



# Sadržaj

<b>Uvod</b>	3
<b>1. Uvod u teoriju grafova</b>	5
1.1. Osnovni pojmovi	5
1.2. Vrste grafova	6
<b>2. Izomorfnost grafova</b>	10
2.1. Definicija izomorfizma	10
2.2. Složenost problema ispitivanja izomorfnosti grafova	12
<b>3. Algoritmi za ispitivanje Izomorfnosti grafova</b>	13
3.1. Algoritam baziran na invarijantama	13
3.2. Algoritam baziran na konstrukciji certifikata grafa	16
3.2.1. Stabla	17
3.2.2. Jednostavni grafovi	20
<b>4. Certifikati grafova s pet vrhova</b>	29
4.1. Nepovezani grafovi s pet vrhova	29
4.2. Stabla s pet vrhova	30
4.3. Povezani grafovi s pet vrhova koji nisu stabla	31
<b>5. Opis izrađene aplikacije</b>	34
5.1. Grafički unos grafova	35
5.2. Matrični unos grafova	37
<b>6. Analiza i evaluacija</b>	41
6.1. Algoritam za izračun cerifikata stabla	41

6.2. Algoritam za izračun certifikata jednostavnog grafa . . . . .	43
<b>Zaključak</b> . . . . .	<b>46</b>
<b>Literatura</b> . . . . .	<b>47</b>
<b>Sažetak</b> . . . . .	<b>49</b>
<b>Abstract</b> . . . . .	<b>50</b>

# Uvod

Grafovi su snažan alat u matematici koji pomaže u modeliranju i analizi mnogih struktura i veza koje se pojavljuju u tehnološkim i društvenim problemima. Grafovi mogu učinkovito predstavljati probleme vezane uz računalne mreže, društvene mreže, prometne sustave, biološke procese i brojna druga polja. Grafovi su postali ključni u istraživanju i razvoju računalnih algoritama zbog njihove sposobnosti da komplikirane sisteme vjerno predstavljaju na jednostavan i intuitivan način. Oni olakšavaju razumijevanje povezanosti između različitih elemenata i prepoznavanje obrazaca koji nisu odmah vidljivi.

Jedan od glavnih problema u teoriji grafova je odgovaranje na pitanje jesu li dva grafa koja različito izgledaju zapravo strukturno ista. U prijevodu, utvrđivanje postojanja, odnosno nepostojanja izomorfizma među grafovima. Drugim riječima, može li se jedan graf presložiti u drugi tako da i dalje sadrži sve inicijalne veze među vrhovima. Ovaj problem ima široku primjenu u raznim područjima, poput kemije, gdje se uspoređuju molekularne strukture, ili računalne vizije, gdje se koristi za prepoznavanje oblika i objekata.

No ispitivanje izomorfnosti grafova je težak kombinatorni problem. Tradicionalni pristup jednostavne iscrpne pretrage nije dovoljno brz i učinkovit kada broj vrhova i bridova postane velik, što može dovesti do dugog vremena izvođenja ili velikih zahtjeva za računalnim resursima. To postaje još veći izazov u stvarnim situacijama gdje trebamo brzo analizirati velike količine podataka, poput analiza društvenih mreža ili bioloških sustava. Zato se sve više pažnje i resursa posvećuje razvoju boljih i bržih algoritama koji mogu pomoći u rješavanju ovog problema.

U ovom radu istraženi su razni pristupi za ispitivanje izomorfnosti grafova, s velikim naglaskom na algoritam koji koristi certifikate grafova za utvrđivanje izomorfnosti. Certifikat se može protumačiti kao potpis grafa koji se koristi za međusobno uspoređivanje

grafova. Sve faze algoritma koji koristi certifikate detaljno su objašnjene, te su prikazane na primjeru kako bi čitatelj što bolje razumio postupak.

U sklopu rada razvijena je računalna aplikacija koja korisnicima omogućuje ispitivanje izomorfnosti grafova. Razvijena računalna aplikacija podržava unos grafova s do uključno 8 vrhova, a njezina učinkovitost analizirana je kroz performanse, s naglaskom na brzinu izvođenja u odnosu na broj vrhova.

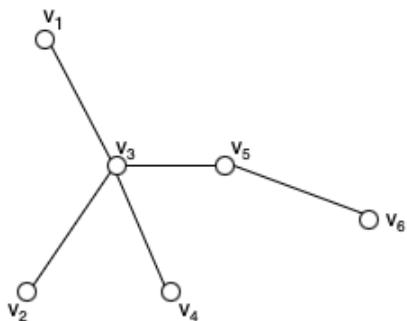
Dodatno, izrađena je klasifikacija grafova s pet vrhova te je na tim grafovima analizirana učinkovitost algoritma obrzirom na složenost povezivanja unutar grafa.

# 1. Uvod u teoriju grafova

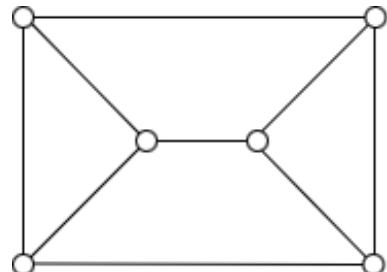
Teorija grafova je matematička disciplina koja se bavi proučavanjem grafova i struktura koje predstavljaju skupove objekata povezanih određenim vezama. Grafovi su korisni u mnogim znanstvenim i tehničkim područjima jer omogućuju modeliranje različitih odnosa među objektima.

## 1.1. Osnovni pojmovi

**Definicija 1.** *Graf  $G$  je trojka koja se sastoji od skupa vrhova  $V(G)$ , skupa bridova  $E(G)$ , te relacije koja povezuje svaki brid s dva vrha (ne nužno različita), koja nazivamo vrhom brida.* [1]



Slika 1.1. Graf  $E_1$



Slika 1.2. Graf  $E_2$

Ovisno o vrsti grafa, vrhovi brida mogu, ali i ne moraju biti različiti, što znači da brid može povezivati vrh i sam sa sobom, u tom slučaju govorimo o petlji.

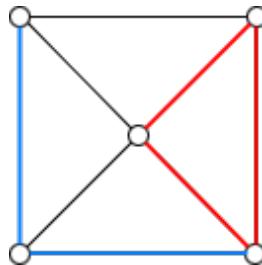
Neki od osnovnih pojmoveva u teoriji grafova koji će biti bitni za razumijevanje ovog rada navedeni su u nastavku.

Ako brid  $e$  spaja vrhove  $v$  i  $w$ , kažemo da su vrhovi  $v$  i  $w$  susjedni te kažemo da je vrh  $v$  incidentan s bridom  $e$ . Stupanj vrha  $v$  grafa  $G$  je broj bridova od  $G$  koji su incidentni s

v. Šetnja u grafu je niz

$$(v_0, e_1, v_1, e_2, v_2, \dots, e_n, v_n),$$

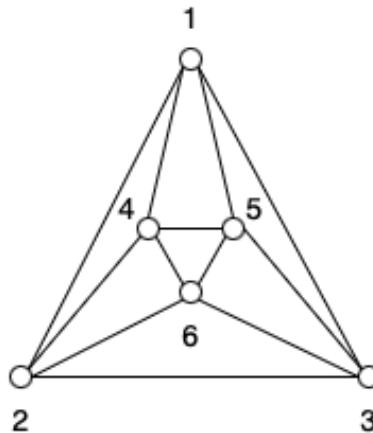
gdje je  $e_i$  brid  $\{v_{i-1}, v_i\}$ , za  $i = 1, \dots, n$ . Put je šetnja u kojoj su svi vrhovi različiti (osim eventualno prvog i zadnjeg). Zatvoreni put zovemo *ciklus*. Vrhovi stupnja jedan zovu se *listovi*.



Slika 1.3. Graf H

Na grafu  $H$  sa slike 1.3. crvenom bojom je označen primjer ciklusa u grafu, a plavom bojom put koji nije zatvoren.

Za graf  $G$  sa slike 1.4., vrhovi 1 i 2 su susjedni, a vrh 1 je incidentan s bridom  $\{1, 2\}$ . Stupanj vrha 1 iznosi 4. Ovaj graf nema niti jedan list, dok graf  $E_1$  ima četiri lista i to vrhovi  $v_1, v_2, v_4, v_6$ .

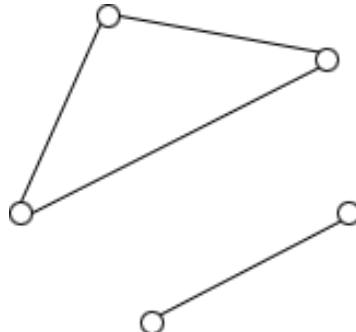


Slika 1.4. Graf G

## 1.2. Vrste grafova

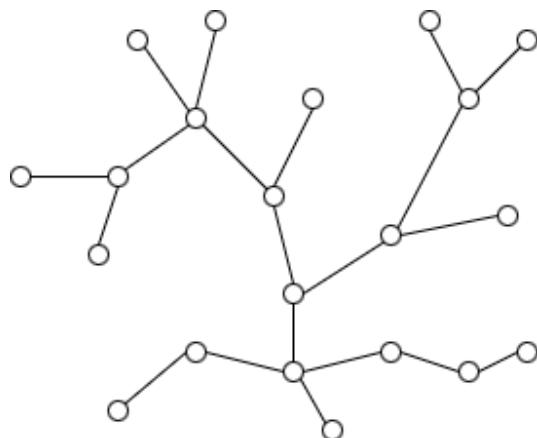
Grafovi se mogu kategorizirati prema različitim kriterijima, a najvažniji za ovaj rad predstavljeni su u nastavku.

Graf  $G$  je *povezan* ako svaki par vrhova u  $G$  pripada nekom putu; u suprotnom, graf  $G$  je *nepovezan*.



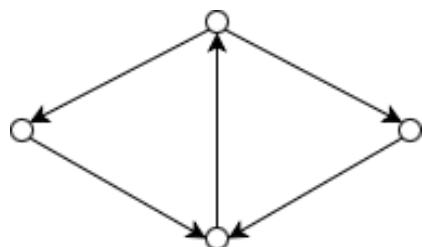
**Slika 1.5.** Primjer jednog nepovezanog grafa

*Stablo* je povezan graf bez ciklusa.



**Slika 1.6.** Primjer jednog stabla

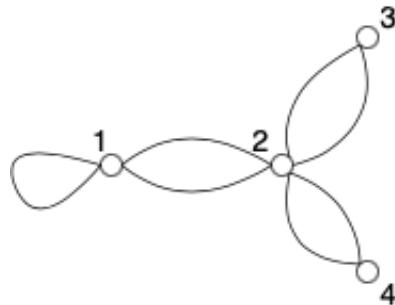
*Usmjereni bridovi* su bridovi koji imaju orijentaciju tako da idu od jednog vrha prema drugome. Graf koji ima usmjerenje bridova zove se *usmjereni graf* ili *digraf*. *Neusmjereni graf* je graf koji nema usmjerenje bridova.



**Slika 1.7.** Primjer usmjerenog grafa

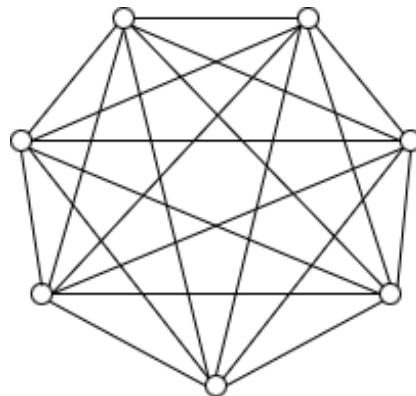
*Multigraf* je graf koji može sadržavati višestruke bridove između dva vrha.

Graf koji nema bridove koji spajaju vrh sa samim sobom (petlje) niti višestruke bridove naziva se *jednostavni graf*, u suprotnom naziva se *višestruki graf*.



**Slika 1.8.** Primjer jednog višestrukog grafa

*Potpuni graf*  $K_n$  je graf s  $n$  vrhova u kojem skup bridova  $E$  sadrži sve moguće dvočlane podskupove skupa vrhova  $V$ .



**Slika 1.9.** Potpuni graf  $K_7$

Graf s  $n$  vrhova i bez bridova zove se *nul graf*  $N_n$ .



