2. Námi získaný výsledek délky nejrychlejší trasy je přibližně průměrem časů tras z navigačních systémů. Do výpočtů navigačních systémů totiž vstupují i jiné parametry než pouze maximální povolená rychlost a klikatost. Když do výpočtu transportního času nebyla zakomponována klikatost, transportní čas vyšel 22 min. a trasa

využila stejné komunikace jako trasa s využitím klikatosti. Klikatost tedy dělá výslednou nejkratší cestu přibližně 2x delší.

Tab. 2: Porovnání cest s trasami navigačních systémů

	Vzdálenost (km)	Čas (min)
Nejkratší	32,2	---
Nejrychlejší bez klikatosti	---	22
Nejrychlejší s klikatostí	---	45
Mapy cz	33,8	33
Mapy Google	34,7	54

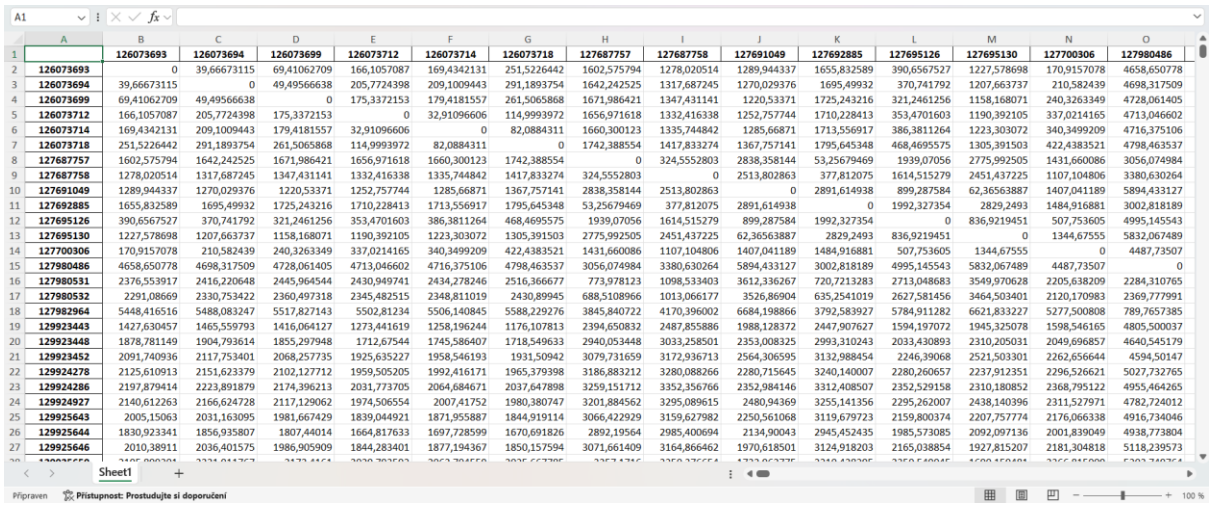


Obr 2: Nejkratší cesta vlevo, nejrychlejší cesta vpravo

Obr 3.: Trasy navigačních systémů

2.1.2. Nalezení nejkratších cest mezi všemi dvojicemi uzlů

Pro nalezení všech vzdáleností byla použita funkce *all_distances*. Funkce využívá algoritmus Dijkstra pro hledání nejkratších cest. Dijkstra algoritmus je spuštěn pro každý startovní uzel a do matice se uloží vzdálenosti od všech koncových bodů. Matice je následně konvertována to tabulky a exportována do souboru XLS (viz obr 4).



