Zadatak 8

Grupa 8

Decembar 2024

1 Povezanost grafova

1.1 Šetnja, staza, put i kontura u multigrafu

- **Šetnja** u grafu ili multigrafu je niz čvorova i grana oblika:

$$v_1, e_1, v_2, e_2, \ldots, e_k, v_{k+1},$$

  gde je svaki čvor v_i povezan granom e_i sa sledećim čvorom v_{i+1}. Šetnja može ponavljati čvorove i grane.

- **Staza** je šetnja u kojoj se sve grane pojavljuju tačno jednom, uz mogućnost pojavljivanja čvorova.

- **Put** je šetnja u kojoj su svi čvorovi različiti. Grane se pojavljuju tačno jednom.

- **Kontura** je šetnja koja počinje i završava se u istom čvoru. Ako se sve grane u konturi pojavljuju tačno jednom, tada se ona naziva **Ojlerova kontura**.

1.2 Povezanost čvorova u grafu

Dva čvora u i v u grafu su **povezana** ako postoji put izmedju njih, tj. niz čvorova i grana koji povezuje u i v. Formalno, čvorovi u i v su povezani ako postoji niz:

$$u = v_1, e_1, v_2, e_2, \ldots, v_k = v,$$

gde svaki čvor v_i i v_{i+1} (za $i = 1, \ldots, k-1$) deli zajedničku granu e_i.

Ako su svi čvorovi u grafu medjusobno povezani, graf se naziva **povezan graf**. Ako graf nije povezan, on se može podeliti na više **komponenti povezanosti**.

Dokaz da je relacija "je povezan" relacija ekvivalencije na skupu čvorova grafa

Da bismo dokazali da je relacija "je povezan" izmedju čvorova relacija ekvivalencije na skupu čvorova grafa, potrebno je pokazati da relacija zadovoljava tri svojstva: refleksivnost, simetričnost i tranzitivnost. Ovo se odnosi na to da li su dva čvora povezana putem puta unutar grafa.

Neka $G = (V, E)$ bude graf, gde je V skup čvorova, a E skup ivica. Relacija "je povezan" izmedju čvorova $u, v \in V$ označava da postoji put od čvora u do čvora v u grafu.

1. Refleksivnost

Za svaki čvor $u \in V$, postoji trivijalan put od u do u, jer je svaki čvor povezan sa sobom. Dakle, relacija je refleksivna.

2. Simetričnost

Ako postoji put od čvora u do čvora v, to znači da možemo krenuti od u i doći do v putem niza ivica. Ovaj put može biti reverzibilan, što znači da možemo ići od v do u putem istih ivica, ali obrnutim redosledom. Dakle, ako je čvor u povezan sa čvorom v, onda je i čvor v povezan sa čvorom u, što znači da je relacija simetrična.

3. Tranzitivnost

Ako postoji put od čvora u do čvora v, i postoji put od v do čvora w, možemo formirati novi put od u do w kombinovanjem oba puta. Dakle, ako je čvor u povezan sa čvorom v, i čvor v je povezan sa čvorom w, onda je čvor u povezan sa čvorom w. Ovo znači da je relacija tranzitivna.

Zaključak

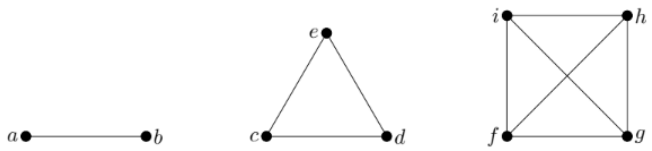
Relacija "je povezan" izmedju čvorova grafa zadovoljava sva tri svojstva (refleksivnost, simetričnost i tranzitivnost), pa je stoga relacija ekvivalencije na skupu čvorova grafa.

1.3 Komponenta povezanosti

Komponenta povezanosti je maksimalan povezan podgraf grafa, to jest svaka komponenta povezanosti grafa je podgraf koji nije sadržan ni u jednom drugom povezanom podgrafu istog grafa.

Broj komponenti povezanosti grafa G, u oznaci $\omega(G)$ jednak je broju klasa ekvivalencije u odnosu na relaciju povezanosti.

Primer: Neka je G=({a, b, c, d, e, f, g, h, i}, E) graf na slici.



Broj komponenti povezanosti datog grafa je $\omega(G) = 3$. Komponente povezanosti su podgrafovi koji su indukovani skupovima čvorova $\{a, b\}$, $\{c, d, e\}$ i $\{f, g, h, i\}$.

Lema: Multigraf G=(V, E, ψ) je povezan ako $\omega(G) = 1$.

Dokaz: G je povezan akko za svaka dva čvora u, v $\in V$ postoji uv-put u G. To dalje važi akko svi čvorovi pripadaju istoj klasi ekvivalencije u odnosu na realciju "je povezan sa", što važi akko $\omega(G) = 1$.

