# Rješavanje problema energetski optimalnog puta

**Knez, Dominik**

**SVEUČILIŠTE U ZAGREBU**
**FAKULTET PROMETNIH ZNANOSTI**

Dominik Knez

# RJEŠAVANJE PROBLEMA ENERGETSKI OPTIMALNOG PUTA

**ZAVRŠNI RAD**

Zagreb, svibanj 2023.

**SVEUČILIŠTE U ZAGREBU**
**FAKULTET PROMETNIH ZNANOSTI**

**ZAVRŠNI RAD**

# RJEŠAVANJE PROBLEMA ENERGETSKI OPTIMALNOG PUTA

# SOLVING ENERGY-DEPENDENT SHORTEST PATH PROBLEM

**Mentor:** dr. sc. Tomislav Erdelić

**Student:** Dominik Knez

**JMBAG:** 0035227317

Zagreb, svibanj 2023.

# RJEŠAVANJE PROBLEMA ENERGETSKI OPTIMALNOG PUTA

Sažetak:

Prijelaz automobilske industrije na električna vozila konstantno se ubrzava i time se drastično povećava udio električnih vozila na globalnom tržištu. Uz sve prednosti električnih vozila dolazi i jedan veliki nedostatak, ograničeni kapacitet baterije tj. mali doseg vožnje.

Jedno od mogućih rješenja za povećanje dosega vožnje električnih vozila, ne zahtijevajući povećanje kapaciteta baterije, je primjena rutiranja vozila prema energetski optimalnom putu. Problem rutiranja vozila direktno vezan za područje inteligentnih transportnih sustava. Cilj ovog rada je rješavanje problema energetski optimalnog puta na cestovnoj mreži i izrada interaktivnog grafičkog korisničkog sučelja za jednostavno i učinkovito korištenje od strane krajnjih korisnika.

U svrhu postizanja ovog cilja izrađen je program pomoću razvojnog okruženja Microsoft Visual Studio i programskog jezika C#. Kroz rad je detaljno objašnjen Dijkstra algoritam za rješavanje problema optimalnog puta te implementacija Fibonaccijeve hrpe strukture podataka koja značajno ubrzava izvršavanje algoritma. Rezultat rada nudi praktičan uvid u rješenje problema energetski optimalnog puta pomoću izrađenog interaktivnog grafičkog korisničkog sučelja i omogućava provođenje energetsko-prostornih analiza na cestovnoj mreži.

**Ključne riječi:** C#; Električna vozila ; Energetski optimalni put ; Dijkstra algoritam; Fibonacci heap struktura podataka

# SOLVING ENERGY-DEPENDENT SHORTEST PATH PROBLEM

Abstract:

The transition of the automotive industry to electric vehicles is constantly accelerating and thus the share of electric vehicles on the global market is drastically increasing. Along with all the advantages of electric vehicles comes one major drawback, the limited capacity of the battery, i.e. a small driving range.

One of the possible solutions to increase the driving range of electric vehicles, without requiring an increase in battery capacity, is routing of electric vehicles according to the energy-dependent shortest path. The problem of vehicle routing is directly related to the field of intelligent transport systems. The aim of this research is to solve energy-dependent shortest path on the road network and create an interactive graphical user interface for easy and efficient use by end users.

In order to achieve this aim, a program was created using the Microsoft Visual Studio development environment and the C# programming language. The paper explains in detail the operation of the Dijkstra algorithm for solving the shortest path problem and the implementation of the FibonacciHeap data structure, which significantly speeds up the execution of the algorithm. The result of this research offers a practical insight into the solution of the energy-dependent shortest path problem using the created interactive graphic user interface and enables the implementation of energy-spatial analyses on the road network.

**Key Words:** C#; Electric vehicles; Energy-dependent shortest path; Dijkstra algorithm; Fibonacci heap data structure

**SVEUČILIŠTE U ZAGREBU**
**FAKULTET PROMETNIH ZNANOSTI**
ODBOR ZA ZAVRŠNI RAD

Zagreb, 5. svibnja 2023.

Zavod: **Zavod za inteligentne transportne sustave**
Predmet: **Algoritmi i programiranje**

# ZAVRŠNI ZADATAK br. 7167

Pristupnik: **Dominik Knez (0035227317)**
Studij: Inteligentni transportni sustavi i logistika
Smjer: Inteligentni transportni sustavi

Zadatak: **Rješavanje problema energetski optimalnog puta**

Opis zadatka:

Cilj rada je izraditi programsku podršku za rješavanje problema energetski najkraćeg puta na mreži. Prvo je potrebno proučiti podatke o cestovnim segmentima u digitalnoj karti Republike Hrvatske zajedno s njihovim energetskim profilima, spremljenih u tekstualnoj datoteci. Potom je potrebno učitati podatke u aplikaciju i kreirati graf cestovne mreže na temelju podataka u datoteci. Nakon kreiranja grafa, potrebno je implementirati neki od algoritama za pronalazak prostorno najkraćeg puta na grafu. Potom je nakon provjere rada algoritma, potrebno modificirati graf i algoritam da pronalaze energetski najkraći put na grafu. Dodatno, za odabir početnih parametara i prikaz rješenja problema potrebno je izraditi interaktivno grafičko sučelje s pozadinskom kartom. Na kraju je potrebno napraviti kratku analizu podataka koristeći izrađenu aplikaciju.

Mentor:

dr. sc. Tomislav Erdelić

Predsjednik povjerenstva za završni ispit:

# Sadržaj

# 1 UVOD

Energetski troškovi u prometnom sektoru su visoki i predstavljaju veliki izazov za zaštitu okoliša i energetsku održivost svijeta. Prema statistikama prometni sektor odgovoran je za gotovo 30% svjetske potrošnje energije dok vozila za cestovni promet troše najviše energije u prometu, preko 80%, (1,2). Većina energije koja se trenutno koristi dolazi iz fosilnih goriva, čije izgaranje stvara štetne stakleničke plinove. Ova emisija stakleničkih plinova povećala se za 50% od 1990. godine, što je glavni uzrok globalnog zatopljenja. Osim ekoloških posljedica koji štete ljudima i okolišu, emisija stakleničkih plinova također ima negativne ekonomske učinke, s godišnjim gubicima od stotinjak milijardi dolara, (3).

Kako bi se riješio ovaj problem, primjenjuju se alternativni oblici energije u cestovnom prometu koji bi mogli doprinijeti dekarbonizaciji. Električna energija trenutno je najraširenija oblik alternativne energije u cestovnom prometu, a autoindustrija se sve više orijentira prema električnim vozilima. Unatoč brojnim prednostima električnih vozila, jedan od nedostataka je ograničen doseg vožnje zbog kapaciteta baterije. Proizvođači vozila neprestano poboljšavaju baterijske tehnologije, aerodinamiku i smanjuju masu električnih vozila kako bi povećali njihov doseg. Međutim, ova poboljšanja su ograničena i imaju svoje granice. Jedno od mogućih rješenja povećanja dosega vožnje je rutiranje električnih vozila prema energetski optimalnom putu. U tom kontekstu, inteligentni transportni sustavi (ITS) igraju ključnu ulogu u usmjeravanju prometa i optimizaciji vožnje.

ITS predstavlja upravljačku i informacijsko-komunikacijsku nadogradnju klasičnog prometnog sustava kojom se postižu značajna poboljšanja u svim dijelovima prometnog sustava. Osim smanjenja potrošnje energije i emisija štetnih plinova, ITS omogućava povećanje sigurnosti i učinkovitosti prometa, te doprinosi održivosti cestovnog prometa (4).

Naslov završnog rada je rješavanje problema energetski optimalnog puta te je rad podijeljen u 7 tematskih cjelina:

1. Uvod

2. Električna vozila
3. Opis podataka cestovne mreže
4. Pronalazak optimalnog puta na grafu
5. Fibonaccijeva hrpa
6. Grafičko sučelje
7. Zaključak.

U drugom poglavlju opisana je podjela električnih vozila, navedene su njihove prednosti te je prikazana važnosti kapaciteta baterije u kontekstu energetski optimalnog puta. Nadalje, objašnjava se povezanost između električnih vozila, problema energetski optimalnog puta i inteligentnih transportnih sustava.

U trećem poglavlju opisani su dobiveni podatci o cestovnoj mreži koji su korišteni za izradu ovog rada te je objašnjen proces učitavanja tih podataka u program.

U četvrtom poglavlju detaljno se obrađuje modeliranje cestovne mreže grafom, pronalazak optimalnog puta na grafu počevši prikazom problema sedam mostova Königsberga, nakon čega se detaljno objašnjava teorija grafova te primjena Bellman-Ford i Dijkstrinog algoritma u rješavanju problema optimalnog puta.

U petom poglavlju je analiziran rad Fibonaccijeve hrpe, uključujući objašnjenje operacija koje se primjenjuju na hrpu te su prikazani rezultati implementacije Fibonaccijeve hrpe za ubrzavanje rada Dijkstrinog algoritma.

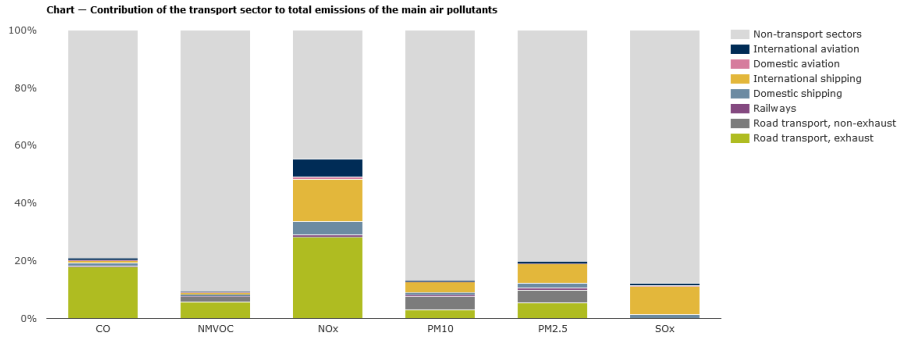
U šestom poglavlju je opisano i prikazano izrađeno interaktivno grafičko sučelje za analizu energetskih optimalnih puteva na cestovnoj mreži.

U sedmom poglavlju dan je zaključak na temelju vlastitog istraživanja.

Na kraju rada, navedena je literatura koja je korištena za izradu rada, kao i popis slika i tablica koje su referencirane u radu.

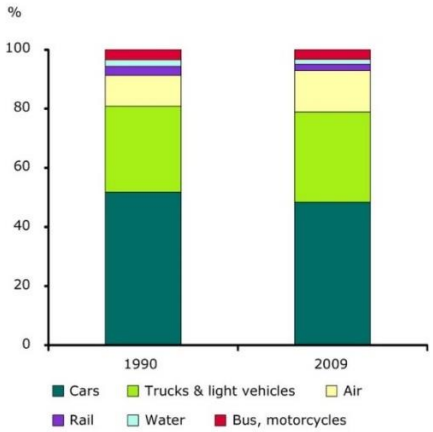
# 2 ELEKTRIČNA VOZILA

Iako su konvencionalna goriva, poput benzina i dizela, dugo vremena bila dominantni izbor za cestovni promet, donose mnoge ekološke i zdravstvene probleme. Osim što su dostupna u ograničenim količinama jer se dobivaju iz fosilnih goriva, njihovo izgaranje stvara štetne zagađivače poput: dušikovog dioksida, čestica, ozona, sumporovog dioksida, ugljikovog monoksida, benzena i toksičnih metala koji imaju negativan utjecaj na zdravlje ljudi i okoliš. Na **Slika 1** prikazana je ukupna emisija glavnih zagađivača zraka iz cijelog prometnog sektora, (5).




**Slika 1.** Ukupna emisija glavnih zagađivača zraka u prometu, *(6)*

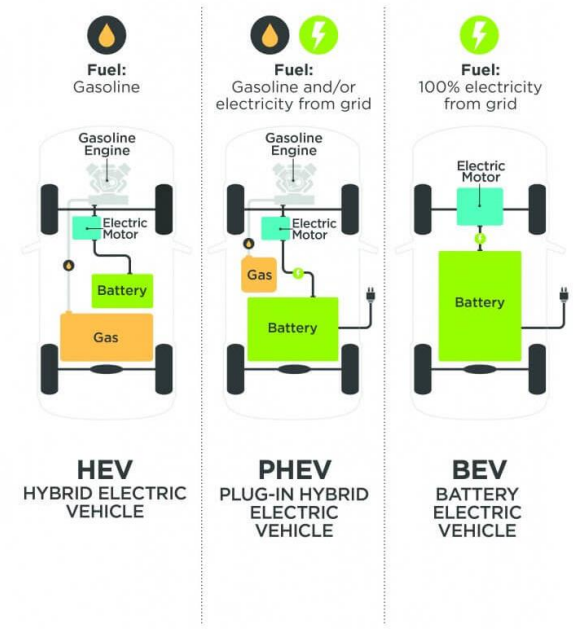
Cestovni promet koji troši preko 80% energije u ukupnom prometnom sektoru, prikazano na **Slika 2**, najviše počinje koristiti alternativne oblike energije kako bi se doprinijelo procesu dekarbonizacije prometnog sustava, (2).




**Slika 2.** Potrošnja energije prema načinu prijevoza, *(2)*

Među mnogim vrstama alternativnih oblika energije poput: vodika, biodizela, prirodnog plina, propana, etanola, električna energija se ističe kao najpraktičniji oblik energije za proizvodnju i najučinkovitiji za korištenje kod cestovnih vozila.

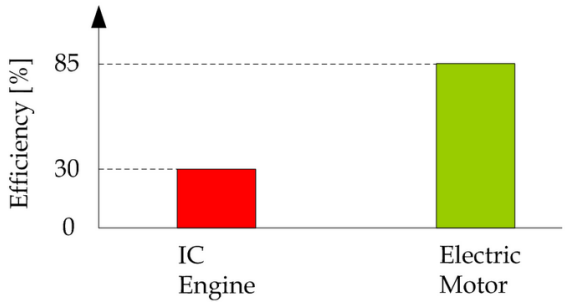
Električna vozila su prisutna još od kraja 19. stoljeća, no tek u posljednje vrijeme doživljavaju značajan porast popularnosti zbog povećanja isplativosti, ali i iznimno pozitivnog utjecaja na okoliš. Postoje tri glavne vrste električnih vozila. Prva vrsta su potpuno električna vozila koja koriste bateriju za pohranu električne energije. Baterije se pune priključivanjem vozila na izvor električne energije. Električna energija iz baterije pokreće električni motor vozila, čime se postiže potpuno električni pogon. Druga vrsta su plug-in hibridna električna vozila koja također koriste baterije za napajanje električnog motora. Osim električnog motora, ova vozila imaju i benzinski ili dizelski motor. Baterije se pune na isti način kao i kod potpuno električnih vozila, a električni motor može raditi samostalno ili u kombinaciji s benzinskim ili dizelskim motorom. Treća vrsta su hibridna električna vozila koja posjeduju klasični benzinski ili dizelski motor i dodatni električni motor. Električni motor koristi energiju pohranjenu u baterijama, ali se baterije ne pune putem vanjskog izvora električne energije. Kada se baterije isprazne, višak snage benzinskog ili dizelskog motora ili generatorsko kočenje se koriste za punjenje baterija, pri čemu se kinetička energija pretvara u električnu energiju. Podjela električnih vozila prikazana je na **Slika 3**, (7,8).




