Univerzitet u Beogradu

Matematički fakultet

Master rad

Programska realizacija konstruktivnog dokaza teoreme o indukovanom bojenju grafova sa tri boje

Student:
Marinela Parović

Mentor:
dr Miodrag Živković

7. decembar 2018

Sadržaj:

1 Sažetak

Pored klasičnih bojenja čvorova i/ili grana grafa, u posljednje vrijeme razmatraju se i varijante indukovanog bojenja grafova. Poseban slučaj je bojenje grana bojama iz zadatog skupa koje indukuje bojenje čvorova, tako da svakom čvoru odgovara „boja" jednaka multi-skupu boja grana susjednih čvorova. Pri tome se zahtijeva da susjedni čvorovi budu obojeni različitim „bojama". Preciznije, multi-skup susjed-razlikujuće bojenje grana grafa je dodjela boja granama grafa tako da se za svaku granu grafa razlikuju multi-skupovi boja grana incidentnih čvorovima koji su krajevi te grane. Bojan Vučković je u radu [1] dokazao da se svaki neusmjeren normalan graf (graf bez izolovanih grana) može na ovakav način obojiti sa tri boje, što je u opštem slučaju i optimalan broj boja. Ova teorema je dokaz hipoteze koja je postavljena 2004. godine, i poboljšanje je ranije poznatog rezultata da je ovakvo bojenje neusmjerenih grafova bez izolovanih grana moguće realizovati sa četiri boje.

U ovom radu je opisan postupak multi-skup susjed-razlikujućeg 3-bojenja grana grafa, koji je implementiran u programskom jeziku *Python* korišćenjem biblioteke *networkx* za rad sa grafovima. Program je primijenjen na neke familije grafova kao i na svih 11716571 povezanih grafova sa tačno 10 čvorova. U radu se takođe razmatra i algoritam slučajnog bojenja grana grafa i broj boja koje su potrebne ovom algoritmu za uspješno multi-skup susjed-razlikujuće bojenje grafa na svim povezanim grafovima sa najmanje 3, a najviše 10 čvorova.

2 Uvod i motivacija

Ukoliko je na mapi Južne Amerike (slika 1) potrebno obojiti države tako da su svake dvije države koje imaju zajedničku granicu obojene različitim bojama, mapa se može obojiti sa četiri boje. Pitanje je da li je ovo tačno za svaku mapu? Primijetimo prvo da države na mapi Južne Amerike nije moguće obojiti sa manje od četiri boje tako da svake dvije države koje imaju zajedničku granicu budu obojene različitim bojama. Naime, svake dvije od država Brazil, Argentina, Bolivija i Paragvaj imaju zajedničku granicu, pa su za bojenje ove četiri države potrebne četiri boje.

Vjerovatno je jasno zašto želimo da susjedne države obojimo različitim bojama. Jedan od razloga za to može biti kako bismo ih lakše razlikovali na mapi. Međutim, manje je jasno zašto bismo pomislili da su četiri boje dovoljne za bojenje država na svakoj mapi. Vjerovatno možemo zamisliti veoma komplikovanu mapu sa velikim brojem država od kojih se neke graniče sa nekoliko država i koja je konstruisana tako da je možda potrebno mnogo boja kako bi se mapa uspješno obojila na željeni način.

Iako ovaj problem može djelovati kao zanimljivo pitanje, ali ne mnogo više od toga, ispostavlja se da je ovo pitanje intrigiralo mnoge poznate matematičare decenijama. Njihovi pokušaji ka pronalaženju dokaza da je za bojenje proizvoljne mape dovoljno četiri boje značajno su doprinijeli razvoju oblasti poznate kao *teorija grafova*, a posebno oblasti bojenja grafova poznate kao *hromatska teorija grafova*. Ovaj problem poznat je pod nazivom „Problem četiri boje“.

Problem četiri boje 1. *Da li se države na svakoj mapi mogu obojiti sa najviše četiri boje tako su svake dvije države sa zajedničkom granicom obojene različitim bojama?*

Ovaj problem je dugo bio neriješen. Prvi put se pominje još 1852. godine, a njime su se bavili Augustus de Morgan (*Augustus de Morgan*), Vilijam Rovan Hamilton (*William Rowan Hamilton*), Artur Kejli (*Arthur Cayley*), Džordž Dejvid Birkhof (*George David Birkhoff*) kao i Alfred Kempe (*Alfred Kempe*), koji je dao pogrešan dokaz za bojenje mape sa četiri boje, koji je smatran ispravnim cijelu deceniju. Kempeov doprinos je ipak bio koristan. On je predložio da se mapi pridruži graf i to tako da se svakoj državi pridruži čvor i da se dva čvora povežu granom u grafu ako i samo ako dvije njima odgovarajuće države imaju zajedničku granicu. Ovim se dobija graf dualan mapi. Na taj način, problem bojenja država na mapi tako da svake dvije susjedne države budu obojene različitim bojama svodi se na problem bojenja čvorova grafa dualnog mapi, tako da su svaka dva čvora povezana granom obojena različitim bojama. Graf je planaran ako se može predstaviti u ravni tako da mu se grane međusobno ne sijeku. Ispostavlja se da je graf dualan mapi planaran, pa problem četiri boje dobija novu formulaciju u terminima grafova.

Problem četiri boje 2. *Da li se čvorovi svakog planarnog grafa mogu obojiti sa najviše četiri boje tako da su svaka dva čvora koja su povezana granom obojena različitim bojama?*

Ovaj problem je konačno riješen 1976. godine. Haken i Appel su, uz pomoć računara, dokazali da je ispravno bojenje čvorova proizvoljnog planarnog grafa moguće izvesti sa četiri boje.

Na početku razvoja hromatske teorije grafova razmatrana su „obična“ bojenja čvorova i grana grafa. *Bojenje čvorova grafa* je dodjela boja svakom od čvorova grafa. Bojenje se smatra ispravnim ukoliko su svaka dva susjedna čvora grafa obojena različitim bojama. *Bojenje grana grafa* je dodjela boja svakoj od grana grafa. Ovo bojenje se smatra ispravnim ukoliko su svake dvije susjedne grane grafa obojene različitim bojama. Međutim, hromatska teorija grafova čiji je razvoj inspirisan željom da se riješi problem četiri boje je otišla dalje od bojenja čvorova i grana grafa, pa se danas razmatraju bojenja definisana na različite načine. Jedno od takvih bojenja je multi-skup susjed-razlikujuće bojenje grafova koje je tema ovog rada.

Slika 1: Bojenje država na mapi Južne Amerike sa četiri boje tako da su svake dvije susjedne države obojene različitim bojama.

3 Osnovni pojmovi

Definicija 3.1. *Graf G je konačan neprazan skup V objekata koje nazivamo čvorovi, zajedno sa skupom E koji sadrži dvo-elementne podskupove skupa V koje nazivamo granama grafa G.*

Skup čvorova grafa G označavamo sa $V(G)$, dok skup grana grafa G označavamo sa $E(G)$. Za element $\{u, v\}$ koristi se skraćena oznaka uv. U radu se razmatraju samo neusmjereni grafovi, stoga je redosljed čvorova u skupu $\{u, v\}$ nebitan, odnosno za sve čvorove $u, v \in V$, ako je $uv \in E$, onda je i $vu \in E$.

Definicija 3.2. *Granu koja spaja čvor sa njim samim nazivamo petlja.*

Definicija 3.3. *Graf je prost ukoliko nema petlji.*

U radu su razmatrani samo prosti grafovi.

Definicija 3.4. *Dva čvora u i v su susjedna u grafu G ukoliko postoji grana $uv \in E(G)$. Dvije grane su susjedne ukoliko imaju zajednički čvor. Čvor $u \in V(G)$ je incidentan sa granom $e \in E(G)$ ukoliko je $e = uv$, za neki čvor $v \in V(G)$.*

Definicija 3.5. *Za graf G kažemo da je potpun ili kompletan ukoliko su svaka dva međusobno različita čvora susjedna u G. Kompletan graf sa n čvorova označavamo sa K_n.*

Definicija 3.6. *Za skup čvorova $S \subseteq V(G)$ kažemo da je dominirajući skup grafa G ukoliko svaki čvor iz skupa $V(G)$ ili pripada skupu S ili bar jedan njemu susjedan čvor u G pripada skupu S.*

Definicija 3.7. *Za skup čvorova $S \subseteq V(G)$ kažemo da je nezavisan skup u odnosu na graf G ukoliko ne sadrži dva čvora koji su susjedni u grafu G.*

Definicija 3.8. *Za graf H kažemo da je podgraf grafa G ukoliko je $V(H) \subseteq V(G)$ i $E(H) \subseteq E(G)$. Za skup $S \subseteq V(G)$ graf $G[S]$ je indukovani podgraf grafa G skupom S ukoliko ima skup čvorova S i dva čvora iz S su susjedni u $G[S]$ ako i samo ako su susjedni u G.*

Definicija 3.9. *Za dva čvora u i v grafa G $uv-$put u G je niz čvorova koji počinje sa u, završava se sa v i takav je da su svaka dva uzastopna čvora na tom putu susjedni u G. Put u grafu G u kome nema ponavljanja čvorova nazivamo prostim putem. Put kod koga su prvi i posljednji čvor jednaki nazivamo zatvorenim putem. Zatvoreni prost put koji se sastoji od bar 3 čvora nazivamo ciklus. Ciklus dužine k, za $k \geq 3$, označavamo sa C_n.*

Definicija 3.10. *Dva čvora u i v grafa G su povezana ukoliko u G postoji $uv-$put.*

Definicija 3.11. *Graf G je povezan ukoliko su svaka dva čvora u G međusobno povezana.*

Definicija 3.12. *Za povezani, pravi podgraf H grafa G kažemo da je komponenta od G ukoliko H sadrži maksimalni podskup čvorova grafa G koji je povezan.*

Definicija 3.13. *Za graf kažemo da je normalan ukoliko nema izolovanih grana.*

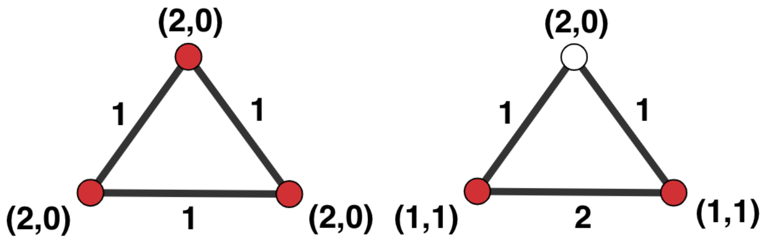
Definicija 3.14. *Graf G je bipartitan ukoliko postoji ispravno bojenje njegovih čvorova sa dvije boje.*

Drugim riječina, graf G je bipartitan ukoliko se skup njegovih čvorova $V(G)$ može predstaviti kao disjunktna unija skupova U i W tako da svaka grana iz $E(G)$ povezuje čvor iz skupa U sa čvorom iz skupa W.

Definicija 3.15. *Graf G je $k-$partitni graf ukoliko se njegov skup čvorova $V(G)$ može predstaviti kao disjunktna unija k nezavisnih skupova koje nazivamo particijama grafa G. Graf G se naziva i multipartitni graf.*

Primijetimo da je bipartitni graf specijalan slučaj $k-$partitnog grafa za $k = 2$.

