Zadatak 7

Grupa 8

Decembar 2024

1 Uvod u grafove

1.1 Istorijski osvrt na grafove: Problem sedam mostova Kenigsberga

Teorija grafova je matematička disciplina koja proučava strukture poznate kao grafovi, sastavljene od čvorova (tačaka) povezanih granama (linijama). Njen nastanak datira iz 18. veka kada je Leonhard Ojler (Leonhard Euler) rešio čuveni problem sedam mostova Kenigsberga.

Kenigsberg, grad u Pruskoj (današnji Kalinjingrad u Rusiji), bio je presecen rekom Pregel, sa četiri kopnene mase povezane sa sedam mostova. Gradjani su se pitali da li je moguće proći preko svih mostova tačno jednom i vratiti se na početnu tačku. Ojler je 1736. godine dokazao da to nije moguće, postavljajući temelje teoriji grafova.

Ojler je ovaj problem predstavio koristeći apstraktnu strukturu grafa: kopnene mase je predstavio kao čvorove, a mostove kao grane koje ih povezuju. Definisao je koncept **Ojlerove ture**, puta u grafu koji prolazi svaku granu tačno jednom, i pokazao da za postojanje takve ture čvorovi grafa moraju imati paran stepen (broj grana koje iz njih izlaze). Problem Kenigsberga nije ispunjavao ovaj uslov, što ga čini nemogućim.

1.2 Oblasti primene teorije grafova

Teorija grafova ima široku primenu u različitim naučnim i tehničkim oblastima, uključujući:

1. **Informatika i računarstvo**:
    - Algoritmi za pretragu i optimizaciju mreža (npr. algoritmi za pretragu u dubinu i širinu).
    - Društvene mreže i analize povezanosti.
    - Raspodela resursa u računarstvu (npr. rasporedjivanje zadataka).
2. **Transport i logistika**:

- Planiranje optimalnih ruta (npr. GPS navigacija, problem trgovačkog putnika).
- Upravljanje železničkim i avionskim mrežama.

3. **Električne mreže i komunikacije**:
   - Dizajniranje efikasnih električnih mreža.
   - Modelovanje i analiza interneta i telekomunikacionih mreža.
4. **Biologija i hemija**:
   - Analiza molekularnih struktura (npr. hemijski grafovi).
   - Proučavanje ekosistema i interakcija izmedju vrsta.
5. **Društvene nauke**:
   - Analiza društvenih mreža (npr. istraživanje uticaja i propagacije informacija).
   - Organizacija timova i mreža saradnje.
6. **Ekonomija i finansije**:
   - Modelovanje tržišta i mreža trgovine.
   - Analiza tokova novca i resursa.

Teorija grafova pruža moćne alate za rešavanje problema u svim ovim oblastima, omogućavajući jednostavno modelovanje kompleksnih sistema.

1.3 Definicija usmerenog i neusmerenog multigrafa

Definicija. *Usmeren multigraf* (multidigraf) je uredjena trojka $G = (V, E, \psi)$, gde je:

(i) $V \neq \emptyset$ konačan skup čvorova,

(ii) E skup grana, pri čemu je $V \cap E = \emptyset$, i

(iii) $\psi : E \to \{(u, v) : u, v \in V, u \neq v\}$ funkcija incidencije.

Definicija. *Neusmeren multigraf* je uredjena trojka $G = (V, E, \psi)$, gde je:

(i) $V \neq \emptyset$ konačan skup čvorova,

(ii) E skup grana, pri čemu je $V \cap E = \emptyset$, i

(iii) $\psi : E \to \{\{u, v\} : u, v \in V, u \neq v\}$ funkcija incidencije.

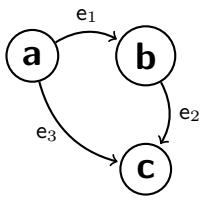
Razlika izmedju usmerenog i neusmerenog multigrafa je u tome sto funkcija incidencije u neusmerenom multigrafu preslikava grane u *neuredjene parove* cvorova.