**Slika 1.10.** Nul graf  $N_3$

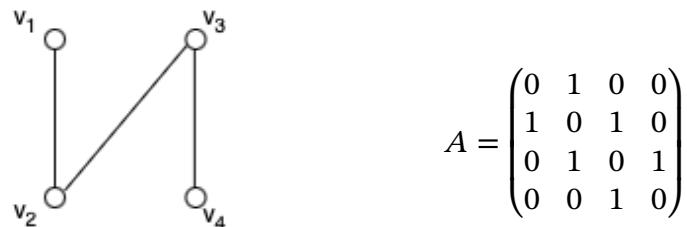
U ostatku rada, pretpostavlja se da su svi grafovi jednostavni i neusmjereni te to neće biti posebno naglašavano.

Bitno je napomenuti da se graf, osim grafički, može prikazati i pomoću matrice susjedstva. Matrica susjedstva daje informacije o povezanim vrhovima u grafu. Redovi i

stupci matrice  $A = [a_{ij}]$  prestavljanju vrhove grafa. Ako je  $a_{ij} = 0$ , između vrhova  $v_i$  i  $v_j$  ne postoji brid, a ako je  $a_{ij} = 1$  onda taj brid postoji. Na glavnoj dijagonali se uvijek nalaze nule ako je promatrani graf jednostavan. Matrica susjedstva je uvijek simetrična jer ako postoji brid između vrhova  $v_i$  i  $v_j$ , onda postoji i vrh između vrhova  $v_j$  i  $v_i$ . Slijedi formalna definicija matrice susjedstva te primjer za graf  $G$  sa slike 1.11.

**Definicija 2.** Neka su vrhovi numerirani s  $1, 2, \dots, |V|$  na proizvoljan način. *Matrica susjedstva* grafa  $G = (V, E)$  je matrica  $A = (a_{ij})$  dimenzije  $|V| \times |V|$  definirana na sljedeći način: [2]

$$a_{ij} = \begin{cases} 1 & \text{ako je } (i, j) \in E, \\ 0 & \text{inače.} \end{cases}$$



Slika 1.11. Graf  $G$

## 2. Izomorfnost grafova

### 2.1. Definicija izomorfizma

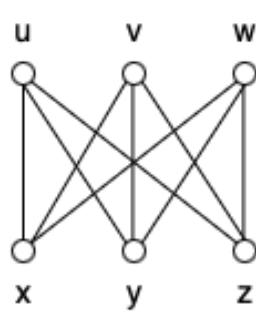
Problem izomorfnosti grafova istražuje kada su dva grafa strukturno identična. Formalna definicija slijedi.

**Definicija 3.** Grafovi  $\mathcal{G}_1 = (V_1, E_1)$  i  $\mathcal{G}_2 = (V_2, E_2)$  su *izomorfni* ako postoji bijekcija  $f : V_1 \rightarrow V_2$  takva da

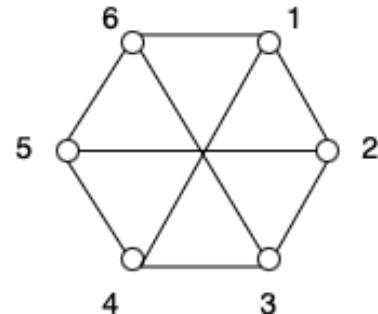
$$\{f(x), f(y)\} \in E_2 \text{ ako i samo ako } \{x, y\} \in E_1.$$

Preslikavanje  $f$  se naziva *izomorfizam* između  $\mathcal{G}_1$  i  $\mathcal{G}_2$ .

Ključna karakteristika izomorfizma je očuvanje strukture, odnosno bez obzira na grafički prikaz grafa ili imena vrhova, njihova povezanost ostaje nepromijenjena. Na primjer, ako su dva vrha povezana bridom u prvom grafu, njihovi odgovarajući vrhovi moraju biti povezani bridom u drugom grafu.



Slika 2.1. Graf  $G_1$

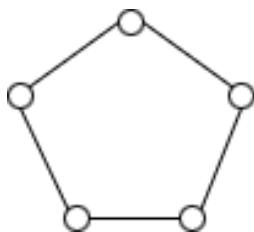


Slika 2.2. Graf  $G_2$

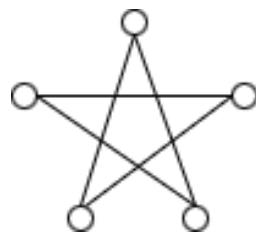
Da bi se pokazalo da su grafovi  $G_1$  i  $G_2$  sa slika 2.1. i 2.2. izomorfni, potrebno je pronaći odgovarajuću bijekciju. Neka je bijekcija  $f : V(G_1) \rightarrow V(G_2)$  definirana kao:

$$f(u) = 1, \quad f(v) = 3, \quad f(w) = 5, \quad f(x) = 2, \quad f(y) = 4, \quad f(z) = 6$$

Bijekcija  $f$  je izomorfizam između  $G_1$  i  $G_2$  jer svaki brid između čvorova u  $G_1$  odgovara bridu između odgovarajućih čvorova u  $G_2$ . Na primjer, ako je  $(u, y) \in E(G_1)$ , tada je  $(f(u), f(y)) = (1, 4) \in E(G_2)$ , i obrnuto. Analogno može se pokazati da to vrijedi za sve ostale bridove u grafovima, čime je očuvana relacija susjedstva. Može se naći i druga bijekcija koja čuva relacije susjedstva, no da bi se dokazalo da su grafovi izomorfni dovoljno je pronaći jednu takvu bijektivnu funkciju.

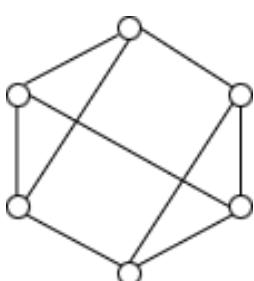


Slika 2.3. Graf  $G_3$

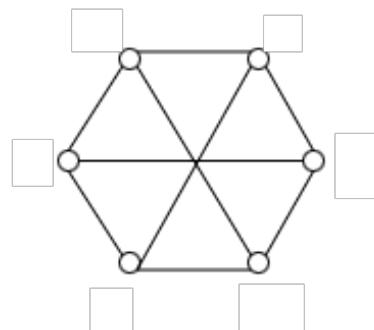


Slika 2.4. Graf  $G_4$

Za neke parove grafova je već na prvi pogled jasno da između njih postoji izomorfizam. Takav je slučaj s grafovima  $G_3$  i  $G_4$  sa slika 2.3. i 2.4. Za već spomenute grafove  $G_1$  i  $G_2$ , izomorfnost nije odmah očita, ali može se naslutiti jer oba grafa imaju jednak broj vrhova, bridova i stupnjeve vrhova.



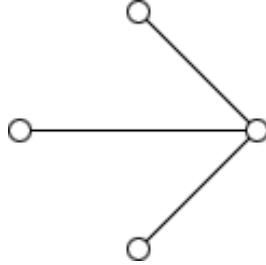
Slika 2.5. Graf  $G_6$



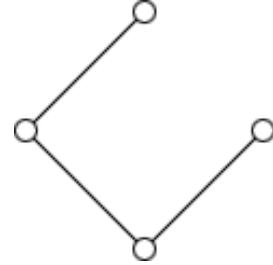
Slika 2.6. Graf  $G_7$

Na slikama 2.5. i 2.6. prikazani su neizomorfni grafovi  $G_6$  i  $G_7$ . Oni imaju jednak broj vrhova, bridova i iste stupnjeve vrhova, a to može navesti na pogrešan zaključak da su izomorfni. Zaključak da grafovi nisu izomorfni može se donijeti analizom njihovih ciklusa. Graf  $G_6$  sadrži ukupno 8 različitih ciklusa, dok ih graf  $G_7$  ima 7, što pokazuje da

oni nemaju istu strukturu. S druge strane, za grafove  $G_8$  i  $G_9$ , prikazane na slikama 2.7. i 2.8. očito je da nisu izomorfni, a to se vidi iz različitih stupnjeva njihovih vrhova.



Slika 2.7. Graf  $G_8$



Slika 2.8. Graf  $G_9$

## 2.2. Složenost problema ispitivanja izomorfnosti grafova

Da bi se mogla analizirati složenost problema ispitivanja izomorfnosti grafova, uvode se iduće definicije.

**Definicija 4.** *Klasa P problema* obuhvaća one probleme koji su rješivi u polinomijalnom vremenu. To su problemi koji se mogu riješiti algoritmom čije vrijeme izvršavanja raste polinomijalno s veličinom ulaznih podataka  $O(n^k)$  gdje je  $k$  konstanta, a  $n$  veličina ulaza.

**Definicija 5.** *Klasa NP problema* obuhvaća probleme za koje se rješenje može provjeriti u polinomijalnom vremenu. To znači da, ako nam je ponuđeno potencijalno rješenje problema, možemo ga provjeriti u vremenu koje raste polinomijalno s veličinom ulaznih podataka.

**Definicija 6.** *NP-kompletni problemi* su najteži problemi u klasi NP zato jer ako pronađemo rješenje u polinomijalnom vremenu za neki NP-kompletan problem, tada možemo riješiti sve NP probleme u polinomijalnom vremenu.

NP-kompletni problemi su oni problemi za koje se vjeruje da je teško pronaći rješenje, no lako je provjeriti potencijalno rješenje. Problem ispitivanja izomorfnosti grafova pripada skupini problema čija je složenost neodređena. Dok se za specifične vrste grafova, poput stabala, ispitivanje izomorfnosti može riješiti u polinomijalnom vremenu, za općenite grafove problem još nije potvrđen kao NP-kompletan niti je dokazano rješiv u polinomijalnom vremenu. Zbog navedenog, općenito, problem određivanja izomorfnosti grafova je izazovan za matematičare i računalne stručnjake. [2]

## 3. Algoritmi za ispitivanje izomorfnosti grafova

Postoje dva pristupa, odnosno algoritma za ispitivanje izomorfnosti grafova, a razlikuju se po složenosti i efikasnosti. U ovom poglavlju razmotrit će se osnove algoritma za ispitivanje izomorfnosti grafova baziranog na invarijantama, dok će se algoritam koji računa certifikate grafova detaljno obraditi.

### 3.1. Algoritam baziran na invarijantama

Invarijanta grafa se može protumačiti kao svojstvo koje ostaje nepromijenjeno neovisno o načinu na koji je graf prikazan. U prijevodu, ako su dva grafa izomorfna, njihova invarijanta bit će ista, no treba imati na umu da jednaka vrijednost invarijante može biti znak izomorfnosti, ali ga ne garantira. Korištenjem invarijanti se može brzo utvrditi jesu li dva grafa različita, odnosno ako su im invarijante različite onda oni sigurno nisu izomorfni. Invarijante se koriste kako bi se smanjio broj usporedbi te da se brzo otkriju razlike između grafova. Slijedi formalna definicija invarijante.

**Definicija 7.** [3] Neka je  $\mathcal{F}$  obitelj grafova. *Invarijanta* na  $\mathcal{F}$  je funkcija  $\Phi$  s domenom  $\mathcal{F}$  takva da vrijedi:

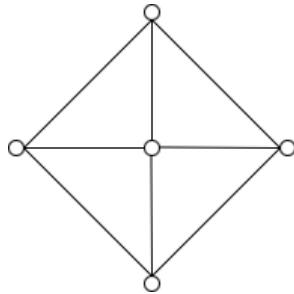
$$\Phi(G_1) = \Phi(G_2), \quad \text{ako je } G_1 \text{ izomorfan s } G_2.$$

Drugim riječima, invarijanta je funkcija koja daje isti rezultat za sve grafove koji su izomorfni.

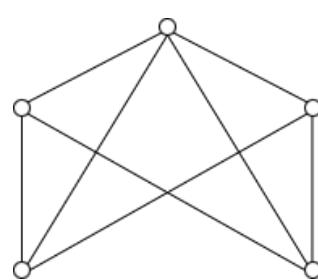
Ako se ustanovi da dva grafa imaju različite vrijednosti za neku invarijantu, algoritam može odmah zaključiti da oni nisu izomorfni i tu algoritam završava. U suprotnom, ako su invarijante jednake, moguće je da su grafovi izomorfni, ali to zahtijeva daljnju provjeru.

