Računalna aplikacija za određivanje značajki grafa

Švenjak, Lorena

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1243

RAČUNALNA APLIKACIJA ZA ODREĐIVANJE ZNAČAJKI GRAFA

Lorena Švenjak

Zagreb, siječanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1243

RAČUNALNA APLIKACIJA ZA ODREĐIVANJE ZNAČAJKI GRAFA

Lorena Švenjak

Zagreb, siječanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 2. listopada 2023.

ZAVRŠNI ZADATAK br. 1243

Pristupnica:	**Lorena Švenjak (0119039893)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentorica:	izv. prof. dr. sc. Anamari Nakić

Zadatak: **Računalna aplikacija za određivanje značajki grafa**

Opis zadatka:

Graf G je uređeni par (V, E) konačnog nepraznog skupa vrhova V i konačnog skupa E dvočlanih podskupova skupa V, koje zovemo bridovi. Matematički pojam grafa prirodni je model različitih mreža, transportnih, električnih ili društvenih. Poznavanje značajki dobivenog grafa daje inženjerima uvid u mreže koje su u fokusu njihovog rada. U ovom radu izradit će se aplikacija za određivanje značajki grafa poput povezanosti, eulerovosti, hamiltonovosti, kromatskog broja i kromatskog indeksa. Interaktivna računalna aplikacija omogućit će korisniku unos proizvoljnog grafa te će u najkraćem mogućem vremenu izvijestiti o osnovnim značajkama grafa.

Rok za predaju rada: 26. siječnja 2024.

Sadržaj

Uvod

Teorija grafova je grana matematike koja proučava pojam grafa i njegova svojstva. Grafovi su često korisni za modeliranje sustava u kojima postoji međusobna povezanost između elemenata. Neki od primjera su dizajniranje računalnih ili prometnih mreža, optimizacija transporta, modeliranje bioloških sustava i evolucijskih odnosa, analiza tržišnih trendova… Kako bismo mogli opisati neki sustav korištenjem grafova, korisno je da možemo brzo i lako provjeriti njihova svojstva.

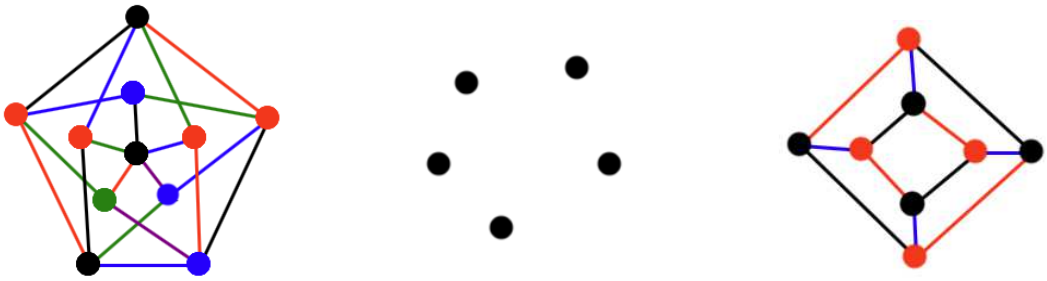
U ovom radu ćemo proučavati jednostavnost, povezanost, eulerovost, hamiltonovost, mostove, kromatski broj i kromatski indeks grafa. Prvo ćemo objasniti svaku od navedenih značajki, a zatim opisati izrađenu aplikaciju koja ih provjerava. Aplikacija je interaktivna – omogućuje korisniku unos grafa i izvještava ga o njegovim osnovnim značajkama. Na kraju ćemo dati upute za korištenje aplikacije.

1. Osnovni pojmovi

Za početak ćemo se upoznati s najvažnijim pojmovima u teoriji grafova.

Definicija 1.1. ***Graf*** *G je uređeni par (V,E), gdje je V(G) neprazni skup vrhova, a E(G) je skup bridova.*

Slika 1.1 prikazuje primjere grafova.



Grötzschov graf | Nul graf N_5 | Kocka

Slika 1.1 Primjeri grafova

Brid koji spaja različite krajeve naziva se pravi brid, dok je brid čiji se krajevi podudaraju petlja. Između krajeva mogu postojati i višestruki bridovi.

Definicija 1.2. *Neka brid $e \in E$ spaja vrhove $u, v \in V$. Tada su vrhovi u i v* ***incidentni*** *s bridom e. Vrhovi u i v tada su* ***susjedni****.*

Definicija 1.3. *Neka je $v \in V$ jedan od vrhova grafa G.* ***Stupanj*** *vrha v definiramo kao broj bridova grafa G s kojima je vrh v incidentan. Pritom se svaka petlja broji kao dva brida.*

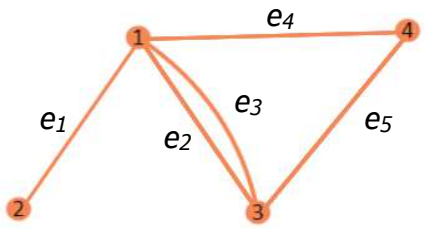
Definicija 1.4. ***Šetnju*** *u grafu definira niz vrhova i bridova poredanih naizmjenično, i to tako da je brid e_i incidentan s vrhovima v_{i-1} i v_i, $1 \leq i \leq k$, gdje je k broj vrhova u nizu. Ako su svi bridovi šetnje međusobno različiti, onda je riječ o* ***stazi****.* ***Put*** *je staza koja ima sve međusobno različite vrhove.*

Definicija 1.5. ***Ciklus*** *je zatvorena staza koja ima međusobno različite sve unutarnje vrhove (vrhove koji nisu krajnji).*

U nekim slučajevima, bit će nam korisno prikazati graf u obliku matrice. Na primjer, za rješavanje zadataka iz područja elektrotehnike, često je potrebno koristiti matricu incidencije određenog grafa.

Definicija 1.6. ***Matrica incidencije*** *M grafa G koji sadrži n vrhova i m bridova je n × m matrica, M = [m_{ij}], gdje element m_{ij} označava koliko je puta vrh v_i incidentan s bridom e_j.*

Slika 1.2 prikazuje graf za koji smo konstruirali pripadajuću matricu incidencije *M*.



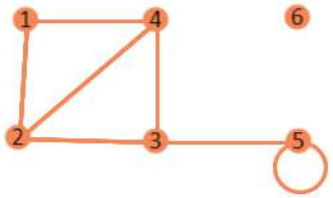
Slika 1.2 Primjer grafa

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

Pri izradi programskog rješenja za ispitivanje svojstava grafa, bilo je važno osmisliti prikladan način prikaza ulaznih podataka koje dajemo programu. Ulazni podatci moraju potpuno opisati veze između vrhova i bridova grafa. To smo postigli korištenjem matrice susjedstva.

Definicija 1.7. ***Matrica susjedstva*** *A grafa G koji sadrži n vrhova je kvadratna matrica reda n, A = [a_{ij}], gdje element a_{ij} označava koliko bridova spaja vrh v_i s vrhom v_j.*

Ako graf sadrži petlje u vrhu *i*, vrijednost elementa matrice susjedstva a_{ii} mora biti dvostruko veća od broja petlji u tom vrhu. Slika 1.3 prikazuje graf čija je matrica susjedstva *A*.



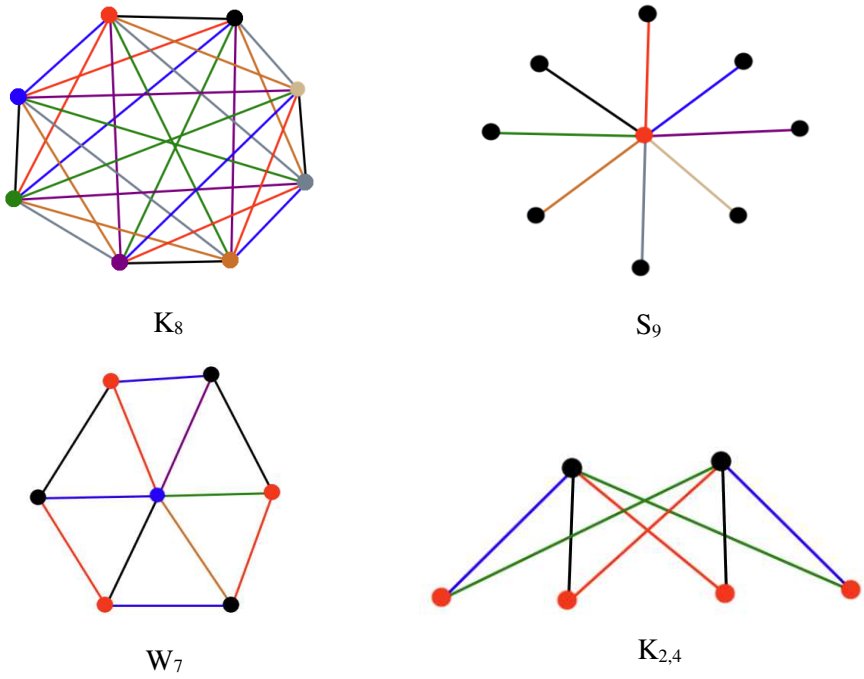
Slika 1.3 Primjer grafa

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

2. Svojstva grafa

2.1. Jednostavnost

Graf *G* je jednostavan ako nema nijednu petlju ni višestruke bridove. Ako u grafu postoji petlja ili su neka dva vrha povezana s više od jednog brida, graf nije jednostavan. Tada se radi o *općem* grafu, gdje nema ograničenja na broj petlji i bridova između istih dvaju vrhova. Primjere nekih jednostavnih grafova možemo vidjeti na slici (Slika 2.1.1).



Slika 2.1.1 Primjeri jednostavnih grafova

Potpuni graf K_n je graf u kojem su svi vrhovi međusobno povezani. Zvijezda S_n je naziv za graf u kojem postoji jedan vrh (središnji vrh) povezan s ostalim vrhovima, ali među ostalim vrhovima nema bridova. Kotač W_n je graf dobiven tako da svaki vrh iz ciklusa C_{n-1} povežemo s jednim dodatnim vrhom (središnji vrh). Na Slici 2.1.1 su prikazani K_8, S_9 i W_7. Bipartitni graf je onaj kojemu vrhove možemo razdijeliti u dva disjunktna skupa, tako da je svaki brid incidentan s jednim vrhom iz jednog te jednim vrhom iz drugog disjunktnog skupa. Na slici (Slika 2.1.1) je prikazan potpuni bipartitni

graf $K_{2,4}$, gdje brojevi u indeksu označavaju broj vrhova u jednom, odnosno drugom disjunktnom skupu.

Jednostavni graf može imati minimalno nula bridova. Tada se radi o nul-grafu, tj. grafu koji sadrži samo vrhove. Maksimalan broj bridova jednostavnog grafa je jednak broju bridova potpunog grafa. U potpunom grafu s ukupnim brojem vrhova n, svaki vrh ima stupanj $n - 1$, pa je broj bridova takvog grafa $\frac{n(n-1)}{2}$.

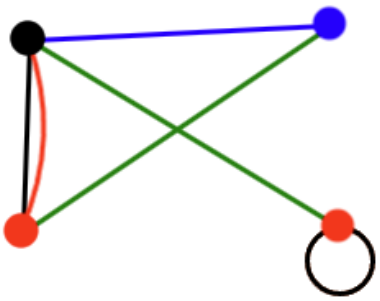
2.2. Povezanost

Definicija 2.2.1. *Graf je **povezan** ako se svaka dva vrha u grafu mogu povezati putem.*

Ako graf nije povezan, možemo ga prikazati kao uniju više grafova. Za grafove $G_1 = (V_1, E_1)$ i $G_2 = (V_2, E_2)$ koji nemaju zajedničke vrhove ni bridove, unija je definirana kao $(V_1 \cup V_2, E_1 \cup E_2)$.

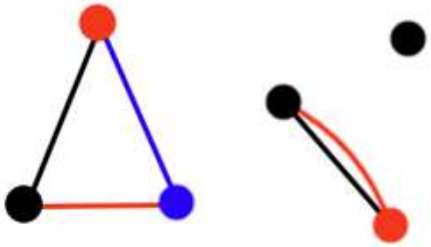
Definicija 2.2.2. ***Komponenta povezanosti** grafa G je podgraf od G koji je povezan te nije dio niti jednog većeg povezanog podgrafa od G.*

Iz prethodne definicije slijedi da je graf povezan ako ima samo jednu komponentu povezanosti. Broj komponenti povezanosti označavat ćemo sa $c(G)$. Na slici (Slika 2.2.1) vidimo primjer povezanog grafa.



Slika 2.2.1 Povezani graf

Slika 2.2.2 prikazuje primjer nepovezanog grafa s tri komponente.



Slika 2.2.2 Nepovezani graf

Definicija 2.2.3. ***Stablo*** *je povezani graf koji ne sadrži ciklus.* ***Razapinjuće stablo*** *grafa G je podgraf koji je stablo koje sadrži sve vrhove grafa G.*

Pronalaženje razapinjućih stabala grafa ima razne primjene, najviše u elektrotehnici [4]. Strujni krug možemo prikazati kao graf čiji vrhovi predstavljaju čvorove strujnog kruga, a bridovi predstavljaju trošila. Korištenjem matrice incidencije i Kirchhoffovih zakona, možemo pronaći iznose struja i napona u strujnom krugu. U nekim je problemima koristan Kirchhoffov matrični teorem o stablima [1].