Primeri

Primer usmerenog multigrafa

Neka je $G = (V, E, \psi)$, gde je:

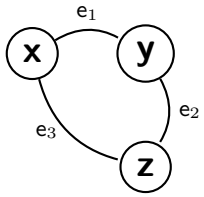
- $V = \{a, b, c\}$,
- $E = \{e_1, e_2, e_3\}$,
- $\psi(e_1) = (a, b)$,
- $\psi(e_2) = (b, c)$,
- $\psi(e_3) = (a, c)$.



Primer neusmerenog multigrafa

Neka je $G = (V, E, \psi)$, gde je:

- $V = \{x, y, z\}$,
- $E = \{e_1, e_2, e_3\}$,
- $\psi(e_1) = \{x, y\}$,
- $\psi(e_2) = \{y, z\}$,
- $\psi(e_3) = \{x, z\}$.



1.4 Definicija prostog grafa

Definicija. *Prost graf* je uredjena dvojka $G = (V, E)$, gde je:

(i) $V \neq \emptyset$ konačan skup čvorova (temena),

(ii) $E \subseteq \{\{u, v\} : u, v \in V, u \neq v\}$ skup grana (neuredjenih parova različitih čvorova).

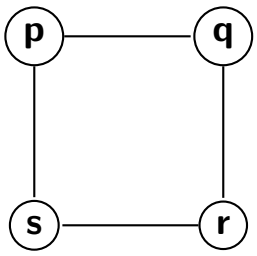
U prostom grafu:

- izmedju svaka dva čvora može postojati najviše jedna grana,
- nisu dozvoljene petlje (grane koje povezuju čvor same sa sobom).

Primer prostog grafa

Neka je $G = (V, E)$, gde je:

- $V = \{p, q, r, s\}$,
- $E = \{\{p, q\}, \{q, r\}, \{r, s\}, \{p, s\}\}$.



1.4.1 Osnovni pojmovi vezani za proste grafove

Neka je $G = (V, E)$ prost graf.

- **Susedni čvorovi**: Dva čvora $u, v \in V$ su *susedni* ako su spojeni granom, tj. ako važi $\{u, v\} \in E$.
- **Grana incidentna na čvor**: Kažemo da je grana $e = \{u, v\} \in E$ *incidentna* na čvorove u i v.
- **Stepen čvora**: *Stepen* čvora $v \in V$, označen sa $\deg_G(v)$, predstavlja broj čvorova koji su susedni sa v, tj. broj grana incidentnih na v.
- **Minimalni stepen grafa**: Označava se sa $\delta(G)$ i predstavlja najmanji stepen bilo kog čvora u grafu:

$$\delta(G) = \min_{v \in V} \deg_G(v).$$

- **Maksimalni stepen grafa**: Označava se sa $\Delta(G)$ i predstavlja najveći stepen bilo kog čvora u grafu:

$$\Delta(G) = \max_{v \in V} \deg_G(v).$$

1.4.2 Osobine prostih grafova

Lema. Neka je $G = (V, E)$ prost graf. Tada važi:

$$\sum_{v \in V} \deg_G(v) = 2|E|.$$

Teorema. Prost graf ima paran broj čvorova neparnog stepena.

Posledica. Ako je broj čvorova prostog grafa neparan, onda u njemu postoji bar jedan čvor parnog stepena.

Posledica. Neka je $G = (V, E)$ prost graf sa $|V| = n$ i $|E| < n$. Tada postoji čvor $v \in V$ takav da je $\deg_G(v) \leq 1$.

Neke specijalne klase prostih grafova

Kompletan graf

Kompletan graf K_n je prost graf u kome su svi čvorovi medjusobno povezani, tj. za svaka dva različita čvora postoji tačno jedna grana koja ih spaja.