Za odabir invarijante postoji mnogo opcija. Potrebno je odabrati invarijantu koja je jednostavna za izračunavanje i koja razlikuje što veći broj grafova. Najčešći primjeri invarijanti korištenih u praksi su sljedeći.

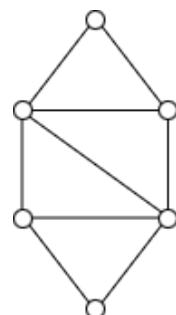
- **Broj trokuta:** Broj trokuta u grafu, odnosno broj podgrafova s tri vrha je također invarijanta [3].



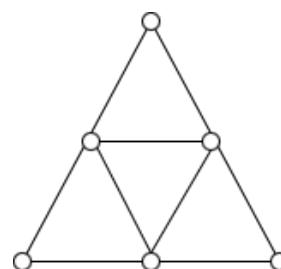
**Slika 3.1.** Graf  $H_1$



**Slika 3.2.** Graf  $H_2$



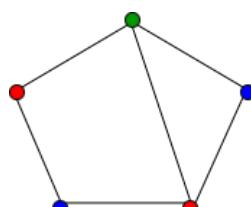
**Slika 3.3.** Graf  $F_1$



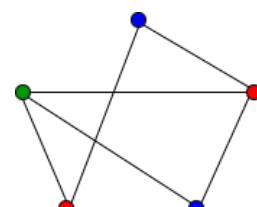
**Slika 3.4.** Graf  $F_2$

Oba grafa sa slika 3.1. i 3.2. imaju četiri podgraфа s tri vrha, odnosno invarijanta im je jednaka i može se pokazati njihova izomorfost. No to što dva grafa imaju jednak invarijant ne znači da su oni nužno izomorfni.  $F_1$  i  $F_2$  sa slika 3.3. i 3.4. imaju jednak broj trokuta, ali oni nisu izomorfni.

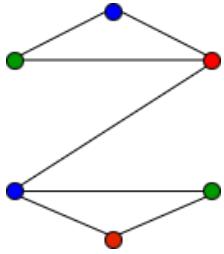
- **Kromatski broj:** Najmanji broj boja potreban za obojanje vrhova grafa tako da nijedan par susjednih vrhova nema istu boju [4].



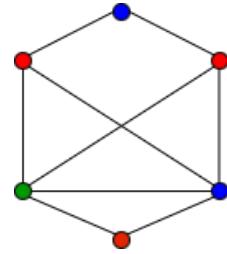
**Slika 3.5.** Graf  $J_1$



**Slika 3.6.** Graf  $J_2$



Slika 3.7. Graf  $I_1$



Slika 3.8. Graf  $I_2$

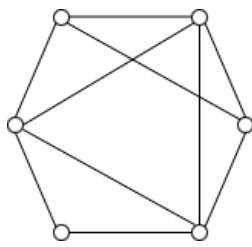
Na slikama 3.5. i 3.6. prikazana su dva izomorfna grafa i jasno se vidi da oni imaju jednak kromatski broj. Grafovi sa slika 3.7. i 3.8. također imaju jednaku invariantu, no oni nisu izomorfni.

- **Distribucija stupnjeva:** Funkcija

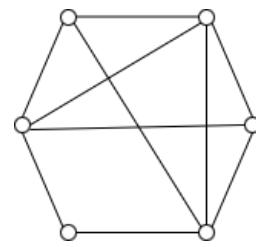
$$\Phi(G) = [|B[0]|, |B[1]|, \dots, |B[n - 1]|]$$

predstavlja invariantu za familiju grafova, gdje je  $|B[i]|$  broj vrhova stupnja  $i$  u grafu  $G$ . Ako su dva grafa s  $n$  vrhova izomorfna, imat će jednak broj vrhova svakog stupnja  $i$ , za svaki  $i = 0, 1, \dots, n - 1$  [3].

Distribucija stupnjeva grafova  $H_1$  i  $H_2$  sa slika 3.1. i 3.2. je jednaka i iznosi  $[0, 0, 0, 4, 1]$  što je i očekivano jer su oni izomorfni. Grafovi sa slika 3.9. i 3.10. nisu izomorfni, ali imaju istu distribuciju vrhova i ona je jednaka  $[0, 0, 1, 2, 3]$ .



Slika 3.9. Graf  $K_1$



Slika 3.10. Graf  $K_2$

Ove invariantne omogućuju algoritmu da brzo eliminira neizomorfne grafove, čime se smanjuje složenost problema i broj usporedbi. Osnovni koraci algoritma za ispitivanje izomorfnosti grafova baziranog na invariantama su sljedeći:

1. **Odabir invariantnih funkcija.** Na početku se odabire skup funkcija koje definiraju karakteristične osobine grafa. Pomoću tih funkcija se grupiraju vrhovi istih

karakteristika.

2. **Inicijalizacija particije.** Svi vrhovi grafa se inicijalno nalaze u jednoj particiji.
3. **Konstrukcija particija.** Koristeći invarijantne funkcije cilj je postupno dijeliti tu particiju u manje particije koristeći vrijednosti invarijantnih funkcija. Vrhovi koji imaju istu vrijednost za odabranu invarijantu ostaju u istom bloku, dok se vrhovi s različitim vrijednostima razdvajaju.
4. **Usporedba particija za oba grafa.** Ako u bilo kojem koraku grupiranje vrhova u  $G_1$  i  $G_2$  nije isto, zaključuje se da grafovi nisu izomorfni.
5. **Provjera izomorfnosti.** Ako su sva grupiranja vrhova ista u oba grafa, tada postoji mogućnost da su grafovi izomorfni. U tom slučaju potrebno je pronaći bijekciju koja čuva strukturu grafa, a to se može postići korištenjem backtracking algoritma. Ako takva bijekcija postoji, grafovi su izomorfni.

Ovaj algoritam je posebno efikasan za velike skupove grafova jer može brzo eliminirati očito neizomorfne grafove, čime se smanjuje opterećenje na računalne resurse i vrijeme izvršavanja.

Primjer algoritma koji koristi invarijante za ispitivanje izomorfnosti grafova je Weisfeiler-Lehman algoritam. WL algoritam iterativno boji vrhove grafa prema bojama njihovih susjeda, čime postupno stvara jedinstvene oznake za vrhove istih karakteristika. Kada se boje prestanu mijenjati, dobiveni obrazac bojenja može poslužiti kao invarijanta. [5]

### 3.2. Algoritam baziran na konstrukciji certifikata grafa

U prethodnom algoritmu izomorfnost dvaju grafova se pokazivala konstruiranjem izomorfizma između njih. Ovaj algoritam preskače tu konstrukciju i pristupa problemu utvrđivanja izomorfnosti na drugačiji način, kroz izradu certifikata. Certifikat se može shvatiti kao jedinstveni potpis grafa. Ako su dva grafa izomorfna, njihovi certifikati će biti isti, dok će se u suprotnom slučaju razlikovati. Formalno, certifikat grafa se definira na sljedeći način.

**Definicija 8.** [3] *Certifikat*  $\text{Cert}()$  za obitelj grafova  $\mathcal{F}$  je funkcija takva da za svaki  $\mathcal{G}_1, \mathcal{G}_2 \in$

$\mathcal{F}$ ,

$$\text{Cert}(\mathcal{G}_1) = \text{Cert}(\mathcal{G}_2) \text{ ako i samo ako su } \mathcal{G}_1 \text{ i } \mathcal{G}_2 \text{ izomorfni.}$$

Iako se algoritam temeljen na invarijantama razlikuje od onog koji se oslanja na certifikate, certifikat se može smatrati posebnom vrstom invarijante [3]. Certifikat grafa se može povezati s pojmom *kanonskog oblika*. Kanonski oblik grafova predstavlja njihov jedinstveni prikaz unutar klase izomorfizma. To znači da, kada dva grafa dijele isti kanonski oblik, oni su izomorfni. Certifikat zapravo služi kao kanonski oblik grafa jer daje jednoznačan potpis svakom grafu u njegovoj klasi izomorfizma [6]. Konstrukcija certifikata ovisi o vrsti grafa, pa se algoritmi razlikuju za stabla i jednostavne grafove.

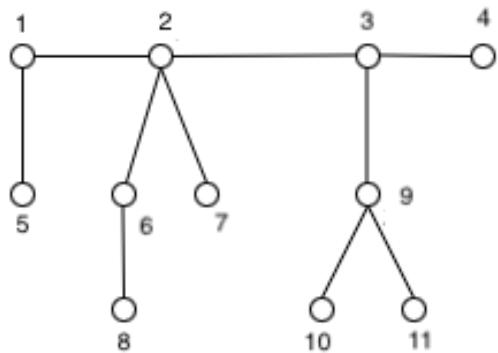
### 3.2.1. Stabla

Kao što je već definirano kod uvoda u teoriju grafova, stablo je povezani graf bez ciklusa te je postupak konstrukcije certifikata relativno jednostavan u odnosu na općeniti graf. Koraci algoritma su sljedeći. [3]

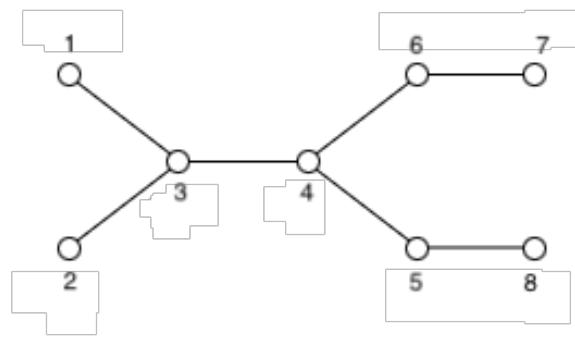
1. Svi vrhovi grafa  $G$  se označavaju oznakom 01.
2. Dok god stablo  $G$  ima više od dva vrha:
  - Za svaki vrh  $x$  u  $G$  koji nije list:
    - (a) Neka je  $Y$  skup koji sadrži oznake listova susjednih vrhu  $x$ , te oznaku vrha  $x$  s uklonjenim početnim 0 i završnim 1.
    - (b) Oznaka vrha  $x$  se zamjenjuje s konkatenacijom oznaka u  $Y$ , sortirano u rastućem leksikografskom poretku, s dodatnim 0 na početku i 1 na kraju.
    - (c) Uklanjaju se svi listovi susjedni vrhu  $x$ .
  - 3. Ako preostane samo jedan vrh  $x$ , oznaka tog vrha se uzima kao certifikat stabla.
  - 4. Ako preostanu dva vrha  $x$  i  $y$ , oznake vrhova  $x$  i  $y$  spojene u rastućem leksikografskom poretku se uzimaju kao certifikat stabla.

Vrhovi iz trećeg i četvrtog koraka algoritma se nazivaju *centar* stabla.

Slijedi prikaz koraka algoritma na primjeru grafova sa jednim i dva centra, sa slika 3.11. i 3.12.



