SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

**Robert Manestar**

**Josip Petanjek**

**Sabina Pintar**

ANALIZA SLABOSTI GRADSKE PROMETNE MREŽE

PROJEKTNI ZADATAK

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Robert Manestar

Matični broj: 0016126820

Studij: *Informacijsko i programsko inženjerstvo*

Josip Petanjek

Matični broj: 0016124756

Studij: *Informacijsko i programsko inženjerstvo*

Sabina Pintar

Matični broj: 0177050711

Studij: *Informacijsko i programsko inženjerstvo*

ANALIZA SLABOSTI GRADSKE PROMETNE MREŽE

PROJEKTNI ZADATAK

Mentor:

Doc. dr. sc. Marcel Maretić

Varaždin, siječanj 2020.

Sadržaj

[Sadržaj iii](#_Toc30881412)

[1. Problem 1](#_Toc30881413)

[1.1. Povezanost s teorijom grafova 1](#_Toc30881414)

[1.2. Ranjivost ulica prometne mreže 1](#_Toc30881415)

[1.2.1. Računanje referentne vrijednosti 2](#_Toc30881416)

[1.2.2. Računanje kritičnosti pojedinih bridova 3](#_Toc30881417)

[1.3. Ranjivost raskrižja prometne mreže 3](#_Toc30881418)

[1.3.1. Računanje referentne vrijednosti 4](#_Toc30881419)

[1.3.2. Računanje kritičnosti pojedinih vrhova 5](#_Toc30881420)

[2. Zadaci 6](#_Toc30881421)

[2.1. Zadatak 1 6](#_Toc30881422)

[2.2. Rješenje zadatka 1 7](#_Toc30881423)

[2.3. Zadatak 2 14](#_Toc30881424)

[2.4. Rješenje zadatka 2 15](#_Toc30881425)

[2.5. Zadatak 3 25](#_Toc30881426)

[2.6. Rješenje zadatka 3 26](#_Toc30881427)

1. Problem

Uslijed naglog razvoja urbanih sredina pojedine gradske prometne mreže postale su izložene različitim problemima u vidu nedostupnosti pojedinih dijelova grada zbog prevelikih gužvi ili nemogućnosti korištenja pojedinih ulica. Slabosti se očituju u nemogućnosti prometne mreže da se učinkovito nosi s posljedicama određenih događaja kao što su blagdani, kvar ceste, prometna nesreća i sl. Zbog ovakvih događaja pojedine ulice postaju nedostupne na neodređeno vrijeme i to može prouzrokovati brojne probleme u prometu. Cilj je otkriti ulice koje u slučaju nedostupnosti mogu potencijalno prouzrokovati najveće probleme u prometu te na osnovu dobivenih rezultata poboljšati gradsku prometnu mrežu.

* 1. Povezanost s teorijom grafova

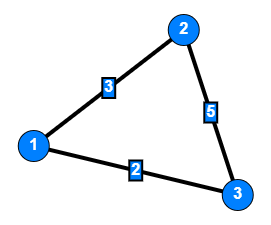
Opisani problem moguće je riješiti primjenom teorije grafova. Naime, gradska prometna mreža može se na intuitivan način može zamisliti kao graf, gdje raskrižja čine vrhove grafa, a ulice njegove bridove. Također, riječ je o težinskom grafu pri čemu težine predstavljaju broj minuta potreban da se određena ulica ili brid pređe normalnim hodom. U pojedinim grafovima koriste se i usmjereni bridovi (lukovi), koji predstavljaju jednosmjerne ulice. Nadalje, analiza slabosti prometne mreže vršit će se korištenjem dva jednostavna osmišljena algoritma koji se zasnivaju na teoriji grafova (Appert i Laurent, 2007).

* 1. Ranjivost ulica prometne mreže

Često zbog različitih razloga pojedine ulice mogu postati nedostupne. Ova analiza koristit će se za otkrivanje ulica čija nedostupnost može prouzrokovati najveće probleme za određenu prometnu mrežu. Analiza se provodi tako što se prvo izračuna zbroj svih najkraćih udaljenosti između svakog raskrižja, odnosno zbroj duljina najkraćih puteva između vrhova grafa. Dobivena vrijednost predstavlja referentnu vrijednost. Nakon toga pojedini brid se izbacuje iz grafa i ponavlja se računanje zbroja najkraćih puteva između vrhova, s tim da se izbačeni brid ne uzima u obzir. Dobivena vrijednost mora biti jednaka ili veća od referentne vrijednosti dobivene kod cijelog grafa. Razlika ove vrijednosti i referentne vrijednosti označava kritičnost brida. Nadalje, što je razlika veća, to je veći gubitak dostupnosti u prometnoj mreži.

Algoritam se ponavlja za sve ostale bridove grafa. Nakon toga utvrđuje se brid s najvećom vrijednosti, odnosno najkritičniji brid.

Provođenje algoritma prikazati ćemo na sljedećem jednostavnom grafu:



* + 1. Računanje referentne vrijednosti

Dakle, potrebno je izračunati i zbrojiti duljine najkraćih puteva između vrhova:

* i ostalih vrhova,
* i ostalih vrhova i
* i ostalih vrhova.

