Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva
Zavod za elektroniku, mikroelektroniku, računalne i inteligentne sustave

Analiza velikih skupova podataka

Autor: Lucija Šikić
Ak. god. 2020./2021.

5. Laboratorijska vježba

U petoj laboratorijskoj vježbi zadatak je programski ostvariti Girvan-Newmanov algoritam za particioniranje grafa te ga primijeniti na problem detekcije zajednica u društvenim mrežama.

1. Girvan-Newmanov algoritam

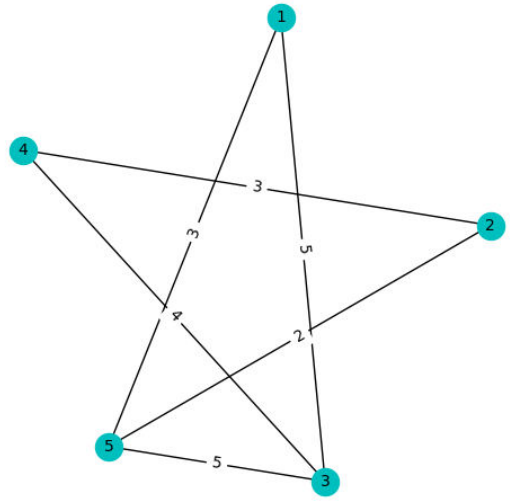
Girvan-Newmanov algoritam jest metoda particioniranja grafa objavljena 2004. godine u [1]. Metodu odlikuju sljedeće posebnosti: uporaba „razdvajajuće“ tehnike (engl. *„divisive“ technique*) i korak ponovna izračuna (engl. *recalculation step*). Primjenom „razdvajajuće“ tehnike iterativno se uklanjaju bridovi iz grafa, čime se on posljedično raspada na zajednice. Bridovi koje se uklanjanju odabiru se temeljem odabrane mjere iz skupa mjera različitosti (engl. *betweenness measure*). U tu se svrhu najčešće koristi mjera bridna centralnost, koja je povezana s brojem najkraćih puteva između svih parova čvorova grafa koji sadrže promatrani brid. Druga posebnost metode, odnosno ponavljanje izračuna definirane mjere nakon uklanjanja svakog brida, pokazala se ključnom za njezin uspjeh.

Algoritam radi na sljedeći način. U svakoj se iteraciji računa iznos mjere centralnosti svakog brida mreže te se uklanja onaj brid čija je centralnost najveća. Ukoliko više bridova ima najveću centralnost, uklanjaju se svi takvi bridovi grafa. Proces se ponavlja sve dok u mreži ima bridova.

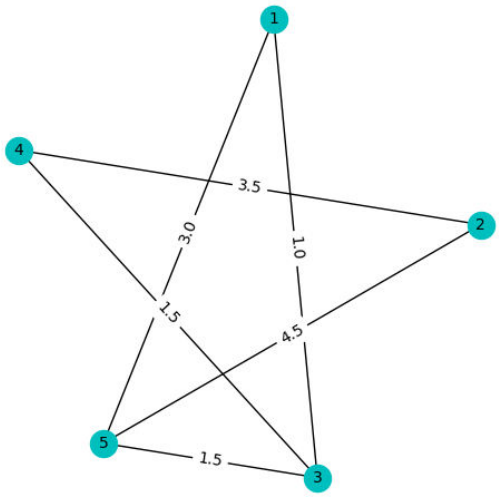
Pseudokod algoritma dan je u nastavku.

```
edges = G.get_edges()
while edges:
  edge_betweenness = calculate_betweenness(G)
  edges_to_remove = get_edges_with_highest_betweenness(edge_betweenness)
  for edge in edges_to_remove:
        remove_edge(edge, G)
  edges = G.get_edges()
```

Prikažimo na primjeru rad algoritma. Na Slici 1 i Slici 2 nalazi se isti graf. Na Slici 1 na bridovima grafa navedene su težine, dok su na Slici 2 navedeni iznosi bridne centralnosti pojedinog brida na početku rada algoritma. (Izračun bridne centralnosti opisat ćemo u odjeljku 6.)

