

# Izabrani algoritmi teorije grafova

---

Krstić, Dana

Master's thesis / Diplomski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Split, Faculty of Science / Sveučilište u Splitu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: <https://um.nsk.hr/um:nbn:hr:166:851179>

Rights / Prava: [In copyright](#)/[Zaštićeno autorskim pravom.](#)

Download date / Datum preuzimanja: **2025-01-06**

Repository / Repozitorij:

[Repository of Faculty of Science](#)



PRIRODOSLOVNO–MATEMATIČKI FAKULTET  
SVEUČILIŠTA U SPLITU

ĐANA KRSTIĆ

**IZABRANI ALGORITMI TEORIJE  
GRAFOVA**

DIPLOMSKI RAD

Split, rujan 2024.

PRIRODOSLOVNO–MATEMATIČKI FAKULTET  
SVEUČILIŠTA U SPLITU

ODJEL ZA MATEMATIKU

**IZABRANI ALGORITMI TEORIJE  
GRAFOVA**

DIPLOMSKI RAD

Student(ica):

Đana Krstić

Mentor(ica):

prof. dr. sc. Damir Vukičević

Split, rujan 2024.

TEMELJNA DOKUMENTACIJSKA KARTICA

PRIRODOSLOVNO–MATEMATIČKI FAKULTET  
SVEUČILIŠTA U SPLITU  
ODJEL ZA MATEMATIKU

DIPLOMSKI RAD  
**IZABRANI ALGORITMI TEORIJE  
GRAFOVA**

Đana Krstić

**Sažetak:**

*Rad je centriran na odabranim algoritmima iz teorije grafova. Pojedini problemi, koji se općenito smatraju NP-teškima, postaju rješivi u polinomijalnom vremenu kad se primijene na specifične podklase savršenih grafova. Rad prikazuje algoritme za provjeru ispravnosti tipa tih grafova i složenost tih algoritama. Osim toga, rad uključuje i algoritam za sortiranje podatka u grafu, bavi se permutacijskim grafovima i njihovim bojanje, te provjeru planarnosti grafa.*

**Ključne riječi:**

*Matrica susjedstva, lista susjedstva, složenost algoritma, tranzitivni turnir, savršeni graf, množenje vrhova, triangulirani graf, savršena eliminacijska shema, leksikografski BFS, tranzitivna orijentacija, graf usporedivosti, forsiranje, klasa boja, dekompozicijska shema, simpleks, TRO algoritam, permutacijski graf, labeliranje permutacije, kanonska strategija sortiranja, planarnost, G-prihvatljiv, blok, komad, mjesta kontakta*

**Podatci o radu:**

*85 stranica, 30 slika, 6 literaturnih navoda, hrvatski*

**Mentor(ica):** *prof. dr. sc. Damir Vukičević*

TEMELJNA DOKUMENTACIJSKA KARTICA

**Članovi povjerenstva:**

*doc. dr. sc. Aljoša Šubašić*

*doc. dr. sc. Tanja Vojković*

Povjerenstvo za diplomski rad je prihvatilo ovaj rad *26. rujna 2024.*

BASIC DOCUMENTATION CARD

FACULTY OF SCIENCE, UNIVERSITY OF SPLIT  
DEPARTMENT OF MATHEMATICS

MASTER'S THESIS  
**SELECTED ALGORITHMS IN GRAPH  
THEORY**

Đana Krstić

**Abstract:**

*The thesis is centered on selected algorithms in graph theory. Certain problems, which are usually considered NP-difficult, solvable in polynomial time when applied to specific subclasses of perfect graphs. The thesisThe paper presents algorithms for verifying the correctness of these graph types and their complexity. Additionally, the thesis includes an algorithm for sorting data stored in as a graph, it handles permutation graphs and their coloring, and verification of graph planarity.*

**Key words:**

*Adjacency matrix, adjacency list, algorithm complexity, transitive tournament, perfect graph, multiplication of vertices, triangulated graph, perfect elimination scheme, lexicographic BFS, transitive orientation, comparability graph, forcing, color class, decomposition scheme, simplex, TRO algorithm, permutation graph, permutation labeling, canonical sorting strategy, planarity, G-admissible, block, piece, points of contact*

**Specifications:**

*85 pages, 30 images, 6 literature citations, Croatian*

**Mentor:** *prof. dr. sc. Damir Vukičević*

**Committee:**

BASIC DOCUMENTATION CARD

*doc. dr. sc. Aljoša Šubašić*

*doc. dr. sc. Tanja Vojković*

This thesis was approved by a Thesis committee on *26th of September, 2024.*

# Uvod

Teorija grafova je grana matematike koja se bavi, kao što iz naziva možemo intuitivno zaključiti, grafovima. Graf je vrsta matematičkog objekta s mnogim primjenama. Najviše se koriste za olakšani prikaz različitih vrsta podataka. Primjerice, želimo li isplanirati rute raznošenja paketa, puno je jednostavnije to prikazati grafički. Potom možemo odrediti najbolje rute raznošenja uz pomoć matematičke teorije. Za samo rješavanje problema moramo osmisliti prikladan algoritam.

Sami algoritmi, metode rješavanja problema, znaju biti jako neefikasni ovisno o veličini problema, dostupnim resursima i sličnim čimbenicima. Cilj matematičara i/ili programera je taj da osmisli algoritme koji će biti što efikasniji tj. koji imaju izvedivu primjenu u pravome svijetu. Neki algoritmi postaju puno efikasniji primjenom matematičkih znanja, a zbog prikazivosti problema u grafičkom obliku, posebnu pozornost ćemo obratiti na znanja iz teorije grafova. Primjena tih znanja može značajno olakšati/ubrzati korake algoritma, no čak i tada neki algoritmi ostaju neprihvatljivi. Takvi algoritmi, u nekim slučajevima, mogu postati prihvatljivi za specifične tipove grafova.

U ovome diplomskome radu ćemo predstaviti nekoliko takvih tipova grafova koji olakšavaju rad algoritama, te algoritme koji ih prepoznaju. Osim spomenutih algoritama, predstaviti ćemo i algoritme koji rade određene funkcije na danom grafu, primjerice sortiranje.



# Sadržaj

Uvod	vii
Sadržaj	viii
<b>1 Uvodni pojmovi</b>	<b>1</b>
1.1 Uvodni pojmovi iz područja teorije grafova . . . . .	1
1.2 Algoritmi i spremanje grafa u računalo . . . . .	7
<b>2 Sortiranje</b>	<b>13</b>
2.1 Topološko sortiranje . . . . .	13
<b>3 Savršeni grafovi</b>	<b>21</b>
3.1 Triangulirani grafovi . . . . .	29
3.2 Graf usporedivosti i TRO . . . . .	43
<b>4 Permutacijski grafovi</b>	<b>61</b>
<b>5 Planarni grafovi</b>	<b>69</b>
5.1 Testiranje planarnosti grafa . . . . .	72
<b>Literatura</b>	<b>77</b>

# Poglavlje 1

## Uvodni pojmovi

Prije nego što počnemo predstavljati algoritme, trebat ćemo se upoznati s osnovnim pojmovima teorije grafova kao što su definicije grafa, ciklusa, bojanja... Ovi pojmovi će se koristiti kroz cijeli rad, a pojmovi specifični određenom poglavlju će se uvesti u spomenutom poglavlju. Osim pojmova iz teorije grafova, uvest ćemo i definiciju algoritma i kako se grafovi mogu spremati u računalo.

### 1.1 Uvodni pojmovi iz područja teorije grafova

Uvest ćemo temeljni pojam rada - samu definiciju grafa. U istoj definiciji uvodimo pojmove grafa i usmjerenog grafa.

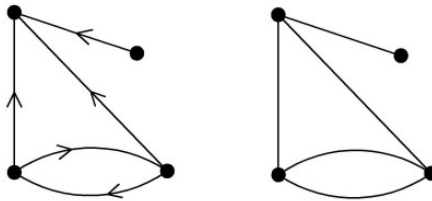
**Definicija 1.1** ***Graf** je uređena trojka  $G = (V, E, \phi)$ , gdje je  $V \neq \emptyset$  skup čije elemente nazivamo **vrhovi**,  $E$  skup čije elemente nazivamo **bridovi**,  $V \cap E = \emptyset$ , te je  $\phi$  funkcija koja svakom bridu pridružuje neuređeni par vrhova. Funkcija  $\phi$  se naziva **incidencijska funkcija**.*

### 1.1. Uvodni pojmovi iz područja teorije grafova

Ako incidencijska funkcija  $\phi$  pridružuje svakom vrhu uređeni par vrhova, tada kažemo da je graf **usmjeren**. Usmjerene bridove iz ovakvog pridruživanja nazivamo **lukovi**.

Kada vrijedi  $\phi(e) = \{u, v\}$ , onda kažemo da su vrhovi  $u$  i  $v$  krajevi brida  $e$ . Kažemo da su vrhovi  $u$  i  $v$  **incidentni** s bridom  $e$ . Kraće pišemo  $e = uv$ , te još kažemo da su vrhovi  $u$  i  $v$  *susjedni*. Kod usmjerenih grafova, kad vrijedi  $\phi(e) = (u, v)$  kažemo da luk  $e$  ima početak u vrhu  $u$  i kraj u vrhu  $v$ .

Graf bez vrhova je također graf i takav graf se naziva **prazan graf**. Uvedimo još i pojam podgrafa.



Slika 1.1: Usmjereni i neusmjereni graf - slika iz skripte *Teorija Grafova* - Anka Golemac

**Definicija 1.2** Za graf  $H$  kažemo da je **podgraf** grafa  $G$  tj.  $H \subseteq G$ , ako  $V_H \subseteq V_G$  i  $E_H \subseteq E_G$ , gdje  $V_H, E_H$  su redom skupovi vrhova i bridova grafa  $H$ ,  $V_G, E_G$  su redom skupovi vrhova i bridova grafa  $G$ .

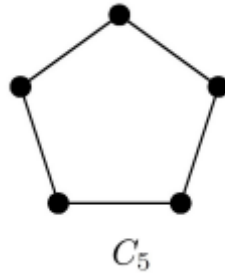
Ovisno o vrhovima i bridovima, postoje i grafovi koji nose posebne nazive.

**Ciklus** sa  $n$  vrhova je neprazni graf sa skupom vrhova  $V = \{v_1, \dots, v_n\}$ , te skupom bridova  $E = \{v_1v_2, v_2v_3, \dots, v_{n-1}v_n, v_nv_1\}$ . Ciklus označavamo sa  $C_n$ . Ciklus duljine  $k$  tj. **k-ciklus**, je ciklus sa  $k$  vrhova i  $k$  bridova. Brid koji povezuje dva vrha ciklusa se naziva **kabel**.

Ciklus u kojem su svaka dva vrha povezana kabelom tvori potpun graf.

### 1.1. Uvodni pojmovi iz područja teorije grafova

**Potpuni graf** je graf kojemu su svaka dva vrha susjedna i označavamo ga sa  $K_n$ .



Slika 1.2: Ciklus sa 5 vrhova - slika iz skripte *Teorija Grafova - Anka Golemac*.

**Definicija 1.3** *Klika* je podskup vrhova  $K$  grafa  $G$  takav da su svaka dva različita vrha iz  $K$  susjedna u  $G$ . Broj klika grafa  $G$  označavamo sa  $\omega(G)$ .

Promotrimo vezu između vrhova i bridova tako da uvedemo pojam stupanj vrha. Za vrh  $v$  kažemo da je **stupnja**  $n$  ako je incidentan sa  $n$  bridova, te pišemo  $\deg(v) = n$ . Uz pojam orijentiranog grafa uvodimo i pojmove ulaznog i izlaznog stupnja. **Izlazni stupanj** vrha  $v$  orijentiranog grafa je broj lukova s početkom u  $v$ , a **ulazni stupanj** vrha  $v$  je broj lukova s krajem u  $v$ . Redom ćemo ih označavati sa  $\text{indeg}(v)$  i  $\text{outdeg}(v)$ . Vrh čiji je izlazni stupanj 0 naziva se **dno**.

Skup vrhova susjednih vrhu  $v$  označavat ćemo sa  $\text{Adj}(v)$ .

Vezu između stupnjeva i bridova prikazujemo tzv. lemom o rukovanju.

**Teorem 1.4 (Lema o rukovanju)** *U svakome grafu  $G = (V, E)$  zbroj stupnjeva je paran. Vrijedi  $\sum_{v \in V} \deg(v) = 2m$ , gdje je  $m$  broj bridova u  $G$ .*

Iz neusmjerenog grafa, možemo dobiti usmjereni što prikazujemo sljedećom definicijom.

### 1.1. Uvodni pojmovi iz područja teorije grafova

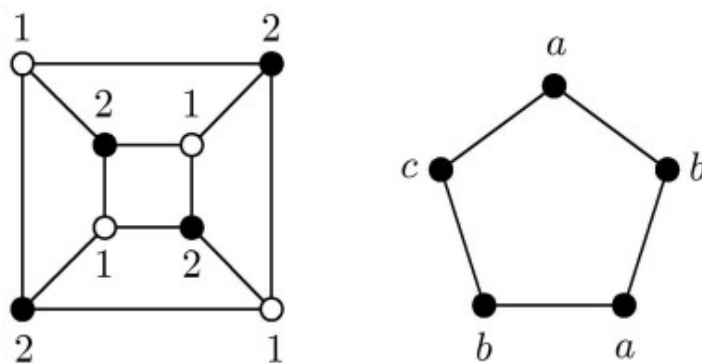
**Definicija 1.5** *Orijentacija* grafa  $G$  je bilo koji usmjereni graf  $\vec{G}$  u kojem brid  $\{u, v\}$  dobiva orijentaciju  $(u, v)$  ili  $(v, u)$ .

Sad uvodimo pojam koji je često korišten u primjeni i, specifičnije, u algoritmima. Pojam bojanja grafa.

**Definicija 1.6** Neka  $B = \{1, \dots, k\}$ . Za graf  $G$ ,  $k$ -bojanje vrhova je preslikavanje  $f : V(G) \rightarrow B$ . Elemente skupa  $B$  nazivamo **bojama**, te kažemo da je vrh  $v$  obojen bojom  $b$  ako  $f(v) = b$ .

Bojanje grafa je **pravilno** ako su susjednim vrhovima pridružene različite boje. Graf je  **$k$ -obojuv** ako postoji pravilno bojanje njegovih vrhova.

Ako je graf  $k$ -obojuv i nije  $(k-1)$ -obojuv, onda kažemo da je **kromatski broj**  $\chi(G)$  grafa  $G$  jednak  $k$ .



Slika 1.3: Pravilno (lijevo) i nepravilno (desno) bojanje - slika iz skripte *Teorija Grafova - Anka Golemac*.

Za graf  $G$ , **stabilan skup** je skup vrhova među kojima nikoja dva nisu susjedna. **Stabilnost grafa**  $\alpha(G)$  je vrijednost koja označava veličinu najvećeg stabilnog skupa u grafu. **Pokrivač** grafa  $G$  je podskup njegovih vrhova takvih da svaki brid grafa ima barem jedan kraj u tom podskupu. Sa  $k(G)$ ,

### 1.1. Uvodni pojmovi iz područja teorije grafova

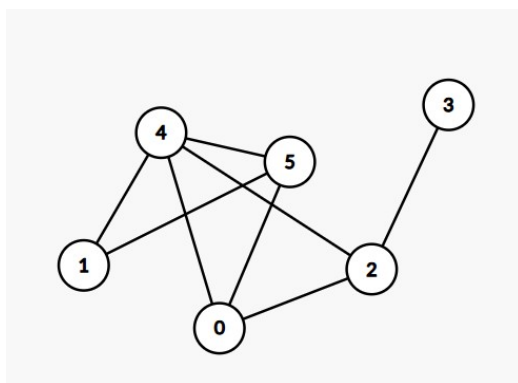
označimo broj pokrivača klike tj. najmanji broj potpunih podgrafova grafa  $G$  potrebnih da se pokriju vrhovi iz  $G$ .

