

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA STAVEBNÍ, OBOR GEODÉZIE A KARTOGRAFIE
KATEDRA GEOMATIKY

GEOINFORMATIKA

Úloha 3 – Nejkratší cesta grafem

Rok	Semestr	Skupina	Vypracovali	Datum
2025/2026	ZS		František Gurecký Vítek Veselý	28.12.2025

1 Zadání

Implementujte Dijkstru 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 Euklidovskou vzdálenost
- nejmenší transportní čas (2 varianty)

Každou z variant otestujte pro dvě různé cesty. Výsledky umístěte do tabulky, vlastní cesty vizualizujte. Dosažené výsledky porovnejte s vybraným navigačním SW.

2 Rešené bonusové úlohy

V rámci úlohy se podařilo vyřešit veškeré bonusové úlohy, jejich popis je níže v postupu řešení.

- Řešení úlohy pro grafy se záporným ohodnocením
- Nalezení nejkratších cest mezi všemi uzly
- Nalezení minimální kostry Kruskal
- Nalezení minimální kostry Prime
- Využití heuristiky Weighted Union
- Využití heuristiky Path Compression

3 Vstupní data

Před analýzou bylo nutné nejprve graf vytvořit. Pro to byly zvolena Open Street Map, jelikož má poměrně podrobná a dobře dostupná data. Zároveň je na jejich bázi vytvořena aplikaci Mapy.cz, která by nám teoreticky měla poskytnout lepší výsledky při porovnání.

Samotná tvorba také není přímočará, v našem případě bylo zvoleno načíst data přímo v kódu a vytvořit převodní skript, který data OSM stáhne, načte přímo jako graf díky knihovny *osmnx*, která tento krok velmi usnadní. Po načtení dat jako graf následuje výpočet jednotlivých vah a následné uložení do strukturovaného souboru pro minimalizaci opětovného stahování dat.

3.1 Volba ohodnocení

Ohodnocení byla pro porovnání zvolena 4, a to:

- Euklidovská vzdálenost
- Doba projetí
- Doba projetí efektivní rychlostí
- Křivost hrany

3.1.1 Euklidovská vzdálenost

Jedná se o atribut *length* každé hrany, není počítána ze souřadnic, tedy vzdušnou čarou.

3.1.2 Doba projetí

Zde se projeví vhodnost volby dat OSM. Každý uložený prvek silnice nese také atribut *maxspeed*, není tedy nutné data klasifikovat podle tříd a rychlost tak kvalifikovaně odhadovat. Doba projetí je dána vztahem

$$t = \frac{s}{v} \tag{1}$$

kde t je čas projetí, v je maximální povolená rychlosť a s je délka hrany. Po správném převedení rychlosti na m/s získáme dobu v sekundách, kterou by trvalo hranu projet.

3.1.3 Doba projetí efektivní rychlostí

Toto je variace předchozího ohodnocení, předchozí vzorec je ještě doplněn o uvážení křivosti k

$$t = \frac{s}{v/k} \quad (2)$$

Křivost je pak dána poměrem skutečné délky hrany a vzdálenosti koncových uzelů vypočtenou ze souřadnic, tedy

$$k = \frac{s_h}{\sqrt{\Delta x_{12}^2 + \Delta y_{12}^2}} \quad (3)$$

kde s_h je zmiňovaná skutečná délka.

3.1.4 Křivost hrany

Nakonec byla křivost zvolena také jako samostatné ohodnocení grafu, její výpočet je popsán vztahem 3.

Jelikož jsou všechna zvolená ohodnocení přímo úměrná, nebylo třeba žádné hodnoty převracet. Bylo by to nutné například pro ohodnocení hran přímo jejich maximální povolenou rychlostí. Zde by bez převrácení algoritmus volil hrany, kde je rychlosť menší, protože byla cesta levnější.

4 Popis algoritmů

V této kapitole jsou popsány algoritmy aplikované na graf vytvořený ze vstupních dat popsaných v předchozí kapitole.

Nejprve jsou data načtena, je vytvořen graf a po uložení do souboru je načten do námi zvolené struktury.

```
[  
 {  
     pocatek1: {  
         konec1: ohodnoceni1  
         konec2: ohodneceni2  
     }  
 },  
 dalsi graf...  
 ]
```

Efektivně se tedy jedná o pole grafů, kde každý graf je slovníkem, ve kterém jsou klíči počáteční uzly a hodnotami slovník, ve kterém jsou klíče koncové uzly s hodnotou ohodnocením této hrany.

4.1 Dijkstrův algoritmus

Prvním popsaným algoritmem je Dijkstrův algoritmus, který hledá nejkratší cesty v ohodnocených grafech. Předpokládáme, že žádná hrana nemá záporné ohodnocení. Pro urychlení výpočtu byla použita implementace s binární haldou (`heapq`) [1], čímž se dosáhne časové složitosti

$$O((V + E) \log V).$$

Vstup: Ohodnocený graf G a počáteční vrchol v_0

Výstup: Ohodnocení vrcholů h a pole předchůdců P

1. Pro všechny vrcholy v :
 - (a) $h(v) \leftarrow +\infty$, $P(v) \leftarrow$ nedefinováno
2. $h(v_0) \leftarrow 0$
3. Inicializuj prioritní haldu H a vlož $(0, v_0)$
4. Dokud je halda H neprázdná:
 - (a) Vyber a odstraň (d, v) s nejmenším d z haldy
 - (b) Pokud $d \neq h(v)$, pokračuj na další iteraci
 - (c) Pro všechny následníky w z vrcholu v :
 - i. Pokud $h(w) > h(v) + \ell(v, w)$:
 - A. $h(w) \leftarrow h(v) + \ell(v, w)$
 - B. $P(w) \leftarrow v$
 - C. Vlož $(h(w), w)$ do haldy

Tento algoritmus využívá také principu relaxace hran. Tento princip je popsán v bodu c) algoritmu výše. Samotná relaxace zajišťuje to, že jsou ke všem vrcholům ukládány skutečně nejkratší cesty a tedy ohodnocení. Kontroluje, zda je ohodnocení následujícího vrcholu větší než součet ohodnocení aktuálního vrcholu a hrany, která tyto vrcholy spojuje, pokud ano, uloží aktuální vrchol jako nového předka, změní jeho ohodnocení na nižší hodnotu a vloží následující vrchol zpět do haldy.