Bipartitan graf

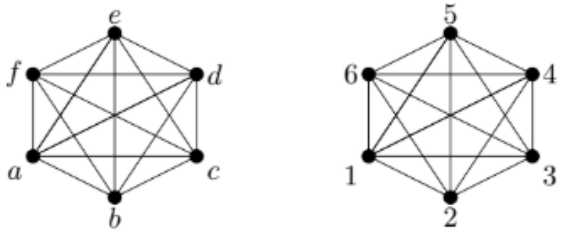
Graf $G = (V_1 \cup V_2, E)$ je *bipartitan* ako su ispunjeni sledeći uslovi:

- Skup čvorova V je podeljen na disjunktne podskupove V_1 i V_2,
- Svaka grana povezuje jedan čvor iz V_1 i jedan iz V_2.

Ako je svaki čvor iz V_1 povezan sa svakim čvorom iz V_2, onda je graf *kompletan bipartitan* i označava se sa $K_{m,n}$, gde je $|V_1| = m$, $|V_2| = n$.

1.5 Jednakost i izomorfizam

Jednakost - možemo reći da iz definicije grafa direktno sledi da su dva grafa jednaka akko imaju jednake skupove čvorova i jednake skupove grana. Tako grafovi G1, čiji je skup čvorova V(G1)={a, b, c, d, e, f}, i G2, čiji je skup čvorova V(G2)={1, 2, 3, 4, 5, 6} nisu jednaki.



G_1 G_2

Izomorfizam - za grafove koji imaju osobinu da preimenovanjem čvorova postaju jednaki kažemo da su izomorfni.

Definicija: Neka su $G_1 = (V_1, E_1)$ i $G_2 = (V_2, E_2)$ prosti grafovi. Kažemo da je G_1 izomorfan sa G_2 ako postoji bijektivno preslikavanje $h : V_1 \to V_2$ sa osobinom

$$u, v \in E_1 \quad \text{akko} \quad \{h(u), h(v)\} \in E_2.$$

Za takvu funkciju h kažemo da je izomorfizam grafa G_1 u graf G_2.

Jedan izomorfizam bi bio:

$$h = \begin{pmatrix} a & b & c & d & e & f \\ 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix}$$

Potrebni uslovi za izomorfizam:

Teorema: Neka su dati izomorfni grafovi G_1=(V_1, E_1) i G_2=(V_2, E_2). Tada je:

1. $|V_1| = |V_2|$;
2. $|E_1| = |E_2|$;
3. $\deg_{G_1}(v) = \deg_{G_2}(h(v))$ za svaki čvor $v \in V_1$;

Dokaz: Sve tri osobine slede direktno iz definicije izomorfnih grafova.

Primer dva neizomorfna grafa koja zadovoljavaju potrebne uslove za izomorfizam

Graf G_1

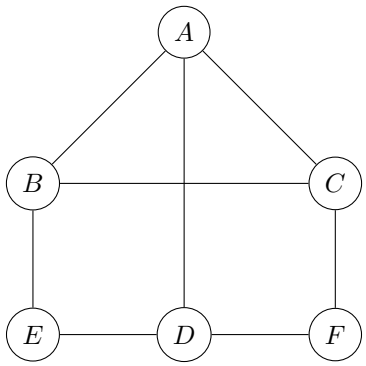
Graf G_1 ima sledeće čvorove i grane:

- Čvorovi: $V = \{A, B, C, D, E, F\}$,
- Grane: $E = \{(A, B), (A, C), (A, D), (B, C), (B, E), (C, F), (D, E), (D, F)\}$.

Distribucija stepena čvorova:

$$\deg(A) = 3, \deg(B) = 3, \deg(C) = 3, \deg(D) = 3, \deg(E) = 2, \deg(F) = 2.$$

Vizualizacija grafa G_1:



Graf G_2

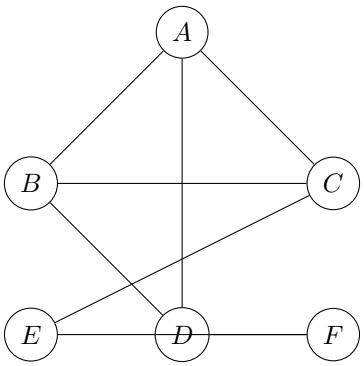
Graf G_2 ima sledeće čvorove i grane:

- Čvorovi: $V = \{A, B, C, D, E, F\}$,
- Grane: $E = \{(A, B), (A, C), (A, D), (B, C), (B, D), (C, E), (D, F), (E, F)\}$.