Uvedimo pojam povezanosti grafa. Prije uvođenja pojma povezanosti, definirajmo šetnju, stazu i put.

**Definicija 1.7 Šetnja** grafa  $G$  je konačni niz vrhova  $v_i$  i bridova  $e_i$  oblika  $v_0, e_1, v_1, e_2, \dots, e_l, v_k$ . **Staza** je šetnja čiji su bridovi svi međusobno različiti. **Put** je staza s međusobno različitim vrhovima.

Sada možemo uvesti pojam povezanosti grafa.

**Definicija 1.8** Graf  $G$  je **povezan** ako su mu svaka dva vrha povezana putem, a inače je **nepovezan**. Povezani podgraf (koji nije pravi podgraf ni jednog drugog povezanog podgrafa od  $G$ ) naziva se **komponenta povezanosti** od  $G$ .



Slika 1.4: Povezani graf sa 6 vrhova.

**Primjer 1.9** Promotrimo prethodni graf.  $4 - 0 - 2 - 4 - 5$  bi bio primjer šetnje u grafu. Zbog različitosti bridova, ovo bi također bila i staza, ali zbog ponavljanja vrha 4, ovo nije put.

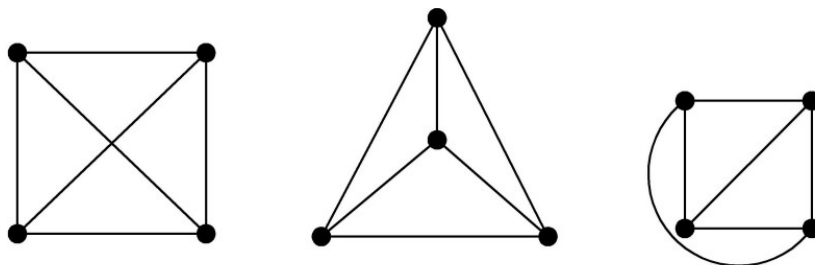
### 1.1. Uvodni pojmovi iz područja teorije grafova

**Definicija 1.10** Podskup  $S \subseteq V$  je **separator vrhova** vrhova  $v$  i  $w$  (koji nisu susjedni) ako uklanjanjem podskupa vrhova  $S$ ,  $v$  i  $w$  se razdvajaju u dvije odvojene komponente povezanosti.  $S$  u tom slučaju nazivamo separator  $v$ - $w$ . U slučaju da ne postoji pravi podskup  $S$  koji je **separator**  $v$ - $w$ , onda kažemo da je  $S$  **minimalni separator vrhova**  $v$  i  $w$ .

**Primjer 1.11** Promotrimo ponovno graf na Slici 1.4. Separator vrhova 4 i 3 (tj. separator 3-4), je  $S = \{2\}$  jer se uklanjanjem vrha 2, vrhovi 3 i 4 razdvajaju u različite komponente povezanosti.

**Definicija 1.12** Za graf kažemo da je **planaran** ili **smjestiv u ravninu** ako se može nacrtati u ravnini bez presijecanja bridova. Područje ravnine određeno smještanjem grafa se nazivaju **strane**.

**Definicija 1.13** Planarno smještenje grafa  $G$  se naziva **ravninski graf**.



Slika 1.5: Potpuni graf s 4 vrha je planaran. - slika iz skripte *Teorija Grafova* - Anka Golemac

**Definicija 1.14** Neka  $G$  graf. Za brid iz grafa  $G$  kažemo da je **most** ako njegovim uklanjanjem se povećava broj komponenti grafa.

Na grafu sa Slike 1.4 brid 23 je most.

## 1.2. Algoritmi i spremanje grafa u računalo

**Teorem 1.15** *Za planarno smještenje povezanog grafa  $G = (V, E)$  vrijedi  $f + |V| - 2 = |E|$ , gdje  $f$  broja strana grafa.*

Sa  $d(f)$  označimo stupanj strane  $f$  grafa  $G$  tj. stupanj strane je broj bridova u njezinom rubu, gdje se rezni bridovi broje dvostruko.

**Lema 1.16** *Za planarno smještenje jednostavnog grafa  $G$  vrijedi*

$$2m = \sum_i d(f_i).$$

**Korolar 1.17** *Za svaki povezani, planarni grafa  $G = (V, E)$ , vrijedi  $|E| \leq 2|V| - 4$ .*

Trebat ćemo još i pojam funkcije.

**Definicija 1.18** *Neka su  $X$  i  $Y$  neprazni skupovi. **Funkcija** je relacija  $f \subseteq X \times Y$  za koju vrijedi:*

$$(\forall x \in X)(\exists! y \in Y) y = f(x).$$

Funkcija iz definicije je *bijekcija* ako vrijedi:

$$(\forall y \in Y)(\exists! x \in X) y = f(x).$$

## 1.2 Algoritmi i spremanje grafa u računalo

Ne postoji stroga definicija algoritma, ali uvedimo *algoritam* kao proces ili niz pravila za rješavanje problema u konačno mnogo koraka. U računalu, algoritmi su niz naredbi napisanih u redoslijedu koje računalo razumije i izvršava, kako bi dobili željeni rezultat.

Kako bi algoritam bio ispravan, treba vrijediti sljedeće:

- 1) Mora biti jednoznačno određen prvi korak.



## 1.2. Algoritmi i spremanje grafa u računalo

- 2) Svaki sljedeći korak mora biti jednoznačno određen.
- 3) Algoritam mora sadržavati konačno mnogo koraka.
- 4) Algoritam se sastoji od konačno mnogo osnovnih instrukcija koje jednoznačno određuju njegov završetak.
- 5) Svaki algoritam ima konačno mnogo ulaznih vrijednosti i one su jedinstvene obzirom na problem koji trebaju riješiti.

U računarstvu, postoji pojam *klasa kompleksnosti*. Klase kompleksnosti grupiraju probleme ovisno o vremenu i prostoru potrebnom za njihovo rješavanje. Za potrebe ovog diplomskog rada, predstavimo P, NP i NP-teške klase. *P* klasa je klasa problema odluke (odgovor na pitanje mora biti da ili ne) koja se može riješiti determinističkom strojem (za svako stanje stroja, sljedeće stanje je jedinstveno određeno) u polinomijalnom vremenu. *NP* klasa je klasa problema odluke koji se mogu riješiti nedeterminističkim strojem u polinomijalnome vremenu. Rješenja *NP* klase je teško naći, ali se verifikacija problema može obaviti u polinomijalnome vremenu. *NP-teška* klasa je klasa problema u kojoj je svaki problem barem težak kao i najteži problem u *NP* klasi. Verifikacija ovakvih problema traje dugo. Primjer *NP*-teškog problema je problem sume podskupa. Problem glasi: Za dani skup prirodnih brojeva i zadanu vrijednost sume, postoji li podskup danog skupa čija je suma jednaka zadanoj.

Uz algoritme većemo i pojam kompleksnosti algoritma. *Kompleksnost algoritama* nam služi da procijenimo može li program uopće završiti u konačnom vremenu i s danim resursima. Postoje različite vrste složenosti, ali najznačajnije su vremenska i prostorna složenost. Prostorna složenost daje procjenu koliko će memorije algoritam zauzeti - svako spremanje varijable, pripremljeno ili ne, oduzima memoriju. Vremenska složenost nam govori vrijeme potrebno za izvršenje algoritma.

## 1.2. Algoritmi i spremanje grafa u računalo

Osim prostorne i vremenski složenosti, postoje još tri vrste kompleksnosti: kompleksnost najboljeg, kompleksnost najgoreg i kompleksnost prosječnog slučaja. Najčešće se gleda kompleksnost najgoreg slučaja, tj. maksimalan broj koraka za izvođenje nekog algoritma  $A$ .

**Definicija 1.19** *Neka su  $f, g : \mathbb{N} \rightarrow \mathbb{R}^+$  funkcije. Kažemo da je  $g$  **asimptotska gornja međa** od  $f$  ako postoji  $M > 0$  i  $n_0 \in \mathbb{N}$  takvi da za svaki  $n \in \mathbb{N}$ ,  $n \geq n_0$  vrijedi  $f(n) \leq Mg(n)$ . Označavamo  $f(n) = O(g(n))$ .*

Oznaka  $O$  se naziva i  $O$ -notacija, te označava gornju granicu ili najgori slučaj.

Rast kompleksnosti može biti konstantan, linearan, polinomijalan, logaritamski, eksponencijalan i faktorijelski. Polinomijalna kompleksnost je prihvatljiva, dok je eksponencijalna neprihvatljiva za velike vrijednosti jer može dovesti do algoritama kojima trebaju godine da se izvrše. Iako će se algoritam izvršiti, smatra se da je toliko vremensko trajanje izvršavanja neprihvatljivo.

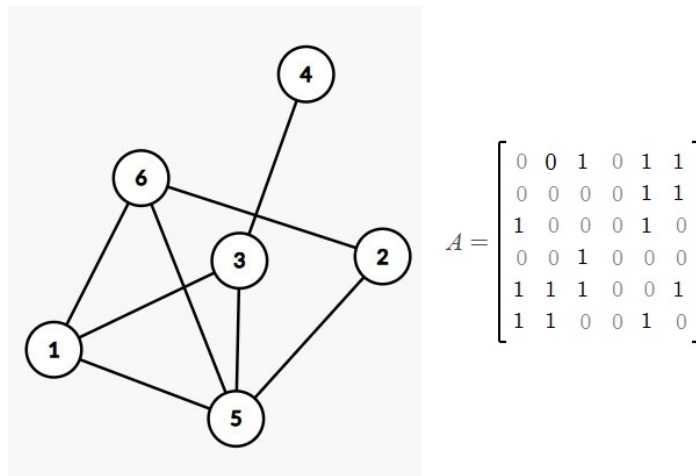
**Primjer 1.20**  $O(1) < O(\log n) < O(\sqrt{n}) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n) < O(10^n) < O(n!)$

Kompleksnost se računa neovisno o mogućnostima računala kao što su brzina procesora, brzina diska, kompilera...

Osvrnimo se sada algoritme primjenjive na probleme teorije grafova. Prije dizajniranja algoritama, moramo grafove spremiti u računalo. Kako bi unijeli graf u računalo, trebamo unijeti sve informacije o grafu. Primarno, mislimo na njihove vrhove i bridove tj. vrhove i njihove susjednosti. Postoje dva glavna načina unosa susjednosti grafa u računalo: matrica i lista susjeda.

**Definicija 1.21** *Neka je  $G = (V, E)$  graf sa  $n$  vrhova. **Matrica susjedstva** grafa  $G$  je  $n \times n$  matrica  $A = [a_{ij}]$ , gdje  $a_{ij}$  broj bridova koji spajaju vrhove  $v_i$  i  $v_j$  iz  $G$ .*

## 1.2. Algoritmi i spremanje grafa u računalo



Slika 1.6: Primjer matrica susjedstva.

Pohranjivanje susjeda na ovaj način je jednostavno jer većina programskih jezika već ima ugrađene matrice ili slične strukture. Ovaj način spremanja bridova nam omogućuje brzo pronalaženje informacije o susjednosti jer je dovoljno provjeriti je li npr. u  $i$ -tom stupcu i  $j$ -tom retku broj različit od 0 kako bi utvrdili susjednost vrhova  $v_i$  i  $v_j$ . Koristeći takvo spremanje, provjeravanje susjednosti se može provesti u konstantnom vremenu - neovisno o količini podataka (uvijek je isti postupak traženja). Problem se jedino može pojaviti kod usmjerenih grafova u kojima je poredak bitan. Tada odredimo da npr. redci predstavljaju početni vrh usmjerenog ruba, a stupci krajnji. Tako npr. u retku  $i$  i stupcu  $j$  dobivamo informaciju o postojanju usmjerenog brida od  $v_i$  do  $v_j$ .

Jedan od nedostataka korištenja matrice susjedstva kao sredstva pohrane grafa je taj da u slučaju da želimo naći sve susjede od vrha  $v_i$  moramo prolaziti kroz cijeli redak/stupac kako bi vidjeli gdje je vrijednost različita od 0. Ovaj postupak je složenosti  $O(n)$ , gdje je  $n$  broj vrhova. Također, još jedan problem je količina potrošene memorije. Za graf s  $n$  vrhova, moramo rezervirati  $n^2$  mjesta u memoriji. S obzirom da neki od vrhova ne moraju biti

## 1.2. Algoritmi i spremanje grafa u računalo

susjedni, prostor se troši nepotrebno na pohranu nula. Ovo se može popraviti sa listom susjeda.

**Lista susjeda** nije jedna lista, već lista lista - svaki vrh grafa ima svoju listu. Na  $i$ -tom mjestu, koje reprezentira vrh  $v_i$  se nalazi lista koja sadrži sve njegove susjede. Umjesto liste možemo podliste susjeda spremati u dictionary tip podatka što je efikasnije. Dictionary je tip podatka koji sprema vrijednosti u parovima - ključ:vrijednost. Za razliku od lista, dictionary nema poredak elemenata. Dictionary je efikasniji od matrice jer olakšava pronalaženje elemenata. Primjerice, za pronaći vrijednost u listi od  $n$  elemenata, u najgorem slučaju trebalo bi  $O(n)$  jer moramo provjeravati svaki element liste. Ovaj postupak u dictionary-u bi trajao  $O(1)$  jer bismo odmah "skočili" na mjesto na kojem se nalazi vrijednost.

**Primjer 1.22** *Promotrimo graf lijevo na Slici 1.6. Spremimo li liste u listu, uzimajući da  $i$ -ta pozicija u listi predstavlja listu susjeda  $i$ -tog vrha, lista susjedstva za graf je:  $[[3, 5, 6], [5, 6], [1, 5], [3], [1, 2, 3, 6], [1, 2, 3]]$ . U obliku dictionary-a, ovo bi izgledalo:  $\{1 : [3, 5, 6] , 2 : [5, 6] , 3 : [1, 5] , 4 : [3] , 5 : [1, 2, 3, 6] , 6 : [1, 2, 3]\}$ .*

Za razliku od matrice susjedstva koja zauzima  $n^2$  mjesta za memoriju, lista susjedstva zauzima  $2m$  mjesta, gdje je  $m$  broj bridova. Npr. ako su vrhovi  $v_i$  i  $v_j$  susjedni, to znači da se  $v_i$  sadržan na listi od  $v_j$  i obrnuto, pa se zauzima manje mjesta. Za usmjerene grafove, samo odaberemo želimo li da pozicija u glavnoj listi određuje je li vrh početni ili krajnji u odnosu na susjedne mu vrhove iz unutarnje liste.