K rekonstruování cesty se používá funkce, kterou lze popsat algoritmem níže. [2]

Vstup: Pole předchůdců P , počáteční vrchol s , koncový vrchol t

Výstup: Cesta z s do t ve správném pořadí

1. Inicializuj prázdný seznam $path$
2. $v \leftarrow t$
3. **while** $v \neq s$ a $v \neq \text{None}$:
 - (a) Přidej vrchol v na konec seznamu $path$
 - (b) $v \leftarrow P(v)$
4. Přidej vrchol s do seznamu $path$
5. **Vrat** seznam $path$

4.2 Bellman–Fordův algoritmus

Bellman–Fordův algoritmus je pomalejší než Dijkstra, ale na rozdíl od něj dokáže zpracovat grafy se zápornými ohodnoceními hran. Algoritmus opakovaně relaxuje všechny hrany a v poslední iteraci detekuje případný záporný cyklus. Podle implementace vrací [-1] při jeho nalezení. [1]

Vstup: Ohodnocený graf G a počáteční vrchol v_0

Výstup: Ohodnocení h nebo indikace záporného cyklu

1. Pro všechny vrcholy v :
 - (a) $h(v) \leftarrow +\infty$
2. $h(v_0) \leftarrow 0$
3. Pro $i = 1$ až $n - 1$ (kde n je počet vrcholů):
 - (a) Pro každou hranu (u, v) :
 - i. Pokud $h(u) + \ell(u, v) < h(v)$:
 - A. $h(v) \leftarrow h(u) + \ell(u, v)$
 - B. $P(v) \leftarrow u$
 4. Pro každou hranu (u, v) :
 - (a) Pokud $h(u) + \ell(u, v) < h(v)$:
 - i. Vrat [-1] — existuje záporný cyklus
 5. Vrat hodnoty h a pole P

4.3 Kruskalův algoritmus

Kruskalův algoritmus počítá minimální kostru grafu (MST). Implementace využívá datovou strukturu Union–Find s kompresí cesty a union podle ranku. Hrany jsou nejprve seřazeny podle váhy.

Vstup: Ohodnocený graf G

Výstup: Minimální kostra T

1. Inicializuj množinu T jako prázdnou
2. Seřad všechny hrany (u, v, w) podle rostoucí váhy w
3. Pro každý vrchol v :
 - (a) $\text{parent}(v) \leftarrow v$
 - (b) $\text{rank}(v) \leftarrow 0$
4. Pro každou hranu (u, v, w) v seřazeném pořadí:
 - (a) Pokud $\text{Find}(u) \neq \text{Find}(v)$:
 - i. $\text{Union}(u, v)$
 - ii. Přidej hranu (u, v, w) do T
5. Vrat T

Procedura Find

Zde využíváme heuristiku komprese cesty, konkrétně jednopruhodovou se skokem na prarodiče. [2]

1. **Procedura** $\text{Find}(u)$:
 - (a) **while** $\text{parent}(u) \neq u$:
 - i. $u \leftarrow \text{parent}(\text{parent}(u))$ (komprese skokem na prarodiče)
 - ii. $u \leftarrow \text{parent}(u)$
 - (b) **return** u

Procedura Union

Pro tuto proceduru je využívána heuristika váženého sjednocení, je tedy komplexnější rozhodování o tom, jak stromy sjednotit. [2]

1. **Procedura** Union(u, v):

- (a) $r_u \leftarrow \text{Find}(u), r_v \leftarrow \text{Find}(v)$
- (b) Pokud $r_u = r_v$: vrat *false*
- (c) Pokud $\text{rank}(r_u) < \text{rank}(r_v)$: $\text{parent}(r_u) \leftarrow r_v$
- (d) Jinak pokud $\text{rank}(r_v) < \text{rank}(r_u)$: $\text{parent}(r_v) \leftarrow r_u$
- (e) Jinak:
 - i. $\text{parent}(r_u) \leftarrow r_v$
 - ii. $\text{rank}(r_v) \leftarrow \text{rank}(r_v) + 1$
- (f) vrat *true*

4.4 Primův algoritmus

Primův algoritmus také počítá minimální kostru grafu, avšak odlišným způsobem: začíná z počátečního uzlu a postupně přidává nejlevnější hranu vedoucí do dosud nenavštíveného vrcholu. Implementace využívá prioritní frontu. **Vstup:** Ohodnocený souvislý graf G a počáteční vrchol v_0

Výstup: Minimální kostra T

1. Pro všechny vrcholy v :
 - (a) $\text{stav}(v) \leftarrow \text{nenalezený}$
2. Inicializuj prázdnou kostru T
3. Inicializuj prioritní haldu H
4. Vlož $(0, v_0, \text{none})$ do H (váha, vrchol, rodič)
5. Dokud je halda H neprázdná:
 - (a) Vyber (w, v, u) s nejmenší vahou
 - (b) Pokud $\text{stav}(v) = \text{uzavřený}$: pokračuj
 - (c) $\text{stav}(v) \leftarrow \text{uzavřený}$
 - (d) Pokud $u \neq \text{none}$, přidej hranu (u, v, w) do T
 - (e) Pro všechny následníky x vrcholu v :
 - i. Pokud $\text{stav}(x) \neq \text{uzavřený}$:
 - A. Vlož $(\ell(v, x), x, v)$ do H
6. Vrat T

5 Výstupní data

Pro všechna ohodnocení byly dle zadání otestovány dvě cesty. Pro lepší porovnání bylo nutné přidat vypočítat stejné kritérium pro různě ohodnocené cesty. To však není problém, stačí upravit rekostrukční algoritmus z 4.1 tak, že mu pro jednu cestu dodáme všechny ohodnocené grafy, aby bylo možné nasčítat veškerá ohodnocení.

Výsledky byly porovnány numericky, graficky a také s dostupnými nástroji, konkrétně s aplikacemi *Mapy.cz* a *Google Maps*.

5.1 Numerické výsledky

Jak bylo zmíněno, pro všechny nejkratší cesty byla spočtena všechna ohodnocení, výsledky jsou sepsány v tabulkách níže.

