ALGORITMI ZA BOJENJE GRAFOVA

Gojić, Andreja

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

ALGORITMI ZA BOJENJE GRAFOVA

Završni rad

Andreja Gojić

Osijek, 2021.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 28.08.2021.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Andreja Gojić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4199, 24.07.2018.
OIB studenta:	24332401510
Mentor:	Izv. prof. dr. sc. Alfonzo Baumgartner
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Algoritmi za bojenje grafova
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	28.08.2021.
Datum potvrde ocjene Odbora:	08.09.2021.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 08.09.2021.

Ime i prezime studenta:	Andreja Gojić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4199, 24.07.2018.
Turnitin podudaranje [%]:	10

Ovom izjavom izjavljujem da je rad pod nazivom**: Algoritmi za bojenje grafova**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Alfonzo Baumgartner

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA

o odobrenju za pohranu i objavu ocjenskog rada

kojom ja __Andreja Gojić_____, OIB: __24332401510_____, student/ica Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek na studiju _Preddiplomski sveučilišni studij Računarstvo_, kao autor/ica ocjenskog rada pod naslovom: _Algoritmi za bojenje grafova**,**

dajem odobrenje da se, bez naknade, trajno pohrani moj ocjenski rad u javno dostupnom digitalnom repozitoriju ustanove Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek i Sveučilišta te u javnoj internetskoj bazi radova Nacionalne i sveučilišne knjižnice u Zagrebu, sukladno obvezi iz odredbe članka 83. stavka 11. *Zakona o znanstvenoj djelatnosti i visokom obrazovanju* (NN 123/03, 198/03, 105/04, 174/04, 02/07, 46/07, 45/09, 63/11, 94/13, 139/13, 101/14, 60/15).
Potvrđujem da je za pohranu dostavljena završna verzija obranjenog i dovršenog ocjenskog rada. Ovom izjavom, kao autor/ica ocjenskog rada dajem odobrenje i da se moj ocjenski rad, bez naknade, trajno javno objavi i besplatno učini dostupnim:

a) široj javnosti
b) studentima/icama i djelatnicima/ama ustanove
c) široj javnosti, ali nakon proteka 6 / 12 / 24 mjeseci (zaokružite odgovarajući broj mjeseci).

**U slučaju potrebe dodatnog ograničavanja pristupa Vašem ocjenskom radu, podnosi se obrazloženi zahtjev nadležnom tijelu Ustanove.*

__Osijek, 08.09.2021._____ _____

(mjesto i datum) (vlastoručni potpis studenta/ice)

Sadržaj

1. UVOD

Teorija grafova je široko prisutna u rješavanju problema iz raznih područja poput računalne tehnologije, genetike, komunikacijskih znanosti te mnogih drugih. Pojedini dijelovi teorije grafova nastali su upravo za potrebom primjene u rješavanju problema iz nekog drugog područja pa je tako za problem električnih mreža osmišljena teorija acikličkih grafova, za nabrajanje izomera organskih spojeva je započelo proučavanje stabala, dok je bojenjem karata razvijeno bojenje grafova.

Na samom početku rada je odgovoreno na pitanja što su to grafovi, kakve vrste grafova postoje te na koji način se mogu zapisati. U nastavku rada je pojašnjeno što je bojenje grafova, kako je došlo do potrebe za bojenjem grafova te u kojim stvarnim problemima se primjenjuje, zatim su predstavljeni postojeći algoritmi pomoću kojih se grafovi mogu obojiti. Nakon toga su opisane korištene tehnologije prilikom izrade aplikacije u kojoj je implementiran odabrani algoritam bojenja grafova, a na kraju rada je objašnjeno na koji način se izvodi aplikacija te njeno djelovanje na primjeru konkretnog grafa.

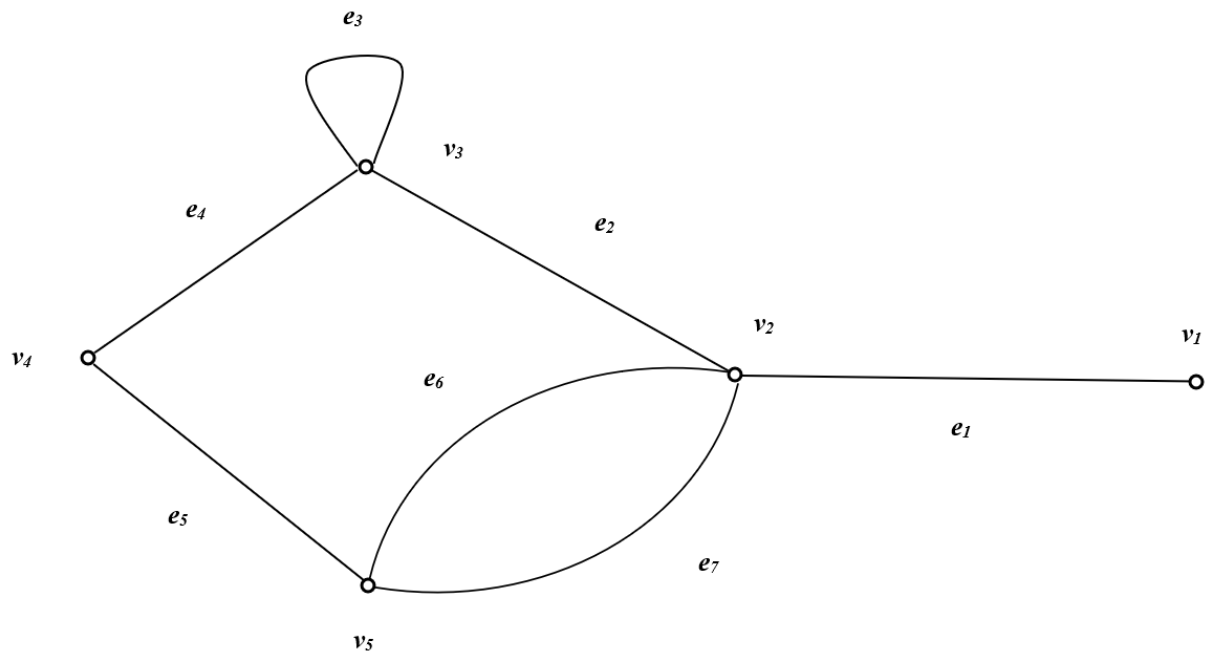
1.1. Zadatak završnog rada

Cilj ovog rada je teorijski opisati problem bojenja grafova, istražiti postojeće algoritme za njegovo rješavanje te pokazati gdje je ovakav pristup rješavanja problema pronašao svoju primjenu. U praktičnom dijelu rada potrebno je odabrati jedan od postojećih algoritama te ga implementirati u proizvoljnom programskom jeziku, isprobati njegov rad na nekoliko primjera, a rezultat prikazati u grafičkom ili tekstualnom obliku.

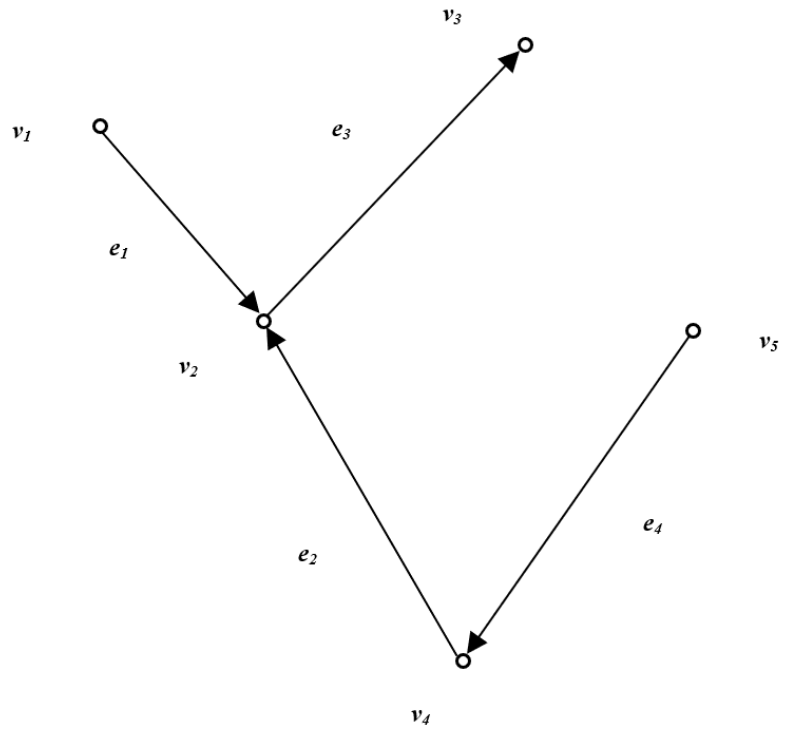
2. TEORIJA GRAFOVA

Grafovi su nazvani tako jer se mogu grafički predstaviti te nam taj način prikazivanja pomaže razumjeti mnoga njihova svojstva. Svaki vrh označen je točkom, a svaki brid linijom spajajući točke koje predstavljaju njegove krajeve kao što je prikazano na slici 1.1. Bridovi mogu biti usmjereni i neusmjereni, ukoliko su na bridovima određeni smjerovi tada se graf naziva usmjerenim grafom, a u suprotnom neusmjerenim grafom. Neusmjereni graf se nalazi na slici 1.1., dok usmjereni na slici 1.2.. [1]

Graf $G = (V, E)$ sastoji se od skupova $V = \{v_1, v_2, \ldots\}$, koji označava vrhove, te $E = \{e_1, e_2, \ldots\}$, koji obuhvaća bridove, na način da svaki brid međusobno povezuje dva vrha. Vrhovi v_i, v_j povezani s bridom e_k nazivaju se krajnji vrhovi brida e_k. Bridovi mogu spajati dva različita vrha ili mogu izlaziti i ulazi u isti vrh, tada se nazivaju petljom poput petlje e_3 na slici 1.1. Također dva vrha mogu biti povezana s dva ili više brida, oni su tada linearni bridovi, kao što su brid e_6 i e_7. [2]