Definicija: Neka je $G = (V, E, \psi)$ sa osobinom $\omega(G) = k \geq 1$.

- čvor $v \in V$ je razdelni (ili artikulacioni) ako je $\omega(G - v) > k$,
- grana $e \in E$ je razdelni (ili most) ako je $\omega(G - e) > k$.

Tvrdnja:

Prost graf $G = (V, E)$ sa n cvorova i manje od $n - 1$ grana nije povezan.

Dokaz:

- Neka je $G = (V, E)$ prost graf sa $|V| = n$ i $|E| < n - 1$.
- Znamo da je minimalni broj grana potreban da bi graf bio povezan jednak $n - 1$ (sto odgovara broju grana stabla sa n cvorova).
- Ako G ima manje od $n - 1$ grana, onda u G mora postojati najmanje jedna komponenta koja nije povezana sa ostatkom grafa.
- Na primer, ukoliko posmatramo neki cvor $v \in V$, i v nema dovoljno susednih grana da bude povezan sa svim ostalim cvorovima, tada G nije povezan.
- Stoga, svaki graf sa $|E| < n - 1$ grana ne moze biti povezan.

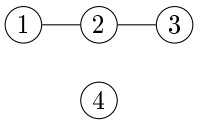
Primer: Razmotrimo graf $G = (V, E)$ sa $V = \{1, 2, 3, 4\}$ i $E = \{\{1, 2\}, \{2, 3\}\}$.

Ovaj graf ima:

- Broj cvorova: $|V| = 4$,

- Broj grana: $|E| = 2$, sto je manje od $n - 1 = 3$.

Graf nije povezan jer cvor 4 nije u istoj komponenti kao ostali cvorovi.
Ilustracija:



Kao sto se vidi iz ilustracije, cvor 4 nije povezan sa ostalim cvorovima, sto pokazuje da graf nije povezan.

Primer prostog grafa za $n = 5$

Zadovoljavanje uslova

Dati su sledeći uslovi koje graf mora zadovoljiti:

- Graf je **prost**, što znači:
  - Nema petlji (grane koje povezuju čvor sa samim sobom),
  - Nema višestrukih grana izmedju bilo koja dva čvora.
- Graf ima $n = 5$ čvorova i najmanje $n - 1 = 4$ grana.
- Graf **nije povezan**, što znači da postoji više nepovezanih komponenti.

Primer grafa

Graf sa $n = 5$ čvorova i $n - 1 = 4$ grane, koji nije povezan, može biti definisan na sledeći način:

- Čvorovi: $V = \{1, 2, 3, 4, 5\}$,
- Grane: $E = \{\{1, 2\}, \{2, 3\}, \{4, 5\}\}$.

Ovaj graf ima dve komponente:

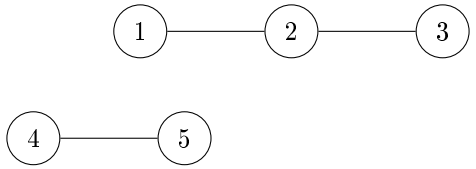
- Prva komponenta: Čvorovi $\{1, 2, 3\}$, povezani sa 2 grane ($\{1, 2\}$ i $\{2, 3\}$),
- Druga komponenta: Čvorovi $\{4, 5\}$, povezani sa 1 granom ($\{4, 5\}$).

Ukupan broj grana je $|E| = 3$.

Analiza

- Graf ima $n = 5$ čvorova, što zadovoljava uslov.
- Broj grana je $|E| = 3$, što je manje od $n - 1 = 4$, ali to je moguće jer graf nije povezan.
- Graf nije povezan jer postoje dve odvojene komponente:
  - Prva komponenta: $\{1, 2, 3\}$,
  - Druga komponenta: $\{4, 5\}$.

Vizuelni prikaz grafa



Uklanjanje grane iz konture u prostom grafu

Teorema. Neka je $G = (V, E)$ povezan prost graf koji sadrži konturu. Ako se iz G ukloni jedna grana koja pripada konturi, dobijeni graf $G' = (V, E \setminus \{e\})$ ostaje povezan.

Dokaz. Po definiciji, kontura (ciklus) u grafu je niz čvorova i grana:

$$v_1, e_1, v_2, e_2, \ldots, v_k, e_k, v_1,$$

gde su $v_1, \ldots, v_k$ medjusobno različiti čvorovi ($k \geq 3$), a $e_i = \{v_i, v_{i+1}\} \in E$ za $i = 1, \ldots, k-1$, i $e_k = \{v_k, v_1\}$. Dakle, kontura predstavlja zatvorenu putanju u grafu.