Hodnocení/hodnota	vzdálenost [m]	čas [s]	čas s uvážením křivosti [s]	součet křivosti [-]
nejkratší	135977,89	8071	8237	507,684
nejrychlejší	143860,36	6737	6974	315,604
nejrychlejší (křivost)	143892,11	6739	6966	317,863
nejpřímější	181440,62	10295	11072	182,258

Tabulka 1: Hodnoty cen cest mezi body 2440 a 38295

Hodnocení/hodnota	vzdálenost [m]	čas [s]	čas s uvážením křivosti [s]	součet křivosti [-]
nejkratší	46235,77	2598	2630	170,450
nejrychlejší	48220,98	2192	2405	121,894
nejrychlejší (křivost)	48073,93	2197	2287	128,052
nejpřímější	52162,87	2307	2412	96,407

Tabulka 2: Hodnoty cen cest mezi body 5417 a 32215

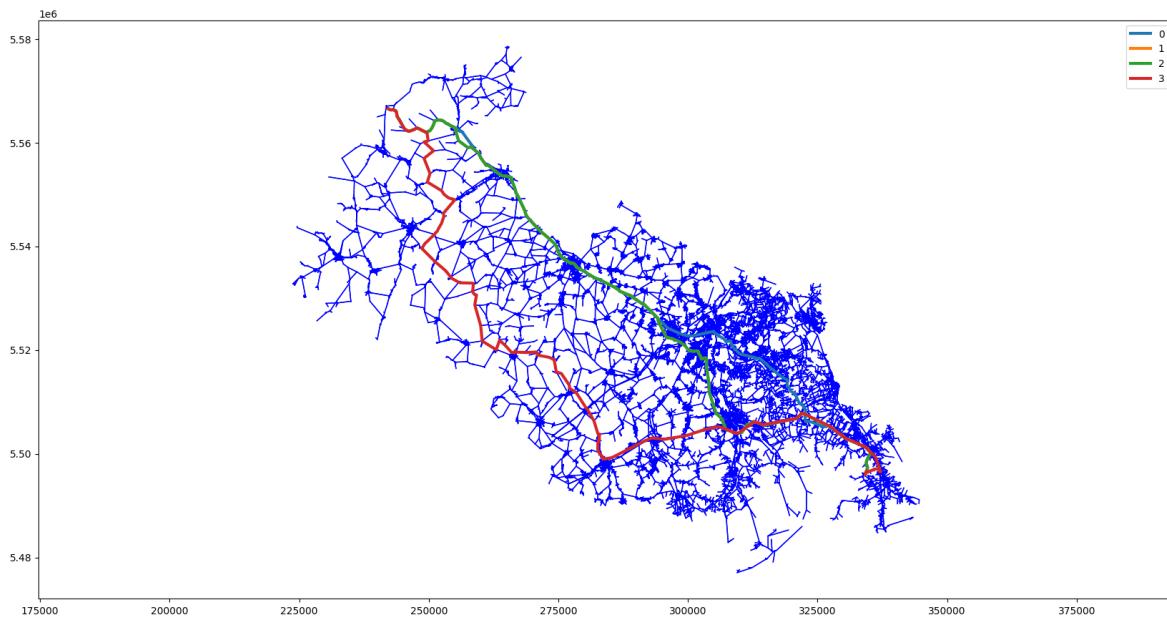
Je v pořádku, že právě na diagonále jsou vždy nejnižší ceny cest, jelikož podle těchto kritérií byly cesty hodnoceny.

Velmi podobných výsledků dosahují nejrychlejší cesty, uvážení křivosti trasy udělalo ve výsledku minimální rozdíl a nevyplatí se tedy jezdit příliš rovněji. Nelze přímo porovnat časy mezi nejrychlejšími cestami, jelikož jsme koeficient křivosti vytvořili uměle, zdá se však, že započtení křivosti přidá cestám zhruba 4% času navíc.

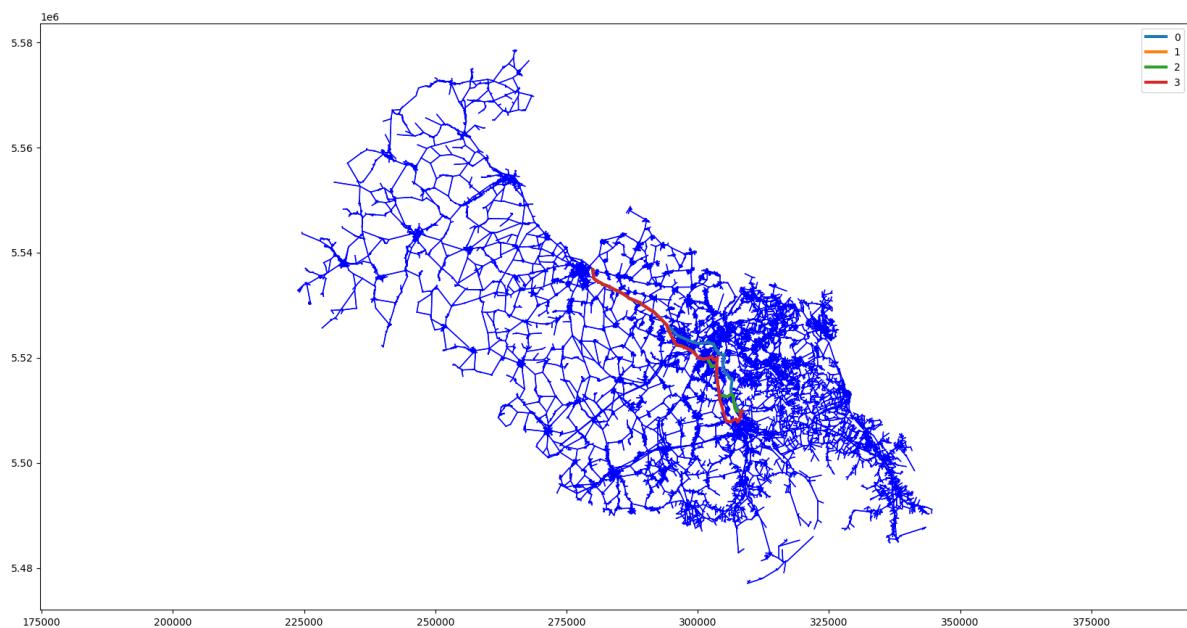
Také se ukázalo, že nejrychlejší cesta z daleka nemusí být nejkratší a naopak.

5.2 Grafické výsledky

Níže jsou v jednom grafu zobrazeny veškeré počítané cesty. V legendě odpovídá linie 0 ohodnocení vzdáleností, 1 času pro přejetí hran, 2 času pro přejetí hran s uvážením křivosti a 3 jako křivost hrany.