Slika 1.1, *Prikaz neusmjerenog grafa*



Slika 1.2. *Prikaz usmjerenog grafa*

Graf koji ne sadrži petlje i paralelne bridove naziva se jednostavnim grafom. U nekim literaturama o teoriji grafova graf se uvijek definira kao jednostavan graf, ali u većini primjena grafova u inženjerstvu je nužno koristiti ili paralelne bridove ili petlje ili oboje. [2]

Graf se naziva konačnim ako su skupovi vrhova $G(V)$ i bridova $G(E)$ konačni. Graf koji nije konačan naziva se beskonačnim grafom. U ovom radu bavit razmatrat će se samo konačnim grafovima. [3]

Dva grafa G i H su identični ako su $V(G) = V(H)$ i $E(G) = E(H)$. Ako su dva grafa identična oni se tada mogu prikazati identičnim dijagramima. Međutim, i za grafove koji nisu identični je moguće da u osnovi imaju isti dijagram. [1]

Neka su $G = (V(G), E(G))$ i $H = (V(H), E(H))$ dva grafa. Tada je izomorfizam grafa od G do H par funkcija (ϕ, θ), gdje su $\phi : V(G) \rightarrow V(H)$ i $\theta : E(G) \rightarrow E(H)$ bijektne funkcije. Ako je (ϕ, θ) izomorfizam grafa, tada je par inverznih preslikavanja (ϕ^{-1}, θ^{-1}) također izomorfizam grafa. Primijetimo da bijekcija ϕ zadovoljava uvjet da su u i v krajnji vrhovi brida e grafa G ako i samo ako $\phi(u)$ i $\phi(v)$ su krajnji vrhovi ruba $\theta(e)$ u grafu H. izomorfizam između grafova označava se simbolom $\simeq$. [3]

Jednostavan graf u kojem je svaki par različitih vrhova povezan s bridom naziva se potpuni graf te se označava s K_n, gdje n predstavlja broj vrhova. Bipartitni graf je graf u kojem se vrhovi mogu podijeliti u dva podskupa X i Y, tako da svaki brid povezuje jedan vrh iz podskupa X s jednim vrhom iz podskupa Y. Bipartitni grafovi se označavaju s $K_{m,n}$, gdje m označava broj vrhova iz podskupa X, a n broj vrhova iz podskupa Y. Prazan graf ili nul-graf je onaj graf koji nema niti jedan brid. [3]

Grafovi se mogu zapisati pomoću matrice susjedstva ili pomoću popisa susjedstva. Matrica susjedstva predstavlja kvadratnu matricu koja ima broj stupaca i redaka jednako koliko i vrhova. Svaki stupac i svaki redak služi za jedan vrh, odnosno svaki vrh ima jedan stupac i jedan redak, a oni se popunjavaju nulama i jedinicama ukoliko postoji brid između dva vrha ili petlja jednog vrha. Popisom susjedstva se zapisuju bridovi na način da postoji popis svih vrhova te svaki vrh sadrži niz vrhova s kojima je povezan. [1]

3. BOJENJE GRAFOVA

Za graf *G* se kaže da je *k*-obojiv ukoliko je moguće obojiti svaki vrh s jednom od *k* boja na način da dva susjedna vrha ne budu iste boje. Ako je graf *k*-obojiv, a nije ga moguće obojiti s *k-1* boja, kaže se da je to *k*-kromatski graf te je tada *k* njegov kromatski broj $\chi(G)$. Dakle *k* je minimalan broj boja potrebnih kako bi se vrhovi grafa obojili tako da dva susjedne vrha ne budu iste boje. Na isti način se može definirati i kromatski broj $\chi'(G)$ koji predstavlja najmanji broj boja *k* kojim se boje bridovi grafa. Stoga se bojenje grafova može podijeliti na bojenje vrhova i bojenje vrhova. [3]

3.1. Primjena bojenja grafova

Francis Guthrie, tadašnji student matematike, prvi je otkrio problem bojenja grafova 1852. godine. Prilikom bojenja karte koja se sastojala od šesnaest engleskih okruga primijetio je da su potrebne četiri boje kako bi obojio sve susjedne županije različitom bojom. Kako bi ovaj problem pretvorili u bojenje grafova potrebno je županije promatrati kao vrhove grafa, a bridove grafa odredi na osnovu susjednih županija s kojima svaka županija graniči. Ovakav pristup bojenja grafova se može primijeniti samo na planarne grafove, a s obzirom da sve karte možemo promatrati kao planarne grafove, dakle primjenjivo je na sve karte. [4]

Osim u kartografiji, bojenje grafova pronašlo je svoju primjenu i u raznim planiranjima, poput planiranja rasporeda sati u obrazovnom sustavu ili nekim drugim postrojenjima koje zahtijevaju rezervaciju prostorija i održavanje aktivnosti za određene grupe ljudi kako bi se izbjeglo preklapanje termina, zatim u planiranju letova zrakoplova, za rješavanje sudoku zagonetki, postavljanje GSM (engl. *Groups special Mobile*) mreže, dodjeljivanje registara procesora prilikom obavljanja određenih zadataka te razne druge primijene. Na primjeru određivanja rasporeda sati bit će objašnjena primjera bojenja grafova. [5]

Prilikom određivanja rasporeda sati važno je definirati da svaki profesor u jednom terminu može predavati samo jedan predmet te svaki predmet predaje samo jedan profesor. Prema tome profesori se promatraju kao jedan skup *t*, predmeti kao skup *s*, a termini kao skup *p*, gdje termini skup *p* povezuje profesore s predmetima. Na osnovi toga može se konstruirati bipartitni graf u kojem će svako podudaranje biti potencijalno dodjeljivanje pojedinog profesora određenom predmetu, a do konačnog se rješenja problema dolazi particioniranjem rubova grafa na minimalan broj podudaranja koji se također moraju obojiti s minimalnim brojem boja. [5]

3.2. Algoritmi za bojenje grafova

Algoritmi se sastoje od niza naredbi koje svojim slijedom dolaze do rješenja problema, oni su „recepti“ za rješavanje određenih matematičkih problema. Svaki korak algoritma je točno definiran, a broj koraka mora biti konačan kako bi se uspješno riješio problem. Svaki algoritam mora ispuniti pet uvjeta, a to su definiranost, efikasnost, konačnost te ulazne i izlazne vrijednosti. Algoritmi mogu biti pisani bilo kojim jezikom kojim se ljudi sporazumijevaju, programskim jezikom ili pseudokodom koji predstavlja kombinaciju prethodna dva. [2]

Kako je ranije rečeno grafovi se mogu bojiti na način da se boje vrhovi ili bridovi, stoga se algoritmi za bojenje grafova mogu podijeliti u dvije skupine, prva skupina su algoritmi za bojenje vrhova, dok je druga skupina algoritama za bojenje bridova. Prema [6] postoji dvanaest postojećih algoritama za rješavanje problema bojenja grafova, a oni su opisani u sljedećim potpoglavljima.

3.2.1. Iscrpno pretraživanje (engl. *Exhaustive search*)

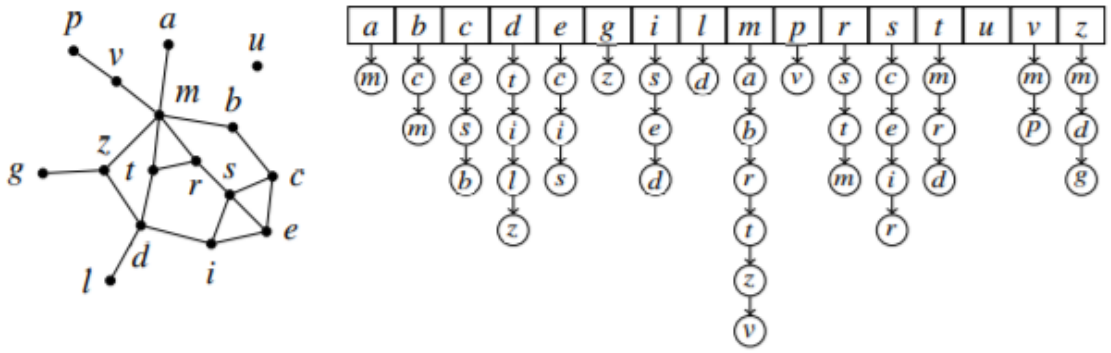
S obzirom na cijeli broj $q \geq 1$ i graf G s vrhom skup V, ovaj algoritam pronalazi bojenje vrhova pomoću q boja ako postoji, pseudokod ovog algoritma se nalazi na slici 3.1.. [6]

```
[X1] Za svako preslikavanje f : V → [1, 2, ..., q] čini
    [X2] Ako svaki vrh vw zadovoljava f(v) ≠ f(w) onda
        Završi s f kao rezultat
```

Slika 3.1. *Pseudokod algoritma iscrpnog pretraživanja*

Kako bi se odredilo vrijeme potrebno da se ovaj algoritam izvrši, treba pretpostaviti matematički model koji uključuje ponavljanje svih preslikavanja od jednog konačnog skupa A do drugog konačnog skupa B u vremenu O(|B||A|) (Korak X1), ili prolazak kroz sve bridove grafa u vremenu O(n+m) (Korak X2). Ako se pretpostavi da je ulazni graf prikazan pomoću niza nizova, odnosno popisa susjedstva; niz koji s indeksom v sadrži N(v) nizova, kao na slici 3.2.. Dakle za graf s n vrhova i m bridova vremenska složenost iznosi O(qn(n + m)). [6]

Pojednostavljene varijante ovog algoritma omogućuju rješavanje drugih problema bojenja grafova. Na primjer, za pronalaženje popisa bojenja, ograničava se raspon vrijednosti za svaku *f(v)* na danom popisu te se za pronalaženje boje brida ponavljaju preslikavanja $f : E \rightarrow \{1, 2, ..., q\}$. Druga modifikacija je brojanje broja boja umjesto pronalaženja samo jedne. Ova proširenja pružaju osnovne algoritme za bojenje popisa, bojenje bridova, kromatski polinom, kromatski indeks, itd. [6]



Slika 3.2. *Zapis grafa* [6]