Definicija 3.16. *Kompletan $k-$partitni graf je $k-$partitni graf u kome su svaka dva čvora koja pripadaju različitim particijama povezana granom. Kompletan $k-$partitni graf G sa particijama veličina $n_1, n_2, ..., n_k$ označava se sa $K_{n_1,...,n_k}$.*



(a) (b)

Slika 2: Bojenje grafa K_3 sa dvije boje.

4 Multi-skup susjed-razlikujuće 3-bojenje grana grafa

Prilikom bojenja grafova, za oznake boja ćemo koristiti prirodne brojeve, odnosno, ukoliko graf bojimo sa k boja, tada boje uzimaju vrijednosti iz skupa $\{1, 2, ..., k\}$. Ukoliko postoji ispravno bojenje grafa sa k boja, kažemo da je graf $k-$obojiv.

Neka je G normalan prost graf i neka je $c : E(G) \to \{1, 2, ..., k\}$ bojenje grana grafa, pri čemu ne pretpostavljamo ispravnost ovog bojenja, odnosno, susjedne grane mogu biti obojene istom bojom. Za c kažemo da je $k-$bojenje grafa G. Kôd čvora v, u oznaci $code(v)$, za svaki čvor $v \in V(G)$, je uređena $k-$torka $(c_1, c_2, ..., c_k)$, gdje je $c_i, i \in \{1, 2, ..., k\}$ broj grana incidentnih čvoru v koje su obojene bojom i.

Ukoliko svaka dva susjedna čvora grafa G imaju različite kodove boja, takvo bojenje nazivamo multi-skup susjed-razlikujuće bojenje grana grafa ili skraćeno *msde-bojenje.* Najmanji broj k za koji postoji ispravno $k-$msde bojenje grafa G nazivamo multi-skup susjed razlikujući indeks, ili skraćeno *msde-indeks* grafa G, i označavamo ga sa $\chi^e_m(G)$.

Jasno je da ukoliko graf sadrži izolovanu granu, tada ne postoji ispravno msde-bojenje tog grafa. Zbog toga se msde-bojenje razmatra samo za normalne grafove.

U nastavku je teorema koja daje gornju granicu za broj boja potrebnih za multi-skup susjed-razlikujuće bojenje (msde-indeks) proizvoljnog normalnog grafa. Teoremu je dokazao Bojan Vučković u radu [1].

Teorema 1. *Za svaki normalan graf G važi $\chi^e_m(G) \leq 3$.*

Primijetimo prvo da su tri boje u opštem slučaju neophodne za uspješno multi-skup susjed-razlikujuće bojenje proizvoljnog normalnog grafa. Naime, postoje grafovi koji se ne mogu obojiti sa manje od tri boje. Primjeri takvih grafova su $K_n, n \geq 3$ i $C_n, n \geq 3$ i $n \not\equiv 0$ $(mod\ 4)$. Sva moguća bojenja grana grafa K_3 sa dvije boje (označavamo ih sa 1 i 2), kao i odgovarajući kodovi čvorova prestavljeni su na slici 2. Međutim, ova bojenja nisu ispravna msde-bojenja, pa je očigledno da ne postoji ispravno msde-bojenje za ovaj graf sa manje od tri boje.

Primijetimo takođe da, ako je tvrđenje teoreme zadovoljeno za svaki povezan graf sa bar dvije grane, očigledno je zadovoljeno i za svaki normalan graf, jer je normalan graf ili povezan graf sa bar dvije grane, ili disjunktna unija dvije ili više komponenti povezanosti od kojih je svaka povezan graf sa bar dvije grane.

U nastavku je dat pseudo-kod algoritma za multi-skup susjed-razlikujuće bojenje uz prateća objašnjenja, dok je algoritam implementiran u programskom jeziku *Python* korišćenjem biblioteke *networkx* za rad sa grafovima.

U dokazu teoreme koristi se pomoćna lema koja je korišćena i prilikom implementacije. Slijedi tvrđenje i dokaz leme.

Lema 1. *Neka je G neki indukovani podgraf grafa H, pri čemu je G povezani bipartitni graf sa podjelom čvorova $\{V_1, V_2\}$. Dalje, neka je ϕ bojenje grana iz skupa $E(H)\backslash E(G)$ proizvoljnim bojama. Tada za proizvoljno $v \in V_1$ postoji proširenje bojenja ϕ na graf H koje svakoj od grana iz $E(G)$ dodjeljuje boju iz skupa $\{1, 2\}$ tako da svi čvorovi iz $V_1\backslash\{v\}$ imaju u H paran (neparan) broj incidentnih grana obojenih bojom 1, dok svi čvorovi iz V_2 imaju u H neparan(paran) broj incidentnih grana obojenih bojom 1.*

Dokaz: Dokazaćemo tvrđenje leme tako da svi čvorovi iz $V_1\backslash\{v\}$ imaju u H paran broj incidentnih grana obojenih bojom 1, dok svi čvorovi iz V_2 imaju u H neparan broj incidentnih grana obojenih bojom 1. U drugom slučaju dokaz je analogan.

Prvo, svim granama grafa G dodjeljujemo boju 1. Dok god postoje dva ili više čvorova u $V_1\backslash\{v\}$ koji imaju neparan broj incidentnih grana obojenih bojom 1, primjenjujemo sljedeći postupak. Neka su w_1 i w_2 dva takva čvora. Razmjenjujemo boje 1 i 2 svakoj od grana na putu između w_1 i w_2 u grafu G. Na ovaj način se mijenja parnost broja grana incidentnih sa w_1 i w_2 obojenih bojom 1, dok parnost broja grana obojenih bojom 1 incidentnih sa bilo kojim čvorom iz G različitim od w_1 i w_2 ostaje ista. Na kraju ovog postupka, postoji najviše jedan čvor u $V_1\backslash\{v\}$ koji ima neparan broj incidentnih grana obojenih bojom 1. Dalje, dok god postoje dva ili više čvorova u V_2 koji imaju paran broj incidentnih grana obojenih bojom 1, primenjujemo sljedeći postupak sličan prethodnom. Neka su u_1 i u_2 dva takva čvora. Razmjenjujemo boje 1 i 2 svake od grana na putu između u_1 i u_2 u grafu G. Na kraju ovog postupka, postoji najviše jedan čvor u V_2 koji ima paran broj incidentnih grana obojenih bojom 1. Ukoliko postoje čvorovi $w \in V_1\backslash\{v\}$ sa neparnim brojem incidentnih grana obojenih bojom 1, i $u \in V_2$ sa parnim brojem incidentnih grana obojenih bojom 1, tada razmjenjujemo boje 1 i 2 svakoj od grana na putu između w i u u G. Dakle, sada postoji najviše jedan čvor y iz $(V_1\backslash\{v\}) \cup V_2$ sa neželjenom parnošću broja incidentnih grana obojenih bojom 1. Ukoliko takav čvor postoji, označimo ga sa y. Razmjenjujemo boje 1 i 2 svakoj od grana na putu između y i v u G, čime dobijamo željeno bojenje grana grafa. □

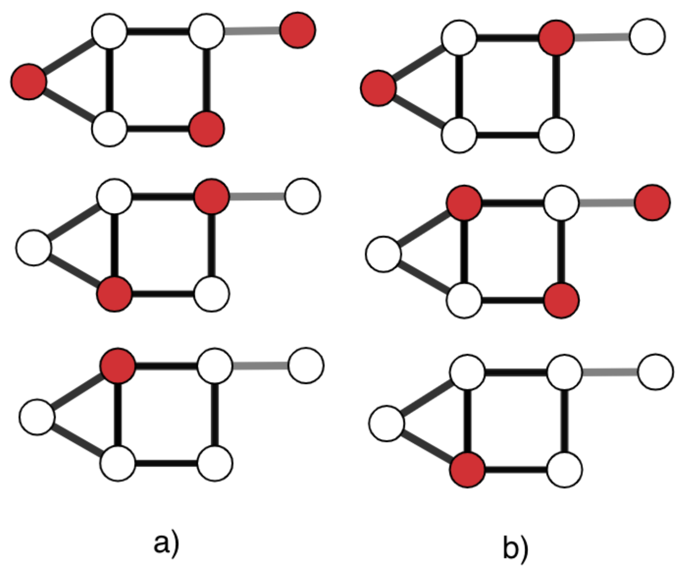
U pseudokodu algoritma za bojenje pretpostavićemo da postoji funkcija koja implementira bojenje u skladu sa lemom, koja se zove `bojenje_po_lemi` i koja se poziva sa:

```
bojenje_po_lemi(H, G, v, V1, V2, boja1, boja2, prvi_paran),
```

gdje je G indukovani podgraf grafa H, koji je povezani bipartitni graf sa podjelom cvorova {`V1`, `V2`}, a `v` je „istaknuti“ cvor iz skupa `V1`, `boja1` i `boja2` su boje 1 i 2 iz leme, dok posljednji argument može imati vrijednost iz skupa {`True`, `False`} i ukoliko je vrijednost `True` graf G se boji tako da svi čvorovi iz `V1\{v}` imaju paran broj grana obojenih bojom `boja1`, a svi čvorovi iz `V2` imaju neparan broj incidentnih grana obojenih bojom `boja1`. Ukoliko argument `prvi_paran` ima vrijednost `False` graf G se boji tako da svi čvorovi iz `V1\{v}` imaju neparan broj grana obojenih bojom `boja1`, a svi čvorovi iz `V2` imaju paran broj incidentnih grana obojenih bojom `boja1`.

4.1 Algoritam za multi-skup susjed-razlikujuće 3-bojenje grafova

Pretpostavimo da svaki čvor grafa ima polje `kod` koje je tročlani niz i predstavlja njegov kôd inicijalizovan nulama, kao i da svaka grana grafa ima polje `boja` koje uzima vrijednost iz skupa



Slika 3: Dva različita razlaganja čvorova grafa na nezavisne dominirajuće skupove. Čvorovi predstavljeni crvenom bojom su u skupovima $V_i, i = 1, 2, 3$. Na vrhu je skup V_1, u sredini je V_2, a na dnu je skup V_3.

$\{1, 2, 3\}$, i takođe je inicijalizovano nulom. Definicija funkcije počinje ključnom riječju `def`, a algoritam je opisan nizom funkcija. Takođe, komentari su linijski i počinju znakom `#`, a odgovarajućim nazubljivanjem koda određeni su početak i kraj svakog bloka. Indeksiranje elemenata kolekcije počinje od jedinice.

Prvi korak algoritma je podjela skupa čvorova grafa `G` na `n` disjunktnih skupova `V[1], V[2], ..., V[n]` tako da je `V[1]` nezavisni dominirajući skup grafa `G`, `V[i]` je nezavisni dominirajući skup grafa `G[V\(V[1] ∪ ... ∪ V[i-1])]`, za sve `1 < i < n`, dok je `V[n]` nezavisan skup. Primijetimo da, za svako `v` iz skupa `V[i]`, čvor `v` ima bar po jednog susjeda u svakom od skupova `V[j]` za `1 <= j < i`, jer su svi `V[j]` dominirajući skupovi, a nema nijednog susjeda u `V[i]`, jer je `V[i]` nezavisan skup. Ovakvo razlaganje skupa čvorova grafa nije jedinstveno, kao što se može vidjeti u primjeru na slici 3.