Distribucija stepena čvorova:

$$\deg(A) = 3, \deg(B) = 3, \deg(C) = 3, \deg(D) = 3, \deg(E) = 2, \deg(F) = 2.$$

Vizualizacija grafa G_2:



Dokaz da grafovi nisu izomorfni

Iako oba grafa imaju istu distribuciju stepena čvorova $\{3, 3, 3, 3, 2, 2\}$, struktura grafova je različita:

- Graf G_1 sadrži tročlanu kliku $\{A, B, C\}$ (tri medjusobno povezana čvora).
- Graf G_2 nema tročlanu kliku. Umesto toga, ima ciklus B, C, D, E, F sa dodatnim čvorom A povezanom sa B, C, D.

Zbog ove razlike u strukturi, ne postoji bijektivno preslikavanje čvorova koje čuva veze izmedju čvorova, što dokazuje da grafovi nisu izomorfni.

Izomorfizam kao relacija ekvivalencije

Neka skup svih grafova označimo sa $\mathcal{G}$. Definišemo binarnu relaciju ρ na skupu $\mathcal{G}$ tako da za dva grafa G_1 i G_2 važi $G_1 \, \rho \, G_2$ ako i samo ako su G_1 i G_2 izomorfni.

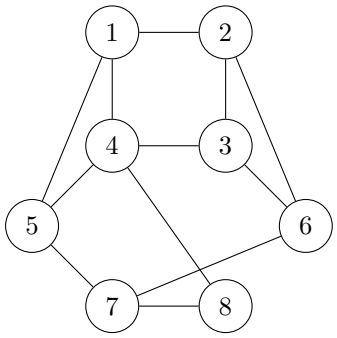
Refleksivnost: Svaki graf G je izomorfan samom sebi, jer postoji identitetno preslikavanje $f : V(G) \to V(G)$, koje čuva veze.

Simetričnost: Ako je G_1 izomorfan sa G_2, postoji bijekcija $f : V(G_1) \to V(G_2)$. Obrnuta funkcija f^{-1} je takodje bijekcija koja čuva veze, pa je G_2 izomorfan sa G_1.

Tranzitivnost: Ako je G_1 izomorfan sa G_2 preko bijekcije f, i G_2 izomorfan sa G_3 preko bijekcije g, tada je kompozicija $g \circ f$ bijekcija izmedju $V(G_1)$ i $V(G_3)$ koja čuva veze, pa je G_1 izomorfan sa G_3.

Budući da relacija R zadovoljava refleksivnost, simetričnost i tranzitivnost, zaključujemo da je izomorfizam relacija ekvivalencije na skupu grafova.

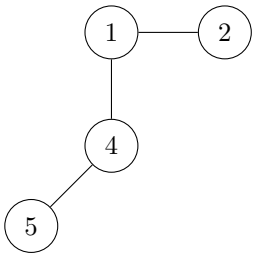
1.6 Operacije nad grafovima



Originalni graf G

Definicije

1. **Podgraf:**
Podgraf $G_1 = (V_1, E_1)$ grafa $G = (V, E)$ je graf sa osobinom $V_1 \subseteq V$ i $E_1 \subseteq E$. G_1 je pravi podgraf grafa G ako je $V \neq V_1$. Ovde treba primetiti da je G_1 takodje graf, što znači da je $E_1 \subseteq \binom{V_1}{2}$.