3.2.2. Pohlepno bojenje (engl. *Greedy colouring*)

Za dani graf G s najvećim stupnjem Δ i određenim vrhovima $v_1, v_2, \ldots, v_n$ ovaj algoritam pronalazi boje vrhova s $max_i\ |\{\ j < i:\ v_j v_i\ \in E\ \}| + 1 \leq \Delta + 1$ boja, prikaz pseudokoda ovoga algoritma se nalazi na slici 3.3..[6]

[G1] $i = 0$

[G2] ***Povećaj*** i, ***ako je*** $i = n + 1$ ***onda***

Završi *s f kao rezultat*

[G3] ***Izračunaj*** *skup* $C = \cup_{j<i} f(v_j)$ *već dodijeljenih boja susjedima od vrha* v_i

[G4] ***Ako*** $c = 1, 2, \ldots$ ***nije*** $c \in C$ ***onda***

$i = f(v_i)$

Slika 3.3. *Pseudokod algoritma pohlepnog bojenja*

Broj boja c u koraku G4 može postići najveću vrijednost $|C|$, koji je ograničen brojem susjeda v_i između $v_1, v_2, \ldots, v_{i-1}$. Ovaj algoritam utvrđuje da je $\chi(G) \leq \Delta(G) + 1$. Za vrijeme izvođenja oba koraka, **G3** i **G4**, uzimaju najviše $O(1 + \deg v_i)$ operacije. Zbrajajući sve i, ukupno vrijeme provedeno u koracima **G3** i **G4** je asimptotski omeđen s $n + (\deg v_1 + \deg v_2 + \cdots + \deg v_n) = n + 2m$. Tako, algoritam pohlepnog bojenja zahtijeva vrijeme $O(n + m)$. [6]

Algoritam pohlepnog bojenja prilično dobro radi na slučajnim grafovima, bez obzira na redoslijed vrhova. Za gotovo sve grafove s n vrhova koristi $n/(\log n - 3\log\log n)$ boje, što je otprilike dvostruko veća od optimalne vrijednosti, stoga ovaj algoritam nije najučinkovitiji. [6]

3.2.3. Bipartitni algoritam (engl. *Bipartition*)

Neka je dan povezani graf *G*, ovaj algoritam traži dvije boje za njegovo bojenje ako on postoji. U suprotnom prikazuje neparan ciklus, pseudokod ovoga algoritma se nalazi na slici 3.4.. [6]

[B1] $f(v_1) = 1$

Q = *prazan niz*

Za svakog *susjeda* w *od* v_1

| $r(v_1) = w$

| ***Dodaj*** w *nizu* Q

[B2] ***Ako je*** Q *prazan niz*

| ***Idi*** *na korak* **[B3]**

Inače

| ***ukloni*** *prvi vrh* v *iz* Q *i postavite* $f(v)$ *na boju koja još nije dodijeljena* $r(v)$

| ***Za svakog*** *susjeda* w *od* v

| | ***Ako*** w *nije obojen i nije* $w \in Q$

| | | $r(w) = v$

| | | ***Dodaj*** w *na kraj niza* Q

| ***Ponovi*** **[B2]**

[B3] ***Za svaki*** *brid* vw ***provjeri*** *jesu li* $f(v) \neq f(w)$

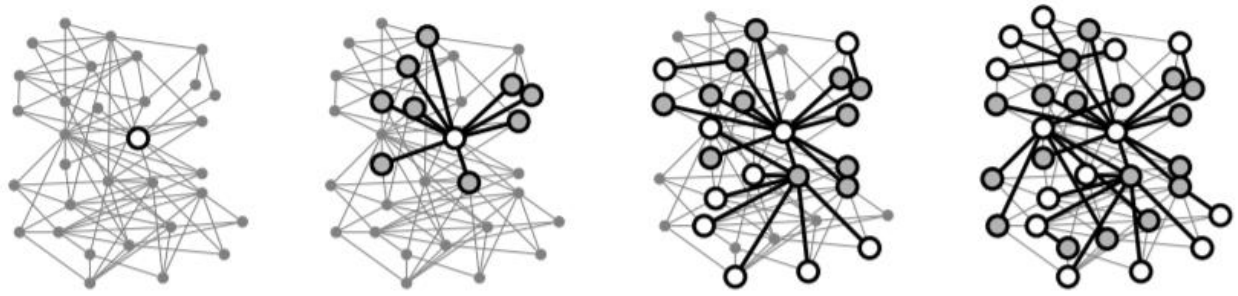
Ako *su različiti* ***završi*** *s* f *kao rezultat*

[B4] Neka je vw brid s $f(v) = f(w)$ i neka je u najbliži zajednički roditelj v i w u stablu definiranom r. ***Iznesi*** *putanju* $w, r(w), r(r(w)), \ldots, u$, *nakon čega slijedi preokret puta* $v, r(v), r(r(v)), \ldots, u$, *nakon čega slijedi rub* vw.

Slika 3.4. *Pseudokod bipartitnog algoritma*

Bipartitni algoritam radi na principu potvrde, to jest može provjeriti njegova ispravnost. U slučaju kada je ciklus neparan tada graf nije moguće obojiti s dvije boje. U koraku B4 se provjerava

ima li ulazni graf neparnu duljinu na način da se promatraju dva puta, put w, r(w), r(r(w)) ,. . . , u te put v, r(v), r(r(v)) ,. . . , svaki vrh je obojen drugačijom bojom od prethodnog vrha, stoga krajnje točke oba puta moraju biti obojene istom bojom ukoliko ti putovi sadrže jednak broj bridova. Način izvođenja bipartitnog algoritma je prikazan na slici 3.5..[6]



Slika 3.5. *Prikaz rada bipartitnog algoritma* [6]

3.2.4. Ograničenje palete (engl. *Palette restriction*)

Za dani graf *G*, ovaj algoritam boja graf s tri boje, ukoliko on postoji, prikaz pseudokoda ovoga algoritma se nalazi na slici 3.6.. [6]

[P1] ***Za svaki*** *vrh v* ***nasumično odaberi*** *popis boja L(v) iz skupa {1, 2}, {2, 3} i {1, 3}*

[P2] ***Idi*** *na korak* **B1**

Ako *je graf obojiv* ***završi***

Inače idi *na korak* **P1**

Slika 3.6. *Pseudokod algoritma ograničenja palete* [6]

Svakom vrh *v* ulaznog grafa svaka pojedina boja *f(v)* može biti dodijeljena s vjerojatnošću $\frac{2}{3}$. Prema tome popis koji je kreiran u koraku **P1** će imati rješenje s vjerojatnošću $(\frac{2}{3})^n$, iz toga proizlazi da je očekivani broj ponavljanja $(\frac{2}{3})^n$, a pritom svakom ponavljanju treba polinomijalno vrijeme. [6]

3.2.5. Widgersonov algoritam (engl. *Wigderson's algorithm*)

Za dani 3-kromatski graf *G*, ovaj algoritam pronalazi bojenje vrhova s $O(\sqrt{n})$ boja, pseudokod ovog algoritma prikazan je na slici 3.7.. [6]

[W1] *c = 1*

[W2] ***Ako*** *ne postoji vrh v s* deg *v* ≥ $[\sqrt{n}]$

Idi *na korak* **W3**

Inače

Idi *na korak* **B1** *za 2_bojenje susjedstava G[N(v)] s bojama c i c + 1*

Ukloni *N(v) iz G*

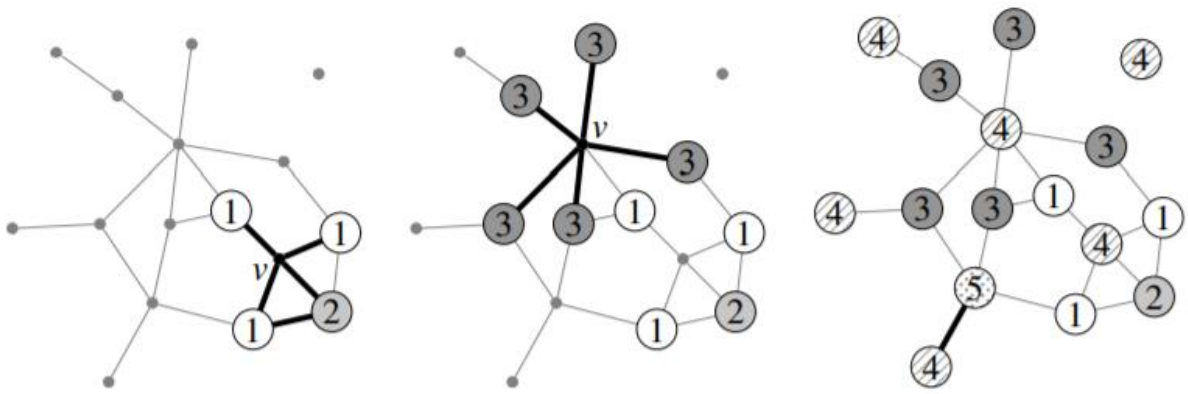
c = c + $\chi(G[N(v)])$

Ponovi *korak* **W2**

[W3] ***Idi*** *na korak* **G1** *za bojenje preostalih vrhova s bojama c, c + 1, …, c +* $\sqrt{n}$

Slika 3.7. *Pseudokod Widgersonova algoritma*

Vrijeme rada jasno je ograničeno $O(n + m)$. Za analizu broja boja, prvo moramo provjeriti korak **W2**. Budući da je *G* 3-bojiv, podgraf je isto tako inducirano s N (v) ∪ {v}. Sada, ako *G*[*N (v)*] zahtijeva 3 boje, tada *G*[*N*(*v*) ∪ {*v*}] zahtijeva 4, pa je *G*[*N*(*v*)] dvobojno i stoga je korak **W2** točan. Korak **W2** može se izvesti najviše $O(\sqrt{n})$ puta, pri čemu svaka koristi najviše dvije boje. Korak **W3** troši još $\sqrt{n}$ boja prema pohlepnom algoritmu. Widgersonov algoritam prirodno se proteže na grafove s $\chi(G) > 3$. U ovom slučaju, korak **W2** poziva Widgersonov algoritam rekurzivno za bojenje (χ (*G*)-1)-bojivih susjeda. Dobiveni algoritam koristi $O(n^{1-1/(1-\chi(G))})$ boja. Prikaz izvođenja Widgersonova algoritma nalazi se na slici 3.8..[6]