Neka je `v` čvor iz skupa `V[i]` i neka je `v.kod = [b1, b2, b3]`. Algoritam boji grane grafa `G` tako da su, osim u nekoliko specijalnih slučajeva, zadovoljena sljedeća svojstva.

1. Ukoliko je `i = 2*k+3` za `k >= 1`, tada `b1 >= 1`, `b2 = k` i `b3` je neparan broj veći od 1.
2. Ukoliko je `i = 2*k+2` za `k >= 1`, tada `b1 = k`, `b2 >= 1` i `b3` je paran broj veći od 0.
3. Ukoliko je `i = 3`, tada `b1 = 0`, `b2 >= 1` i `b3 >= 1`.
4. Ukoliko je `i = 2` i `v` ima susjeda u `V[j]` za `j > 2`, tada `b1 + b2 >= 2` i `b3 = 0`.
5. Ukoliko je `i = 1` i `v` ima susjeda u `V[j]` za `j > 2`, tada `b2 = 0` i `b3 >= 1`.

6. Ukoliko je `v` ∈ `V[1]` ili `v` ∈ `V[2]` i `v` nema susjeda van `V[1]` ∪ `V[2]` tada samo gledamo da `v` ima drugačiji kôd boja od svih svojih susjeda.

Algoritam bojenja na početku dodjeljuje boju 3 svim granama koje povezuju čvorove iz skupa `V[1]` sa čvorovima iz skupova `V[3], V[4], ..., V[n]`. Nakon toga bojenje ide unazad počev od skupa `V[n]` pa do skupa `V[2]`. Naredna funkcija `msde3_bojenje_grafa` prikazuje opisani algoritam za multi-skup susjed-razlikujuće 3-bojenje grafa G. Pretpostavljamo da je na raspolaganju funkcija koja vrši particionisanje skupa `V(G)` na nezavisne dominirajuće skupove. Za čvor `v` grafa `G` sa `G[v]` označava se lista susjeda čvora `v` u `G`. Takođe, koriste se uobičajene oznake za aritmetičke, relacione i logičke operatore. Koristi se i operator `in` koji provjerava pripadnost elementa listi ili skupu i vraća vrijednost `True` ili `False`, kao i operator `duzina` koji vraća broj elemenata liste ili skupa.

```
# opis: multi-skup susjed-razlikujuce 3-bojenje grafa G
# ulaz: graf G
def msde3_bojenje_grafa(G):
    # podjela skupa cvorova grafa na nezavisne dominirajuce skupove
    V = particionisanje_na_nezavisne_dominirajuce_skupove(G)
    n = V.duzina
    %# bojenje grana od V[1] ka V[3],...,V[n] bojom 3
    for v in V[1]:
        for u in G[v] && !(u in V[2]):
            oboji(G, v, u, 3)
    for i in [n, n-1, ..., 6]:
        if i%2 == 0:
            oboji_parno_Vi(G, i, V)
        else:
            oboji_neparno_Vi(G, i, V)
    if n >= 5:
        oboji_V5(G, V)
    if n >= 4:
        oboji_V4(G, V)
    if n >= 3:
        oboji_V3(G, V)
    if n >= 2:
        oboji_V2(G, V)
```

U nastavku je opisan postupak bojenja svakog od skupova `V[i]`, koje podrazumijeva dodjelu boja granama koje su incidentne čvorovima iz skupa `V[i]`. Bojenje grana incidentih čvorovima iz skupa `V[i]` za `1 <= i <= n` zavisi od vrijednosti broja `i`. Pošto su prethodno obojene sve grane incidentne čvorovima iz skupova V[k] za svako i < k <= n, prilikom bojenja skupa V[i] dodjeljuju se boje granama koje povezuju čvorove iz skupa V[i] sa čvorovima iz skupova V[j] za svako 1 <= j < i. Prilikom bojenja skupa V[i], za svaki od čvorova v u tom skupu dodjeljuje se boja granama incidentnim tom čvoru.

Funkcija oboji(G, v1, v2, boja) boji granu između čvorova v1 i v2 u grafu G zadatom bojom i ažurira kodove čvorova v1 i v2. Operator & označava presjek skupova.

Funkcija oboji_parno_Vi boji skup V[i] za parno i veće od 5. Ako je v proizvoljan čvor iz skupa V[i] za parno i veće od 5, onda za v.kod = [a1, a2, a3] važi sljedeće. Pošto v ima bar jednog susjeda u skupu V[1] važi da je a3 >= 1. Za i = n važi da je a1 = a2 = 0, dok za i < n važi da je a1 = 0 i a2 >= 0. Za svaki čvor v iz skupa V[i] boje se grane koje povezuju v sa čvorovima iz skupova V[j] za 1 <= j < i. Ako je j paran broj veći od 3 sve grane od v ka skupu V[j] boje se bojom 2. Ako je j neparan broj veći od 3 jedna od grana od v ka V[j] boji se bojom 1, a sve ostale grane od v ka čvorovima iz V[j] boje se bojom 3. Preostalo je obojiti grane od čvora v ka čvorovima iz skupova V[2] i V[3]. Neka je u proizvoljan susjed čvora v u skupu V[2] i w proizvoljan susjed čvora v u skupu V[3]. Svim granama koje povezuju čvor v sa čvorovima iz skupa V[2]\{u} dodjeljuje se boja 2, a svim granama koje povezuju čvor v sa čvorovima iz skupa V[3]\{w} dodjeljuje se boja 3. Dodatno, grani vu dodjeljuje se boja 1. Ako je i = 2*k+2, gdje je k >= 2, tada je v.kod = [k, b2, b3] i važi da je b2 >= 1 i b3 >= 1. Preostalo je obojiti granu vw. Ukoliko je b3 paran broj granu vw bojimo bojom 2, a ukoliko je b3 neparan broj granu vw bojimo bojom 3. Konačno, imamo da je v.kod = [k, c2, 2*c3], gdje je c2 >= 1 i c3 >= 1.

```
# opis: funkcija boji skup V[i] za parno i>5
# ulaz: graf G, paran broj i veci od 5, niz nezavisnih dominirajucih skupova V
def oboji_parno_Vi(G, i, V):
    for v in V[i]:
        # bojimo grane od v ka V[j] za 4<=j<i
        for j in [4, ..., i-1]:
            if j%2 == 1:
                # k oznacava da li je prva grana koja povezuje v sa cvorom iz
                # V[j] obojena bojom 1, sve ostale grane boje se bojom 3
                k = False
                for u in G[v]:
                    if u in V[j]:
                        if k && u in V[j]:
                            oboji(G, v, u, 1)
                            k = True
                        else if u in V[j]:
                            oboji(G, v, u, 3)
            else:
                # sve grane izmedju v i V[j] se boje bojom 2
                for u in G[v]:
```

```
                if u in V[j]:
                    oboji(G, v, u, 2)
    # u je proizvoljan susjed cvora v u skupu V[2]
    u = (G[v] & V[2])[1]
    oboji(G, v, u, 1)
    # w je proizvoljan susjed cvora v u skupu V[3]
    w = (G[v] & V[3])[1]
    for n in G[v]:
        if n in V[2] && n != u:
            oboji(G, v, n, 2)
        if n in V[3] && n != w:
            oboji(G, v, n, 3)
    # ako v ima paran broj incidentnih grana obojenih bojom 3, vw se boji
    # bojom 2
    if v.kod[3]%2 == 0:
        oboji(G, v, w, 2)
    # inace se boji bojom 3
    else:
        oboji(G, v, w, 3)
```

Naredna funkcija je `oboji_neparno_Vi` i ona boji skup `V[i]` za neparno `i` veće od 5. Bojenje skupa `V[i]` za neparno `i` veće od 5 vrši se na sljedeći način. Ako je `v` proizvoljan čvor iz skupa `V[i]` za neparno `i` veće od 5, onda za `v.kod = [a1, a2, a3]` važi sljedeće. Pošto `v` ima bar jednog susjeda u skupu `V[1]` važi da je `a3 >= 1`. Za `i = n` važi da je `a1 = a2 = 0`, dok za `i < n` važi da je `a1 >= 0` i `a2 = 0`. Za svaki čvor `v` iz skupa `V[i]` boje se grane koje povezuju `v` sa čvorovima iz skupova `V[j]` za `1 <= j < i`. Ako je `j` neparan broj veći od 4 sve grane od `v` ka skupu `V[j]` boje se bojom 1. Ako je `j` paran broj veći od 4 jedna od grana od v ka `V[j]` boji se bojom 2, a sve ostale grane od `v` ka čvorovima iz `V[j]` boje se bojom 3. Grane koje povezuju čvor `v` sa čvorovima iz skupa `V[4]` boje se bojom 3. Preostalo je obojiti grane od čvora `v` ka čvorovima iz skupova `V[2]` i `V[3]`. Neka je `u` proizvoljan susjed čvora `v` u skupu `V[2]` i `w` proizvoljan susjed čvora `v` u skupu `V[3]`. Svim granama koje povezuju čvor `v` sa čvorovima iz skupa `V[2]\{u}` dodjeljuje se boja 1, a svim granama koje povezuju čvor `v` sa čvorovima iz skupa `V[3]\{w}` dodjeljuje se boja 3. Ako je `i = 2*k+3`, gdje je `k >= 2`, tada je `v.kod = [b1, k-1, b3]` i važi da je `b1 >= 1` i `b3 >= 2`. Preostalo je obojiti grane `vu` i `vw`. Ukoliko je `b3` paran broj granu `vu` bojimo bojom 2, a granu `vw` bojimo bojom 3. Ukoliko je `b3` neparan broj granu `vu` bojimo bojom 1, a granu `vw` bojimo bojom 2. Konačno, imamo da je `v.kod = [c1, k, 2*c3+1]`, gdje je `c1 >= 1` i `c3 >= 1`.