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1		126073693	126073694	126073699	126073712	126073714	126073718	127687757	127687758	127691049	127692885	127695126	127695130	127700306	127980486
2	126073693	0	39,66673115	69,41062709	166,1057087	169,4342131	251,5226442	1602,575794	1278,020514	1289,944337	1655,832589	390,6567527	1227,578698	170,9157078	4658,650778
3	126073694	39,66673115	0	49,49566638	205,7724398	209,1009443	291,1893754	1642,242525	1317,687245	1270,029376	1695,49932	370,741792	1207,663737	210,582439	4698,317509
4	126073699	69,41062709	49,49566638	0	175,3372153	179,4181557	261,5065868	1671,986421	1347,431141	1220,53371	1725,243216	321,2461256	1158,168071	240,3263349	4728,061405
5	126073712	166,1057087	205,7724398	175,3372153	0	32,91096606	114,9993972	1656,971618	1332,416338	1252,757744	1710,228413	353,4701603	1190,392105	337,0214165	4713,046602
6	126073714	169,4342131	209,1009443	179,4181557	32,91096606	0	82,0884311	1660,300123	1335,744842	1285,66871	1713,556917	386,3811264	1223,303072	340,3499209	4716,375106
7	126073718	251,5226442	291,1893754	261,5065868	114,9993972	82,0884311	0	1742,388554	1417,833274	1367,757141	1795,645348	468,4695575	1305,391503	422,4383521	4798,463537
8	127687757	1602,575794	1642,242525	1671,986421	1656,971618	1660,300123	1742,388554	0	324,5552803	2838,358144	53,25679469	1939,07056	2775,992505	1431,660086	3056,074984
9	127687758	1278,020514	1317,687245	1347,431141	1332,416338	1335,744842	1417,833274	324,5552803	0	2513,802863	377,812075	1614,515279	2451,437225	1107,104806	3380,630264
10	127691049	1289,944337	1270,029376	1220,53371	1252,757744	1285,66871	1367,757141	2838,358144	2513,802863	0	2891,614938	899,287584	62,36563887	1407,041189	5894,433127
11	127692885	1655,832589	1695,49932	1725,243216	1710,228413	1713,556917	1795,645348	53,25679469	377,812075	2891,614938	0	1992,327354	2829,2493	1484,916881	3002,818189
12	127695126	390,6567527	370,741792	321,2461256	353,4701603	386,3811264	468,4695575	1939,07056	1614,515279	899,287584	1992,327354	0	836,9219451	507,753605	4995,145543
13	127695130	1227,578698	1207,663737	1158,168071	1190,392105	1223,303072	1305,391503	2775,992505	2451,437225	62,36563887	2829,2493	836,9219451	0	1344,67555	5832,067489
14	127700306	170,9157078	210,582439	240,3263349	337,0214165	340,3499209	422,4383521	1431,660086	1107,104806	1407,041189	1484,916881	507,753605	1344,67555	0	4487,73507
15	127980486	4658,650778	4698,317509	4728,061405	4713,046602	4716,375106	4798,463537	3056,074984	3380,630264	5894,433127	3002,818189	4995,145543	5832,067489	4487,73507	0
16	127980531	2376,553917	2416,220648	2445,964544	2430,949741	2434,278246	2516,366677	773,978123	1098,533403	3612,336267	720,7213283	2713,048683	3549,970628	2205,638209	2284,310765
17	127980532	2291,08669	2330,753422	2360,497318	2345,482515	2348,811019	2430,89945	688,5108966	1013,066177	3526,86904	635,2541019	2627,581456	3464,503401	2120,170983	2369,777991
18	127982964	5448,416516	5488,083247	5517,827143	5502,81234	5506,140845	5588,229276	3845,840722	4170,396002	6684,198866	3792,583927	5784,911282	6621,833227	5277,500808	789,7657385
19	127923443	1427,630457	1465,559793	1416,064127	1273,441619	1258,196244	1176,107813	2394,650832	2487,855886	1988,128372	2447,907627	1594,197072	1945,325078	1598,546165	4805,500037
20	127923448	1878,781149	1904,793614	1855,297948	1712,67544	1745,586407	1718,549633	2940,053448	3033,258501	2353,008325	2993,310243	2033,430893	2310,205031	2049,696857	4640,545179
21	127923452	2091,740936	2117,753401	2068,257735	1925,635227	1958,546193	1931,50942	3079,731659	3172,936713	2564,306595	3132,988454	2246,39068	2521,503301	2262,656644	4594,50147
22	127924278	2125,610913	2151,623379	2102,127712	1959,505205	1992,416171	1965,379398	3186,883212	3280,088266	2280,715645	3240,140007	2280,260657	2237,912351	2296,526621	5027,732765
23	127924286	2197,879414	2223,891879	2174,396213	2031,773705	2064,684671	2037,647898	3259,151712	3352,356766	2352,984146	3312,408507	2352,529158	2310,180852	2368,795122	4955,464265
24	127924927	2140,612263	2166,624728	2117,129062	1974,506554	2007,41752	1980,380747	3201,884562	3295,089615	2480,94369	3255,141356	2295,262007	2438,140396	2311,527971	4782,724012
25	127925643	2005,15063	2031,163095	1981,667429	1839,044921	1871,955887	1844,919114	3066,422929	3159,627982	2250,561068	3119,679723	2159,800374	2207,757774	2176,066338	4916,734046
26	127925644	1830,923341	1856,935807	1807,44014	1664,817633	1697,728599	1670,691826	2892,19564	2985,400694	2134,90043	2945,452435	1985,573085	2092,097136	2001,839049	4938,773804
27	127925646	2010,38911	2036,401575	1986,905909	1844,283401	1877,194367	1850,157594	3071,661409	3164,866462	1970,618501	3124,918203	2165,038854	1927,815207	2181,304818	5118,239573