Matrični teorem o stablima. *Neka graf G sadrži n vrhova i m bridova te ne sadrži petlje. Neka je M matrica incidencije grafa G. Konstruiramo matricu N tako da u svakom stupcu matrice M zamijenimo jednu jedinicu s -1. Neka je* $Q = NN^T$*. To znači da su na glavnoj dijagonali matrice Q stupnjevi vrhova. Izbacimo i-ti redak i i-ti stupac iz matrice Q i nazovimo je* Q_{ii}*. Broj razapinjućih stabala grafa označimo s T(G). Tada za svaki i = 1,…,n vrijedi*

$$T(G) = \det Q_{ii}.$$

Dokaz. Iz Binet-Cauchyjevog teorema vrijedi: $\det Q_{ii} = \sum_C \det C * \det C^T = \sum_C (\det C)^2$, gdje C prolazi po svim (n – 1) × (n – 1) podmatricama matrice N bez i-tog retka. Stupci od C čine podgraf od G s n – 1 bridova i n vrhova. Želimo dokazati da je $\det C = \pm 1$ ako ti bridovi čine stablo, inače $\det C = 0$.

Ako bridovi ne čine stablo, onda postoji komponenta koja ne sadrži i. Odgovarajući redci tih komponenata su linearno zavisni, tako da je $\det C = 0$.

Ako bridovi čine stablo, onda postoji vrh $j_1 \neq i$ stupnja 1. Označimo brid koji je jedini incidentan tom vrhu s e_1. Ako izbrišemo taj vrh i brid, dobit ćemo stablo s n – 2 brida. Sada postoji vrh $j_2 \neq$ i stupnja 1 kojemu je jedino incidentan brid e_2. Nastavimo

uklanjati vrhove i bridove tako da zadnji uklonjeni budu j_{n-1} i e_{n-1}. Konstruiramo matricu C' tako da dovodimo j_k u k-ti redak, a e_k u k-ti stupac. Matrica C' je donja trokutasta i ima elemente ±1 na glavnoj dijagonali. Slijedi da je det C = det C' = ±1.

□

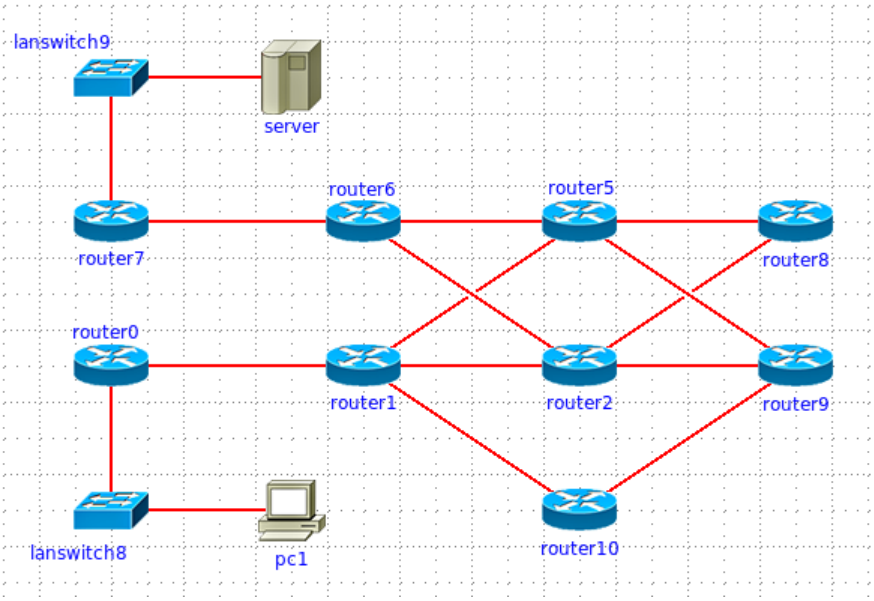
Ako rješavamo problem u kojem strujni krug možemo predstaviti potpunim grafom K_n, lako ćemo naći broj razapinjućih stabala korištenjem Cayleyevog teorema [1] .

TEOREM 1 (Cayley). *Broj $T(K_n)$ razapinjućih stabala potpunog grafa s n vrhova jednak je n^{n-2}.*

Ako graf *G* predstavlja strujni krug pri čemu su bridovi žice jediničnog otpora koje se sastaju u vrhovima, možemo koristiti Matrični teorem o stablima za pronalaženje otpora između vrhova *a* i *b*. Uzimajući u obzir Kirchhoffove zakone o struji i naponu, dolazimo do zaključka da je navedeni otpor jednak omjeru broja razapinjućih stabala koji sadrže brid *ab* i ukupnog broja razapinjućih stabala *T(G)*.

Osim u elektrotehnici, broj razapinjućih stabala grafa može biti koristan za kontroliranje gužve u prometu. Zamislimo prometne ceste u određenom gradu kao mrežu. Ako sve ceste u tom gradu tvore samo jednu osnovnu strukturu povezanu stablom, tada može doći do svakodnevnih gužvi u prometu. Dodavanje novih cestovnih veza stvara dodatne rute za putovanje od jednog do drugog mjesta, povećavajući broj različitih načina kako se ljudi mogu kretati gradom. To smanjuje gužve jer ljudi imaju više opcija za putovanje.

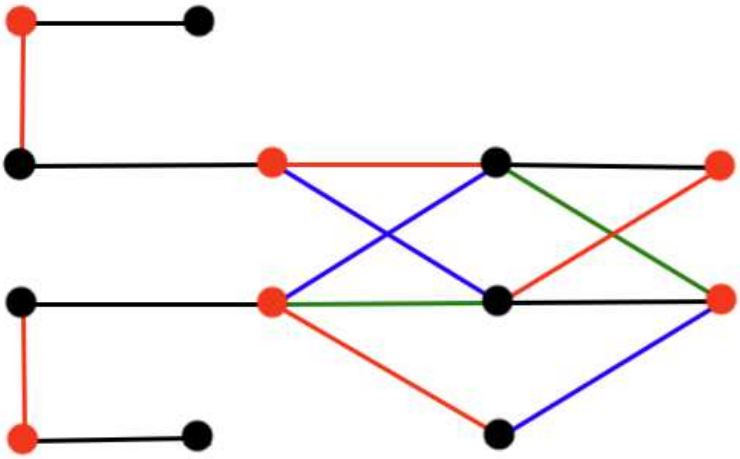
Sličan primjer je internetski promet. Internet je velika mreža koja povezuje različite uređaje. Kako bismo vidjeli neku web stranicu na svom uređaju, podaci putuju kroz različite poslužitelje i usmjeritelje dok ne stignu do nas. Na slici (Slika 2.2.3) vidimo jednostavan primjer više mogućih ruta za slanje podataka od poslužitelja (*server*) do računala (*pc1*).



Slika 2.2.3 Prikaz modela internetske mreže

Uređaje možemo prikazati kao vrhove grafa povezane bridovima koji predstavljaju žice (Slika 2.2.4). Želimo pronaći optimalnu rutu kojom podatci mogu putovati od poslužitelja do računala. Zamislimo da je naš model dio neke veće mreže, tj. da na usmjeritelje mogu dolaziti i podatci s nekih drugih poslužitelja. Upravljanje rutama na internetu obično ovisi o raznim čimbenicima, uključujući stanje mreže, opterećenje pojedinih ruta, dostupnost i moguće smetnje. Ako je jedna ruta zagušena ili ima problema, mrežni uređaji mogu automatski preusmjeriti promet na drugu rutu kako bi izbjegli zastoje i osigurali kontinuitet usluge. Ovaj proces se naziva dinamičko usmjeravanje. Postoje razni algoritmi usmjeravanja koji se koriste se za određivanje najbolje rute u mrežama. Ako se podatci nađu u situaciji gdje bi ruta kojom bi inače prošli stvorila ciklus, algoritam bi ih trebao usmjeriti na drugu rutu koja ne uključuje ciklus. Za mrežu s prethodne slike (Slika 2.2.3), zadatak algoritma se može svesti na pronalazak stabla koje je podgraf grafa (Slika 2.2.4), takav da sadrži vrhove koji predstavljaju poslužitelj i računalo. Algoritam bi trebao usporediti više takvih stabala i naći ono koje predstavlja optimalnu rutu.

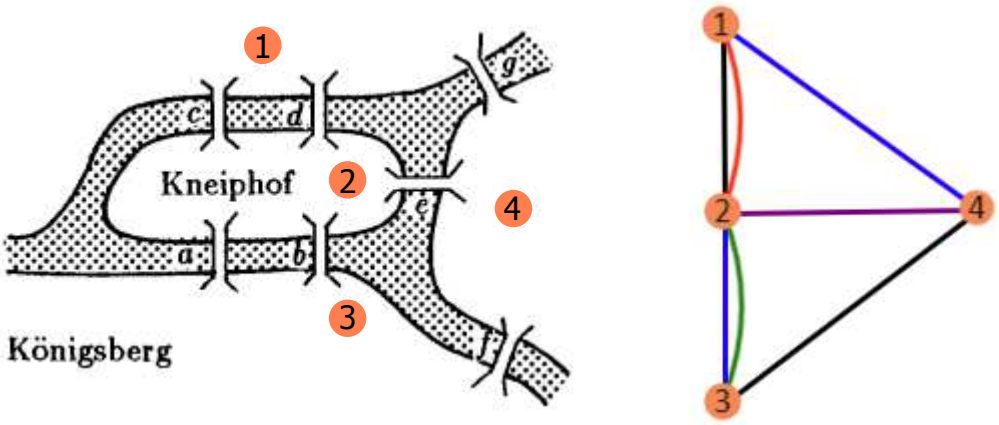
Ukratko, broj različitih ruta ili mogućih razapinjućih stabala može utjecati na to koliko će biti gužvi. Što je njihov broj veći, manje će vjerojatno doći do zagušenja.



Slika 2.2.4 Graf modela internetske mreže

2.3. Eulerovost i semi-eulerovost

Jedan od najpoznatijih problema u teoriji grafova je problem mostova u Königsbergu [2]. Današnji Kalinjingrad bio je podijeljen rijekom Pregel na dva dijela, a na rijeci i njenim pritocima bilo je sedam mostova koji su povezivali dva dijela kopna i dva otoka. (Slika 2.3.1 [7]).



Slika 2.3.1 Skica mostova u Königsbergu i prikaz u obliku grafa

Pitanje je glasilo: „Je li moguće proći preko svakog mosta točno jednom i vratiti se na početnu točku?“ Leonhard Euler, jedan od najvećih svjetskih matematičara, bavio se

ovim pitanjem u 18. stoljeću. Njegov pristup rješavanju problema prikazan je na prethodnoj slici (Slika 2.3.1 [7]). Predstavio je kopno i otoke kao vrhove, a mostove kao bridove grafa.

Definicija 2.3.1. *Graf je **eulerovski** ako sadrži eulerovsku stazu (zatvorenu stazu koja prolazi svakim bridom točno jednom).*

Rješavanje problema mostova u Königsbergu svodi se na provjeru eulerovosti grafa. Možemo pokušati pronaći eulerovsku stazu na grafu s prethodne slike. Kojim god redoslijedom prolazimo bridove, ne možemo je naći. Zaključujemo da graf nije eulerovski. Dakle, nije moguće prijeći preko svakog mosta točno jednom i vratiti se na početnu točku. Umjesto traženja eulerovske staze, Euler je otkrio lakši način rješavanja ovog problema. Dokažimo sljedeći teorem [3].

TEOREM 2. *Povezani graf G je eulerovski ako i samo ako je svaki njegov vrh parnog stupnja.*

Dokaz. $\Rightarrow$: Ako graf ima Eulerovu stazu, svaki put kada uđemo u neki vrh grafa, iz njega moramo i izaći. To znači da će vrh biti incidentan s bridovima koji se mogu razdijeliti u dva skupa: oni koji ulaze u vrh i oni koji izlaze iz vrha. Jasno je da broj članova jednog i drugog skupa mora biti jednak. To povlači da je ukupan broj članova oba skupa (broj bridova incidentnih s promatranim vrhom) jednak dvostrukom broju članova skupa „ulaznih“ bridova, a to je uvijek paran broj.

$\Leftarrow$: Dokažimo da je najdulja staza u grafu Eulerova staza. Neka je $T = v_0, e_1, v_1, \ldots, e_m, v_m$ najdulja staza grafa. Eulerova staza je zatvorena pa moramo dokazati da vrijedi $v_0 = v_m$. Pretpostavimo suprotno. Ako vrijedi $v_0 \neq v_m$, onda je vrh v_0 incidentan s neparnim brojem bridova iz staze T. Vrijedi da je svaki vrh grafa parnog stupnja, što znači da postoji brid grafa koji nije dio staze T, a incidentan je s vrhom v_0. Taj brid možemo dodati u stazu T, što znači da prvotno odabrana staza T nije bila najdulja. To je kontradikcija iz koje zaključujemo da ne vrijedi $v_0 \neq v_m$, što povlači da vrijedi $v_0 = v_m$.

Eulerova staza prolazi svakim bridom grafa, tako da moramo dokazati da su $e_1, \ldots, e_m$ zaista svi bridovi grafa G. Prvo gledamo slučaj kada vrijedi $V(T) \neq V(G)$. Neka postoji jedan vrh grafa v_k koji nije uključen u najdulju stazu. Znamo da je graf G povezan, što znači da je vrh v_k sigurno povezan bridom s nekim vrhom koji je dio staze T. Taj brid

možemo dodati u stazu T, što znači da prvotno odabrana staza T nije bila najdulja. To je kontradikcija.