**Slika 3.11.** Stablo s dva centra [7]

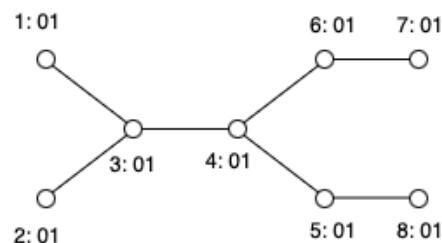


**Slika 3.12.** Stablo s jednim centrom [8]

### Primjer 1. Izračun certifikata za stablo s jednim centrom

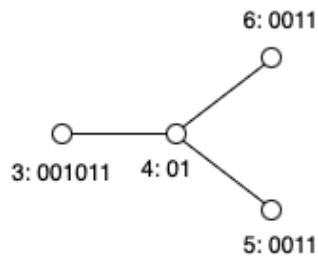
**Broj vrhova: 8**

- Unutrašnji vrh 3:  $Y = \{01, 01\}$
- Unutrašnji vrh 4:  $Y = \{\}$
- Unutrašnji vrh 5:  $Y = \{01\}$
- Unutrašnji vrh 6:  $Y = \{01\}$



**Broj vrhova: 4**

- Unutrašnji vrh 4:  $Y = \{001011, 0011, 0011\}$



**Broj vrhova: 1**

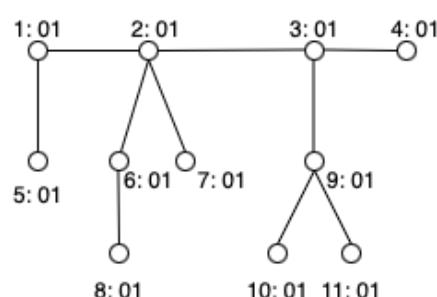
4: 0001011001100111

**Certifikat:** 0001011001100111

**Primjer 2. Izračun certifikata za stablo s dva centra**

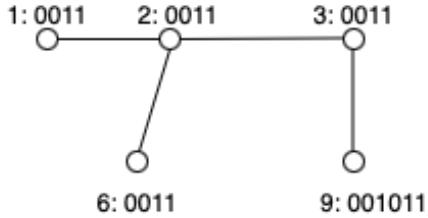
**Broj vrhova: 11**

- Unutrašnji vrh 1:  $Y = \{01\}$
- Unutrašnji vrh 2:  $Y = \{01\}$
- Unutrašnji vrh 3:  $Y = \{01\}$
- Unutrašnji vrh 6:  $Y = \{01\}$
- Unutrašnji vrh 9:  $Y = \{01, 01\}$

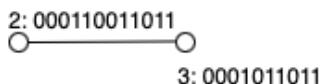


**Broj vrhova: 5**

- Unutrašnji vrh 2:  $Y = \{0011, 0011, 01\}$
- Unutrašnji vrh 3:  $Y = \{001011, 01\}$



**Broj vrhova: 2**



**Certifikat:** 0001011011000110011011

### 3.2.2. Jednostavni grafovi

Za razliku od stabla, postupak konstrukcije certifikata jednostavnog grafa je složeniji. Za potrebu detaljne analize algoritma koriste se razni novi pojmovi, uključujući i matricu susjedstva grafa. Matrica susjedstva je potrebna kako bi se mogao definirati certifikat grafa koji nije stablo.

Za jedan graf postoji više matrica susjedstva, ovisno o broju različitih permutacija vrhova grafa. Za bilo koju permutaciju  $\pi$ , elementi iznad glavne dijagonale tvore binarni niz koji se može interpretirati kao binarni broj. Za izradu binarnog niza uzimaju se elementi iznad glavne dijagonale i zapisuju se stupac po stupac. Najmanji binarni broj dobiven među svim mogućim permutacijama vrhova je certifikat grafa [3].

$$\text{Cert}(G) = \min\{\text{Num}_\pi(G) : \pi \in \text{Sym}(V)\}, \quad (3.1)$$

gdje  $\text{Sym}(V)$  označava skup svih permutacija vrhova iz  $V$ . Problematično je što ovaj pristup ima veliku računalnu složenost zbog faktorskog rasta broja permutacija, a to ga čini neefikasnim za velike grafove [6].

Na primjer za graf koji ima  $n = 13$  vrhova, broj mogućih permutacija vrhova iznosi  $n!$ , a to je čak 6227020800 mogućih permutacija vrhova. Stoga je potrebno pronaći neko ograničenje na permutacije kako bi se smanjio prostor pretraživanja. To se postiže ograničenjem na permutacije koje su stabilne.

ničavanjem skupa permutacija na one koje poštuju strukturne karakteristike grafa, kao što su susjedstva vrhova ili određene relacije između blokova u particijama. Takva ograničenja dobivaju se korištenjem uravnoteženih particija u odnosu na graf  $G = (V, E)$ .

**Definicija 9.** [9] Particija  $\pi = \{C_0, C_1, \dots, C_{m-1}\}$  skupa od  $n$  vrhova grafa  $G$  je *uravnotežena* ako za svaki par indeksa  $i, j \in \{0, 1, \dots, m-1\}$  postoji nenegativan cijeli broj  $b_{i,j}$  takav da svaki vrh  $v \in C_i$  ima točno  $b_{i,j}$  susjeda u  $C_j$ , neovisno o izboru vrha  $v$ . Matrica *uravnotežene particije* je  $B_\pi = (b_{i,j})$ .

Drugim riječima, vrhovi grafa su podijeljeni u blokove tako da svaki vrh u jednoj grupi ima isti broj susjeda u svakoj drugoj grupi.

**Definicija 10.** [3] *Diskretna particija*  $B$  je particija u kojoj svaki blok  $B[j]$  sadrži točno jedan element, odnosno vrijedi  $|B[j]| = 1$  za svaki  $j$ , gdje  $0 \leq j < k$ .

Ako je particija  $\pi$  iz Definicije 9 dodatno i diskretna, onda je matrica  $B_\pi$  ujedno i matrica susjedstva grafa  $G$ .

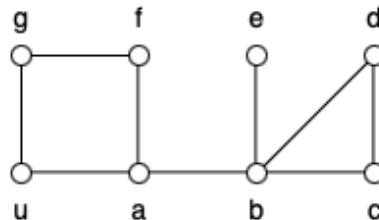
Još jedan način na koji se može smanjiti prostor pretraživanja je usitnjavanje particija. Usitnjavanjem se zadržavaju samo one permutacije koje poštuju veze među vrhovima grafa te se na taj način dobije manji skup mogućih permutacija i ubrzava se usporedba.

**Definicija 11.** [3] Uređena particija  $B$  je *usitnjavanje* uređene particije  $A$  ako vrijedi:

1. svaki blok  $B[i]$  od  $B$  je sadržan u nekom bloku  $A[j]$  od  $A$ ; i
2. ako su  $u \in A[i_1]$  i  $v \in A[j_1]$  te vrijedi  $i_1 \leq j_1$ , tada je  $u \in B[i_2]$  i  $v \in B[j_2]$  s  $i_2 \leq j_2$ .

### Primjer 3. Uravnotežena particija

Particija  $[\{e\}, \{c, d\}, \{f, u\}, \{g\}, \{a\}, \{b\}]$  je uravnotežena particija grafa  $G$  sa slike 3.13.



Slika 3.13. Graf  $G$  [1]

#### **Primjer 4. Usitnjavanje particije**

Neka su:

$$A = [\{0, 1, 2, 3, 4, 5\}, \{6, 7\}]$$

$$B = [\{0, 3\}, \{1, 2\}, \{5\}, \{4\}, \{6, 7\}], C = [\{0, 3\}, \{6, 7\}, \{1, 2\}, \{4\}, \{5\}]$$

Particija  $B$  je usitnjavanje particije  $A$  jer svaki blok iz  $B$  odgovara podskupu nekog bloka iz particije  $A$  i redoslijed blokova u  $B$  odgovara redoslijedu blokova u  $A$ . Particija  $C$  nije usitnjavanje particije  $A$  jer je redoslijed blokova u  $C$  narušen obzirom na redoslijed u  $A$ . U nastavku je detaljno objašnjen postupak usitnjavanje particija, koji je ključan za dobivanje relevantnih permutacija.

#### **Postupak usitnjavanje particije [3]**

Neka je  $A$  uređena particija i neka je  $D_T(v) = |N_G(v) \cap T|$ , gdje je  $T$  bilo koji blok particije  $A$  i  $N_G(v)$  susjedstvo vrha  $v$  u grafu  $G$ .

1.  $B = A$ .
2.  $S$  je lista koja sadrži blokove particije  $B$ .
3. Dok god vrijedi  $S \neq \emptyset$ :
  - (a) Blok  $T$  se uklanja iz  $S$ ;
  - (b) za svaki blok  $B[i]$  iz  $B$ :
    - i. za svaki  $h$ , definira se  $L[h] = \{v \in B[i] : D_T(v) = h\}$ ;
    - ii. ako postoji više od jednog nepraznog bloka u  $L$ :
      - A. Blok  $B[i]$  se zamjenjuje s nepraznim blokovima iz  $L$  poredanim prema indeksu  $h$ ,  $h = 0, 1, \dots, n - 1$ ;
      - B. Neprazni blokovi iz  $L$  se dodaju na kraj liste  $S$ .

Rezultirajuća particija  $B$  će zbog koraka 3(b)i biti uravnotežena.

Primjer postupka usitnjavanje particije je prikazan za graf sa slike 3.13. sa početnom particijom koja ima jedan blok koji sadrži sve vrhove grafa.

### Primjer 5. Usitnjavanje na uravnoteženu particiju

$$A = [\{a, b, c, d, e, f, g, u\}]$$

Početna particija je:

$$B = [\{a, b, c, d, e, f, g, u\}]$$

$$S = [\{a, b, c, d, e, f, g, u\}]$$

Uklanja se blok  $T = \{a, b, c, d, e, f, g, u\}$  iz  $S$  te se računa:  $L[0] = \emptyset$ ,  $L[1] = \{e\}$ ,  $L[2] = \{c, d, f, g, u\}$ ,  $L[3] = \{a\}$ ,  $L[4] = \{b\}$ . Budući da se u  $L$  nalaze četiri neprazna bloka, blok  $B[0] = \{a, b, c, d, e, f, g, u\}$  se zamjenjuje sa blokovima  $\{e\}, \{c, d, f, g, u\}, \{a\}, \{b\}$  i blokovi iz  $L$  se dodaju na kraj liste  $S$ . Dobivene particije su:

$$\begin{aligned} D_{\{a,b,c,d,e,f,g,u\}} : \quad B &= [\{e\}, \{c, d, f, g, u\}, \{a\}, \{b\}] \\ S &= [\{e\}, \{c, d, f, g, u\}, \{a\}, \{b\}] \end{aligned}$$

Postupak se ponavlja za svaki blok  $T$  iz  $S$  te se istim postupkom dobivaju sljedeći rezultati.

$$\begin{aligned} D_{\{e\}} : \quad B &= [\{e\}, \{c, d, f, g, u\}, \{a\}, \{b\}] \\ S &= [\{c, d, f, g, u\}, \{a\}, \{b\}] \end{aligned}$$

$$\begin{aligned} D_{\{c,d,f,g,u\}} : \quad B &= [\{e\}, \{c, d, f, u\}, \{g\}, \{a\}, \{b\}] \\ S &= [\{a\}, \{b\}, \{c, d, f, u\}, \{g\}] \end{aligned}$$

$$\begin{aligned} D_{\{a\}} : \quad B &= [\{e\}, \{c, d\}, \{f, u\}, \{g\}, \{a\}, \{b\}] \\ S &= [\{b\}, \{c, d, f, u\}, \{g\}, \{c, d\}, \{f, u\}] \end{aligned}$$

$$\begin{aligned} D_{\{b\}} : \quad B &= [\{e\}, \{c, d\}, \{f, u\}, \{g\}, \{a\}, \{b\}] \\ S &= [\{c, d, f, u\}, \{g\}, \{c, d\}, \{f, u\}] \end{aligned}$$

$$\begin{aligned} D_{\{c,d,f,u\}} : \quad B &= [\{e\}, \{c, d\}, \{f, u\}, \{g\}, \{a\}, \{b\}] \\ S &= [\{g\}, \{c, d\}, \{f, u\}] \end{aligned}$$

$$D_{\{g\}} : \quad B = [\{e\}, \{c, d\}, \{f, u\}, \{g\}, \{a\}, \{b\}]$$

$$S = [\{c, d\}, \{f, u\}]$$

$$D_{\{c, d\}} : \quad B = [\{e\}, \{c, d\}, \{f, u\}, \{g\}, \{a\}, \{b\}]$$

$$S = [\{f, u\}]$$

$$D_{\{f, u\}} : \quad B = [\{e\}, \{c, d\}, \{f, u\}, \{g\}, \{a\}, \{b\}]$$

$$S = []$$

Postupkom usitnjavanja je dobivena uravnotežena particija iz primjera 3

$$B = [\{e\}, \{c, d\}, \{f, u\}, \{g\}, \{a\}, \{b\}]$$

### **Postupak dobivanja diskretne particije [3]**

Unatoč tome što je dobivena particija uravnotežena, ona ne određuje u potpunosti redoslijed vrhova jer nije diskretna. Cilj je nastaviti proces usitnjavanja sve dok se ne postigne diskretna particija.