Obr. 4: Matice nejkratších cest

2.2. Nalezení minimální kostry

Pro zpracování úlohy nalezení minimální kostry pomocí Kruskalova algoritmu byl použit stejný graf jako u předchozí úlohy. Data byla na počátku skriptu konvertována do seznamů struktury:

$E = [[U_1, V_2, W_1], [U_1, V_2, W_1],... [U_k, V_k, W_k]],$

kde U a V představují jednotlivé uzly a W váhy mezi nimi. Na závěr byla výsledek vizualizován pomocí funkcí knihovny *osmnx*.

Cílem Kruskalova algoritmu je nalezení podmnožiny hran neorientovaného grafu, jenž propojuje všechny uzly, a to s minimální celkovou váhou. Algoritmus může být použit za předpokladu, že je graf souvislý a s hranami nezáporného ohodnocení. Algoritmus vyhledá minimální kostru grafu tak, že vždy vybere hranu s nejmenší vahou, která zároveň nevytváří cyklus. K tomu používá strukturu union-find, která slouží pro efektivní kontrolu, zda oba konce hrany patří ke stejnému podstromu. Pokud je zjištěno, že hrana nespojuje vrcholy ve stejném podstromu, je přidána do minimální kostry. To nastane ve třech možných situacích:

1. Ani jeden uzel hrany neleží v jiném podstromu – je vytvořen nový podstrom obsahující danou hranu
2. Jeden uzel hrany se nachází v nějakém jiném podstromu – hrana je připojena k podstromu.
3. Oba uzly hrany jsou součástí různých podstromů – dva podstromy jsou spojeny hranou do jednoho podstromu.

V námi implementovaném řešení efektivitu struktury union-find zlepšují dvě heuristiky – Path Compression a Weighted Union.

Princip Kruskalova algoritmu:

1. **Seřazení hran podle váhy**
    - Seřazení hran grafu vzestupně podle jejich vah
2. **Inicializace množiny stromů**
    - Každý vrchol je samostatný strom (disjoint set)
    - Celková váha minimální kostry iniciována na hodnotu 0
3. **Iterace přes hrany**
    - Pokud hrana nespojuje vrcholy ve stejném podstromu, přidáme ji do minimální kostry

4. **Ukončení iterativní fáze**
    - Proces ukončen ve chvíli, kdy jsou všechny vrcholy spojeny či les obsahuje pouze jeden strom
    - Výstupem je seznam hran tvořících minimální kostru a její celková váha

V algoritmu byly použity funkce:

1. *__init__*

- inicializuje množinu stromů tak, že každý uzel je svým vlastním rodičem (parent node) a má váhu 0

2. *find*

- obsahuje metodu Path Compression, více v 2.2.3.

3. *union*

- obsahuje metodu Weighted Union, více v 2.2.2.