Gledamo drugi slučaj: ako najdulja staza T ima jednak broj vrhova kao graf G, ali nema jednak broj bridova, to znači da postoji neki brid grafa e_k koji nije dio staze T. Taj brid mora biti incidentan s neka dva vrha (ili jednim) iz grafa G. Kako su ti vrhovi također dio staze T, slijedi da je i brid e_k dio staze T, što je kontradikcija.

Dokazali smo da je odabrana najdulja staza Eulerova staza.

□

Koristeći Teorem 2 možemo lako riješiti problem mostova u Königsbergu. Odmah vidimo da u grafu postoje vrhovi neparnog stupnja, što znači da graf nije eulerovski i da nije moguće prijeći preko svakog mosta točno jednom i vratiti se na početak. Sada zamislimo da je problem malo drugačiji i da pitanje glasi: „Je li moguće proći preko svakog mosta točno jednom?“ Dakle, sada nas ne zanima vraćanje u početnu točku. I ovaj problem možemo riješiti bez traženja staze koja bi sadržavala sve bridove grafa.

Definicija 2.3.2. *Graf je **semieulerovski** ako sadrži stazu koja prolazi svakim bridom točno jednom i nije zatvorena.*

Rješenje problema svodi se na provjeru semieulerovosti grafa. Koristit ćemo sljedeći korolar.

KOROLAR 1. *Povezani graf G je semieulerovski ako i samo ako su točno dva njegova vrha neparnog stupnja.*

Dokaz. Ako graf ima stazu koja prolazi svakim bridom, to znači da svaki vrh koji nije početak ni kraj te staze ima paran stupanj. Takav vrh je incidentan s bridovima koji se mogu podijeliti na „ulazne“ i „izlazne“. Ukupan broj bridova incidentnih s tim vrhom je dvostruki broj „ulaznih“ bridova, što je uvijek paran broj.

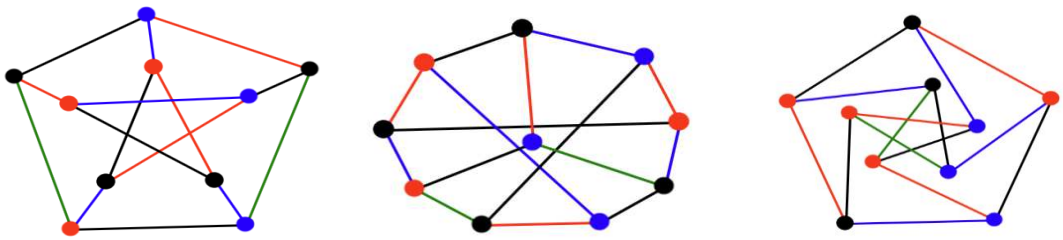
Ako ta staza nije zatvorena, to znači da su početni i krajnji vrh neparnog stupnja. Pretpostavimo suprotno. Ako postoji krajnji vrh otvorene staze koji ima paran stupanj, onda postoji brid koji je incidentan krajnjem vrhu, a nismo ga uključili u stazu. To znači da staza ne prolazi svim bridovima grafa, tj. graf nije semieulerovski. Dobivamo kontradikciju iz čega slijedi da početni i krajnji vrh moraju biti neparnog stupnja.

Dakle, „unutrašnji“ vrhovi otvorene staze koja prolazi svakim bridom su uvijek parnog stupnja, a krajnja dva vrha su uvijek neparnog stupnja, što povlači da vrijedi tvrdnja korolara.

□

Sa slike (Slika 2.3.1 [7]) vidimo da graf nije semieulerovski jer ima više od dva vrha neparnog stupnja. To znači da nije moguće prijeći preko svakog mosta točno jednom.

Ispitat ćemo eulerovost poznatog grafa koji se često koristi u mnogim primjerima, a to je Petersenov graf. To je 3-regularan graf (svaki vrh je stupnja 3) koji ima 10 vrhova i 15 bridova. Na slici (Slika 2.3.2) vidimo da graf nije eulerovski jer sadrži vrhove neparnog stupnja.



Slika 2.3.2 Različiti načini prikaza Petersenovog grafa

Graf nije ni semieulerovski jer ima više od dva vrha neparnog stupnja.

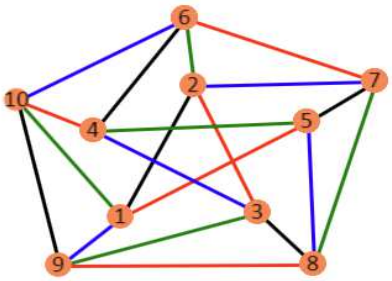
Primjer 1. Kako možemo modificirati broj bridova Petersenovog grafa tako da novi graf bude eulerovski ili semieulerovski?

Dodavanjem novih pet bridova postižemo to da svaki vrh ima paran stupanj pa je graf eulerovski. Odredimo eulerovsku stazu sa slike (Slika 2.3.3):

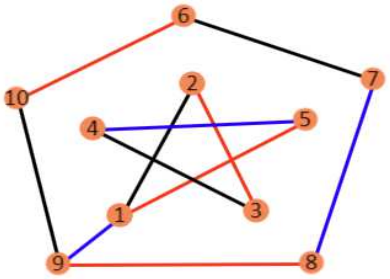
$1 \to 2 \to 3 \to 4 \to 5 \to 1 \to 10 \to 4 \to 6 \to 2 \to 7 \to 5 \to 8 \to 3 \to 9 \to 10 \to 6 \to 7 \to 8 \to 9 \to 1.$

Ako iz Petersenovog grafa obrišemo četiri brida, dobivamo semieulerovski graf. Odredimo otvorenu stazu koja prolazi svim bridovima točno jednom sa slike (Slika 2.3.4):

$1 \to 2 \to 3 \to 4 \to 5 \to 1 \to 9 \to 10 \to 6 \to 7 \to 8 \to 9.$



Slika 2.3.3 Eulerovski graf



Slika 2.3.4 Semieulerovski graf

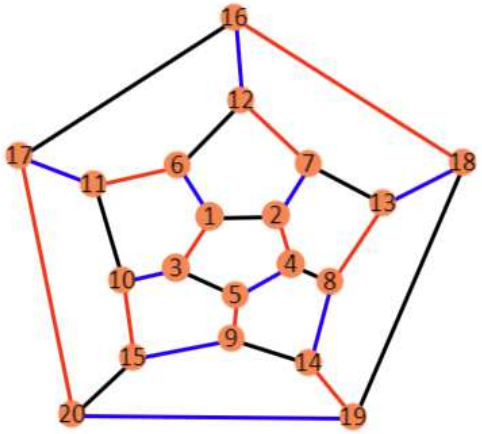
2.4. Hamiltonovost i semi-hamiltonovost

William Rowan Hamilton bio je irski matematičar koji je u 19. stoljeću proučavao problem obilaska svih vrhova grafa. Definirajmo najvažnije pojmove.

Definicija 2.4.1. ***Hamiltonov put*** *je put (staza kojoj su svi vrhovi različiti) koji sadrži sve vrhove grafa.*

Definicija 2.4.2. ***Hamiltonov ciklus*** *je zatvorena staza koja sadrži sve vrhove grafa i ima međusobno različite sve unutarnje vrhove (vrhove koji nisu krajnji). Za graf G kažemo da je* ***hamiltonovski*** *ako ima Hamiltonov ciklus.*

Hamilton je pokušavao riješiti problem u kojem trgovački putnik mora obići gradove i vratiti se u početni grad. Svaki grad može posjetiti točno jedanput i ne smije dvaput proći istom cestom. Problem je prikazan grafom dodekaedra, gdje vrhovi predstavljaju gradove, a bridovi ceste među njima (Slika 2.4.1). Rješavanje problema svodi se na traženje Hamiltonovog ciklusa. Sa slike (Slika 2.4.1) možemo vidjeti da postoji sljedeći Hamiltonov ciklus: 14 → 19 → 18 → 16 → 17 → 20 → 15 → 10 → 11 → 6 → 12 → 7 → 13 → 8 → 4 → 2 → 1 → 3 → 5 → 9 → 14.

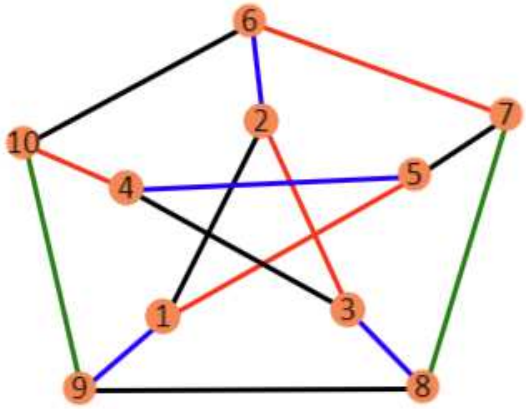


Slika 2.4.1 Graf dodekaedra

Dakle, u ovom slučaju, trgovački putnik može obići svaki grad jednom bez ponovnih prolazaka istom cestom i vratiti se u grad iz kojeg je krenuo.

Definicija 2.4.3. ***Semihamiltonovski graf*** *je graf koji sadrži Hamiltonov put (put kroz svaki vrh).*

Pogledajmo primjer Petersenovog grafa (Slika 2.4.2). Petersenov graf nije hamiltonovski jer nema Hamiltonov ciklus, ali jest semihamiltonovski jer ima Hamiltonov put, npr. $1 \to 2 \to 3 \to 4 \to 5 \to 7 \to 8 \to 9 \to 10 \to 6$.

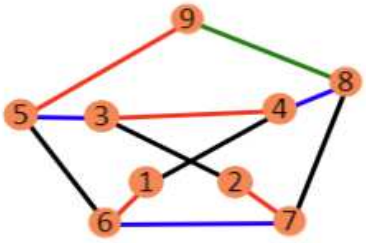


Slika 2.4.2 Petersenov graf

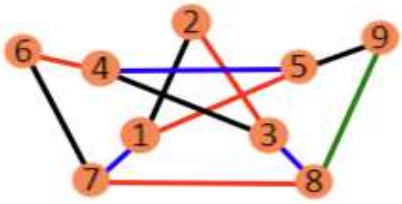
Petersenov graf je najmanji 3-regularan graf bez mostova koji ne sadrži Hamiltonov ciklus. On je jedan od četiriju poznatih protuprimjera Lovaszove slutnje [5] koja kaže da svaki povezani graf koji je tranzitivan po vrhovima sadrži Hamiltonov put. Graf *G*

koji je tranzitivan po vrhovima posjeduje svojstvo da za bilo koji par vrhova v_1 i v_2 u G postoji automorfizam na grafu G koji preslikava vrh v_1 u v_2. Automorfizam predstavlja bijektivno preslikavanje grafa na sama sebe, tako da je očuvana struktura grafa. Drugim riječima, automorfizam je izomorfizam grafa na samoga sebe.

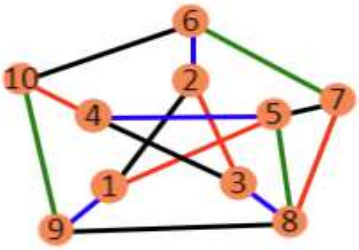
Primjer 2. Još jedna zanimljivost jest da možemo lako dobiti hamiltonovski graf ako Petersenovom grafu oduzmemo bilo koji vrh, ili ako dodamo brid između bilo koja dva vrha pazeći da graf i dalje ostane jednostavan. Slika 2.4.3 prikazuje grafove nastale oduzimanjem jednog vrha (grafovi a. i b.) te grafove nastale dodavanjem jednog brida (grafovi c. i d.). Ispod svakog grafa je napisan redoslijed vrhova pronađenog Hamiltonovog ciklusa.



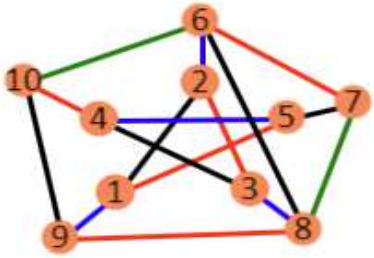
a. $1 \to 4 \to 3 \to 2 \to 7 \to 8 \to 9 \to 5 \to 6 \to 1$



b. $6 \to 4 \to 5 \to 9 \to 8 \to 3 \to 2 \to 1 \to 7 \to 6$



c. $1 \to 2 \to 3 \to 4 \to 5 \to 8 \to 7 \to 6 \to 10 \to 9 \to 1$



d. $1 \to 2 \to 3 \to 4 \to 5 \to 7 \to 8 \to 6 \to 10 \to 9 \to 1$

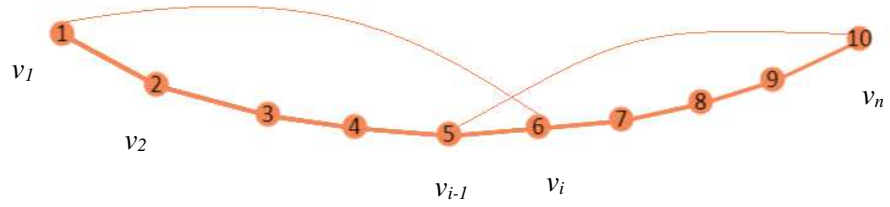
Slika 2.4.3 Hamiltonovski grafovi

Uzimajući u obzir prethodne primjere, jasno je da veći minimalni stupanj grafa (najmanji stupanj vrha u tom grafu) daje veću vjerojatnost da je graf hamiltonovski. Što je veći stupanj vrha, to je vrh povezan s više drugih vrhova pa intuitivno slijedi da je veća vjerojatnost pronalaska Hamiltonovog ciklusa. U nekim slučajevima, za

ispitivanje hamiltonovosti ne moramo tražiti Hamiltonov ciklus. Tu nam mogu pomoći sljedeći teoremi.