1. Potrebno je pronaći prvi blok u uravnoteženoj particiji  $B$  koji sadrži više od jednog vrha. Taj blok se označava kao  $T$ .
2. Blok  $T$  veličine  $m$  veće od jedan se dijeli na dva bloka: prvi blok ima veličinu 1, dok drugi blok ima veličinu  $m - 1$ . Sve takve podjele se moraju razmotriti.
3. Nakon podjele, primjenjuje se algoritam za usitnjavanje kako bi se postigla uravnoteženost particije. Proces podjele i usitnjavanja se ponavlja sve dok svaki blok particije ne bude veličine jedan, čime se postiže diskretna particija  $P'$  koja određuje jedan od redoslijeda za razmatranje.

### **Postupak određivanja certifikata grafa [3]**

1. Algoritam započinje inicijalnom particijom  $A$  koja sadrži jedan blok u kojemu se nalaze svi vrhovi grafa.
2. Nad particijom  $A$  se poziva algoritam usitnjavanja particije te on rezultira uravno-

teženom particijom  $B$ .

3. Ako je  $B$  diskretna, tada particija  $B$  već definira jedinstveni redoslijed vrhova  $\pi$  i algoritam može izračunati certifikat  $\text{Num}_\pi(G)$  za graf  $G$ .
4. Ako  $B$  nije diskretna, poziva se algoritam dobivanja diskretne particije. Svaka dobivena diskretna particija  $P'$  generira potencijalni redoslijed  $\pi$  za vrhove grafa.
5. Tijekom pretraživanja, održava se najmanja pronađena vrijednost  $\text{Num}_\mu(G)$ , gdje je  $\mu$  redoslijed vrhova koji trenutno daje najmanji certifikat za graf. Ako se tijekom pretraživanja otkrije manji broj od trenutnog  $\text{Num}_\mu(G)$ ,  $\mu$  se ažurira na taj redoslijed.
6. Za pretraživanje se koristi **backtracking algoritam** koji postupno gradi djelomične redoslijede i provjerava uvjete kako bi eliminirao grane pretraživanja koje ne mogu rezultirati manjim certifikatom. Ako se tijekom pretraživanja ustanovi da trenutni djelomični redoslijed  $\pi$  daje broj veći od trenutnog  $\text{Num}_\mu(G)$ , daljnje grane pretraživanja se odbacuju.
7. Proces se ponavlja sve dok se ne ispitaju sve relevantne permutacije koje zadovoljavaju strukturu uravnotežene i diskretne particije. Kada se ispitaju sve mogućnosti, najmanji pronađeni broj  $\text{Num}_\mu(G)$  predstavlja certifikat grafa  $G$ .

U nastavku je prikazan izračun certifikata za graf sa slike 3.13.

#### **Primjer 6. Izračun certifikata grafa**

U primjeru 5 je već prikazan postupak usitnjavanja te je resultantna uravnotežena particija inicijalnog usitnjavanja

$$B = [\{e\}, \{c, d\}, \{f, u\}, \{g\}, \{a\}, \{b\}] .$$

Prvi blok koji sadrži više od jednog elementa je blok  $\{c, d\}$ , dakle dalje se provodi usitnjavanja particija za sve moguće podjele tog bloka:

$$[\{e\}, \{c\}, \{d\}, \{f, u\}, \{g\}, \{a\}, \{b\}] \quad \text{i} \quad [\{e\}, \{d\}, \{c\}, \{f, u\}, \{g\}, \{a\}, \{b\}] .$$

Kako se dijeljenjem bloka nije postigla diskretna particija, usitnjavanje se ponavlja.

$$B = [\{e\}, \{c\}, \{d\}, \{f, u\}, \{g\}, \{a\}, \{b\}]$$

$$S = [\{e\}, \{c\}, \{d\}, \{f, u\}, \{g\}, \{a\}, \{b\}]$$

$$D_{\{e\}} : \quad B = [\{e\}, \{c\}, \{d\}, \{f, u\}, \{g\}, \{a\}, \{b\}]$$

$$S = [\{c\}, \{d\}, \{f, u\}, \{g\}, \{a\}, \{b\}]$$

$$D_{\{c\}} : \quad B = [\{e\}, \{c\}, \{d\}, \{f, u\}, \{g\}, \{a\}, \{b\}]$$

$$S = [\{d\}, \{f, u\}, \{g\}, \{a\}, \{b\}]$$

$$D_{\{d\}} : \quad B = [\{e\}, \{c\}, \{d\}, \{f, u\}, \{g\}, \{a\}, \{b\}]$$

$$S = [\{f, u\}, \{g\}, \{a\}, \{b\}]$$

$$D_{\{f, u\}} : \quad B = [\{e\}, \{c\}, \{d\}, \{f, u\}, \{g\}, \{a\}, \{b\}]$$

$$S = [\{g\}, \{a\}, \{b\}]$$

$$D_{\{g\}} : \quad B = [\{e\}, \{c\}, \{d\}, \{f, u\}, \{g\}, \{a\}, \{b\}]$$

$$S = [\{a\}, \{b\}]$$

$$D_{\{a\}} : \quad B = [\{e\}, \{c\}, \{d\}, \{f, u\}, \{g\}, \{a\}, \{b\}]$$

$$S = [\{b\}]$$

$$D_{\{b\}} : \quad B = [\{e\}, \{c\}, \{d\}, \{f, u\}, \{g\}, \{a\}, \{b\}]$$

$$S = []$$

Nakon što su svi elementi u listi  $S$  obrađeni, particija  $B$  je uravnotežena, ali još uvijek nije diskretna. Radi se podjela sljedećeg bloka koji sadrži više od jednog elementa.

$$B = [\{e\}, \{c\}, \{d\}, \{f\}, \{u\}, \{g\}, \{a\}, \{b\}]$$

Budući da je  $B$  diskretna dobiven je poredak koji predstavlja kandidata za izračun certifikata grafa.

$$\pi_1 = [e, c, d, f, u, g, a, b].$$

Za dobiveni poredak zapisuje se odgovarajuća matrica susjedstva.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Zapisivanjem elemenata iznad glavne dijagonale dobiva se prvi kandidat za certifikat grafa.

$$\text{Num}_{\pi_1}(G) = (001000000000110001101110001)_2$$

Kao što je navedeno u opisu algoritma, sve podjele blokova se moraju razmotriti. Sukladno tome, drugi potencijalni poredak je

$$\pi_2 = [e, c, d, u, f, g, a, b].$$

i pripadni kandidat za certifikat

$$\text{Num}_{\pi_2}(G) = (001000000000110001101110001)_2.$$

Time je završen postupak za prvu podjelu bloka  $\{c, d\}$  i sada se usitnjava particija

$$B = [\{e\}, \{d\}, \{c\}, \{f, u\}, \{g\}, \{a\}, \{b\}].$$

Analognim postupkom usitnjavanja dobije se uravnotežena, ali ne i diskretna particija

$$B = [\{e\}, \{d\}, \{c\}, \{f, u\}, \{g\}, \{a\}, \{b\}].$$

Podjelom bloka  $\{f, u\}$  dobivaju se još dva potencijalna poretki

$$\pi_3 = [e, d, c, f, u, g, a, b]$$

i

$$\pi_4 = [e, d, c, u, f, g, a, b].$$

Zapisivanjem elemata iznad glavne dijagonale dobivena su još dva kandidata za certifikat grafa.

$$\text{Num}_{\pi_3}(G) = (001000000000110001101110001)_2$$

i

$$\text{Num}_{\pi_4}(G) = (001000000000110001101110001)_2.$$

Budući da je

$$\text{Num}_{\pi_1}(G) = \text{Num}_{\pi_2}(G) = \text{Num}_{\pi_3}(G) = \text{Num}_{\pi_4}(G) = (001000000000110001101110001)_2$$

**certifikat graf je**

$$\mathbf{C} = (001000000000110001101110001)_2 = 33579889.$$

## 4. Certifikati grafova s pet vrhova

Klasifikacija grafova služi za grupiranje grafova na temelju njihovih svojstva. Konkretno za grafove s pet vrhova, cilj ih je podijeliti u klase tako da svaka klasa sadrži samo izomorfne grafove. Svaka klasa će imati svog predstavnika i biti će popisani svi međusobno neizomorfni grafovi s pet vrhova te njihovi certifikati. Budući da identifikacija svih takvih neizomorfnih grafova nije jednostavna i nije tema ovog rada, oni će biti preuzeti iz literature [10].

Grafovi će biti razvrstani u tri kategorije. Nepovezani, stabla i povezani grafovi koji nisu stabla.

### 4.1. Nepovezani grafovi s pet vrhova

Slijede svi neizomorfni nepovezani grafovi s pet vrhova i njihovi certifikati  $C$ . Ukupno ih ima 13.



**Slika 4.1.**  $C = 0$



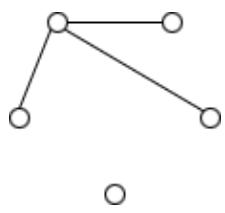
**Slika 4.2.**  $C = 1$



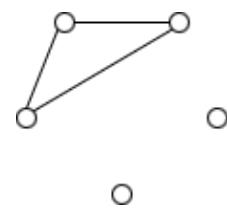
**Slika 4.3.**  $C = 20$



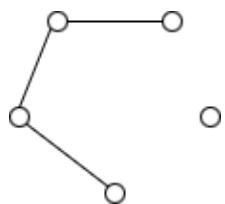
**Slika 4.4.**  $C = 3$



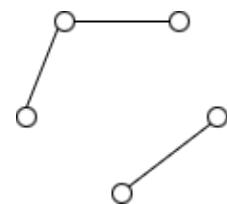
**Slika 4.5.**  $C = 7$



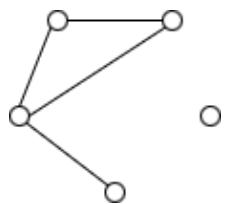
**Slika 4.6.**  $C = 19$



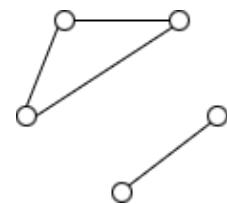
**Slika 4.7.**  $C = 21$



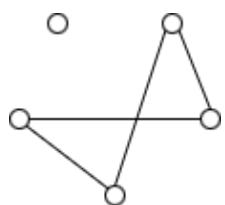
**Slika 4.8.**  $C = 28$



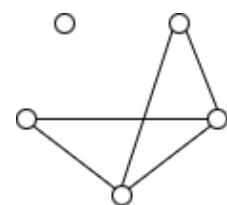
**Slika 4.9.**  $C = 23$



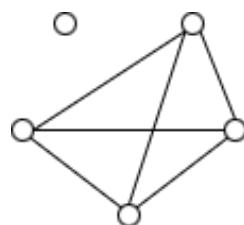
**Slika 4.10.**  $C = 531$



**Slika 4.11.**  $C = 54$



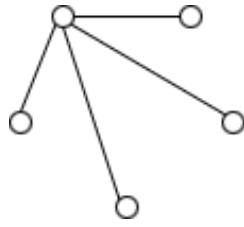
**Slika 4.12.**  $C = 55$



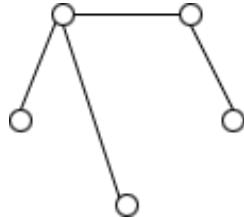
**Slika 4.13.**  $C = 183$

## 4.2. Stabla s pet vrhova

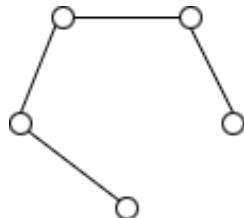
Budući da je stablo povezani graf koji nema ciklusa, jasno je da neizomorfnih stabala sa 5 vrhova nema puno. Slijede u nastavku.



**Slika 4.14.**  $C = 0010101011_2 = 171$



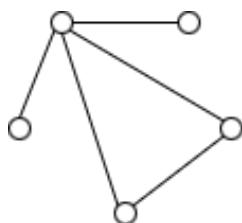
**Slika 4.15.**  $C = 0010110011_2 = 179$



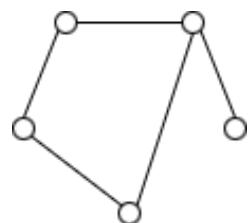
**Slika 4.16.**  $C = 0001100111_2 = 103$

### 4.3. Povezani grafovi s pet vrhova koji nisu stabla

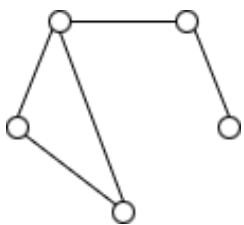
Ovakvih neizomorfnih grafova s pet vrhova ima puno, a se trebaju razmotriti svi povezani jednostavni grafovi s pet i više bridova.