**Slika 3.** Vrste električnih vozila, *(9)*

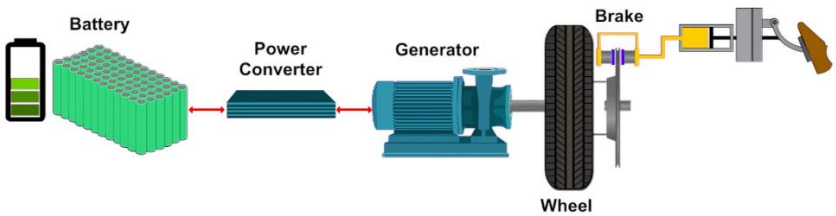
## 2.1 Učinkovitost električnih vozila

Električna vozila ne samo da pridonose procesu eliminacije emisije zagađivača, već imaju i brojne druge prednosti u odnosu na vozila s motorima na unutarnje izgaranje. Jedna od najvećih prednosti je učinkovitost elektromotora u usporedbi s motorima na unutarnje izgaranje. Elektromotori imaju minimalnu učinkovitost od oko 70%, a maksimalnu sve do 97% dok motori na unutarnje izgaranje imaju učinkovitost između 10% i 45%. Na **Slika 4** prikazana je prosječna učinkovitost motora na unutarnje izgaranje i elektromotora, (10).




**Slika 4.** Učinkovitost motora na unutarnje izgaranje i elektromotora, *(10)*

Energetska učinkovitost električnih vozila može se dodatno povećati primjenom generatorskog kočenja. Pri vožnji nizbrdo ili pri određenoj snazi kočenja, kinetička energija pretvara se u električnu energiju preko kotača i šalje se natrag u bateriju, čime se dodatno povećava domet vozila. Ovaj proces, prikazan na **Slika 5**, naziva se rekuperacija energije i to je još jedna dodatna prednost električnih vozila, (11).




**Slika 5.** Proces rekuperacija energije, *(11)*

## 2.2 Kapacitet baterije

Iako električna vozila imaju brojne prednosti u odnosu na vozila s motorima na unutarnje izgaranje, jedan veliki nedostatak je njihov ograničeni doseg vožnje. Uspoređujući prosječni doseg vožnje električnih vozila i vozila na konvencionalna goriva, današnja električna vozila imaju doseg od oko 430 km, dok vozila na konvencionalna goriva mogu prijeći u prosjeku oko 700 km prije nego što se moraju napuniti. Uz to, električna vozila se obično sporije pune od vozila na konvencionalna goriva. Primjerice, Hyundai Ioniq 5 je jedno od vozila koje najbrže puni svoju bateriju na trenutnom tržištu električnih vozila, gdje mu je potrebno 18 minuta punjenja na električnoj punionici snage od 350kW, da bi se postigao doseg vožnje od 245 km. Ako se to pretvori u doseg od 100 km, Ioniq-u 5 treba oko 5 minuta za punjenje, dok bi vozilu s motorom na unutarnje izgaranje bilo potrebno manje od minute da napuni spremnik gorivom za 100 km vožnje, (12). Daljnjim razvojem tehnologije, poboljšanju u proizvodnji baterija i efikasnijim iskorištavanjem kapaciteta baterije, doseg električnih vozila će se značajno povećati. To je već vidljivo s trenutnim najnaprednijim komercijalnim vozilom Lucid Air Grand Touring koje ima maksimalni domet od oko 830 km, (13).

Baterijski kapacitet električnih vozila izražava se u kilovat-satima (kWh), što zapravo predstavlja istu mjernu jedinicu za energiju kao i džul, pri čemu 1 kWh iznosi 3600 kilodžula (kJ), (14). Doseg električnih vozila usko je vezan uz kapacitet baterije, što znači da je prosječni doseg od 430 km zapravo odraz prosječnog kapaciteta baterije od 75 kWh, (12). Važno je istaknuti da baterije električnih vozila imaju ograničeni životni vijek. Većina baterija ima postavljenu granicu trajanja od oko 160.000 km, no proizvođači kontinuirano rade na povećanju tog broja iz godine u godinu. Također, bitno je spomenuti da baterije godišnje gube otprilike 2,3% svog kapaciteta uslijed procesa degradacije, (15). Ovaj proces degradacije rezultira smanjenjem ukupnog kapaciteta baterije tijekom vremena.

Mjerna jedinica kWh bit će značajna u daljnjoj izradi rada jer će se koristiti za rješavanje problema optimalnog energetskog puta, uz primjenu podataka o potrošnji energije izraženih u toj mjernoj jedinici.

## 2.3 Veza između električnih vozila, problema energetski optimalnog puta i ITS-a

Jedno moguće rješenje povećanja dosega vožnje električnih vozila je rutiranje takvih vozila prema energetski optimalnom putu. Rješavanje takvog problema direktno je vezano s područjem prometa nazvanim inteligentni prometni sustavi (ITS).

ITS predstavlja skup aplikacija, usluga i infrastrukturne opreme koji omogućuju upravljačku i informacijsko-komunikacijsku nadgradnju klasičnoga sustava prometa i transporta kojim se postiže poboljšanje odvijanje prometa, učinkovitiji prijevoz putnika i roba, poboljšanje sigurnosti u prometu, udobnost i zaštita putnika, manja onečišćenja okoliša, itd., (4). ITS obuhvaća 11 funkcionalnih područja, prikazanih u **Tablica 1**, i 32 temeljne usluge. Jedno od funkcionalnih područja je upravljanje prometom i operacijama, te se problem rutiranja vozila prema energetski optimalnom putu može se svrstati u to područje, (16).

**Tablica 1.** Prikaz funkcionalnih područja ITS-a, *(16)*

| Broj | Funkcionalno područje |
|---|---|
| **1** | Informiranje putnika |
| **2** | Upravljanje prometom i operacijama |
| **3** | Vozila |
| **4** | Prijevoz tereta |
| **5** | Javni prijevoz |
| **6** | Žurne službe |
| **7** | Elektronička plaćanja vezana uz transport |
| **8** | Sigurnost osoba u cestovnom prijevozu |
| **9** | Nadzor vremenskih uvjeta i okoliša |
| **10** | Upravljanje odazivom na velike nesreće |
| **11** | Nacionalna sigurnost i zaštita |

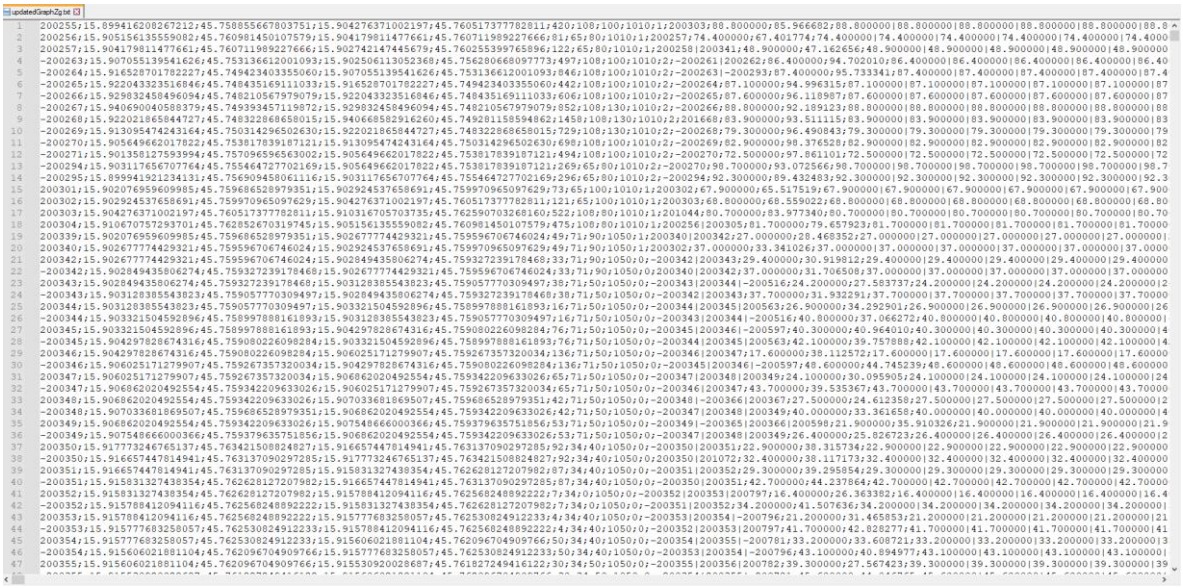
# 3 OPIS PODATAKA CESTOVNE MREŽE

Za rješavanje problema energetski optimalnog puta potrebni su određeni podatci. Za ovaj rad dodijeljen je dio podataka koji su prikupljeni u okviru projekta SORDITO. Podaci su prikupljeni tijekom petogodišnjeg razdoblja od kolovoza 2009. do listopada 2014. godine. U procesu prikupljanja podataka korišteno je ukupno 4908 vozila opremljenih navigacijskim uređajima. Navigacijski uređaji emitirali su signale približno svakih 100 metara dok su vozila bila u pokretu, te svakih 5 minuta kada su vozila bila zaustavljena. Kao rezultat tog prikupljanja, dobiveno je 6.55 milijardi zapisa o kretanju vozila s ukupnom pohranom podataka od 320 GB, (17). U ovom radu, razmatra se cestovna mreža koja obuhvaća širu okolicu grada Zagreba prikazanu na **Slika 6**, a dodijeljena .txt datoteka ima veličinu od 402 MB.

**Slika 6.** Razmatrana cestovna mreža

## 3.1 Struktura podataka

Prikupljeni GPS podaci strukturirani su po CSV (engl. *Coma Separated Values*) formatu. Isječak podataka prikazan je na **Slika 7**. Svaki redak u CSV datoteci predstavlja skup podataka o jednom linku koji označava jedan cestovni segment (link). Dodijeljena datoteka sadrži ukupno 43.992 redaka, odnosno cestovnih segmenata.



**Slika 7.** Isječka podataka iz .txt datoteke u CSV formatu

Podaci tj. atributi za pojedini link su u jednom retku međusobno odvojeni točka-zarezom (;) i prema redoslijedu su prikazani u **Tablica 2**.

**Tablica 2.** Popis atributa linka

| **Atribut** | **Opis** |
|---|---|
| $ID$ | ID cestovnog segmenta (cijeli broj) |
| $X_A$ | Geografska duljina početne točke linka (A) |
| $Y_A$ | Geografska širina početne točke linka (A) |
| $X_B$ | Geografska duljina početne točke linka (B) |
| $Y_B$ | Geografska širina početne točke linka (B) |
| $L$ | Duljina linka u metrima |
| $v_m$ | Brzina linka u km/h (statična brzina) postavljena od tvrtke Mireo d.d. |
| $v_{og}$ | Ograničenje brzine linka u km/h |
| $TL$ | Tip linka: 1010 su autoceste, a 1080 kvartovske male ceste, gradacija ide po 10 između tih vrijednosti |
| $ZS$ | Zastavica smjera: moguće vrijednosti 0,1,2 ili 3.<br>0 – po linku se može putovati u dva smjera – u tom slučaju u datoteci postoje dva retka za link, jedan za pozitivni i jedan za negativni ID linka.<br>1 – po linku se putuje u jednom smjeru, a ID linka je pozitivna vrijednost, i to od točke A prema točci B<br>2- po linku se putuje u jednom smjeru, a ID link je negativne vrijednosti, i to od točke B prema točci A<br>3- sve isto kao i za vrijednost 0, ali 3 predstavlja da je cesta zbog nekog razloga zatvorena |
| $SL$ | Susjedni linkovi: između dvije susjedne točke-zarez (;) nalazi se lista susjednih linkova, pri čemu su ID-evi linkova međusobno odvojeni vertikalnom crtom (\|) |

| | |
|---|---|
| $v_{free}$ | Brzina slobodnog toka u km/h |
| $v_{avg}$ | Prosječna brzine linka u km/h |
| $P_v$ | Profil brzine u km/h: između dvije susjedne točke-zarez (;) nalazi se 288 vrijednosti brzina međusobno odvojenih vertikalnom crtom (\|). Svaka od vrijednosti predstavlja prosječnu brzinu unutar pojedinog peto-minutnog intervala. Prva vrijednost predstavlja brzinu unutar intervala 00:00-00:05, druga unutar intervala 00:05-00:10, a zadnja unutar intervala 23:55-00:00. |
| $E_{avg}$ | Prosječna potrošnja energije na linku u kWh |
| $P_e$ | Profil energije u kWh: između dvije susjedne točke-zarez (;) nalazi se 288 vrijednosti potrošnje energije međusobno odvojene vertikalnom crtom (\|). Svaka od vrijednosti predstavlja prosječnu potrošnju energije unutar pojedinog peto-minutnog intervala. Prva vrijednost predstavlja potrošnju energije unutar intervala 00:00-00:05, druga unutar intervala 00:05-00:10, a zadnja unutar intervala 23:55-00:00. |

## 3.2 Izračun potrošnje energije

Budući da pri prikupljanju podataka nije bilo dostupnih informacija o potrošnji energije, potrebno je izvršiti njezinu procjenu. Za tu svrhu koristi se jednadžba prikazana na **Slika 8**, koja omogućuje izračun sile F potrebne za ubrzanje vozila i savladavanje otpora nagiba, kotrljanja i zraka. Parametri koji se koriste u jednadžbi opisani su u **Tablica 3**. Ako je rezultirajuća sila $F \geq 0$, to znači da vozilo ubrzava i potrebna mu je snaga za kretanje. S druge strane, ako je $F < 0$, dolazi do usporavanja (kočenja) ili vožnje nizbrdo, pri čemu se energija vraća u bateriju, (17).

$$F = \underbrace{mg\sin\alpha}_{\text{Nagib}} + \underbrace{c_r mg\cos\alpha}_{\text{Kotrljanje}} + \underbrace{0.5c_d\rho Av^2}_{\text{Zrak}} + \underbrace{ma}_{\text{Ubrz.}}$$

**Slika 8.** Jednadžba za izračun sile, *(18)*

**Tablica 3.** Parametri jednadžbe za izračun sile, *(19)*

| Oznaka | Jedinica | Opis |
|---|---|---|
| $m$ | $kg$ | Masa vozila (praznog) |
| $a$ | $m/s^2$ | Ubrzanje vozila |
| $v$ | $m/s$ | Brzina vozila |
| $g$ | $m/s^2$ | Gravitacijska konstanta |
| $f$ | | Faktor mase rotirajućih dijelova vozila |

| | | |
|---|---|---|
| $\alpha$ | $rad$ | Nagib prometnice |
| $c_{rr}$ | | Koeficijent trenja kotrljanja |
| $c\omega$ | | Koeficijent otpora zraka |
| $\rho$ | $kg/m^3$ | Gustoća zraka |
| $A$ | $m^2$ | Prednja površina vozila |

Da bi se izračunala potrošnja energije, potrebno je uzeti u obzir specifične parametre za odabrani električni automobil. U ovom slučaju, kao primjer vozila koristi se Nissan Leaf 2014, čije tehničke specifikacije su prikazane u **Tablica 4**.