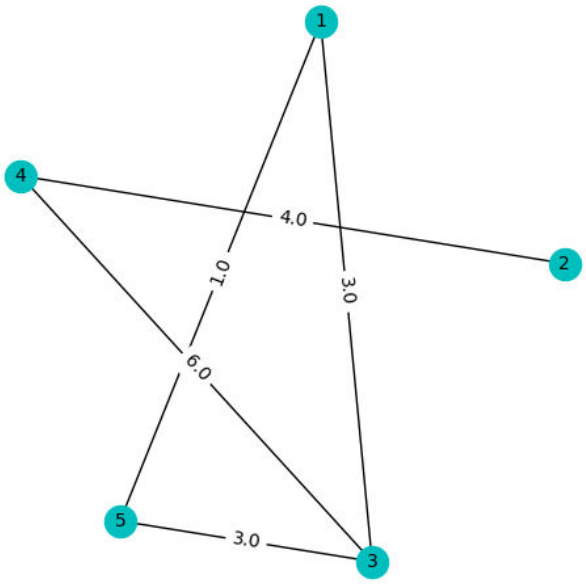


Slika 1

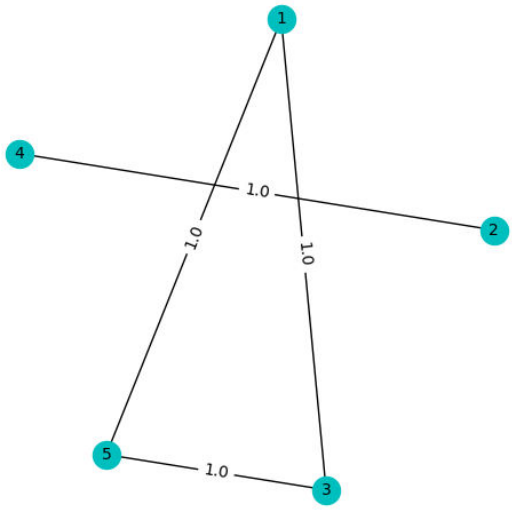


Slika 2

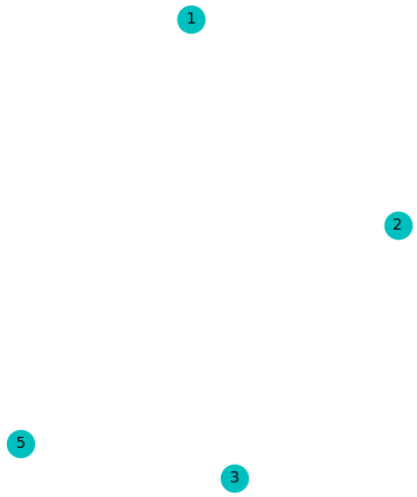
Na Slikama 3, 4 i 5 (tim redom) vidimo kako su se tijekom iteracija algoritma uklanjali bridovi te mijenjali iznosi bridne centralnosti u grafu. U svakoj su iteraciji uklonjeni bridovi najvećeg iznosa bridne centralnosti. Tako je u prvoj iteraciji uklonjen brid (2, 5), u drugoj iteraciji brid (3, 4), a u posljednjoj, trećoj iteraciji, uklonjeni su svi preostali bridovi grafa.



Slika 3



Slika 4



Slika 5

2. Struktura zajednice

Strukturu zajednice formalno definiramo kao particioniranje skupa čvorova u određen broj grupa (zajednica) tako da svi čvorovi koji pripadaju bilo kojoj takvoj grupi zadovoljavaju određeno svojstvo relativne povezanosti. Glavno je svojstvo takva particioniranja postojanje guste međusobne povezanosti između čvorova neke zajednice te istovremena rijetka povezanost među čvorovima različitih zajednica.