Slika 3.8. *Prikaz rada Widgersonova algoritma* [6]

3.2.6. Algoritam kontrakcije (engl. *Contraction*)

Za dani graf *G*, ovaj algoritam će vratiti niz koeficijenata (a_0, a_1, …, a_n) kromatskog polinoma P(*G*, *q*) = $\sum_{i=0}^{n} a_i q^i$, prikaz pseudokoda ovoga algoritma nalazi se na slici 3.9.. [6]

[C1] ***Ako*** *G nema bridova* ***vrati*** *niz koeficijenata (0, 0, …, 1) koji odgovara polinomu* $P(G, q) = q^n$

[C2] ***Odaberi*** *brid e*

$G' = G/e$

$G'' = G - e$

Idi *na korak* **C1** *za P(G', q) i P(G'', q)*

Vrati *polinom P(G', q) – P(G'', q)*

Slika 3.9. *Pseudokod algoritma kontrakcije*

U koraku **[C2]** prilikom poziva rekurzije će nastati niz (a'_0, a'_1, …, a'_n) za *P*(*G'*, *q*) i niz (a''_0, a''_1, …, a''_n) za *P*(*G''*, *q*), a kao rezultat će vratiti niz ($a'_0 - a''_0$, $a'_1 - a''_2$, …, $a'_n - a''_n$). [6]

Za analizu vremena izvođenja neka je *T*(*r*) broj izvršavanja koraka **C1** za grafove s *n* vrhova i *m* rubova, gdje je $r = n + m$. Konstruirana dva grafa u koraku **C2** imaju veličinu $n - 1 + m - 1 = r - 2$ i $n + m - 1 = r - 1$ pa *T* zadovoljava $T(r) = T(r - 1) + T(r - 2)$. Ovo je dobro poznato ponavljanje s rješenjem $T(r) = O(\phi^r)$, gdje je $\phi = \frac{1}{2}(1 + \sqrt{5})$ zlatni omjer. Dakle, ovaj algoritam zahtijeva $\phi^{n+m}\text{poly}(n) = O(1.619^{n+m})$ vrijeme. [6]

3.2.7. Dinamičko programiranje (engl. *Dynamic programming*)

Za dani graf *G*, ovaj algoritam izračunava tablicu *T* s $T(W) = \chi(G[W])$ za svaki $W \subseteq V$, prikaz pseudokoda ovoga algoritma nalazi se na slici 3.10.. [6]

[D1] ***Za svaki*** $W \subseteq V$ ***stvori*** *tablicu T(W)*

$T(\emptyset) = 0$

[D2] ***Navedi*** *podskupove vrhova* $W_1, W_2, \ldots, W_{2n} \subseteq V$

Idi *na korak* **D3**

Završi

[D3] $T(W) = 1 + \min T(W \setminus S)$ *gdje se minimun uzima nad svim nepraznim nezavisnim skupovima S u G[W]*

Slika 3.10. *Pseudokod algoritma dinamičkog programiranja*

Redoslijed podskupova u glavnoj petlji u koraku **D2** osigurava rukovanje svakim skupom prije bilo kojeg od njegovih nadskupova. Konkretno, sve vrijednosti $T(W \setminus S)$ potrebne u koraku **D3** će prethodno izračunati, pa je algoritam dobro definiran. Minimiziranje u koraku **D3** provodi se iteracijom nad svim $2^{|W|}$ podskupovima W. Dakle, ukupno vrijeme izvođenja ovog algoritma unutar je polinomijalnog faktora od $\sum_{W \subseteq V} 2^{|W|} = \sum_{k=0}^{n} \binom{n}{k} 2^k = 3^n$. [6]

Može se primijetiti da se za S može pretpostaviti da je maksimalni neovisni skup, to jest, nije pravi podskup nekog drugog neovisnog skupa. Da bi se to dokazalo, neka je f optimalno bojanje, razmatra se razred boje $S = f - 1$. Ako S nije maksimalan, tada se više puta odabere vrh v koji nije susjedan sa S i postavi $f(v) = 1$. [6]

Razmatranjem disjunktne unije $\frac{1}{3}k$ trokuta, vidimo da postoje grafovi s k vrhova s $3^{\frac{k}{3}}$ maksimalno neovisnih skupova. Poznato je da je ovo također gornja granica te da se maksimalni neovisni skupovi mogu nabrojati unutar polinomijalnog faktor te granice. Ova teorija se sada može primijeniti na koraku **D3** te je njegova izmjena prikazana na slici 3.11.. [6]

[D3'] *T(W) = 1 +* min*T(W \ S) gdje se minimun uzima nad svim maksimalnim nezavisnim skupovima S u G[W]*

Slika 3.11. *Pseudokod nadopunjenog koraka **D3***

Sada će ukupno vrijeme izvođenja algoritma dinamičkog programiranja biti opisano polinomijalnim faktorom $\sum_{k=0}^{n} \binom{n}{k} 3^{\frac{k}{3}} = (1 + 3^{\frac{1}{3}})^n = O(2.443^n)$. Ovo se dugo vremena smatrao najbržim algoritmom za pronalazak polinomijalnog broja. [6]

3.2.8. Lawlerov algoritam (engl. *Lawler's algorithm*)

Za dani graf G, ovaj algoritam boja graf s 3 boje, ako on postoji. Pseudokod ovoga algoritma prikazan je na slici 3.12.. [6]

[L1] ***Za svaki*** *maksimalni neovisno skup S od G* ***idi*** *na korak* **L2**

[L2] ***Idi*** *na korak* **B1** *za traženje bojenja f : V | S → {1, 2} od G – S*

Ako *bojenje postoji*

| ***Za svaki*** *v ∈ S*

| | *f(v) = 3*

Završi *s f kao razultat*

Slika 3.12. *Pseudokod Lawlerova algoritma*

Za izvođenje Lawlerova algoritma može se promatrati samo korak **L2** koji sadrži $3^{\frac{n}{3}}$ ponavljanja. Tako da se ovaj algoritam izvršava u vremenu $3^{\frac{n}{3}}\text{poly}(n) = O(1.442^n)$. [6]

3.2.9. Algoritam isključivanja-uključivanja (engl. *Inclusion–exclusion*)

S obzirom na graf G i cijeli broj $q \geq 1$, ovaj algoritam određuje može li graf biti q-obojiv, prikaz pseudokoda nalazi se na slici 3.13.. [6]

[I1] $g(\emptyset) = 0$

Za svaki *neprazni podskup* $W \subseteq V$

  Odaberi $v \in W$

  $g(W) = g(W \setminus \{v\}) + g(W \setminus N[v]) + 1$

[I2] ***Ako je*** $\sum_{W \subseteq V}(-1)^{|V \setminus W|}(g(W))^q > 0$

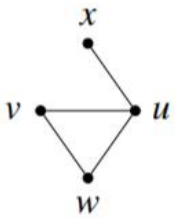
  Ispiši *'Da'*

Inače

  Ispiši *'Ne'*

Slika 3.13. *Pseudokod algoritma isključivanja-uključivanja*

U koracima **I1** i **I2** potrebno je vrijeme $2n$ poly(n), a algoritam zahtijeva tablicu s $2n$ unosi. Slika 3.14. prikazuje proračune algoritma isključivanja-uključivanja na malom grafu za $q = 2$ i $q = 3$, s $a_q(W) = (-1)^{|V \setminus W|}(g(W))^q$. Zbroj unosa u stupac a_2 je 0 pa nema 2-bojenja. Zbroj unosa u stupcu a_3 je 18, dakle postoje 3-bojenja. Način na koji algoritam isključivanja-uključivanja radi prikazan je na slici 3.14.. [6]



W	g	a_2	a_3	W	g	a_2	a_3
$\emptyset$	0	0	0	$\{v,w\}$	2	4	8
$\{u\}$	1	−1	−1	$\{v,x\}$	3	9	27
$\{v\}$	1	−1	−1	$\{w,x\}$	3	9	27
$\{w\}$	1	−1	−1	$\{u,v,w\}$	3	−9	−27
$\{x\}$	1	−1	−1	$\{u,v,x\}$	4	−16	−64
$\{u,v\}$	2	4	8	$\{u,w,x\}$	4	−16	−64
$\{u,w\}$	2	4	8	$\{v,w,x\}$	5	−25	−125
$\{u,x\}$	2	4	8	V	6	36	216

Slika 3.14. *Prikaz rada algoritma isključivanje-uključivanje* [6]

Uz male izmjene, ovaj algoritam može raditi i pri rješavanju drugih problema bojenja kao što su kromatski polinom i bojenje popisa, također u vremenu i prostoru $2n$ poly(n). Trenutno je

ovo najbrži poznati algoritam za takve probleme. Za kromatski polinom, prostor se može smanjiti na $O(1{,}292n)$, uz održavanje $2n$ poly(n) vremena rada. [6]

3.2.10. Vizingov algoritam (engl. *Vizing's algorithm*)

Za dani graf G, ovaj algoritam će pronaći bojenje bridova s najviše $\Delta(G) + 1$ boja u vremenu $O(nm)$, prikaz pseudokoda ovoga algoritma nalazi se na slici 3.14.. [6]

[V1] ***Poredaj*** *proizvoljno bridove* $e_1, e_2, \ldots, e_m$

$i = 0$

[V2] $i = i + 1$

Ako je $i = m + 1$ ***onda***

| ***Završi***

Inače

| $vw = e_i$

[V3] ***Ako je*** *boja c slobodna za v i w* ***onda***

| $f(vw) = c$

| ***Idi*** *na korak* **V2**

[V4] $w_0 = w$

Odaberi *slobodnu boju u* w_0 *i* ***nazovi ju*** *1*

Neka je vw_1 *povezan s v koji je obojen u 1*

[V5] ***Odaberi*** *slobodnu boju u* w_1 *i* ***nazovi ju*** *2*

Ako je *2 slobodna u v* ***onda***

| $f(vw_0) = 1$

| $f(vw_1) = 2$

| ***Idi*** *na korak* **V2**

Inače

| *Neka je* vw_2 *povezan s v koji je obojen u 2*

| $r = 2$

[V6] ***Odaberi*** *slobodnu boju u* w_r *i* ***nazovi ju*** $r + 1$

Ako je $r + 1$ *slobodna u v* ***onda***

| $f(vw_r) = c_{r+1}$

| ***Idi*** *na korak* **V2**

Inače

Neka je vw_{r+1} povezan s v koji je obojen u r + 1

Ako se *svaka boja 1, 2, ..., r pojavljuje oko w_{r+1}* ***onda***

r = r + 1

Ponovi *korak* **V6**

[V7] *Neka j ∈ {1, 2, ..., r} bude slobodna boja u w_{r+1} i neka 0 bude slobodna boja u v koja je drugačija od j*