**Tablica 4.** Karakteristike vozila Nissan Leaf 2014, *(17)*

| **Opis** | **Oznaka** | **Vrijednost** |
|---|---|---|
| Kapacitet baterije | $Q$ | 24 $kWh$ |
| Masa | $m$ | 1145 $kg$ |
| Koeficijent trenja kotrljanja | $c_{rr}$ | 0.008 |
| Koeficijent otpora zraka | $c\omega$ | 0.35 |
| Prednja površina vozila | $A$ | 1.9 $m^2$ |
| Faktor mase rotirajućih dijelova vozila | $f$ | 1.01 |
| Koeficijent motora i pogonskog sklopa | $\mu_m$ | 0.9 |
| Koeficijent između pogona i motora | $\mu_g$ | 0.8 |
| Pomoćna snaga | $P_0$ | 450 $W$ |
| Minimalna brzina za vraćanje energije | $v_{min}$ | 10 $km/h$ |

Gustoća zraka $\rho$ je postavljena na vrijednost od 1.2 kg/m$^3$, dok je gravitacijsko ubrzanje $g$ postavljeno na 9.81 m/s$^2$. Vrijednosti $v$, $a$ i $\alpha$ nisu povezane s tehničkim specifikacijama vozila, već s karakteristikama samog cestovnog segmenta. Brzine $v$ koje se koriste u izračunu temelje se na prikupljenim brzinama tijekom dana u projektu SORDITO. Za akceleraciju, vrijednosti su preuzete iz rada za vozilo Nissan Leaf, pri čemu su vrijednosti dodijeljene prema odgovarajućim intervalima brzine, kao što je vidljivo u **Tablica 5**. Za izračun nagiba prometnih segmenata koriste se digitalni elevacijski podaci Europske službe Copernicus. Svaki cestovni segment ima određenu razliku u nadmorskoj

visini ($\Delta h$) i zračnu udaljenost ($d$). Nagib se tada izračunava primjenom jednadžbe: $\alpha = arctan\frac{\Delta h}{d}$, (17).

**Tablica 5.** Akceleracija prema intervalima brzine za Nissan Leaf 2014, *(17)*

| Interval brzine $[km/h]$ | Akceleracija $[m/s^2]$ |
|---|---|
| $[0, 30\rangle$ | 0.61 |
| $[30, 51\rangle$ | 0.53 |
| $[51, 72\rangle$ | 0.37 |
| $[72, 93\rangle$ | 0.41 |
| $[93, 102\rangle$ | 0.28 |
| $[102, \infty\rangle$ | 0.05 |

Nakon što se dobije rezultirajuća sila, moguće je izračunati snagu baterije prema jednadžbi prikazanoj na **Slika 9**, gdje $\mu_m$ predstavlja koeficijent prijenosa između elektromotora i pogonskog sustava, dok je $\mu_g$ omjer pretvorbe mehaničke energije na kotačima u kemijsku energiju pohranjenu u bateriji. Važno je napomenuti da se energija vraća u bateriju samo ako je sila F manja od nule, a brzina veća od $v_{min}$. Kada je izračunata snaga baterije, moguće je izračunati trenutnu potrošnju energije $E$ prema jednadžbi $E = P_b \Delta t$, gdje $\Delta t$ predstavlja vrijeme putovanja na cestovnom segmentu, (17).

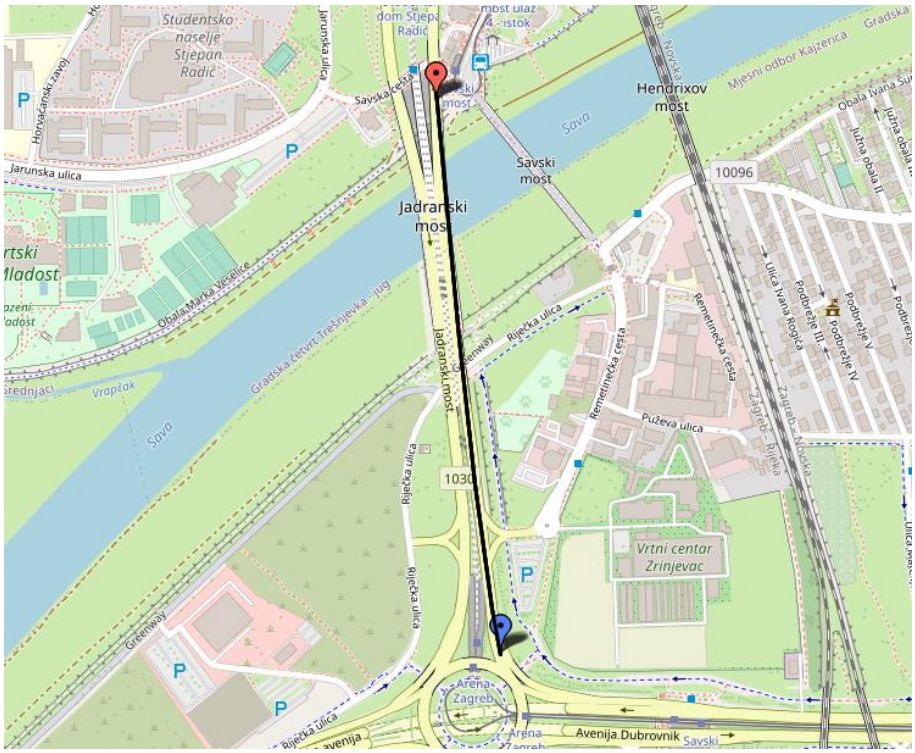
$$P_b = \begin{cases} \mu_e(\mu_m F v + P_0), & \text{if } F \geq 0 \\ \begin{cases} 0, & \text{if } v \leq v_{min} \\ F v \mu_g + P_0, & \text{else} \end{cases}, & \text{if } F < 0 \end{cases}$$

**Slika 9.** Jednadžba za izračun snage baterije, *(18)*

## 3.3 Primjer cestovnog segmenta

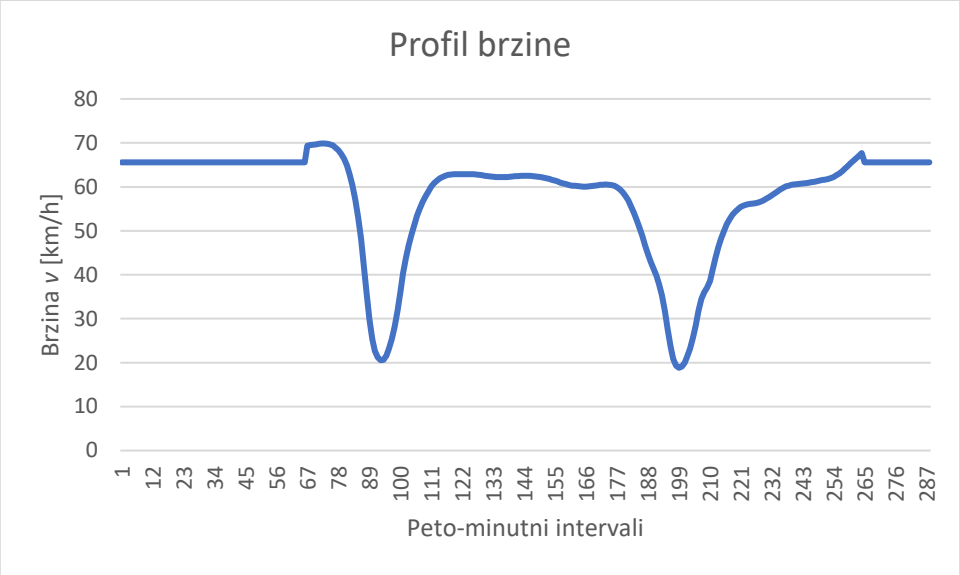
U svrhu lakšeg razumijevanja, prikazan je Jadranski most na **Slika 10** kao primjer cestovnog segmenta. U datoteci taj segment je identificiran pod ID brojem -214695. Duljina tog cestovnog segmenta iznosi 340 metara, a postavljeno ograničenje brzine je 60

km/h. Tip cestovnog segmenta s oznakom 1040 ukazuje da se radi o glavnoj cesti, a zastavica smjera s brojem 2 označava da se putovanje tim segmentom odvija samo u jednom smjeru. Također, segment ima samo jedan susjedni link koji je identificiran brojem -214694. Prosječna brzina kretanja vozila na Jadranskom mostu iznosi 53.89 km/h, dok je prosječna potrošnja energije 0.072 kWh.

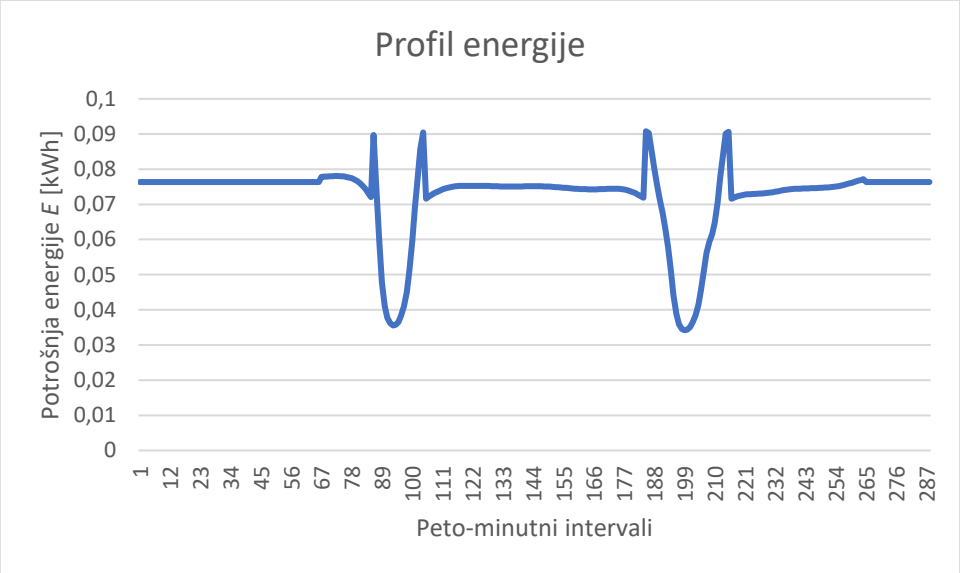



**Slika 10.** Primjer cestovnog segmenta -214695

U sklopu analize bitno je spomenuti profil brzine i profil potrošnje energije koji se sastoje od 288 vrijednosti, pri čemu svaka vrijednost prikazuje prosječnu brzinu ili prosječnu potrošnju energije u peto-minutnom intervalu. Na **Slika 11** prikazan je profil brzine na Jadranskom mostu, gdje se mogu uočiti padovi brzine oko 90. i 200. vrijednosti. Ove vrijednosti mogu se točno pretvoriti u vremenske oznake, koje odgovaraju vršnim satima prometa, odnosno vremenima oko 7:30 i 16:30 sati. U tim vremenskim periodima često dolazi do prometnih zagušenja, što rezultira sporijim kretanjem vozila u usporedbi s brzinom vožnje u slobodnom toku. Slično tome, na **Slika 12** koja prikazuje profil energetske potrošnje, može se primijetiti da se za ista vremenska razdoblja energetska potrošnja smanjuje. To je posljedica sporijeg kretanja vozila uslijed prometnih zagušenja u vršnim satima, što rezultira manjom potrošnjom energije.




**Slika 11.** Profil brzine na Jadranskom mostu



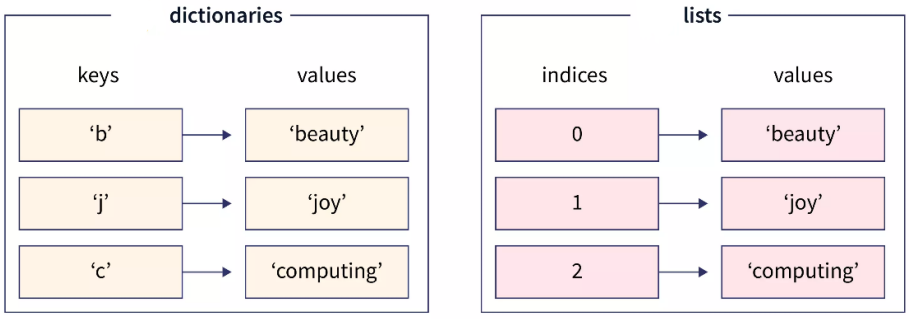

**Slika 12.** Profil energije na Jadranskom mostu

## 3.4 Učitavanje podataka

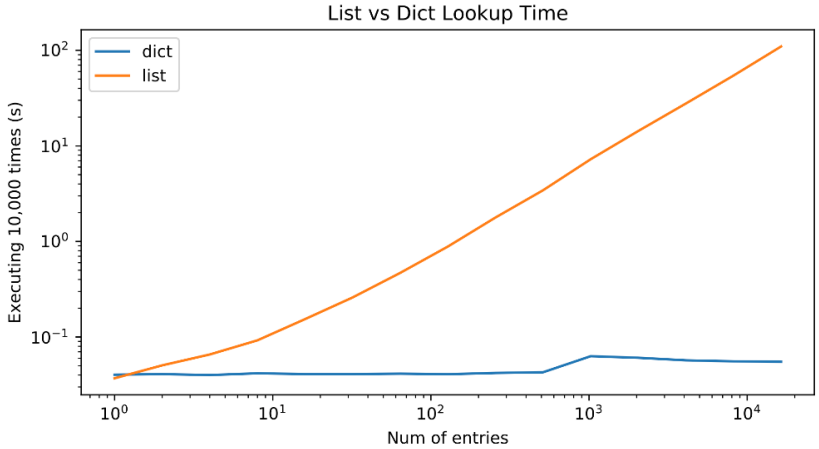
Za učitavanje podataka korišten je softverski paketa Visual Studio 2022, i programski jezik C#. Prvi korak je stvaranje klase koja će obuhvaćati sve atribute navedene u tablici 2. Nakon stvaranja klase potrebno je uspostaviti pristup čitanju podacima iz .txt datoteke. U C#-u se za čitanje podataka iz datoteke koristi klasa StreamReader kojoj se pristupa preko

imenskog prostora „using System.IO“. Sljedeći korak je učitavanje svih redaka datoteke i stvaranja objekta klase s odgovarajućim vrijednostima atributa za svaki redak datoteke.

Svaki stvoreni objekt klase potrebno je spremiti. Postoje više struktura podataka koje se koriste za spremanje objekata klase, ali možda dvije najpoznatije su- lista i mapa (engl. Dictionary). Za razliku od liste gdje se elementima pristupa preko indeksa, mapa koristi jedinstveni ključ za svaki element preko kojeg im se pristupa, (20). Usporedba pristupa elementima u listi i mapi prikazana je na **Slika 13**. Zbog toga što mapa ima jedinstveni ključ za svaki element spremljen u posebnim strukturama kao što su stabla i sl., pretraživanje određenog elementa u mapi je mnogo brže u usporedbi s listom, što je prikazano na **Slika 14**. Dodavanje/ažuriranje elementa u mapu može trajati duže u odnosnu na dodavanje istog u listu, ali za primjenu u ovome radu, puno je veći naglasak na brzom dohvatu podataka Stoga se u ovom radu objekti klase spremaju u mapu radi značajno poboljšanih performansi programa, (21).




**Slika 13.** Usporedba pristupa elementima u listi i mape, *(20)*



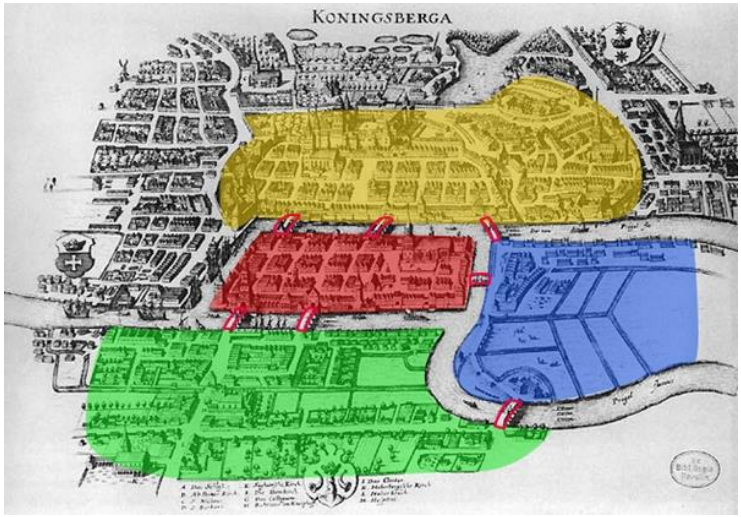

**Slika 14.** Razlika u vremenu pretraživanja liste i mape, *(21)*

# 4 PRONALAZAK OPTIMALNOG PUTA NA GRAFU

Prometna mreža se može promatrati kao skup cesta i raskrižja tj. čvorova i bridova koji zapravo čine graf. Za pronalazak optimalnog puta na grafu potrebno je poznavanje teorije grafova.