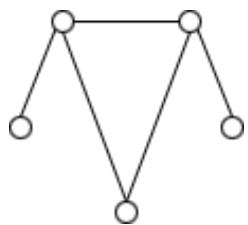
**Slika 4.17.**  $C = 31$



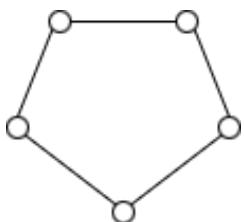
**Slika 4.18.**  $C = 62$



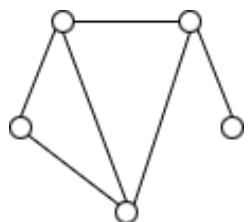
**Slika 4.19.**  $C = 199$



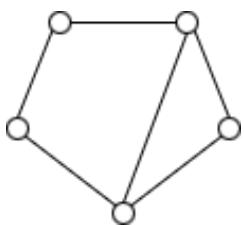
**Slika 4.20.** C = 59



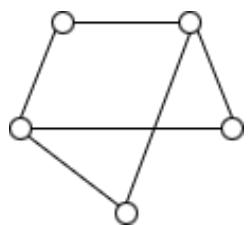
**Slika 4.21.**  $C = 220$



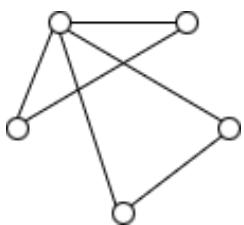
**Slika 4.22.**  $C = 187$



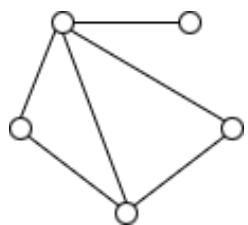
**Slika 4.23.** C = 221



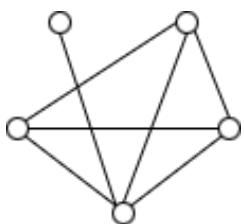
**Slika 4.24.**  $C = 126$



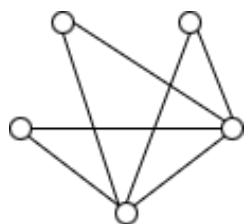
**Slika 4.25.** C = 207



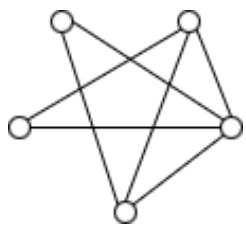
**Slika 4.26.**  $C = 63$



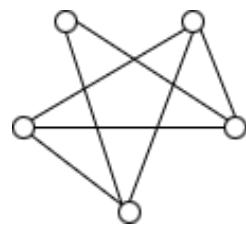
**Slika 4.27.** C = 191



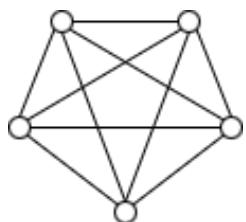
**Slika 4.28.**  $C = 127$



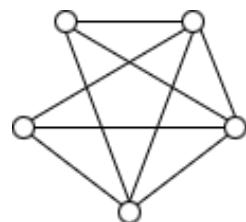
Slika 4.29.  $C = 223$



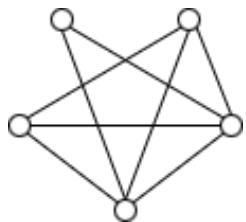
Slika 4.30.  $C = 254$



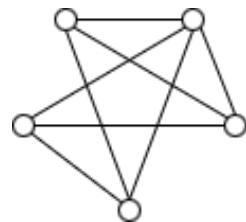
Slika 4.31.  $C = 1023$



Slika 4.32.  $C = 511$



Slika 4.33.  $C = 255$



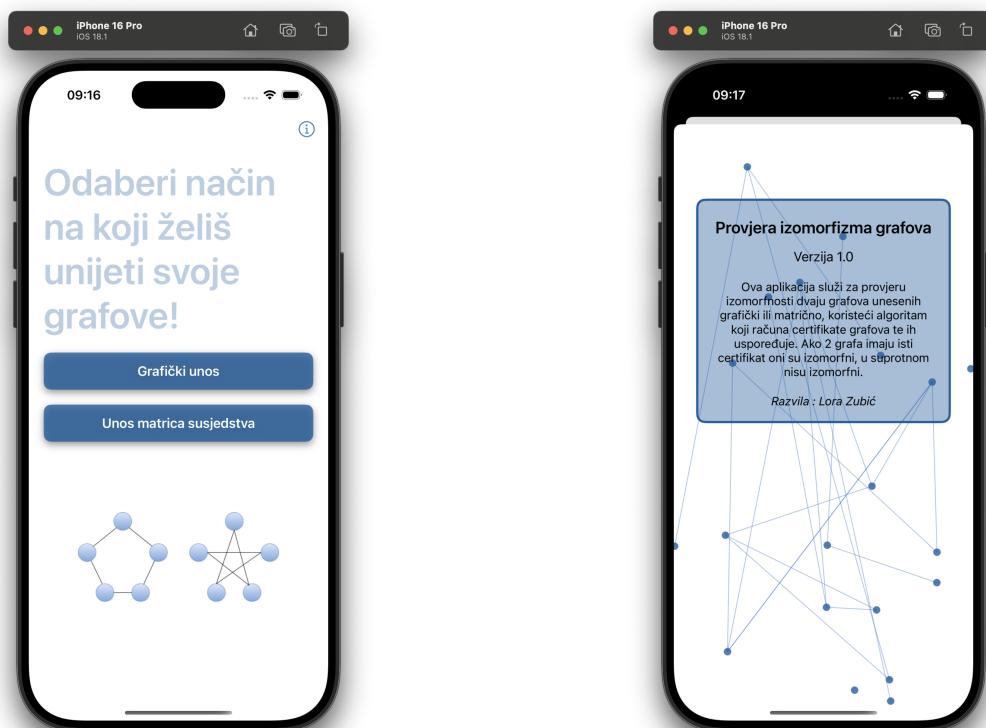
Slika 4.34.  $C = 495$

Time su popisani svi neizomorfni grafovi s pet vrhova i njihovi certifikati te se jasno vidi da niti jedna dva grafa nemaju isti certifikat.

## 5. Opis izrađene aplikacije

Cilj izrade aplikacije je omogućavanje korisnicima provjeru izomorfnosti grafova na intuitivan i jednostavan način koristeći grafičko sučelje i matrice susjedstva. Također, aplikacija korisnicima daje uvid u jedinstveni potpis grafa. U nastavku su objašnjene mogućnosti aplikacije te kako se ista koristi.

Početni ekran nudi mogućnost izbora između dva načina unosa grafova, matrično i grafički, kao što je prikazano na slici 5.1. Na početnom ekranu, osim odabira načina unosa grafova, u gornjem desnom kutu se nalazi gumb koji prikazuje detaljnije informacije o aplikaciji kao što su verzija, opis i razvojnik. Dodatna zanimljivost ekrana sa informacijama je pozadina koja prikazuje nasumično generiran graf kao sa slike 5.2.



Slika 5.1. Odabir načina unosa grafova

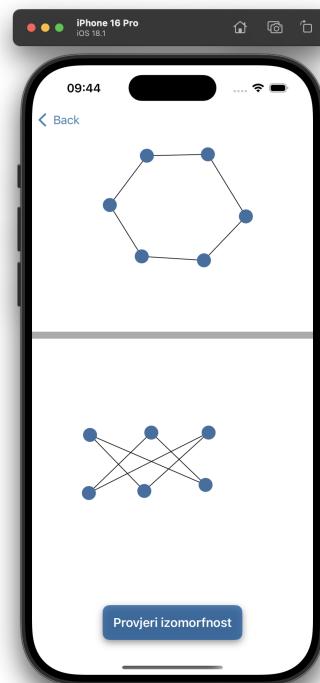
Slika 5.2. Informacijski ekran

## 5.1. Grafički unos grafova

Ako korisnik izabere grafički unos grafova, prikazat će mu se ekran sa dva platna za grafičko crtanje, odvojena horizontalnom linijom po sredini ekrana. Na svako platno korisnik treba nacrtati po jedan graf. Klikom na platno se unosi vrh grafa, a potezanjem linije između dva vrha unosi se brid grafa. Primjer unosa dvaju grafova prikazan je na slici 5.4.



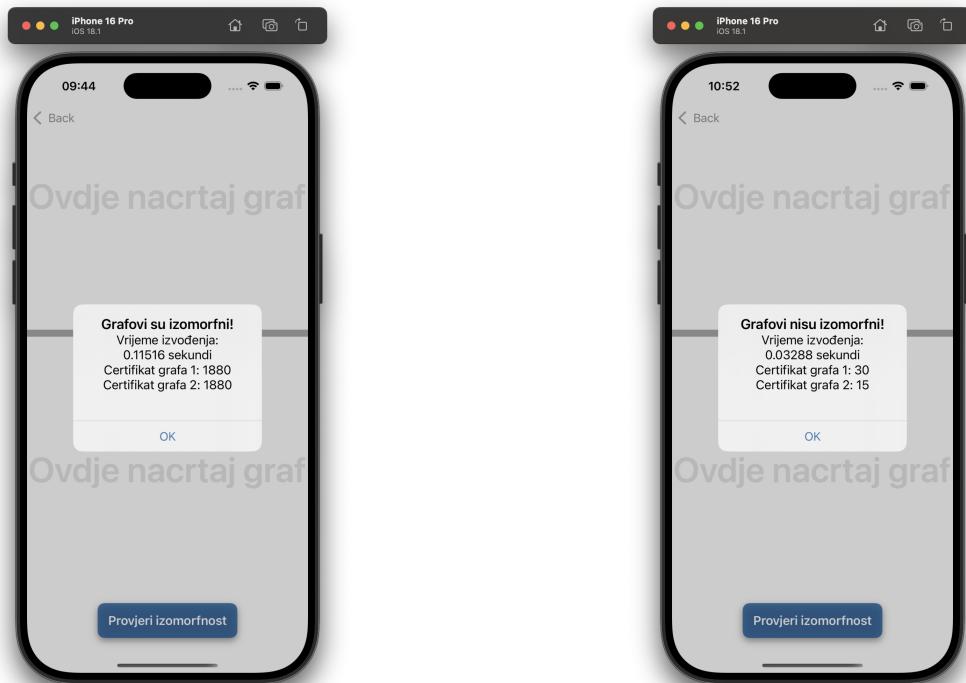
Slika 5.3. Grafički unos grafova



Slika 5.4. Primjer grafičkog unosa grafova

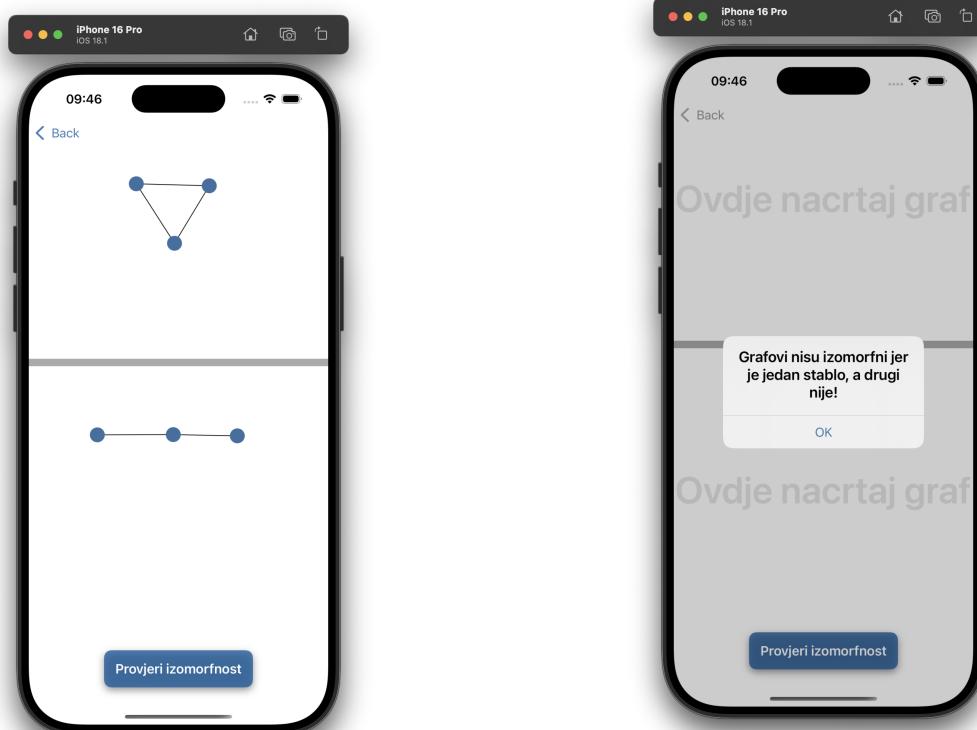
Nakon što su grafovi uneseni, klikom na gumb "Provjeri izomorfnost", pokreće se algoritam za provjeru izomorfosti. Kada algoritam završi, prikazuje se iskočni prozor u kojem je prikazano vrijeme izvršavanja algoritma, certifikati obaju grafova i informacija o izomorfnosti unesenih grafova. Klikom na "Ok" iskočni prozor se zatvara i platna su opet spremna za unos grafova. Dotični iskočni prozor prikazan je na slikama 5.5. i 5.6.