Obrázek 1: Hodnoty cen cest mezi body 2440 a 38295



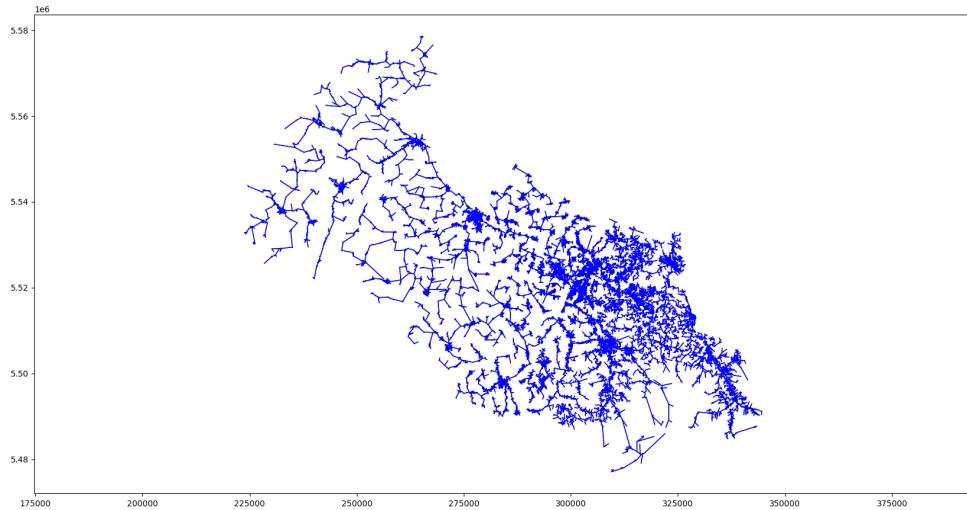
Obrázek 2: Hodnoty cen cest mezi body 5417 a 32215

Jak napověděly již numerické výsledky, rozdíl mezi ohodnocením 1 a 2 je velmi nepatrny, i v grafice se tedy kryjí a nejsou dobře vidět obě z nich.

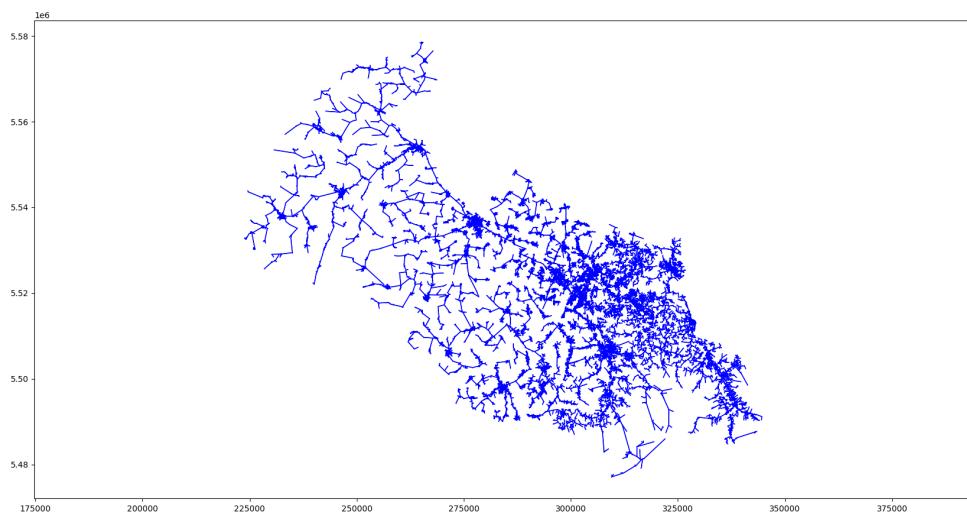
První cesty byly voleny jako co nejvzdálenější, cesty se tedy liší poměrně výrazně, jelikož mají prakticky celý graf na to najít jinou cestu. Korelace mezi variabilitou a vzdáleností uzlů potvrzují také druhé cesty, které se nevětví zdaleko tak, jak v prvním obrázku. Druhé cesty byly voleny spíše z osobních důvodů, Moravskoslezský kraj je protažen od severozápadu k jihovýchodu a je poměrně úzký v kolmém směru, většina infrastruktury s vyšší maximální povolenou rychlostí je tedy orientována podobně.

5.3 Minimální kostry

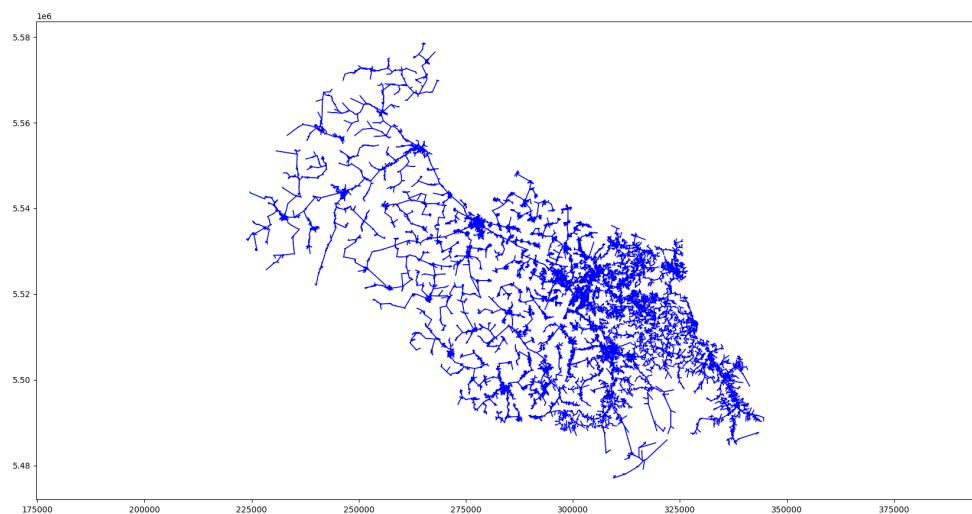
Dále jsou zde ukázány minimální kostry, jako počáteční bod pro Primův algoritmus byl zvolen bod 0.