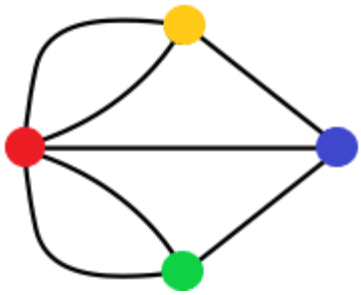
## 4.1 Problem sedam mostova Königsberga

U 18. stoljeću u gradu Königsbergu u Pruskoj, postojalo je sedam mostova preko rijeke koja je prolazila kroz grad, što je prikazano na **Slika 15**. Građani Königsberga osmislili su igru čiji cilj je bio pronaći rutu gradom koja bi uključivala prelazak preko svakog mosta samo jednom te povratak na mjesto iz kojeg se krenulo. Ta igra je zaintrigirala švicarskog matematičara Leonhard Eulera koji je 1736. godine riješio problem sedam mostova Königsberga, (22).




**Slika 15.** Raspored mostova Königsberga, *(22)*

Euler je pretpostavio da je izbor rute unutar kopnene površine irelevantan te da je jedino važan redoslijed mostova preko kojih se prelazi. Ta pretpostavka mu je omogućila da eliminira sve značajke osim kopnenih površina i mostova. Time je Euler topologiju grada pretvorio u graf, prikazano na **Slika 16**, gdje su kopnene površine na grafu prikazane kao čvorovi tj. vrhovi, a mostovi kao bridovi, (23).

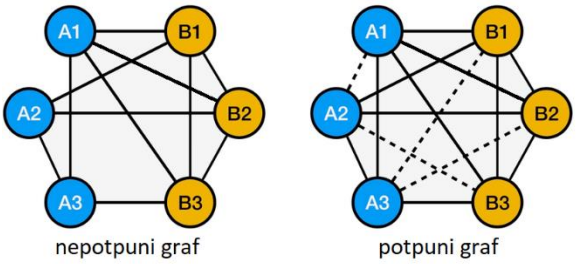


**Slika 16.** Prikaz problema sedam mostova Königsberga pomoću grafa, *(23)*

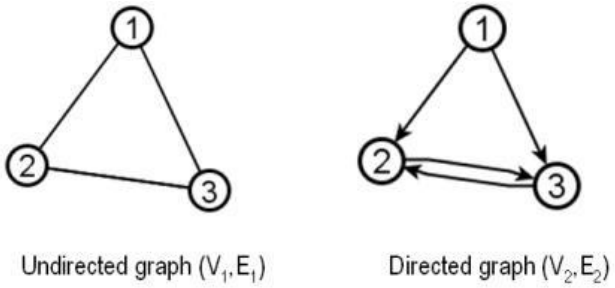
Euler je zaključio da mogućnost prolaska kroz graf ovisi o stupnjevima čvorova gdje je stupanj čvora broj bridova koji ga dodiruju. Time je dokazao da željeni put postoji samo ako graf ima niti jedan ili dva čvora neparnog stupnja. Takav put se naziva Eulerov put, a budući da graf koji odgovara problemu sedam mostova Königsberga ima četiri čvora neparnog stupnja, ne može postojati Eulerov put. Smatra se da je problem sedam mostova Königsberga bio prvi problem teorije grafova, (23).

## 4.2 Teorija grafova

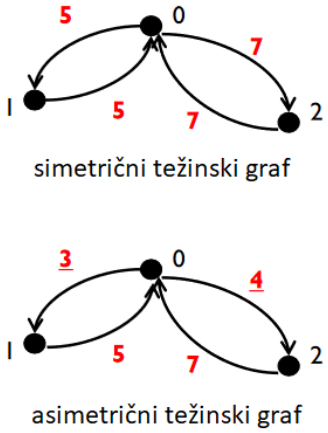
Grana matematike koja se bavi proučavanjem grafova naziva se teorija grafova. Graf se u kontekstu teorije grafova sastoji od skupa vrhova i skupa bridovima. Grafovi se mogu podijeliti na različite vrste kao što su: potpuni i nepotpuni, usmjereni i neusmjereni te težinski simetrični i asimetrični grafovi. Podjela grafova prikazana je na **Slika 17**, **Slika 18** i**Slika 19**. U potpunom grafu, svaki čvor je povezan sa svim ostalim čvorovima bridom, dok u nepotpunom grafu neke veze ne postoje. U usmjerenom grafu brid se promatra kao dvije veze od vrha A do vrha B gdje se svakoj vezi može pridijeliti njena orijentacija (A->B, B->A), dok se kod neusmjerenog grafa brid promatra kao jedna veza koja nema orijentaciju. Ako se govori o usmjerenom grafu onda taj graf može biti težinski simetričan ili asimetričan. U težinski simetričnom grafu težina bridova između dva vrha uvijek je jednaka bez obzira na smjer u kojem se promatra. S druge strane, u težinski asimetričnom grafu težine bridova između dva vrha su različite ovisno o smjeru u kojem se promatraju. Kod neusmjerenog grafa težinski graf može biti samo simetričan, (24,25).




**Slika 17.** Prikaz potpunog i nepotpunog grafa, *(26)*




**Slika 18.** Prikaz usmjerenog i neusmjerenog grafa, *(27)*




**Slika 19.** Prikaz simetričnog i asimetričnog grafa, *(25)*

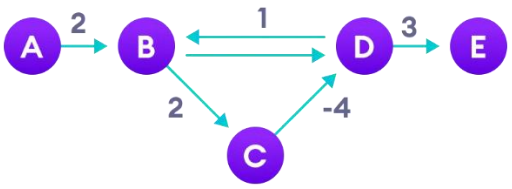
## 4.3 Algoritam za pronalazak optimalnog puta

Jedan od problema koji se proučavaju u teoriji grafova je pronalazak optimalnog puta između zadanih vrhova grafa. Dijkstrin algoritam je najpoznatiji algoritam za pronalazak optimalnog puta od jednog vrha prema svim ostalim vrhovima. Za pronalazak optimalnog

puta Dijkstrin algoritam koristi postavljene težine bridova. Međutim, kada se radi o rješavanju problema energetski optimalnog puta, težine bridova se izražavaju kao potrošnja energije koja može biti negativna jer električna vozila mogu regenerirati energiju i vraćati ju u bateriju. To predstavlja problem za Dijkstrin algoritam koji radi samo s pozitivnim vrijednostima težina, (28).

### 4.3.1 Bellman-Ford algoritam

Pri izvođenju Dijkstrinog algoritma na grafu s negativnim težinama postoji opasnost od dobivanja krivog rješenja zbog negativnih ciklusa. Na primjer, na **Slika 20** prikazan je negativan ciklus gdje bi Dijkstrin algoritam krenuo iz vrha A(0) (u zagradi je težina vrha) prema vrhu B(2) te bi algoritam imao opciju izabrati sljedeći vrh C(4) ili D(3). Kako vrh D ima manju težinu od vrha C algoritam bira vrh D(3) i nastavlja do vrha E gdje je krajnja težina 6. Međutim, ovdje se može uočiti pogreška u rješenju jer da je izabrani vrh bio C(4), put do vrha D bio bi -4, što bi težinu vrha D smanjilo na 0. Time bi krajnja težina vrha E bila 3, što je manje od rješenja dobivenog Dijkstrinim algoritmom, (28).




**Slika 20.** Prikaz negativnog ciklusa, *(28)*

Da bi se izbjegle pogreške koje se mogu pojaviti pri izvođenju Dijkstrinog algoritma na grafu s negativnim težinama, potrebno je graf pretvoriti u graf s pozitivnim težinama. To se može postići primjenom Johnsonove tehnike i primjenom Bellman-Ford algoritma. Postupak je prikazan Algoritmom 1.

Bellman-Ford algoritam vrlo je sličan Dijkstrinom algoritmu, ali se razlikuje u načinu na koji obrađuje vrhove. Dok Dijkstrin algoritam pamti neobrađene vrhove i obrađuje samo njih, Bellman-Ford algoritam prolazi kroz sve vrhove u svakoj iteraciji. Iz tog razloga

Bellman-Ford algoritam ima mogućnost rada s grafom koji ima negativne težine. Međutim, njegova složenost je $O(E * V)$, što je veće od složenosti Dijkstrinog algoritma koja iznosi $O(E + V \log V)$. Ovdje E predstavlja broj bridova, a V broj vrhova u grafu. Iako ima veću složenost, Bellman-Ford algoritam je potrebno samo jednom pokrenuti kako bi se graf pripremio za Dijkstrin algoritam, (17).

Primjenom Johnsonove tehnike u graf se dodaje novi vrh $Q$ povezan sa svim ostalim vrhovima bridom težine 0. Zatim se nad tim novim grafom pokreće Bellman-Ford algoritam počevši od vrha q, a rezultat algoritma su putevi minimalne težine od vrha q do svakog drugog vrha u grafu. Ove vrijednosti se spremaju u atribut klase svakog vrha kako bi se mogle dalje koristiti u Dijkstrinom algoritmu, a vrh q i njegovi bridovi se potom izbacuju iz grafa. Na ovaj način se graf pretvara u graf s pozitivnim težinama i spreman je za daljnje korištenje Dijkstrinim algoritmom, (17,28,29).

**Algoritam 1.** Bellman-Ford algorithm with Johnson's technique, (17)

**Input:** Graph *G*, start vertex $v_0$

```
 1: Q ← Initialize list of unprocessed vertices
 2: Add v0 to G
 3: for each vertex v in graph G do
 4:    if v = v0 then
 5:       value(v) ← 0
 6:    else
 7:       value(v) ← ∞
 8:       neighbor(v0 )← v
 9:    end if
10:    preceding(v) ← None
11:    Add v to Q
12: end for
13: while Q is not empty do
14:    u ← Remove vertex form Q with the lowest value
15:    for each neighboring vertex v of vertex u do
16:       val_new ← value(u) + w_u,v
17:       if val_new < value(v) then
18:          value(v) ← val_new
19:          preceding(v) ← u
20:       end if
21:    end for
22: end while
23: Remove v0 from G
```

### 4.3.2 Dijkstra algoritam

Kada je graf pretvoren u graf s pozitivnim težinama može se pokrenut algoritam nazvan po nizozemskom računalnom znanstveniku Edsgeru W. Dijkstri koji ga je opisao 1959. godine, (30).

Algoritam je prikazan Algoritmom 2. i počinje tako što se svi vrhovi grafa dodaju na popis neobrađenih vrhova. Zatim se početni vrh postavlja kao polazište pretraživanja grafa te se njegova težina postavlja na vrijednost 0 dok se svim ostalim vrhovima težina postavlja na beskonačnu vrijednost. Također, atribut prethodni vrh svakog vrha u grafu postavlja se na ništa, engl. *null*. U svakoj iteraciji algoritam prolazi kroz sve susjedne vrhove trenutnog vrha koji se nalaze na popisu neobrađenih vrhova. Za svaki susjedni vrh računa se ukupna težina do njega koja se sastoji od težine trenutnog vrha, težine brida koji povezuje trenutni i susjedni vrh i razlici vrijednosti trenutnog i susjednog vrha dobivenih u Bellman-Ford algoritmu. Ako je nova težina manja od trenutne težine do susjednog vrha, ažurira se vrijednost težine do susjednog vrha i u atribut prethodni vrh se upisuje trenutni vrh . Nakon što su svi susjedni vrhovi obrađeni, trenutni vrh se izbacuje iz popisa neobrađenih vrhova i traži se novi neobrađeni vrh s najmanjom težinom prikazan Algoritmom 3. Postupak se ponavlja sve dok skup neobrađenih vrhova nije prazan. Kao rezultat Dijkstrinog algoritma dobiva se najkraća težinska udaljenost od početnog vrha do svakog drugog vrha u grafu. Kako bi se prikazao put od početnog vrha do bilo kojeg drugog vrha, može se koristiti petlja koja počinje od tog drugog vrha i iterativno se kreće prema početnom vrhu prikazana Algoritmom 4. U svakoj iteraciji, petlja se pomiče na prethodni vrh trenutnog vrha sve dok atribut prethodni vrh ne dođe do vrijednosti null koju samo ima početni vrh. Na taj način se može dobiti cijeli put od početnog vrha do odabranog vrha, tako da se vrhovi koji su prolazili kroz petlju dodaju u listu putanje. **Slika 21** prikazuje vizualni primjer rezultata dobivenog primjenom Dijkstrinog algoritma te algoritma za prikazivanja puta između dva vrha grafa, (17,30).

**Algoritam 2.** Energy-dependent Dijkstra algorithm, (17)

**Input:** Graph *G*, start vertex $v_0$

```
Q ← Initialize list of unprocessed vertices
for each vertex v in graph G do
    if v = v0 then
        value(v) ← 0
    else
        value(v) ← ∞
    end if
    preceding(v) ← None
    Add v to Q
end for
while Q is not empty do
    u ← Remove vertex form Q with the lowest value
    for each unprocessed neighboring vertex v of vertex u do
        val_new ← value(u) + w_u,v + BF_value(u) − BF_value(v)
        if val_new < value(v) then
            value(v) ← val_new
            preceding(v) ← u
        end if
    end for
end while
```

**Algoritam 3.** Remove vertex with the lowest value algorithm, (19)

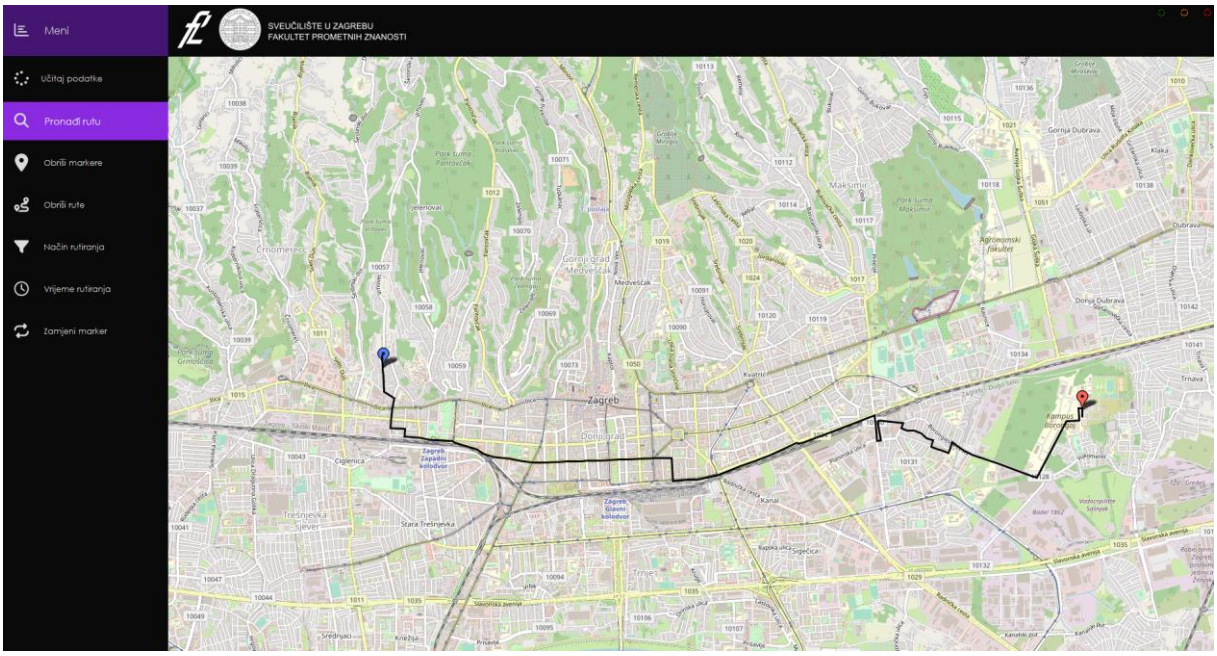
**Input:** List of unprocessed vertices *Q*

```
value_min ← ∞
u ← None
for each vertex v in list Q do
    if value(v) < value_min then
        value_min = value(v)
        u ← v
    end if
end for
Return u
```