Funkcije `oboji_neparno_Vi` i `oboji_parno_Vi` na sličan način tretiraju čvorove iz skupova `V[j]` ukoliko je `j` iste parnosti kao `i`, uz razliku da boje prilikom bojenja grana ka skupovima `V[j]` u ovim funkcijama variraju između 1 i 2. Takođe, u funkciji `oboji_parno_Vi` se grane ka skupu `V[2]` boje sa 2, dok se u funkciji `oboji_neparno_Vi` grane ka skupu `V[2]` boje sa 1. Grane ka skupu `V[3]` se u oba slučaja boje sa 3. Bojenje grane `vw` zavisi od parnosti broja grana incidentnih sa `v` obojenih bojom 3.

```
# opis: funkcija boji skup V[i] za neparno i>5
# ulaz: graf G, neparan broj i veci od 5, niz nezavisnih dominirajucih skupova V
def oboji_neparno_Vi(G, i, V):
    for v in V[i]:
        # bojimo grane od v ka V[j] za 5<=j<i
        for j in [5, 6, ..., i-1]:
            if j%2 == 1:
                # sve grane izmedju v i V[j] se boje bojom 1
                for u in G[v]:
                    if u in V[j]:
                        oboji(G, v, u, 1)
            else:
                # k oznacava da li je prva grana koja povezuje v sa cvorom iz
                # V[j] obojena bojom 2, sve ostale grane boje se bojom 3
                k = False
                for u in G[v]:
                    if u in V[j]:
                        if !k && u in V[j]:
                            oboji(G, v, u, 2)
                            k = True
                        else if u in V[j]:
                            oboji(G, v, oboji, 3)
        # u je proizvoljan susjed cvora v u skupu V[2]
        u = (G[v] & V[2])[1]
        # w je proizvoljan susjed cvora v u skupu V[3]
        w = (G[v] & V[3])[1]
        # boje se grane od v ka V[4], V[3], V[2]
        for n in G[v]:
            # svaka grana koja povezuje v sa V[4] se boji bojom 3
            if n in V[4]:
                oboji(G, v, n, 3)
            # svaka grana koja povezuje v sa V[3] osim vw se boji bojom 3
            if n in V[3] && n != w:
                oboji(G, v, n, 3)
            # svaka grana koja povezuje v sa V[2] osim vu se boji bojom 1
            if n in V[2] and n != u:
                oboji(G, v, n, 1)
        # jedine neobojene grane incidentene sa v su vu i vw, one se boje u
```

```
        # zavisnosti od parnosti broja grana incidentnih sa v obojenih bojom 3
        if v.kod[3]%2 == 0:
            oboji(G, v, u, 2)
            oboji(G, v, w, 3)
        else:
            oboji(G, v, u, 1)
            oboji(G, v, w, 2)
```

Slijede funkcije `oboji_V5`, `oboji_V4` i `oboji_V3`, kojima se redom boje grane incidentne čvorovima iz skupova `V[5]`, `V[4]` i `V[3]`.

Bojenje skupa `V[5]` vrši se na sljedeći način. Ako je `v` proizvoljan čvor iz skupa `V[5]` onda za `v.kod = [a1, a2, a3]` važi: `a1 >= 0`, `a2 = 0` i `a3 >= 1`. Potrebno je obojiti grane koje povezuju čvor `v` sa čvorovima iz skupova `V[4]`, `V[3]` i `V[2]`. Neka je `u` proizvoljan susjed čvora `v` u skupu `V[2]`, `w` proizvoljan susjed čvora `v` u skupu `V[3]` i `y` proizvoljan susjed čvora `v` u skupu `V[4]`. Za svaki čvor `x` iz skupa `V[4]`\\`{y}` koji je susjed čvora `v` radimo sljedeće. Ukoliko `x` ima incidentnu granu obojenu bojom 1, granu `vx` bojimo bojom 3, a inače granu `vx` bojimo bojom 1. Dalje, sve grane koje povezuju čvor `v` sa čvorovima iz skupa `V[2]`\\`{u}` bojimo bojom 1, dok sve grane između čvorova iz skupa `V[3]`\\`{w}` i čvora `v` bojimo bojom 3. Dakle, važi da je `v.kod = [b1, 0, b3]`, gdje je `b1 >= 0` i `b3 >= 1` i preostalo je da se oboje grane `vy`, `vw` i `vu`.

Ukoliko čvor `y` nema incidentnu granu koja je obojena bojom 1, grana `vy` boji se bojom 1, a bojenje grana `vu` i `vw` zavisi od vrijednosti broja `b3`.

(i) `b3` je paran broj. Grana `vu` se boji bojom 2, a grana `vw` se boji bojom 3.

(ii) `b3 = 1`. Grane `vu` i `vw` se boje bojom 2.

(iii) `b3` je neparan broj veći od 3. Grana `vu` se boji bojom 1, a grana `vw` se boji bojom 2.

Dakle, imamo da je `v.kod = [c1, 1, 2*c3+1]`, gdje je `c1 >= 1` i `c3 >= 1` ili je `v.kod = [c1, 2, 1]`, gdje je `c1 >= 1`. Pošto za svako `u` $\in$ `V[j]` za `j > 5` imamo da je `u.kod = [x1, x2, x3]` gdje je `x2 >= 2` i `x3 >= 1` ili je `x3` paran broj, to se kôd čvora `v` razlikuje od kodova svih njegovih susjeda.

Ukoliko čvor `y` ima incidentnu granu koja je obojena bojom 1, bojenje grana `vu`, `vw` i `vy` zavisi od vrijednosti brojeva `b1` i `b3`.

(i) `b3` je paran broj. Grana `vu` se boji bojom 1, grana `vw` se boji bojom 2, a grana `vy` se boji bojom 3.

(ii) `b3` je neparan broj i `b1 = 0`. Ovo znači da čvor `v` nema incidentnih čvorova u skupovima `V[j]` za neparno `j` veće od 5, jer bi u suprotnom postojala grana incidentna sa `v` obojena bojom 1. Grana `vu` se boji bojom 1, a grane `vw` i `vy` se boje bojom 2.

(iii) `b3` je neparan broj i `b1 >= 1`. Grana `vu` se boji bojom 2, a grane `vw` i `vy` se boje bojom 3.

Dakle, u drugom slučaju imamo da je `v.kod = [1, 2, 2*c3+1]`, gdje je `c3 >= 0` i čvor `v` nema susjeda ni u jednom od skupova `V[j]` za neparno `j` veće od 5, pa je njegov kôd različit od kodova njemu incidentnih čvorova, jer oni imaju paran broj incidentnih grana obojenih bojom 3. U ostalim slučajevima imamo da je `v.kod = [c1, 1, 2*c3+1]`, gdje je `c1 >= 1` i `c3 >= 1`. Pošto za svako `u` ∈ `V[j]` za `j > 5` imamo da je `u.kod = [x1, x2, x3]` gdje je `x2 >= 2` i `x3 >= 1` ili je `x3` paran broj, to se kôd čvora `v` razlikuje od kodova svih njegovih susjeda.

```
# opis: funkcija boji skup V[5]
# ulaz: graf G i niz nezavisnih dominirajucih skupova V
def oboji_V5(G, V):
    for v in V[5]:
        # u, w and y su 3 proizvoljna susjeda cvora v u skupovima
        # V[2], V[3] i V[4] redom
        u = (G[v] & V[2])[1]
        w = (G[v] & V[3])[1]
        y = (G[v] & V[4])[1]
        for n in G[v]:
            # ako je n u V[4]\{y}
            if n in V[4] && n != y:
                # ako n ima incidentnu granu obojenu bojom 1, vn se boji bojom 3
                if n.kod[1] >= 1:
                    oboji(G, v, n, 3)
                # inace se vn boji bojom 1
                else:
                    oboji(G, v, n, 1)
            # ako je n u V[3]\{w}, vn se boji bojom 3
            if n in V[3] && n != w:
                oboji(G, v, n, 3)
            # ako je n u V[2]\{u}, vn se boji bojom 1
            if n in V[2] && n != u:
                oboji(G, v, n, 1)
        # jedine neobojene grane incidentne sa v su vu, vw i vy
        b3 = v.kod[3]
        # bojenje grane vy zavisi od parnosti broja y.kod[1]
        if y.kod[1] == 0:
            oboji(G, v, y, 1)
            # bojenje vu i vw zavisi od vrijednosti b3
            if b3%2 == 0:
                oboji(G, v, u, 2)
                oboji(G, v, w, 3)
            else if b3 == 1:
                oboji(G, v, u, 2)
```

```
                oboji(G, v, w, 2)
            else:
                oboji(G, v, u, 1)
                oboji(G, v, w, 2)
        else:
            b1 = v.kod[1]
            # bojenje vy, vu i vw zavisi od vrijednosti b1 i b3
            if b3%2 == 0:
                oboji(G, v, u, 1)
                oboji(G, v, w, 2)
                oboji(G, v, y, 3)
            else if b3%2 == 1 && b1 == 0:
                oboji(G, v, u, 1)
                oboji(G, v, w, 2)
                oboji(G, v, y, 2)
            else:
                oboji(G, v, u, 2)
                oboji(G, v, w, 3)
                oboji(G, v, y, 3)
```

Bojenje skupa `V[4]` vrši se na sljedeći način. Ako je `v` proizvoljan čvor iz skupa `V[4]` onda za `v.kod = [a1, a2, a3]` važi: `a1 <= 1`, `a2 >= 0` i `a3 >= 1`. Potrebno je obojiti grane koje povezuju čvor `v` sa čvorovima iz skupova `V[3]` i `V[2]`. Neka je `u` proizvoljan susjed čvora `v` u skupu `V[2]`, a `w` proizvoljan susjed čvora `v` u skupu `V[3]`. Sve grane koje povezuju čvor `v` sa čvorovima iz skupa `V[2]\{u}` bojimo bojom 2, dok sve grane između čvorova iz skupa `V[3]\{w}` i čvora `v` bojimo bojom 3. Potrebno je još obojiti grane `vu` i `vw`. Neka je sada `v.kod = [b1, b2, b3]`. Imamo da je `b1 <= 1`, `b2 >= 0` i `b3 >= 1`. U zavisnosti od vrijednosti brojeva `b1` i `b2` razlikujemo tri slučaja.

(i) `b1 = 0` i `b2 = 0`. Zbog načina kako su bojene grane između skupova `V[4]` i `V[5]` zaključujemo da čvor `v` nema susjeda u skupu `V[5]`. Grana `vu` se boji bojom 1, a grana `vw` se boji bojom 2. Sada su obojene sve grane incidentne čvoru `v` i važi da je `v.kod = [1, 1, b3]`. Svaki čvor iz skupa `V[j]` za `j >= 6` ima bar dvije incidentne grane obojene bojom 1 (ako je `j` paran broj) ili dvije incidentne grane obojene bojom 2 (ako je `j` neparan broj). Pošto čvor `v` nema susjeda u skupu `V[5]` to čvor `v` ima različit kôd od svih čvorova u skupovima `V[j]` za `j > 4`.

(ii) `b1 = 0` i `b2 >= 1`. Ukoliko je `b3` paran broj, grana `vu` se boji bojom 1, a grana `vw` se boji bojom 2. Ukoliko je `b3` neparan broj, grana `vu` se boji bojom 1, a grana `vw` se boji bojom 3. U oba slučaja imamo da je `v.kod = [1, c2, 2*c3]` gdje je `c2 >= 1` i `c3 >= 1`. Pošto je svaki čvor iz skupa `V[j]` za parno `j > 5` incidentan sa bar dvije grane obojene bojom 1, a svaki čvor iz skupa `V[j]` za neparno `j >= 5` incidentan sa neparnim brojem grana obojenih bojom 3, čvor `v` ima različit kôd od svih čvorova u skupovima `V[j]` za `j > 4`.