TEOREM 3 (Ore). [1] *Ako jednostavan graf G ima n ≥ 3 vrha i ako je zbroj stupnjeva svakog para nesusjednih vrhova veći ili jednak broju vrhova, onda je G hamiltonovski graf.*

Dokaz. Pretpostavimo da *G* nije hamiltonovski. Odaberimo neka dva vrha koja nisu susjedna i spojimo ih bridom. Ponavljajmo postupak sve dok ne dobijemo graf koji bi, da mu dodamo još jedan brid, imao Hamiltonov ciklus. Jasno je da graf ima Hamiltonov put. Označimo prvi vrh Hamiltonovog puta s v_1, a posljednji s v_n. Kako graf nema Hamiltonov ciklus, vrhovi v_1 i v_n nisu susjedni. Sa Slike 2.4.4 vidimo da vrh v_1 može biti susjedan bilo kojem vrhu $v_2,\dots,v_n$, a vrh v_n može biti susjedan bilo kojem vrhu $v_1,\dots,v_{n-1}$. Broj vrhova iz niza $v_2,\dots,v_n$ koji su susjedni vrhu v_1 je stupanj vrha v_1, a broj vrhova iz niza $v_1,\dots,v_{n-1}$ koji su susjedni vrhu v_n je stupanj vrha v_n. Prema hipotezi teorema, zbroj stupnjeva vrhova v_1 i v_n mora biti barem *n*. Iz toga slijedi da mora postojati vrh v_i susjedan vrhu v_0 takav da je vrh v_{i-1} susjedan vrhu v_n. To znači da postoji sljedeći Hamiltonov ciklus (Slika 2.4.4) : $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_{i-1} \rightarrow v_n \rightarrow v_{n-1} \rightarrow \dots \rightarrow v_i \rightarrow v_1$. To je kontradikcija naše pretpostavke, stoga zaključujemo da *G* jest hamiltonovski graf.



Slika 2.4.4 Hamiltonov ciklus

□

TEOREM 4 (Dirac). [1] *Ako jednostavan graf G ima n ≥ 3 vrha i minimalni stupanj* $\delta \geq \frac{n}{2}$*, onda je G hamiltonovski.*

Dokaz. Teorem dokazujemo koristeći Teorem 3. Ako svaki vrh ima minimalan stupanj *n/2*, to znači da je zbroj stupnjeva bilo koja dva nesusjedna vrha barem *n*. Po Teoremu 3, graf je hamiltonovski.

□

2.5. Most

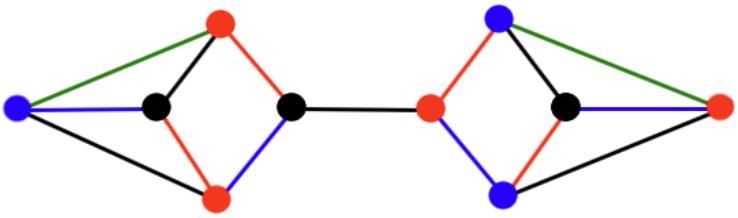
Definicija 2.5.1. *Neka je G graf koji sadrži vrhove iz skupa V i bridove iz skupa E. Neka je brid e ∈ E incidentan s vrhovima* v_1, $v_2 \in V$. *Brid e je most grafa G ako njegovim uklanjanjem više ne postoji put koji spaja vrhove* v_1 *i* v_2.

Iz prethodne definicije je jasno da je most brid čijim uklanjanjem u povezanom grafu dobivamo nepovezani graf. U nepovezanom grafu, most je brid čijim uklanjanjem povećavamo broj komponenti povezanosti (članova unije povezanih grafova kojom možemo prikazati nepovezani graf).

Slika 2.5.1 prikazuje primjer stabla. Stablo je graf kojem je svaki brid most.

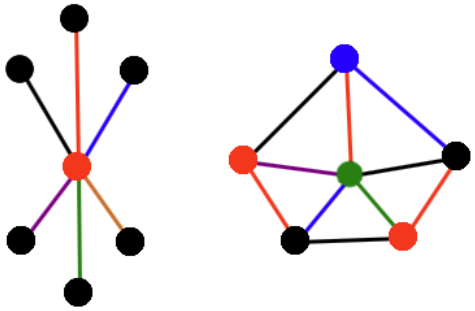
Slika 2.5.1 Stablo

Slika 2.5.2 prikazuje još jedan primjer povezanog grafa koji ima most. Jasno je da bi uklanjanjem središnjeg brida graf postao nepovezan, tako da je taj brid most.



Slika 2.5.2 Primjer mosta u povezanom grafu

Sljedeća slika (Slika 2.5.3) prikazuje primjer nepovezanog grafa s dvije komponente, zvijezdom S_7 i kotačem W_6. Ako bismo uklonili bilo koji brid zvijezde, dobili bismo nepovezani graf s tri komponente: zvijezdom S_6, kotačem W_6 te jednim nepovezanim vrhom. Dakle, svaki brid zvijezde je most pa graf (Slika 2.5.3) ima 6 mostova.



Slika 2.5.3 Primjer mosta u nepovezanom grafu

2.6. Kromatski broj i kromatski indeks

Definicija 2.6.1. ***Kromatski broj*** *$\chi(G)$ je najmanji broj boja kojima možemo obojiti vrhove grafa G tako da ne postoje dva susjedna vrha kojima je pridružena ista boja.* ***Kromatski indeks*** *$\chi'(G)$ je najmanji broj boja kojima možemo obojiti bridove grafa tako da ne postoje dva susjedna brida kojima je pridružena ista boja.*

Kažemo da je graf *k*-obojiv ako se vrhovi mogu obojiti u *k* ili manje boja, tako susjedni vrhovi nemaju istu boju. Ako je graf *k*-obojiv, ali nije *(k-1)*-obojiv, onda je *k* kromatski broj grafa. Ako u grafu ne postoje bridovi, vrijedi $\chi(G) = 1$. Kažemo da je graf bridno *k*-obojiv ako se bridovi mogu obojiti u *k* boja tako da su susjedni bridovi obojeni različitom bojom. Ako takav graf nije bridno *(k-1)*-obojiv, onda je *k* njegov kromatski indeks.

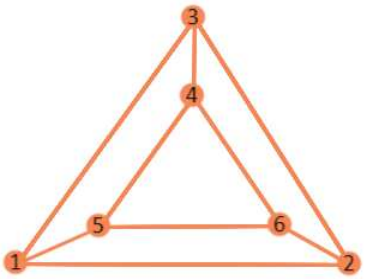
TEOREM 5 (Brooks). [1] *Neka je G povezani graf bez petlji i dvostrukih bridova koji nije potpun. Neka je najveći stupanj nekog vrha u G jednak $\Delta \geq 3$. To znači da je G Δ-obojiv.*

Primjer 3. Šest igrača želi igrati igru. Slika 2.6.1 prikazuje igrače kao vrhove grafa, a bridovi spajaju igrače koji će igrati jedan protiv drugog. Igrači se moraju podijeliti u timove, i to tako da ni u jednom timu nisu dva igrača koji će igrati jedan protiv drugoga. Koji je najmanji broj timova koji se može sastaviti?

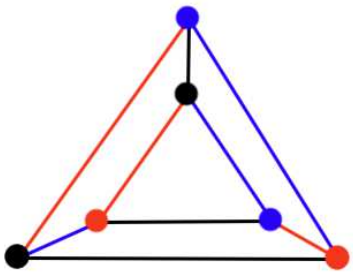
Neka je svaki tim predstavljen drugom bojom. To znači da dva susjedna vrha na grafu (Slika 2.6.1) ne smiju biti iste boje. Rješenje ovog problema svodi se na pronalazak kromatskog broja grafa. Sa slike (Slika 2.6.2) vidimo da je kromatski broj 3.

Svaki igrač mora igrati protiv svakog igrača s kojim je susjedan. U jednom danu se može igrati više mečeva, ali jedan igrač u jednom danu smije sudjelovati samo u jednom meču. Koji je najmanji broj dana potreban za završiti igru?

Neka je svaki dan u kojem se igra predstavljen drugom bojom. To znači da dva brida incidentna istom vrhu na grafu ne smiju biti iste boje. Moramo pronaći kromatski indeks grafa. Sa slike (Slika 2.6.2) je jasno da je kromatski indeks 3.



Slika 2.6.1 Grafički prikaz igrača



Slika 2.6.2 Graf s obojenim vrhovima i bridovima

Ako je graf k-partitan, to znači da je svaki brid incidentan s neka dva vrha koji ne pripadaju istom podskupu particije. U svakom podskupu, vrhovi su iste boje, što znači da nisu susjedni. Za takav graf vrijedi $\chi(G) = k$. (Slika 2.6.3)

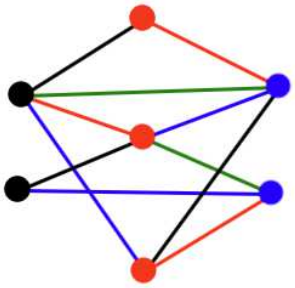
TEOREM 6 (König). [1] *Neka je G bipartitan graf s najvećim stupnjem nekog vrha Δ. Za takav graf vrijedi $\chi'(G) = \Delta$.*

TEOREM 7 (Vizing). [1] *Neka je G jednostavan graf s najvećim stupnjem Δ. Tada za njegov kromatski indeks vrijedi* $\Delta \leq \chi'(G) \leq \Delta + 1$.

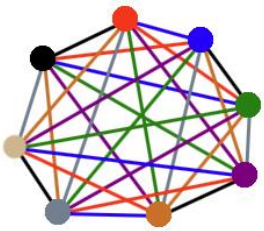
Slika 2.6.3 prikazuje 3-partitan graf za koji vrijedi $\chi(G) = 3$. Maksimalni stupanj grafa je 4 i vrijedi $\chi'(G) = 4$.

Mnogi grafovi imaju kromatski indeks $\chi'(G) = \Delta$. Jedan primjer grafa koji ima kromatski indeks $\chi'(G) = \Delta + 1$ je Petersenov graf (Slika 2.3.2). Njegov maksimalni stupanj je 3, a kromatski indeks 4.

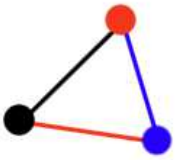
Ako je graf potpun i ima neparan broj vrhova n, onda vrijedi $\chi'(G) = n$. Ako je broj vrhova paran, onda $\chi'(G) = n - 1$. (Slika 2.6.3)



3-partitni graf: $\chi = 3$, $\chi' = 4$



K_8: $\chi = 8$, $\chi' = 7$



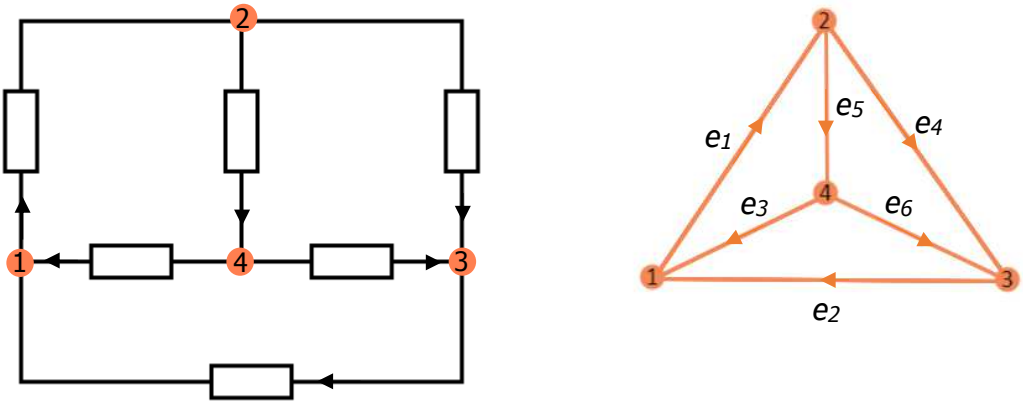
K_3: $\chi = 3$, $\chi' = 3$

Slika 2.6.3 Primjeri grafova

3. Neke primjene teorije grafova

3.1. Elektrotehnika

Primjenu teorije grafova u elektrotehnici pokazat ćemo na primjeru strujnog kruga. Strujni krug možemo prikazati kao graf čiji vrhovi predstavljaju čvorove, a bridovi grane (dijelove strujnog kruga kojima prolazi ista struja). Korištenjem Kirchhoffovih zakona, možemo pronaći iznose struja i napona u strujnom krugu. Slika 3.1.1 prikazuje shemu strujnog kruga i prikaz u obliku grafa. Strelice označavaju smjer struje.