**Algoritam 4.** Dohvaćanje putanje, (19)

**Input:** finsih vertex $v_{end}$

```
Q ← Initialize list of vertices
v ← v_end
while preceding(v) ≠ null do
    Add v to Q
    v ← preceding(v)
end while
```




**Slika 21.** Vizualni prikaz rezultata dobivenog Dijkstrinim algoritmom

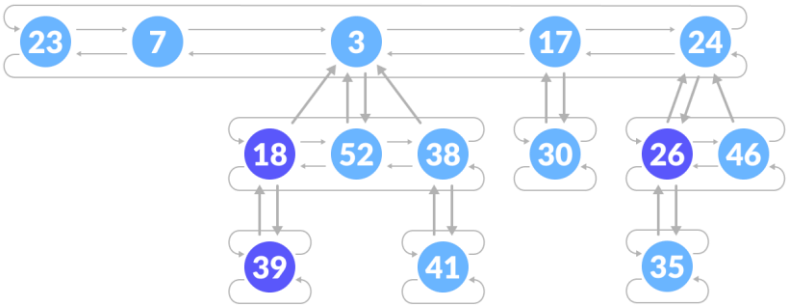
Dijkstrin algoritam se najčešće implementira korištenjem nesortiranog polja za popis neobrađenih vrhova. Međutim, ova implementacija ima složenost od $O(n)$ za pronalaženje i izbacivanje vrha s najmanjom težinom iz polja, što značajno usporava algoritam jer se u svakoj iteraciji traži novi neobrađeni vrh s najmanjom težinom. Zbog toga je u ovom radu korištena implementacija drugačije podatkovne strukture polja koja omogućuje brži pristup neobrađenom vrhu s najmanjom težinom, (17,30). Koristeći navedenu podatkovnu strukturu postiže se složenost algoritma $O(E + V \log V)$. No ako se radi o grafu slabe gustoće u kojem je $E \approx V$, kakva je razmatran u ovom radu jer jedan vrh u prosjeku ima 2.53 izlaznih bridova, dolazi se do složenosti $O(V\ log\ V)$, (17).

# 5 FIBONACCIJEVA HRPA

Da bi se ubrzao rad i smanjila sveukupna složenost Dijkstrinog algoritma potrebno je riješiti problem uskog grla koji nastaje pronalaženjem i izbacivanjem vrha s najmanjom težinom iz nesortiranog polja za skup neobrađenih vrhova. Najbolja implementacija koja omogućuje najmanju složenost i najbrži rad Dijkstrinog algoritma koristi Fibonacci Heap strukturu podataka engl. *Fibonacci Heap data structure*.

## 5.1 Struktura Fibonaccijeve hrpe

Fibonaccijeva hrpa se sastoji od kolekcije stabala pri čemu svako stablo zadovoljava svojstvo da je vrijednost djeteta uvijek veća ili jednaka vrijednosti roditelja. Ta stabla su međusobno povezana kroz kružnu dvostruko povezanu listu, prikazano na **Slika 22**, (31).

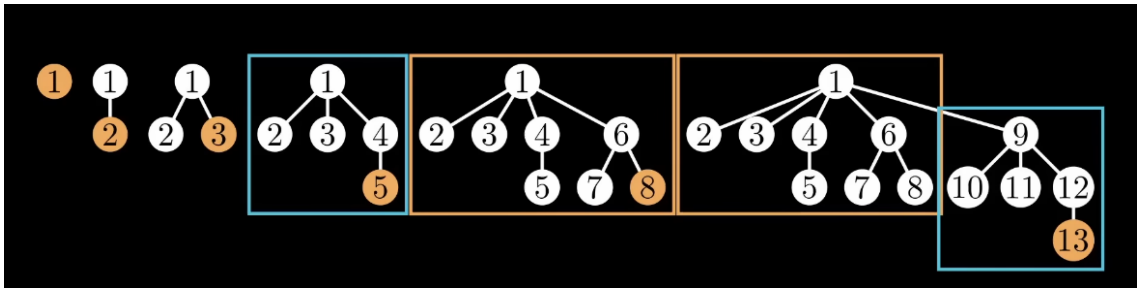



**Slika 22.** Fibonacci Heap struktura, *(31)*

Svako stablo u Fibonaccijevoj hrpi se sastoji od čvorova koji sadrže vrijednosti i reference na svoje dijete, roditelja i brata. Fleksibilnost strukture Fibonaccijeve hrpe proizlazi iz činjenice da stabla nemaju propisan oblik, što omogućuje da se neke operacije izvode na "lijen" način, čime se smanjuje složenost operacija. Međutim, u određenom trenutku hrpu je potrebno uvesti u red kako bi se održalo brzo vrijeme rada algoritma.

Nakon uvođenja hrpe u red, primjećuje se prema **Slika 23** da je broj elemenata stabla stupnja n jednak zbroju elemenata stabla stupnja n-1 i n-2. Ovi brojevi koji se dobivaju

zbrajanjem prethodna dva nazivaju se Fibonaccijevi brojevi, te je iz te povezanosti proizašao naziv Fibonaccijeva hrpa, (32).



**Slika 23.** Pojava Fibonaccijevih brojeva u uređenoj Fibonaccijevoj hrpi, *(32)*

## 5.2 Operacije na Fibonaccijevoj hrpi

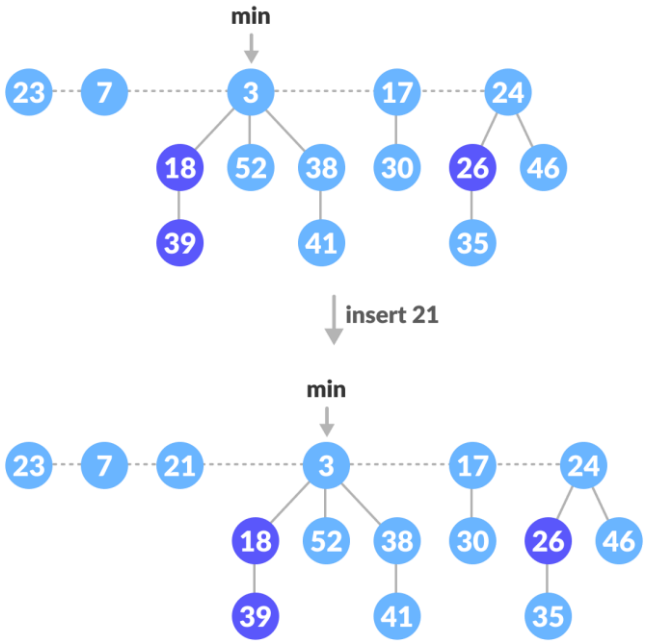
Povezanost stabala kroz kružnu dvostruko povezanu listu omogućuje određenim operacijama vrlo kratko vrijeme izvedbe.

Glavne operacije na Fibonaccijevoj hrpi su:

- Umetanje, engl. *Insert,*
- Pronađi minimum, engl. F*ind Min,*
- Spajanje, engl. *Union,*
- Izbaci minimum, engl. E*xtract Min,*
- Smanji vrijednost, engl. *Decrease Key,* (31).

### 5.2.1 Operacija Insert

Prvi korak u operaciji insert je stvaranje novog čvora. Nakon toga se provjera je li hrpa prazna. Ako je hrpa prazna, novi čvor se postavlja kao korijenski čvor i označava se kao minimum. U suprotnom, stvoreni čvor se dodaje u popis korijena i ažurira se minimalna vrijednost. Operacija insert prikazana je na **Slika 24** gdje se ubacuje vrijednost 21 u Fibonaccijevu hrpu. Složenost operacije insert je $O(1)$, (31).



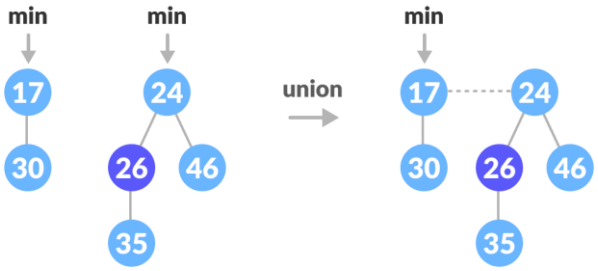

**Slika 24.** Operacija insert, *(31)*

### 5.2.2 Operacija Find Min

Fibonaccijeva hrpa omogućava brzo dohvaćanje čvora s minimalnom vrijednosti jer hrpa uvijek ima pokazivač na taj čvor. Stoga se operacija *find min* izvršava jednom linijom koda, tj. dohvatom pokazivača na minimalni čvor. Ova operacija je vrlo jednostavna i izvodi se u $O(1)$ vremenskoj složenosti, (31).

### 5.2.3 Operacija Union

Spajanje dva ili više stabala moguće je pomoću operacije *union*. Prvi korak operacije je spajanje korijena svih stabala. Nakon toga se ažurira pokazivač minimalne vrijednosti prolaskom kroz popis korijena i pronalaska minimuma. Ova operacija prikazana je na **Slika 25** gdje se spajaju dva stabla. Korijeni stabala 17 i 24 se spajaju te se pokazivač minimalne vrijednosti stavlja na čvor vrijednosti 17. Operacija union izvodi se u $O(1)$ vremenskoj složenosti, (31).




**Slika 25.** Operacija union, *(31)*

### 5.2.4 Operacija Extract Min

Jedan od koraka Dijkstrinog algoritma je pronalaženje i izbacivanje vrha s najmanjom težinom iz polja. Operacija *extract min* će obavljati točno tu funkciju. Rad operacije objašnjen je kroz primjer i korake prikazane na **Slika 26**.

Operacija *extract min* pronalazi i izbacuje vrh s najmanjom vrijednosti iz hrpe, ali obavlja i vrlo bitnu operaciju prilagođavanja hrpe.

a) Vidljiv je početni izgled Fibonaccijeve hrpe.
b) Čvor s minimalnom vrijednosti se briše, a njegova djeca se dodaju u popis korijena. Pokazivač minimuma se postavlja na sljedeći korijen u popisu. U primjeru čvor 3 se briše, njegova djeca 18, 52, 38 dodaju se u listu korijena te se pokazivač stavlja na čvor 17.
c) U ovom koraku se korijeni sortiraju prema njihovom stupnju, tj. broju djece, kako bi se stvorilo stablo gdje svaki korijen ima jedinstveni stupanj. Korijeni 23, 7, 21, 52 grupiraju se u stupanj 0, korijeni 18, 38, 17 u stupanj 1, a korijen 24 u stupanj 2.
d) Kroz sljedeće korake se korijeni spajaju da se postigne stablo s korijenima jedinstvenih stupnjeva. Korijeni 7 i 23 se spajaju jer imaju isti stupanj (0), čvor 7 dodaje se u popis korijena, a čvor 23 dodaje se kao dijete čvoru 7 jer ima veću vrijednost. Ažurira se stupanj čvora 7 na vrijednost 1 jer sada ima jedno dijete.
e) Korijeni 7 i 17 spajaju se jer imaju isti stupanj (1). Čvor 17 postaje dijete jer ima veću vrijednost. Stupanj čvora 7 postaje 2.

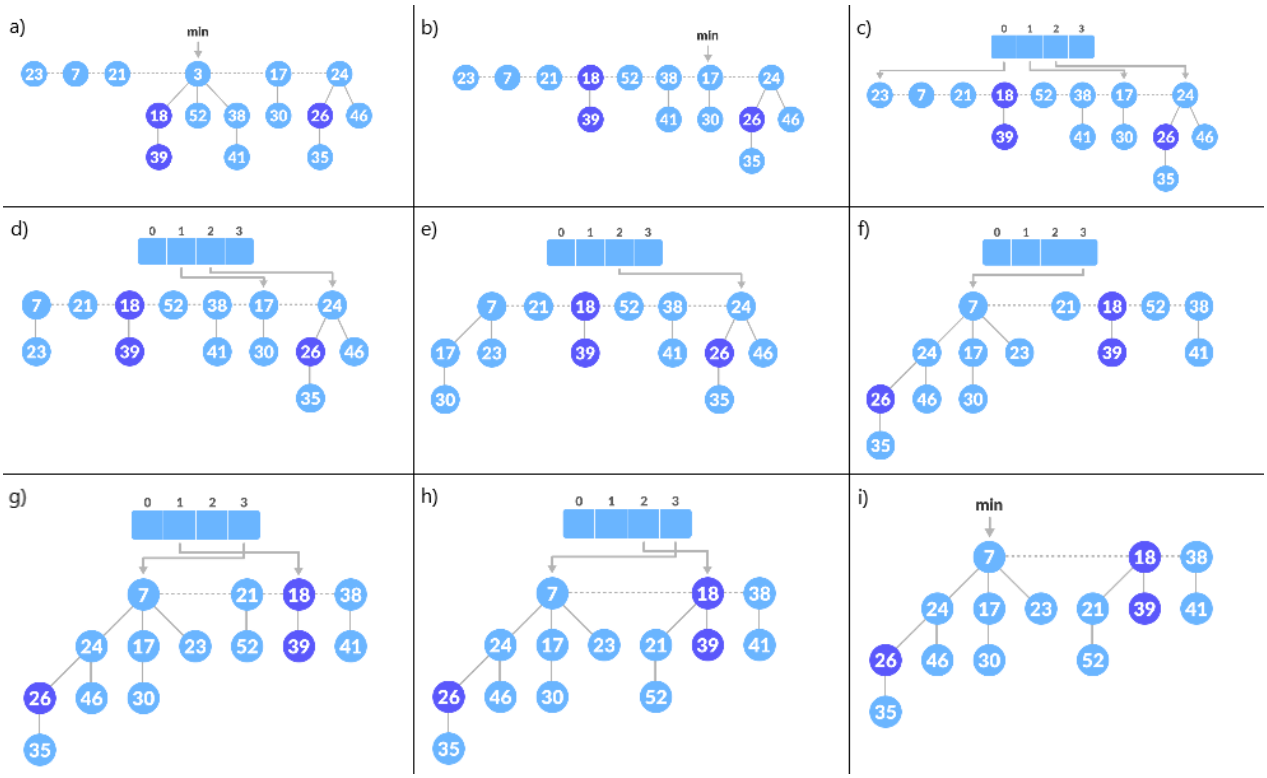
f) Korijeni 7 i 24 spajaju se jer imaju isti stupanj (2). Čvor 24 postaje dijete jer ima veću vrijednost. Stupanj čvora 7 postaje 3.

g) Korijeni 21 i 52 spajaju se jer imaju isti stupanj (0). Čvor 52 postaje dijete jer ima veću vrijednost. Stupanj čvora 21 postaje 1.

h) Korijeni 18 i 21 spajaju se jer imaju isti stupanj (1). Čvor 21 postaje dijete jer ima veću vrijednost. Stupanj čvora 18 postaje 2.

i) Svi korijeni imaju jedinstveni stupanj, time završava spajanje. Pokazivač se ažurira i postavlja na korijen s najmanjom vrijednosti. Konačni izgleda hrpe je prikazan.