(iii) `b3 = 1`. Grana `vu` se boji bojom 2. Ukoliko je `b3` paran broj, grana `vw` se boji bojom 2, a ukoliko je `b3` neparan broj, grana `vw` se boji bojom 3. Sada su obojene sve grane incidentne čvoru `v` i imamo da je `v.kod = [1, c2, 2*c3]` gdje je `c2 >= 1` i `c3 >= 1`. Kao i u prethodnom slučaju, zaključujemo da čvor `v` ima različit kôd od svih čvorova u skupovima `V[j]` za `j > 4`.

```
# opis: funkcija boji skup V[4]
# ulaz: graf G i niz nezavisnih dominirajucih skupova V
def oboji_V4(G, V):
    for v in V[4]:
        # u i w su proizvoljni susjedi cvora v u skupovima V[2] i V[3] redom
        u = (G[v] & V[2])[1]
        w = (G[v] & V[3]][1]
        for n in G[v]:
            # ako je n u skupu V[3]\{w} grana vn se boji bojom 3
            if n in V[3] && n != w:
                oboji(G, v, n, 3)
            # ako je n u skupu V[2]\{u} grana vn se boji bojom 2
            if n in V[2] && n != u:
                oboji(G, v, n, 2)
        # jedine neobojene grane incidentne sa v su vu i vw
        # kod cvora v je [b1, b2, b3]
        b1 = v.kod[1]
        b2 = v.kod[2]
        b3 = v.kod[3]
        if b1 == 0 && b2 == 0:
            oboji(G, v, u, 1)
            oboji(G, v, w, 2)
        else if b1 == 0 && b2 >= 1:
            if b3%2 == 0:
                oboji(G, v, u, 1)
                oboji(G, v, w, 2)
            else:
                oboji(G, v, u, 1)
                oboji(G, v, w, 3)
        else:
            oboji(G, v, u, 2)
            if b3%2 == 0:
                oboji(G, v, w, 2)
            else:
                oboji(G, v, w, 3)
```

Bojenje skupa `V[3]` vrši se na sljedeći način. Ako je `v` proizvoljan čvor iz skupa `V[3]` onda za `v.kod = [a1, a2, a3]` važi: `a1 = 0`, `a2 >= 0` i `a3 >= 1`. Potrebno je obojiti grane koje povezuju čvor `v` sa čvorovima iz skupa `V[2]`. Svim granama koje povezuju čvor `v` sa čvorovima iz skupa `V[2]` dodjeljuje se boja 2. Sada važi da je `v.kod = [0, b2, b3]` gdje je `b2 >= 1` i `b3 >= 1`. Pošto su svi čvorovi iz skupova `V[j]` za `j > 3` incidentni sa bar jednom granom obojenom bojom 1, zaključujemo da čvor `v` ima različit kôd od svih čvorova u skupovima `V[j]` za `j > 3`.

```
# opis: funkcija boji skup V[3]
# ulaz: graf G i niz nezavisnih dominirajucih skupova V
def oboji_V3(G, V):
    for v in V[3]:
        # sve grane od cvora v ka skupu V[2] se boje bojom 2
        for n in G[v]:
            if n in V[2]:
                oboji(G, v, n, 2)
```

U nastavku je funkcija `oboji_V2` kojom se boji skup `V[2]`. Za pseudokod ove funkcije pretpostavićemo da je na raspolaganju funkcija `promijeni_boju(G, v, u, boja1, boja2)` koja u grafu `G` mijenja boju grane između čvorova `v` i `u` iz `boja1` u `boja2` uz ažuriranje kodova čvorova `v` i `u`. Operator `|` označava skupovnu uniju, dok operator `-` označava skupovnu razliku. Pretpostavimo da funkcija `skup()` pravi prazan skup, kao i da je na raspolaganju funkcija `dodaj` koja dodaje element koji se proslijedi kao argument u skup. Pored toga, koristi se funkcija `ima_granu` koja provjerava da li između zadatih čvorova u grafu postoji grana, funkcija `podgraf` koja vraća indukovani podgraf grafa za proslijeđeni skup čvorova, kao i funkcija `povezane_komponente` koja vraća skup povezanih komponenti grafa. Ukoliko je graf povezan, povratna vrijednost je taj graf. Preciznije, ova funkcija vraća skupove čvorova koji generišu povezane komponente grafa. Koristi se naredna pomoćna funkcija `sve_obojene_sa_1`, koja provjerava da li su sve grane grafa `G` od zadatog čvora `v` ka čvorovima koji nisu u skupu `V` obojene bojom `1`.

```
# opis: funkcija koja provjerava da li su sve grane iz cvora v ka cvorovima
# koji nisu u skupu V obojene bojom 1
# ulaz: graf G, cvor v iz G, skup V koji je podskup skupa cvorova grafa
# izlaz: True ako su sve trazene grane obojene bojom 1, u suprotnom False
def sve_obojene_sa_1(G, v, V):
    # za sve susjede cvora v
    for s in G[v]:
        # koji nisu u skupu V, i takvi su da je grana sv obojena
        if !(s in V) && G[s][v].boja != 0:
            # ako je boja grane sv razlicita od 1, vraca se False
```

```
            if G[s][v].boja != 1:
                    return False
    # inace se vraca True
    return True
```

Bojenje skupa `V[2]` vrši se na sljedeći način. Za proizvoljan čvor `v` iz skupa `V[2]`, potrebno je obojiti grane koje povezuju čvor `v` sa čvorovima iz skupa `V[1]`. Primijetimo da za svako `u` ∈ `V[j]` za `j >= 4` čvor `u` ima tačno jednu incidentnu granu obojenu bojom 3 samo ako važi da je `u.kod = [1, 1, 1]` i `u` ∈ `V[4]` ili `u.kod = [c1, 2, 1]`, gdje je `c1 >= 1` i `u` ∈ `V[5]`.

Neka je `W1` skup svih čvorova iz skupa `V[1]` koji imaju bar jednog susjeda u nekom od skupova `V[l]` za `l > 2`. Dalje, neka je `U1` skup svih čvorova iz skupa `V[2]` koji imaju bar jednog susjeda u nekom od skupova `V[l]` za `l > 2`. Uvedimo i oznake `W2 = V[1]\W1` i `U2 = V[2]\U1`.

U funkciji se prvo dodjeljuje boja 1 svim granama koje povezuju čvorove iz skupa `W1` sa čvorovima iz skupa `V[2]`. Kasnije će se boja nekih od ovih grana mijenjati u 3 nakon čega čvorovi iz skupa `W1` i dalje neće imati incidentnih grana obojenih bojom 2. Sada dodjeljujemo boje granama koje povezuju čvorove iz skupova `W2` i `V[2]`. Neka je `H = G[W2` ∪ `V[2]]` i neka su `C[j]`, `1 <= j <= k` povezane komponente grafa `H`. Dakle, `H` je indukovani podraf grafa `G` generisan skupom čvorova `W2` ∪ `V[2]`. Grane grafa `C[j]`, za svako `1 <= j <= k` bojimo tako da se kodovi susjednih čvorova u skupovima `C[j]` razlikuju, dok će za svako `u` ∈ `U1` takvo da je `u.kod = [c1, c2, c3]` biti zadovoljen jedan od sljedećih uslova:

(i) `c3 = 0`,

(ii) `c1 >= 2`, `c2` je neparan broj i `c3 = 1`,

(iii) `c1 >= 1`, `c2 = 0`, `c3 = 1` i čvor `u` ima tačno jednog susjeda u skupu `W1` koji ima bar dvije incidentne grane obojene bojom 3,

(iv) `c1 >= 1`, `c2 = 0`, `c3 >= 1` i čvor `u` nema susjeda u skupu `W1`.

Dakle, čvor `u` ∈ `U1` će imati različit kôd od svih svojih susjeda iz skupa `W1`, kao i iz skupa `V[l]` za `l > 2`.

Sada proizvoljnim redosljedom bojimo svaku od povezanih komponenti `C[j]`, `1 <= j <= k` grafa `H`. Neka je `L = C[j]`. U zavisnosti od toga da li `V(L)` sadrži čvor iz skupa `U2` razlikujemo dva slučaja.

1. Skup `V(L)` sadrži čvor `u` iz skupa `U2`. Dalje se grananje vrši u zavisnosti od toga da li čvor `u` ima susjeda u skupu `W1`.

   (i) Čvor `u` ima susjeda `w` ∈ `W1`. Prema lemi 1 bojimo grane grafa `L` bojama 1 i 2 tako da svi čvorovi iz skupa `V(L)` ∪ `(V[2]\{u}` imaju paran broj incidentnih grana obojenih bojom 2, a svi čvorovi iz skupa `V(L)` ∪ `W2` imaju neparan broj incidentnih grana obojenih bojom 2. Ukoliko čvor `u` ima paran broj incidentih grana obojenih bojom 2 bojenje je završeno, pošto u tom slučaju svaki od čvorova grafa `L` ima različit kôd boja od svih svojih susjeda. Ukoliko čvor `u` ima neparan broj incidentnih grana obojenih bojom 2, mijenjamo boju grane `uw` iz 1 u 3. Sada nijedan susjed čvora `u` u skupu `W1` nema nijednu incidentnu granu obojenu bojom 2 i nijedan susjed čvora `u` u skupu `W2` nema nijednu incidentnu granu obojenu bojom 3. Kako čvor `u` ima bar po jednu incidentnu granu bojenu bojama 2 i 3, zaključujemo da čvor `u` ima različit kôd boja od svih svojih susjeda, pa i svaki čvor grafa L ima različit kôd boja od svih svojih susjeda.

(ii) Čvor `u` nema susjednih čvorova u skupu `W1`. Prema lemi 1 bojimo grane grafa `L` bojama 1 i 2 tako da svi čvorovi iz skupa `V(L)` $\cup$ `(V[2]`$\setminus$`{u}` imaju neparan broj incidentnih grana obojenih bojom 2, a svi čvorovi iz skupa `V(L)` $\cup$ `W2` imaju paran broj incidentnih grana obojenih bojom 2. Ukoliko čvor `u` ima paran broj incidentih grana obojenih bojom 2 bojenje je završeno, pošto u tom slučaju svaki od čvorova grafa `L` ima različit kôd boja od svih svojih susjeda. Ukoliko čvor `u` ima paran broj incidentnih grana obojenih bojom 2, mijenjamo u 3 boju svake od grana incidentnih sa čvorom `u`. Ukoliko čvor `u` ima samo jednog susjeda `w`, tada, pošto je G povezan graf za bar dvije grane, znamo da čvor `w` osim čvora `u` ima bar još jednog susjeda. Zbog toga, čvorovi `w` i `u` imaju različite kôdove boja. Ukoliko čvor `u` ima više od jednog susjeda, i u ovom slučaju čvor `u` ima različit kôd boja od svih svojih susjeda, pošto nijedan od njih nema više od jedne incidentne grane obojene bojom 3.

Sada nijedan susjed čvora `u` u skupu `W1` nema nijednu incidentnu granu obojenu bojom 2 i nijedan susjed čvora `u` u skupu `W2` nema nijednu incidentnu granu obojenu bojom 3. Kako čvor `u` ima bar po jednu incidentnu granu bojenu bojama 2 i 3, zaključujemo da čvor `u` ima različit kôd boja od svih svojih susjeda, pa i svaki čvor grafa L ima različit kôd boja od svih svojih susjeda. Dalje, svaki čvor `w` koji je susjed čvora `u` nema incidentnih grana obojenih bojom 3, pa čvor `w`, kao i bilo koji drugi čvor grafa `L`, ima različit kôd boja od svih svojih susjeda.