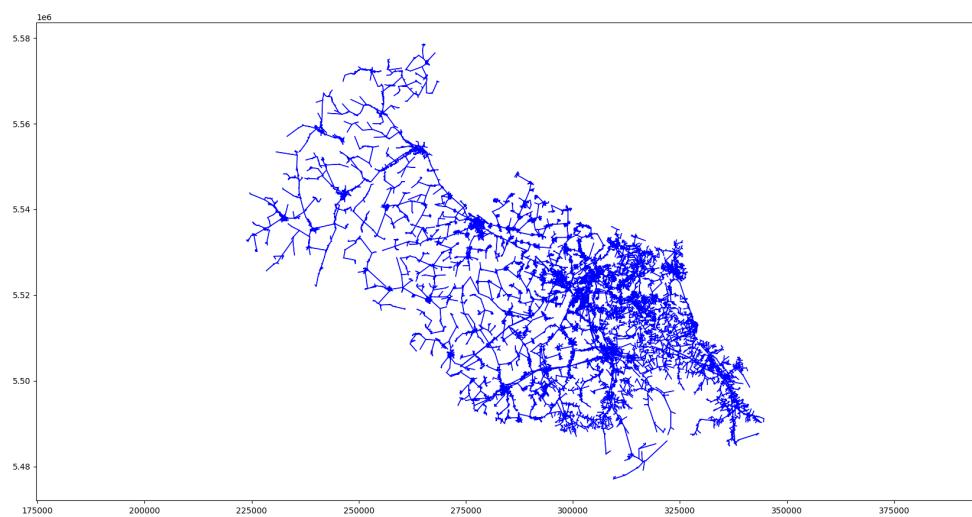
Obrázek 3: Minimální kostra Kruskalova algoritmu pro ohodnocení 0



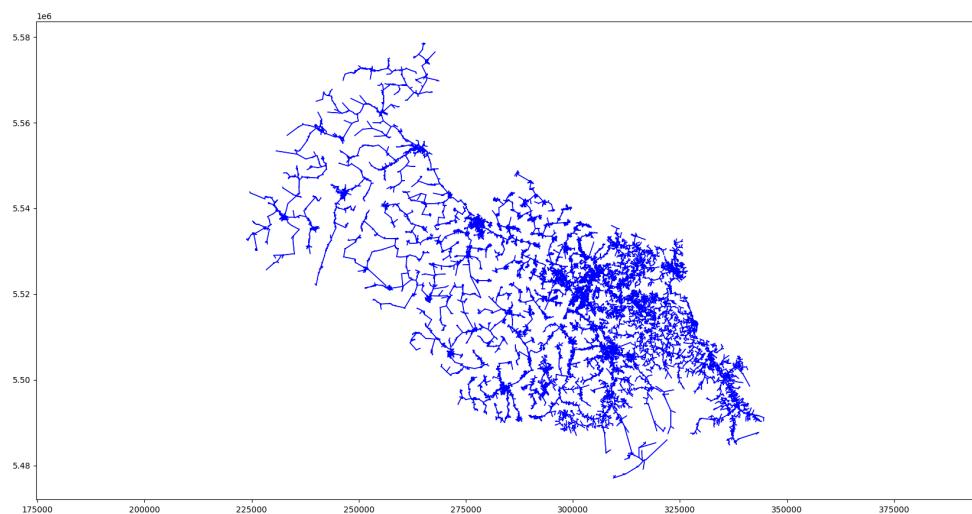
Obrázek 4: Minimální kostra Kruskalova algoritmu pro ohodnocení 1



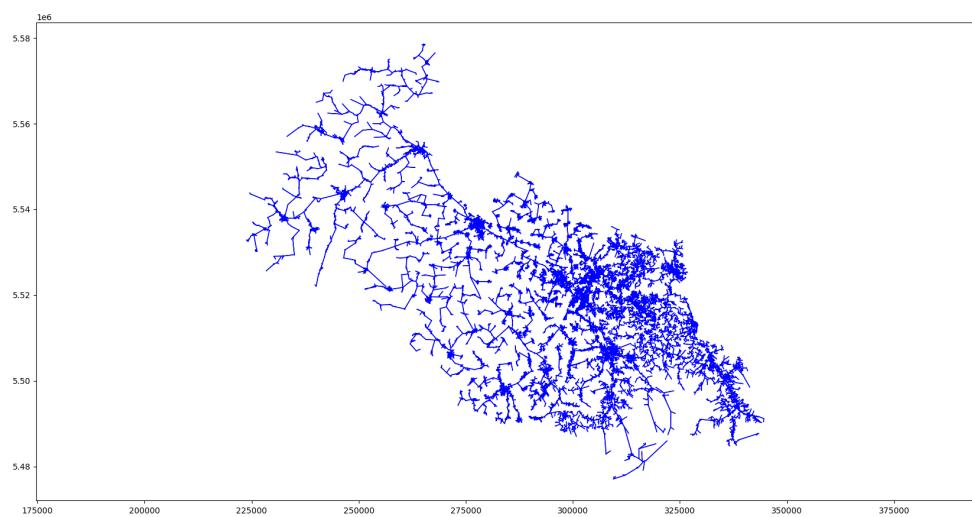
Obrázek 5: Minimální kostra Kruskalova algoritmu pro ohodnocení 2



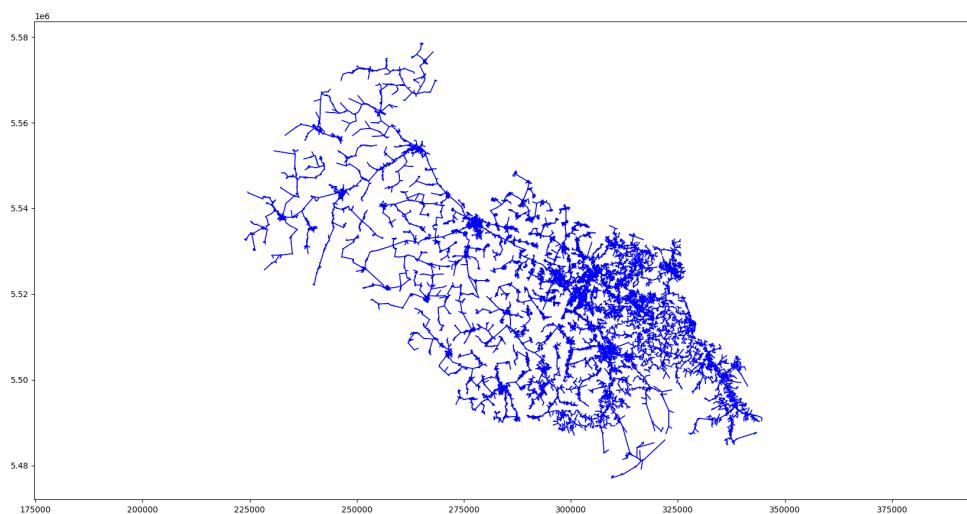
Obrázek 6: Minimální kostra Kruskalova algoritmu pro ohodnocení 3