Operacija *extract min* izvodi se u $O(log\ n)$ vremenskoj složenosti, (31).




**Slika 26.** Koraci operacije extract min, *(31)*

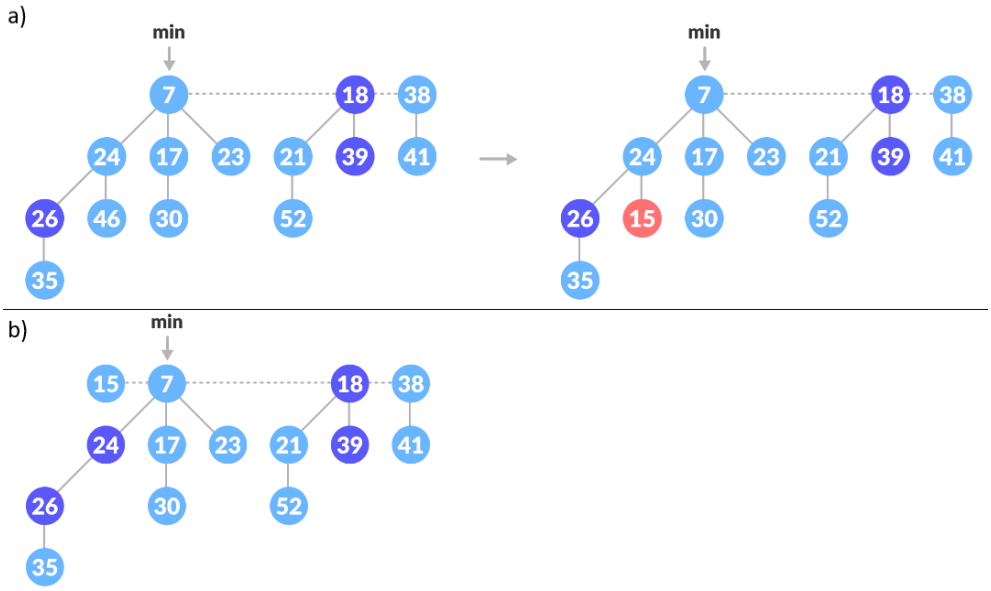
### 5.2.5 Operacija Decrease Key

Ažuriranje vrijednosti težine vrha je isto proces koji se izvršava kroz Dijkstrin algoritam. Taj proces će obavljati operacija *decrease key*.

Operacija *decrease key* smanjuje vrijednost određenog čvora i prilagođava strukturu kako bi se održala svojstva hrpe. Operacija je objašnjena kroz dva primjera kako bi se u potpunosti objasnio njen rad, (33).

Prvi primjer prikazan je na **Slika 27**.

a) Čvoru 46 se smanjuje vrijednost na 15. Važno je uočiti kako su neki čvorovi označeni tamno plavom bojom što znači da su izgubili već jedno dijete. Kada čvor izgubi dijete po drugi put, taj čvor se reže i dodaje u popis korijena kako bi se očuvala struktura hrpe i ispunila svojstva hrpe.
b) Čvor 15 je manji od svojeg roditelja 24, reže se i dodaje u popis korijena, a 24 se označava tamno plavom bojom jer je izgubio jedno dijete.




**Slika 27.** Operacija decrease key - prvi primjer, *(33)*

Kako bi se shvatilo označavanje čvorova tamno plavom bojom slijedi drugi primjer prikazan na **Slika 28**.

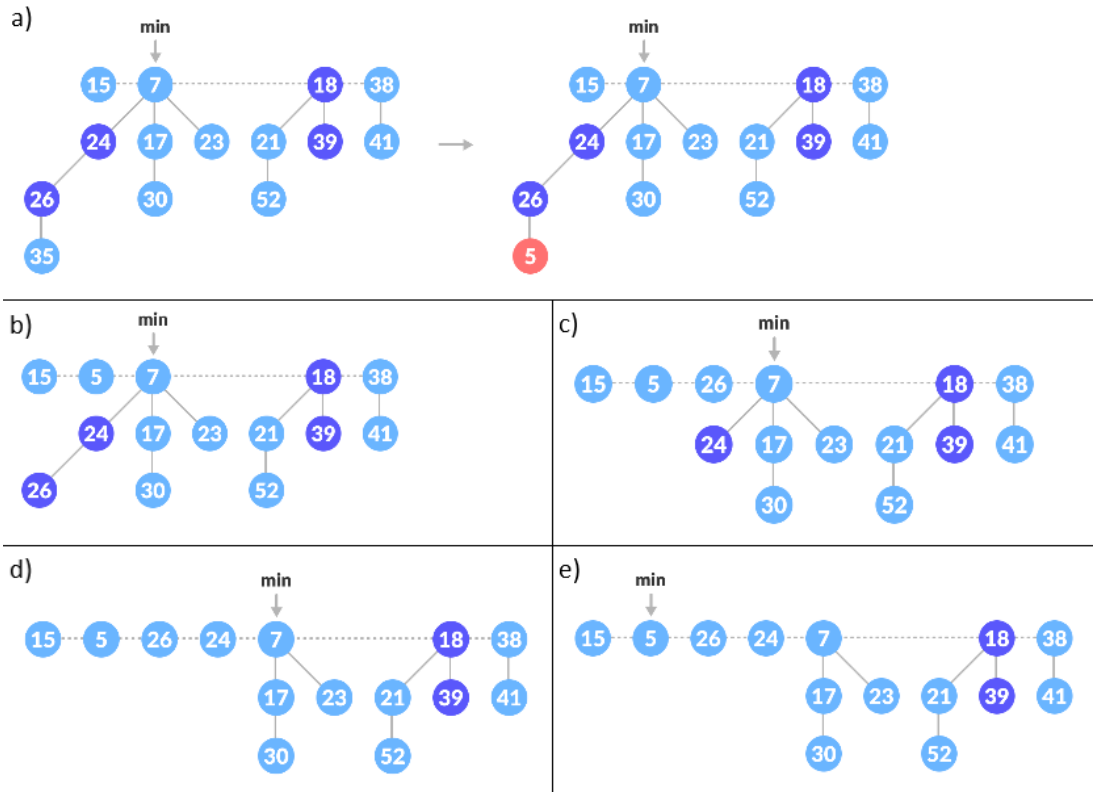
a) Čvoru 35 se smanjuje vrijednost na 5.
b) Čvor 5 je manji od svojeg roditelja 26, reže se i dodaje u popis korijena, a čvor 26 se označava da je izgubio drugo dijete (u prošlom primjeru je bio označen, izgubio je jedno dijete).

c) Čvor 26 se reže i dodaje u popis korijena jer je izgubio dva djeteta, a čvor 24 se označava da je izgubio drugo dijete (u prošlom primjeru je izgubio jedno dijete).

d) Čvor 24 se reže i dodaje u popis korijena jer je izgubio dva djeteta.

e) Pokazivač se ažurira i postavlja na korijen s najmanjom vrijednosti. Konačni izgleda hrpe je prikazan. Svi čvorovi koji su izgubili dva djeteta (izrezani su i dodani u popis korijena) označavaju se svijetlo plavom bojom tj. da nemaju izgubljene djece.

Operacija *decrease key* izvodi se u $O(1)$ vremenskoj složenosti, (33).



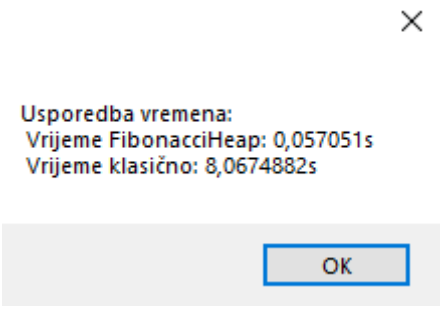

**Slika 28.** Operacija decrease key - drugi primjer, *(33)*

## 5.3 Rezultati implementacije Fibonaccijeve hrpe

U ovom radu su napravljene dvije verzije Dijkstrinog algoritma kako bi se mogli usporediti rezultati brzine izvršavanja algoritma. Prva verzija Dijkstrinog algoritma koristi nesortirano polje za popis neobrađenih vrhova gdje pronalaženje i izbacivanje vrha s najmanjom težinom polja stvara usko grlo u performansama algoritma. U ovom slučaju,

Dijkstrin algoritam ima vremenskoj složenosti u $O(V^2)$. U drugoj verziji Dijkstrinog algoritma implementirana je Fibonaccijeva hrpa koja koristi operacije poput extract min i decrease key koje znatno ubrzavaju izvršavanje programa. U ovom slučaju, Dijkstrin algoritam ima vremensku složenost $O(V\ log\ V)$ zbog grafa male gustoće objašnjeno u prethodnom potpoglavlju, (17).

Za usporedbu izvršavanja navedenih implementacija Dijkstrinog algoritma, u programu su dodana dva tajmera koji mjere vrijeme izvršavanja algoritma. Za razumijevanje rezultata bitno je napomenuti veličinu razmatranog grafa, koji ima 43992 vrhova i 111350 bridova. Također, algoritmi su izvršeni na računalu s AMD Ryzen 5 5600X (4.2 GHz) procesorom i 16 GB (3600 MHz) radne memorije. Korištenjem nesortiranog polja za popis neobrađenih vrhova, vrijeme izvršavanja Dijkstrinog algoritma iznosi oko 8 sekundi. Korištenjem Fibonaccijeve hrpe u drugoj verziji algoritma, vrijeme izvršavanja se drastično smanjuje na oko 0.05 sekundi, što je više od 140 puta brže vrijeme izvršavanja. Stoga je upotreba Fibonaccijeve hrpe izuzetno korisna za ubrzavanje Dijkstrinog algoritma. Na **Slika 29** su prikazani rezultati vremena izvršavanja algoritma unutar programa.




**Slika 29.** Statistički prikaz rezultata vremena izvršavanja algoritma

# 6 GRAFIČKO SUČELJE

Za vizualizaciju rezultata i ruta dobivenih Dijkstrinim algoritmom potrebna je izrada interaktivnog grafičkog korisničkog sučelja. U ovom radu, za izradu sučelja korišten je Windows Forms, dio .NET Frameworka koji je namijenjen razvoju desktop aplikacija u programskom jeziku C#. Windows Forms je zapravo skup biblioteka koje omogućuju jednostavnu izradu interaktivnog grafičkog korisničkog sučelja za Windows operacijski sustav, (34). Za izradu sučelja korišteno je razvojno sučelje Visual Studio 2022.

## 6.1 NuGet paketi

NuGet paketi predstavljaju kolekcije softverskih biblioteka, aplikacija ili dijelova koda koji se mogu lako preuzeti i koristiti u programskom jeziku C#. Ovi paketi su stvoreni od strane programera koji žele podijeliti svoj kod s drugim korisnicima i učiniti ga dostupnim za korištenje u njihovim projektima, (35).

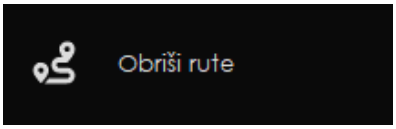
Korišteni dodatni paketi su:

- GMap.NET
- FibonnaciHeap
- FontAwesome.Sharp

Za prikaz rute na grafičkom sučelju potrebna je digitalna karta. Windows Forms sam po sebi ne podržava prikazivanje digitalnih karata, stoga je za prikazivanje ruta korišten programski paket GMap.NET, koji je instaliran putem NuGet Package Manager-a unutar Visual Studio razvojnog okruženja. GMap.NET je besplatan softverski paket koji omogućuje integraciju digitalne karte u Windows Forms aplikaciju. On pruža širok raspon platformi za prikazivanje digitalne karte, uključujući popularne opcije poput Google Maps, OpenStreetMap, Yahoo! Maps i Bing Maps. Osim osnovnih funkcionalnosti prikazivanja karte, GMap.NET paket pruža mogućnosti dodavanja oznaka (markera) na kartu, prikazivanja ruta, stvaranja poligona, izračunavanja udaljenosti između markera itd, (36).

Zbog kompleksnosti implementacije Fibonaccijeve hrpe, korišten je FibonnaciHeap besplatni softverski paket. Ovaj paket omogućuje vrlo jednostavnu implementaciju Fibonaccijeve hrpe bez potrebe za dodatnim pisanjem koda. U paketu su već dostupne metode za sve operacije koje se mogu izvršiti nad Fibonaccijevom hrpom.

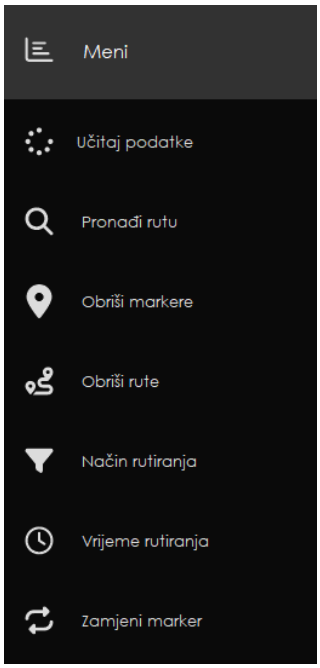
Za dodavanje modernog izgleda sučelju Windows Forms-a, korišten je besplatni softverski paket FontAwesome.Sharp. Ovaj paket pruža kolekciju od čak 2020 ikona koje se mogu lako odabrati i dodati unutar gumba. Na **Slika 30** prikazan je primjer izgleda gumba korištenjem FontAwesome.Sharp paketa. Kroz jednostavno korištenje ovih ikona, sučelje dobiva atraktivan vizualni element koji olakšava prepoznavanje funkcionalnosti gumba i poboljšava korisničko iskustvo.