Ako uklonimo jednu granu e_j iz konture, ostaje putanja koja povezuje sve čvorove konture. Pošto u konturi postoji više puteva izmedju čvorova (jer se kontura može „zaobići“), uklanjanjem jedne grane se ne prekida povezanost izmedju čvorova konture.

Takodje, pošto je ceo graf povezan, a uklonjena grana je deo konture (dakle, njeno uklanjanje ne prekida jedinu vezu izmedju bilo koja dva čvora), to znači da i graf bez te grane ostaje povezan.

1.4 Program za rad sa grafovima

```
from itertools import permutations
from collections import deque

#klasa za cvor grafa
class Cvor:
    def __init__(self,broj) -> None:
        #kao sadrzaj stavljamo samo neki broj
        self._sadrzaj = broj
        self._stepen = 0

    @property
    def sadrzaj(self):
        return self._sadrzaj

    @property
    def stepen(self):
        return self._stepen
```

```
    def __str__(self):
        return str(self._sadrzaj)

#klasa za granu grafa
class Grana:
    def __init__(self,pocetak,kraj) -> None:
        #inicijalizacija pocetka i kraja grane, to su
            cvorovi
        self._pocetak = pocetak
        self._kraj = kraj

    @property
    def pocetak(self):
        return self._pocetak

    @property
    def kraj(self):
        return self._kraj

    #metoda koja vraca krajeve
    def krajevi(self):
        return self._pocetak,self._kraj

    #metoda koja vraca suprotni cvor
    def suprotan(self, v):
        if not isinstance(v, Cvor):
            print('v nije cvor')
            return
        if self._kraj == v:
            return self._pocetak
        elif self._pocetak == v:
            return self._kraj
        print('v nije cvor grane')

    #metoda koja omogucava da grana bude kljuc mape
    def __hash__(self):
        return hash((self._pocetak,self._kraj))

    def __str__(self):
        return str(self._pocetak.sadrzaj)+",
            "+str(self._kraj.sadrzaj)

#klasa za graf
class Graf:
    def __init__(self):
        self._ulazne_grane = {}
        self._izlazne_grane = {}

    #metoda koja proverava da li je cvor u grafu
```

```
def validacija(self,x):
    if not isinstance(x,Cvor):
        print('Objekat nije cvor')
        return False
    if x not in self._ulazne_grane:
        print('Cvor ne pripada grafu')
        return False
    return True

#metoda koja proverava da li je cvor u grafu po
    sadrzaju
def validacija_po_sadrzaju(self,x):
    for cvor in self._ulazne_grane:
        if cvor.sadrzaj==x:
            return True
    return False

#metoda koja proverava da li je grana u grafu
def validacija_grane_po_sadrzaju(self,u,v):
    for cvor in self._ulazne_grane:
        if cvor.sadrzaj==u:
            for drugi_cvor in
                self._ulazne_grane[cvor]:
                if drugi_cvor.sadrzaj==v:
                    return True

    for cvor in self._izlazne_grane:
        if cvor.sadrzaj==u:
            for drugi_cvor in
                self._izlazne_grane[cvor]:
                if drugi_cvor.sadrzaj==v:
                    return True

    return False

#metoda koja vraca broj cvorova u grafu
def broj_cvorova(self):
    return len(self._ulazne_grane)

#metoda koja vraca broj grana u grafu
def broj_grana(self):
    broj=0
    for cvor in self._ulazne_grane:
        broj+=len(self._ulazne_grane[cvor])
    return broj

#metoda koja vraca sve cvorove u grafu
def cvorovi(self):
    return self._ulazne_grane.keys()
```

```
#metoda koja vraca sve grane u grafu
def grane(self):
    skup=set()
    for cvor in self._ulazne_grane:
        for grana in
            self._ulazne_grane[cvor].values():
             skup.add(grana)
    return skup

#metoda koja vraca granu izmedju dva cvora
def grana(self,u,v):
    validacija=self.validacija(u)
    if not validacija:
        return None
    validacija=self.validacija(v)
    if not validacija:
        return None
    return self._ulazne_grane[v].get(u)

#metoda koja dodaje cvor u graf
def dodaj_cvor(self,x):
    cvor=Cvor(x)
    self._ulazne_grane[cvor]={}
    self._izlazne_grane[cvor]={}
    return cvor

#metoda koja dodaje granu u graf i poveca rang cvora
def dodaj_granu(self,u,v):
    #provera da li cvorovi postoje
    validacija=self.validacija(u)
    if not validacija:
        return
    validacija=self.validacija(v)
    if not validacija:
        return

    grana=Grana(u,v)
    self._ulazne_grane[v][u]=grana
    self._izlazne_grane[u][v]=grana
    u._stepen+=1
    v._stepen+=1

#metoda koja proverava da li je graf prost
def prost(self):