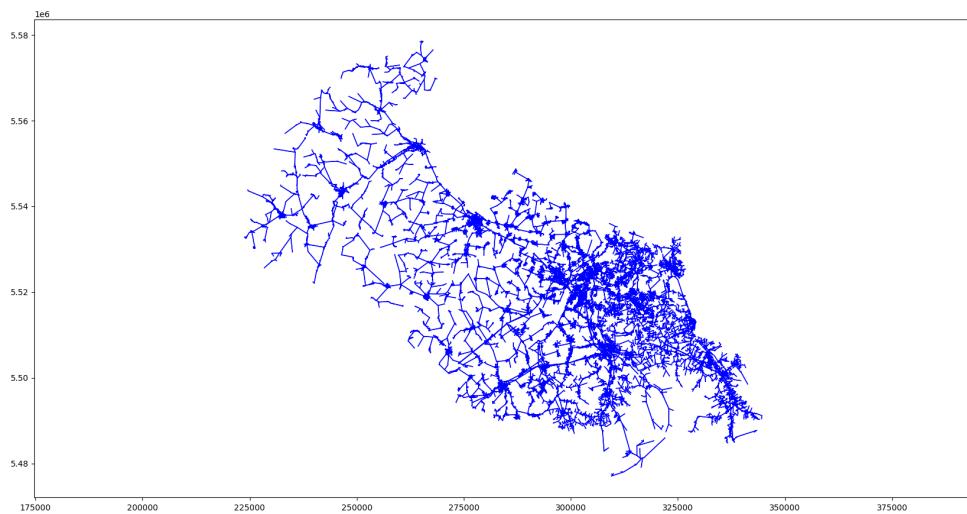
Obrázek 7: Minimální kostra Primova algoritmu pro ohodnocení 0 z uzlu 0



Obrázek 8: Minimální kostra Primova algoritmu pro ohodnocení 1 z uzlu 0



Obrázek 9: Minimální kostra Primova algoritmu pro ohodnocení 2 z uzlu 0

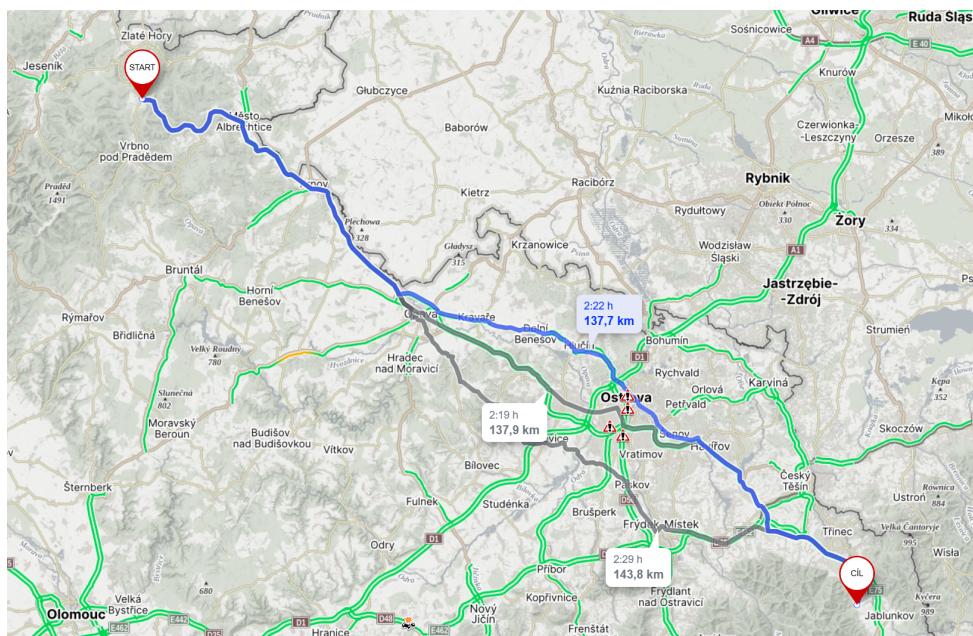


Obrázek 10: Minimální kostra Primova algoritmu pro ohodnocení 3 z uzlu 0

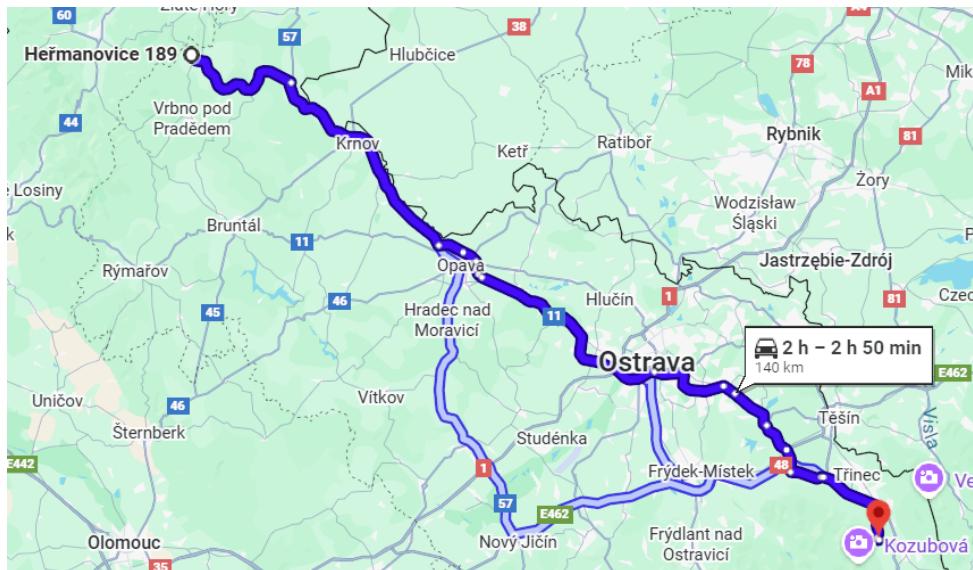
Jelikož se jedná o poměrně velkou plochu jsou si kostry na pohled podobné. Při bližším zkoumání jsou však vidět patrné rozdíly mezi jednotlivými ohodnoceními. Bylo by možné měnit tvar kostry Primova algoritmu volbou jiného počátečního bodu.