Na početku rada Girvan-Newmanova algoritma svi čvorovi (korisnici) pripadaju istoj zajednici. Tijekom iteracija algoritma uklanjaju se veze među korisnicima, čime broj zajednica raste. Preciznije, uklanjanjem brida u nekoj iteraciji, jedna povezana komponenta grafa može se raspasti na dvije komponente (koje jedna s drugom više nisu povezane), čime se jedna zajednica dijeli na dvije. Dva korisnika pripadaju istoj zajednici ako su izravno ili neizravno povezani (ako između njih postoji put u grafu). U konačnici je broj zajednica jednak broju korisnika u mreži.

Objasnimo na primjeru iz prvog odjeljka podjelu mreže u zajednice. Na Slici 2 i 3 postoji samo jedna zajednica, a njoj pripadaju svi korisnici mreže. Na Slici 4 došlo je do stvaranja nove zajednice uslijed uklanjanja bridova (2, 3) i (4, 5). Sada imamo dvije zajednice: {2, 4} i {1, 3, 5}. U posljednjoj iteraciji algoritma, prikazanoj na Slici 5, svaki korisnik izdvojen je u zasebnu zajednicu.

Ideja algoritma jest odrediti trenutak ovog procesa u kojem je podjela na zajednice najbolja moguća. Da bismo ocijenili koliko je određena podjela dobra, treba nam odgovarajuća mjera.

3. Modularnost

Modularnost (engl. *modularity*) je numerički indeks za procjenu odnosa jakosti veza unutar zajednica i jakosti veza među zajednicama. Drugim riječima, modularnošću mjerimo kvalitetu particioniranja grafa.

Formalno, modularnost definiramo na sljedeći način. Neka je mreža, čija je ukupna težina bridova m, podijeljena na zajednice. Nadalje, neka je V skup korisnika u mreži te neka je za korisnika u iznos ukupne težine svih bridova koji izlaze iz čvora[1] koji predstavlja tog korisnika označen kao k_u. Također, pridružimo oznaci A_{uv} težinu brida[2] između korisnika u i v. Uvedimo i funkciju $\delta(u, v)$ koja iznosi 1 ako su u promatranoj particiji grafa korisnici u i v u istoj zajednici, a 0 ako to nije slučaj. Uz navedene oznake modularnost particije možemo definirati formulom (1), pri čemu suma prolazi po svih $|V|^2$uređenih parova vrhova grafa.

$$Q = \frac{1}{2m}\sum_{u,v\in V}\left[A_{uv} - \frac{k_u k_v}{2m}\right]\delta(u, v) \quad (1)$$

Ako za neku podjelu mreže u zajednice, odnosno particiju grafa, modularnost iznosi 0, to znači da struktura zajednice u toj mreži nije snažnija od nasumično definirane strukture zajednice iste mreže. S druge strane velik iznos modularnosti pokazatelj je funkcionalne podjele mreže u zajednice.

Usporedbom iznosa modularnosti mreže po iteracijama Girvan-Newmanova algoritma moguće je pronaći optimalnu podjelu promatrane mreže.

4. Ulazni podaci

Za potrebe laboratorijske vježbe preuzet je javno dostupan skup podataka „Social circles: Twitter“ [2] koji sadrži informacije o društvenoj mreži *Twitter*. Društvena mreža sadrži preko 80 tisuća čvorova koji predstavljaju korisnike te više od 1.8 milijuna veza kojima su oni međusobno povezani, ali za ovu vježbu pripremljeni su znatno manji uzorci tih podataka jer je veći naglasak na točnosti, a manji na skalabilnosti algoritma.