Nedostatak liste je što može biti sporija od matrice ovisno o problemu. Primjerice, za ukloniti brid  $v_i v_j$  moramo prvo provjeriti postoji li uopće - u najgorem slučaju moramo provjeriti sve elemente lista ako nismo sigurni

## 1.2. Algoritmi i spremanje grafa u računalo

na kojoj poziciji se nalazi, a u prosjeku (ako pretpostavimo da je po sredini liste) traje  $O(m/n)$ . Pronalaženje u matrici susjedstva traje, kao što smo ranije napomenuli,  $O(1)$ , što je očito bolje. Samo brisanje iz liste traje  $O(1)$ , pa je ukupno vrijeme potrebno za uklanjanje brida iz liste susjeda  $O(m/n)$ .

Uočimo da je prosječno vrijeme nalaženje svih susjeda od specifičnog vrha u listi susjedstva još uvijek  $O(m/n)$  što je brže od  $O(n)$  potrebnog u matrici susjedstva.

Zaključno, odabir matrice ili liste ovisi većinski o tipu problema, te o količini dostupne memorije.

# Poglavlje 2

## Sortiranje

### 2.1 Topološko sortiranje

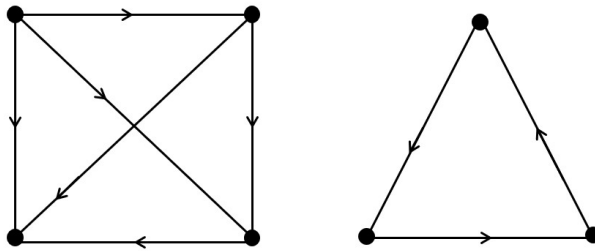
U pravome svijetu, često se događa da imamo niz nekakvih događaja koje imaju različite odnose i prioritete. Kad imamo jednostavan  $A \rightarrow B \rightarrow C$  odnos, trivijalno je odrediti prioritet izvođenja događaja. Nažalost, postoje kompliciranije situacije gdje možda imamo još ovakvih nizova koji su nevezani ili parcijalno vezani. Primjerice, ako pišemo software i postoje ovisnosti u izvršavanju - neki file-ovi ne mogu raditi prije nego što se drugi izvrše. Topološki sort može odrediti optimalni redoslijed pokretanja fileova koji su dio software-a bez da se software sruši. Još neki primjeri su organizacija rasporeda, organizacija odrade posla, optimizacija kompilera ...

Svi ovi odnosi se mogu prikazati u obliku usmjerenog grafa, ali graf ne bi smio imati usmjerenih ciklusa jer bismo onda imali loop koji se ne bi mogao sortirati. Npr. da imamo događaj  $A$  koji povlači događaj  $B$ , ali i  $B$  povlači  $A$ , to se ne bi moglo sortirati jer ne postoji jasno definiran završetak. Dakle, topološko sortiranje se primjenjuje za pronalaženje redoslijeda izvršavanja radnji prikazivih pomoću usmjerenog acikličkog grafa što ćemo prikazati u

## 2.1. Topološko sortiranje

ovome poglavlju.

**Definicija 2.1** Neka  $G = (V, E)$  potpuni usmjereni graf s  $n$  vrhova. Za orijentaciju  $\vec{G}$  kažemo da je **tranzitivni turnir** ako za svaku trojku vrhova  $x, y, z \in V$ , vrijedi da  $xy \in E$  i  $yz \in E$  povlači  $xz \in E$ .



Slika 2.1: Tranzitivni turnir (lijevo) i graf čija orijentacija nije tranzitivni turnir (desno).

Drugim riječima,  $E$  nema 3-ciklusa. Tranzitivni turnir je usmjereni aciklički graf na kojem je primjenjivo topološko sortiranje spomenuto ranije.

Sljedeći teorem daje algoritam u linearnom vremenu za prepoznavanje tranzitivnih turnira.

**Teorem 2.2** Neka  $\vec{G}$  orijentacija potpunog grafa  $K_n$ . Sljedeće tvrdnje su ekvivalentne:

(i)  $\vec{G}$  je tranzitivni turnir

(ii)  $\vec{G}$  je acikličan

Štoviše, postoji linearno uređenje vrhova  $[v_1, \dots, v_n]$  takvo da vrijede sljedeća svojstva:

(iii)  $\text{indeg}(v_i) = i - 1, \forall i$

(iv)  $v_i v_j \in \vec{G}$  ako i samo ako  $i < j$

## 2.1. Topološko sortiranje

**Dokaz.**  $(i) \Rightarrow (ii)$  Iz definicije tranzitivnog turnira proizlazi činjenica da nema 3-ciklusa. Pretpostavimo da graf sadrži  $k$ -ciklus, gdje je  $k$  najmanji mogući. Tada postoji kabel koji povezuje 2 vrha ciklusa i tvori manji ciklus, što je kontradikcija s minimalnošću od  $k$ . Graf je acikličan.

$(ii) \Rightarrow (iii)$  Ako je  $\vec{G}$  acikličan, onda on ima dno i označimo ga s  $v_n$ . Vrijedi  $\text{indeg}(v_n) = n - 1$  zbog potpunosti. Uklanjanjem  $v_n$  iz grafa, dobiva se manji aciklički usmjereni graf. Indukcijom ponavljamo postupak i dokazujemo da tvrdnja vrijedi.

$(iii) \Rightarrow (iv)$  Indukcijom.

$(iv) \Rightarrow (i)$  Očito. ■

Prethodni teorem nam daje da je za određivanje je li graf tranzitivni turnir, dovoljno izračunati sve  $\text{indeg}$  i potvrditi da nema jednakih.

Postoji problem generalniji od problema provjere da li je orijentacija grafa tranzitivni turnir, a to je provjera da li je graf acikličan i usmjeren (svaki tranzitivni turnir je ujedno i aciklični, usmjereni graf). Algoritmi koji rade s acikličkim, usmjerenim grafovima imaju širu primjenu za razliku od tranzitivnih turnira. Jedan takav algoritam je algoritam topološkog sortiranja.

**Definicija 2.3** *Linearno sortiranje*  $[v_1, \dots, v_n]$  grafa  $G$ ,  $v_i \in V$ , koje zadovoljava svojstvo  $v_i v_j \in \vec{G} \rightarrow i < j, \forall i, j$  naziva se **topološko sortiranje** grafa  $G$ .

- 
- 1: **for**  $j \leftarrow |V|$  **to** 1 **step**  $-1$  **do**
  - 2:     Nađi dno  $v$  od preostalog grafa i označi ga s  $v_j$ ;
  - 3:     Izbriši  $v$  i sve rubove incidentne s  $v$  iz grafa;
  - 4: **end for**
- 

Prethodni algoritam prolazi kroz sve vrhove trenutnog grafa, te pronalazi



## 2.1. Topološko sortiranje

dno koje sigurno postoji zbog usmjerene acikličnosti grafa. Labelira ga i uklanja mu incidentne bridove, te ponavlja postupak na manjem tranzitivnom turniru (zbog svojstva (iv) iz Teorema 2.2). Uklanjanjem i labeliranjem dna osigurava da se npr. u slučaju  $u \rightarrow v$ ,  $v$  u finalnom topološkom sortiranju pojavi nakon  $u$ .

Predstavimo jednu implementaciju prethodnog algoritma. Algoritam će primiti usmjereni graf  $G$ , te vraćati poruku ako nije acikličan. Ako je, vraća *SORT* tj. topološki sortiranu listu vrhova grafa, te listu *DFSNUM* u koju spremamo numeraciju vrhova.

Koristimo i pomoćnu funkciju *TOPSORT*() koja radi na principu DFS (*Depth-First-Search*). DFS je poznati način pretraživanja grafa koji kreće od danog početnog vrha, te rekurzivno posjećuje svaki susjedni vrh prije povratka na prethodni. DFS algoritam prolazi što je dublje moguće kroz svaku granu prije nego što prijeđe na sljedeću.

*TOPSORT* za dani vrh  $x$  prvo označi da je posjećen (*DFSNUM* pridruži vrijednost  $i$ ). Potom prolazi kroz sve susjede od  $x$  i provjerava jesu li već posječeni (nisu ako je *DFSNUM* od susjeda jednak 0). Ako nije posjećen, rekurzivno se na susjedu poziva *TOPSORT*. Ako je posjećen, provjerava se je li sortiran (je li *SORT* različit od 0). Ako nije, graf nije acikličan. Na kraju, *TOPSORT* sortira  $x$ .

Pokažimo implementaciju topološkog sortiranja temeljenog na prethodno opisanome:

## 2.1. Topološko sortiranje

---

**Algorithm 1** Topološko sortiranje

---

**Input:** Usmjereni graf  $G$  pohranjen kao lista susjeda

**Output:**  $SORT$  (topološki sortirani vrhovi iz  $G$ ). Alternativno, poruka o acikličnosti grafa.

```
1: for each  $x \in V$  do
2:    $DFSNUM(x) \leftarrow 0$ ;
3:    $SORT(x) \leftarrow 0$ ;
4: end for
5:  $j \leftarrow |V|$ ;
6:  $i \leftarrow 0$ ;
7: for each  $x \in V$  do
8:   if  $DFSNUM(x) == 0$  then
9:      $TOPSORT(x)$ ;
10:  end if
11: end for
12: return  $DFSNUM, SORT$ ;
```

---

Algoritam za  $TOPSORT()$  je sljedeći:

## 2.1. Topološko sortiranje

---

**Algorithm 2** TOPSORT

---

**Input:** Vrh  $x$ , te  $i, j$ , SORT, DFSNUM i susjedi iz originalnog algoritma.

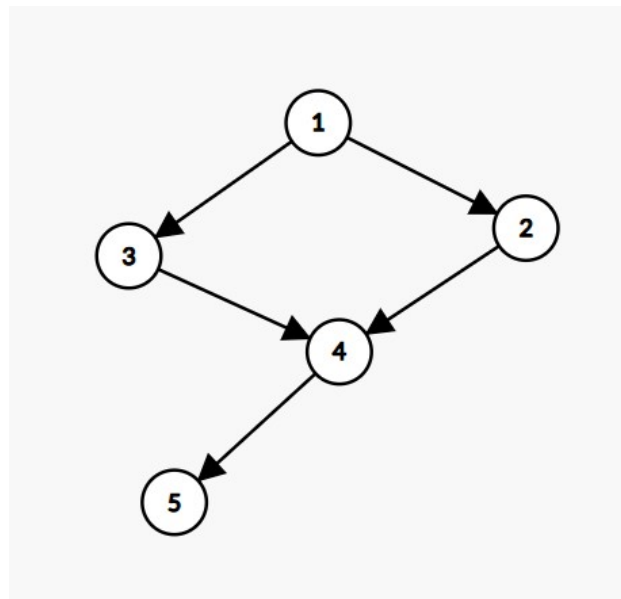
**Output:** Poruka o cikličnosti  $G$

```
1:  $i \leftarrow i + 1$ 
2:  $DFSNUM(x) \leftarrow i$ 
3: for each  $y \in Adj(x)$  do
4:   if  $DFSNUM(y) == 0$  then
5:      $TOPSORT(y)$ 
6:   else if  $SORT(y) == 0$  then
7:     return "G nije acikličan"
8:   end if
9: end for
10:  $SORT(x) \leftarrow j$  //smanjujemo labelu od  $x$  na manju od prethodno dodeljene
11:  $j \leftarrow j - 1$ 
```

---

Promotrimo postupak topološkog sortiranja na primjeru.

## 2.1. Topološko sortiranje



Slika 2.2: Graf s 5 vrhova.

Provedimo algoritam na prethodnom grafu. Pretpostavimo da je susjednost spremljena u listu. Sortiranje započnimo od vrha 1.

Rekurzivni proces algoritma je vidljiv na slici ispod:

$$\begin{array}{l}
 \text{TOPSORT(1)} \left\{ \begin{array}{l} \text{DFSNUM(1)} \leftarrow 1 \\ \text{TOPSORT(2)} \\ \text{TOPSORT(3)} \\ \text{SORT(1)} \leftarrow 1 \end{array} \right. \left\{ \begin{array}{l} \text{DFSNUM(2)} \leftarrow 2 \\ \text{TOPSORT(4)} \\ \text{SORT(2)} \leftarrow 3 \end{array} \right. \left\{ \begin{array}{l} \text{DFSNUM(4)} \leftarrow 3 \\ \text{TOPSORT(5)} \\ \text{SORT(3)} \leftarrow 4 \end{array} \right. \left\{ \begin{array}{l} \text{DFSNUM(5)} \leftarrow 4 \\ \text{SORT(5)} \leftarrow 5 \end{array} \right. \\
 \left\{ \begin{array}{l} \text{DFSNUM(3)} \leftarrow 5 \\ \text{SORT(3)} \leftarrow 2 \end{array} \right.
 \end{array}$$

Počinjemo sa  $\text{TOPSORT}(1)$ , te se redom ugnježđeno pozivaju  $\text{TOPSORT}(2)$ ,

## 2.1. Topološko sortiranje

$TOPSORT(4)$  i  $TOPSORT(5)$ . Nakon što se ponovno vratimo u  $TOPSORT(1)$ , poziva se još i  $TOPSORT(3)$ .

Finalni rezultat sortiranja je:

Vrh	DFSNUM	SORT
1	1	1
2	2	3
3	5	2
4	3	4
5	4	5

# Poglavlje 3

## Savršeni grafovi

Savršeni grafovi su bitni u računalstvu zbog svoje široke primjene u rješavanju optimizacijskih problema, te dizajnu algoritama. Podsjetimo se oznaka uvedenih u uvodnom poglavlju potrebnih za definiranje savršenog grafa.

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

$\omega(G)$  predstavlja veličinu maksimalne klike grafa  $G$  tj. broj njezinih vrhova.

$\chi(G)$  predstavlja kromatski broj grafa  $G$  tj. minimalan broj boja potrebnih za pravilno bojanje vrhova. Uočimo da je skup vrhova koji su obojani istom bojom stabilan.

$\alpha(G)$  predstavlja veličinu najvećeg stabilnog skupa grafa  $G$ .

$k(G)$  predstavlja kardinalnost najmanjeg pokrivača grafa klikama tj. minimalan broj potpunih podgrafova  $A_i$  takvih da  $V = A_1 \cup \dots \cup A_{k(G)}$ .

**Teorem 3.1** *Vrijedi  $\omega(G) = \alpha(G^C)$ , te  $\chi(G) = k(G^C)$ , gdje  $G^C$  komplement grafa  $G$ .*

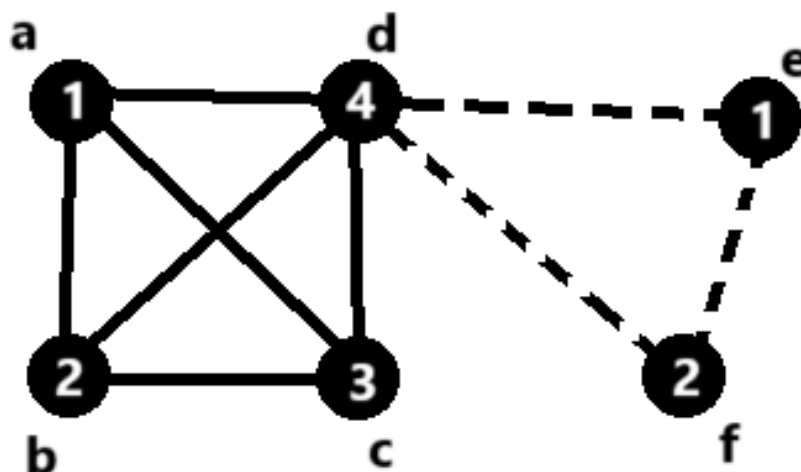
**Dokaz.** Znamo da vrijedi da  $e \in E$  ako i samo ako  $e \notin E^C$ . Neka je  $k \in \mathbb{N}_0$ .