Program odabire prikladni algoritam za provjeru izomorfnosti tako da za oba grafa odredi radi li se o stablu ili ne. Ako su oba grafa stabla, odabire se algoritam za konstrukciju certifikata stabala, a ako oba grafa nisu stabla onda se certifikati grafova konstruiraju pomoću algoritma za općenite grafove. U slučaju da je jedan graf stablo a drugi nije, program odmah zaključuje da grafovi nisu izomorfni te to javlja korisniku.



**Slika 5.5.** Iskočni prozor za izomorfne grafove

**Slika 5.6.** Iskočni prozor za neizomorfne grafove

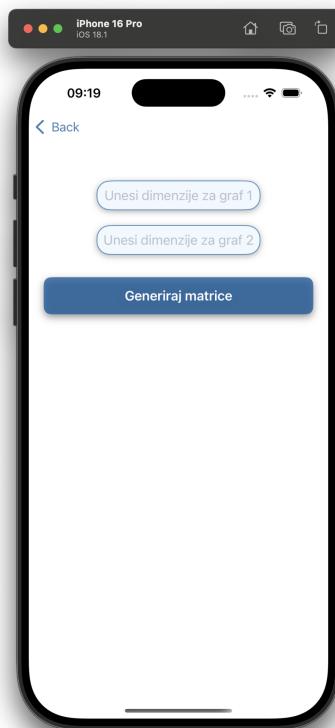


**Slika 5.7.** Stablo i graf s ciklusom

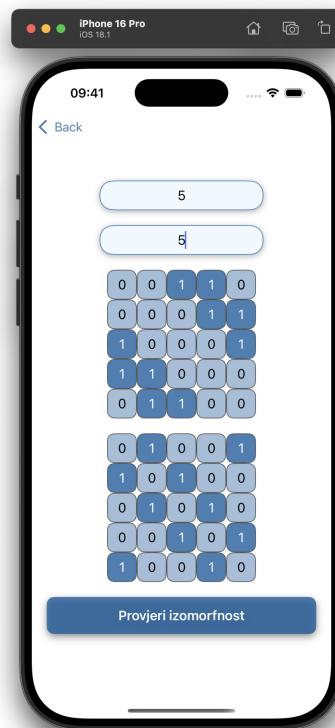
**Slika 5.8.** Iskočni prozor za stablo i graf s ciklusom

## 5.2. Matrični unos grafova

Na slici 5.9. je prikazan slučaj kad korisnik odabere matrični unos grafova. Ekran se sastoji od dva polja u koja se trebaju unijeti dimenzije matrica i gumba "Generiraj matrice" koji generira matrice unesenih dimenzija. Klikom na pojedini element matrice, njegova vrijednost se mijenja iz nule u jedinicu i obrnuto. To je tako kako bi unos matrica bio što brži i jednostavniji, obzirom na to da matrica susjedstva grafa sadrži isključivo nule i jedinice. Matrice sa slike 5.10. su matrice susjedstva grafova sa slika 2.3. i 2.4.

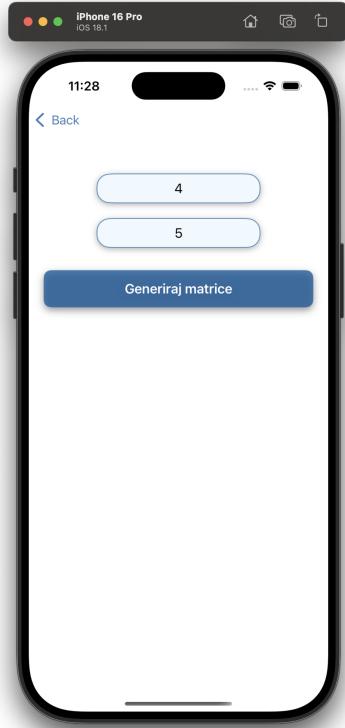


Slika 5.9. Unos dimenzija matrica



Slika 5.10. Unos matrica

Ako korisnik uneše različite dimenzije matrica i klikne na gumb "Generiraj matrice", program javlja da matrice susjedstva moraju imati jednakе dimenzije jer u suprotnom grafovi sigurno nisu izomorfni. To je zato što dimenzija matrice susjedstva također predstavlja i broj vrhova grafa. Opisani scenarij je prikazan na slikama 5.11. i 5.12.



**Slika 5.11.** Unos različitih dimenzija matrica



**Slika 5.12.** Iskočni prozor za Unos različitih dimenzija matrica

Kao i kod grafičkog unosa grafova, nakon unosa matrica, klikom na gumb "Provjeri izomorfnost" pojavljuje se iskočni prozor sa istim informacijama. Klikom na "Ok" u iskočnom prozoru, maska se čisti te je ponovo omogućen unos dimenzija matrica.



**Slika 5.13.** Iskočni prozor za izomorfne grafove



**Slika 5.14.** Iskočni prozor za neizomorfne grafove



**Slika 5.15.** Iskočni prozor s upozorenjem



**Slika 5.16.** Iskočni prozor s upozorenjem

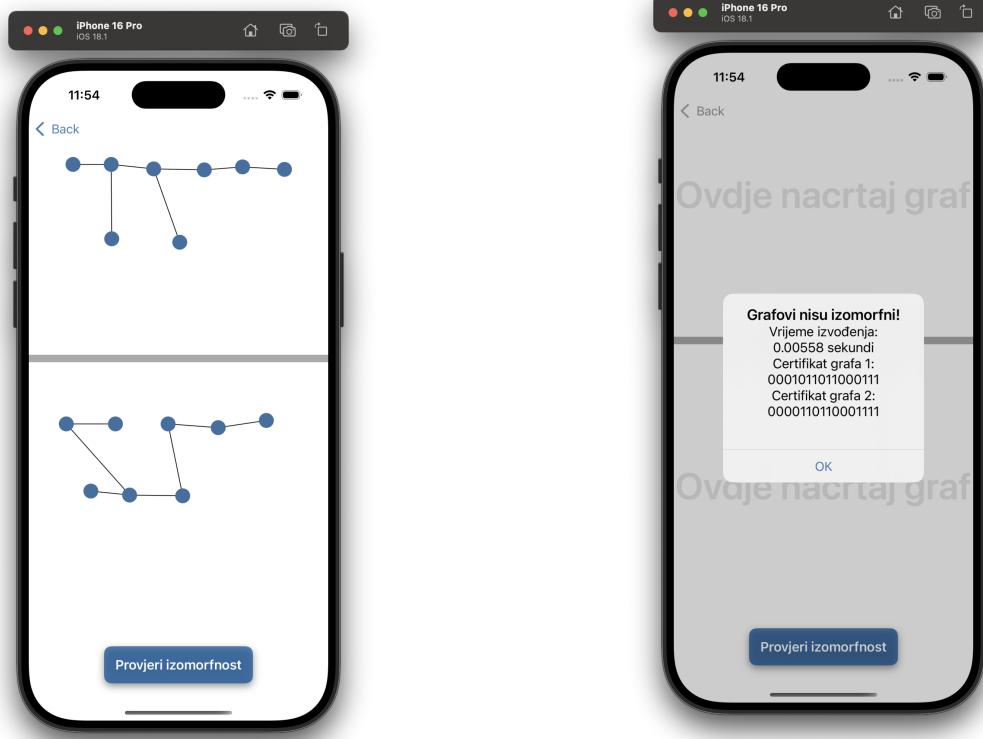
Budući da je problem određivanja izomorfnosti grafova kompleksan i sve teži sa porastom broja vrhova, aplikacija omogućava provjeru izomorfnosti grafova za grafove do uključno 8 vrhova te ukoliko je broj vrhova veći od 8, program javlja upozorenje prikazano na slikama 5.16. i 5.15.

## 6. Analiza i evaluacija

Kao što je već spomenuto kod pregleda aplikacije, za svaki par unesenih grafova iskazano je vrijeme izvršavanja algoritma za provjeru izomorfizma. To će biti korisno pri analazi brzine algoritma za različite vrste grafova.

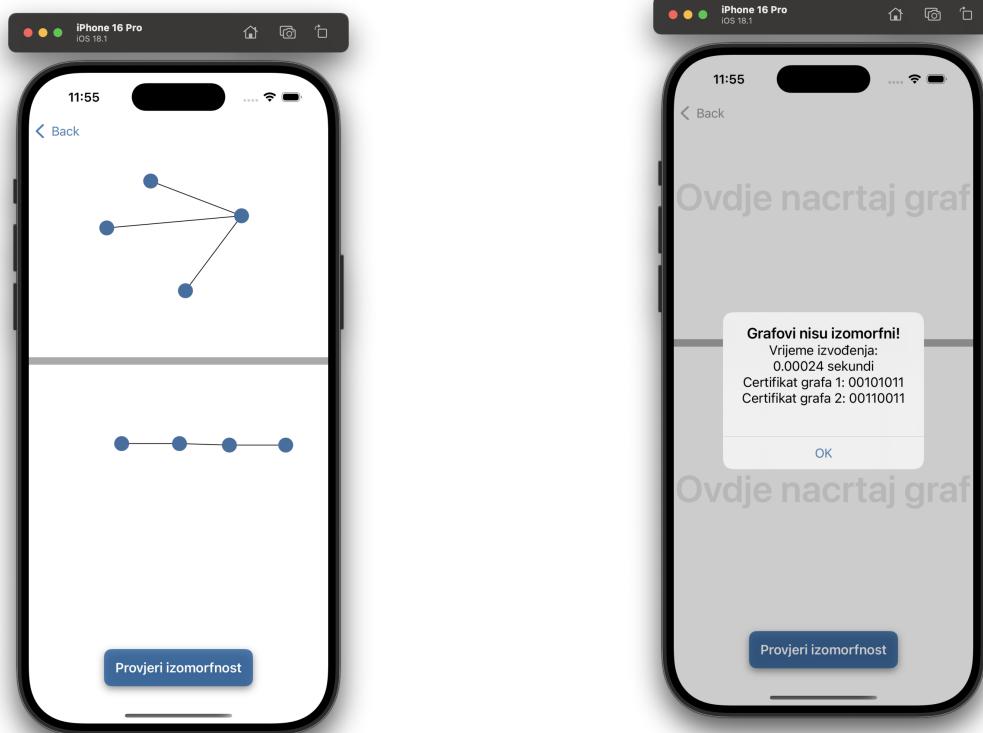
### 6.1. Algoritam za izračun cerifikata stabla

Iz koraka algoritma može se zaključiti da on ima linearnu vremensku složenost  $O(n)$  jer se svaki vrh i brid obrađuju konstantan broj puta. Iz toga se može zaključiti da je ovaj algoritam učinkovit za provjeru izomorfizma za stabala s većim brojem vrhova. Na slikama 6.1. i 6.2. se vidi vrijeme izvršavanja algoritma za dva stabla s osam vrhova te ono nije značajno dugo. Iako vrijeme izvršavanja algoritma za grafove s osam vrhova nije značajno dugo, ono je dugotrajnije u usporedbi s grafovima na slici 6.3. koji imaju četiri vrha i dosta kraće vrijeme izvršavanja algoritma. Dakle vrijeme izvršavanja algoritma raste s porastom broja vrhova stabla.