**Slika 30.** Izgleda gumba korištenjem FontAwesome.Sharp paketa

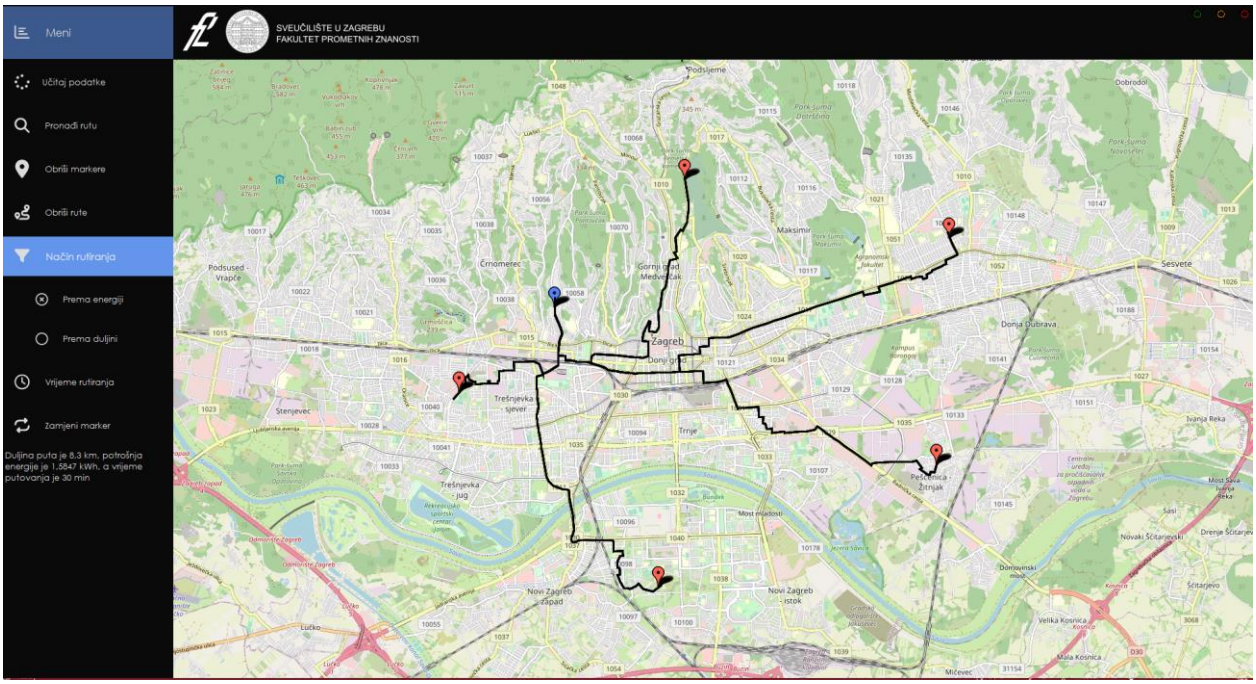
## 6.2 Opis izrađenog grafičkog sučelja

Grafičko korisničko sučelje koje je razvijeno omogućuje interaktivnu i intuitivnu upotrebu korisnicima. Kroz jednostavan i pregledan meni, prikazan na **Slika 31**, korisnik može odabrati razne funkcije koje su dostupne u aplikaciji. Uz meni prikazuje se digitalna karta koju korisnik putem kursora miša može pomicati te na nju može postaviti početni marker i više završnih markera.




**Slika 31.** Prikaz menija unutar grafičkog sučelja

Za početak korištenja aplikacije potrebno je učitati podatke klikom na gumb "Učitaj podatke" koji učitava podatke i tekstualne datoteke i izrađuje graf spreman za određivanje energetski optimalnog puta. Odabirom željenih točaka (markera) na digitalnoj karti i klikom na "Pronađi rutu", korisnik pokreće Dijkstrin algoritam i kao rezultat se iscrtavaju rute od početnog markera do svakog završnog. U slučaju da korisnik želi obrisati sve dodane markere i stvorene rute na karti, to može učiniti klikom na gumb "Obriši markere". Ako želi obrisati samo stvorene rute, korisnik može kliknuti na gumb "Obriši rute". Klikom na gumb "Način rutiranja" pojavljuju se dvije opcije iz padajućeg izbornika. Moguće je odabrati hoće li se rutiranje vršiti prema najkraćoj duljini ili najmanjoj potrošnji energije. Korisnik također može upisati početno vrijeme rutiranja klikom na gumb "Vrijeme rutiranja". Kako bi zamijenio odabrani završni marker s početnim markerom, korisnik može kliknuti na gumb "Zamijeni markere". Na karti je moguće kliknuti na rutu, nakon čega će se u meniju prikazati podaci o duljini puta, utrošenoj energiji i vremenu putovanja. Konačno, pojedine markere ili rute je moguće obrisati na karti duplim desnim klikom na njih. Izgleda interaktivnog grafičkog korisničkog sučelja vidljiv je na **Slika 32**.

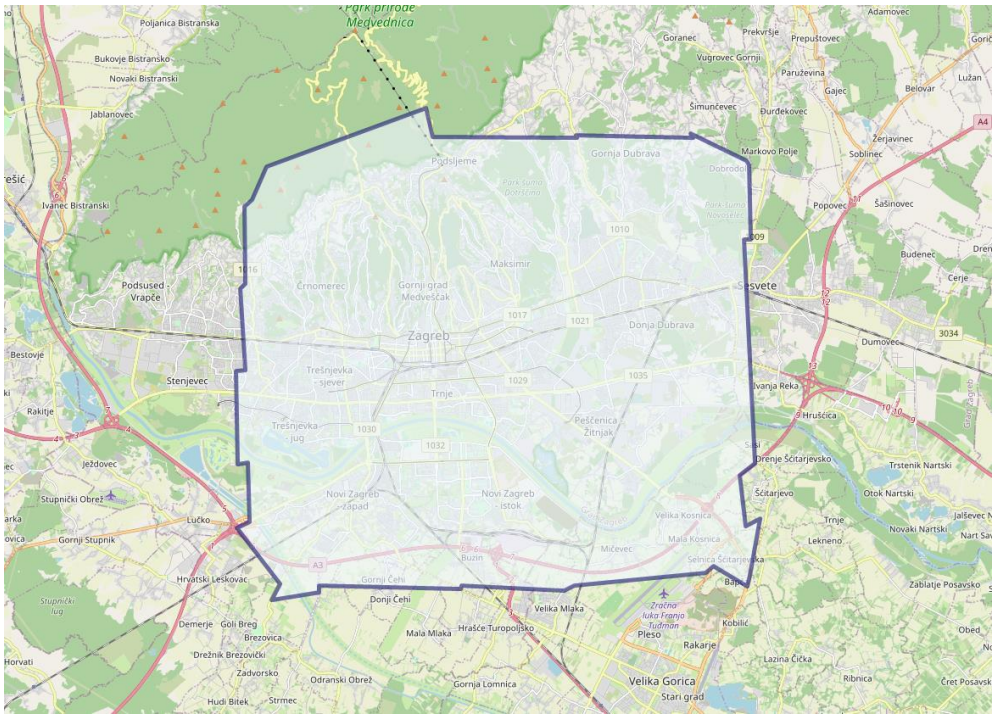



**Slika 32.** Vizualni prikaz grafičkog sučelja

### 6.2.1 Dodavanje markera

Dodavanje markera na kartu ne predstavlja problem jer GMap.NET ima već ugrađenu funkcionalnost za to. Međutim, izazov se javlja prilikom povezivanja dodanog markera na karti s odgovarajućim cestovnim segmentom. Kako bi se marker pridružio odgovarajućem cestovnom segmentu, koriste se geografske koordinate zemljopisne širine i dužine koje se dobiju prilikom postavljanja markera na kartu. Koristeći te podatke, izrađuje se vektorska projekcija markera na svaki cestovni segment. Vektorska projekcija određuje lokaciju na cestovnom segmentu gdje se nalazi projicirana točka markera. Nakon toga, računaju se udaljenosti između markera i lokacija vektorskih projekcija. Cestovni segment s najmanjom udaljenošću između markera i projicirane točke na njemu označava da pripada odabranom markeru.

Kako je rad ograničen podacima cestovne mreže koja obuhvaća širu okolicu grada Zagreba, korištena je funkcionalnost GMap.NET-a za izradu poligona, koji ograničava dodavanje markera izvan tog područja. Poligon je konstruiran pomoću vršnih točaka koje odgovaraju lokacijama dane cestovne mreže, kako je prikazano na **Slika 33**.




**Slika 33.** Prikaz izrađenog poligona

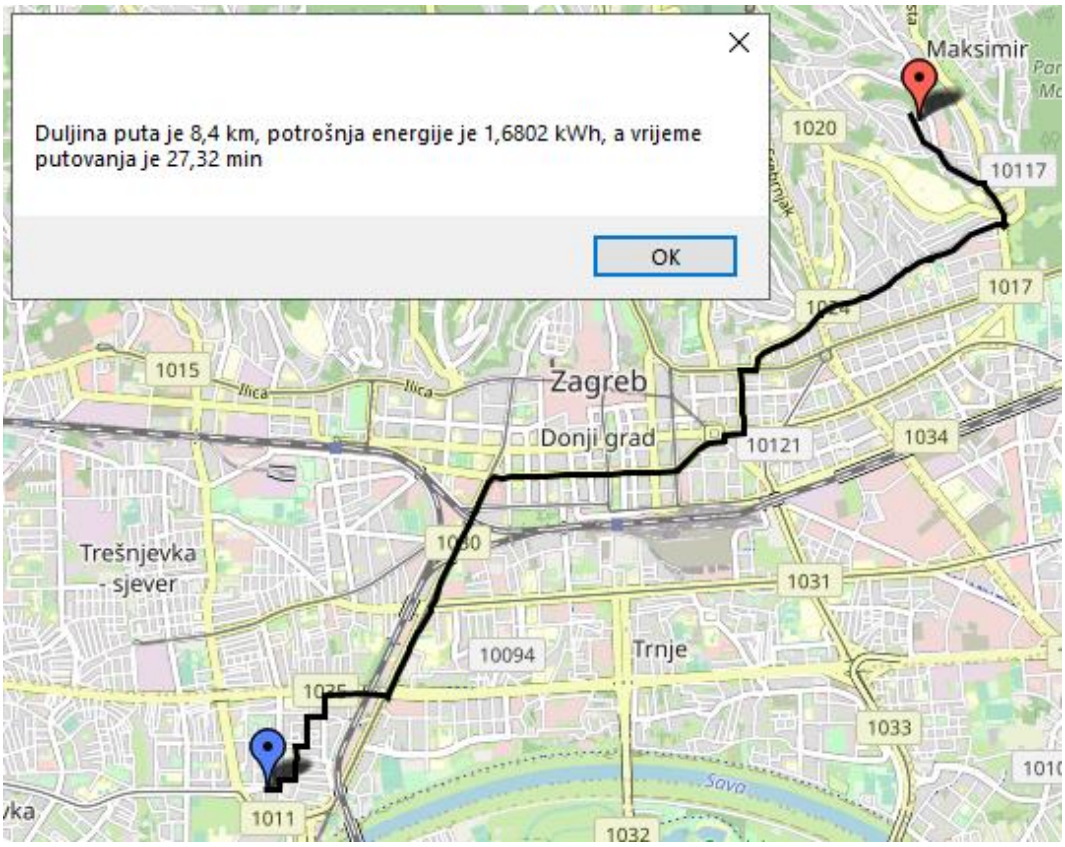
U grafičkom korisničkom sučelju je omogućeno dodavanje više markera i iscrtavanje više ruta. Kada korisnik odabere početni marker, koji predstavlja početnu točku za izvođenje Dijkstrinog algoritam, i završne markere izvršava se Dijkstrin algoritam kako bi se pronašle optimalne rute, te se te rute iscrtavaju na karti. Jedna zanimljiva funkcionalnost koja je implementirana omogućuje korisniku da dodaje nove markere i iscrtava rute do njih, bez potrebe ponovnog pokretanja Dijkstrinog algoritma, ako se početni marker nije promijenio. To je moguće postići iz razloga što se prilikom pokretanja Dijkstrinog algoritma s određenom početnom lokacijom izračunavaju najbolje rute do svake postojeće lokacije. Ova optimizacija omogućuje efikasnije iscrtavanje ruta, a značajna razlika u performansama je primjetna posebno u slučajevima kada Dijkstrin algoritam nije optimiziran s Fibonaccijevom hrpom. U takvim slučajevima, izvršavanje Dijkstrinog algoritma može potrajati nekoliko sekundi, što znači da bi iscrtavanje ruta također bilo značajno usporeno. Stoga, ova funkcionalnost pruža poboljšan korisnički doživljaj i brži prikaz ruta.

### 6.2.2 Način rutiranja

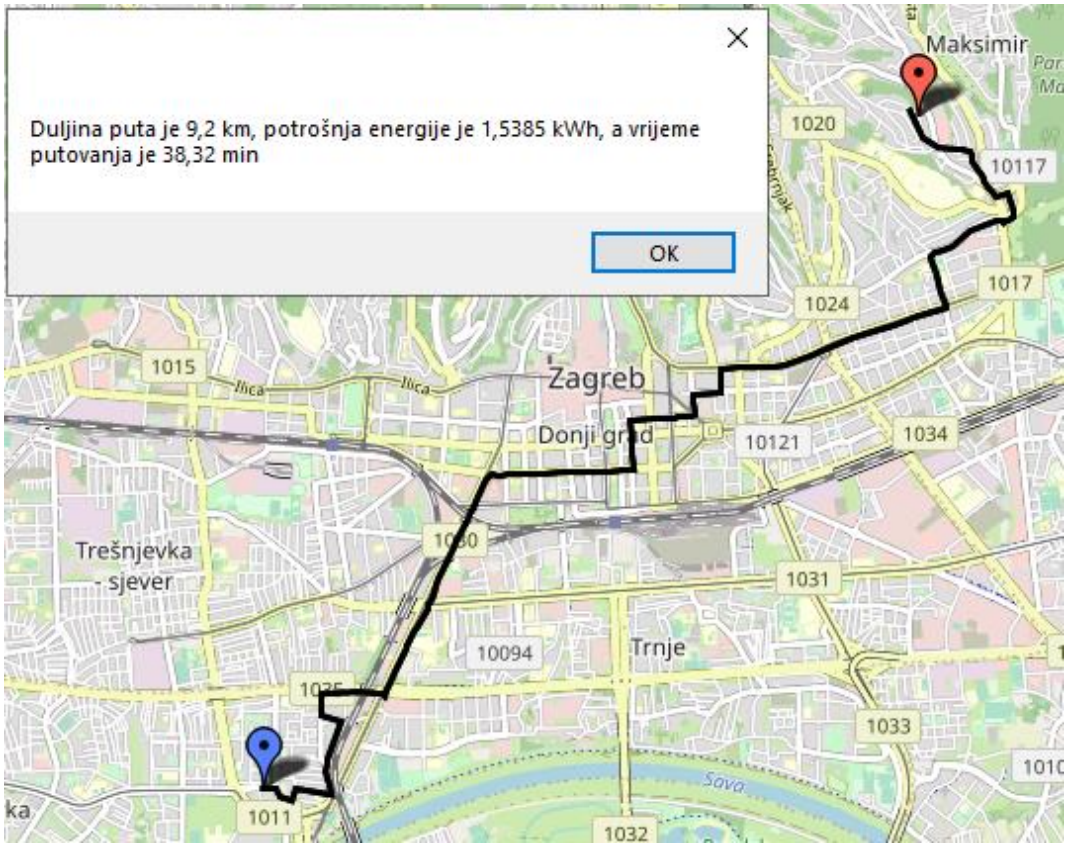
U procesu izrade rješenja za pronalaženje optimalnog energetskog puta, inicijalni fokus bio je na izradi algoritma za pronalaženje najkraćeg puta prema duljini zbog svoje jednostavnije izvedbe. Nakon uspješne implementacije algoritma za najkraći put prema duljini, izrađen je algoritam za pronalaženje optimalnog energetskog puta. Kao rezultat, u aplikaciji je moguće odabrati dva načina rutiranja: najkraća udaljenost i najmanja potrošnja energije. Uz to dodana je opcija promjene vremena polaska. Ako u vremenu polaska ništa nije upisano, rutiranje prema najmanjoj potrošnji energije se izračunava s vrijednostima prosječne potrošnje energije. Međutim, ako je upisano vrijeme polaska tada se rutiranje prema najmanjoj potrošnji energije izračunava s vrijednostima profila potrošnje energije.

Na **Slika 34**, **Slika 35** i **Slika 36** prikazane su različite dobivene rute i vrijednosti ukupne duljine puta, ukupne potrošnje energije i vremena putovanja pri različitim načinima rutiranja, uz zadržavanje istih početnih i završnih točaka. **Slika 34** prikazuje rutiranje prema najkraćoj udaljenosti gdje je duljina puta 8.4 km što je kraće u usporedbi s druga dva primjera. Na **Slika 35** prikazano je rutiranja prema najmanjoj potrošnji energije gdje je potrošnja energije 1,5385 kWh, dok je u prvom primjeru potrošnja energije 1,6802 kWh. U trećem primjeru prikazanom na **Slika 36** vrši se rutiranje prema najmanjoj energetskoj potrošnji s vremenom polaska u 16:00 sati. Vidljivo je da je vrijeme putovanja od 49,17 min duže u odnosu na prva dva primjera jer se rutiranje izvodi za vrijeme vršnog sata kada se stvara prometno zagušenje. Međutim, potrošnja energije je niža u usporedbi s prva dva primjera jer se vozilo kreće manjim brzinama što rezultira manjom potrošnjom energije.