Podaci:

|  |  |
| --- | --- |
|  | broj vrhova grafa |
|  | broj bridova grafa |
|  | zbroj duljina najkraćih puteva između vrhova (referentna vrijednost) |
|  | duljina najkraćeg puta između vrha i |

* + 1. Računanje kritičnosti pojedinih bridova

Započet ćemo s bridom između vrhova i . Dakle, računamo zbroj udaljenosti najkraćih puteva između svakog vrha, ali bez navedenog brida.

A picture containing object, clock

Description automatically generated

Podaci:

|  |  |
| --- | --- |
|  | broj vrhova grafa |
|  | broj bridova grafa |
|  | brid između vrhova  i |
|  | zbroj duljina najkraćih puteva između  vrhova bez brida |
|  | kritičnost brida |
|  | duljina najkraćeg puta između vrha  i |

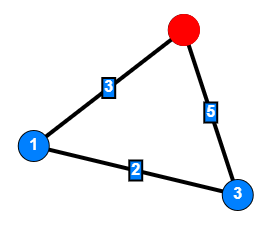
Postupak se ponavlja za ostale bridove grafa. Zatim se odredi najkritičniji brid, odnosno brid koji ima najveću kritičnu vrijednost.

* 1. Ranjivost raskrižja prometne mreže

Često zbog različitih razloga pojedina raskrižja mogu postati nedostupna. Ova analiza koristit će se za otkrivanje raskrižja čija nedostupnost može prouzrokovati najveće probleme za određenu prometnu mrežu. Analiza je djelomično slična analizi ranjivosti ulica prometne mreže. Jedna od razlika je u tome što se za svaki vrh posebno računa referenta vrijednost. Odabere se jedan od vrhova i izbaci se iz grafa. Bridovi izbačenog vrha ostaju i mogu se koristiti pri pronalasku najkraćeg puta. Nakon toga računa se zbroj svih najkraćih udaljenosti između svakog raskrižja. Dobivena vrijednost predstavlja referentnu vrijednost za izbačeni vrh. U nastavku algoritma izbacuju se bridovi izbačenog vrha i opet se računa zbroj svih najkraćih udaljenosti između svakog raskrižja, ali bez bridova izbačenog vrha. Dobivena vrijednost mora biti jednaka ili veća od referentne vrijednosti dobivene kod cijelog grafa. Razlika ove vrijednosti i referentne vrijednosti označava kritičnost vrha. Što je dobivena razlika veća, to je veći gubitak dostupnosti u prometnoj mreži. Algoritam se ponavlja za sve ostale vrhove grafa. Nakon toga utvrđuje se vrh s najvećom vrijednosti, odnosno najkritičniji vrh.

* + 1. Računanje referentne vrijednosti

Odaberemo jedan od vrhova, recimo da je to vrh . Nakon odabiranja potrebno je izračunati i zbrojiti duljine najkraćih puteva između vrhova, odnosno bez vrha . Nakon uklanjanja vrha njegovi bridovi ostaju i mogu se koristiti za pronalazak najkraćeg puta. U ovom slučaju oni se mogu zamisliti kao jedan brid duljine koji povezuje vrhove i .

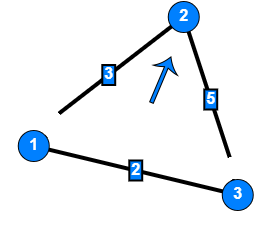


Podaci:

|  |  |
| --- | --- |
|  | broj vrhova grafa |
|  | broj bridova grafa |
|  | zbroj duljina najkraćih puteva između vrhova bez vrha 2 (referentna vrijednost) |
|  | duljina najkraćeg puta između vrha i |

* + 1. Računanje kritičnosti pojedinih vrhova

Započet ćemo s vrhom . Dakle, računamo zbroj udaljenosti najkraćih puteva između svakog vrha, ali bez navedenog vrha i njegovih bridova. Ostaju nam samo vrhovi i .



Podaci:

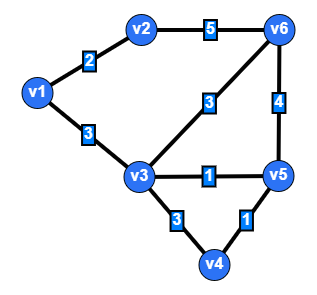
|  |  |
| --- | --- |
|  | broj vrhova grafa |
|  | broj bridova povezanih s vrhom |
|  | broj bridova grafa bez bridova |
|  | zbroj duljina najkraćih puteva između vrhova bez vrha i bridova |
|  | kritičnost vrha |
|  | duljina najkraćeg puta između vrha  i |

Sad se odabire sljedeći vrh i računa njegova referentna vrijednost i kritičnost. Ovo se ponavlja dok nisu obrađeni svi vrhovi, a nakon toga se odabire najkritičniji vrh, tj. vrh koji ima najkritičniju vrijednost.