Ako je  $\omega(G) = k$  tj.  $k$  je veličina maksimalne klike, onda je  $k$  vrhova iz klike međusobno povezano. U  $G^C$ , ti vrhovi neće biti povezani pa će tvoriti

stabilni skup veličine  $k$  koji će biti maksimalan zbog maksimalnosti klike. Suprotni smjer je analogan, pa vrijedi  $\omega(G) = \alpha(G^C)$ .

Neka je sad  $\chi(G) = k$  tj.  $V = B_1 \cup \dots \cup B_k$ , gdje je  $B_i$  ( $i = 1, \dots, k$ ) skup vrhova obojan u boju  $i$ . Skupovi  $B_i$  su stabilni. U  $G^C$ , vrhovi iz klase  $B_i$  će svi biti međusobno povezani, pa će vrhovi u  $B_i$  u  $G^C$  tvoriti potpuni skup tj. kliku od  $G^C$ . Zaključujemo  $k(G^C) = k = \chi(G)$ .

■



**Primjer 3.2** Promotrimo graf  $G$  na Slici 3.2. Vrhovi grafa su označeni s  $V = \{a, b, c, d, e, f\}$ .

$\chi(G) = 4$ , te je na slici prikazano jedno bojanje gdje su boje označene s brojevima od 1 do 4.

$\alpha(G) = 2$ , te je  $\{b, e\}$  primjer stabilnog skupa grafa.

$\omega(G) = 4$ , te je maksimalna klika podgraf od  $G$  induciran s vrhovima  $\{a, b, c, d\}$  (bridovi nacrtani ispunjenom linijom).

$k(G) = 2$ , te se  $V$  može prikazati kao suma klike  $abcd$  (ispunjena linija) i klike  $def$  (iscrtkana linija).

Označimo sa  $G_A$  podgraf grafa  $G$  induciran skupom vrhova  $A \subseteq V$ .

**Definicija 3.3** Neusmjereni graf  $G$  sa svojstvima:

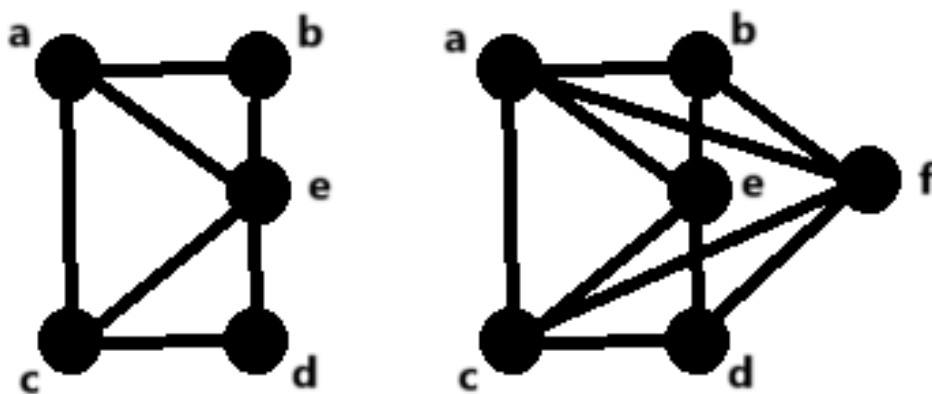
$$S1) \omega(G_A) = \chi(G_A), \forall A \subseteq V$$

$$S2) \alpha(G_A) = k(G_A), \forall A \subseteq V$$

naziva se **savršeni graf**.

Iz teorema 3.1 slijedi da ako vrijedi S1) za graf  $G$ , onda vrijedi S2) za njegov komplement  $G^C$ .

Uvedimo pojam *množenja vrhova*. Neka je  $G$  graf sa vrhom  $v$ . Graf  $Gv$  dobivamo dodavanjem novog vrha  $w$  koji je susjedan svim  $u \in Adj(v)$ . Tada kažemo da je  $Gv$  dobiven množenjem vrha  $v$  sa  $G$ .



Slika 3.1: Graf  $G$  (lijevo) i graf  $Gv$  (desno).

Pojam množenja grafa možemo i proširiti.



Neka je  $G = (V, E)$  neusmjereni graf, te neka  $v_1, \dots, v_n \in V$ . Neka je  $h = (h_1, \dots, h_n)$  vektor takav da  $h_i \in \mathbb{N}_0$ ,  $i = 1, \dots, n$ . Za graf  $H = Gh$  dobiven zamjenom svakog  $v_i$  sa stabilnim skupom  $\{v_i^1, \dots, v_i^{h_i}\}$  sa  $h_i$  elemenata, te za koji vrijedi:  $v_i^q v_j^k \in E_H$  ako i samo ako  $v_i v_j \in E$ , kažemo da je dobiven iz  $G$  **množenjem vrhova**.

Može se pokazati ekvivalencija svojstava iz definicije savršenog grafa. Svojstva  $S1)$  i  $S2)$  će biti ekvivalentna i sa trećim svojstvom :

$$S3) \quad \omega(G_A)\alpha(G_A) \geq |A|, \forall A \subseteq V.$$

Uvedimo prvo dvije pomoćne leme.

**Lema 3.4** *Neka je  $H = Gh$  graf dobiven iz  $G = (V, E)$  množenjem vrhova.*

*Vrijede sljedeće tvrdnje:*

- 1) *Za  $G$  vrijedi svojstvo  $S1) \Rightarrow$  Za  $H$  vrijedi svojstvo  $S1)$ .*
- 2) *Za  $G$  vrijedi svojstvo  $S2) \Rightarrow$  Za  $H$  vrijedi svojstvo  $S2)$ .*

**Dokaz.** Provedimo dokaz indukcijom. Lema vrijedi za  $|V| = 1$ .

Pretpostavimo da lema vrijedi za sve grafove sa manje od  $|V|$  vrhova. Neka  $H = Gh$ . Ako je  $h_i = 0$ , za neki  $i$ , onda se  $H$  dobije iz  $G - v_i$  množenjem vrhova. Primijetimo da ako vrijedi  $S1)$  za  $G$ , onda vrijedi i za  $G - v_i$ . Analogno za  $S2)$ . Iz pretpostavke indukcije vrijede svojstva 1) i 2).

Sada možemo pretpostaviti da je svaki  $h_i \geq 1$ . Pošto se  $H$  može dobiti kao niz manjih množenja, dovoljno je pokazati rezultat za  $H = Gv$ . Sa  $w$  označimo dodanu kopiju od  $v$ .

[1] Pretpostavimo da za  $G$  vrijedi  $S1)$ . Vrijedi  $\omega(Gv) = \omega(G)$  zbog nesusjednosti od  $v$  i  $w$ . Neka je  $\omega(G)$  broj boja s kojima je  $G$  obojen, te neka su  $v$  i  $w$  obojani istom bojom. Tada smo dobili bojanje od  $Gv$  sa  $\omega(Gv)$  boja, pa za  $Gv$  vrijedi  $S1)$ .

Krenemo li od  $\chi(G)$ , znamo da se  $v$  i  $w$  mogu obojati istom bojom u  $Gv$  zbog njihove nesusjednosti, pa  $\chi(G) = \chi(Gv)$ . Pošto su  $v$  i  $w$  obojani istom bojom, vrijedi  $\omega(Gv) = \omega(G)$ , pa vrijedi  $S1$ ) za  $Gv$ .

Pokažimo da  $S1$ ) vrijedi i za sve inducirane mu podgrafe. Neka  $B \subseteq V(Gv)$ . Ako je  $w \in B$ , onda je  $B$  inducirani podgraf od  $G$ , pa vrijedi  $S1$ ). Neka  $w \notin B$  tj. neka je  $B = B_1 \cup \{w\}$  inducirani podgraf od  $Gv$ .  $\omega(G_B)$  je ili jednak  $\omega(G_{B_1})$  (ako postoje vrhovi maksimalne klike od  $B_1$  s kojima  $w$  nije susjedan) ili jednak  $\omega(G_{B_1}) + 1$  (ako je  $w$  susjedan sa svim vrhovima iz maksimalne klike od  $B_1$  - tada tvori veću kliku). Analogno na  $\chi(G_B)$ . Iz pretpostavke  $S1$ ) za  $G$  vrijedi ili  $\omega(G_B) = \omega(B_1) + 1 \Rightarrow \chi(G_B) = \chi(G_{B_1}) + 1$  ili  $\omega(G_B) = \omega(B_1) \Rightarrow \chi(G_B) = \chi(G_{B_1})1$ , pa  $S1$ ) vrijedi za svaki inducirani podgraf od  $Gv$ .

2) Pretpostavimo da za  $G$  vrijedi  $S2$ ). Tvrdimo da  $\alpha(Gv) = k(Gv)$ . Sa  $N$  označimo pokrivač klika od  $G$ ,  $|N| = k(G) = \alpha(G)$ ,  $K_v$  klika od  $N$  koja sadrži  $v$ . Tada može vrijediti jedno od sljedećega:

2.1)  $v$  je sadržan u stabilnom skupu  $S$  od  $G$  tj.  $|S| = \alpha(G)$ .

Tada je  $S \cup \{w\}$  stabilan skup od  $Gv$ , pa  $\alpha(Gv) = \alpha(G) + 1$ . Pošto je  $N \cup \{w\}$  pokrivač od  $Gv$ , vrijedi  $k(Gv) \leq k(G) + 1 = \alpha(G) + 1 = \alpha(Gv) \leq k(Gv)$  tj.  $\alpha(Gv) = k(Gv)$ .

Pokažimo da vrijedi za svaki  $B \subseteq Gv$ . Odaberimo proizvoljni  $B$ . Ako su  $v, w \in B$ , onda vrijedi analogna logika iz prethodno provedenog dokaza 2.1), te je  $S \cup \{w\}$  stabilan skup od  $G_B$  i  $\alpha(G_B) = \alpha(G_{B-v}) + 1$ . Ako  $w \notin B$ , onda  $S1$ ) za  $Gv$  slijedi direktno iz pretpostavke. Slučaj kad  $w \in B, v \notin B$  ne treba promatrati jer je  $w$  uveden kao kopija  $v$ , pa nema smisla gledati slučaj kad  $w \in B, v \notin B$ .

2.2) Niti jedan maksimalni stabilni skup ne sadrži  $v$ .

Tada  $\alpha(Gv) = \alpha(G)$ . Svaka klika od  $N$  siječe maksimalni stabilni skup

točno jednom, ovo posebno vrijedi i za  $K_v$ .

$v$  nije element niti jednog maksimalnog, stabilnog skupa, pa  $D = K_v - \{v\}$  siječe svaki maksimalni stabilni skup od  $G$  točno jednom. Ovo povlači:

$$\alpha(G_{V-D}) = \alpha(G) - 1 \Rightarrow k(G_{V-D}) = \alpha(G_{V-D}) = \alpha(G) - 1 = \alpha(G_v) - 1$$

Pokrivač od  $G_v$  dobivamo tako da uzmemo pokrivač klike od  $G_{V-D}$  veličine  $\alpha(G_v) - 1$  sa dodanom klikom  $D \cup \{w\}$ . Tada  $\alpha(G_v) = k(G_v)$ .

Promotrimo opet podgraf od  $G_v$  induciran skupom vrhova  $B$ . Ako ni  $v$  ni  $w$  nisu u  $B$ , vrijedi trivijalno iz pretpostavke. Ako su oba u  $B$ , onda vrijedi ista logika opisana u 2.2) i za  $G_v$  jer  $w$  ne povećava broj najvećeg stabilnog skupa grafa, nego samo tvori novu kliku sa  $D$ . Ako je  $v$  u  $B$ , a  $w$  nije, onda se  $S2)$  nasljeđuje iz  $G$ , jer nismo uveli novi vrh. Tada  $S2)$  vrijedi za  $G_v$ .

■

**Lema 3.5** *Neka je  $G$  neusmjereni graf čiji koji zadovoljava  $S2)$ , te neka je  $H$  graf dobiven iz  $G$  množenjem vrhova. Ako za  $G$  vrijedi  $S3)$ , vrijedi i za  $H$ .*

**Dokaz.** Neka vrijedi  $S3)$  za  $G$ . Pretpostavimo suprotno - odaberimo  $H$  takav da se dobije množenjem najmanjeg mogućeg broja vrhova iz  $G$ , ali neka je  $H$  i takav da za njega ne vrijedi  $S3)$ . Tada  $\omega(H)\alpha(H) \leq |X|$ , gdje  $X$  vrhovi od  $H$  i  $S3)$  ne vrijedi za svaki inducirani podgraf od  $H$ . Smijemo pretpostaviti da se  $G$  množi barem sa 1 (iz dokaza Leme 3.4), te da se neki vrh u množi sa  $h \geq 2$ .

Neka su  $U = \{u^1, \dots, u^h\}$  vrhovi od  $H$  susjedni  $u$ . Promotrimo vrh  $u^1$ . Zbog minimalnosti od  $H$ ,  $S3)$  vrijedi za  $H_{X-u^1}$ . Slijedi:

$$|X| - 1 = |X - u^1| \stackrel{(*)}{\leq} \omega(H_{X-u^1})\alpha(H_{X-u^1}) \leq \omega(H)\alpha(H) \stackrel{(**)}{\leq} |X| - 1.$$

(\*) - jer  $H_{X-u^1}$  inducirani podgraf od  $H$ , pa primijenimo  $S3$ ).

(\*\*) - jer smo pretpostavili  $\omega(H)\alpha(H) \leq |X|$ .

$$\text{Sada } \omega(H_{X-u^1})\alpha(H_{X-u^1}) = \omega(H)\alpha(H)$$

Definirajmo:

$p = \omega(H_{X-u^1}) = \omega(H)$ , jer uklanjanje  $u^1$  ne mijenja veličinu maksimalne klike,

$q = \alpha(H_{X-u^1}) = \alpha(H)$ , jer uklanjanje  $u^1$  ne mijenja veličinu maksimalnog stabilnog skupa,

$$pq = \omega(H)\alpha(H) = |X| - 1.$$

Iz Leme 3.4 slijedi da  $H_{X-U}$  zadovoljava  $S2$ ) (jer  $H_{X-U}$  dobiven iz  $G - u$  množenjem vrhova). Tada se  $H_{X-U}$  može pokriti sa  $q$  potpunih podgrafova od  $H$  i neka su  $K_1, \dots, K_q$  ti podgrafovi. Smijemo pretpostaviti da su oni međusobno disjunkt i da vrijedi  $|K_1| \geq \dots \geq |K_q|$ .

Vrijedi:

$$\sum_{i=1}^q |K_i| = |X - U| = |X| - h = pq - (h - 1)$$

.

Zbog  $|K_i| \leq p$ , najviše  $h - 1$  od  $K_i$  ne pridonose sumi, pa  $|K_1| = \dots = |K_{q-h+1}| = p$ .

Neka  $H'$  podgraf od  $H$  induciran sa  $X' = K_1 \cup \dots \cup K_{q-h+1} \cup \{u^1\}$ . Tada  $|X'| = p(q - h + 1) + 1 < pq + 1 = |X|$ .

Iz minimalnosti od  $H$  slijedi da  $\omega(H')\alpha(H') \geq |X'|$ .