Slika 3.1.1 Shema strujnog kruga i pripadajući graf

Prvo ćemo konstruirati matricu incidencije A usmjerenog grafa. Redci matrice predstavljat će vrhove grafa, a stupci bridove. Ako vrh i nije incidentan bridu j, pripadajući element matrice a_{ij} bit će 0. Ako je vrh i incidentan bridu j, gledamo smjer strelice brida u odnosu na vrh. Ako brid „izlazi“ iz vrha i, element matrice a_{ij} bit će 1, a ako brid „ulazi“ u vrh i, a_{ij} će biti -1 [4].

$$A = \begin{bmatrix} 1 & -1 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & -1 \\ 0 & 0 & 1 & 0 & -1 & 1 \end{bmatrix}.$$

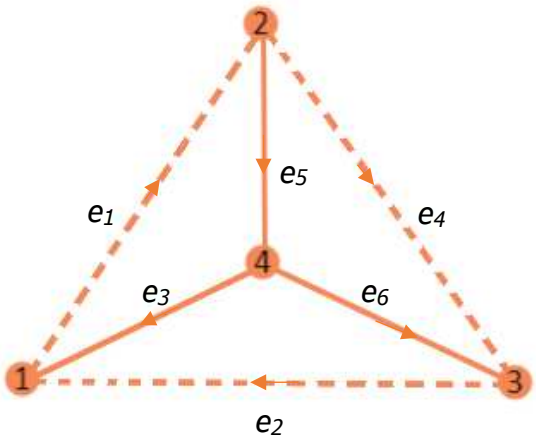
Definirajmo vektor struja grana I kao vektor čiji su elementi iznosi struje u pojedinoj grani:

$$I = \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \\ i_5 \\ i_6 \end{bmatrix}.$$

Prvi Kirchhoffov zakon kaže da zbroj struja koje ulaze u neki čvor strujnog kruga mora biti jednak zbroju struja koje izlaze iz tog čvora. Prema tome, za matricu incidencije *A* i vektor struje *I* vrijedi

$$A * I = \begin{bmatrix} 1 & -1 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & -1 \\ 0 & 0 & 1 & 0 & -1 & 1 \end{bmatrix} * \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \\ i_5 \\ i_6 \end{bmatrix} = \begin{bmatrix} i_1 - i_2 - i_3 \\ -i_1 + i_4 + i_5 \\ i_2 - i_4 - i_6 \\ i_3 - i_5 + i_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

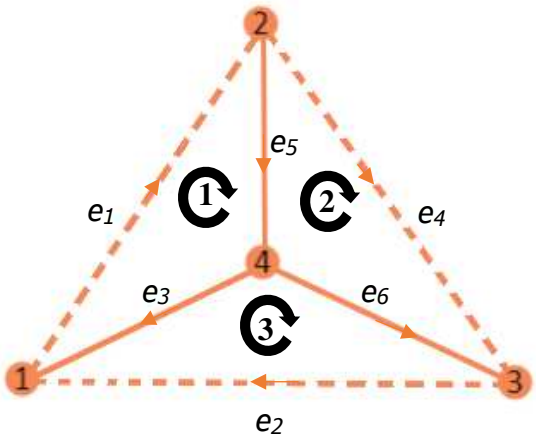
Sada ćemo pronaći jedno razapinjuće stablo grafa. Na slici (Slika 3.1.2) razapinjuće stablo čine vrhovi i bridovi označeni punom crtom. Takvi bridovi nazivaju se grane stabla (e_3, e_5, e_6), a bridovi grafa (Slika 3.1.1) koji nisu dio odabranog razapinjućeg stabla su spone ili neovisne grane (e_1, e_2, e_4) [4].



Slika 3.1.2 Razapinjuće stablo grafa i spone

Temeljna petlja je naziv za petlju koju čine grane razapinjućeg stabla uz jednu dodanu sponu. U ovom slučaju pronalazimo tri temeljne petlje (Slika 3.1.3) i konstruiramo spojnu matricu. Spojna matrica *S* je matrica kojoj redci predstavljaju temeljne petlje,

a stupci sve grane i spone (Slika 3.1.3). Proizvoljno odaberemo smjer svake temeljne petlje.



Slika 3.1.3 Temeljne petlje

Ako *j*-ti brid ne pripada *i*-toj temeljnoj petlji, pripadajući element matrice s_{ij} bit će *0*. Ako *j*-ti brid pripada *i*-toj temeljnoj petlji, gledamo kojeg je smjera: ako se smjer brida *j* podudara sa smjerom temeljne petlje *i*, element matrice s_{ij} bit će *1*, u suprotnom će biti *-1*. Vrijedi

$$S = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & -1 & -1 \\ 0 & 1 & -1 & 0 & 0 & 1 \end{bmatrix}.$$

Drugi Kirchhoffov zakon kaže da je algebarski zbroj napona u svakoj nezavisnoj petlji strujnog kruga jednak nuli. Definirajmo vektor napona grana *U* kao vektor čiji su elementi iznosi napona u pojedinoj grani:

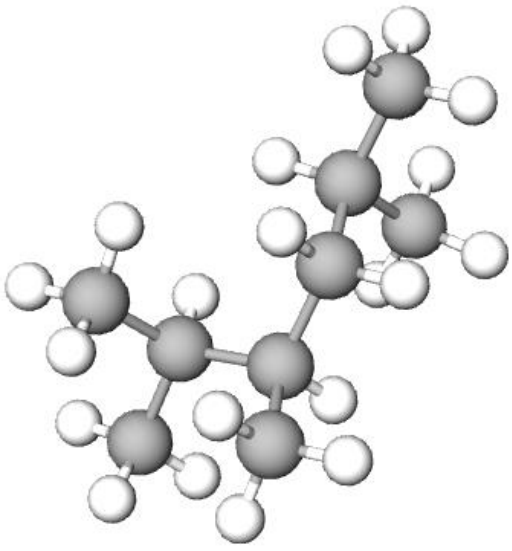
$$U = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{bmatrix}.$$

Iz toga slijedi:

$$S * U = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & -1 & -1 \\ 0 & 1 & -1 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{bmatrix} = \begin{bmatrix} u_1 + u_3 + u_5 \\ u_4 - u_5 - u_6 \\ u_2 - u_3 + u_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

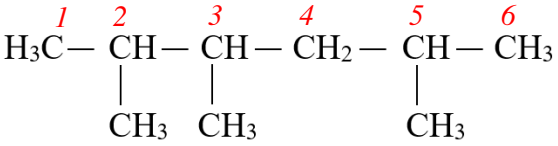
3.2. Kemija

Teorija grafova koristi se za modeliranje i analiziranje kompleksnih kemijskih struktura. Molekularne strukture često se prikazuju pomoću grafova, gdje vrhovi predstavljaju atome, a bridovi veze između njih. Ovakva reprezentacija olakšava vizualizaciju i analizu molekularnih svojstava. Korisna je i za rješavanje problema identifikacije izomera (molekula s istim kemijskim formulama koje imaju različite strukture). Kemijske reakcije također se mogu modelirati pomoću grafova: bridovi u grafu predstavljaju kemijske veze, a putovi u grafu odražavaju različite reakcijske putove. Slika 3.2.1 prikazuje primjer kemijskog spoja 2,3,5-trimetilheksana.



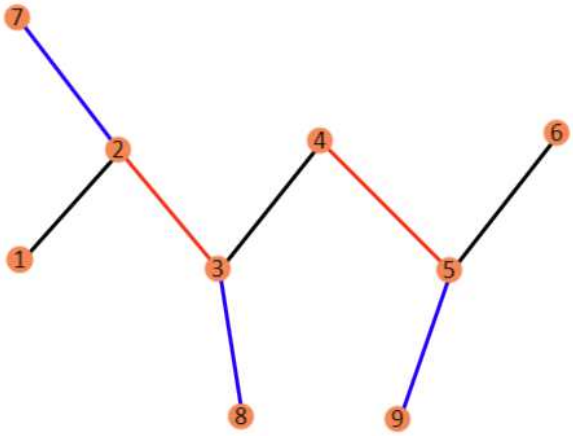
Slika 3.2.1 Model 2,3,5-trimetilheksana

Spoj se sastoji od ukupno 9 atoma ugljika (C) i 20 atoma vodika (H). Sljedeća slika (Slika 3.2.2) prikazuje sažetu strukturnu formulu 2,3,5-trimetilheksana.



Slika 3.2.2 Sažeta strukturna formula 2,3,5-trimetilheksana

Kemijski spoj ćemo prikazati grafički tako da jedan vrh grafa predstavlja jedan atom ugljika, a bridovi predstavljaju jednostruke veze među njima (Slika 3.2.3).



Slika 3.2.3 Prikaz 2,3,5-trimetilheksana u obliku grafa

Često je korisno prikazati udaljenosti između vrhova uz pomoć matrice. Konstruirat ćemo matricu udaljenosti D za prethodni graf. Element matrice udaljenosti d_{ij} predstavlja udaljenost (duljinu najkraćeg puta) između i-tog i j-tog vrha [6].

$$D = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 2 & 3 & 5 \\ 1 & 0 & 1 & 2 & 3 & 4 & 1 & 2 & 4 \\ 2 & 1 & 0 & 1 & 2 & 3 & 2 & 1 & 3 \\ 3 & 2 & 1 & 0 & 1 & 2 & 3 & 2 & 2 \\ 4 & 3 & 2 & 1 & 0 & 1 & 4 & 3 & 1 \\ 5 & 4 & 3 & 2 & 1 & 0 & 5 & 4 & 2 \\ 2 & 1 & 2 & 3 & 4 & 5 & 0 & 3 & 5 \\ 3 & 2 & 1 & 2 & 3 & 4 & 3 & 0 & 4 \\ 5 & 4 & 3 & 2 & 1 & 2 & 5 & 4 & 0 \end{bmatrix}$$

Američki kemičar Herbert Wiener je u 20. st. proučavao kemijska svojstva alkana (acikličkih zasićenih ugljikovodika). Došao je do spoznaje da je zbroj udaljenosti svih parova atoma (izuzev atoma vodika) u nekoj strukturi povezan s mnogim različitim

fizikalnim i kemijskim svojstvima. Tu veličinu nazivamo Wienerov indeks ili Wienerov broj [6], za koji vrijedi (1):

$$W = \frac{1}{2}\sum_{i}\sum_{j} d_{ij}\,. \tag{1}$$

Wienerov indeks za 2,3,5-trimetilheksan iznosi *96*.

Osim udaljenosti atoma, za ispitivanje raznih svojstava važan je i broj veza atoma u strukturi. Hrvatsko-američki znanstvenik Milan Randić proučavao je načine povezanosti atoma u molekuli i predstavio veličinu indeksa povezanosti χ, koji se po njemu još naziva i Randićev indeks [6] (2):

$$\chi(\mathrm{G}) = \sum_{\mathrm{q}\in\mathrm{E(G)}} \frac{1}{geometrijska\ srednja\ vrijednost\ q}. \tag{2}$$

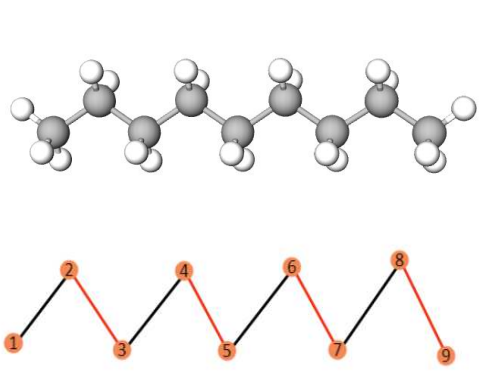
Geometrijska srednja vrijednost brida q koji spaja vrhove v_i i v_j se računa kao $\sqrt{\deg(v_i) * \deg(v_j)}$. Randićev indeks korelira s važnim svojstvima alkana, poput vrelišta, topljivosti u vodi i veličine molekule. Za 2,3,5-trimetilheksan iznosi *4.0366*.

Još jedna fizikalna veličina koja opisuje strukturu kemijskog spoja jest Balabanov indeks *J* [6]. Alexandru Balaban je rumunjski kemičar koji je osamdesetih godina prošlog stoljeća definirao veličinu koja uzima u obzir zbroj prosječnih udaljenosti atoma u strukturi spoja (3):

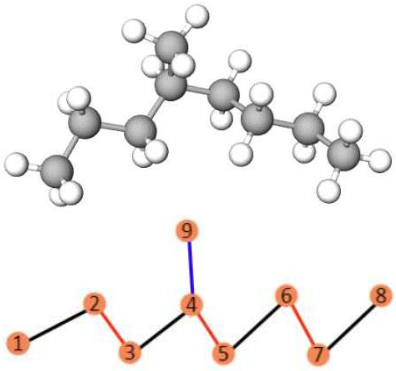
$$J(G) = \frac{M}{c+1}\sum_{\mathrm{q}\in\mathrm{E(G)}} \frac{1}{\sqrt{d_i d_j}}. \tag{3}$$

M označava broj bridova u grafu, *c* je ciklomatski broj (najmanji broj bridova koji se trebaju izbrisati tako da graf nema ciklus), a d_i je zbroj elemenata matrice udaljenosti *D* u retku *i*. Balabanov indeks za 2,3,5-trimetilheksan je *3.3766*.

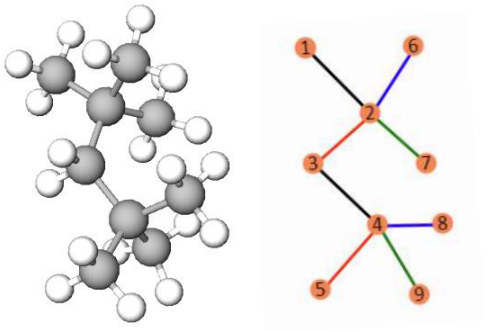
2,3,5- trimetilheksan je jedan od 35 strukturnih izomera molekule C_9H_{20}. Slika 3.2.4 prikazuje još neke izomere iste molekule i njihove grafove.