1. Zadaci
   1. Zadatak 1

Na slici je prikazan manji dio prometne mreže. Sve ulice su dvosmjerne. Napravite analizu ranjivosti ulica i raskrižja prometne mreže i odgovorite na sljedeća pitanja:

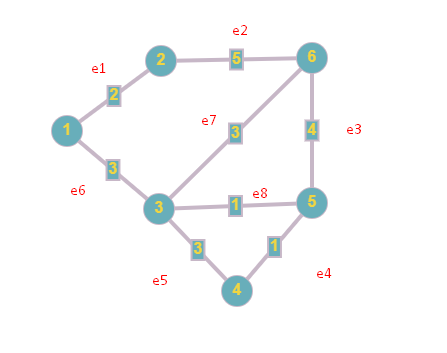
1. Koja ulica je najkritičnija?
2. Koji vrh je najkritičniji?
3. Koji algoritam ste koristili prilikom računanja najkraćih puteva između vrhova i zašto? Napomena: Ako ste koristili neki alat ili program, napišite koji algoritam on koristi)
4. Je li moguće obići sve ulice ovog dijela grada tako da se prođe svakom ulicom samo jednom i da se vratimo u početnu ulicu? Ako jest, navedite tu šetnju i što ona predstavlja?
5. Jeste li primijetili kakvih nedostataka ili redundantnosti algoritma u računanju najkraćih puteva između čvorova? Prodiskutirajte i predložite moguće poboljšanje.



* 1. Rješenje zadatka 1

1. Koja ulica je najkritičnija?

Ovaj problem se rješava kako je naznačeno u opisu problema, prvo se računaju duljine najkraćih puteva između vrhova, te se rezultati sumiraju. Za ovo ćemo koristiti alat *Graph Online*, dostupan na <https://graphonline.ru/en/> [1]. Sam izrađeni graf dostupan je na <http://graphonline.ru/en/?graph=dqRWThMqlPCdRCrw> .



Najkraće puteve nalazimo preko Dijkstrinog algoritma, te je rezultat sljedeća matrica:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | v1 | v2 | v3 | v4 | v5 | v6 |
| v1 | 0 | 2 | 3 | 5 | 4 | 6 |
| v2 | 2 | 0 | 5 | 7 | 6 | 5 |
| v3 | 3 | 5 | 0 | 2 | 1 | 3 |
| v4 | 5 | 7 | 2 | 0 | 1 | 5 |
| v5 | 4 | 6 | 1 | 1 | 0 | 4 |
| v6 | 6 | 5 | 3 | 5 | 4 | 0 |

Njezina suma je

Nadalje računamo kritičnost pojedinih bridova.

Slijede matrice s najkraćim udaljenostima pojedinih vrhova, bez naznačenog brida.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | v1 | v2 | v3 | v4 | v5 | v6 |
| v1 | 0 | 11 | 3 | 5 | 4 | 6 |
| v2 | 11 | 0 | 8 | 10 | 9 | 5 |
| v3 | 3 | 8 | 0 | 2 | 1 | 3 |
| v4 | 5 | 10 | 2 | 0 | 1 | 5 |
| v5 | 4 | 9 | 1 | 1 | 0 | 4 |
| v6 | 6 | 5 | 3 | 5 | 4 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | v1 | v2 | v3 | v4 | v5 | v6 |
| v1 | 0 | 2 | 3 | 5 | 4 | 6 |
| v2 | 2 | 0 | 5 | 7 | 6 | 8 |
| v3 | 3 | 5 | 0 | 2 | 1 | 3 |
| v4 | 5 | 7 | 2 | 0 | 1 | 5 |
| v5 | 4 | 6 | 1 | 1 | 0 | 4 |
| v6 | 6 | 8 | 3 | 5 | 4 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | v1 | v2 | v3 | v4 | v5 | v6 |
| v1 | 0 | 2 | 3 | 6 | 4 | 6 |
| v2 | 2 | 0 | 5 | 8 | 6 | 5 |
| v3 | 3 | 5 | 0 | 3 | 1 | 3 |
| v4 | 6 | 8 | 3 | 0 | 4 | 6 |
| v5 | 4 | 6 | 1 | 4 | 0 | 4 |
| v6 | 6 | 5 | 3 | 6 | 4 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | v1 | v2 | v3 | v4 | v5 | v6 |
| v1 | 0 | 2 | 3 | 5 | 4 | 6 |
| v2 | 2 | 0 | 5 | 7 | 6 | 5 |
| v3 | 3 | 5 | 0 | 2 | 1 | 3 |
| v4 | 5 | 7 | 2 | 0 | 1 | 5 |
| v5 | 4 | 6 | 1 | 1 | 0 | 4 |
| v6 | 6 | 5 | 3 | 5 | 4 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | v1 | v2 | v3 | v4 | v5 | v6 |
| v1 | 0 | 2 | 3 | 5 | 4 | 6 |
| v2 | 2 | 0 | 5 | 7 | 6 | 5 |
| v3 | 3 | 5 | 0 | 2 | 1 | 3 |
| v4 | 5 | 7 | 2 | 0 | 1 | 5 |
| v5 | 4 | 6 | 1 | 1 | 0 | 4 |
| v6 | 6 | 5 | 3 | 5 | 4 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | v1 | v2 | v3 | v4 | v5 | v6 |
| v1 | 0 | 2 | 10 | 12 | 11 | 7 |
| v2 | 2 | 0 | 8 | 10 | 9 | 5 |
| v3 | 10 | 8 | 0 | 2 | 1 | 3 |
| v4 | 12 | 10 | 2 | 0 | 1 | 5 |
| v5 | 11 | 9 | 1 | 1 | 0 | 4 |
| v6 | 7 | 5 | 3 | 5 | 4 | 0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | v1 | v2 | v3 | v4 | v5 | v6 |
| v1 | 0 | 2 | 3 | 5 | 4 | 7 |
| v2 | 2 | 0 | 5 | 7 | 6 | 5 |
| v3 | 3 | 5 | 0 | 2 | 1 | 5 |
| v4 | 5 | 7 | 2 | 0 | 1 | 5 |
| v5 | 4 | 6 | 1 | 1 | 0 | 4 |
| v6 | 7 | 5 | 5 | 5 | 4 | 0 |