Iz  $p = \omega(H) \geq \omega(H')$ , slijedi  $\alpha(H') \geq |X'|/p > q - h + 1$ .

Neka  $S'$  stabilni skup od  $H'$  veličine  $q - h + 2$ .  $u^1 \in S'$  jer bi inače, po

definiciji  $H'$ , sadržavao dva vrha kilke.

Sada je  $S = S' \cup U$  stabilni skup od  $H$  veličine  $q + 1$ , što je kontradikcija s definicijom  $q$  (jer je  $q$  po definiciji maksimalna veličina stabilnog skupa).

■

**Teorem 3.6 (Teorem savršenog grafa)** *Za neusmjereni graf  $G = (V, E)$ , sljedeće tvrdnje su ekvivalentne:*

$$S1) \omega(G_A) = \chi(G_A), \forall A \subseteq V$$

$$S2) \alpha(G_A) = k(G_A), \forall A \subseteq V$$

$$S3) \omega(G_A)\alpha(G_A) \geq |A|, \forall A \subseteq V$$

**Dokaz.**

$S1) \Rightarrow S3)$  Pretpostavimo da  $\omega(G_A)$  broj boja s kojim možemo obojati graf  $G_A$ . Pošto ima maksimalno  $\alpha(G_A)$  vrhova dane boje, slijedi da  $\omega(G_A)\alpha(G_A) \geq |A|$ .

$S3) \Rightarrow S1)$  Neka  $G = (V, E)$  zadovoljava svojstvo  $S3)$ . Indukcijom po broju vrhova se može pokazati da svaki podgraf induciran od pravog podskupa vrhova u  $G$  zadovoljava  $S1) - S3)$ . Za  $|V| = 1$ ,  $S1)$  vrijedi trivijalno. Pretpostavimo da teorem vrijedi za sve grafove sa manje vrhova od  $G$ . Pokažimo i da  $\omega(G) = \chi(G)$ .

Da je  $S$  stabilan skup od  $G$  takav da  $\omega(G_{V-S}) < \omega(G)$ , mogli bismo  $S$  obojati jednom bojom, a  $G_{V-S}$  s preostalih  $\omega(G) - 1$  boja. Tada imamo  $\chi(G) \leq \omega(G)$ .

Pretpostavimo da  $\chi(G) < \omega(G)$ . Neka  $G_{V-S}$  ima  $\omega(G)$ -kliku  $K(S)$  za svaki stabilni  $S$  od  $G$ . Označimo sa  $\mathbb{S}$  kolekciju svih stabilnih skupova od  $G$ . Uočimo da vrijedi  $S \cap K(S) = \emptyset$ . Za svaki  $x_i \in V$ , sa  $h_i$  označimo broj klika  $K(S)$  koje sadrže  $x_i$ . Neka  $H = (X, F)$  dobiven iz  $G$  množenjem  $x_i$  sa  $h_i$ . Po Lemi 3.5 vrijedi  $\omega(H)\alpha(H) \geq |X|$ . Vrijedi:

### 3.1. Triangulirani grafovi

$$|X| = \sum_{x_i \in V} h_i = \sum_{S \in \mathbb{S}} |K(S)| = \omega(G)|\mathbb{S}|,$$

$$\omega(H) \leq \omega(G),$$

$$\alpha(H) = \max_{T \in \mathbb{S}} \sum_{X_i \in T} h_i = \max_{T \in \mathbb{S}} \left[ \sum_{S \in \mathbb{S}} |T \cap K(S)| \right] \leq |\mathbb{S}| - 1$$

Iz ovoga slijedi  $\omega(H)\alpha(H) \leq \omega(G)(|\mathbb{S}| - 1) < |X|$  što je kontradikcija.

S2)  $\iff$  S3) Iz prethodno dokazanog slijedi :

$G$  zadovoljava S2)  $\iff G^C$  zadovoljava S1)  $\iff G^C$  zadovoljava S3)  
 $\iff G$  zadovoljava S3) ■

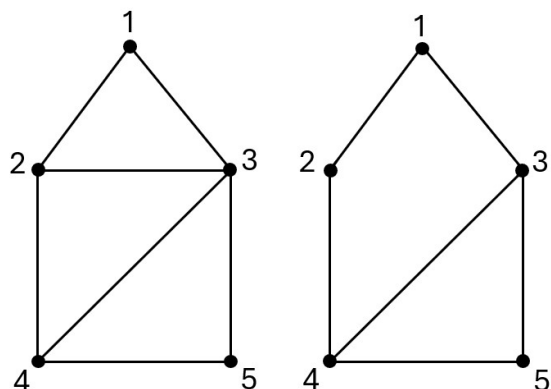
## 3.1 Triangulirani grafovi

Jedan primjer savršenog grafa je tzv. triangulirani graf. Triangulirani grafovi imaju brojne primjene u računarstvu, ali glavna im je uloga da se njihovim korištenjem pojednostavljaju brojni algoritmi. Primjerice, triangulirani grafovi se mogu koristiti u optimiziranju redoslijeda upita koji se unose u baze podataka, u radu na Bayesovim neuronskim mrežama, modeliranju društvenih mreža, u biologiji za razumijevanje interakcije gena...

Još jedan dobar primjer je pronalaženje maksimalne klike trianguliranog grafa. Problem pronalaženja maksimalne klike se smatra NP teškim problemom. Na trianguliranim grafovima, taj problem se može riješiti u linearnome vremenu.

**Definicija 3.7** Za neusmjereni graf  $G$  kažemo da je **trianguliran** ako svaki ciklus duljine strogo veće od 3 sadrži kabl.

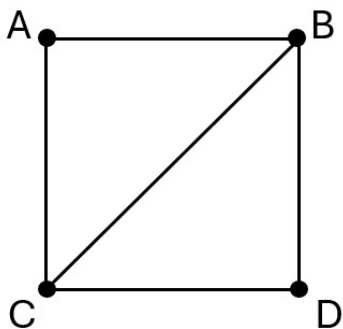
### 3.1. Triangulirani grafovi



Slika 3.2: Primjer trianguliranog (lijevo) i netrianguliranog (desno) grafa.

Za vrh  $v$  grafa  $G$  kažemo da je *jednostavan* ako  $\text{Adj}(v)$  inducira potpuni podgraf od  $G$  tj.  $\text{Adj}(v)$  je klika iako ne nužno maksimalna.

Posebno, vrh  $w$  stupnja 0 također smatramo jednostavnim jer je  $\text{Adj}(w) = \emptyset$  trivijalno klika.



Slika 3.3: Graf s 4 vrha.

**Primjer 3.8** U grafu 3.3,  $A$  i  $D$  su jednostavni vrhovi, a  $B$  i  $C$  nisu jer brid  $\{A, D\}$  nije u grafu.

Može se pokazati da svaki triangulirani graf ima barem dva jednostavna vrha (dokaz kasnije u poglavlju), te su 1965. godine Fulkerson i Gross

### 3.1. Triangulirani grafovi

predložili iterativni način prepoznavanja triangularnih grafova temeljen na ovome svojstvu. Ideja je ta da se ponavljano traže jednostavni vrhovi i eliminiraju iz grafa sve dok ili ostanemo bez vrhova - i time dokažemo da je graf trianguliran - ili dođemo do toga da više nemamo jednostavnih vrhova - tada graf nije trianguliran.

Za algebarsko definiranje prethodno opisanog algoritma trebamo definirati *savršenu shemu*.

**Definicija 3.9** Neka  $G = (V, E)$  neusmjereni graf i neka  $\sigma = [v_1, \dots, v_n]$  jedan poredak njegovih vrhova. Kažemo da je  $\sigma$  **savršena shema za uklanjanje vrhova** ili *savršena shema* ako je svaki  $v_i$  jednostavan vrh inducirano podgrafa  $G_{v_i, \dots, v_n}$ .

Drugim riječima, svaki skup  $X_i = \{v_j \in Adj(v_i) \mid j > i\}$  je potpun.

Uočimo da savršena shema ne mora biti jedinstvena.

**Primjer 3.10** Promotrimo graf lijevo na Slici 3.2. Njegova shema nije jedinstvena. Primjerice,  $[1, 2, 4, 3, 5]$  i  $[5, 4, 3, 2, 1]$  su sheme zadanog grafa.

**Teorem 3.11** Neka  $G = (V, E)$  neusmjereni graf. Sljedeće tvrdnje su ekvivalentne:

- (i)  $G$  je trianguliran
- (ii)  $G$  ima savršenu shemu. Dodatno, svaki jednostavni vrh može biti početak savršene sheme.
- (iii) Svaki minimalni separator vrhova inducira potpuni podgraf od  $G$ .

**Dokaz.**  $(iii) \Rightarrow (i)$  Neka  $[a, x, b, y_1, \dots, y_k, a]$ , gdje  $k \geq 1$ , jednostavni ciklus u  $G$ . Bilo koji minimalni separator  $a$ - $b$  mora sadržavati vrhove  $x$  i  $y_i$ , za neki  $i$  takav da  $xy_i \in E$ , što će predstavljati kabel ciklusa.



### 3.1. Triangulirani grafovi

$(i) \Rightarrow (iii)$  Pretpostavimo da  $S$  minimalni separator  $a$ - $b$  s odgovarajućim komponentama  $G_A$  i  $G_B$  koje sadrže vrhove  $a$  i  $b$  redom. Zbog minimalnosti od  $S$ , svaki  $x$  iz  $S$  je susjedan nekom vrhu iz  $A = V(G_A)$  i nekom vrhu iz  $B = V(G_B)$ , pa za bilo koja dva vrha  $x$  i  $y$  iz  $S$  postoje putevi  $xa_1 \dots a_r y$  i  $yb_1 \dots b_t x$ , gdje  $a_i \in A$  i  $b_i \in B$ , i neka su putevi odabrani tako da budu minimalne duljine. Slijedi da je  $xa_1 \dots a_r y b_1 \dots b_t x$  jednostavni ciklus duljine barem 4 što povlači da mora imati kabele. Ali, pošto po definiciji separatora vrhova vrijedi  $a_i b_j \notin E$  i pošto po minimalnosti odabranih putova  $a_i a_j \notin E$  i  $b_i b_j \notin E$ , jedini mogući kabel je  $xy$  iz  $E$ , pa smo došli do kontradikcije.

Kako bismo mogli nastaviti s dokazom, trebamo prvo pokazati da vrijedi Diracova lema.

**Lema 3.12 (Dirac)** *Svaki triangulirani graf  $G = (V, E)$  ima jednostavni vrh. Štoviše, ako  $G$  nije klika, onda ima dva nesusjedna jednostavna vrha.*

**Dokaz.** Ako je  $G$  potpun graf, lema je trivijalna (po definiciji jednostavnog vrha).

Pretpostavimo da  $G$  ima dva nesusjedna vrha  $a$  i  $b$ , te da lema vrijedi za sve grafove s manje vrhova od  $G$ . Neka je  $S$  minimalni separator  $a$ - $b$ , te neka su  $G_A$  i  $G_B$  komponente povezanosti grafa  $G_{V-S}$  koje redom sadrže  $a$  i  $b$ . Promotrimo komponentu  $A$ . Indukcijom, ili  $G_{A+S}$  ima dva nesusjedna vrha od kojih jedan mora biti u komponenti koja sadrži  $a$  (pošto  $S$  inducira potpuni podgraf po Teoremu 3.11) ili  $G_{A+S}$  je sam potpun, te je bilo koji vrh iz komponente koja sadrži  $a$  jednostavan u  $G_{A+S}$ .

Pošto  $Adj(A) \subseteq A + S$ , onda je jednostavni vrh od  $G_{A+S}$  u  $A$  je jednostavan u cijelom  $G$ . Analogno za  $B$ .

■

### 3.1. Triangulirani grafovi

Diracova lema nam daje mogućnost odabira barem dva vrha za svaku poziciju prilikom konstruiranja sheme. Sada možemo odabrati neki vrh  $v_n$  kojeg ćemo izbjegavati kako bismo ga mogli staviti na zadnje mjesto tj.  $n$ -to mjesto. Potom biramo  $v_{n-1}$  koji je susjedan sa  $v_n$  kojeg ćemo sačuvati za  $(n-1)$ -vo mjesto u shemi. Susjeda biramo kako bismo očuvali definiciju savršene sheme. Potom biramo susjeda  $v_{n-2}$  od  $v_{n-1}$  za  $(n-2)$ -o mjesto u shemi itd. Takvim procesom se shema kreira unatrag.

Sada nastavimo s dokazom Teorema 3.11.

$(i) \Rightarrow (ii)$  Po Diracovoj lemi, iz trijangularnosti od  $G$  slijedi postojanje jednostavnog vrha. Označimo taj vrh sa  $v_1$  i dodamo ga na kraj naše savršene sheme koja je za sada prazna. Pošto je  $G_{V-v}$  trianguliran i manji od  $G$ , onda je i  $G_{V-v}$  trianguliran. Tada opet primijenimo Diracovu lemu i odabremo jednostavni vrh  $w$  iz  $G_{V-v}$  i stavljamo ga iza  $v$  u shemi. Nastavljanjem postupka dobivamo savršenu shemu.

$(ii) \Rightarrow (i)$  Neka  $C$  jednostavni ciklus od  $G$ , te neka  $v$  vrh iz  $C$  s najmanjim indeksom u savršenoj shemi. Pošto  $|Adj(v) \cap C| \geq 2$ , onda nadolazeća jednostavnost od  $v$  garantira postojanje kabela u  $C$ .

■

Matematičari Rose, Tarjan i Leuker su predstavili algoritam, temeljen na Diracovoj lemi, za prepoznavanje triangularnih grafova koji se može izvesti u linearnom vremenu. Algoritam koristi leksikografski BFS. BFS tj. *breadth-first search* je način pretrage grafa takav da se prvo pretraži odabrani vrh, potom se pretražuju svi vrhovi s prve razine tj. svi vrhovi susjedni originalnom vrhu. Nakon toga prelazimo na drugu razinu tj. vrhove koji su susjedni

### 3.1. Triangulirani grafovi

s vrhovima s prve razine (koje još nisu posjećeni). Postupak se nastavlja dok svi vrhovi nisu pretraženi. Općenito, BFS pretražuje jednu po jednu razinu vrhova gdje je razina  $r$  skup svih vrhova susjednih vrhovima iz razine  $r - 1$ . Leksikografski BFS koristi leksikografski uređaj za biranje vrhova. Primjerice,  $[1, 2, 3, 4] < [1, 3, 5]$ .

Idući algoritam, koji koristi leksikografski BFS, će vraćati poredak vrhova  $P$  čiji će inverz tvoriti savršenu eliminacijsku shemu. Naime,  $P(i) = v$  govori da se vrh  $v$  nalazi na  $i$ -toj poziciji u savršenoj eliminacijskoj shemi tj. inverz  $P^{-1}$  vraća poziciju danog vrha u savršenoj shemi.