Nonan



4-metiloktan



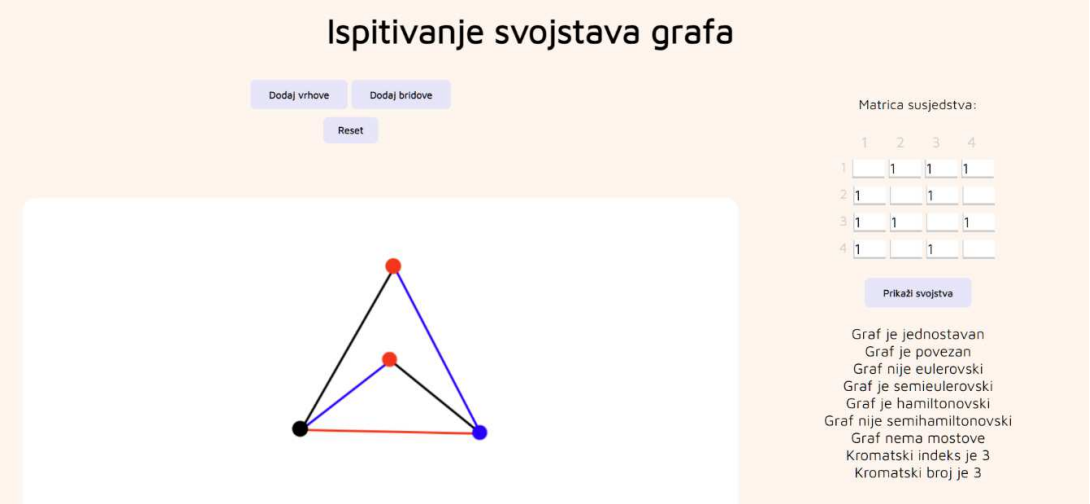
2,2,4,4-tetrametilpentan

Slika 3.2.4 Izomeri molekule C_9H_{20}

4. Razvoj aplikacije

4.1. Tehničke značajke

Aplikacija za ispitivanje svojstava grafa omogućuje korisniku da na interaktivan način provjeri je li neki graf jednostavan, povezan, eulerovski, semieulerovski, hamiltonovski, semihamiltonovski, ima li mostove te koji su mu kromatski broj i kromatski indeks. Slika 4.1.1 prikazuje korisničko sučelje nakon provjere svojstava.



Slika 4.1.1 Izgled korisničkog sučelja

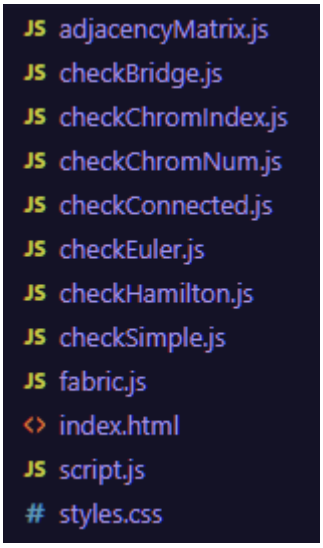
Aplikacija traži korisnika unos podataka koji opisuju graf. Korisnik klikom miša ucrtava vrhove svog grafa, nakon čega se pojavljuje matrica susjedstva. Korisnik može dodati bridove ispunjavanjem matrice susjedstva ili izravno ucrtati klikom miša na vrhove incidentne željenom bridu. Klikom na gumb „Prikaži svojstva“, pokreće se program koji učitava matricu susjedstva. Uneseni podatci prolaze kroz niz funkcija koje provjeravaju navedena svojstva. Tijekom rekurzivnog traženja kromatskog broja i kromatskog indeksa, vrhovima i bridovima se dodjeljuju boje. Izlazni parametri su rezultati testova navedenih svojstava te prikaz grafa. Vrhovi i bridovi grafa prikazani su na mjestima na kojima ih je ucrtao korisnik te su obojeni u skladu s pronađenim kromatskim brojem i kromatskim indeksom.

4.1.1. Razvojno okruženje

Projekt je izrađen pomoću web tehnologija, uključujući HTML, CSS i JavaScript, a za manipulaciju *canvas* elemenata koristi se *Fabric.js* biblioteka. Uređivač koda za implementaciju projekta je *Visual Studio Code*. Projekt je testiran korištenjem *Live Servera* u okviru *Visual Studio Codea* te pokretan u *Chrome* web pregledniku.

4.1.2. Struktura projekta

Koristili smo HTML za osnovnu strukturu korisničkog sučelja aplikacije. Izgled i oblikovanje elemenata na stranici ostvareno je u CSS-u. Za izradu programskog rješenja koristili smo JavaScript programski jezik uz *Fabric* JavaScript biblioteku koja omogućava crtanje grafa. Na slici (Slika 4.1.2) su prikazane datoteke projekta.



Slika 4.1.2 Struktura projekta

Datoteka *index.html* se sastoji od dva glavna strukturna blok elementa: *header* u kojem je deklariran naslov aplikacije (nalazi se na vrhu i proteže cijelom širinom ekrana) te *grid-container* koji omogućava da ostatak stranice ima oblik tablice koja se sastoji od jednog retka i dva stupca. Prvi stupac sadrži objekte tipa button, a to su gumbi „Dodaj vrhove“, „Dodaj bridove“ i „Reset“ te objekt tipa *canvas* čije korištenje omogućava fabric.js biblioteka. Canvas služi za korisnikovo ucrtavanje vrhova grafa, a nakon provedenog ispitivanja svojstava, na njemu se prikazuju obojeni vrhovi i bridovi. Drugi stupac sadrži strukturni blok *matrix* u kojem je prikazana matrica susjedstva

ispod koje se nalazi gumb „Prikaži svojstva“. U drugom stupcu se također nalazi strukturni blok *result* u kojem su prikazani rezultati ispitivanja svojstava.

Glavne funkcionalnosti projekta izvedene su u datoteci script.js. Klikom na gumb „Dodaj vrhove“ poziva se funkcija `activateAddingVertex()`. Tada je za korisnika omogućeno dodavanje vrhova grafa na ekranu. Klikom na područje elementa canvas, poziva se funkcija `startAddingVertex()` koja povećava brojač vrhova `numOfVertices` za jedan, pronalazi koordinate kursora u trenutku klika i na tom mjestu na ekranu prikazuje krug unutar kojega je prikazan njegov redni broj. Prilikom dodavanja vrhova, svi su obojeni coral bojom heksadecimalnog zapisa #FF7F50, a kasnije će im boja biti promijenjena.

Klikom na gumb „Dodaj bridove“ poziva se funkcija `uploadTable()` iz datoteke *adjacencyMatrix.js*. Ona stvara kvadratnu matricu čije dimenzije ovise o globalnoj varijabli `numOfVertices`, odnosno o broju vrhova. Korisniku se prikazuje prazna matrica koju treba ispuniti te gumb „Prikaži svojstva“. Korisnik može ispuniti matricu crtanjem bridova klikom miša na krajnje vrhove željenog brida ili izravnim upisivanjem brojeva u matricu.

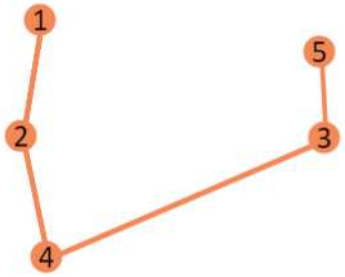
Klikom na gumb „Prikaži svojstva“ poziva se funkcija `calculate()`. Ako je korisnik u matricu unio neočekivanu vrijednost, prikazuje se upozorenje i korisnik može ispraviti grešku. Ako je sve u redu, pokreće se ispitivanje svojstava za graf određen matricom susjedstva.

4.1.2.1 Provjera jednostavnosti

Prvo se provjerava je li graf jednostavan pozivanjem funkcije `checkSimple()` iz datoteke *checkSimple.js*. Prolazimo kroz elemente matrice susjedstva. Ako naiđemo na element koji predstavlja više bridova između dva vrha, funkcija vraća *false*. Ako naiđemo na petlju (brid koji je incidentan samo s jednim vrhom), funkcija također vraća *false*. Ako prođemo cijelu matricu, funkcija vraća *true* i graf je jednostavan.

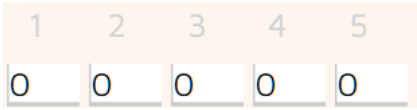
4.1.2.2 Provjera povezanosti

Zatim se provjerava povezanost grafa pozivom funkcije `checkConnected()` iz datoteke *checkConnected.js*. Graf je povezan ako možemo naći šetnju koja obuhvaća bilo koji par vrhova. Pogledajmo primjer grafa koji ima pet vrhova (Slika 4.1.3).



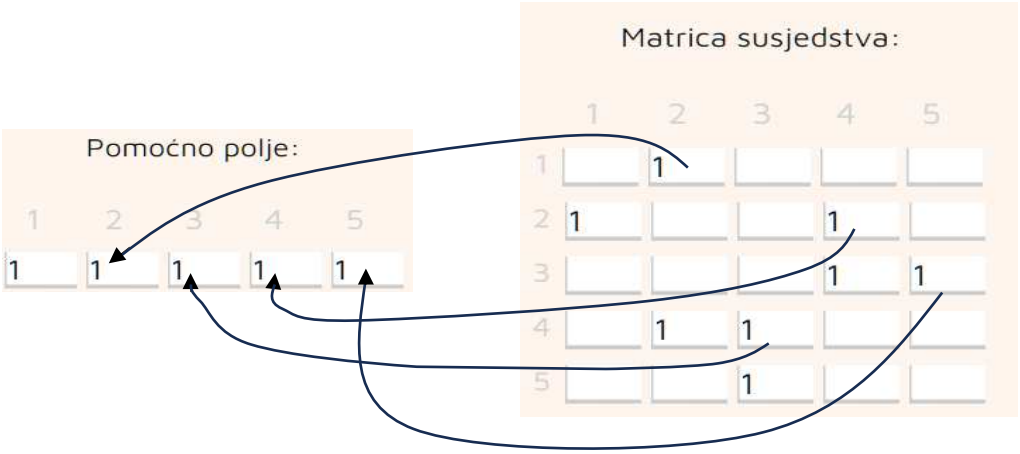
Slika 4.1.3 Primjer grafa

Definiramo pomoćno polje `tempArr` čiji broj elemenata je jednak broju vrhova te im dodjeljujemo vrijednost nula (Slika 4.1.4).



Slika 4.1.4 Pomoćno polje

Prvom elementu (koji predstavlja prvi vrh) dodjeljujemo vrijednost jedan. Prolazimo po matrici susjedstva. Gledamo prvi red. Ako naiđemo na brid koji spaja neki vrh s prvim vrhom, u polju `tempArr` dodjeljujemo vrijednost jedan elementu koji predstavlja taj vrh te smatramo da smo te vrhove „prošli“. U sljedećim redovima, za svaki brid na koji naiđemo, provjeravamo spaja li neki vrh koji još nismo „prošli“ s nekim koji jesmo. Ako je to istina, onda u pomoćnom polju elementu koji predstavlja taj vrh dodjeljujemo vrijednost jedan, kao i elementima koji predstavljaju njemu susjedne vrhove, a koje prethodno nismo „prošli“ (Slika 4.1.5).



Slika 4.1.5 Prikaz ispunjavanja pomoćnog polja

Kada dođemo do kraja matrice, provjeravamo imaju li svi elementi pomoćnog polja vrijednost jedan. Ako nemaju, funkcija vraća indeks elementa matrice susjedstva koji prvi nije povezan s početnim vrhom i vraća *false*. U suprotnom vraća *true* i graf je povezan.

4.1.2.3 Provjera eulerovosti

Funkcije za provjeru eulerovosti deklarirane su u datoteci *checkEuler.js*. Ako je graf povezan, poziva se funkcija `checkEulerForConnected()`. Definirana je varijabla `numOfOdd` koja će služiti kao brojač vrhova neparnog stupnja i dodijeljena joj je vrijednost nula. Za povezani graf je jednostavno provjeriti je li eulerovski. Potrebno je samo provjeriti postoji li vrh neparnog stupnja. Za svaki red matrice susjedstva (vrh), brojimo broj bridova. Ako taj broj bude neparan, graf nije eulerovski i brojač vrhova neparnog stupnja se povećava za jedan. Ako i samo ako je na kraju brojač `numOfOdd` jednak dva, graf je semieulerovski. Ako smo došli do kraja matrice bez pronalaska vrha s neparnim stupnjem, graf je eulerovski, što povlači da nije semieulerovski. Slika 4.1.6 prikazuje provjeru stupnjeva vrhova grafa s prethodne slike (Slika 4.1.3). U ovom slučaju je broj neparnih stupnjeva jednak dva, što znači da je graf semieulerovski.