Podatci koje ćemo koristiti zadani su u sljedećem obliku. U ulaznoj datoteci najprije su zapisane veze između svih korisnika, pri čemu redak oblika *“id1 id2”* znači da su korisnici s identifikatorima *id1* i *id2* međusobno povezani. Redci nakon njih opisuju svojstva korisnika: svaki redak sadrži identifikator korisnika i vektor svojstava, pri čemu 1 u vektoru označava da je korisniku dodijeljeno neko svojstvo, a 0 da mu nije dodijeljeno. Na Slikama 6 i 7 primjeri su navedenog prvog i drugog dijela ulazne datoteke. Identifikatori korisnika prirodni su brojevi.

[1] Graf je neusmjeren pa svi bridovi koji ulaze u čvor ujedno i izlaze iz njega.

[2] Ako dva korisnika nisu povezana ili ako je *u = v*, ta težina iznosi 0.

```
163629705 32122637
31477674 163629705
88491375 28719244
28719244 163629705
28719244 31477674
88491375 163629705
88491375 31477674
163629705 21938120
```

Slika 6

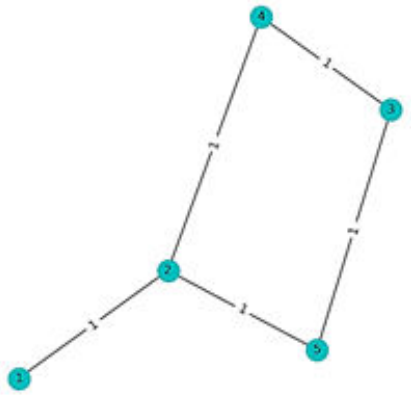
```
295724283 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
104148298 1 1 1 1 1 1 0 0 1 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
88491375 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 1 0
32122637 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
142960504 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 1
75914648 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 1
31477674 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
21938120 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
163629705 0 1 0 0 1 0 0 1 0 0 1 1 0 0 1 0 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0
28719244 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 1 1 0 0 0 1 0 1
24954077 1 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
```

Slika 7

5. Konstruiranje grafa

Graf nad kojim primjenjujemo Girvan-Newmanov algoritam gradimo tako da iz podataka o korisnicima najprije konstruiramo netežinski graf, a njegovim vezama potom dodjeljujemo iznos sličnosti dobiven iz vektora svojstava. Budući da težine veza trebaju reflektirati udaljenost dvaju korisnika, odnosno njihovu „nesličnost“, u konačnici svakom bridu dobivenog grafa težinu definiramo tako da od maksimalnog iznosa sličnosti oduzimamo njegovu sličnost umanjenu za 1.

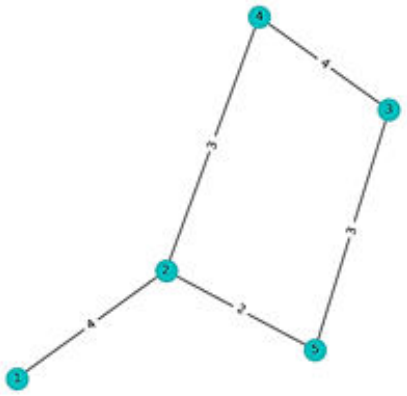
Pojasnimo opisani postupak na primjeru. Iz podataka o povezanosti korisnika konstruiran je graf na Slici 8. Svakom je bridu pridružena težina iznosa 1 te ga možemo smatrati netežinskim grafom. Svojstva korisnika za analizirani primjer prikazana su na Slici 9.



Slika 8

```
1 0 0 0 0 1
2 0 1 1 0 0
3 1 1 0 0 1
4 0 0 1 0 1
5 1 1 1 0 0
```

Slika 9



Slika 10

U sljedećem koraku za svaka dva povezana korisnika netežinskog grafa potrebno je pronaći retke u datoteci koji vektorima svojstava predstavljaju ta dva korisnika. Potom su težine veza (koje iznose 1) promatranih korisnika zamijenjene mjerom sličnosti vektora svojstava. Tu mjeru definiramo kao broj jednakih elemenata na istoj poziciji promatranih vektora. Iz datoteke prikazane na Slici 9 vidimo da je maksimalan iznos mjere sličnosti vektora svojstava jednak 5.