---

**Algorithm 3** Leksikografski BFS

---

**Input:** Liste susjeda neusmjerenog grafa  $G$

**Output:** Uređenje  $P$  vrhova iz  $G$

```
1: Incijaliziramo vrijednosti od  $P$  na  $null$  (označimo svaki vrh s  $null$ )
2: Incijaliziramo vrijednosti od  $L$  na  $\{\}$ 
3: for  $i \leftarrow n : 1$  do
4:   Odaberimo vrh  $v$  označen s  $null$  čija je vrijednost  $L$  leksikografski
   najveća.
5:    $P(i) = v$  //Numeriranje vrha  $v$  s indeksom  $i$ 
6:   for svaki nenumerirani vrh  $w$  iz  $Adj(v)$  do
7:     Dodaj  $i$  u  $L[w]$  //Dodavanje  $i$  u labelu od  $w$ 
8:   end for
9: end for
10: Return  $P$ ;
```

---

Za svaki  $i$ , označimo s  $L_i(x)$  labelu od  $x$  kad je  $i$ -ti vrh numeriran. Primjerice,  $L_{n-1}(x) = \{n\}$  ako i samo ako  $x \in Adj(P(n))$ , gdje  $P$  numeriranje vrhova iz algoritma. Labela od  $x$  je skup poredan leksikografski unazad. Vrh

### 3.1. Triangulirani grafovi

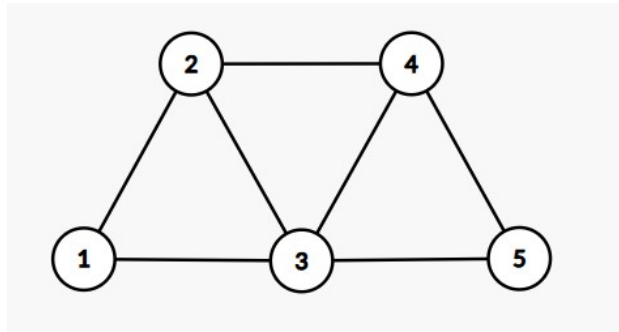
$x$  s leksikografski najvećom labelom je kandidat za jednostavni vrh u trenutnom koraku. Vrijede sljedeća svojstva:

$$(L1) L_i(x) \leq L_j(x), j \leq i$$

$$(L2) L_i(x) \leq L_i(y) \Rightarrow L_j(x) \leq L_j(y), j \leq i$$

(L3) Ako  $P^{-1}(a) < P^{-1}(b) < P^{-1}(c)$  i  $c \in Adj(a) - Adj(b)$ , onda postoji vrh  $d \in Adj(b) - Adj(a)$  takav da  $P^{-1}(c) < P^{-1}(d)$

**Primjer 3.13** Pronađimo savršenu eliminacijsku shemu za sljedeći graf.



vrh	1		2		3		4		5		6	
	L	P	L	P	L	P	L	P	L	P	L	P
1	{}	-	{5}	-	{5}	4	{5}	4	{5}	4	{5}	4
2	{}	-	{5}	-	{5,4}	-	{5,4}	3	{5,4}	3	{5,4}	3
3	{}	-	{}	5	{}	5	{}	5	{}	5	{}	5
4	{}	-	{5}	-	{5}	-	{5,3}	-	{5,3}	2	{5,3}	2
5	{}	-	{5}	-	{5}	-	{5}	-	{5,2}	-	{5,2}	1

Pošto su sve vrijednosti od  $L$  prazni skupovi, biramo proizvoljno vrh 3, stavljamo  $P(5) = 3$  i update-amo vrijednosti skupova od  $L$ . Pošto u idućem koraku nema najveće vrijednosti  $L$ , proizvoljno biramo vrh 1. U idućem koraku je leksikografski najveći  $L(3) = \{5, 4\}$ , pa biramo vrh 3 itd. dok ne dobijemo cijeli  $P$ .

**Napomena!** Poredak  $P$  se povećava što je vrh bliže korijenu pretraživanog stabla.

### 3.1. Triangulirani grafovi

Podsjetimo se da  $P^{-1}(v) = i$  označava poziciju  $i$  vrha  $v$  u eliminacijskoj shemi.

**Teorem 3.14** *Neusmjereni graf  $G = (V, E)$  je trianguliran ako i samo ako je poredak  $P$  dobiven Algoritmom 3 savršena shema.*

**Dokaz.**  $\Rightarrow$  Dokaz provodimo indukcijom. Za  $|V| = n = 1$ , dokaz trivijalan. Pretpostavimo da teorem vrijedi za sve grafove sa manje od  $n$  vrhova,  $n \in \mathbb{N}$ , te neka  $P$  poredak koji dobijemo kad provedemo algoritam na trianguliranom  $G$ . Indukcijom, dovoljno je pokazati da je  $v = P(1)$  jednostavan vrh od  $G$ .

Pretpostavimo suprotno tj. da  $v$  nije jednostavan. Odaberimo  $v_1, v_2 \in \text{Adj}(v)$  takve da  $v_1v_2$  nije brid u  $G$  i takve da je  $v_2$  najveći mogući obzirom na poziciju u  $P$ . Prisjetimo se da vrijedi gornja napomena.

Promotrimo sljedeći induktivni postupak. Pretpostavimo da za dane vrhove  $v_1, \dots, v_m$  vrijedi:  $\forall i, j > 0$

- (1)  $vv_i \in E$  ako i samo ako  $i \leq 2$ ,
- (2)  $v_iv_j \in E$  ako i samo ako  $|i - j| = 2$ ,
- (3)  $P^{-1}(v_1) < P^{-1}(v_2) < \dots < P^{-1}(v_m)$ ,
- (4)  $v_j$  je najveći vrh ovisno o  $P$  za koji vrijedi  $v_{j-2}v_j \in E$  i  $v_{j-3}v_j \notin E$ .

Označimo  $v_0 = v$  i  $v_{-1} = v_1$ . Već smo pokazali slučaj sa dva brida.

$v_{m-2}, v_{m-1}, v_m$  zadovoljavaju hipotezu svojstva (L3) za  $a, b, c$ , pa biramo  $v_{m+1}$  za najveći vrh veći od  $v_m$  koji je susjedan vrhu  $v_{m-1}$ , ali nije susjedan  $v_{m-2}$ . Ako je  $v_{m+1}$  susjedan  $v_{m-3}$ , onda svojstvo (L3) primijenjeno na  $v_{m-3}, v_{m-2}, v_{m+1}$  povlači postojanje vrha većeg od  $v_{m+1}$  koji je susjedan s  $v_{m-2}$ , ali ne i sa  $v_{m-3}$ . Pošto znamo da  $v_{m+1}$  veći od  $v_m$ , dolazimo do kontradikcije sa svojstvom (4). Stoga,  $v_{m+1}$  nije susjedan s  $v_{m-3}$ . Još iz svojstava (1), (2) i činjenice da svaki ciklus sa 4 ili više vrhova ima kabel, slijedi  $v_iv_{m+1} \notin E$ ,  $\forall i = 0, \dots, m-4, m$ .

### 3.1. Triangulirani grafovi

Opisani induktivni postupak se može provoditi beskonačno što je kontradikcija s konačnošću grafa. Stoga  $v$  mora biti jednostavan.

◁ Iz Teorema 3.11.

■

Ovime je dokazana ispravnost prethodnog algoritma. Pokažimo još i moguću složenost leksikografskog BFS.

U koraku 1, inicijalizacija svakog vrha traje  $O(n)$ , gdje je  $n$  broj vrhova. Koraci od 2 do 5 (for petlja) su složenosti  $O(n + m)$ , gdje je  $m$  broj bridova, jer moramo proći bar kroz svaki vrh i brid u postupku.

Promotrimo konkretnu implementaciju linija 5 i 6, te uvedimo potrebne varijable.

Neka je  $Q$  red skupova  $S_l = \{v \in V : \text{labela}(v) = l \text{ i } P^{-1}(v) \text{ nije definiran}\}$  poredanih leksikografski od najmanjeg prema najvećem.

Svaki  $S_l$  predstavimo kao duplo-vezanu listu tj. listu gdje svaki element i njegov susjed obostrano pokazuju jedno na drugo. Originalno se u  $Q$  nalazi samo  $V$  tj.  $S_\emptyset = V$ . Ovo znači da svi vrhovi za skup labela imaju prazan skup.

Uvedimo i funkciju *FLAG* koju ćemo inicijalizirati tako da svakom skupu iz  $Q$  pridružimo 0. *FLAG* se koristi kako bismo znali jesmo li u stanju update-anja ili brisanja.

*SET*( $w$ ) je referentna funkcija koja pokazuje na  $S_i$  u kojem je vrh  $w$ . Prilikom stvaranja novog  $S_{i+1}$ , *SET*( $w$ ) će prestati pokazivati  $S_i$  i početi pokazivati na  $S_{i+1}$ . Lista *LIST* je originalno prazna i ona prati stanja skupova vrhova kroz algoritam.  $P$  i  $P^{-1}$  su predstavljeni nizovima.

Vrh  $v$  biramo iz zadnjeg skupa iz  $Q$  (leksikografski maksimum) i brišemo ga iz *SET*( $v$ ), te kreiramo novi  $S_i$  za svaki stari  $S_l$  koji je sadržavao nenumerirani vrh  $w \in \text{Adj}(v)$ .

### 3.1. Triangulirani grafovi

---

**Algorithm 4** Implementacija linija 5 i 6 Algoritma 3

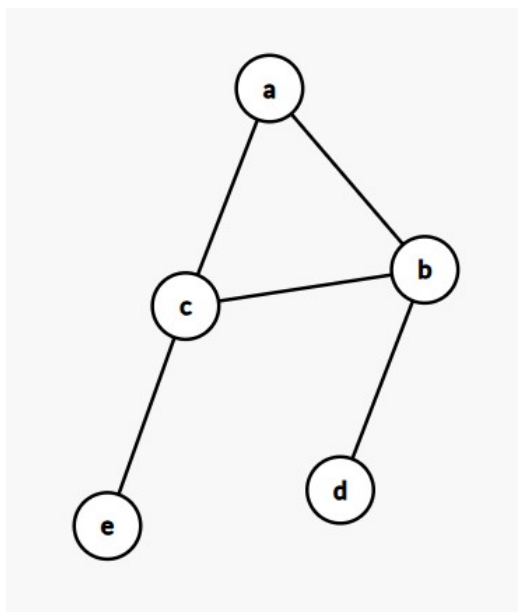
---

```
1: for svaku nenumerirani  $w \in Adj(v)$  do
2:   if  $FLAG(SET(w)) == 0$  then
3:     Kreirajmo novi skup  $S$  i ubacimo ga u  $Q$  odmah nakon  $SET(w)$ ;
4:      $FLAG(SET(w)) = 1$ ;
5:      $FLAG(S) = 0$ ;
6:     (Stavljamo pokazivač od  $SET(w)$  na  $LIST$ )
7:   end if
8:   Postavimo  $S$  odmah nakon  $SET(w)$  u  $Q$ 
9:   Izbrisimo  $w$  iz  $SET(w)$ 
10:  Dodajmo  $w$  u  $S$ 
11:   $SET(w) \rightarrow S$ ;
12: end for
13: for svaki skup  $T$  u  $LIST$  do
14:    $FLAG(T) = 0$ ;
15:   if  $T == \emptyset$  then
16:     Izbrisi  $T$  iz  $Q$ ;
17:   end if
18: end for
```

---

### 3.1. Triangulirani grafovi

Složenost implementacije je  $|Adj(v)|$ , pa je finalna složenost  $O(|Adj(v)|) + O(n) + O(n + m) = O(n + m)$ .



Slika 3.4: Graf s 5 vrhova

**Primjer 3.15** *Prikažimo stanja od  $Q$  za vrijeme provedbe leksikografskog BFS na grafu 3.4.*

$Q = [S_\emptyset] = \{a, b, c, d, e\}$  izbacimo  $e$

$Q = [\{a, b, d\}, \{c\}]$  (primijetimo  $\{a, b, d\} < \{c\}$ ), izbacimo  $c$

$Q = [\{a, b\}, \{d\}]$  izbacimo  $d$

$Q = [\{a\}, \{b\}]$  izbacimo  $b$

$Q = \emptyset$

Prikažimo i algoritam za testiranje savršene sheme potreban kako bismo mogli s Leksikografskim BFS prepoznavati triangulirane grafove.



### 3.1. Triangulirani grafovi

---

**Algorithm 5** Test - savršena shema

---

**Input:** Lista susjeda od  $G$  i poredak  $P$  od  $V$ .

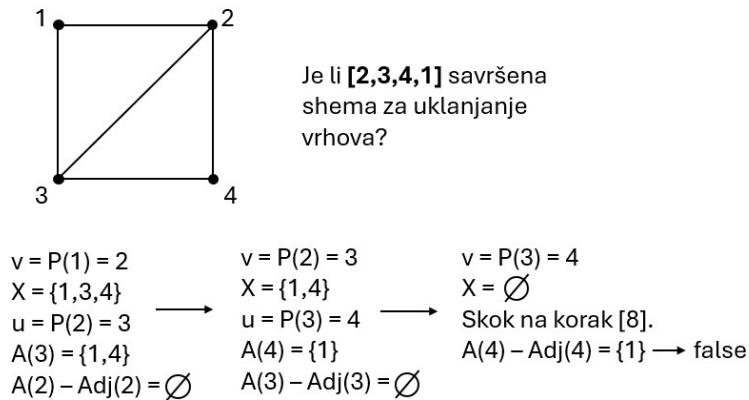
**Output:** true ako  $P$  savršena shema, inače false

```
1:  $A(v) = \emptyset \forall v$ 
2: for  $i = 1 : (n-1)$  do
3:    $v = P(i)$ ;
4:    $X = \{x \in Adj(v) : P^{-1}(v) < P^{-1}(x)\}$ ; //skup vrhova koji se pojav-
   ljuju nakon  $v$  u  $P$ 
5:   Ako  $X == \emptyset$ , idi na [8]
6:    $u = P(\min P^{-1}(x) : x \in X)$ ;
7:   Dodaj  $X - \{u\}$  u  $A(u)$ ;
8:   if  $A(v) - Adj(v) \neq \emptyset$  then
9:     return false
10:  end if
11:  return true
12: end for
```

---

Algoritam za svaki vrh provjerava tvore li preostali vrhovi kliku. U slučaju da za neki vrh preostali podgraf ne tvori kliku, vraća se *false*. Pozivom algoritma, lista  $A(u)$  skuplja sve vrhove za koje se kasnije provjerava jesu li susjedni s  $u$  tj. u  $A(u)$  spremamo vrhove s kojima  $u$  mora biti povezan kako bi  $P$  bila savršena shema. Provjera se vrši tek kad  $u = P(i)$  kako u  $P^{-1}(v)$ -toj iteraciji ne bi došlo do pretrage  $Adj(u)$ .

### 3.1. Triangulirani grafovi



Slika 3.5: Postupak algoritma za testiranje savršene sheme. Za dani graf,  $[2, 3, 4, 1]$  nije savršena shema.

Za  $P$  i  $P^{-1}$  koristimo nizove, a  $Adj(v)$  i  $A(v)$  spremamo u liste. Složenost koraka od [4 do 7] se može izvesti istovremeno u jednom prolasku kroz  $Adj(v)$ , a skok u koraku [5] će se izvesti točno  $j - 1$  puta, gdje je  $j$  broj povezanih komponenti u  $G$ . Korak [8] koji traži vrh u  $A(v)$  koji nije susjedan  $v$  se može izvesti sa složnošću  $O(|Adj(v)| + |A(v)|)$  ako koristimo niz  $TEST$  duljine  $n$  koji inicializiran nule na sljedeći način:

### 3.1. Triangulirani grafovi

---

```
1: for  $w \in Adj(v)$  do
2:    $TEST(w) = 1$ ;
3: end for
4: for  $w \in A(v)$  do
5:   if  $TEST(w) == 0$  then
6:     return "nije prazno";
7:   end if
8: end for
9: for  $w \in Adj(v)$  do
10:   $TEST(w) = 0$ ;
11: end for
12: return "prazno";
```

---

Ukupna složenost algoritma za testiranje savršene sheme je proporcionalna

$$|V| + \sum_{v \in V} |Adj(v)| + \sum_{u \in V} |A(u)|$$

, gdje  $A(u)$  finalna vrijednost.