Slika 4.1.6 Provjera stupnja svakog vrha

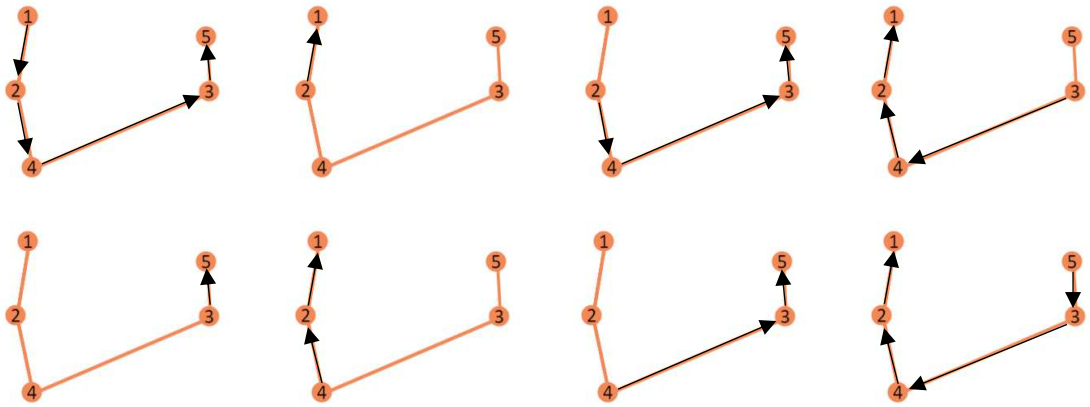
Ako graf nije povezan, poziva se funkcija `checkEulerForDisconnected()`. Koristimo varijablu `indexOfDisconnected` u kojoj je nakon poziva funkcije `checkConnected()` spremljen indeks vrha koji prvi nije spojen s prvim vrhom u matrici. Provjeravamo postoji li brid koji spaja taj vrh s nekim drugim (ili istim) vrhom:

- ako ne postoji, stvaramo novu matricu u kojoj isključujemo taj vrh i provjeravamo je li graf koji je predstavljen novom matricom povezan. Ako jest, poziva se funkcija `checkEulerForConnected()`, a ako nije, poziva se `checkEulerForDisconnected()`, tj. funkcija poziva samu sebe.
- ako postoji, provjeravamo je li taj brid petlja (ili ako postoji više bridova, provjeravamo jesu li svi bridovi petlje). Ako jest, provjeravamo je li ta petlja jedini brid u grafu (ako ima više petlji, provjeravamo da su to jedini bridovi u grafu). Ako jest, graf je eulerovski (i nije semieulerovski). Ako u grafu postoji još bridova, graf nije eulerovski ni semieulerovski.

4.1.2.4 Provjera hamiltonovosti

Zatim provjeravamo je li graf hamiltonovski i semihamiltonovski pozivom funkcije `checkHamiltonian()` iz datoteke *checkHamilton.js*. Ako se graf sastoji od samo jednog vrha bez bridova, graf je hamiltonovski i nije semihamiltonovski. U suprotnom tražimo hamiltonovski ciklus (ciklus koji povezuje sve vrhove). Definiramo pomoćnu varijablu `tempArr` kao polje dimenzije broja vrhova i na početku svakom elementu

dodjeljujemo vrijednost nula. U for-petlji imamo broj iteracija jednak broju vrhova, a služi za pozivanje funkcije `hamiltonianRecursion()`. U toj funkciji pokušavamo dobiti ciklus koji obuhvaća sve vrhove tako što pozivamo funkciju `addVertex()` koja vraća *true* ako je poslani parametar vrh koji je susjedan vrhu koji smo zadnje upisali u pomoćno polje i ako taj vrh nismo već „prošli“. Nakon dodavanja vrha, funkcija `hamiltonianRecursion()` poziva samu sebe. Ako ne dođe do zadnjeg vrha, vraća *false*. Slika 4.1.7 prikazuje traženje Hamiltonovog ciklusa na primjeru grafa. Rekurzija vraća *false* i graf nije hamiltonovski.



Slika 4.1.7 Postupak traženja Hamiltonovog ciklusa

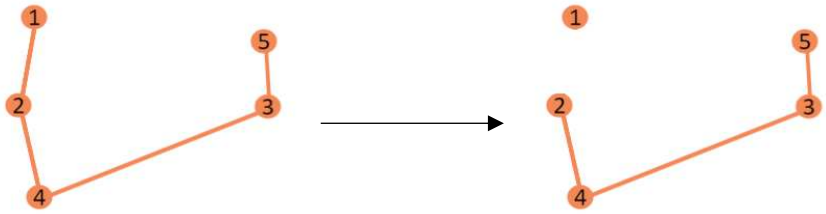
Paralelno s traženjem Hamiltonovog ciklusa, provjerit ćemo i semihamiltonovost. Zato imamo for-petlju kojom provjeravamo postoji li put koji obuhvaća sve vrhove tako što u svakoj iteraciji krećemo od sljedećeg vrha i pozivamo `hamiltonianRecursion()`. Ako smo nakon dodavanja vrhova u pomoćno polje došli do zadnjeg vrha, provjeravamo je li on spojen s prvim:

- ako nije, graf nije hamiltonovski, ali jest semihamiltonovski.
- ako jest, provjeravamo ima li graf više od dva vrha:
    - ako ima, graf je hamiltonovski i nije semihamiltonovski.
    - ako graf ima samo dva vrha, provjeravamo jesu li spojeni s više od jednog brida:
        - ako jesu, graf je hamiltonovski i nije semihamiltonovski.
        - ako nisu, graf nije hamiltonovski i jest semihamiltonovski.

4.1.2.5 Provjera mostova

Nakon toga provjeravamo ima li graf mostove pozivom funkcije `bridgeRecursion()` iz datoteke *checkBridge.js*. Stvaramo pomoćnu matricu `tempArr` u koju upisujemo vrijednosti elemenata matrice susjedstva. Ovisno o povezanosti grafa, tražimo mostove na sljedeći način:

- ako je graf povezan, pozivamo funkciju `checkBridge()`. Unutar te funkcije prolazimo kroz matricu susjedstva, brišemo brid na koji prvi naiđemo (mijenjamo vrijednost elementa u matrici susjedstva) i provjeravamo je li takav graf (s promijenjenom matricom) povezan. Ako nije, našli smo most. Ako jest, vraćamo staru vrijednost u promijenjeni element matrice i brišemo drugi brid te ponavljamo postupak dok ne nađemo most. Ako dođemo do kraja matrice, zaključujemo da nema mosta. Slika 4.1.8 prikazuje pronalazak mosta na povezanom grafu.

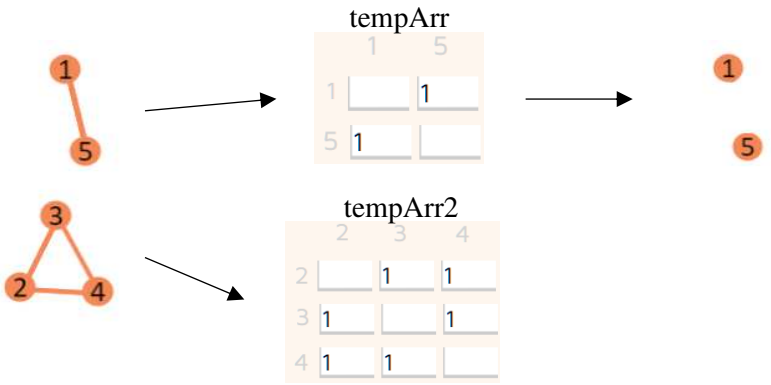


Slika 4.1.8 Postupak traženja mosta na povezanom grafu

- ako graf nije povezan, definiramo pomoćno polje `verticesArray` dimenzije broja vrhova i svim elementima dodjeljujemo vrijednost nula. Kako budemo posjećivali vrhove, mijenjat ćemo vrijednosti pripadajućih elemenata polja u jedan. Krećemo od prvog vrha i tražimo vrhove koji su povezani s njim. Kada prođemo cijelu matricu, definiramo dvije nove matrice, `tempArr` i `tempArr2`:
    - prva matrica se sastoji od vrhova koji su povezani s prvim vrhom originalne matrice susjedstva (to su vrhovi koji su predstavljeni elementima polja `verticesArray` koji imaju vrijednost jedan).

- druga matrica se sastoji od ostalih vrhova originalne matrice susjedstva (onih vrhova koji su predstavljeni elementima polja `verticesArray` koji imaju vrijednost nula).

Slika 4.1.9 prikazuje slučaj traženja mosta u nepovezanom grafu. Pozivamo funkciju `checkBridge()` kojoj kao parametar dajemo matricu `tempArr`. Ako je most pronađen, izlazimo uz funkcije `bridgeRecursion()`. Ako nije, funkcija `bridgeRecursion()` poziva samu sebe i kao parametar uzima matricu `tempArr2` i cijeli postupak se ponavlja dok ne nađemo most ili dođemo do kraja originalne matrice susjedstva, kada zaključujemo da graf nema most.



Slika 4.1.9 Postupak traženja mosta na nepovezanom grafu

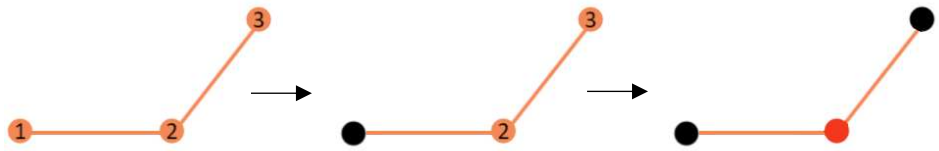
4.1.2.6 Provjera kromatskog broja

Jedno od svojstava koje tražimo je i kromatski broj grafa. Funkcija `checkChromaticNum()` sadrži for-petlju s brojem iteracija jednakim broju vrhova grafa (jer je to najveći mogući kromatski broj za taj graf). U svakoj iteraciji se poziva funkcija `isChromaticNum()` kojoj se kao parametar daje redni broj iteracije. Unutar te funkcije definiramo varijablu `tempArr` koja služi kao pomoćno polje dimenzije broja vrhova te se svakom elementu dodjeljuje vrijednost nula. Poziva se funkcija `chromaticNumRecursion()`. Ona kao parametre uzima broj za koji ispitujemo je li kromatski, pomoćno polje `tempArr` čiji elementi predstavljaju vrhove kojima dodjeljujemo boje te vrh koji pokušavamo obojiti. U for-petlji pozivamo

funkciju `addColorNum()` kojom želimo dodijeliti boju trenutnom vrhu (boja se može dodijeliti vrhu ako nije već dodijeljena susjednom vrhu):

- ako uspijemo, funkcija `chromaticNumRecursion()` poziva samu sebe za sljedeći vrh. Ako obojimo sve vrhove, funkcija vraća *true* i pronađen je kromatski broj. Funkcija `isChromaticNum()` vraća *true* i unutar funkcije `checkChromaticNum()` se izlazi iz for-petlje.
- ako nismo uspjeli obojiti vrh, idemo u sljedeću iteraciju for-petlje kojom se vrh pokušava obojiti drugom bojom ponovnim pozivom funkcije `addColorNum()`, a maksimalni broj iteracija je jednak broju za koji ispitujemo je li kromatski. Ako nikako ne uspijemo obojiti vrh, `chromaticNumRecursion()` vraća *false*, `isChromaticNum()` vraća *false*, a unutar funkcije `checkChromaticNum()` idemo u sljedeću iteraciju for-petlje kojom provjeravamo je li sljedeći broj jednak kromatskom broju grafa.

Slika 4.1.10 prikazuje postupak bojenja vrhova.

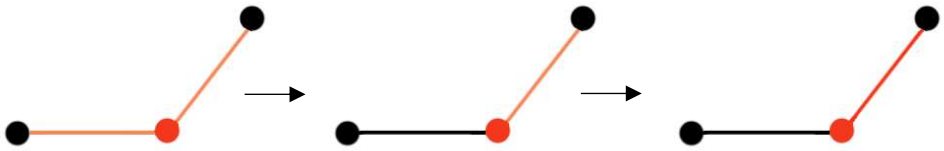


Slika 4.1.10 Postupak traženja kromatskog broja

4.1.2.7 Provjera kromatskog indeksa

Prije traženja kromatskog indeksa, pozivamo funkciju `drawEdges()` kojom ćemo nacrtati bridove grafa, a onda ćemo ih kasnije, paralelno s traženjem kromatskog indeksa, bojiti. Pozivamo funkciju `checkChromaticIndex()`. Prije toga smo definirali varijablu `edgeMatrix` kao matricu susjedstva bridova grafa. To jest, elementu *ij* matrice susjedstva bridova je pridružena vrijednost jedan ako je *i*-tom bridu susjedan *j*-ti brid. Kromatski indeks ćemo naći pronalaskom kromatskog broja matrice susjedstva bridova. Funkcije koje koristimo su analogne onima za traženje kromatskog

broja, uz to što u funkciji `chromaticIndRecursion()` dodjeljujemo boje bridovima i prikazujemo ih na ekranu (Slika 4.1.11).



Slika 4.1.11 Postupak traženja kromatskog indeksa

Korisniku se osim prikazanog grafa s obojenim vrhovima i bridovima prikazuje i rezultat poziva funkcije `calculate()`, tj. sva ispitana svojstva.

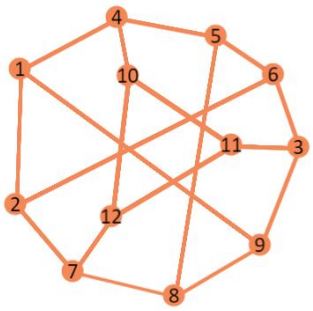
4.2. Upute za korištenje

Pokretanjem aplikacije, korisniku je prikazana stranica s naslovom „Ispitivanje svojstava grafa“. Ostatak stranice podijeljen je u dva dijela: lijevi dio služi za prikaz grafa, a desni za prikaz matrice susjedstva i ispitanih svojstava. (Slika 4.2.1).