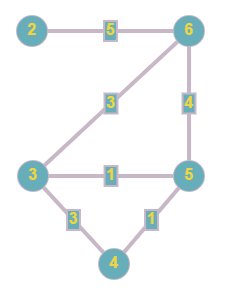
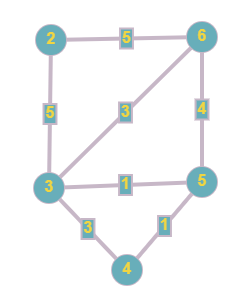
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | v1 | v2 | v3 | v4 | v5 | v6 |
| v1 | 0 | 2 | 3 | 6 | 7 | 6 |
| v2 | 2 | 0 | 5 | 8 | 9 | 5 |
| v3 | 3 | 5 | 0 | 3 | 4 | 3 |
| v4 | 6 | 8 | 3 | 0 | 1 | 5 |
| v5 | 7 | 9 | 4 | 1 | 0 | 4 |
| v6 | 6 | 5 | 3 | 5 | 4 | 0 |

Dakle, dobivamo da je **e4** najkritičnija ulica.

1. Koji vrh je najkritičniji?

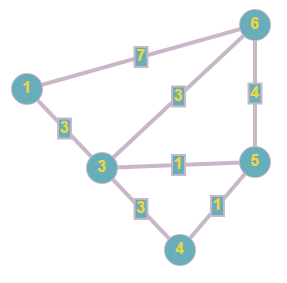
Ponovo pratimo upute problema, računamo zbroj najkraćih puteva između vrhova bez određenog vrha (referentnu vrijednost), te zbroj duljina najkraćih puteva između vrhova bez određenog vrha i njegovih bridova.

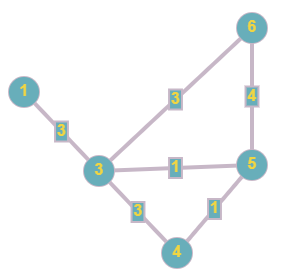
* Bez vrha 1



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | v2 | v3 | v4 | v5 | v6 |
| v2 | 0 | 8 | 10 | 9 | 5 |
| v3 | 8 | 0 | 2 | 1 | 3 |
| v4 | 10 | 2 | 0 | 1 | 5 |
| v5 | 9 | 1 | 1 | 0 | 4 |
| v6 | 5 | 3 | 5 | 4 | 0 |

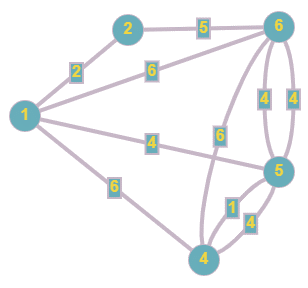
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | v2 | v3 | v4 | v5 | v6 |
| v2 | 0 | 5 | 7 | 6 | 5 |
| v3 | 5 | 0 | 2 | 1 | 3 |
| v4 | 7 | 2 | 0 | 1 | 5 |
| v5 | 6 | 1 | 1 | 0 | 4 |
| v6 | 8 | 3 | 5 | 4 | 0 |

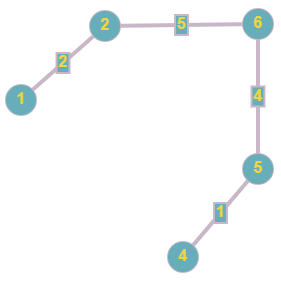
* Bez vrha 2



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | v1 | v3 | v4 | v5 | v6 |
| v1 | 0 | 3 | 5 | 4 | 6 |
| v3 | 3 | 0 | 2 | 1 | 3 |
| v4 | 5 | 2 | 0 | 1 | 5 |
| v5 | 4 | 1 | 1 | 0 | 4 |
| v6 | 6 | 3 | 5 | 4 | 0 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | v1 | v3 | v4 | v5 | v6 |
| v1 | 0 | 3 | 5 | 4 | 6 |
| v3 | 3 | 0 | 2 | 1 | 3 |
| v4 | 5 | 2 | 0 | 1 | 5 |
| v5 | 4 | 1 | 1 | 0 | 4 |
| v6 | 6 | 3 | 5 | 4 | 0 |