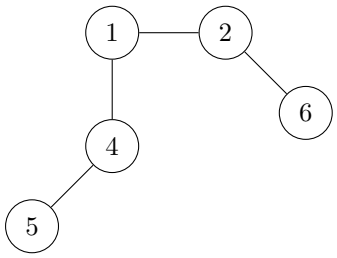


Podgraf G_1

2. **Podgraf indukovan skupom čvorova:**
Podgraf indukovan skupom čvorova $V_1 \subseteq V$ grafa $G = (V, E)$ je graf $G_1 = (V_1, E_1)$ sa osobinom

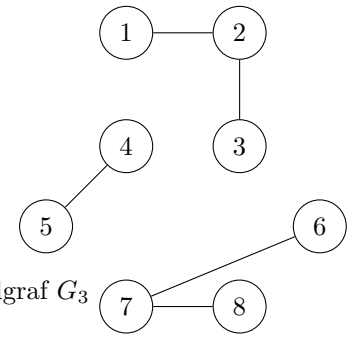
$$E_1 = E \cap \binom{V_1}{2},$$

tj. sa osobinom da je $\{u, v\}$ grana u G_1 ako i samo ako $u, v \in V_1$ i $\{u, v\} \in E$.



Indukovani podgraf G_2

3. **Pokrivajući podgraf:**
Pokrivajući podgraf je podgraf G_1 koji sadrži sve čvorove grafa G ($V_1 = V$), ali može sadržavati podskup grana ($E_1 \subseteq E$).



Pokrivajući podgraf G_3

1.7 Program za rad sa grafovima

```
    from itertools import permutations

#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

##testiranje
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. 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("Kraj")
        break
    else:
        print("Nepostojeca opcija")
```