Slika 4.2.1 Izgled stranice nakon pokretanja aplikacije

Klikom na gumb „Dodaj vrhove“, otključava se mogućnost crtanja (dodavanja vrhova klikom miša) na lijevoj strani. Korisnik dodaje sve vrhove grafa kojem želi ispitati svojstva. Na svakom vrhu se ispisuje njegov redni broj radi lakšeg snalaženja. Klikom

na gumb „Dodaj bridove“, korisnik može klikom miša ucrtavati bridove i to na način da prvo klikne na jedan, a zatim na drugi vrh s kojim će željeni brid biti incidentan. Brid se odmah pojavljuje na ekranu (Slika 4.2.2).



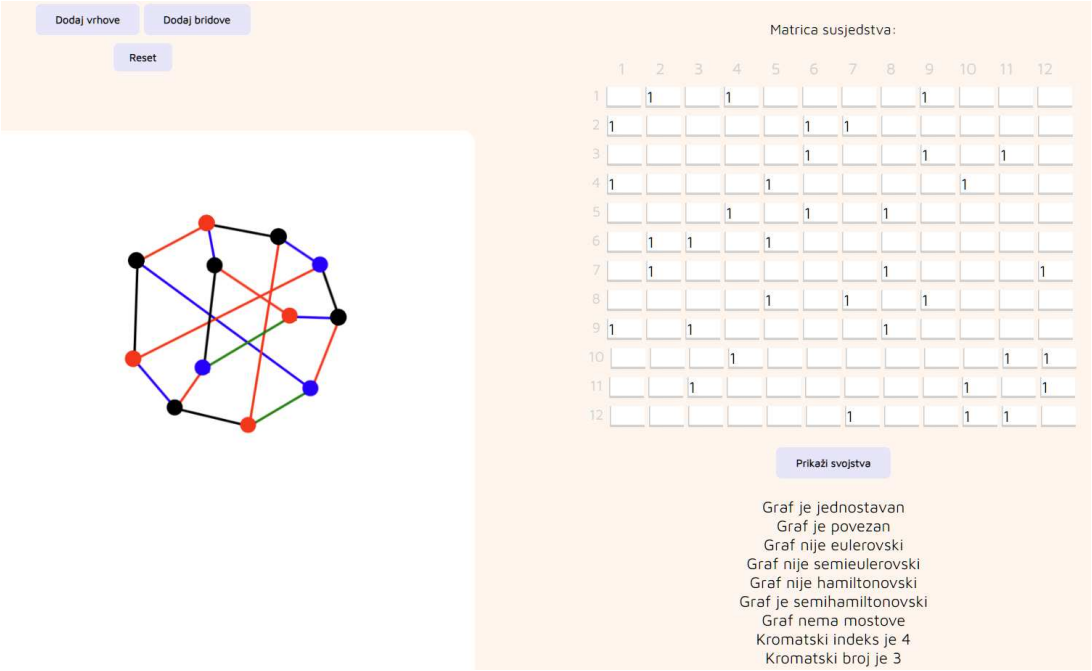
Slika 4.2.2 Izgled prostora za crtanje nakon ucrtavanja grafa

Klikom na gumb „Dodaj bridove“, korisniku se s desne strane prikazuje prazna matrica koju ispunjava na sljedeći način: element *ij* matrice predstavlja brid između vrhova *i* i *j* te mora imati vrijednost koja odgovara broju bridova između dva vrha. Ako između dva vrha nema bridova, može se upisati nula ili ostaviti prazno. Ako neki vrh i ima petlje, u element matrice *ii* se upisuje vrijednost dvostruko veća od broja petlji za taj vrh. Nakon klika na gumb „Prikaži svojstva“, aplikacija upozorava korisnika na netočan unos ako matrica nije simetrična, ako je vrijednost elementa na mjestu *ij* gdje je $i = j$ neparan te ako je unesena vrijednost koja nije cijeli broj.

Dakle, korisnik može unijeti bridove grafa izravnim ucrtavanjem na lijevu stranu ekrana klikom miša na odabrane vrhove, ili ispunjavajući matricu susjedstva. Korisnik može i kombinirati ta dva načina.

Nakon ispravnog unošenja vrijednosti elemenata u matricu, korisnik klikom na gumb „Prikaži svojstva“ pokreće učitavanje matrice u program i testiranje traženih svojstava. Ako se radi o jako složenom grafu (s velikim kromatskim indeksom), moguće je da će program trebati nekoliko sekundi za provjeru svih svojstava. Nakon izvođenja programa, korisnik na lijevoj strani može vidjeti graf kojem su vrhovi i bridovi obojeni u skladu s pronađenim vrijednostima za kromatski broj i kromatski indeks. Na desnoj

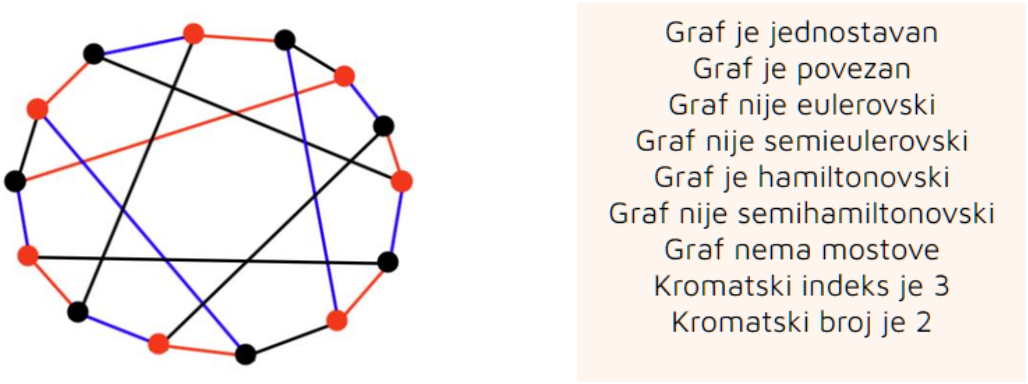
strani i dalje vidi matricu susjedstva ispod koje se nalaze rezultati ispitivanja (Slika 4.2.3).



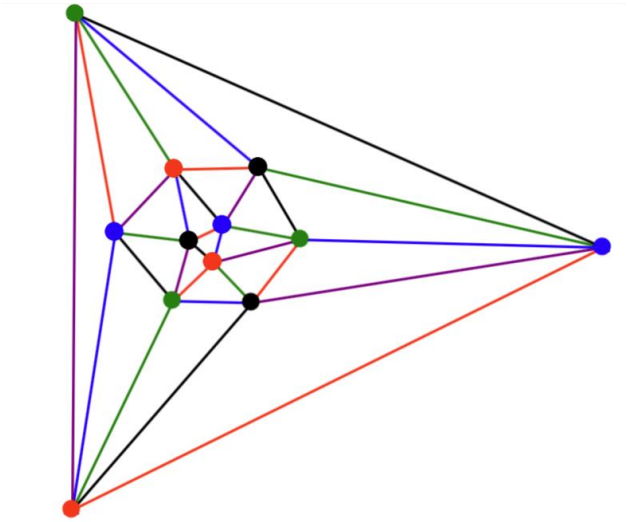
Slika 4.2.3 Rezultat provjere svojstava

4.3. Testiranje

Na sljedećim slikama (Slika 4.3.1, Slika 4.3.2, Slika 4.3.3, Slika 4.3.4) prikazani su primjeri grafova i rezultati ispitivanja njihovih svojstava.

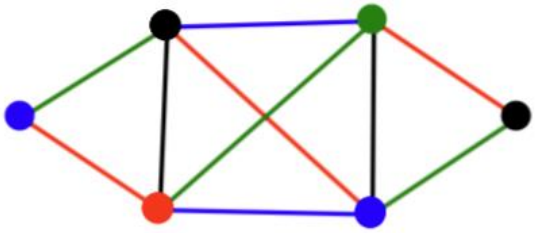


Slika 4.3.1 Heawoodov graf i njegova svojstva



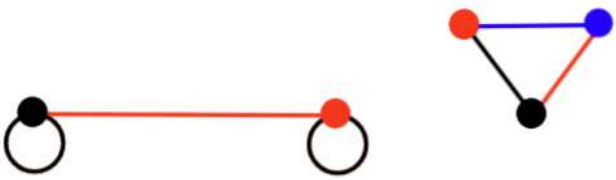
Graf je jednostavan
Graf je povezan
Graf nije eulerovski
Graf nije semieulerovski
Graf je hamiltonovski
Graf nije semihamiltonovski
Graf nema mostove
Kromatski indeks je 5
Kromatski broj je 4

Slika 4.3.2 Ikosaedar i njegova svojstva



Graf je jednostavan
Graf je povezan
Graf je eulerovski
Graf nije semieulerovski
Graf je hamiltonovski
Graf nije semihamiltonovski
Graf nema mostove
Kromatski indeks je 4
Kromatski broj je 4

Slika 4.3.3 Primjer eulerovskog grafa



Graf nije jednostavan
Graf nije povezan
Graf nije eulerovski
Graf nije semieulerovski
Graf nije hamiltonovski
Graf nije semihamiltonovski
Graf ima mostove
Kromatski indeks je 3
Kromatski broj je 3

Slika 4.3.4 Primjer nepovezanog grafa

Zaključak

Cilj završnog rada bio je opisati svojstva grafova te napraviti računalnu aplikaciju za njihovo ispitivanje.

Definirani su sljedeći pojmovi: jednostavnost, povezanost, eulerovost, hamiltonovost, mostovi, kromatski broj i kromatski indeks grafa. Izrađena je interaktivna aplikacija koja omogućava korisniku provjeru navedenih svojstava. Korisnik unosi vrhove i bridove grafa, nakon čega se određuju njegove značajke, a rezultat se prikazuje na ekranu. Vrhovi i bridovi su obojeni u skladu s pronađenim kromatskim brojem i kromatskim indeksom grafa.

Aplikacija ima ograničenja s obzirom na broj vrhova i bridova. Za grafove s maksimalnim stupnjem vrha manjim od sedam, aplikacija bi trebala dati rješenje za manje od dvije sekunde. Za složenije grafove, izvođenje programa može trajati više od minute. Na brzinu programa najviše utječu rekurzivne funkcije za traženje kromatskog broja i kromatskog indeksa. S obzirom da je korišteno iscrpno pretraživanje prolazeći po vrhovima po njihovom rednom broju, sam raspored ucrtanih vrhova utječe na brzinu pronalaska kromatskog broja grafa.

Literatura

[1] Veljan, D. *Kombinatorna i diskretna matematika*. 1. izdanje. Zagreb: Algoritam, 2001.

[2] Lint, J. H., Wilson, R. M. *A Course in Combinatorics*. 2. izdanje. Cambridge: Cambridge University Press, 2001.

[3] Kovačević, D., Krnić, M., Nakić, A., Pavčević, M. O. *Diskretna matematika 1*. Zagreb: FER, 2020.

[4] Naglić, V. *Osnovi teorije mreža*. Zagreb: Sveučilišna naklada Liber, 1988.

[5] Majstorović, S., Boras, L. *Petersenov graf*. Math.e: Hrvatski matematički elektronički časopis, (2015). Poveznica: https://hrcak.srce.hr/file/212654; pristupljeno 12. siječnja 2024.

[6] Burch, K. J. *Chemical applications of graph theory*, Mathematical Physics in Theoretical Chemistry, (2019.), str. 261-294. Poveznica: https://e-tarjome.com/storage/panel/fileuploads/2019-08-26/1566803663_E12926-e-tarjome.pdf

[7] Slika 2.3.1 – Lint, J. H., Wilson, R. M. *A Course in Combinatorics*. 2. izdanje. Cambridge: Cambridge University Press, 2001., str. 6

Sažetak

Računalna aplikacija za određivanje značajki grafa

Teorija grafova ima razne primjene, kako u matematici, tako i u drugim područjima. Neka od najvažnijih svojstava grafova su jednostavnost, povezanost, eulerovost, hamiltonovost, postojanje mostova, kromatski broj i kromatski indeks. Cilj ovog završnog rada bio je proučiti i opisati navedena svojstva i napraviti aplikaciju za njihovo ispitivanje.

Prvo je svaka od ispitivanih značajki detaljno objašnjena te su dani primjeri grafova. Dokazani su najvažniji teoremi. Navedene su neke primjene teorije grafova, npr. u kemiji i elektrotehnici. Izrađena je interaktivna računalna aplikacija koja omogućava korisniku određivanje značajki grafa. Za izradu aplikacije korišten je JavaScript programski jezik i Visual Studio Code uređivač koda. Na kraju je opisan način rada programa i dane su upute za korištenje.

Ključne riječi: graf, jednostavan, povezan, Euler, Hamilton, kromatski broj, kromatski indeks, aplikacija, JavaScript

Summary

A Computer Application for Determining Graph Features

Graph theory has various applications, both in mathematics and in other fields. Some of the most important properties of graphs are simplicity, connectivity, Eulerianity, Hamiltonianity, existence of bridges, chromatic number and chromatic index. The goal of this undergraduate thesis was to study and describe the mentioned properties and to create an application for their verification.

First, each of the examined features was thoroughly explained, accompanied by examples of graphs. The most important theorems were proven. Some applications of graph theory were mentioned, such as in chemistry and electrical engineering. An interactive computer application was developed, enabling users to determine the features of a graph. JavaScript programming language and Visual Studio Code editor were used for the application development. The program's functionality and instructions for usage were described at the end.

Keywords: graph, simple, connected, Euler, Hamilton, chromatic number, chromatic index, application, JavaScript