U posljednjem koraku konstrukcije grafa, težine bridova postavljamo tako da od maksimalnog iznosa mjere sličnosti vektora oduzimamo iznos sličnosti promatranog brida

umanjen za 1[3]. Primjerice, težina brida (3, 5), koja na početku iznosi 1, najprije je temeljem mjere sličnosti vektora svojstava postavljena na 3, potom je dobiveni iznos umanjen za 1 te oduzet od 5. Dakle, težina brida (3, 5) u konačnici iznosi: 5 – (3 – 1) = 3. Na Slici 10 nalazi se konačan graf.

6. Izračun bridne centralnosti

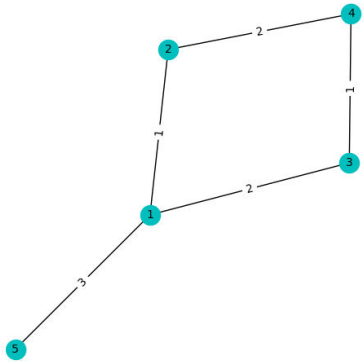
Temelj rada Girvan-Newmanova algoritma izračun je bridne centralnosti bridova u grafu. Kako bismo mogli ispravno izračunati bridnu centralnost, najprije je potrebno pronaći najkraći put između svaka dva korisnika u grafu. Ukoliko postoji samo jedan takav put, svakom se bridu na tom putu centralnost uvećava za 1[4]. Ukoliko između neka dva korisnika postoji N najkraćih puteva, tada se iterira po putevima, pri čemu se svakom bridu na trenutačno promatranu putu vrijednost centralnosti uvećava za 1/N.

Ovaj postupak može se implementirati na sljedeći način. Najprije, primjenom rekurzije možemo pronaći sve puteve koji počinju u nekom fiksnom čvoru grafa. Alternativno, umjesto rekurzije, moguće je iterativno generirati najprije sve puteve koji sadrže jedan brid, pa iz njih sve puteve koji sadrže dva brida, pa tri brida, i tako dalje. U oba slučaja treba paziti da se čvorovi unutar pojedinog puta ne ponavljaju (inače bi puteva bilo beskonačno mnogo). Među svim pronađenim putevima od čvora A do čvora B treba pronaći onaj s najkraćom duljinom u smislu zbroja težina njegovih bridova. Ako takvih ima N, tada za svaki od njih prolazimo po njegovim bridovima i uvećavamo im centralnost za 1/N.

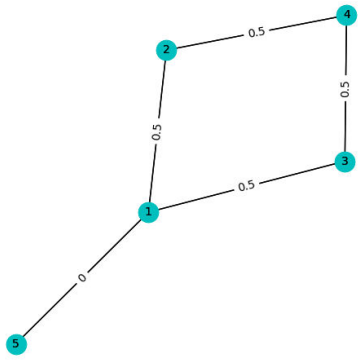
Pogledajmo za primjer graf na Slici 11, na čijim su bridovima prikazane udaljenosti korisnika. Od korisnika 1 do korisnika 4 postoje dva najkraća puta (1 – 2 – 4 i 1 – 3 – 4) pa se stoga bridovi, inicijalno postavljeni na 0, na svakom putu uvećavaju za ½, što je vidljivo na Slici 12. Od korisnika 1 do korisnika 2, 3 i 5 postoji točno jedan najkraći put (1 – 2, 1 – 3 i 1 – 5) te se centralnost tih bridova uvećava za jedan u svakom od tih slučajeva. Na Slici 13 prikazani su iznosi bridne centralnosti nakon analize svih puteva od korisnika 1 do svih njemu dohvatljivih korisnika u mreži. Sličan postupak treba provesti i za svakog drugog početnog korisnika.

[3] Ovo je potrebno provesti kako bi težine bridova grafa bile veće od 0.