    for cvor in self._ulazne_grane:
        if len(self._ulazne_grane[cvor])>1:
            return False
    return True

#metoda koja proverava da li je graf potpuni
```

```
def potpuni(self):
    for cvor in self._ulazne_grane:
        if
            len(self._ulazne_grane[cvor])!=self.broj_cvorova()-1:
            return False
    return True

#metoda koja proverava da li su dva grafa jednaka
def jednaki(self,graf):
    if self.broj_cvorova()!=graf.broj_cvorova() or
        self.broj_grana()!=graf.broj_grana():
        return False
    for cvor in self._ulazne_grane:
        if cvor not in graf._ulazne_grane:
            return False
        for grana in self._ulazne_grane[cvor]:
            if grana not in
                graf._ulazne_grane[cvor]:
                return False
    return True

#metoda koja proverava da li su dva grafa izomorfna
def izomorfni(self, graf):
    if self.broj_cvorova() != graf.broj_cvorova()
        or self.broj_grana() != graf.broj_grana():
        return False

    stepeni1 = [cvor.stepen for cvor in
        self._ulazne_grane]
    stepeni2 = [cvor.stepen for cvor in
        graf._ulazne_grane]

    stepeni1.sort()
    stepeni2.sort()

    if stepeni1 != stepeni2:
        return False

    cvorovi_self = list(self.cvorovi())
    cvorovi_graf = list(graf.cvorovi())
    for perm in permutations(cvorovi_graf):
        mapa = {cvorovi_self[i]: perm[i] for i in
            range(len(cvorovi_self))}

        is_valid = True
        for cvor in self._ulazne_grane:
            for drugi_cvor in
                self._ulazne_grane[cvor]:
                if (mapa[cvor] not in
                    graf._ulazne_grane or
```

```
                    mapa[drugi_cvor] not in
                    graf._ulazne_grane[mapa[cvor]]):
                     is_valid = False
                     break
            if not is_valid:
                break

        if is_valid:
            return True

    return False

#metoda koja proverava da li je graf podgraf
def podgraf(self,graf):
    for cvor in self._ulazne_grane:
        ima_cvor=False
        for cvor_grafa in graf._ulazne_grane:
            if cvor_grafa.sadrzaj==cvor.sadrzaj:
                ima_cvor=True
                break
        if not ima_cvor:
            return False

        for drugi_cvor in self._ulazne_grane[cvor]:
            ima_granu=False

            for drugi_cvor_grafa in
                graf._ulazne_grane[cvor_grafa]:
                if
                    drugi_cvor_grafa.sadrzaj==drugi_cvor.sadrzaj:
                    ima_granu=True
                    break

            if not ima_granu:
                return False
    return True

#metoda koja proverava da li je graf pokrivajuci
    podgraf
def pokrivajuci_podgraf(self,graf):
    for cvor in self._ulazne_grane:
        ima_cvor=False
        for cvor_grafa in graf._ulazne_grane:
            if cvor_grafa.sadrzaj==cvor.sadrzaj:
                ima_cvor=True
                break
        if not ima_cvor:
            return False

        for drugi_cvor in self._ulazne_grane[cvor]:
```

```
                ima_granu=False

                for drugi_cvor_grafa in
                    graf._ulazne_grane[cvor_grafa]:
                    if
                        drugi_cvor_grafa.sadrzaj==drugi_cvor.sadrzaj:
                        ima_granu=True
                        break

                if not ima_granu:
                    return False

        for cvor in graf._ulazne_grane:
            ima_cvor=False
            for cvor_grafa in self._ulazne_grane:
                if cvor_grafa.sadrzaj==cvor.sadrzaj:
                    ima_cvor=True
                    break
            if not ima_cvor:
                return False

        return True

    #metoda koja proverava da li je graf indukovan
        podgraf
    def indukovan_podgraf(self,graf):

        cvorovi=[]
        for cvor in self._ulazne_grane:
            cvorovi.append(cvor.sadrzaj)
            ima_cvor=False
            for cvor_grafa in graf._ulazne_grane:
                if cvor_grafa.sadrzaj==cvor.sadrzaj:
                    ima_cvor=True
                    break
            if not ima_cvor:
                return False

            for drugi_cvor in self._ulazne_grane[cvor]:
                ima_granu=False

                for drugi_cvor_grafa in
                    graf._ulazne_grane[cvor_grafa]:
                    if
                        drugi_cvor_grafa.sadrzaj==drugi_cvor.sadrzaj:
                        ima_granu=True
                        break

                if not ima_granu:
                    return False
```

```
        for cvor in graf._ulazne_grane:
            if cvor.sadrzaj not in cvorovi:
                continue

            cvor_prvog_grafa=None
            for cvor_grafa in self._ulazne_grane:
                if cvor_grafa.sadrzaj==cvor.sadrzaj:
                    cvor_prvog_grafa=cvor_grafa
                    break

            for drugi_cvor in graf._ulazne_grane[cvor]:
                if drugi_cvor.sadrzaj not in cvorovi:
                    continue

                ima_granu=False

                for drugi_cvor_grafa in
                    self._ulazne_grane[cvor_prvog_grafa]:
                    if
                        drugi_cvor_grafa.sadrzaj==drugi_cvor.sadrzaj:
                        ima_granu=True
                        break

                if not ima_granu:
                    return False

        return True

    #metoda koja proverava da li postoji put izmedju
        dva cvora po sadrzaju
    def put(self,u,v):
        validacija=self.validacija_po_sadrzaju(u)
        if not validacija:
            print("Cvor ",u," ne postoji")
            return False
        validacija=self.validacija_po_sadrzaju(v)
        if not validacija:
            print("Cvor ",v," ne postoji")
            return False

        pocetni_cvor=None
        for cvor in self._izlazne_grane:
            if cvor.sadrzaj==u:
                pocetni_cvor=cvor
                break

        prethodnik = {pocetni_cvor.sadrzaj: None}
        poseceni=[]
        dek=deque([pocetni_cvor])
```

```
        ima=False
        while dek:
            cvor=dek.popleft()
            if cvor.sadrzaj==v:
                ima=True
                break
            poseceni.append(cvor)
            for sused in self._izlazne_grane[cvor]:
                if sused not in poseceni:
                    dek.append(sused)
                    prethodnik[sused.sadrzaj]=cvor.sadrzaj

        if ima:
            cvor=v
            ispis=""
            while cvor is not None:
                ispis+=cvor+"<-"
                cvor=prethodnik[cvor]
            ispis=ispis[:-2]
            print(ispis)
            return True

        return False

    #metoda koja proverava da li postoji kontura za
        neki cvor po sadrzaju
    def kontura(self,u):
        validacija=self.validacija_po_sadrzaju(u)
        if not validacija:
            print("Cvor ",u," ne postoji")
            return False

        pocetni_cvor=None
        for cvor in self._izlazne_grane:
            if cvor.sadrzaj==u:
                pocetni_cvor=cvor
                break

        poseceni=[]
        dek=deque([pocetni_cvor])
        prethodnik = {}

        ima=False
        while dek:
            cvor=dek.popleft()
            poseceni.append(cvor)

            ima_nov=False
            for sused in self._izlazne_grane[cvor]:
```

```
                    if sused.sadrzaj==u:
                        ima=True
                        poseceni.append(sused)
                        prethodnik[sused.sadrzaj]=cvor.sadrzaj
                        break
                    if sused not in poseceni:
                        dek.append(sused)
                        prethodnik[sused.sadrzaj]=cvor.sadrzaj
                        ima_nov=True

                if not ima_nov:
                    break

            if ima:
                cvor=u
                ispis=""
                vratio_se=False
                while not vratio_se:
                    ispis+=cvor+"<-"
                    cvor=prethodnik[cvor]
                    if cvor==u:
                        vratio_se=True
                ispis+=u
                print(ispis)
                return True

            return False

        #metoda koja proverava da li je graf povezan
        def povezan(self):
            for cvor in self._ulazne_grane:
                for drugi_cvor in self._ulazne_grane:
                    if cvor.sadrzaj!=drugi_cvor.sadrzaj:
                        ima_put=self.put(cvor.sadrzaj,drugi_cvor.sadrzaj)
                        if not ima_put:
                            return False
            return True

if __name__ == "__main__":
    grafovi={}

    izabran_graf=None
    ime_grafa=None
    while True:
        if izabran_graf is not None:
            print("Izabran graf: ",ime_grafa)
        else:
            print("Nijedan graf nije trenutno izabran")