5.4 Porovnání s aplikacemi

Nyní již máme veškeré výsledky, můžeme je porovnat i s komerční aplikací. Počáteční a koncové uzly byly voleny co nejlépe od oka, nemusí se jednat o přesný uzel, jelikož v generovaných grafech nejsou pro přehlednost vypsány veškeré čísla bodů. Díky znalosti kraje se však podařilo uzly odhadnout velmi dobře.



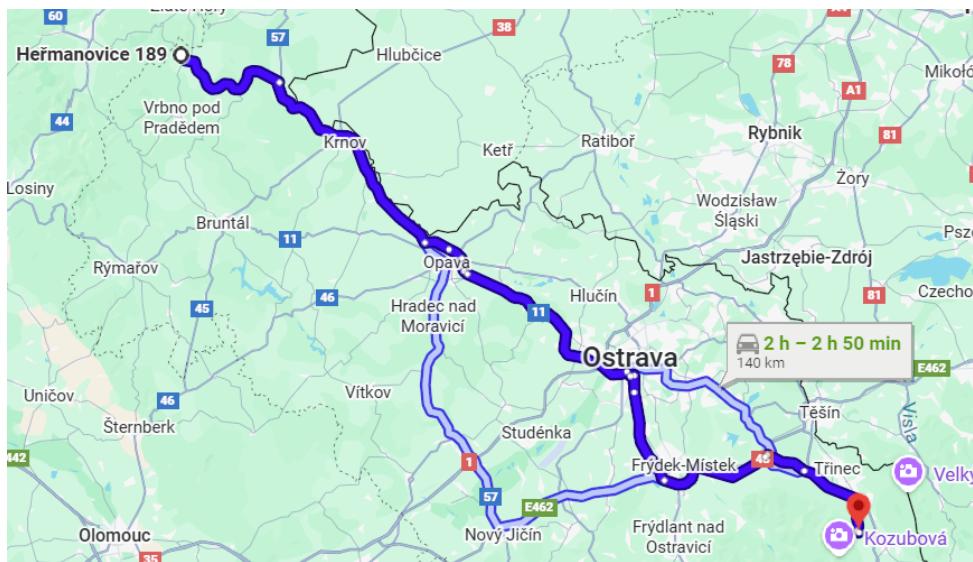
Obrázek 11: Cesta mezi body 2440 a 38295 - Mapy.cz varianta krátká



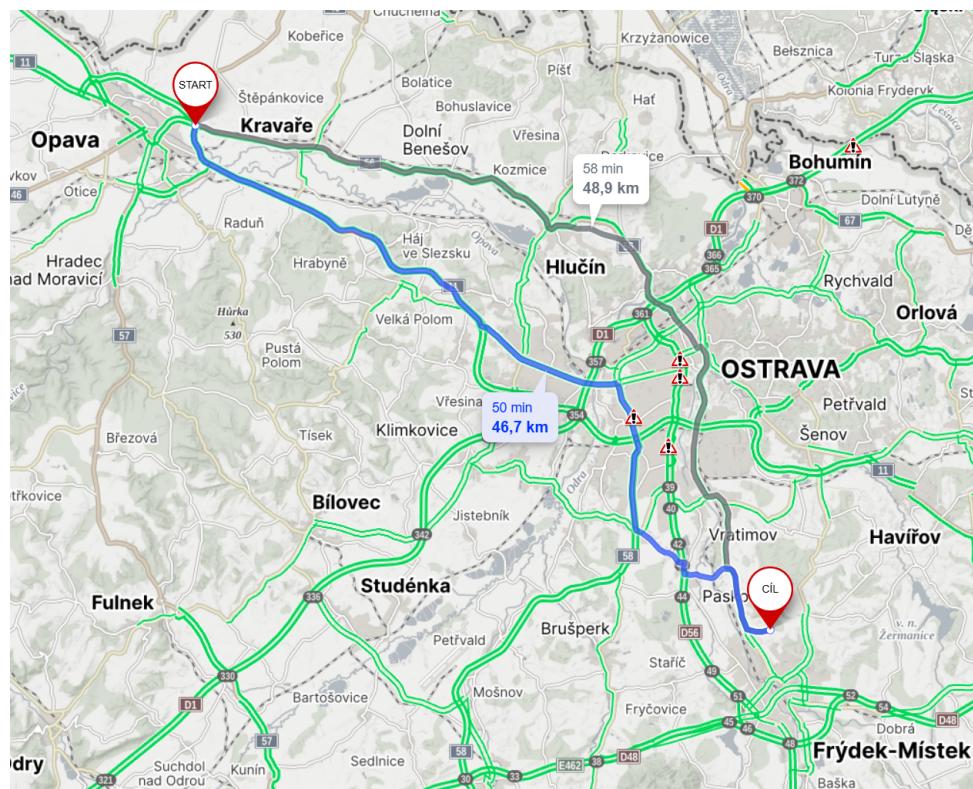
Obrázek 12: Cesta mezi body 2440 a 38295 - Google Maps varianta krátká