Konstruiraj *dva {0, j}-bojiva puta P_j i P_{r+1} iz w_j i w_{r+1} s naizmjeničnim bojama 0, j, 0, j, ...*

Neka je k = j ili k = r + 1 tako da P_k ne završava u v

[V8] ***Promijeni*** *boju brida na P_k izmjenom 0 i j*

Prebaci se *u nižu razinu s k*

Promijeni *boju $f(vw_k) = 0$*

Idi *na korak* **V2**

Slika 3.14. *Pseudokod Vizingova algoritma*

Prilikom određivanja vremena izvođenja ovoga algoritma, bitno je primijetiti da se korak **V6** najviše ponavlja *v* puta te algoritam na kraju mora napustiti taj korak. Korak **V7** zahtijeva najviše vremena potrebnog za izvođenje.{0, j} -puta se može konstruirati u vremenu $O(n)$ ako za svaki vrh treba održavati tablicu upadnih bridova indeksiranu po boji. Dakle, ukupno vrijeme rada Vizingova algoritam je $O(mn)$. [6]

3.2.11. Algoritam metropolis (engl. *Metropolis*)

Za dani graf *G* s najvećim stupnjem Δ i *q*-bojenjem f_0 za $q > 4\Delta$, ovaj algoritam nalazi jednoličnu slučajnu *q*-boju f_T u polinomijalnom vremenu, prikaz pseudokoda ovoga algoritma nalazi se na slici 3.15.. [6]

[M1] $T = = [qn \ln 2n/(q - 4\Delta)]$.

Za svaki $t = 1, 2, \ldots, T$

Idi *na korak* **M2**

Završi

[M2] ***Odaberi*** *nasumično vrh* $v \in V$ *i boju* $c \in \{1, 2, \ldots, T\}$

$f_t = f_{t-1}$

Ako se *c ne pojavljuje među v-ovim susjedima* ***onda***

$f_t(v) = c$

Slika 3.15. *Pseudokod algoritma metropolis*

Početna boja f_0 će se odrediti u polinomijalnom vremenu jer je $q > \Delta + 1$. Da bi se vidjelo kako izbor početne boje f_0 nema utjecaj na rezultat f_T, razmatraju se dvije različite početne boje f_0 i f_0', zatim se izvodi algoritam metropolis na obje, koristeći iste slučajne izbore za v i c u svakom koraku. [6]

Neka $d_t = |\{v : f_t(v) \neq f'_t(v)\}|$ bude broj neobojenih vrhova nakon t izvršavanja koraka **M2**. Svaki korak može promijeniti samo jedan vrh pa je $|d_t - d_{t-1}| = 1$ ili 0 ili -1. $d_t = d_{t-1} + 1$ će biti samo ako je $f_{t-1}(v) = f'_{t-1}(v)$, ali pritom mora biti $f_t(v) \neq f'_t(v)$, dakle točno jedan od dva procesa odbija promjenu boje. Konkretno v mora imati susjeda w s $c = f_{t-1}(v) \neq f'_{t-1}(v)$ ili $f_{t-1}(v) \neq f'_{t-1}(v) = c$. Postoje d_{t-1} izbori za w, stoga $2\Delta d_{t-1}$ izbori za c i v. Slično, postoje $d_t = d_{t-1} - 1$ samo ako se $f_{t-1}(v) \neq f'_{t-1}(v)$ i c ne pojavljuju u susjedstvu od v u $f_{t-1}(v)$ ili $f'_{t-1}(v)$. [6]

Stoga se očekivana vrijednost d_t može ograničini na sljedeći način:

$$\mathbf{E}[d_t] \leq \mathbf{E}[d_{t+1}] + \frac{(q - 2\Delta)\mathrm{E}[d_{t+1}]}{qn} - \frac{2\Delta \mathrm{E}[d_{t+1}]}{qn} = \mathbf{E}[d_{t+1}] \left(\frac{q-4\Delta}{qn}\right). \qquad (3\text{-}1)$$

Poništavajući ovaj argument koristeći $d_0 \leq n$ dobije se:

$$\mathbf{E}[d_t] \leq n \left(1 - \frac{q-4\Delta}{qn}\right)^T \leq n \exp\left(-\frac{T(q-4\Delta)}{qn}\right) \leq n \exp(-\ln 2n) \leq \frac{1}{2}. \qquad (3\text{-}2)$$

Markovljevom nejednakošću, a budući da je d_T nenegativan cijeli broj, može se zaključiti:

$\Pr(f_T = f'_T) = \Pr(d_T = 0) \geq 1 - \Pr(d_T \geq 1) \geq 1 - \mathbf{E}[d_T] \geq \frac{1}{2}$. [6] (3-3)

Prema ovim izračunima vrijeme potrebno za izvođenje algoritma metropolis je $O(n \log n)$. [6]

3.2.12. Algoritam nasummičnog zaokruživanja vektorskog bojenja (engl. *Randomized rounding of vector colouring*)

Za dani 3-kromatski graf G s najvećim stupnjem Δ, ovaj algoritam nalazi q-bojenje u polinomijalnom vremenu, gdje je $\mathbf{E}[q] = O(\Delta^{0,681} \log n)$ očekivana veličina q, a pseudokoda ovoga algoritma prikazan je na slici 3.16.. [6]

[R1] $\varepsilon = 2 \cdot 10^{-5}$

Izračunaj *vektor $(3 + \varepsilon)$-bojenja pomoću poludefiniranog programiranja*

Neka je $\alpha \geq \arccos(-1/(2 + \varepsilon))$ minimalni kut u radijanima između susjednih vrhova

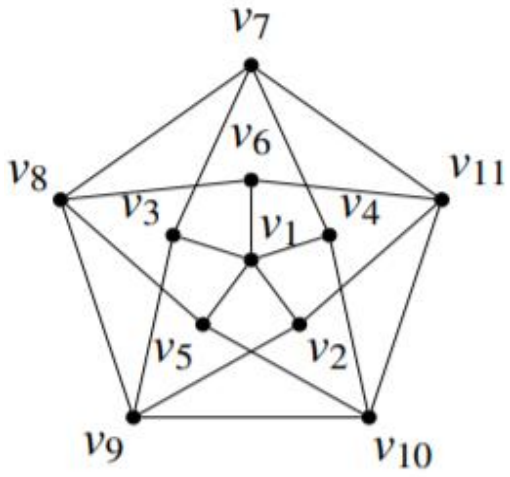
[R2] $r = [\log_{\pi/(\pi-\alpha)}(2\Delta)]$

Konstruiraj *r slučajnih hiperravnina u $\mathbb{R}^n$*

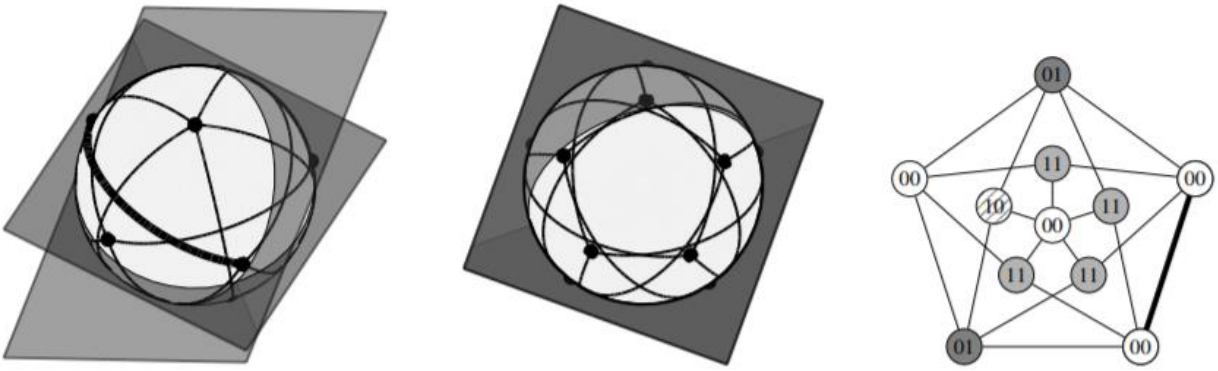
Za svaki *vrh v, neka je $f(v)$ binarni broj $b_r\, b_{r-1} \ldots b_1$, gdje je $b_i = 1$ ako i samo ako je $x(v)$ na pozitivnoj strani i -te hiperravnine H_i*

Slika 3.16. *Pseudokod algoritma nasumičnog zaokruživanja vektorskog bojenja*

Slika 3.18. prikazuje ponašanje algoritma nasumičnog zaokruživanja vektorskog bojenja na vektorskom bojenju s tri boje Grötzschovog grafa sa slike 3.17. Dvije hiperavnine odvajaju vrhove u četiri dijela. Rezultat bojenje vrhova bojama od $\{0,1\}^2$ prikazan je desno. U ovom primjeru skup M monokromatskih rubova, određen u koraku **M3**, sadrži samo jedan rub $v_{10}v_{11}$, na slici 3.18. označen podebljanom linijom. [6]



Slika 3.17. *Grötzschov graf* [6]



Slika 3.18. *Prikaz rada algoritma nasumičnog zaokruživanja vektorskog broja* [6]

U tablici 3.1. prikazani su svi postojeći algoritmi s njihovim vremenskim složenostima.