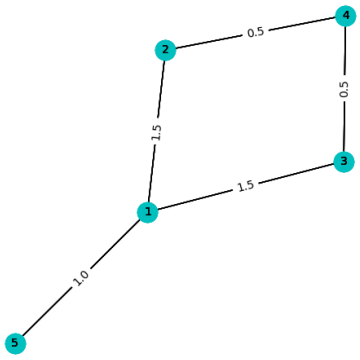
[4] Vrijednosti bridne centralnosti na početku su postavljene na 0.



Slika 11

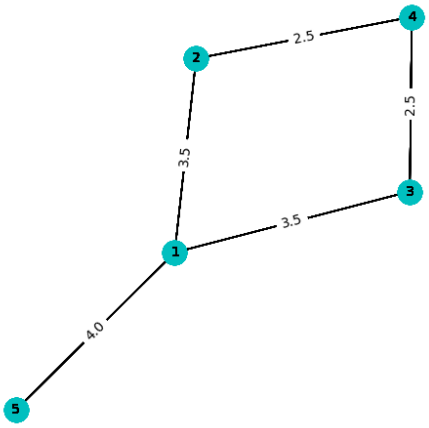


Slika 12



Slika 13

Budući da tijekom analize svih najkraćih puteva u grafu svaki put brojimo dva puta, u konačnici je iznose bridne centralnosti potrebno podijeliti brojem 2. Konačne vrijednosti bridne centralnosti promatranog grafa navedene su na bridovima na Slici 14.



Slika 14

7. Zadatak

Vaš je zadatak za ulaznu datoteku, koja se vašem programu kao i obično prosljeđuje na standardni ulaz (*stdin*), konstruirati graf na prethodno opisan način te na njega primijeniti Girvan-Newmanov algoritam. Kao što je opisano u odjeljku 4, ulazna datoteka sadrži podatke o vezama između korisnika, potom prazan redak, pa podatke o svojstvima korisnika. Za svaki je graf potrebno ispisati kojim su redom uklonjeni bridovi te optimalnu podjelu mreže u zajednice. Tako će, primjerice, za graf koji ima *N* bridova taj ispis sadržavati *N* linija u kojoj se nalaze identifikatori korisnika povezanih bridom koji se uklanja te jednu (posljednju) liniju, u kojoj će biti navedene zajednice mreže, odnosno particija grafa koja je postigla najveći iznos modularnosti tijekom iteracija. Za ulaznu datoteku na Slici 15 izlaz je dan na Slici 16.

U implementaciji nije dopuštena uporaba biblioteka za rad s grafovima kao što je *networkx*, gdje su navedeni algoritam i njegovi dijelovi već dostupni u obliku odgovarajućih funkcija, no možete je koristiti lokalno u svrhu testiranja i debugiranja svoga rješenja.

```
1 3
2 4
3 4
1 5
2 5
5 3

1 0 0 1 0 1
2 0 0 1 0 0
3 1 1 0 1 1
4 0 0 1 1 1
5 0 1 1 0 0
```

Slika 15

```
2 5
3 4
1 3
1 5
2 4
3 5
2-4 1-3-5
```

Slika 16

8. Važne napomene

U svakoj izlaznoj liniji u kojoj je naveden brid koji se uklanja, korisnici kojim su oni (bili) povezani ispisani su u rastućem poretku. Ako se u nekoj iteraciji uklanja više bridova, tada se oni ispisuju u rastućem poretku. Primjerice, u primjeru sa Slika 15 i 16 u prvoj se iteraciji uklanja brid (2, 5), u drugoj iteraciji (3, 4), a u trećoj iteraciji (1, 3), (1, 5), (2, 4) i (3, 5). Bridove je potrebno najprije zapisati tako da je identifikator prvog korisnika manji od identifikatora drugog korisnika, a potom ih, ako ih se više uklanja, sortirati najprije po prvom, a potom po drugom elementu para.