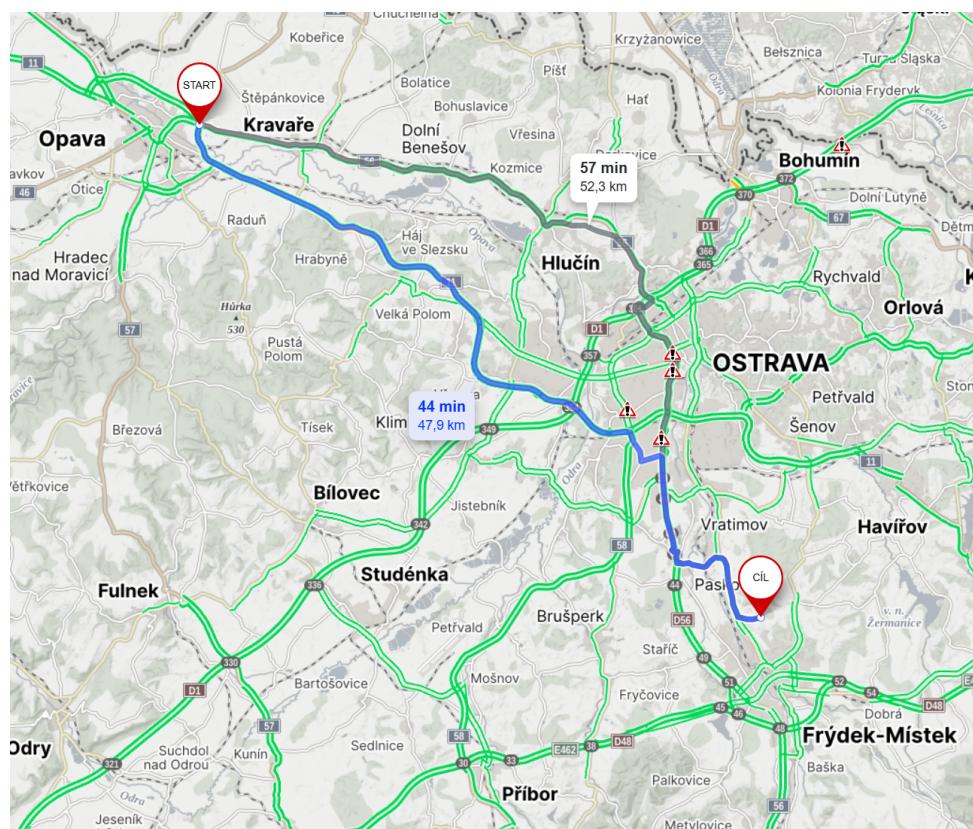
Obrázek 13: Cesta mezi body 52440 a 38295 - Mapy.cz varianta rychlá



Obrázek 14: Cesta mezi body 2440 a 38295 - Google Maps varianta rychlá



Obrázek 15: Cesta mezi body 5417 a 32215 - Mapy.cz varianta krátká



Obrázek 16: Cesta mezi body 5417 a 32215 - Mapy.cz varianta rychlá

Pro druhou cestu vyšla jako nejrychlejší a nejkratší z Google Maps ta samá cesta.



Obrázek 17: Cesta mezi body 5417 a 32215 - Google Maps

Již na pohled mají cesty viditelně odlišný průběh. Provedme tedy numerické porovnání. S variantou krátká bude porovnána nejkratší cesta, tedy 1. ohodnocení a s variantou rychlá bude porovnána cesta s nejmenším časem dojezdu, tedy 2. ohodnocení. Námi dosažené výsledky zaokrouhlíme, stejně jako *Mapy.cz*, na destiny kilometrů a celé minuty. Aplikace *Google Maps* nedává ani přesný čas cesty, uvažuje čas a den odjezdu a ukazuje pouze rozptyl, bude tak brána nejnižší hodnota, která by neměla uvažovat žádné zpomalení na cestě hustotou dopravy. Čas *Google Maps* zaokrouhuje v závislosti na délce cesty nekonzistentně.

Cesta/hodnocení	vzdálenost [km]	čas [min]
Vypočtená (krátká)	136,0	135
Vypočtená (rychlá)	143,9	112
<i>Mapy.cz</i> (krátká)	137,7	142
<i>Mapy.cz</i> (rychlá)	145,2	112
<i>Google Maps</i> (krátká)	140,0	120
<i>Google Maps</i> (rychlá)	146,0	110

Tabulka 3: Hodnoty cen cest mezi body 2440 a 38295

Z numerických hodnot však vidíme, že rozdíly mezi naší implementací a daty obou aplikací jsou nepatrné, nejvíce se stejná varianta liší o 2,5 km a o 22 minut, což jsou velmi slušné výsledky.

Cesta/hodnocení	vzdálenost [km]	čas [min]
Vypočtená (krátká)	46,2	43
Vypočtená (rychlá)	48,2	36
<i>Mapy.cz</i> (krátká)	46,7	50
<i>Mapy.cz</i> (rychlá)	47,9	44
<i>Google Maps</i> (rychlá a krátká)	48	45

Tabulka 4: Hodnoty cen cest mezi body 5417 a 32215

6 Závěr

Podařilo se pochopit a implementovat algoritmy pro vyhledávání nejkratších cest v ohodnocených grafech. Byly vyřešeny veškeré bonusové úlohy, algoritmy a myšlenky byly důkladně popsány a použité zdroje jsou uvedeny v seznamu literatury.

Také byly získány znalosti o algoritmech na výpočet minimálních koster grafů, jelikož se také jedná o velmi důležitou úlohu této disciplíny.

Při spouštění programu doporučujeme opatrně volit, co zobrazovat, často se jedná o náročné úlohy, které trvají poměrně dlouhou dobu. Nejdéle trvá spočítat veškeré kombinace startovních a koncových bodů, natož všechny tyto cesty zobrazit. Samotné zobrazování trvá velmi dlouho, jelikož je graf poměrně podrobný.

Výsledky Bellman-Fordova a Dijkstrova algoritmu se na našich datech shodovaly. Detekci záporného cyklu jsme ověřili pouze nastavením všech vah na záporné hodnoty. Náhodné záporné ohodnocení hran jsme nezkoušeli, jelikož u grafu o necelých 40 000 vrcholech by se správnost výsledku jen těžko kontrolovala.

Z důvodu časové náročnosti jsme hledání cest „každý s každým“ vyzkoušeli pouze na prvních 50 vrcholů. U grafu s necelými 40 000 vrcholy by tato operace trvala desítky minut až hodiny.

Možným jiným řešením by bylo lepší zpřístupnění volby počátečního a koncového bodu uživateli, například kliknutím do mapy a volbou nejbližšího uzlu. Také by se dal graf proředit a některé uzly shluknout dohromady tak, aby nebyla porušena souvislost grafu.

Celkově se plně podařilo splnit zadání.

V Praze dne 28.12.2025

František Gurecký

Vítěk Veselý

Literatura

- [1] Martin Mareš a Tomáš Valla. *Pruvodce labyrintem algoritmů – druhé vydání*. CZ.NIC, 2022. ISBN: 978-80-88168-66-9. URL: <https://pruvodce.ucw.cz/static/pruvodce.pdf>.
- [2] Tomáš Bayer. *Nejkratší cesty grafem*. 2025. URL: <https://github.com/k155cvut/ygei/blob/main/prednasky/geoinf9.pdf>.