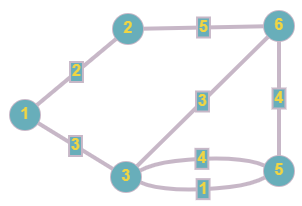
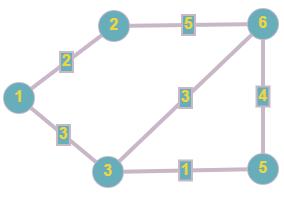
* Bez vrha 3



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | v1 | v2 | v4 | v5 | v6 |
| v1 | 0 | 2 | 5 | 4 | 6 |
| v2 | 2 | 0 | 7 | 6 | 5 |
| v4 | 5 | 7 | 0 | 1 | 5 |
| v5 | 4 | 6 | 1 | 0 | 4 |
| v6 | 6 | 5 | 5 | 4 | 0 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | v1 | v2 | v4 | v5 | v6 |
| v1 | 0 | 2 | 12 | 11 | 7 |
| v2 | 2 | 0 | 10 | 9 | 5 |
| v4 | 12 | 10 | 0 | 1 | 5 |
| v5 | 11 | 9 | 1 | 0 | 4 |
| v6 | 7 | 5 | 5 | 4 | 0 |

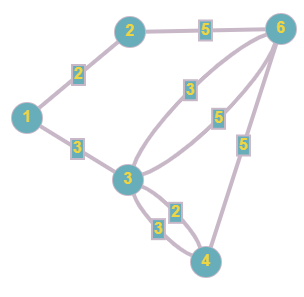
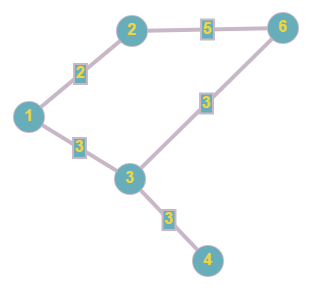
* Bez vrha 4



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | v1 | v2 | v3 | v5 | v6 |
| v1 | 0 | 2 | 3 | 4 | 6 |
| v2 | 2 | 0 | 5 | 6 | 5 |
| v3 | 3 | 5 | 0 | 1 | 3 |
| v5 | 4 | 6 | 1 | 0 | 4 |
| v6 | 6 | 5 | 3 | 4 | 0 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | v1 | v2 | v3 | v5 | v6 |
| v1 | 0 | 2 | 3 | 4 | 6 |
| v2 | 2 | 0 | 5 | 6 | 5 |
| v3 | 3 | 5 | 0 | 1 | 3 |
| v5 | 4 | 6 | 1 | 0 | 4 |
| v6 | 6 | 5 | 3 | 4 | 0 |

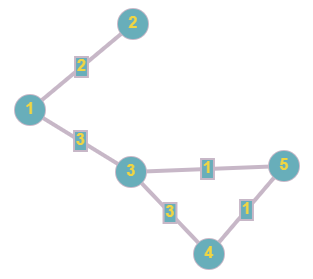
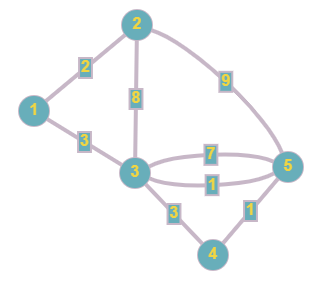
* Bez vrha 5



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | v1 | v2 | v3 | v4 | v6 |
| v1 | 0 | 2 | 3 | 5 | 6 |
| v2 | 2 | 0 | 5 | 7 | 5 |
| v3 | 3 | 5 | 0 | 2 | 3 |
| v4 | 5 | 7 | 2 | 0 | 5 |
| v6 | 6 | 5 | 3 | 5 | 0 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | v1 | v2 | v3 | v4 | v6 |
| v1 | 0 | 2 | 3 | 6 | 6 |
| v2 | 2 | 0 | 5 | 8 | 5 |
| v3 | 3 | 5 | 0 | 3 | 3 |
| v4 | 6 | 8 | 3 | 0 | 6 |
| v6 | 6 | 5 | 3 | 6 | 0 |

* Bez vrha 6



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | v1 | v2 | v3 | v4 | v5 |
| v1 | 0 | 2 | 3 | 5 | 4 |
| v2 | 2 | 0 | 5 | 7 | 6 |
| v3 | 3 | 5 | 0 | 2 | 1 |
| v4 | 5 | 7 | 2 | 0 | 1 |
| v5 | 4 | 6 | 1 | 1 | 0 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | v1 | v2 | v3 | v4 | v5 |
| v1 | 0 | 2 | 3 | 5 | 4 |
| v2 | 2 | 0 | 5 | 7 | 6 |
| v3 | 3 | 5 | 0 | 2 | 1 |
| v4 | 5 | 7 | 2 | 0 | 1 |
| v5 | 4 | 6 | 1 | 1 | 0 |

* Dakle, najkritičniji vrh je vrh 3.

c) Koji algoritam ste koristili prilikom računanja najkraćih puteva između vrhova i zašto? Napomena: Ako ste koristili neki alat ili program, napišite koji algoritam on koristi)