Pošto se  $Adj(v)$  pojavljuje kao dio najviše jednog  $A(u)$ , možemo zamijeniti zadnja dva sumanda sa  $O(|E|)$ . Finalna složenost je tada proporcionalna  $|V| + |E|$ .

**Teorem 3.16** *Algoritam 5 korektno testira je li dan poredak vrhova  $P$  savršena eliminacijska shema. Složenost algoritma je  $|V| + |E|$ .*

**Dokaz.** Dokažimo da je algoritam korektan. Algoritam vraća false u  $P^{-1}(u)$  - toj iteraciji ako i samo ako postoje vrhovi  $u$ ,  $v$  i  $w$ , gdje je  $u$  definiran kao u liniji [4] za vrijeme  $P^{-1}(v)$  - te iteracije, te su  $u$  i  $w$  susjedi od  $v$ , ali  $u$  i  $w$  sami nisu susjedni.

### 3.2. Graf usporedivosti i TRO

Pretpostavimo da  $P$  nije savršena shema, te da algoritam vraća true. Neka  $v$  vrh s najvećim  $P^{-1}(v)$  takvim da  $X = \{x \in Adj(v) : P^{-1}(v) < P^{-1}(x)\}$  nije potpun. Neka  $u$  definiran u liniji [6] u  $P^{-1}(v)$ -toj iteraciji, nakon čega  $X - \{u\}$  u sljedećem koraku dodajemo u  $A(u)$ . Pošto u  $P^{-1}(v)$  - toj iteraciji preskačemo korak [9], svaki  $x \in X - \{u\}$  je susjedan  $u$  i svaki par  $x, y \in X - \{u\}$  je susjedan.

Činjenica da svaki par  $x, y \in X - \{u\}$  je susjedan, slijedi iz maksimalnosti od  $P^{-1}(v)$ , pa je  $X$  potpun, što je kontradikcija. Znači da mora algoritam vraćati false kad nije savršena shema. ■

## 3.2 Graf usporedivosti i TRO

Još jedan poznati primjer savršenih grafova su grafovi usporedivosti, koji će biti definirani u Definiciji 3.17. Baš kao i triangulirani grafovi, primjena grafova usporedivosti dopušta rješavanje problema koji su u općem slučaju (kad graf nije graf usporedivosti) NP-teški, te omogućuju optimiziranje brojnih algoritma. Kao i kod trinaguliranih grafova, grafovi usporedivosti primjenjivi su u mnogim područjima kao što su stvaranje rasporeda, bojanje grafova, biologija, društve mreže... U poglavlju ćemo konstruirati tzv. TRO (Transitive Orinetation) algoritam, te ćemo uvesti matematičku pozadinu potrebnu za definiranje tog algoritma. Predstaviti ćemo i pojam dekompozicije grafa koji će nam biti potreban u ovom postupku i uvesti algoritam za samo provođenje dekompozicije koji je jako koristan u računalstvu, ali i sam za sebe. Primjerice, dekompozicija grafova je korisna u paralelnom programiranju gdje različite komponente grafa možemo rasporediti u različite procese kako bi se mogli istovremeno izvršavali.

Za usmjereni graf  $G$  s orijentiranim skupom bridova  $F$ , sa  $F^{-1}$  ćemo

### 3.2. Graf usporedivosti i TRO

označiti skup orijentiranih bridova s orijentacijom suprotnom od orijentacije bridova u  $F$ . Označimo još sa  $A + B$  uniju disjunktnih skupova  $A$  i  $B$ .

**Definicija 3.17** Neusmjereni graf  $G = (V, E)$  nazivamo **grafom usporedivosti** ako postoji orijentacija  $\vec{G} = (V, F)$  takva da vrijede svojstva:

$$(S1) F \cap F^{-1} = \emptyset,$$

$$(S2) F + F^{-1} = E,$$

$$(S3) F^2 \subseteq F,$$

gdje  $F^2 = \{ac : ab, bc \in F \text{ za neki vrh } b\}$ .  $F$  se naziva **tranzitivna orijentacija** od  $G$ .

**Primjer 3.18** Promotrimo graf s bridovima  $\{\{a, b\}, \{a, c\}\}$ . Promotrimo orijentaciju bridova  $F = \{(a, b), (a, c)\}$ . Nema dvostrukih orijentacija, pa vrijedi (S1), te oba brida imaju orijentaciju, pa vrijedi (S2). Uočimo da je  $F^2 = \emptyset \subseteq F$ , pa vrijedi (S3). Postoji usmjerena orijentacija danih bridova koja zadovoljava svojstva (S1)-(S3), pa je graf s danim bridovima graf usporedivosti. Uočimo da smo mogli odabrati i orijentaciju za koju ne bi vrijedila tranzitivnost npr.  $\{(b, a), (a, c)\}$ .

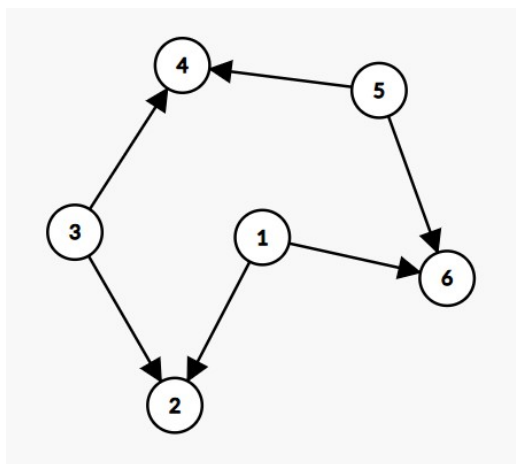
Uvedimo sad još neke pojmove koji će nam biti potrebni za definirati algoritme. Definirajmo binarnu relaciju  $R$  na bridovima neusmjerenog grafa  $G = (V, E)$  na sljedeći način:

$$abRcd \text{ ako i samo ako } \{ a = c \text{ i } bd \notin E \text{ ili } b = d \text{ i } ac \notin E \}$$

Kažemo da  $ab$  *direktno forsira*  $cd$  kada su bridovi  $ab$  i  $cd$  u relaciji  $R$  tj. ovako definirana relacija  $R$  se naziva **relacija forsiranja**.

Primijetimo da je relacija  $R$  irefleksivna tj. brid ne može forsirati samog sebe zato što ne vrijedi da, za neki brid  $xy$ ,  $xy$  forsira  $yx$ , a  $xy = yx$  zbog neusmjerenosti grafa.

### 3.2. Graf usporedivosti i TRO



Slika 3.6: Forsiranje grafa obzirom na orijentaciju brida (1,2).

Refleksivno i tranzitivno zatvorenje  $R^*$  od  $R$  je relacija ekvivalenije, te razdvaja  $E$  u klase koje ćemo nazivati *implikacijskim klasama* od  $G$ . Uočimo,  $ab$  i  $cd$  će biti u istoj implikacijskoj klasi ako i samo ako postoji niz bridova  $ab = a_0b_1Ra_1b_1R \dots a_kb_k = cd$ ,  $0 \leq k$ . Ovakav niz nazivamo  *$R$ -lanac* od  $ab$  do  $cd$ . Tada **ab forsira cd**.

Uočimo da vrijedi  $abRcd$  akko  $baRdc$ , te  $abR^*cd$  akko  $baR^*dc$ .

**Primjer 3.19** Promotrimo put  $G = (\{1, 2, 3, 4\}, \{\{1, 2\}, \{2, 3\}, \{3, 4\}\})$ . Postoji lanac  $12 R 32 R 34$ , pa kažemo da  $12$  forsira  $34$ .

Označimo sa  $\sigma(G)$  kolekciju implikacijskih klasa od  $G$ . Definiramo  $\hat{\sigma} = \{\hat{A} : A \in \sigma(G)\}$ , gdje  $\hat{A} = A \cup A^{-1}$  simetrično zatvorenje od  $A$ . Članovi od  $\hat{\sigma}$  se nazivaju *klasama boja*.

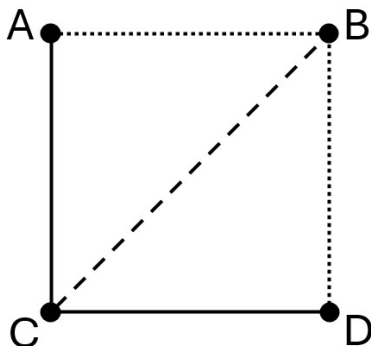
**Primjer 3.20** Graf na slici 3.3 ima šest implikacijskih klasa:

$$A_1 = \{BC\}, A_2 = \{AB, DB\}, A_3 = \{AC, DC\},$$

$$A_1^{-1} = \{CB\}, A_2^{-1} = \{BA, BD\}, A_3^{-1} = \{CA, CD\},$$

### 3.2. Graf usporedivosti i TRO

Tada  $\hat{\sigma} = \{\widehat{A}_1, \widehat{A}_2, \widehat{A}_3\}$ . Na slici 3.7 je prikazano bojanje obzirom na  $\hat{\sigma}$ .



Slika 3.7: Bojanje grafa obzirom na njegovu klasu boja  $\hat{\sigma}$ .

**Teorem 3.21** *Neka je  $A$  implikacijska klasa neusmjerenog grafa  $G$ . Ako  $G$  ima tranzitivnu orijentaciju  $F$ , onda je  $A \cap \widehat{A} = \emptyset$ , te vrijedi jedno od sljedećeg:*

- 1)  $F \cap \widehat{A} = A$
- 2)  $F \cap \widehat{A} = A^{-1}$

**Dokaz.** Za prije definiranu relaciju  $R$  vrijedi  $xyRx'y'$  i  $xy \in F$ , onda  $x'y' \in F$ . Ponavljanjem postupka dobivamo da vrijedi  $F \cap A = \emptyset$  ili da vrijedi  $A \subseteq F$ . Iz  $A \subseteq F + F^{-1}$  slijedi

$$F \cap A = \emptyset \Rightarrow A \subseteq F^{-1} \Rightarrow A^{-1} \subseteq F \Rightarrow F \cap \widehat{A} = A^{-1},$$

a iz  $F \cap F^{-1} = \emptyset$  slijedi

$$A \subseteq F \Rightarrow A^{-1} \subseteq F^{-1} \Rightarrow F \cap A^{-1} = \emptyset \Rightarrow F \cap \widehat{A} = A. \quad \blacksquare$$

Neka  $ab = a_0b_0Ra_1b_1R \dots Ra_kb_k = cd$ . Za svaki  $i = 1, \dots, k$  vrijedi  $a_{i-1}b_{i-1}Ra_ib_{i-1}Ra_ib_i$  jer zbrojeni srednji bridovi daju jedan od preostala dva. Tada vrijedi sljedeća lema.

**Lema 3.22** *Ako  $abR^*cd$ , onda postoji  $R$ -lanac od  $ab$  do  $cd$  oblika  $ab = a_0b_0Ra_1b_0Ra_1b_1Ra_2b_1R \dots Ra_kb_k = cd$ .*

### 3.2. Graf usporedivosti i TRO

Lanac iz prethodne leme nazivamo *kanonskim R-lancem*.

**Lema 3.23 (Lema o trokutu)** *Neka su  $A$ ,  $B$  i  $C$  implikacijske klase neusmjerenog grafa  $G = (V, E)$  takve da  $A \neq B$ ,  $A \neq C^{-1}$  sa bridovima  $ab \in C$ ,  $ac \in B$  i  $bc \in A$ .*

- (1) *Ako  $b'c' \in A$ , onda  $ab' \in C$  i  $ac' \in B$ .*
- (2) *Ako  $b'c' \in A$  i  $a'b' \in C$ , onda  $a'c' \in B$ .*
- (3) *Ni jedan brid u  $A$  nije incidentan sa vrhom  $a$ .*

**Dokaz.** Po prethodnoj lemi postoji kanonski  $R$ -lanac  $bc = b_0c_0Rb_1c_0Rb_1c_1R \dots Rb_kc_k = b'c'$ . Pokažimo prvo (1). Indukcijom po  $i$ , vrijedi sljedeće:

- $$ac_i \in B, b_{i+1}c_i \in A, ac_i \text{ nije u relaciji } R \text{ s } b_{i+1}c_i \Rightarrow ab_{i+1} \in E,$$
- $$b_{i+1}b_i \notin E \Rightarrow ab_{i+1}Rab_i, ab_i \in C,$$
- $$b_{i+1}a \in C^{-1}, b_{i+1}c_{i+1} \in A, b_{i+1}a \text{ nije u relaciji } R \text{ s } b_{i+1}c_{i+1} \Rightarrow ac_{i+1} \in E,$$
- $$c_{i+1}c_i \notin E \Rightarrow ac_{i+1}Rac_i, ac_i \in B.$$

Posebno, iz ovoga vrijedi  $ab' = ab_k \in C$ ,  $ac' = ac_k \in B$ , što dokazuje (1).

Za dokazati svojstvo (2), pretpostavimo  $b'c' \in A$ ,  $a'b' \in C$ . Iz (1) slijedi  $ac' \in B$ . Promotrimo lanac  $ab = a_0b_0Ra_1b_0Ra_1b_1R \dots Ra_l b_l = a'b'$ . Iz ovog lanaca dobivamo  $ac' = a_0c'Ra_1c'R \dots Ra_l c' = a'c'$ . Tada  $ac'R^*a'c'$  i  $a'c' \in B$  dokazuje (2).

- (3) direktno slijedi iz (1). ■

**Teorem 3.24** *Neka je  $A$  implikacijska klasa od neusmjerenog grafa  $G = (V, E)$ . Vrijedi točno jedno od sljedećeg:*

- 1)  $A = \widehat{A} = A^{-1}$
- 2)  $A \cap A^{-1} = \emptyset$ ,  $A$  i  $A^{-1}$  su tranzitivni, te su jedina tranzitivna orijentacija od  $\widehat{A}$ .



### 3.2. Graf usporedivosti i TRO

**Dokaz.** 1) Pretpostavimo da  $A \cap A^{-1} \neq \emptyset$ , te da je  $xy \in A \cap A^{-1}$ . Pretpostavimo još da vrijedi  $xyR^*yx$ . Za bilo koji  $ab \in A$  vrijedi  $abR^*xy$  i  $baR^*yx$ . Iz ekvivalencije relacije  $R^*$  slijedi da  $abR^*ba$  i  $ba \in A$  što povlači  $A = \widehat{A}$ .

2) Pokažimo da vrijedi tranzitivnost. Pretpostavimo  $A \cap A^{-1} = \emptyset$ ,  $xy, yz \in A$ . Pretpostavimo da  $xz \notin E$ . Tada  $xyRzy$ , pa iz toga slijedi da  $zy \in A$ . Sada  $yz \in A^{-1}$ , što je kontradikcija. Tada  $xz \in E$ .

Pretpostavimo da je  $B$  implikacijska klasa od  $G$  koja sadrži  $xz$ , te pretpostavimo da je  $A$  različit od  $B$ . Iz  $A \neq A^{-1}$  i  $xy \in A$  lema o trokutu povlači da  $xy \in B$  što je također kontradikcija. Sada je  $xz \in A$  i  $A$  je tranzitivan. Ovo povlači i tranzitivnost od  $A^{-1}$ .