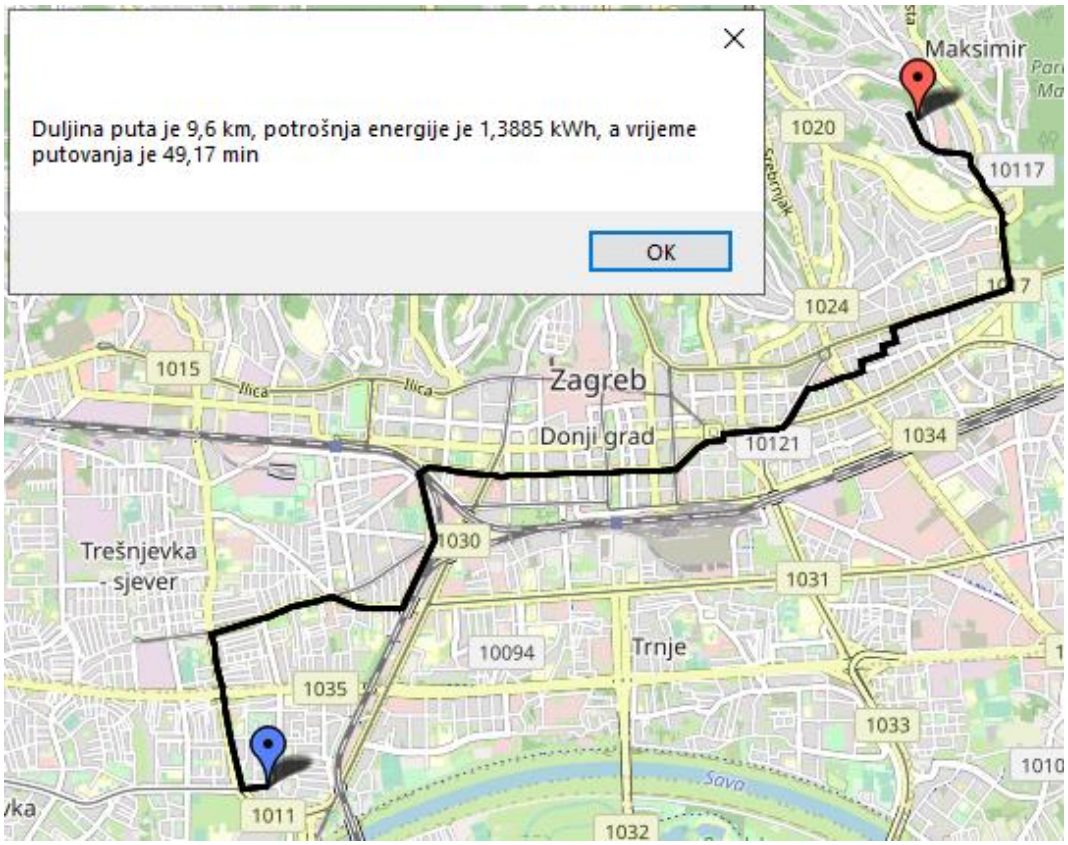
Za izračun ukupnog vremena putovanja koriste se podatci o duljini i prosječnoj brzini koji svaki cestovni segment sadrži. Na temelju tih podataka, vrlo jednostavno se može izračunati vrijeme putovanja na pojedinom cestovnom segmentu koristeći formulu $t = s/v$, gdje je $t$ vrijeme, $s$ duljina i $v$ prosječna brzina. Ukupno vrijeme putovanja za cijelu rutu dobiva se zbrajanjem vremena putovanja svih cestovnih segmenata koji se nalaze na toj ruti.




**Slika 34.** Primjer rutiranja prema najkraćoj udaljenosti




**Slika 35.** Primjer rutiranja prema najmanjoj potrošenoj energiji




**Slika 36.** Primjer rutiranja prema najmanjoj potrošenoj energiji s početkom rutiranja u 16:00 h

# 7 ZAKLJUČAK

Popularnost električnih vozila je u velikom porastu zahvaljujući njihovoj isplativosti i pozitivnom utjecaju na okoliš. Međutim, jedan od izazova s kojim se električna vozila suočavaju je ograničen doseg vožnje. Kroz ovaj rad se rješava problem energetski optimalnog puta kao jedno od rješenja problemu ograničenog dosega vožnje električnih vozila.

Rješavanje problema energetski optimalnog puta počinje analizom i opisom dobivenih podataka, koji se potom učitavaju u program. Nadalje, proučena je teorija grafova, počevši od njenog povijesnog razvoja pa sve do osnovnih koncepta koji su ključni za razumijevanje algoritama za pronalazak optimalnog puta. Jedan od takvih algoritama je Bellman-Ford algoritam koji, uz pomoć Johnsonove tehnike, priprema graf za primjenu Dijkstrinog algoritma. Kroz rad se Dijkstrin algoritam koristi kao glavni algoritam za pronalazak optimalnog energetskog puta.

Budući da Dijkstrin algoritam može biti vremenski zahtjevan, u rad se uvodi Fibonaccijeva hrpa kao optimalna struktura podataka za ubrzanje izvođenje algoritma. Implementacija Fibonaccijeve hrpe rezultira brzim algoritmom za pronalazak optimalnog energetskog puta za električna vozila.

Kao završni korak, razvija se interaktivno korisničko sučelje koje omogućuje praktičnu primjenu razvijenog algoritma. Korisničko sučelje pruža korisnicima mogućnost upravljanja algoritmom i korištenje njegovih rezultata u stvarnom vremenu, čime se postiže praktična primjena rješenja.

Kroz ovaj završni rad, stečeno je dublje razumijevanje problema dosega vožnje električnih vozila te je razvijen algoritam temeljen na teoriji grafova za pronalazak optimalnog energetskog puta. Implementacijom Fibonaccijeve hrpe kao optimalne strukture podataka omogućeno je brže izvršavanje algoritma, dok interaktivno korisničko sučelje pruža praktičnu primjenu algoritma u stvarnim situacijama.

# LITERATURA


1. Energy institute. Potrošnja energije u prometnom sektoru. [Online]. [Pristupljeno: 2023 Svibanj. Preuzeto s: https://www.energyinst.org/exploring-energy/topic/transport.
2. European Environment Agency. Potrošnja energije. [Online].; 2012 [Pristupljeno: 2023 Svibanj. Preuzeto s: https://www.eea.europa.eu/data-and-maps/figures/consumption-by-mode-eu-1.
3. United Nations Development Programme. Emisija stakleničkih plinova. [Online]. [Pristupljeno: 2023 Svibanj. Preuzeto s: https://www.undp.org/sustainable-development-goals#climate-action.
4. Hrvatska tehnička enciklopedija. Inteligentni transportni sustavi. [Online].; 2017 [Pristupljeno: 2023 Svibanj. Preuzeto s: https://tehnika.lzmk.hr/inteligentni-transportni-sustavi/.
5. M. J. Promet i zagađenje zraka. [Online].; 2022 [Pristupljeno: 2023 Svibanj 19. Preuzeto s: https://moodle.srce.hr/2021-2022/pluginfile.php/5798773/mod_resource/content/1/EuP_01_Promet_i_zagadenje_zraka.pdf.
6. European Environment Agency. Onečišćivači zraka. [Online].; 2019 [Pristupljeno: 2023 svibanj. Preuzeto s: https://www.eea.europa.eu/data-and-maps/daviz/contribution-of-the-transport-sector-6#tab-chart_4.
7. U.S. Department of Energy. Alternativna goriva. [Online]. [Pristupljeno: 2023 Svibanj. Preuzeto s: https://afdc.energy.gov/vehicles/electric_basics_ev.html.
8. D. K. Alternativni oblici energije u cestovnom prometu. 2021..
9. Adams-Columbia Electric Cooperative. Vrste električnih vozila. [Online]. [Pristupljeno: 2023 Svibanj. Preuzeto s: https://www.acecwi.com/types-of-electric-vehicles/.
10. IntechOpen. Učinkovitost motora na unutarnje izgaranje i elektromotora. [Online].; 2011 . [Pristupljeno: 2023 Svibanj. Preuzeto s: https://www.intechopen.com/chapters/18661.
11. YouTube. Generatorsko kočenje. [Online].; 2020 [Pristupljeno: 2023 Svibanj. Preuzeto s: . https://www.youtube.com/watch?v=WA32wMdd28g.
12. ZeCar. Doseg električnih vozila. [Online].; 2022 [Pristupljeno: 2023 Svibanj. Preuzeto s: . https://zecar.com/resources/electric-cars-vs-petrol-cars.
13. LucidMotors. Lucid Air Grand Touring. [Online].; 2023 [Pristupljeno: 2023 Svibanj. Preuzeto s: . https://lucidmotors.com/air.
14. WhichCar. Kapacitet baterije (kWh). [Online].; 2022 [Pristupljeno: 2023 Svibanj. Preuzeto s: . https://www.whichcar.com.au/car-advice/what-does-ev-kw-mean.
15. EVBox. Baterije električnih vozila. [Online].; 2022 [Pristupljeno: 2023 Svibanj. Preuzeto s: . https://blog.evbox.com/uk-en/ev-battery-longevity.
16. Wikipedia. Inteligentni transportni sustavi. [Online].; 2023 [Pristupljeno: 2023 Svibanj. Preuzeto s:

. https://hr.wikipedia.org/wiki/Inteligentni_transportni_sustavi.

17 T. E. Solving the Electric Vehicle Routing Problem Using a Hybrid Adaptive Large Neighborhood
. Search Method. [Online].; 2021 [Pristupljeno: 2023 Svibanj. Preuzeto s:
https://dr.nsk.hr/islandora/object/fpz:2603.

18 T. E. Problem usmjeravanja električnih vozila. [Online].; 2019 [Pristupljeno: 2023 Svibanj. Preuzeto
. s: https://www.fpz.unizg.hr/zits/wp-
content/uploads/2019/07/KDI_pregledan_rad_Tomislav_Erdelic.pdf.

19 N. M. Određivanje energetskog profila prometnice na temelju povijesnih zapisa o kretanju vozila.
. [Online].; 2019 [Pristupljeno: 2023 Svibanj. Preuzeto s:
https://dabar.srce.hr/islandora/object/fpz%3A1826.

20 Scaler. Razlika između liste i mape. [Online].; 2022 [Pristupljeno: 2023 Svibanj. Preuzeto s:
. https://www.scaler.com/topics/difference-between-list-and-dictionary-in-python/.

21 StackOverflow. Razlika u vremenu pretraživanja liste i mape. [Online].; 2017 [Pristupljeno: 2023
. Svibanj. Preuzeto s: https://stackoverflow.com/questions/43690191/why-are-dict-lookups-
always-better-than-list-lookups.

22 Bitnine. Problem sedam mostova Königsberga. [Online].; 2016 [Pristupljeno: 2023 Svibanj.
. Preuzeto s: https://bitnine.net/blog-graph-database/graph-database-and-seven-bridges-of-
konigsberg/?ckattempt=2.

23 Wikipedia. Problem sedam mostova Königsberga. [Online].; 2023 [Pristupljeno: 2023 Svibanj.
. Preuzeto s: https://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg.

24 Hrvatska enciklopedija. Teorija grafova. [Online].; 2021 [Pristupljeno: 2023 Svibanj. Preuzeto s:
. https://enciklopedija.hr/natuknica.aspx?ID=70127.

25 T. C. Osnovni pojmovi teorije grafova. [Online].; 2022 [Pristupljeno: 2023 Svibanj. Preuzeto s:
. https://moodle.srce.hr/2022-
2023/pluginfile.php/6854518/mod_resource/content/1/01_2012_10_16%20kineski_postar_ispravak_2016.pdf.

26 ResearchGate.. Potpun i nepotpun graf. [Online].; 2019 [Pristupljeno: 2023 Svibanj. Preuzeto s:
. https://www.researchgate.net/figure/A-Incomplete-network-graph-representation-of-observed-
breeding-crosses-B-The-complete_fig3_331871062.

27 Saint Louis University. Usmjereni i neusmjereni graf. [Online].; 2020 [Pristupljeno: 2023 Svibanj.
. Preuzeto s: https://cs.slu.edu/~esposito/teaching/1080/webscience/graphs.html.

28 Programiz. Bellman-Ford algoritam. [Online]. [Pristupljeno: 2023 Svibanj. Preuzeto s:
. https://www.programiz.com/dsa/bellman-ford-algorithm.

29 GeeksForGeeks.. Johnsonova tehnika. [Online].; 2023 [Pristupljeno: 2023 Svibanj. Preuzeto s:
. https://www.geeksforgeeks.org/johnsons-algorithm/.

30 Wikipedia. Dijkstra algoritam. [Online].; 2023 [Pristupljeno: 2023 Svibanj. Preuzeto s:

. https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm.

31 Programiz. FibonacciHeap. [Online]. [Pristupljeno: 2023 Svibanj. Preuzeto s:
. https://www.programiz.com/dsa/fibonacci-heap.

32 YouTube. FibonacciHeap. [Online].; 2022 [Pristupljeno: 2023 Svibanj. Preuzeto s:
. https://www.youtube.com/watch?v=6JxvKfSV9Ns&list=WL&index=11&.

33 Programiz. FibonacciHeap - Decrease Key. [Online]. [Pristupljeno: 2023 Svibanj. Preuzeto s:
. https://www.programiz.com/dsa/decrease-key-and-delete-node-from-a-fibonacci-heap.

34 Microsoft. Windows Forms. [Online].; 2023 [Pristupljeno: 2023 Svibanj. Preuzeto s:
. https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-7.0.

35 Microsoft.. NuGet paketi. [Online].; 2023 [Pristupljeno: 2023 Svibanj. Preuzeto s:
. https://learn.microsoft.com/en-us/nuget/quickstart/install-and-use-a-package-in-visual-studio.

36 Independent softwear. GMap.NET. [Online].; 2016 [Pristupljeno: ci2023 Svibanj. Preuzeto s:
. http://www.independent-software.com/gmap-net-beginners-tutorial-maps-markers-polygons-routes-updated-for-vs2015-and-gmap1-7.html.

# POPIS SLIKA

# POPIS TABLICA

Sveučilište u Zagrebu
Fakultet prometnih znanosti
Vukelićeva 4, 10000 Zagreb

# IZJAVA O AKADEMSKOJ ČESTITOSTI I SUGLASNOSTI

Izjavljujem i svojim potpisom potvrđujem da je ___završni rad___
(*vrsta rada*)
isključivo rezultat mojega vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu, a što pokazuju upotrijebljene bilješke i bibliografija. Izjavljujem da nijedan dio rada nije napisan na nedopušten način, odnosno da je prepisan iz necitiranog rada te da nijedan dio rada ne krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

Svojim potpisom potvrđujem i dajem suglasnost za javnu objavu završnog/diplomskog rada pod naslovom ___Rješavanje problema energetski optimalnog puta___ , u Nacionalni repozitorij završnih i diplomskih radova ZIR.

Student/ica: Dominik Knez

U Zagrebu, __13.06.2023.__

(ime i prezime, *potpis*)