Korišten alat: <https://graphonline.ru/en/> [1]

Ovaj alat koristi Dijkstrin algoritam za pronalaženje najkraćih puteva između vrhova u težinskom grafu, te smo rezultate zapisali u matrice, izračunali njihove sume te ih primijenili pri rješavanju zadatka.

d) Je li moguće obići sve ulice ovog dijela grada tako da se prođe svakom ulicom samo jednom i da se vratimo u početnu ulicu? Ako jest, navedite tu šetnju i što ona predstavlja?,

Nije moguće obići sve ulice samo jednom i vratiti se u početnu ulicu. Ovo je problem Eulerove ture, „*Eulerova tura na G je tura koja prolazi svakim bridom točno jednom, tj. to je zatvorena Eulerova staza na G.*“. Nadalje „*Graf G je Eulerov graf ako dopušta Eulerovu turu.*“, dok po Eulerovom teoremu znamo da „*Neprazni povezani graf G je Eulerov graf akko su mu svi vrhovi parnog stupnja.*“. Dakle, budući da mu nisu svi vrhovi parnog stupnja, tada ne postoji Eulerova tura odnosno takva šetnja. [2]

e) Jeste li primijetili kakvih nedostataka ili redundantnosti algoritma u računanju najkraćih puteva između čvorova? Prodiskutirajte i predložite moguće poboljšanje.

Za ovaj problem mogli bi smo smanjiti količinu posla korištenjem Floyd-Warshallov algoritma umjesto Dijkstre, jer on nam već daje najkraće udaljenosti između svaka dva vrha u grafu. Nadalje primijetili smo da pri izračuni nekog vrha, dolazimo do istog rezultata kao i kod originalne matrice odnosno sume najkraćih puteva, samo što se mora maknuti odabrani vrh.

* 1. Zadatak 3

Na slici se nalazi pojednostavljeni prikaz središnje prometne mreže grada Ivanca. Također, prikazan je i pripadni graf prometne mreže. Potrebno je napraviti analizu ranjivosti ulica grada Ivanca.

1. Za rješavanje problema ove veličine potrebno je napraviti programsko rješenje koje će na osnovu unesenih podataka o prometnoj mreži ispisati najkritičniju ulicu. Programsko rješenje mora rješavati cijelu domenu problema, a ne samo ovaj graf!
2. Izračunajte vremensku složenost izrađenog algoritma (Pomoćna literatura: Lovrenčić, 2018)
3. Razmislite o mogućem poboljšanju prometne mreže grada na osnovu dobivenih rezultata. Navedite ulicu ili ulice koje bi bile dovoljne da se smanji kritičnost prometne mreže.

A picture containing sky, indoor, table

Description automatically generated

* 1. Rješenje zadatka 3

1. Za rješavanje problema ove veličine potrebno je napraviti programsko rješenje koje će na osnovu unesenih podataka o prometnoj mreži ispisati najkritičniju ulicu. Programsko rješenje mora rješavati cijelu domenu problema, a ne samo ovaj graf!

Programsko rješenje izrađeno je u Pythonu [3], s bibliotekama NetworkX [4] i MatPlotLib [5].

Funkcija sum\_of\_shortest\_paths() izračunava sumu najkraćih puteva u grafu (ili podgrafu) G. U funkciji se iterativno izračunava najkraći put između svih parova vrhova grafa G te se duljine puteva zbrajaju.

def sum\_of\_shortest\_paths(G):

    referent\_val = 0

    for s in G.nodes:

        for t in G.nodes:

                if(nx.has\_path(G,s,t)):

                    referent\_val = referent\_val + nx.shortest\_path\_length(G,

source=s, target=t, weight='weight')

    return referent\_val

Funkcija calculate\_criticallity\_for\_given\_edge() izračunava kritičnost danog brida ee u početnom grafu G. U funkciji se izračunava suma najkraćih puteva u podgrafu H koji ne sadrži brid (ee – edge element) koji je proslijeđen u parametru. Parametar m definira način izvršavanja. Način izvršavanja 0 - Grafovi u kojemu su lukovi između para vrhova (v1, v2) jednakih težina. Način izvršavanja 1 - Grafovi u kojemu lukovi između para vrhova (v1, v2) različitih težina.

def calculate\_criticallity\_for\_given\_edge(G, ee, m):

    try:

        H = G.copy()

        if m==0:

            H.remove\_edge(ee[0],ee[1])

            if(H.has\_edge(ee[1],ee[0])):

                H.remove\_edge(ee[1],ee[0])

        elif m==1:

            H.remove\_edge(ee[0],ee[1])

        edge\_val = sum\_of\_shortest\_paths(H)

        H.clear()

        return edge\_val

    except Exception as e:

        print('', end = '')

Funkcija calculate\_referent\_node\_value() izračunava referentnu vrijednost za dani vrh u početnom grafu G tako da stvori podgraf H u kojemu se uklanja vrh (ne – node element) proslijeđen u parametru i bridovi koji ulaze/izlaze iz njega, zatim se lukovi povezuju s obzirom na susjedne vrhove vrha ne i dodjeljuje im se odgovarajuća težina. Parametar m definira način izvršavanja. Način izvršavanja 0 - Grafovi u kojemu su lukovi između para vrhova (v1, v2) jednakih težina. Način izvršavanja 1 - Grafovi u kojemu lukovi između para vrhova (v1, v2) različitih težina.