Algoritam	**Vremenska složenost**
Iscrpno pretraživanje	q^n poly(n)
Pohlepno bojenje	$O(n + m)$
Bipartitni algoritam	$O(n + m)$
Ograničenje palete	1.5^n poly(n)
Widgersonov algoritam	$O(n + m)$
Algoritam kontrakcije	$O(1.619^{n+m})$
Dinamičko programiranje	3^n poly(n)
Lawlerov algoritam	$O(1.443^n)$
Algoritam isključivanja-uključivanja	2^n poly(n)
Vizingov algoritam	$O(nm)$
Algoritam metropolis	poly(n)
Algoritam nasumičnog zaokruživanja vektorskog broja	poly(n)

Tablica 3.1. *Prikaz svih algoritama s vremenskim složenostima*

4. RAZVOJNO OKRUŽENJE I KORIŠTENE TEHNOLOGIJE

Za programsku realizaciju odabranog algoritma odabrana je Windows Forms aplikacija te je za njenu izradu korišteno Microsoft Visual Studio 2019 razvojno okruženje, programski jezik C# i .NET programski okvir (engl. *Framework*).

4.1. Microsoft Visual Studio 2019

Microsoft Visual Studio 2019 je integrirano razvojno okruženje koje je proizvela tvrtka Microsoft, prva verzija ovog razvojnog okruženja je puštena u korištenje 4. prosinca 2018. godine. Ovo razvojno okruženje se koristi za razvoj računalnih programa, web stranica, web aplikacija te mobilnih aplikacija te podržava 36 različitih programskih jezika. [7]

Glavna obilježja Microsoft Visual Studioa 2019 su uređivač koda (engl. *Code editor*), ispravljač pogrešaka (engl. *Debugger*) te dizajner (engl. *Designer*). Uređivač koda služi za pisanje varijabli, metoda, funkcija i petlji koje opisuju rad ranije spomenutih programa, aplikacija te web stranica, također sadrži opciju IntelliSense za sve podržane jezike koja programeru olakšava pisanje koda na način da ponudi prijedloge za dovršetak naziva naredbe, varijable, metode, funkcije, klase i slično. Ispravljač pogrešaka služi za prepoznavanje pogrešaka u napisanom kodu te njihovo otklanjanje za sve progamske jezike koje podržava Microsoft Visual Studio 2019. Pomoću dizajnera se uređuje grafičko sučelje željenog proizvoda. Web designer, Class designer, Windows Forms Designer, Data designer te mnogi drugi su alati koje sadrži dizajner. [7]

4.2. Programski jezik C#

Programski jezik C# je objektno orijentirani jezik te zajedno sa C i C++ čini obitelj C programskih jezika. On je Microsoftov proizvod koji je dio njihove .NET inicijative, nastao je 2002. godine, a zamišljen je kao jednostavan i moderan jezik. C# se može koristiti za izradu Windows, web i mobilnih aplikacija, ali se također može koristiti i u SQL Server za obavljanje pozadinskog rada baza podataka. [8]

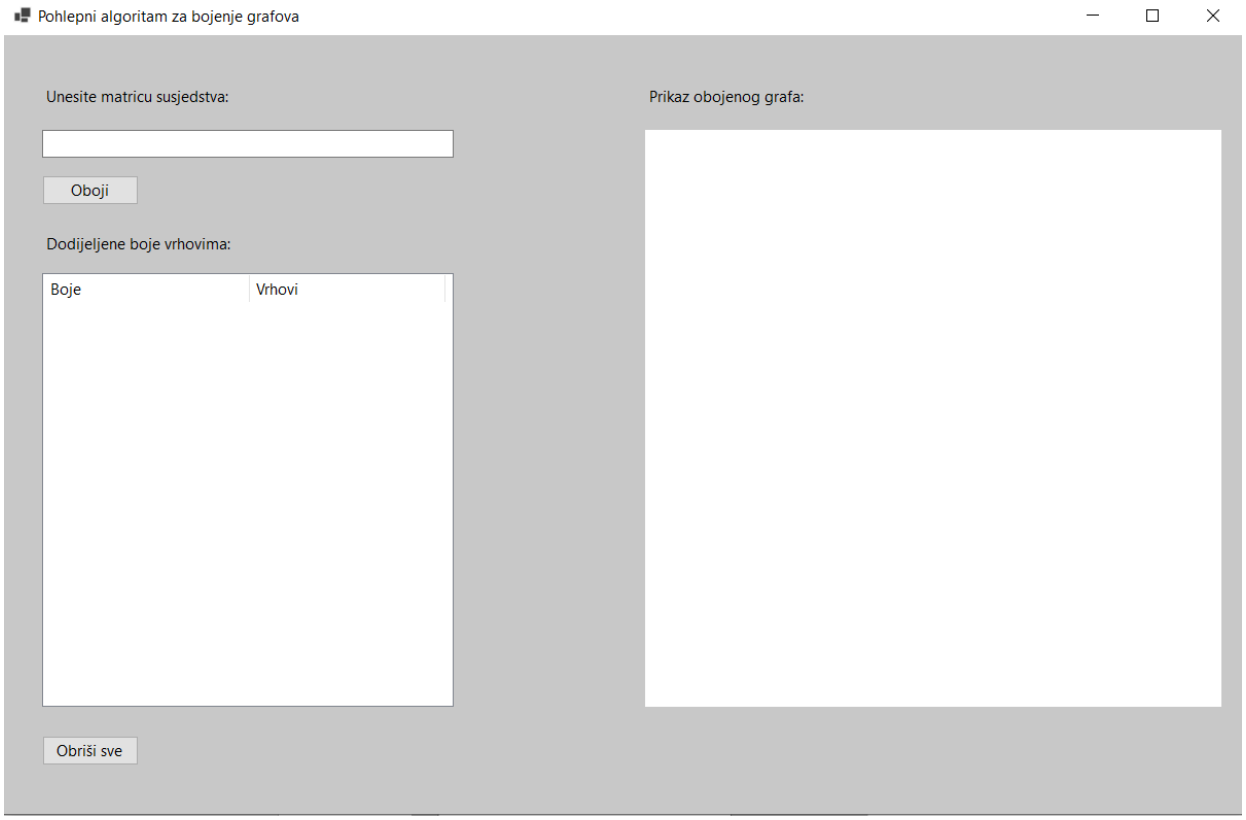
4.3. Windows Forms

Windows Forms je dio .NET programskog okvira, a koristi se kao platforma za izradu aplikacija namijenjenim stolnim i prijenosnim računalima te tabletima. Vizualni elementi koje sadrži Windows Forms aplikacija nalaze se u Windows Forms klasi te omogućuju izmjenu boja, veličina, fontova te lokaciju elemenata korisničkog sučelja, ali i događaje (engl. *Events*) poput klikanja ili povlačenja/ispuštanja (engl. *Drag/drop*). Windows Forms podržava i osnovne

elemente Windows sustava poput gumbova, tekstualnih okvira, okvira za prikaz slika i mnoge druge alate. [9]

5. PROGRAMSKA REALIZACIJA

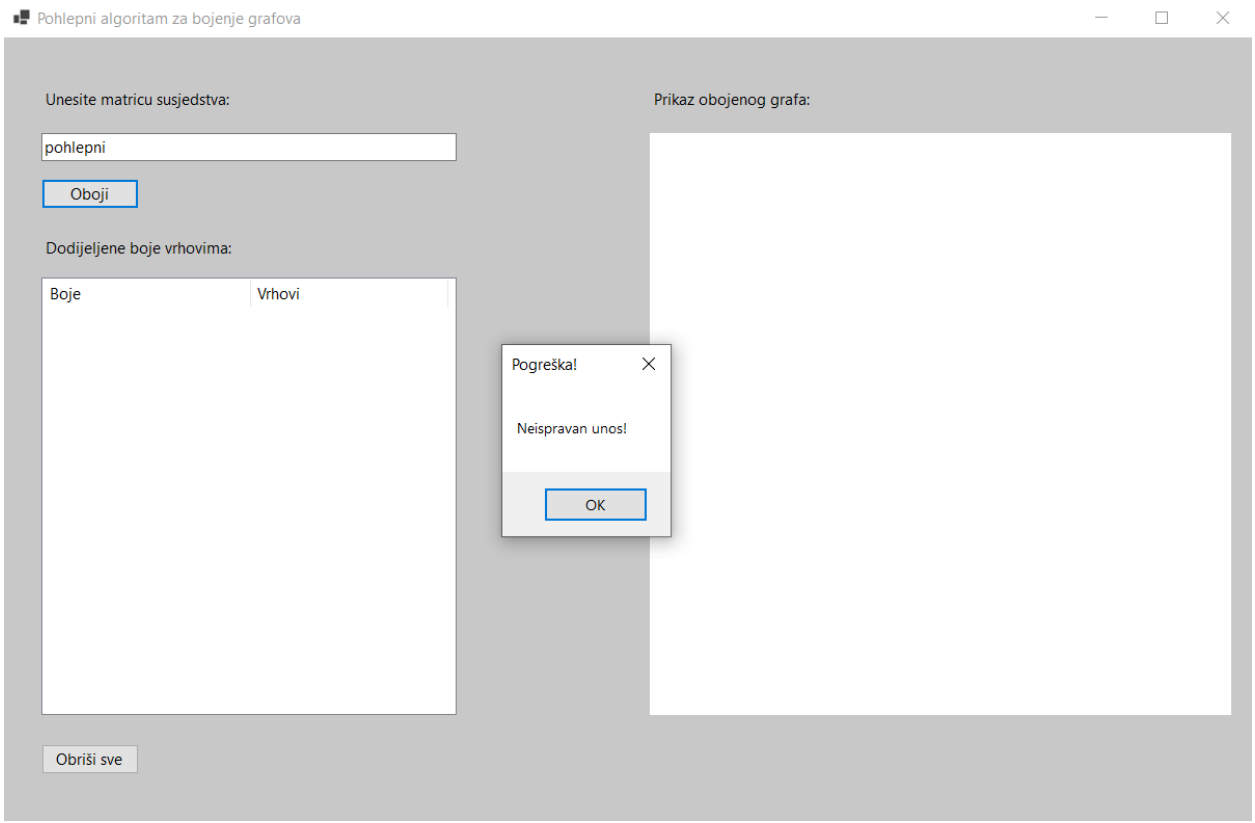
Za izradu programskog rješenja proizvoljno je odabran algoritam pohlepnog bojenja te je također proizvodno odabran programski jezik i razvojno područje koji su u ranijem poglavlju objašnjeni. Aplikacija je nazvana Pohlepni algoritam za bojenje grafova, a njezin početni izgled nalazi se na slici 5.1..