Slika 6.1. Stabla s osam vrhova

Slika 6.2. Vrijeme izvršavanja za stabla s osam vrhova



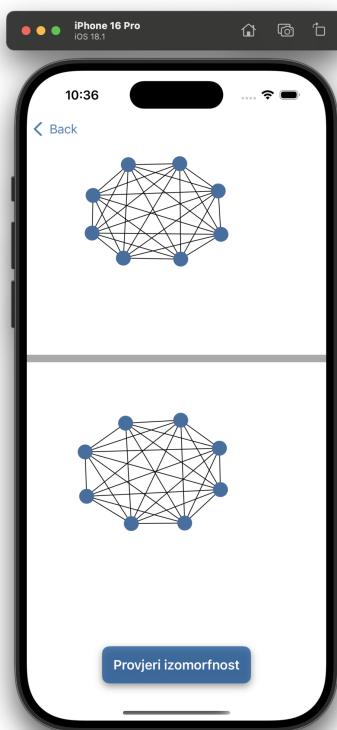
Slika 6.3. Stabla s četiri vrhova

Slika 6.4. Vrijeme izvršavanja za stabla s četiri vrhova

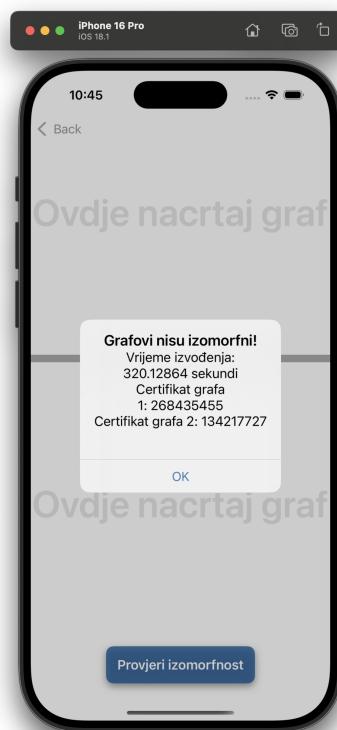
## 6.2. Algoritam za izračun certifikata jednostavnog grafa

Za razliku od algoritma za stabla, ovaj algoritam je vremenski puno složeniji. To je zato što treba pretražiti sve permutacije vrhova grafa kako bi se pronašao minimalni certifikat. Prostor pretraživanja se smanjuje usitnjavanjem particija, ali složenost je i dalje visoka. Lako je zaključiti da je vremenska složenost algoritma visoka, no kao što je već napomenuto kod poglavљa o izomorfnosti grafova, problem ispitivanja izomorfnosti jednostavnih grafova spada u klasu problema čija je vremenska složenost još uvijek neodređena. Nije dokazano da se može riješiti u polinomijalnom vremenu, niti je potvrđeno da je problem NP-potpun.

Unatoč tome, može se analizirati kako se vremenska složenost mijenja ovisno o broju vrhova i strukturi grafa. Na slici 6.6. je prikazano vrijeme izvršavanja algoritma za potpuni graf s osam vrhova. Vrijeme izvršavanja je jako dugo, a to je zato što složenost algoritma raste s porastom broja vrhova grafa i potpuni graf ima najviše bridova, a usitnjavanje particija je znatno sporije jer svi vrhovi imaju isti stupanj.



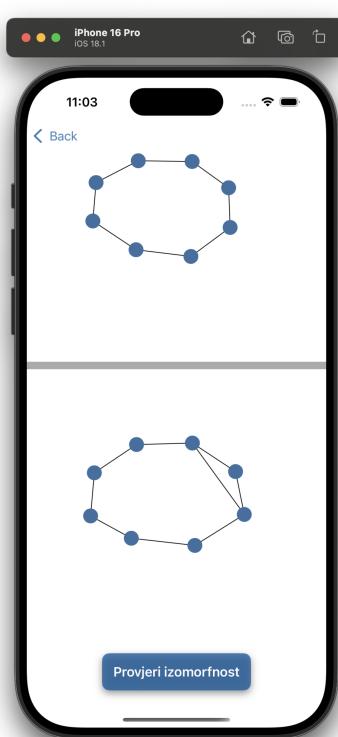
Slika 6.5.  $K_8$  i  $K_8$  bez jednog brida



Slika 6.6. Vrijeme izvršavanja za  $K_8$

Iako graf sa slike 6.7. ima osam vrhova, vrijeme izvršavanja je znatno kraće nego

kod potpunog grafa jer ima mali broj bridova, a to značajno smanjuje složenost matrice susjedstva.

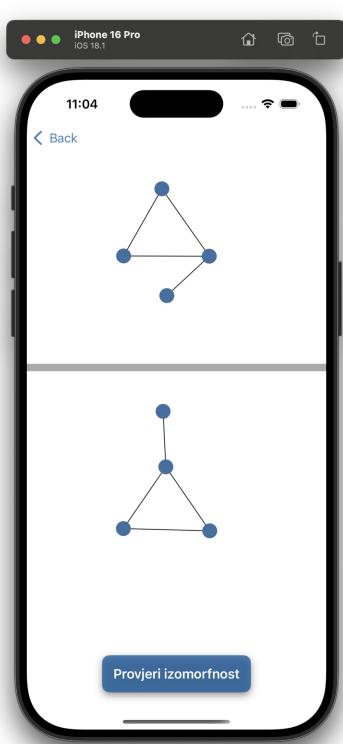


**Slika 6.7.** Povezani ciklus s osam vrhova



**Slika 6.8.** Vrijeme izvršavanja za povezani ciklus s osam vrhova

Treba napomenuti da broj vrhova grafa za koji se računa certifikat ima bitnu ulogu u vremenu izvršavanja algoritma. Tako je na primjer za grafove s četiri vrha sa slike 6.9. vrijeme izvršavanja kraće nego kod ciklusa s osam vrhova sa slike 6.7.



**Slika 6.9.** Graf s četiri vrha



**Slika 6.10.** Vrijeme izvršavanja za graf s četiri vrha

Gusto povezani grafovi poput potpunih otežavaju usitnjavanje jer vrhovi dijeli slične invarijante, dok grafovi s rijetkom strukturom poput ciklusa obično imaju manje mogućih permutacija jer se brzo dolazi do diskretnih particija. Porastom broja vrhova grafa, broj mogućih permutacija eksponencijalno raste, a ako se broj relevantnih permutacija ne može znatno smanjiti usitnjavanjem particija, to znatno povećava vrijeme izvršavanja. U takvim slučajevima algoritam može postati vrlo spor, pogotovo za grafove s puno simetrija ili visokom gustoćom.

## Zaključak

U ovom radu je obrađen algoritam za provjeru izomofnosti grafova koji koristi certifikate. Općenito, određivanje izomorfnosti grafova je težak problem za koji ovaj rad nudi rješenje za grafove s do osam vrhova. Koraci algoritma su detaljno objašnjeni i potkrijepljeni primjerom, a kasnije i implementirani u mobilnoj aplikaciji. Detaljna analiza algoritma pokazuje da algoritam, iako vremenski neodređen, uspješno rješava problem određivanja izomorfnosti grafova s do osam vrhova. Na primjerima je pokazano da za veći broj vrhova vrijeme izvršavanja algoritma raste. Također, ako je struktura grafa takva da svi vrhovi imaju isti stupanj, izračun certifikata može trajati duže jer u tom slučaju graf može imati mnogo različitih permutacija vrhova koje su izomorfne. Moguće je da će algoritam morati razmotriti sve permutacije vrhova kako bi pronašao certifikat za graf. Unatoč navedenim ograničenjima algoritma, aplikacija koja ga implementira učinkovito određuje izomorfnost manjih grafova i korisniku prikazuje njihove certifikate.

## Literatura

- [1] D. B. West, *Introduction to Graph Theory*. Upper Saddle River, NJ: Prentice Hall, 2001.
- [2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, i C. Stein, *Introduction to algorithms, Fourth edition*. Cambridge, Massachusetts London, England: The MIT Press, 2022.
- [3] D. Stinson. i D. Kreher, *Combinatorial algorithms: Generation, Enumeration and Search*. Boca Raton, Florida: CRC Press, 1999.
- [4] *Graph invariants*, 2021., accessed: 13.12.2024. [Mrežno]. Adresa: [https://abelprize.no/sites/default/files/2021-09/Graph\\_invariants.pdf](https://abelprize.no/sites/default/files/2021-09/Graph_invariants.pdf)
- [5] M. Grohe i P. Schweitzer, *The graph isomorphism problem*, *Communications of the ACM*, 2018., accessed: 02.12.2024. [Mrežno]. Adresa: [https://www.lics.rwth-aachen.de/global/show\\_document.asp?id=aaaaaaaaabptevzm](https://www.lics.rwth-aachen.de/global/show_document.asp?id=aaaaaaaaabptevzm)
- [6] L. Babai i E. M. Luks, *Canonical labeling of graphs*, u *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*. New York, NY, USA: ACM, 1983., str. 171–183.
- [7] D. Grinberg, *An Introduction to Graph Theory*. arXiv, 2022.
- [8] I. Nakić, *Diskretna matematika*, 2012., predavanja, ak. god. 2011./2012. PMF-Matematički odsjek.
- [9] A. Staples-Moore, *Equitable partitions in graph theory, REU Program Paper*, 2024.
- [10] A. Nakić i I. Vesel, *Problem klasifikacije grafova*, *Acta Mathematica Spalatensis*, sv. 3, 2020.

- [11] R. Diestel, *Graph Theory*. Berlin, Heidelberg: Springer-Verlag, 2005.
- [12] P. Ahammad, *Graph theory: Isomorphic trees*, <https://towardsdatascience.com/graph-theory-isomorphic-trees-7d48aa577e46>, 2020., accessed: 08.12.2024.
- [13] T. E. Portegys, *Graph isomorphism testing boosted by path coloring, General Graph Identification With Hashing*, 2008.

# Sažetak

## Računalno ispitivanje izomorfnosti grafova

Lora Zubić

Tema ovog rada je algoritam koji rješava problem izomorfnosti grafova. Razvijena je mobilna aplikacija koja implementira algoritam za provjeru izomorfnosti grafova temeljen na izračunu certifikata grafa. Certifikat grafa je jedinstveni potpis koji omogućuje provjeru izomorfnosti između dva grafa. Postupak za izračun certifikata se dijeli na dva algoritma, jedan prilagođen stablima koji ima linearnu vremensku složenost i drugi za općenite grafove, čija je vremenska složenost neodređena. Aplikacija omogućuje unos dvaju grafova s najviše osam vrhova, pri čemu se grafovi mogu unijeti grafički ili pomoću matrica susjedstva. Program bira odgovarajući algoritam za izračun certifikata, a zatim usporedbom certifikata dvaju grafova javlja korisniku jesu li oni izomorfni. U radu su popisani svi neizomorfni grafovi s pet vrhova i njihovi certifikati. Analizirana je i učinkovitost algoritama te rezultati pokazuju kako na vrijeme izvršavanja utječe veličina i struktura grafova.

**Ključne riječi:** izomorfizam grafova; certifikat grafa; klasifikacija;

# **Abstract**

## **Computational Examination of Graph Isomorphism**

Lora Zubić

The topic of this paper is an algorithm that solves the problem of determining graph isomorphism. A mobile application was developed that implements an algorithm for graph isomorphism verification based on graph certificate calculation. Graph certificate is a unique signature that enables the verification of isomorphism between two graphs. The certificate calculation process is divided into two algorithms, one for trees, which has linear time complexity, and another for general graphs, whose time complexity is undefined. The application allows the input of two graphs with up to eight vertices, which can be entered either graphically or using adjacency matrices. The program selects the appropriate algorithm for certificate calculation, compares the certificates of the two graphs and informs the user whether they are isomorphic. The paper also lists all non-isomorphic graphs with five vertices and their certificates. Additionally, the efficiency of the algorithms was analyzed and the results show that execution time is significantly influenced by the size and structure of the graphs.

**Keywords:** graph isomorphism; graph certificate; graph classification