def calculate\_referent\_node\_value(G, ne, m):

    H = G.copy()

    edge\_list\_for\_ne = []

    if m==0:

        for ee in G.edges:

            if ee[1]==ne:

                edge\_list\_for\_ne.append(ee)

    elif m==1:

        for ee in G.edges:

            if ee[1]==ne or ee[0]==ne:

                edge\_list\_for\_ne.append(ee)

    for ee in edge\_list\_for\_ne:

        try:

            H.remove\_edge(ee[0],ee[1])

        except Exception as e:

            print('', end = '')

        try:

            H.remove\_edge(ee[1],ee[0])

        except Exception as e:

            print('', end = '')

    H.remove\_node(ne)

    if m==0:

        for sv in edge\_list\_for\_ne:

            for dv in edge\_list\_for\_ne:

                if sv!=dv:

                    s=sv[0]

                    d=dv[0]

                    w = G[sv[0]][sv[1]][0]['weight'] +

G[dv[0]][dv[1]][0]['weight']

                    H.add\_edge(s, d, weight=w)

        try:

            return sum\_of\_shortest\_paths(H)

        except:

            return 0

    elif m==1:

        for sv in edge\_list\_for\_ne:

            for dv in edge\_list\_for\_ne:

                if  sv!=dv and sv[1]==dv[0]:

                    s=sv[0]

                    d=dv[1]

                    w = G[sv[0]][sv[1]][0]['weight'] +

G[dv[0]][dv[1]][0]['weight']

                    if s!=ne and d!=ne:

                        H.add\_edge(s, d, weight=w)

        try:

            return sum\_of\_shortest\_paths(H)

        except:

            return 0

Funkcija calculate\_sum\_of\_shortest\_paths\_without\_given\_node() izračunava sumu najkraćih puteva između svakog para vrhova u pografu H koji ne sadrži vrh koji je proslijeđen u parametru i lukova koji ulaze/izlaze iz njega. Nakon što se provedu promijene (brisanje lukova i vrha) u podgrafu H poziva se prethodno opisana funkcija sum\_of\_shortest\_paths(). Parametar G predstavlja početni graf (ne – node element) u kojemu se nalazi vrh (ne – node element) kojeg želimo ukloniti iz podgrafa. Funkcija se temelji na stvaranju podgrafa H na kojem se provode prethodno opisane promijene te poziva funkciju sum\_of\_shortest\_paths().

def calculate\_sum\_of\_shortest\_paths\_without\_given\_node(G, ne):

    H = G.copy()

    edge\_list\_for\_ne = []

    for ee in G.edges:

        if ee[1]==ne:

            edge\_list\_for\_ne.append(ee)

    for ee in edge\_list\_for\_ne:

        try:

            H.remove\_edge(ee[0],ee[1])

        except Exception as e:

            print('', end = '')

        try:

            H.remove\_edge(ee[1],ee[0])

        except Exception as e:

            print('', end = '')

    H.remove\_node(ne)

    try:

        return sum\_of\_shortest\_paths(H)

    except:

        return 0

1. Izračunajte vremensku složenost izrađenog algoritma (Pomoćna literatura: Lovrenčić, 2018)

Složenost ovog algoritma je gdje predstavlja broj vrhova grafa.

Naime, za izračun najkraćih udaljenosti između pojedinih vrhova koristili smo poboljšani Dijkstrin algoritam kojem je složenost . Za određivanje najkritičnije ulice potrebno je ovaj algoritam izvesti jednom za izračun referentne vrijednosti te ga ponoviti za svaki brid. U najgorem slučaju, taj se algoritam treba ponoviti + 1 puta, odnosno kada bi u usmjerenom grafu svaki vrh bio povezan sa svakim bridom u jednom i drugom smjeru, bilo bi bridova, što je – . Očito je ( – ) + 1 broj manji od što znači da je vremenska složenost . Dakle, zaključujemo da je ukupna vremenska složenost ovog dijela algoritma .

Nadalje, potrebno je izračunati vremensku složenost za drugi dio algoritma, odnosno za određivanje najkritičnijeg raskrižja. Za to je također potrebno izračunati najkraće udaljenosti za što koristimo poboljšani Dijkstrin algoritam kojem je složenost . Najprije za svaki vrh treba dobiti referentnu vrijednost (), a zatim za svaki vrh treba dobiti zbroj svih najkraćih puteva kada maknemo vrh i pripadne bridove za svaki vrh (). Dakle, poboljšani Dijkstrin algoritam trebamo izvesti ukupno puta. Stoga dobivamo da je ukupna vremenska složenost tog dijela algoritma .

Budući da se ta dva dijela algoritma izvode slijedno, uzimamo da je vremenska složenost cijelog algoritma . [6]

1. Razmislite o mogućem poboljšanju prometne mreže grada na osnovu dobivenih rezultata. Navedite ulicu ili ulice koje bi bile dovoljne da se smanji kritičnost prometne mreže.