Slika 5.1. *Početni izgled aplikacije*

Aplikacija se sastoji od tekstualnog okvira (engl. *TextBox*) u koji se unosi matrica susjedstva, od dva gumba (engl. *Button*), gumb *Oboji* te gumb *Obriši sve*, zatim okvira za grafički prikaz (engl. *PictureBox*) obojenog grafa te okvira za prikaz liste (engl. *ListView*) boje i vrhova koji su određenom bojom obojeni.

Klikom na gumb *Oboji* pokreće se rad programa te on prvo provjerava odgovara li unos iz tekstualnog okvira matrici, ukoliko unos ne odgovara matrici program će stati i pokazat će se prozor s pripadajućom porukom kao na slici 5.2..



Slika 5.2. *Prikaz poruke o neispravnom unosu*

Program će prepoznati ispravan unos ako je matrica susjedstva napisana unutar uglatih zagrada te ako su redci matrice odvojeni točkom sa zarezom, npr. [0 1 1 1; 1 0 0 1; 1 0 0 0; 1 1 0 0]. Nakon što program prepozna da je unos matrice ispravan ispituje je li matrica kvadratna te sadrži li graf kojim se opisuje matrica petlje, u slučaju da matrica nije kvadratna ili da graf sadrži petlje također će se zaustaviti program i pojaviti prozor s obavijesti o tome.

Kada su svi prethodni uvjeti zadovoljeni za svaki se vrh grafa pokreće algoritam pohlepnog bojenja, programski kod kojim je algoritam napisan nalazi se na slici 5.3.

```
List<int>[] pohlepni_algoritam(int vrh, int[,] graf, List<int>[] odabrane_boje)
{
    bool obojen = false;
    for (int j = 0; !obojen && j < odabrane_boje.Length; ++j)
    {
        bool prepreka = false;
        for (int i = 0; !prepreka && i < odabrane_boje[j].Count; ++i)
        {
            int trenutni_vrh = odabrane_boje[j][i];
            if (graf[vrh, trenutni_vrh] != 0)
            {
                prepreka = true;
            }
        }
        if (!prepreka)
        {
            odabrane_boje[j].Add(vrh);
            obojen = true;
        }
    }
    return odabrane_boje;
}
```

Slika 5.3. *Kod algoritma pohlepnog bojenja*

Funkcija za ulazne parametre prima vrh koji se treba obojiti, uneseni graf te niz s listom cijelih brojeva u kojem prvi indeks niza označava indeks boje te on sadrži listu s brojevima vrhova kojima je ta boja dodijeljena. U funkciji je prvo definirana varijabla *obojen* koja predstavlja zastavicu da je predani vrh uspješno obojen, zatim se *for* petljama prolazi kroz graf te se utvrđuje može li se zadani vrh obojiti *j*-tom bojom, ukoliko nema prepreka predani vrh se dodjeljuje *j*-toj boji te funkcija vraća uređeni niz s listama. Nakon prolaska kroz sve vrhove grafa i bojenja vrhova, tj. dodjeljivanja vrhova pojedinim bojama, slijedi crtanje grafa i ispis boja s kojima su vrhovi obojeni.

Kako bi se graf iscrtao na okviru za prikaz slika potrebno je odrediti koordinate na kojima će se pojedini vrh prikazati u okviru za prikaz slika. Određivanje koordinate je postignuto funkcijom *odredi_koordinate* koja se nalazi na slici 5.4.. Funkciji se predaje broj vrha za koji je potrebno odrediti koordinate te se koordinate pohranjuju u globalne nizove za x i y koordinate.

```
void odredi_koordinate(int n)
{
    for (int i = 0; i < n; i++)
    {
        double x = 200 * Math.Cos((2 * Math.PI * i) / n);
        double y = 200 * Math.Sin((2 * Math.PI * i) / n);
        x += 280;
        y += 280;
        int x_p = Convert.ToInt32(x);
        int y_p = Convert.ToInt32(y);
        x_koordinate[i] = x_p;
        y_koordinate[i] = y_p;
    }
}
```

Slika 5.4. *Funkcija za određivanje koordinata*

Kada su koordinate određene pozivaju se funkcije *nacrtaj_vrh* i *nacrtaj_brid* za crtanje vrhova grafa te bridova kojima su povezani vrhovi. Kod funkcije *nacrtaj_vrh* nalazi se na slici 5.5., dok se kod funkcije *nacrtaj_brid* nalazi na slici 5.6.. Funkcija za crtanje vrhova kao ulazne vrijednosti prima niz s listama *odabrane__boje* te ukupan broj vrhova, kada utvrdi da se u listi *j*-te boje nalazi *i*-ti vrh tada ga nacrta *j*-tom bojom koja se nalazi u popisu boja. Funkcija za crtanje bridova za ulaznu vrijednost prima uneseni graf te kada utvrdi da se između dva vrha nalazi brid, nacrta taj brid.

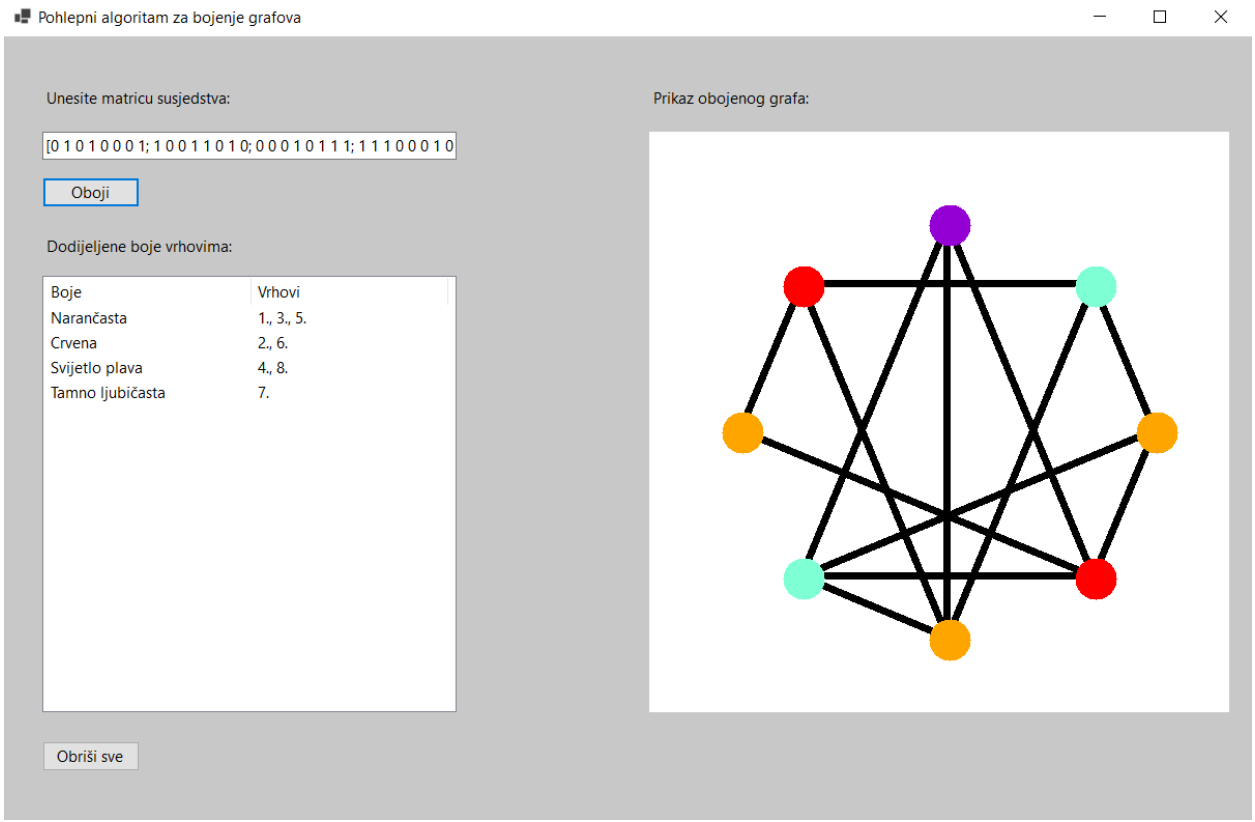
```
void nacrtaj_vrh(List<int>[] odabrane_boje, int n)
{
    for (int j = 0; j < odabrane_boje.Length; j++)
    {
        for (int i = 0; i < odabrane_boje[j].Count; i++)
        {
            for (int k = 0; k < n; k++)
            {
                if (odabrane_boje[j][i] == k)
                {
                    Pen myPen = new Pen(popis_boja[j], 20);
                    g.DrawEllipse(myPen, x_koordinate[k], y_koordinate[k], 20, 20);
                }
            }
        }
    }
}
```

Slika 5.5. *Funkcija za crtanje vrhova*

```
void nacrtaj_brid(int[,] graf)
{
    for (int i = 0; i < graf.Length; i++)
    {
        for (int j = 0; j < graf.Length; j++)
        {
            if (graf[i, j] == 1 && graf[j, i] == 1)
            {
                PointF point1 = new PointF(x_koordinate[i], y_koordinate[i]);
                PointF point2 = new PointF(x_koordinate[j], y_koordinate[j]);
                Pen blackPen = new Pen(Color.Black, 7);
                g.DrawLine(blackPen, point1, point2);
            }
        }
    }
}
```