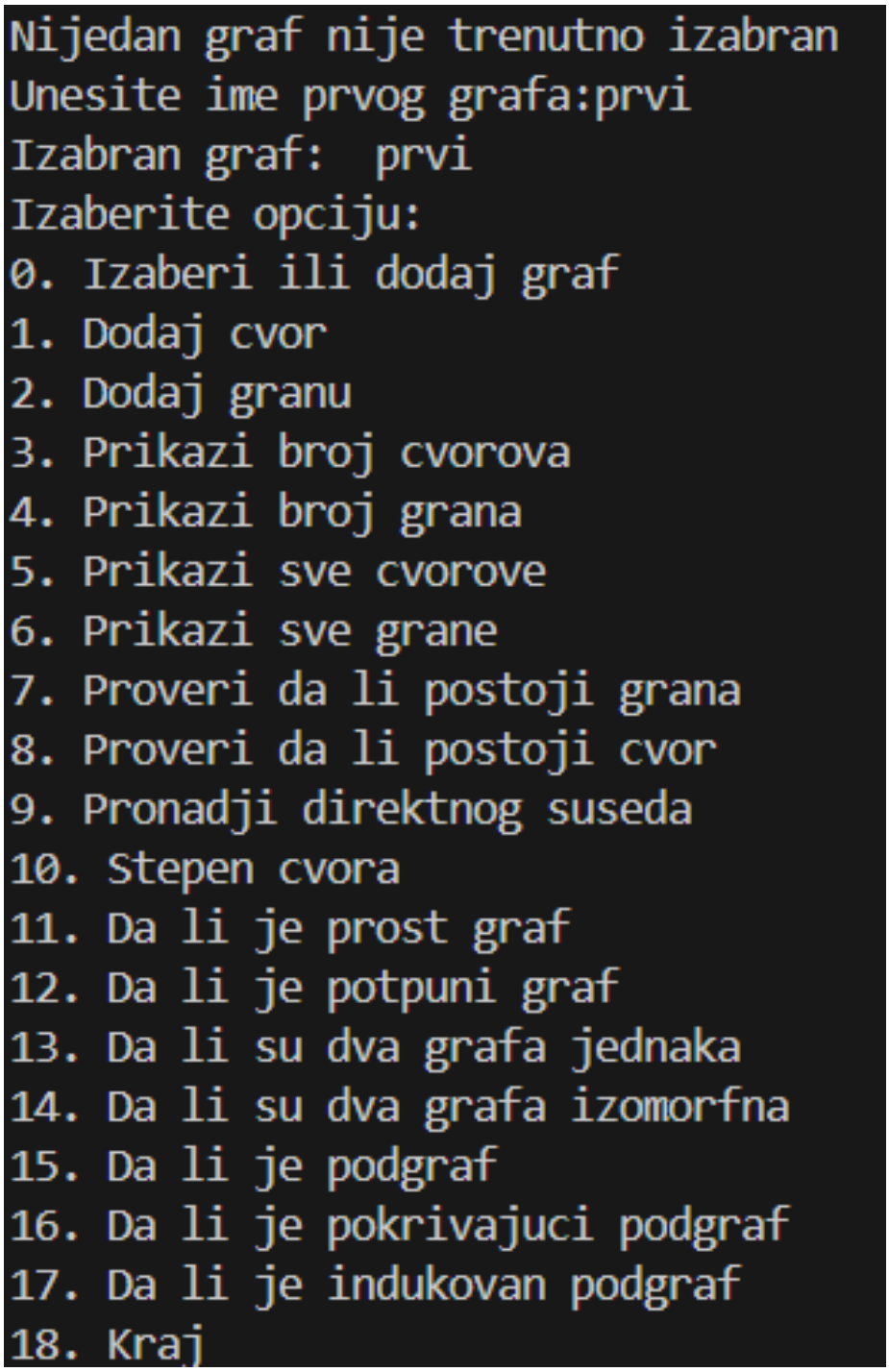


Figure 1: Prikaz svih operacija nad grafovima koje program podržava.

1.8 Zadaci

Zadatak 1

Neka je $G = (V, E)$ graf sa $|V| = n$ i $|E| = m$. Pokažite da je:

$$\sum_{v \in V} \deg(v) = 2m.$$

Rešenje: Brojeći krajeve svih grana u grafu, svaka grana doprinosi dvema čvorovima (krajevima). Dakle, zbir svih stepena čvorova jednak je dvostrukom broju grana:

$$\sum_{v \in V} \deg(v) = 2|E| = 2m.$$

Zadatak 2

Neka je G jednostavan graf sa n čvorova. Koliki je maksimalni broj grana u grafu?

Rešenje: Maksimalni broj grana se postiže kada postoji veza izmedju svakog para čvorova. U tom slučaju:

$$|E| = \binom{n}{2} = \frac{n(n-1)}{2}.$$

Zadatak 3

Neka je G bipartitni graf sa particijama V_1 i V_2, gde $|V_1| = m$ i $|V_2| = n$. Koliki je maksimalni broj grana?

Rešenje: U bipartitnom grafu grane postoje samo izmedju čvorova iz različitih particija. Maksimalni broj grana je:

$$|E|_{\max} = m \cdot n.$$

Zadatak 4

Pokažite da je u svakom grafu broj čvorova neparnog stepena paran.

Rešenje: Iz teoreme o stepenu čvorova:

$$\sum_{v \in V} \deg(v) = 2|E|.$$

Pošto je desna strana parna, zbir stepena čvorova mora biti paran. Čvorovi neparnog stepena mogu doprineti samo u parnom broju.

Zadatak 5

Konstruisati graf G sa 6 čvorova tako da su stepeni čvorova $\{3, 3, 2, 2, 2, 2\}$.

Rešenje: Jedan mogući graf:

- Spojite dva čvora stepena 3 medjusobno i sa po dva čvora stepena 2.
- Ostatak čvorova stepena 2 povežite tako da im stepen bude ispunjen.

Graf je konstruisan i ispunjava date uslove.

Zadatak 6

Nacrtajte $K_{3,3}$ i dokažite da nije planaran graf.

Rešenje: $K_{3,3}$ je potpuni bipartitni graf sa $|V_1| = 3$ i $|V_2| = 3$. Broj grana je $|E| = 9$. Po Kuratovskom teoremu, graf je neplanaran jer sadrži podgraf homeomorfan $K_{3,3}$ ili K_5.