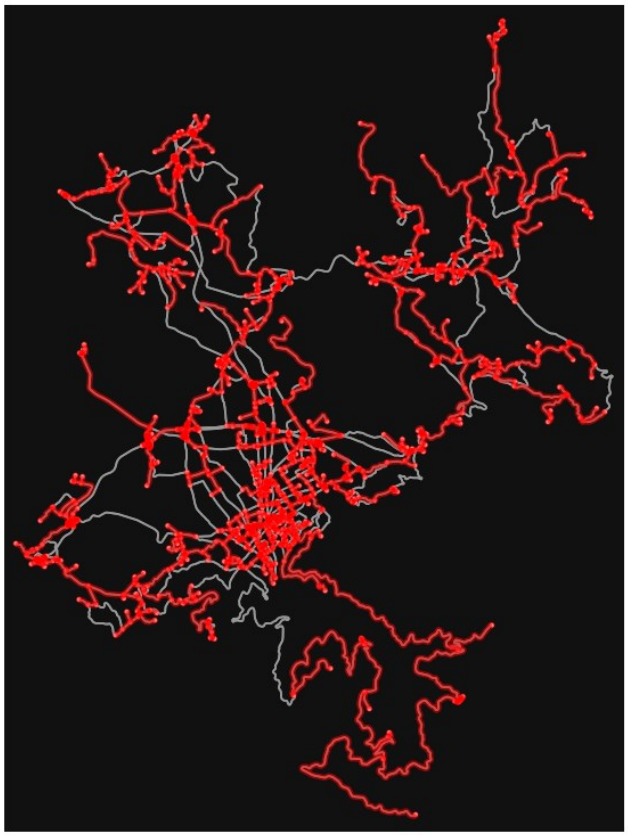
2.2.2. Využití heuristiky Path Compression

Funkce *find* hledá kořen množiny, která obsahuje uzel *v*. Tuto funkci zrychluje metoda Path Compression. Jde o kompresi cesty, která slouží ke zploštění struktury stromu, dojde k úpravě odkazu na kořen (rodiče) uzlu. V našem řešení úlohy byla využita varianta s 1 průchodem. Zkracování délky stromu bylo provedeno přeskočením jedné generace, předchůdcem uzlu se tedy stává rodič rodiče uzlu. Jedná se tedy o postupné zkracování cesty směrem k kořeni.

2.2.3. Využití heuristiky Weighted Union

Funkce *union* spojuje dva oddělené podstromy reprezentované kořeny *root1* a *root2*. Díky využití metody Weighted Union je kořen menšího stromu (s nižším ohodnocením) připojen ke kořenu většího stromu (s vyšším ohodnocením). Pokud jsou kořeny stejně ohodnocené, je vybrán libovolný z nich a hodnost výsledného podstromu se zvýší o 1. Dojde tedy ke změně identifikátoru kratšího seznamu, avšak ID delšího seznamu se nemění.

Vizualizace minimální kostry námi vybraného grafu je lze vidět na obr. 5. a její celková váha je 194,98 m.



Obr. 5: Minimální kostra grafu

3. Závěr

S využitím programovacího jazyka Python byla úspěšně nalezena nejkratší a nejrychlejší cesta mezi dvěma lokalitami oblasti Lousã pomocí Dijkstra algoritmu. Nejrychlejší cesta byla vypočítána v závislosti na povolené maximálních rychlostí daných silnic a indexu klikatosti. Nejkratší cesta měřila 32,2 km a nejrychlejší cesta trvala 45 min. Odchylky vzdáleností trasy vytvořené Dijkstra algoritmem od výsledků navigačních systémů byly poměrně malé, avšak rozdíly času naší nejrychlejší cesty, trasy společnosti Mapy.cz a trasy Google Maps byly velké z důvodů rozdílné metodiky výpočtu. Vytvořena byla také tabulka obsahující nejkratší cesty mezi všemi dvojicemi uzlů. Na závěr byla sestavena minimální kostra grafu pomocí Kruskalova algoritmu, jehož efektivita byla vylepšena dvěma heuristikami – Path Compression a Weighted Union. Celková váha minimální kostry vyšla 194,98 m.