            prvi_graf=Graf()
            ime=input("Unesite ime prvog grafa:")
```

```
        grafovi[ime]=prvi_graf
        izabran_graf=prvi_graf
        ime_grafa=ime
        continue

print("Izaberite opciju:")
print("0. Izaberi ili dodaj graf")
print("1. Dodaj cvor")
print("2. Dodaj granu")
print("3. Prikazi broj cvorova")
print("4. Prikazi broj grana")
print("5. Prikazi sve cvorove")
print("6. Prikazi sve grane")
print("7. Proveri da li postoji grana")
print("8. Proveri da li postoji cvor")
print("9. Pronadji direktnog suseda")
print("10. Stepen cvora")
print("11. Da li je prost graf")
print("12. Da li je potpuni graf")
print("13. Da li su dva grafa jednaka")
print("14. Da li su dva grafa izomorfna")
print("15. Da li je podgraf")
print("16. Da li je pokrivajuci podgraf")
print("17. Da li je indukovan podgraf")
print("18. Da li postoji put izmedju dva cvora")
print("19. Da li postoji kontura izmedju dva
    cvora")
print("20. Da li je graf povezan")
print("21. Kraj")

opcija=int(input())

if opcija==0:
    print("Izaberite opciju:")
    print("1. Izaberi graf")
    print("2. Dodaj graf")
    opcija=int(input())

    if opcija==1:
        print("Unesite koji graf zelite da
            izaberete:")
        for kljuc in grafovi:
            print(kljuc)
        unos=input()

        if unos in grafovi:
            izabran_graf=grafovi[unos]
            ime_grafa=unos
        else:
            print("Ne postoji graf sa tim
```

```
                imenom")
        elif opcija==2:
            ime=input("Unesite ime novog grafa:")
            grafovi[ime]=Graf()
            izabran_graf=grafovi[ime]
            ime_grafa=ime
        else:
            print("Nepostojeca opcija")
    elif opcija==1:
        sadrzaj=input("Unesite sadrzaj cvora:")
        cvor=izabran_graf.dodaj_cvor(sadrzaj)
        print("Dodat cvor: ",cvor)
    elif opcija==2:
        prvi=input("Unesite sadrzaj prvog cvora:")
        drugi=input("Unesite sadrzaj drugog cvora:")
        u=None
        v=None

        validacija=grafovi[ime_grafa].validacija_po_sadrzaju(prvi)
        if validacija:
            for cvor in
                grafovi[ime_grafa].cvorovi():
                if cvor.sadrzaj==prvi:
                    u=cvor
        else:
            continue
        validacija=grafovi[ime_grafa].validacija_po_sadrzaju(drugi)
        if validacija:
            for cvor in
                grafovi[ime_grafa].cvorovi():
                if cvor.sadrzaj==drugi:
                    v=cvor
        else:
            continue

        izabran_graf.dodaj_granu(u,v)
        print("Dodata grana izmedju cvora ",u," i
            cvora ",v)
    elif opcija==3:
        print("Broj cvorova u grafu:
            ",izabran_graf.broj_cvorova())
    elif opcija==4:
        print("Broj grana u grafu:
            ",izabran_graf.broj_grana())
    elif opcija==5:
        print("Cvorovi u grafu: ")
        for cvor in izabran_graf.cvorovi():
            print(cvor)
    elif opcija==6:
        print("Grane u grafu: ")
```

```
        for grana in izabran_graf.grane():
            print(grana)
    elif opcija==7:
        print("Unesite sadrzaj prvog cvora:")
        prvi=input()
        print("Unesite sadrzaj drugog cvora:")
        drugi=input()

        u=None
        v=None

        validacija=izabran_graf.validacija_po_sadrzaju(prvi)
        if validacija:
            for cvor in izabran_graf.cvorovi():
                if cvor.sadrzaj==prvi:
                    u=cvor.sadrzaj
        else:
            print("Cvor ne postoji")
            continue
        validacija=izabran_graf.validacija_po_sadrzaju(drugi)
        if validacija:
            for cvor in izabran_graf.cvorovi():
                if cvor.sadrzaj==drugi:
                    v=cvor.sadrzaj
        else:
            print("Cvor ne postoji")
            continue

        validacija=izabran_graf.validacija_grane_po_sadrzaju(u,v)
        if validacija:
            print("Grana postoji")
        else:
            print("Grana ne postoji")
    elif opcija==8:
        print("Unesite sadrzaj cvora:")
        sadrzaj=input()

        validacija=izabran_graf.validacija_po_sadrzaju(sadrzaj)
        if validacija:
            print("Cvor postoji")
        else:
            print("Cvor ne postoji")
    elif opcija==9:
        print("Unesite sadrzaj cvora:")
        sadrzaj=input()
        validacija=izabran_graf.validacija_po_sadrzaju(sadrzaj)
        if validacija:
            cvor_grafa=None
            for cvor in izabran_graf.cvorovi():
                if cvor.sadrzaj==sadrzaj:
```

```
                    cvor_grafa=cvor
                    break

            vrednosti=[]
            for cvor in
                izabran_graf._ulazne_grane[cvor_grafa]:
                vrednosti.append(cvor.sadrzaj)

            for cvor in
                izabran_graf._izlazne_grane[cvor_grafa]:
                for vrednost in vrednosti:
                    if vrednost==cvor.sadrzaj:
                        break
                vrednosti.append(cvor.sadrzaj)

            print("Direktni susedi cvora
                ",cvor_grafa," su: ",vrednosti)
        else:
            continue
    elif opcija==10:
        print("Unesite sadrzaj cvora:")
        sadrzaj=input()
        validacija=izabran_graf.validacija_po_sadrzaju(sadrzaj)
        if validacija:
            cvor=None
            for cvor_grafa in
                izabran_graf.cvorovi():
                if cvor_grafa.sadrzaj==sadrzaj:
                    cvor=cvor_grafa
                    break
            print("Stepen cvora ",cvor," je:
                ",cvor.stepen)
        else:
            continue
    elif opcija==11:
        prost=izabran_graf.prost()
        if prost:
            print("Graf je prost")
        else:
            print("Graf nije prost")
    elif opcija==12:
        potpuni=izabran_graf.potpuni()
        if potpuni:
            print("Graf je potpuni")
        else:
            print("Graf nije potpuni")
    elif opcija==13:
        print("Unesite ime drugog grafa:")
        ime=input()
        if ime in grafovi:
```

```
            graf=grafovi[ime]
            jednaki=izabran_graf.jednaki(graf)
            if jednaki:
                print("Grafovi su jednaki")
            else:
                print("Grafovi nisu jednaki")
        else:
            print("Ne postoji graf sa tim imenom")
    elif opcija==14:
        print("Unesite ime drugog grafa:")
        ime=input()
        if ime in grafovi:
            graf=grafovi[ime]
            izomorfni=izabran_graf.izomorfni(graf)
            if izomorfni:
                print("Grafovi su izomorfni")
            else:
                print("Grafovi nisu izomorfni")
        else:
            print("Ne postoji graf sa tim imenom")
    elif opcija==15:
        print("Unesite ime drugog grafa:")
        ime=input()

        if ime in grafovi:
            graf=grafovi[ime]
            podgraf=izabran_graf.podgraf(graf)
            if podgraf:
                print("Graf je podgraf")
            else:
                print("Graf nije podgraf")
        else:
            print("Ne postoji graf sa tim imenom")
    elif opcija==16:
        print("Unesite ime drugog grafa:")
        ime=input()

        if ime in grafovi:
            graf=grafovi[ime]
            pokrivajuci_podgraf=izabran_graf.pokrivajuci_podgraf(graf)
            if pokrivajuci_podgraf:
                print("Graf je pokrivajuci podgraf")
            else:
                print("Graf nije pokrivajuci
                    podgraf")
        else:
            print("Ne postoji graf sa tim imenom")
    elif opcija==17:
        print("Unesite ime drugog grafa:")
        ime=input()
```

```
        if ime in grafovi:
            graf=grafovi[ime]

            indukovan_podgraf=izabran_graf.indukovan_podgraf(graf)
            if indukovan_podgraf:
                print("Graf je indukovan podgraf")
            else:
                print("Graf nije indukovan podgraf")
        else:
            print("Ne postoji graf sa tim imenom")
    elif opcija==18:
        print("Unesite sadrzaj prvog cvora:")
        prvi=input()
        print("Unesite sadrzaj drugog cvora:")
        drugi=input()

        setnja=izabran_graf.put(prvi,drugi)
        if setnja:
            print("Postoji put izmedju dva cvora")
        else:
            print("Ne postoji put izmedju dva
                cvora")
    elif opcija==19:
        print("Unesite sadrzaj cvora:")
        sadrzaj=input()

        kontura=izabran_graf.kontura(sadrzaj)
        if kontura:
            print("Postoji kontura za cvor")
        else:
            print("Ne postoji kontura za cvor")
    elif opcija==20:
        povezan=izabran_graf.povezan()
        if povezan:
            print("Graf je povezan")
        else:
            print("Graf nije povezan")
    elif opcija==21:
        print("Kraj")
        break
    else:
        print("Nepostojeca opcija")
```