* Ispis rezultata:

Kriticnost brida (ulice) (1, 2, 0)=21

Kriticnost brida (ulice) (1, 4, 0)=21

Kriticnost brida (ulice) (1, 7, 0)=11

Kriticnost brida (ulice) (2, 1, 0)=21

Kriticnost brida (ulice) (2, 3, 0)=26

Kriticnost brida (ulice) (2, 4, 0)=7

Kriticnost brida (ulice) (3, 2, 0)=26

Kriticnost brida (ulice) (3, 6, 0)=61

Kriticnost brida (ulice) (3, 9, 0)=9

Kriticnost brida (ulice) (4, 1, 0)=21

Kriticnost brida (ulice) (4, 5, 0)=106

Kriticnost brida (ulice) (5, 4, 0)=106

Kriticnost brida (ulice) (5, 6, 0)=194

Kriticnost brida (ulice) (5, 7, 0)=40

Kriticnost brida (ulice) (6, 3, 0)=61

Kriticnost brida (ulice) (6, 5, 0)=194

Kriticnost brida (ulice) (6, 8, 0)=400

Kriticnost brida (ulice) (7, 1, 0)=11

Kriticnost brida (ulice) (7, 5, 0)=40

Kriticnost brida (ulice) (7, 11, 0)=167

Kriticnost brida (ulice) (8, 6, 0)=400

Kriticnost brida (ulice) (8, 9, 0)=123

Kriticnost brida (ulice) (9, 3, 0)=9

Kriticnost brida (ulice) (9, 8, 0)=123

Kriticnost brida (ulice) (9, 14, 0)=371

Kriticnost brida (ulice) (10, 8, 0)=340

Kriticnost brida (ulice) (10, 11, 0)=110

Kriticnost brida (ulice) (10, 12, 0)=327

Kriticnost brida (ulice) (11, 7, 0)=167

Kriticnost brida (ulice) (11, 10, 0)=110

Kriticnost brida (ulice) (11, 16, 0)=18

Kriticnost brida (ulice) (12, 10, 0)=327

Kriticnost brida (ulice) (12, 13, 0)=249

Kriticnost brida (ulice) (12, 16, 0)=55

Kriticnost brida (ulice) (13, 12, 0)=249

Kriticnost brida (ulice) (13, 14, 0)=106

Kriticnost brida (ulice) (13, 15, 0)=13

Kriticnost brida (ulice) (14, 9, 0)=371

Kriticnost brida (ulice) (14, 13, 0)=106

Kriticnost brida (ulice) (14, 15, 0)=39

Kriticnost brida (ulice) (15, 13, 0)=13

Kriticnost brida (ulice) (15, 14, 0)=39

Kriticnost brida (ulice) (15, 16, 0)=14

Kriticnost brida (ulice) (16, 11, 0)=18

Kriticnost brida (ulice) (16, 12, 0)=55

Kriticnost brida (ulice) (16, 15, 0)=14

Rezultati pokazuju da je ulica između vrha 6 i vrha 8 najkritičnija, zato što ima najveću vrijednost, točnije jer bi njezino zatvaranje stvaralo najveću „gužvu“. Postoje mnoga rješenja ovog problema u realnom svijetu, pa ćemo promotriti nekolicinu njih.

Prvo rješenje je stvaranje još jedne dvosmjerne relacije između vrha 6 i 8 koja bi mogla raditi u slučaju zatvaranje prve.

Drugo rješenje je poboljšanje postojećih obilaznica, točnije smanjenje težine na bridovima obilaznice, trenutna obilaznica ide putem v6 ⇒ v3 ⇒ v9 ⇒ v8.

Posljednja mogućnost je dodavanje dvosmjernih ulica susjedima vrhova 6 i 8, dakle mogli bi dodati ulicu iz vrha 5 ili 3 u vrh 8, ili pak ulicu iz vrha 9 ili 10 u vrh 6, te se s time dobiva bolja obilaznica.

# Literatura

|  |  |
| --- | --- |
| [1] | G. Online, »Graph Online,« Graph Online, 2015. [Mrežno]. Available: https://graphonline.ru/en/. [Pokušaj pristupa 25 1 2020]. |
| [2] | p. d. s. B. Divjak, »Elf - Šetnje u grafu. Težinski grafovi,« [Mrežno]. Available: https://elf.foi.hr/pluginfile.php/17708/mod\_page/content/28/grafovi2.pdf. [Pokušaj pristupa 25 1 2020]. |
| [3] | G. v. Rossum, »Python,« Python Software Foundation, 1990. [Mrežno]. Available: https://www.python.org/. [Pokušaj pristupa 25 1 2020]. |
| [4] | N. developers, »NetworkX,« 2014. [Mrežno]. Available: https://networkx.github.io/. [Pokušaj pristupa 25 1 2020]. |
| [5] | J. Hunter, »matplotlib,« 2007. [Mrežno]. Available: https://matplotlib.org/. [Pokušaj pristupa 25 1 2020]. |
| [6] | A. Lovrenčić, Uvod u složenost algoritama, 2018. |