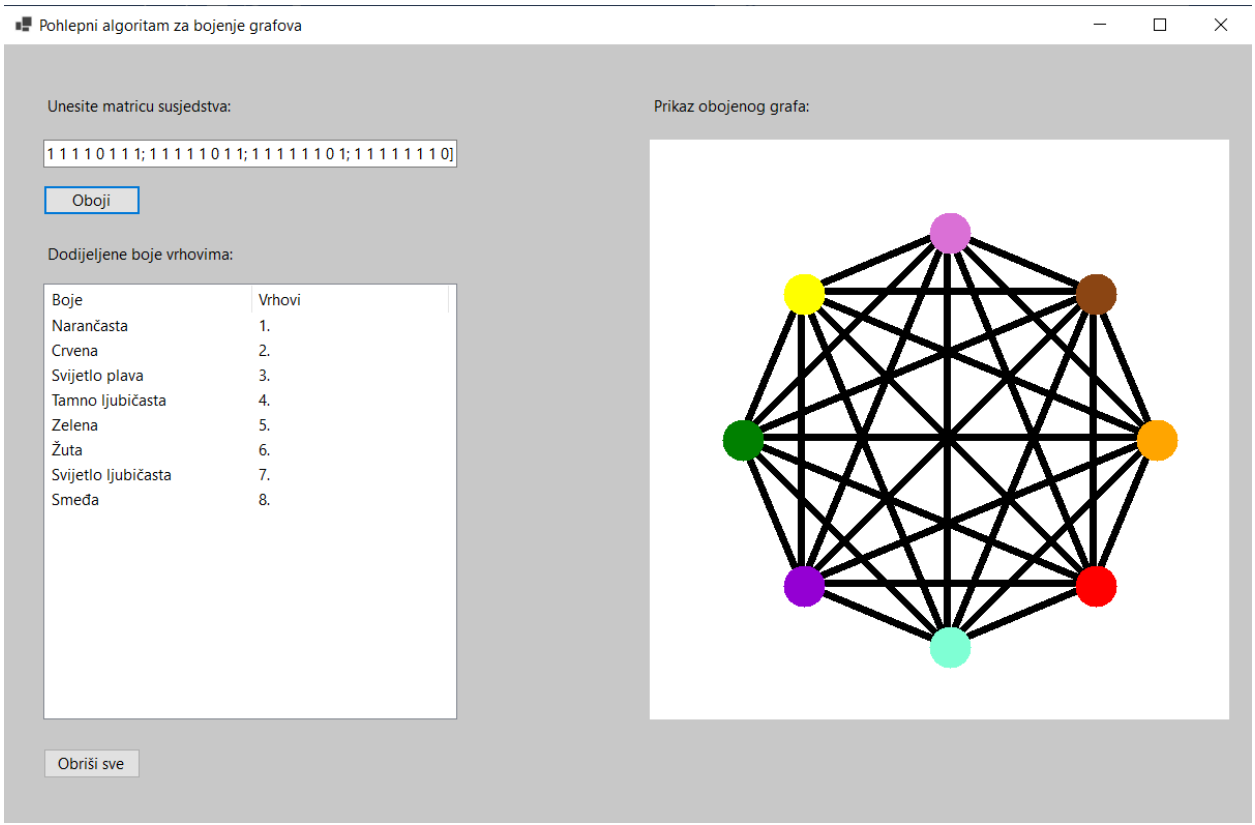
Slika 5.6. *Funkcija za crtanje bridova*

Rad aplikacije za unesenu matricu susjedstva [0 1 0 1 0 0 0 1; 1 0 0 1 1 0 1 0; 0 0 0 1 0 1 1 1; 1 1 1 0 0 0 1 0; 0 1 0 0 0 1 0 0; 0 0 1 0 1 0 0 1; 0 1 1 1 0 0 0 0; 1 0 1 0 0 1 0 0] prikazan je na slici 5.7.



Slika 5.7. *Prikaz rada aplikacije*

Na slici 5.8. nalazi se također graf s osam vrhova, ali za razliku od grafa sa slike 5.7. ovaj je potpun, odnosno svi vrhovi su međusobno povezani. Ako se usporede dobiveni rezultati bojenja vidljivo je da su na slici 5.7. pojedini vrhovi obojeni istim bojama, dok su na slici 5.8. svi vrhovi različitih boja. Razlog tomu je jer graf sa slike 5.7. nije potpun, odnosno vrhovi s istim bojama nisu povezani te prema pravilu za bojenje grafova, kromatski broj toga grafa je manji, dakle vrhovi koji nisu susjedni su obojeni istom bojom, a vrhovi sa slike 5.8. su svi međusobno povezani, stoga moraju biti obojeni različitim bojama.



Slika 5.8. *Prikaz rada aplikacije za potpuni graf*

6. ZAKLJUČAK

U ovom završnom radu cilj je bio otkriti koji sve algoritmi za bojenje grafova postoje, opisati ih te jedan od njih isprogramirati u proizvoljnom programskom jeziku.

Prije samog rješavanja problema završnog rada kako bi se daljnji sadržaj ovog rada mogao bolje razumjeti definiran je pojam grafa, vrha i brida, prikazana je razlika između usmjerenog i neusmjerenog grafa, uz to su definirane i vrste grafova te je opisano na koji način se grafovi mogu zapisati. Nakon toga se razjašnjava problematika ovoga rada, pojašnjeno je koji je cilj bojenja grafova, što je kromatski broj, kako on utječe na bojenje te na koje se načine grafovi mogu bojiti. Prikazana je kratka povijest bojenja grafova kako se Francis Guthrie našao u problemu prilikom bojenja geografske karte engleskih okruga te je za rješavanje problema prvi puta koristio tehniku bojenja grafova. Osim primjene u kartografiji navedene su i neke druge primjene za rješavanje raznih problema, a između ostaloga je opisana važnost ovog pristupa pri rješavanju problema rasporeda sati. Zatim je definiran pojam algoritma te su predstavljeni pronađeni postojeći algoritmi za bojenje grafova, rad svakog algoritma je prikazan pseudokodom te je analizirano vrijeme njegovog izvođenja, odnosno vremenska složenost.

U praktičnom dijelu je u razvojnom okruženju Microsoft Visual Studio 2019 izrađena Windows Forms aplikacija u programskom jeziku C# koja za unesenu matricu susjedstva koristeći algoritam pohlepnog bojenja oboji svaki vrh te grafički prikaže uneseni graf s obojenim vrhovima. Detaljno je objašnjen proces izvođenja algoritma te kako je postignut grafički prikaz rješenja. Aplikacija je testirana na dva različita grafa s osam vrhova, s obzirom da su grafovi, unatoč istom broju vrhova, imali različit broj bridova grafička rješenja su se razlikovala te je objašnjeno kako je došlo do razlika u dobivenim rješenjima.

Moguće ograničenje aplikacije je u prikazu rješenja, ukoliko je zadan graf koji sadrži toliko puno vrhova da se ne može prikazati na zaslonu zbog ograničene veličine samog zaslona. Aplikacija je vrlo jednostavna za korištenje, jedinu prepreku prilikom rukovanja može stvoriti unos matrice susjedstva ukoliko korisnik nije upoznat na koji se način se piše matrica, to se može riješiti unaprjeđenjem aplikacije tako da korisnik sam nacrta graf, a aplikacija samo oboji već ilustrirane vrhove.

LITERATURA

[1] J. A. Bondy and U. S. R. Murty. GRAPH THEORY WITH APPLICATIONS, J.A. Bondy and V.S.R. Muny, New York, 1976.

[2] N. Deo, GRAPH THEORY with Applications to Engineering and Computer Science, Prentice-Hall Inc., New Jersey, 1976.

[3] R. Balakrishnan and K. Ranganathan, A Textbook of Graph Theory, Springer Science+Business Media, New York, 2012.

[4] R.M.R. Lewis - A Guide to Graph Colouring Algorithms and Applications, Springer International Publishing AG Switzerland, New York, 2016.

[5] http://ijmaa.in/v5n4-f/845-849.pdf [9.7.2021.]

[6] https://thorehusfeldt.files.wordpress.com/2010/08/gca.pdf [29.6.2021.]

[7] https://en.wikipedia.org/wiki/Microsoft_Visual_Studio [13.8.2021.]

[8] https://www.keentpoint.com/tutorial/csharp/csharp-Introduction-Language-Evolution-Application-of-csharp [13.8.2021.]

[9] https://en.wikipedia.org/wiki/Windows_Forms [13.8.2021.]

SAŽETAK

Zadatak ovog završnog rada bio je istražiti postojeće algoritme za bojenje grafova te odabrati jedan i implementirati ga u proizvoljnom programskom jeziku. Prvo su opisani teorijski pojmovi vezani uz teoriju grafu kako bi se lakše razumio sam rad, zatim je objašnjeno bojenje grafova te njegova važnost u primjeni na stvarnim problemima. Nakon toga su navedeni postojeći algoritmi za bojenje grafova kao i način njihova rada te vremenska složenost, nadalje su opisane korištene tehnologije te razvojno kruženje u kojem je implementiran odabrani algoritam. Na samome kraju je odabran algoritam pohlepnog bojenja te je u programskom jeziku C# stvorena Windows Forms aplikacija. Windows Forms aplikacija radi na način da se unese matrica susjedstva željenog grafa, a ona će koristeći algoritam pohlepnog bojenja obojiti zadani graf te rješenje prikazati u grafičkom obliku.

Ključne riječi: algoritam pohlepnog bojenja, bojenje grafova, programski jezik C#, teorija grafova

ABSTRACT

Title: Graph coloring algorithms

The aim of this final paper was to investigate the existing algorithms for coloring graphs, select one and implement it in an arbitrary programming language. Firstly, theoretical concepts related to graph theory are described in order to better understand the reseach paper itself, then the coloring of graphs and their importance in dealing to real problems are explained. After that, the existing algorithms for coloring graphs are listed, as well as the way they work and the run time, then are described the used technologies and the development environment in which the selected algorithm is implemented. At the very end, the greedy coloring algorithm was selected and the Windows Forms application was created in the C# programming language. The Windows Forms application works by entering the adjacency matrix of the desired graph, and it will use a greedy coloring algorithm to color the default graph and display the solution in graphical form.

Key words: graph coloring, graph theory, greedy coloring algorithm, programming language C#

PRILOZI

Na CD-u se nalazi:

- Izvorni kod programa
- Završni rad u .pdf formatu
- Završni rad u .docx formatu