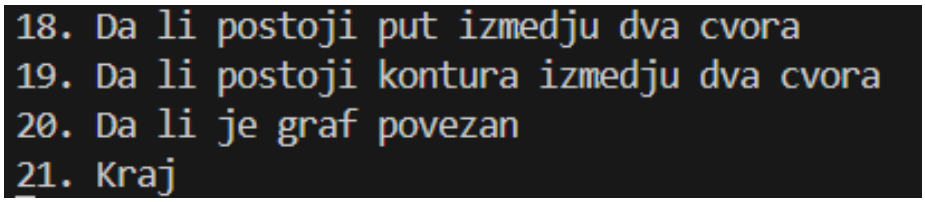


Figure 1: Prikaz dodatnih operacija nad grafovima u odnosu na prethodnu verziju programa.

1.5 Zadaci

Zadatak 1

Dokažite da je graf $G = (V, E)$ povezan ako i samo ako za svaka dva čvora $u, v \in V$ postoji put između u i v.

Rešenje:

$\Rightarrow$ Ako je graf G povezan, po definiciji povezanosti, za svaki par čvorova $u, v \in V$, postoji put između njih. Dakle, G sadrži samo jednu komponentu povezanosti.

$\Leftarrow$ Ako za svaki par čvorova $u, v \in V$ postoji put između njih, to znači da nema izolovanih podgrafova u G. Drugim rečima, svi čvorovi pripadaju jednoj komponenti povezanosti. To dokazuje da je G povezan.

Zaključak: Graf je povezan ako i samo ako postoji put između svakog para čvorova u V.

Zadatak 2

Neka je G graf sa n čvorova i $n-1$ grana. Pokažite da je G povezan ako i samo ako je G acikličan.

Rešenje:

$\Rightarrow$ Pretpostavimo da je G povezan i ima $n-1$ grana. Ako bi G imao ciklus, tada bi broj grana bio veći od $n-1$, što je u suprotnosti sa pretpostavkom. Dakle, G ne može imati ciklus i mora biti acikličan.

$\Leftarrow$ Pretpostavimo da je G acikličan i ima tačno $n-1$ grana. Po definiciji, ako je G acikličan i povezan, onda je to stablo, koje sadrži $n-1$ grana i n čvorova. Dakle, G je povezan.

Ovim je dokazano da je graf sa n čvorova i $n-1$ grana povezan ako i samo ako je acikličan.

Zadatak 3

Neka je G nepovezan graf. Koliki je minimalni broj grana koji treba dodati da bi graf postao povezan?

Rešenje:

Ako G ima k komponenti povezanosti, da bi graf postao povezan, moramo spojiti sve komponente. To se postiže dodavanjem tačno $k-1$ grana.

- Svaka dodata grana povezuje dve različite komponente.
- Kada dodamo $k-1$ grana, sve komponente se spoje u jednu.

Na primer, ako G ima 3 komponente povezanosti ($k=3$), minimalan broj grana je $k-1=2$.

Zadatak 4

Neka je G graf sa n čvorova. Odredite uslov pod kojim je G povezan u pogledu stepena čvorova.

Rešenje:

Dirak-ov teorem daje uslov za povezanost u terminima stepena čvorova:

$$\text{Ako je } \deg(v) \geq \frac{n}{2} \text{ za svaki čvor } v \in V, \text{ tada je graf povezan.}$$

Objašnjenje:

- Ako svaki čvor ima stepen bar $\frac{n}{2}$, tada graf ima dovoljno grana da obezbedi povezanost između svih čvorova.
- Ovaj uslov garantuje da graf ne može imati izolovane čvorove ili komponente.

Zadatak 5

Dokažite da je u povezanom grafu G, za svaki čvor $v \in V$, zbir udaljenosti svih čvorova od v konačan.

Rešenje:

Pošto je G povezan, za svaki čvor v postoji put do svih drugih čvorova u grafu. Definišimo $d(v,u)$ kao udaljenost između čvorova v i u. Tada je zbir udaljenosti svih čvorova od v dat kao:

$$D(v) = \sum_{u \in V} d(v,u).$$

Pošto su sve vrednosti $d(v,u)$ konačne (jer su u i v povezani), zbir $D(v)$ je konačan.

Zadatak 6

Neka je G graf sa n čvorova i m grana. Dokažite da ako G nije povezan, onda broj komponenti povezanosti ispunjava:

$$k \geq n - m.$$

Rešenje:
Ako je G nepovezan, onda se može dekomponovati na k komponenti povezanosti. Svaka komponenta povezanosti mora zadovoljiti sledeći uslov:

$$|E| \leq |V| - 1 \quad \text{(maksimalan broj grana u povezanoj komponenti)}.$$

Kombinovanjem za sve komponente, dobijamo:

$$m \leq n - k,$$

što znači da je broj komponenti povezanosti k veći ili jednak $n - m$.

Zadatak 7

Nacrtajte graf G sa $|V| = 6$ i $|E| = 4$ koji ima tačno dve komponente povezanosti.

Rešenje:
Jedan primer grafa je:

$$V = \{1, 2, 3, 4, 5, 6\}, \quad E = \{\{1, 2\}, \{2, 3\}, \{4, 5\}, \{5, 6\}\}.$$

Ovaj graf ima dve komponente povezanosti:

- Prva komponenta: čvorovi $\{1, 2, 3\}$.
- Druga komponenta: čvorovi $\{4, 5, 6\}$.

Zadatak 8

Neka su G_1 i G_2 dva povezana grafa. Definišite graf G kao njihovu uniju, tj. $G = G_1 \cup G_2$. Pod kojim uslovima je G povezan?

Rešenje:
Graf G je povezan ako i samo ako postoji barem jedna grana između čvora iz G_1 i čvora iz G_2. U suprotnom, G će imati dve komponente povezanosti:

$$\text{Ako } \exists u \in V(G_1), v \in V(G_2) \text{ takvo da } \{u, v\} \in E(G), \text{ tada je } G \text{ povezan.}$$