2. Skup `V(L)` ne sadrži čvor iz skupa `U2`. Označimo sa `X` skup svih čvorova iz skupa `U1` $\cap$ `V(L)` čije su sve grane kojima su povezani sa čvorovima iz skupa `V[l]` za `l > 2`, obojene bojom 1. Dalje, neka je `X0` skup svih čvorova iz skupa `X` koji nemaju susjeda u skupu `W1` i neka je `X1` skup svih čvorova iz skupa `X` koji imaju tačno jednog susjeda u skupu `W1`. Neka je `X2 = X`$\setminus$`(X0` $\cup$ `X1)` i `Y = U1`$\setminus$`X`. Dalje se grananje vrši u zavisnosti od toga da li skup `V(L)` sadrži čvor iz skupa `X2`.

(i) Skup `V(L)` sadrži čvor `u` $\in$ `X2`. Imamo da je `u.kod = [b1, 0, 0]`, gdje je `b1 >= 3`. Prema lemi 1 bojimo grane grafa `L` bojama 1 i 2 tako da svi čvorovi iz skupa `V(L)` $\cup$ `(V[2]`$\setminus$`{u}` imaju paran broj incidentnih grana obojenih bojom 2, a svi čvorovi iz skupa `V(L)` $\cup$ `W2` imaju neparan broj incidentnih grana obojenih bojom 2. Ukoliko čvor `u` ima paran broj incidentih grana obojenih bojom 2 bojenje je završeno, pošto u tom slučaju svaki od čvorova grafa `L` ima različit kôd boja od svih svojih susjeda. Ukoliko čvor `u` ima neparan broj incidentnih grana obojenih bojom 2, neka je `w` jedan od susjeda čvora `u` iz skupa `W1`. Mijenjamo boju grane `uw` iz 1 u 3. Sada imamo da je `u.kod = [c1, c2, 1]`, gdje je `c1 >= 2` i `c2` je neparan broj. Nijedan susjed čvora `u` iz skupa `W1` nema nijednu incidentnu granu obojenu bojom 2. Osim toga, ako je `w` proizvoljan susjed čvora `u` u skupu `V[l]` za `l >= 3` i `w.kod = [x1, x2, x3]` i ako je `x3 = 1`, tada je `x1 < 2` ili je `x2` paran broj. Dakle, čvor `u`, kao i bilo koji drugi čvor iz skupa `L` ima različit kôd boja od svih svojih susjeda.

(ii) Skup `V(L)` ne sadrži čvor iz skupa `X2`. Neka je `Wp` skup svih čvorova iz skupa `W2` koji imaju susjeda u skupu `X0` i neka je `Xp` skup svih čvorova iz skupa `X1` koji imaju susjeda u skupu `Wp`. Sve grane koje povezuju čvorove iz skupa `X0` $\cap$ `V(L)` sa čvorovima iz skupa `Wp` bojimo bojom 3. Dalje, sve grane koje povezuju čvorove iz skupa `Wp` $\cap$ `V(L)` sa čvorovima iz skupa `Y` $\cup$ `Xp` bojimo bojom 2. Za svaki čvor iz skupa `X1`$\setminus$`Xp` mijenjamo boju jedne grane od tog čvora ka nekom čvoru `w` iz skupa `W1` iz 1 u 3. Primijetimo da sada čvor `w` imaju bar dvije incidentne grane obojene bojom 3, dok čvor `u` ima tačno jednu incidentnu granu obojenu bojom 3. Na kraju, neobojenim granama grafa `L` dodjeljujemo boju 1. Nakon opisanog postupka bojenja, za `u` $\in$ `V(L)` $\cap$ `W2` i `u.kod = [b1, b2, b3]` imamo:

a) ako je `u` $\in$ `Wp`, tada `b1 = 0`, `b2 >= 0`, `b3 >= 1`,
b) ako je `u` $\in$ `W2`$\setminus$`Wp`, tada `b1 >= 1`, `b2 = 0`, `b3 = 0`,
c) ako je `u` $\in$ `Wp`, tada `b1 = 0`, `b2 >= 0`, `b3 >= 1`.

Za `v` $\in$ `V(L)` $\cap$ `V[2]` i `v.kod = [c1, c2, c3]` imamo:

a) ako je `v` $\in$ `Y`, tada `c1 >= 0`, `c2 >= 1`, `c3 = 0`,
b) ako je `v` $\in$ `Xp`, tada `c1 >= 1`, `c2 >= 1`, `c3 = 0`,
c) ako je `v` $\in$ `X1`$\setminus$`Xp`, tada `c1 >= 1`, `c2 = 0`, `c3 = 1` i čvor `v` ima samo jednog susjeda u skupu `W1` koji ima bar dvije incidentne grane obojene bojom 3,
d) ako je `v` $\in$ `X0`, tada `c1 >= 1`, `c2 = 0`, `c3 >= 1`, i čvor `v` nema nijednog susjeda u skupu `W1`.

Zbog toga imamo da su kodovi svih susjednih čvorova grafa `L` međusobno različiti.

```
# opis: funkcija boji skup V[2]
# ulaz: graf G i niz nezavisnih dominirajucih skupova V
def oboji_V2(G, V):
    n = V.duzina
    # Vs je unija V[i] za i>2
    # Vs se inicijalizuje na prazan skup
    Vs = skup()
    for i in [3, 4, ..., n]:
        Vs = Vs | V[i]
    # W1 je skup svih cvorova iz V[1] koji imaju bar jednog susjeda u Vs
    W1 = skup()
    for s in V[1]:
        if G[s] & Vs:
            W1.dodaj(s)
    # U1 je skup svih cvorova iz V[2] koji imaju bar jednog susjeda u Vs
    U1 = skup()
    for s in V[2]:
        if G[s] & Vs:
            U1.dodaj(s)

    W2 = V[1] - W1
    U2 = V[2] - U1

    # grane izmedju W1 i V[2] boje se bojom 1
```

```
    for v in W1:
        for w in V[2]:
            if G.ima_granu(v, w):
                oboji(G, v, w, 1)

    # H2 je podgraf od G indukovan skupom cvorova W2 unija V[2]
    H = G.podgraf(W2 | V[2])

    for C in H.povezane_komponente():
        # ako je povezana komponenta jedan cvor, prelaz se na sljedecu iteraciju
        if C.duzina == 1:
            continue

        # oznacimo sa L povezanu komponentu grafa G odredjenu skupom cvorova C
        L = G.podgraf(C)

        HU2 = L & U2
        # ako HU2 nije prazan skup
        if HU2:
            # u je proizvoljan element iz HU2
            u = HU2[1]
            # ako u ima susjeda u W1
            if W1 & G[u]:
                # w je susjed od u u W1
                w = (W1 & G[u])[1]
                # svi cvorovi iz V[2]\{u} imaju paran, dok svi cvorovi iz W2
                # imaju neparan broj incidentnih grana obojenih bojom 2
                bojenje_po_lemi(G, L, u, V[2], W2, 2, 1, True)
                # ako u ima neparan broj incidentnih grana obojenih bojom 2
                # mijenja se boja grane uw iz 1 u 3
                if u.kod[2]%2 == 1:
                    promijeni_boju(G, u, w, 1, 3)
            else:
                # svi cvorovi iz V[2]\{u} imaju neparan, dok svi cvorovi iz W2
                # imaju paran broj incidentnih grana obojenih bojom 2
                bojenje_po_lemi(G, L, u, V[2], W2, 2, 1, False)
                # ako u ima paran broj incidentnih grana obojenih bojom 2
                # mijenjaju se boje grana incidentnih sa u
```

```
                if u.kod[2]%2 == 0:
                    for w in G[u]:
                        promijeni_boju(G, u, w, G[u][w].boja, 3)
        # HU2 je prazan skup
        else:
            # X je skup cvorova iz U1 i L cije su sve grane ka V[k] za
            # k>=3 obojene bojom 1
            X = skup()
            for s in U1 & L:
                if sve_obojene_sa_1(G, s, V[2])
                    X.dodaj(s)
            # X0 je skup cvorova iz X koji nemaju susjeda u W1
            X0 = skup()
            for s in X:
                if (G[s] & W1).duzina == 0:
                    X0.dodaj(s)
            # X1 je skup cvorova iz X koji imaju tacno jednog susjeda u W1
            X1 = skup()
            for s in X:
                if (G[s] & W1).duzina == 1:
                    X1.dodaj(s)

            # uvodimo skupove X2, Y i HX2 po sljedecim formulama
            X2 = (X - X0) | X1
            Y = U1 - X
            HX2 = L & X2
            # ako HX2 nije prazan skup
            if HX2:
                # u je element skupa HX2
                u = HX2[1]
                # svi cvorovi iz V[2]\{u} imaju paran, dok svi cvorovi iz W2
                # imaju neparan broj incidentnih grana obojenih bojom 2
                bojenje_po_lemi(G, L, u, V[2], W2, 2, 1, True)
                # ako u ima neparan broj incidentnih grana obojenih bojom 2
                # mijenjamo boju neke grane od u ka W1 iz 1 u 3
                if u.kod[2]%2 == 1:
                    w = (W1 & G[u])[1]
                    promijeni_boju(G, u, w, 1, 3)
```

```
            # HX2 je prazan skup
            else:
                # Wp je skup svih cvorova iz W2 koji imaju susjeda u X0
                Wp = skup()
                for s in W2:
                    if G[s] & X0:
                        Wp.dodaj(s)
                # Xp je skup svih cvorova iz X1 koji imaju susjeda u Wp
                Xp = skup()
                for s in X1:
                    if G[s] & Wp:
                        Xp.dodaj(s)

                # sve grane izmedju cvorova iz X0 (koji su u L) i Wp boje se
                # bojom 3
                for u in X0 & L:
                    for v in Wp:
                        if G.ima_granu(u, v):
                            oboji(G, u, v, 3)
                # sve grane izmedju cvorova iz Wp (koje su u L) i Y ili Xp
                # boje se bojom 2
                for u in Wp & L:
                    for v in Y | Xp:
                        if G.ima_granu(u, v):
                            oboji(G, u, v, 2)

                # za sve cvorove iz X1\Xp
                for u in X1 - Xp:
                    # w je susjed od u koji je u W1
                    w = (G[u] & W1)[1]
                    # boja grane uw se mijenja iz 1 u 3
                    promijeni_boju(G, u, w, 1, 3)
                # sve neobojene grane grafa L se boje bojom 1
                for v in L:
                    for w in G[v]:
                        if G[v][w].boja == 0:
                            oboji(G, v, w, 1)
```

4.2 Testiranje algoritma za multi-skup susjed-razlikujuće 3-bojenje grafova

Algoritam opisan u tački 4.1 testiran je na skupu svih povezanih grafova sa najmanje 3, a najviše 10 čvorova [4]. Broj svih takvih neizomorfnih grafova prikazan je sljedećom tabelom.