Članove svake zajednice, neovisno o njezinoj duljini, potrebno je sortirati u rastućem slijedu identifikatora. Zajednice je potrebno ispisati u rastućem poretku: najprije se ispisuju zajednice koje sadrže samo jednog korisnika, potom one koje sadrže dva korisnika i tako dalje. Kako biste generirali valjan ispis, preporuča se sortirati elemente svake zajednice u rastućem poretku, a potom skup zajednica sortirati po broju elemenata od najmanje do najveće. Ukoliko dvije zajednice imaju jednak broj elemenata, najprije se ispisuje ona čiji je prvi, tj. najmanji element, manji. Zajednice prilikom ispisa odvojite razmakom, a korisnike u zajednicama povežite znakom „-“, kao na Slici 16.

Kako bi se izbjegla nepodudarnost dobivenih i očekivanih rezultata uslijed numeričkih grešaka, ukupne modularnosti koje su po apsolutnoj vrijednosti manje od 10^{-5} potrebno je zaokružiti na 0. Iz istog je razloga sve dobivene iznose centralnosti i modularnosti potrebno zaokružiti na 4 decimale.

U ulaznoj datoteci mogu postojati vrhovi koji su navedeni sa svojim svojstvima, ali nemaju bridova (izolirani su). Njih ne treba zanemariti, i oni su dio grafa nad kojem provodimo algoritam.

Na stranicama predmeta postavljen je primjer ulazne datoteke s pripadajućim očekivanim izlazom (*labGN_primjer.zip*). Preporučamo provjeru ispravnosti temeljem zadanog primjera prije predaje vježbe na sustav SPRUT. Evaluacija ovog zadatka provodit će se na nekoliko različitih ulaznih datoteka, uključujući i spomenuti primjer.

Vremensko ograničenje za izvođenje programa za sve testne ulaze iznosit će 20s. Ulazni će primjeri imati najviše 25 čvorova, a broj veza neće prelaziti 50. Identifikatori su prirodni brojevi manji od 10^9.

Ulazna točka za rješenja u programskom jeziku *Java* treba biti u razredu **GNAlgorithm**, a ulazna točka za rješenja u programskom jeziku *Python* treba biti u datoteci ***GNAlgorithm.py***.

9. Literatura

[1] https://arxiv.org/pdf/cond-mat/0309508.pdf

Neobavezni dodatak: debug ispis – izlazni podatci uz ispis modularnosti

- *Za mali primjer (Slike 15 i 16):*

```
MODULARITY 0
2 5
MODULARITY 0
3 4
MODULARITY 0.124
1 3
1 5
2 4
3 5
2-4 1-3-5
```

- *Za primjer iz zip datoteke:*

```
MODULARITY 0.04
17758399 18393773
MODULARITY 0.485
15318882 34663865
MODULARITY 0.485
10067152 17758399
MODULARITY 0.485
```

17758399 69393657
MODULARITY 0.5015
18393773 29477388
MODULARITY 0.5015
17104585 18393773
MODULARITY 0.4714
17758399 28706024
MODULARITY 0.4714
25521487 34663865
MODULARITY 0.458
10067152 15318882
15318882 93733494
MODULARITY 0.3822
14911702 29477388
38232487 69193714
MODULARITY 0.2993
28706024 30799813
MODULARITY 0.29
4519121 19299909
4519121 69193714
17758399 34665247
18393773 19299909
18393773 69193714
26109801 40081665
34665247 66833459
40081665 44779307
MODULARITY 0.1706
17758399 34663865
34663865 34665247
34663865 66833459
MODULARITY 0.1048
10067152 69393657
14911702 29459405
17104585 29477388
25521487 30799813
25521487 42080740
30799813 42080740
93733494 456846767
17856149 18906335 20904318 28031327 28566267 34050356 26109801-40081665-
44779307 10067152-15318882-69393657-93733494-456846767 17758399-25521487-
28706024-30799813-34663865-34665247-42080740-66833459 4519121-14911702-
17104585-18393773-19299909-29459405-29477388-38232487-69193714