Broj čvorova	Broj grafova
3	2
4	6
5	21
6	112
7	853
8	11117
9	261080
10	11716571
Ukupno	11989762

Ispravnost bojenja provjerava se funkcijom `provjeri_ispravnost_bojenja` čiji pseudokod slijedi, u kome se koristi funkcija `grane` koja vraća listu grana grafa. Bojenje je ispravno ako su sve grane obojene i ako su kodovi svaka dva susjedna čvora različiti.

```
# opis: funkcija provjerava da li je bojenje ispravno msde-bojenje
# ulaz: graf G
# izlaz: True ako je graf ispravno obojen, inace False
def provjeri_ispravnost_bojenja(G):
    # za svaku granu grafa G
    for e in G.grane():
        # e = uv
        u = e[0]
        v = e[1]
        # ako postoji grana kojoj nije dodijeljena boja iz skupa {1, 2, 3}
        # bojenje je neispravno
        if G[u][v].boja == 0:
            return False
        # ako su kodovi cvorova u i v jednaki, bojenje je neispravno
        if u.kod == v.kod:
            return False
    # inace, bojenje je ispravno
    return True
```

Pored toga, algoritam je testiran na nekim klasama grafova iz *Python* biblioteke *networkx*. Ova biblioteka nudi veliki broj klasa poznatih grafova. Neke od klasa grafova koje su korišćene date su u nastavku. Sva ograničenja za m i n su odabrana tako da se u svakoj od klasa pokrije dovoljan broj grafova različitih veličina. Biblioteka *networkx* omogućava kreiranje ovakvih grafova sa proizvoljnim brojem čvorova.

- Uravnotežena potpuna binarna stabla visine n, gdje je $1 \leq n \leq 15$.
- Teg (barbell) je graf koji se sastoji od dva kompletna grafa K_n i jednog puta dužine m koji ih povezuje. Put koji povezuje dva kompletna grafa K_n naziva se most. Program je testiran na ovakvim grafovima za $n \in \{3, 5, 10, 20, 50, 100, 500, 1000\}$ i $m \in \{1, 2, 5, 10, 20, 100\}$.
- Kompletni grafovi K_n, za $n \in \{5, 10, 20, 50, 100, 250, 500, 1000\}$.
- Kompletni multipartitni grafovi sa n particija od kojih se svaka sastoji od m čvorova. Program je testiran na ovakvim grafovima za $n \in \{5, 10, 20\}$ i $m \in \{10, 50, 100\}$.
- Merdevine (ladder) su graf koji se sastoji od dva puta dužine n u kojima je svaki od n parova čvorova povezan granom. Program je testiran na ovakvim grafovima za $n \in \{5, 10, 20, 50, 100, 250, 500, 1000, 1250, 1500, 1750, 2000, 2500, 3000, 5000, 10000\}$.
- Kružne merdevine (circular ladder) su graf koji se sastoji od dva koncentrična ciklusa C_n u kojima je svaki od n parova koncentričnih čvorova povezan granom. Program je testiran na ovakvim grafovima za $n \in \{5, 10, 20, 50, 100, 250, 500, 1000, 1250, 1500, 1750, 2000, 2500, 3000, 5000, 10000\}$.
- n-dimenzione hiperkocke za $n \in \{2, 3, 4\}$ sa po $m = 10$ čvorova u svakoj „dimenziji“. Program je testiran i na periodičnim verzijama ovih grafova, koje se dobijaju tako što se krajnji čvorovi ovih grafova povežu granama.
- n-dimenzione binarne hiperkocke za $n \in \{2, 3, ..., 15\}$.
- Lilihip (lollipop) graf se sastoji od jednog kompletnog grafa K_n povezanog sa putem dužine m. Program je testiran na ovakvim grafovima za $n \in \{5, 10, 50, 100, 200, 500, 1000\}$ i $m \in \{2, 10, 50, 100\}$.
- Put dužine n za $n \in \{5, 10, 20, 50, 100, 250, 500, 1000, 1250, 1500, 1750, 2000, 2500, 3000, 5000, 10000\}$.
- Zvijezda je graf koji se sastoji od jednog centralnog čvora povezanog sa n spoljašnjih čvorova. Program je testiran na ovakvim grafovima za $n \in \{5, 10, 20, 50, 100, 250, 500, 1000, 1250, 1500, 1750, 2000, 2500, 3000, 5000, 10000\}$.
- Točak je graf koji se sastoji od jednog centralnog čvora povezanog sa $n-1$ spoljašnjih čvorova koji formiraju graf C_{n-1}. Program je testiran na ovakvim grafovima za $n \in \{5, 10, 20, 50, 100, 250, 500, 1000, 1250, 1500, 1750, 2000, 2500, 3000, 5000, 10000\}$.

U sljedećoj tabeli prikazana su vremena izvršavanja algoritma za multi-skup susjed-razlikujuće 3-bojenje za neke od prethodno navedenih grafova određene veličine.

Naziv grafa	n	m	Vrijeme bojenja u sekundama
Uravnoteženo potpuno binarno stablo	12	-	4.079
Teg graf	250	100	1.988
Kompletan graf	500	-	4.853
Merdevine graf	5000	-	3.702
Kružne merdevine graf	5000	-	3.654
Hiperkocka	3	10	0.121
Binarna hiperkocka	12	-	3.295
Lilihip graf	500	100	5.388
Put	10000	-	3.410
Zvijezda	2000	-	3.791
Točak	10000	-	7.040

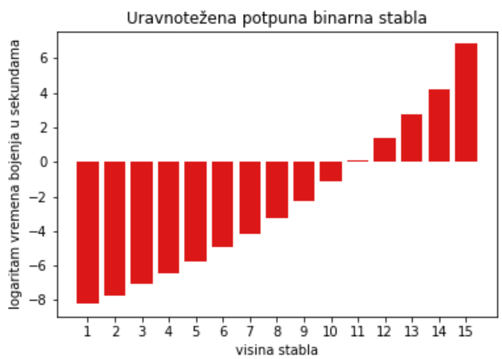
Prikazano vrijeme izvršavanja predstavlja prosječno vrijeme dobijeno ponavljanjem multi-skup susjed-razlikujućeg 3-bojenja deset puta.

Na slici 3 prikazana je zavisnost logaritma trajanja bojenja u sekundama od logaritma veličine grafa (broja čvorova grafa). Na y-osi je prikazan logaritam prosječnog trajanja bojenja u sekundama, dok je na x-osi prikazan logaritam veličine grafa. Vrijeme koje je prikazano predstavlja aritmetičku sredinu vremena dobijenih ponavljanjem bojenja deset puta. Može se primijetiti da je zavisnost prikazana dijagramima linearna za sve klase grafova, sa nagibom koji je blizak broju 2.

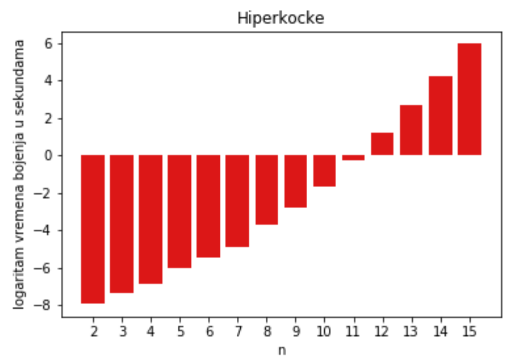
4.3 Algoritam za slučajno bojenje grafova

Pretpostavimo da je dat skup S koji se sastoji od k boja označenih sa $\{1, 2, ..., k\}$. Ideja algoritma je slučajno bojenje grana grafa u proizvoljnom redosljedu bojama iz skupa S, pri čemu sve boje imaju jednaku vjerovatnoću izbora. Tokom bojenja se ažuriraju kodovi čvorova grafa, a nakon što se dodijele boje svim granama, vrši se provjera da li je bojenje ispravno multi-skup susjed-razlikujuće bojenje grafa. Promjenljiva `BROJ_PONAVLJANJA` određuje broj pokušaja bojenja uz fiksiran broj boja, dok promjenljiva `MAKS_BROJ_BOJA` označava maksimalan broj boja sa kojima se pokušava obojiti graf.

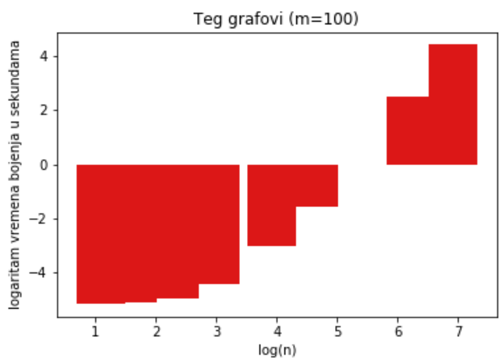
Algoritam vrši bojenje sa najmanje dvije, a najviše `MAKS_BROJ_BOJA` boja i završava se ukoliko se postigne ispravno multi-skup susjed-razlikujuće bojenje, ili se dostigne maksimalan broj pokušaja bojenja maskimalnim brojem boja. Podrazumijevane vrijednosti parametara `BROJ_PONAVLJANJA` i `MAKS_BROJ_BOJA` su redom 10 i 5. Pseudokod algoritma prikazan je u nastavku. U okviru algoritma pretpostavlja se da je na raspolaganju funkcija `slucajan_broj(k)` koja vraća slučajan broj iz skupa $\{1, 2, ..., k\}$, pri čemu svaki od k brojeva ima jednaku vjerovatnoću izbora.



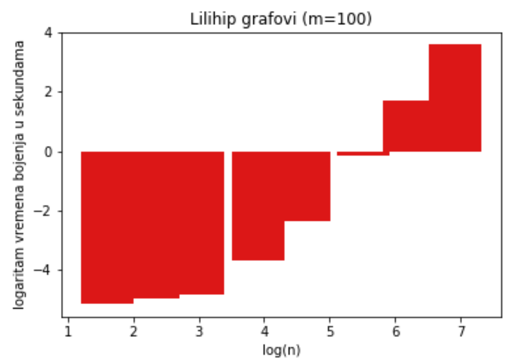
(a) uravnotežena potpuna binarna stabla (na x-osi je prikazana visina stabla)



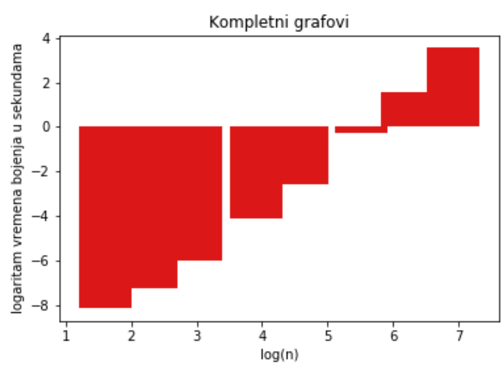
(b) hiperkocke (na x-osi je prikazana dimenzija hiperkocke)



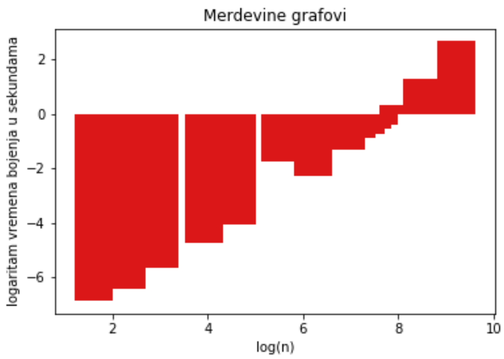
(c) teg grafovi (na x-osi je prikazan $log(n)$, dok je parametar $m = 100$ fiksiran)



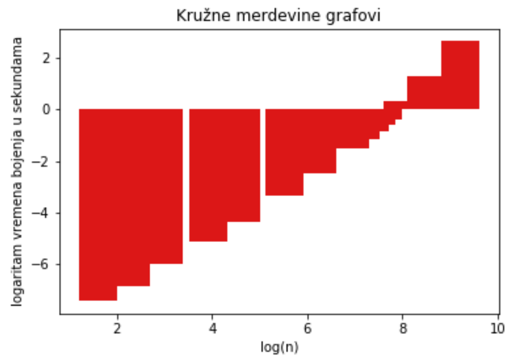
(d) lilihip grafovi (na x-osi je prikazan $log(n)$, dok je parametar $m = 100$ fiksiran)



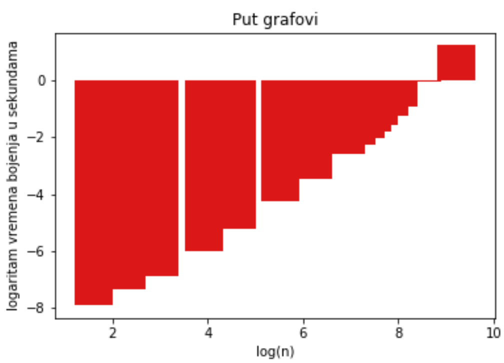
(e) kompletni grafovi (na x-osi je prikazan $log(n)$)



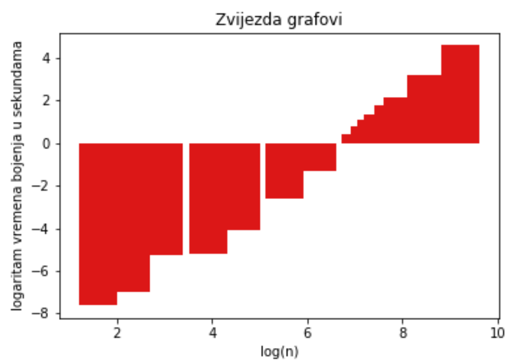
(f) merdevine grafovi (na x-osi je prikazan $log(n)$)



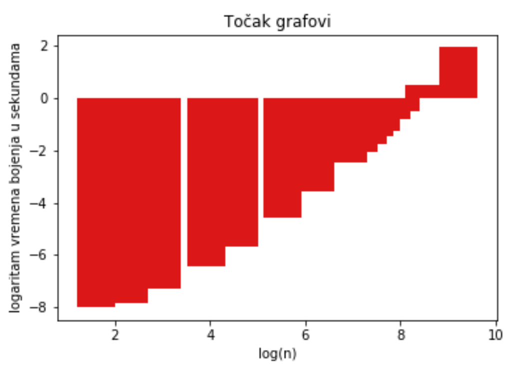
(g) kružne merdevine grafovi (na x-osi je prikazan $log(n)$)



(h) put grafovi (na x-osi je prikazan $log(n)$)



(i) zvijezda grafovi (na x-osi je prikazan $log(n)$)



(j) točak grafovi (na x-osi je prikazan $log(n)$)

Slika 3: Slike prikazuju trajanje multi-skup susjed-razlikujućeg 3-bojenja u sekundama za odabrane klase grafova određenih dimenzija. Na y-osi prikazan je logaritam trajanja bojenja u sekundama, dok je na x-osi prikazan logaritam veličine grafa. Za grafove koji zavise od dva parametra, jedan parametar je fiksiran.

```
# opis: funkcija za slucajno bojenje grafa G
# ulaz: graf G, BROJ_PONAVLJANJA=10, MAKS_BROJ_BOJA=5
# izlaz: True u slucaju uspjesnog msde-bojenja, u suprotnom False
def slucajno_bojenje_grafa(G, BROJ_PONAVLJANJA=10, MAKS_BROJ_BOJA=5):
    # za svaki zadati broj boja, i broj ponavljanja bojenja
    for k in [2,3,...,MAKS_BROJ_BOJA]:
        for i in [1,2,...,BROJ_PONAVLJANJA]:
            # za svaku granu e grafa G
            for e in G.grane():
                u = e[0]
                v = e[1]
                # grana e=uv se boji bojom iz skupa {1, ..., k}
                oboji(G, u, v, slucajan_broj(k))
            if provjeri_ispravnost_bojenja(G):
                return True
    return False
```

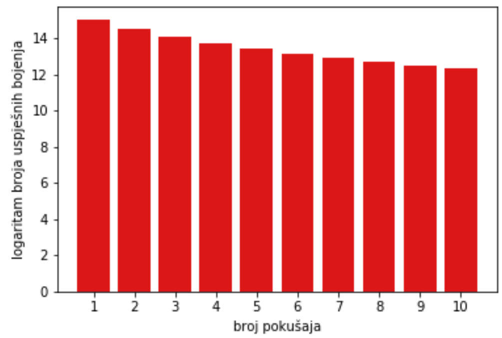
Ovaj algoritam je implementiran u programskom jeziku *Python* korišćenjem biblioteke *networkx*, dok je za izbor slučajnog broja korišćena funkcija *random* iz biblioteke *numpy*. Program je testiran je na 11716571 povezanih grafova sa 10 čvorova uz podrazumijevane vrijednosti parametara `BROJ_PONAVLJANJA` i `MAKS_BROJ_BOJA` 10 i 5 redom.

Ispostavlja se da je 5 boja i 10 pokušaja bojenja grafova svakom od njih bilo dovoljno za uspješno bojenje svih ovih grafova. Štaviše, nijedan graf nije obojen sa 5 boja u posljednjem pokušaju, ali su dva grafa obojena sa 5 boja u pretposljednjem pokušaju. Rezultati bojenja prikazani su sljedećom tabelom.

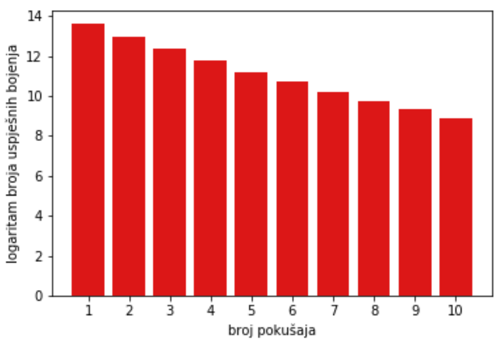
Broj boja	Broj obojenih grafova	Procenat obojenih grafova
2	9909344	84.575%
3	1790070	15.278%
4	17108	0.146%
5	49	<0.001%

Dakle, ovim algoritmom je pronađeno uspješno multi-skup susjed-razlikujuće bojenje sa dvije boje za skoro 85% grafova, dok je više od 99.85% grafova moguće obojiti sa dvije ili tri boje. Stoga slučajno bojenje može biti prihvatljiv način za multi-skup susjed-razlikujuće bojenje grafova ukoliko ne postoji zahtjev da se svaki graf mora obojiti sa najviše tri boje. Pitanje inspirisano ovim eksperimentom je - da li se mogu okarakterisati svi grafovi za koje postoji ispravno multi-skup susjed-razlikujuće bojenje sa dvije boje.

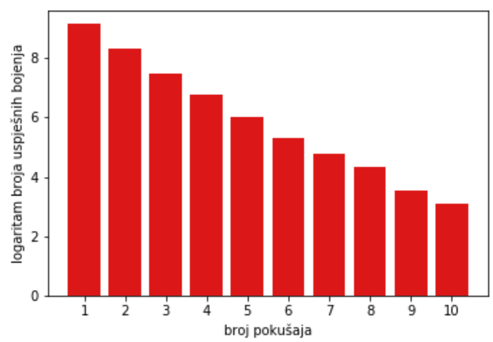
Zanimljivo je uočiti i koliko je pokušaja bilo potrebno da se svaki od grafova uspješno oboji, za svaki od brojeva boja. Odgovarajući grafici prikazani su na slici 4 uz korišćenje logaritamske skale na $y-$osi uz konvenciju da se prikazuje 0 ukoliko nijedan graf nije obojen nekom bojom u nekom broju pokušaja. Može se primijetiti da broj grafova koji su obojeni određenom bojom eksponencijalno opada sa brojem pokušaja. Dakle, većina grafova se oboji sa manjim brojem pokušaja, a jako mali broj grafova sa brojem pokušaja koji je blizak 10. Ukupan broj grafova obojenih sa 5 boja je svega 49, pa slika 4 d) nije reprezentativna za donošenje zaključka o zavisnosti između broja pokušaja i broja uspješno obojenih grafova.



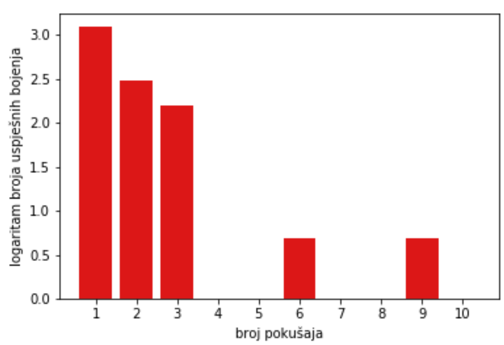
(a) dvije boje



(b) tri boje



(c) četiri boje



(d) pet boja

Slika 4: Slike prikazuju koliko je bojenja bilo uspješno sa kojim brojem pokušaja i sa koliko boja (broj pokušaja je između 1 i 10 i prikazan je na x-osi, broj boja je između 2 do 5, a na y-osi je prikazan logaritam uspješnog broja bojenja grafova).

5 Zaključak

U radu je opisan algoritam za multi-skup susjed-razlikujuće bojenje grafova sa tri boje. Algoritam je implementiran u programskom jeziku *Python* i testiran je na svim povezanim grafovima sa najmanje 3, a najviše 10 čvorova [4], kao i na nekim od poznatih klasa grafova. Pored toga, predstavljen je i algoritam za slučajno bojenje grafova, koji je postigao uspješno multi-skup susjed-razlikujuće bojenje sa najviše tri boje za više od 99% grafova od svih povezanih grafova sa 10 čvorova. Takođe, skoro 85% grafova je na ovaj način obojeno sa dvije boje, pa bi jedan od mogućih narednih zadataka bio da se okarakterišu grafovi za koje postoji multi-skup susjed-razlikujuće bojenje sa dvije boje. Ukoliko bi se takvi grafovi okarakterisali, možda bi za njih postojao jednostavniji algoritam za multi-skup susjed-razlikujuće bojenje od algoritma predstavljenog u ovom radu. Bilo bi zanimljivo i primijeniti neke od heuristika na algoritam slučajnog bojenja, i analizirati uticaj heuristika na broj potrebnih boja i broj potrebnih pokušaja za bojenje.

Literatura

[1] Bojan Vučković, *Multi-set neighbor distinguishing 3-edge coloring,* Discrete Mathematics, Volume 341, Issue 3, 820-824, March 2018.

[2] G. Chartrand, P. Zhang, *Chromatic Graph Theory* Boca Raton: Chapman & Hall/CRC, 2009.

[3] B. D. McKay, A. Piperno, *Practical Graph Isomorphism*, II, J. Symbolic Computation, 2013

[4] Combinatorial Data: https://users.cecs.anu.edu.au/~bdm/data/graphs.html