Pošto je  $A$  implikacijska klasa od  $\widehat{A}$ , iz Teorema 3.21 slijedi da su  $A$  i  $A^{-1}$  jedine tranzitivne orijentacije od  $\widehat{A}$ . ■

Definirajmo simpleks kako bi mogli uvesti pojam dekompozicije grafa.

**Definicija 3.25** Neka  $G = (V, E)$  neusmjereni graf. Potpuni podgraf  $(V(S), S)$  na  $r$  vrhova zovemo **simpleksom ranga  $r$**  ako je svaki brid  $xy$  iz  $S$  sadržan u različitoj klasi boja u  $G$ .

Osim simpleks, postoji još i multipleks. *Multipleks* dobiven iz simpleksa  $S$  ranga  $r$  definiramo kao neusmjereni podgraf  $(V(M), M)$ , gdje  $M = \{xy \in E : xyR^*ab, \text{ za neki } ab \in S\}$ . Alternativno,  $M$  je unija svih klasa boja od  $G$  koje zadovoljavaju da u presjeku sa  $S$  nisu prazne. Za multipleks kažemo da je maksimalan ako nije pravilno sadržan ni u jednom većem multipleksu.

**Primjer 3.26** Na grafu sa Slike 3.7, podgraf sastavljen od vrhova  $a - b - d$  čini simpleks ranga 3.

**Definicija 3.27** Neka  $G = (V, E)$  neusmjereni graf. Particiju skupa bridova  $E = \widehat{B}_1 + \dots + \widehat{B}_k$  zovemo  **$G$ -dekompozicijom** od  $E$  ako je  $B_i$  implikacijska klasa od  $\widehat{B}_1 + \dots + \widehat{B}_k, \forall i = 1, \dots, k$ .

### 3.2. Graf usporedivosti i TRO

Niz bridova  $[x_1y_1, \dots, x_ky_k]$  se naziva **dekompozicijska shema** od  $G$  ako postoji  $G$ -dekompozicija  $E = \widehat{B}_1 + \dots + \widehat{B}_k$  za koju vrijedi da  $x_iy_i \in B_i$ ,  $\forall i = 1, \dots, k$ .

Alternativno, graf ima dekompoziciju ako se može prikazati kao netrivialna kompozicija induciranih si podgrafova. Dekompozicijsku shemu ćemo nazivati još i samo shemom.

Za danu  $G$ -dekompoziciju postoji više odgovarajućih dekompozicijskih shema, ali obrat ne vrijedi. Za danu dekompozicijsku shemu postoji samo jedna odgovarajuća  $G$ -dekompozicija.

Prikažimo sad algoritam koji provodi dekompoziciju danog neusmjerenog grafa  $G$ .

---

**Algorithm 6** Algoritam dekompozicije

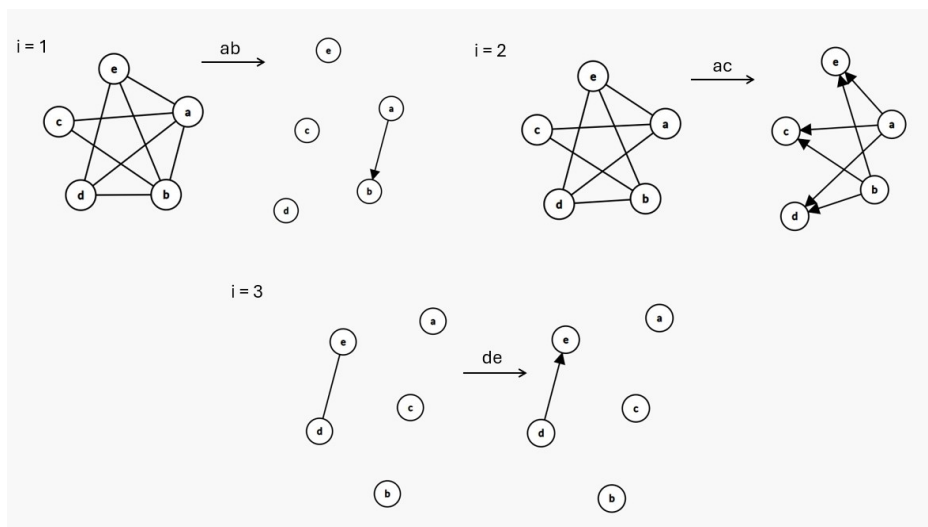
**Input (i preduvjeti):** Neka  $G = (V, E)$  neusmjereni graf, te uzmimo  $i=1$  i  $E_1 = E$ .

**Result:** Dekompozicijska shemu  $[x_1y_1, \dots, x_ky_k]$  i odgovarajuća  $G$ -dekompozicija  $\widehat{B}_1 + \dots + \widehat{B}_k$ .

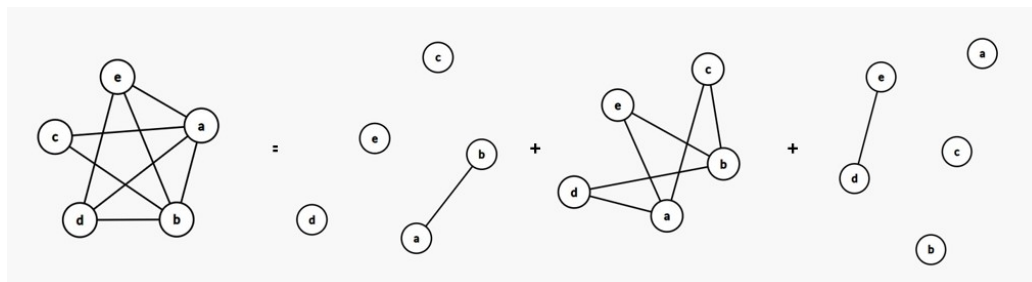
- 1: Odaberimo proizvoljni brid  $e_i = x_iy_i \in E_i$
  - 2: Numerirajmo implikacijske klase  $B_i$  od  $E_i$  koje sadrže  $x_iy_i$
  - 3:  $E_{i+1} := E_i - \widehat{B}_i$
  - 4: **if**  $E_{i+1} = \emptyset$  **then**
  - 5:      $k = i$ ; //broj boja
  - 6:     STOP
  - 7: **else**
  - 8:      $i += 1$ ;
  - 9:     Povratak na korak 1.
  - 10: **end if**
-

### 3.2. Graf usporedivosti i TRO

Primijetimo da možemo uzeti  $y_i x_i$  umjesto  $x_i y_i$ , ali onda u  $G$ -dekompoziciji  $B_i$  mijenjamo s  $B_i^{-1}$ .



Slika 3.8: Primjer provedbe dekompozicijskog algoritma. Grafovi lijevo predstavljaju  $E_i$ , na liniji je odabrani  $x_i y_i$ , a desno su  $B_i$



Slika 3.9:  $E = \widehat{B}_1 + \widehat{B}_2 + \widehat{B}_3$ , sa Slike 3.8

### 3.2. Graf usporedivosti i TRO

**Teorem 3.28** *Neka  $A$  implikacijska klasa od neusmjerenog grafa  $G=(V,E)$ , te neka  $C$  implikacijska klasa od  $E - \hat{A}$ . Vrijedi jedno od sljedećeg:*

- (i)  *$C$  je implikacijska klasa od  $E$ , te je  $A$  je implikacijska klasa od  $E - \hat{C}$ .*
- (ii)  *$C = B + D$ , gdje su  $B$  i  $D$  implikacijske klase od  $E$ , a  $\hat{A} + \hat{B} + \hat{D}$  je multipleks od  $E$  ranga 2.*

**Dokaz.** Uklanjanje  $\hat{A}$  iz  $E$  može uzrokovati stapanje nekih implikacijskih klasa od  $E$ , pa označimo sa  $C$  uniju  $k$  implikacijskih klasa od  $E$ .

Pretpostavimo  $k \geq 2$ . Tada postoji trokut na vrhovima  $a, b, d$  takav da  $bd \in \hat{A}$ , te su ili  $ad \in B$  i  $ab \in D$  ili  $da \in B$  i  $ba \in D$ , gdje su  $B$  i  $D$  različite implikacijske klase od  $E$  koje su sadržane u  $C$ . Bez gubitka općenitosti (jer postupak analogan za  $C^{-1}$ ) pretpostavimo da  $ad \in B$  i  $ab \in D$ . Pretpostavimo da  $B = D^{-1}$ . Tada  $ba, ad \in B$ , ali  $bc \notin B$ , pa po Teoremu 3.24 je  $B = \hat{B} = B^{-1}$ . Ovo povlači  $B = D$  što je kontradikcija, pa  $\hat{B} \cap \hat{D} = \emptyset$  i trokut  $\{a, b, d\}$  je obojan u 3 boje što znači da je  $\hat{A} + \hat{B} + \hat{D}$  multipleks ranga 2.

Bilo koji  $R$ -lanac u  $E - \hat{A}$  koji sadrži bridove iz  $\hat{B}$  i  $\hat{D}$  ne smije sadržavati bridove iz drugih implikacijskih klasa jer svi trokuti iz  $E$  s jednim bridom u  $\hat{A}$  i drugim u  $\hat{B}$ , moraju imati treći u  $\hat{D}$ . Isto vrijedi za sve kombinacije pripadnosti bridova. Tada bi  $R$ -lanac bio izomorfan kao simpleks sa  $\{a, b, d\}$  trokutom, pa je  $k = 2$  i  $C = B + D$ .

Preostaje pokazati da je za  $k = 1$ ,  $A$  implikacijska klasa od  $E - \hat{C}$ . Iz prethodno pokazanoga, ako  $A$  nije implikacijska klasa od  $E - \hat{C}$ , onda je  $C + \hat{A} + \hat{A}_1$  multipleks ranga 2 u  $E$ , gdje je  $A_1$  neka treća implikacijska klasa od  $E$ . Tada  $C$  ne može sam biti implikacijska klasa od  $E - \hat{A}$ , pa  $k \neq 1$  što je kontradikcija.

■

### 3.2. Graf usporedivosti i TRO

Napokon, iskažimo i dokažimo TRO teorem koji će nam biti jako važan za konstrukciju algoritma.

**Teorem 3.29 (TRO teorem)** *Neka  $G = (V, E)$  neusmjereni graf s  $G$ -dekompozicijom  $E = \widehat{B}_1 + \dots + \widehat{B}_k$ . Sljedeće tvrdnje su ekvivalentne:*

- (i)  $G$  je graf usporedivosti,
- (ii)  $A \cap A^{-1} = \emptyset$ , za svaku implikacijsku klasu  $A$  od  $E$ ,
- (iii)  $B_i \cap B_i^{-1} = \emptyset, i = 1, \dots, k$ ,
- (iv) svaki [”ciklus”] bridova  $v_1v_2, \dots, v_qv_1 \in E$  takvi da  $v_{q-1}v_1, v_qv_2, v_{i-1}v_{i+1} \notin E$  je parne duljine,  $i = 2, \dots, q - 1$ .

*Dodatno, uz ova svojstva,  $B_1 + \dots + B_k$  je tranzitivna orijentacija od  $E$ .*

**Dokaz.** (i)  $\Rightarrow$  (ii) Iz Teorema 3.21.

(ii)  $\Rightarrow$  (iii) Dokaz indukcijom. Pošto je  $B_1$  implikacijska klasa od  $E$ , onda vrijedi  $B_1 \cap B_1^{-1} = \emptyset$ . Za  $k=1$ , dokaz gotov. Pretpostavimo da implikacija vrijedi za sve  $G$ -dekompozicije grafa duljine  $< k$ . Tada, posebno, implikacija vrijedi i za  $E - \widehat{B}_1$ . Neka  $C$  implikacijska klasa od  $E - \widehat{B}_1$ . Po Teoremu 3.28, ili je  $C$  implikacijska klasa od  $E$ , pa vrijedi  $C \cap C^{-1} = \emptyset$ , ili je  $C = B + D$ , gdje su  $B$  i  $D$  implikacijske klase od  $E$  za koje vrijedi  $\widehat{B} \cap \widehat{D} = \emptyset$ . Ovo povlači :

$$C \cap C^{-1} = (B + D) \cap (B^{-1} + D^{-1}) = (B \cap B^{-1}) + (D \cap D^{-1}) = \emptyset$$

Indukcijom dobijemo  $B_i \cap B_i^{-1} = \emptyset, \forall i = 2, \dots, k$ .

(iii)  $\Rightarrow$  (i) Neka je  $E = \widehat{B}_1 + \dots + \widehat{B}_k$   $G$ -dekompozicija od  $E$ ,  $B_i \cap B_i^{-1} = \emptyset$ . Po Teoremu 3.24,  $B_1$  je tranzitivan. Za  $k=1$ , implikacija vrijedi. Pretpostavimo da implikacija vrijedi za sve  $G$ -dekompozicije grafa duljine  $< k$ . Po pretpostavci,  $F = B_2 + \dots + B_k$  je tranzitivna orijentacija od  $E - \widehat{B}_1$ . Preostaje pokazati da  $B_1 + F$  tranzitivna.

### 3.2. Graf usporedivosti i TRO

Neka  $xy, yz \in B_1 + F$ . Ako su oba vrha u  $B_1$  ili su oba u  $F$ , onda po tranzitivnosti  $B_1$  i  $F$ ,  $xz \in B_1 + F$ . Stoga, bez gubitka općenitosti, pretpostavimo  $xy \in B_1, yz \in F$ , što povlači  $xyR^*zy$  tako da  $xz \in E$ .

Pokažimo da  $xz$  mora biti u  $B_1 + F$ . Da nije, onda bi  $zx$  bio u  $B_1 + F$ . Ali tada  $zx \in B_1, xy \in B_1$  što povlači  $zy \in B_1$  što je kontradikcija, te  $zx \in F, yz \in F$ , pa  $yx \in F$  što povlači još jednu kontradikciju. Sada  $xz$  mora biti u  $B_1 + F$ . Analogno  $xy \in F, yz \in B_1$  povlači  $xz \in B_1 + F$ .

Vrijedi da je  $B_1 + \dots + B_k$  tranzitivna orijentacija od  $E$ .

(iv)  $\iff$  (i) Neka  $v_1v_2 \in A \cap A^{-1} \neq \emptyset$ . Po Lemi 3.22, postoji  $R$ -lanac oblika  $v_1v_2 R v_3v_2 R v_3v_4 R \dots R v_qv_{q-1} R v_qv_{q+1} = v_2v_1$ .

Pošto sve prve koordinate imaju neparan indeks,  $q$  je očito neparan. Također,  $v_1v_2v_3 \dots v_qv_1$  je ciklus, pa dobivamo kontradikciju.

Ako je  $E$  ciklus neparne duljine  $q$ , onda  $v_1v_2 R v_3v_2 R v_3v_4 R \dots R v_qv_{q-1} R v_qv_1 R v_2v_1$  predstavlja jedan  $R$ -lanac u  $E$ , što povlači  $A \cap A^{-1} \neq \emptyset$  za implikacijske klase koje sadrže  $v_1v_2$  što je kontradikcija.

■

Kombinacijom TRO teorema i dekompozicijskog algoritma napokon dobivamo TRO algoritam koji nam služi za prepoznavanje grafova usporedivosti i pridruživanja tim grafovima tranzitivne orijentacije.

### 3.2. Graf usporedivosti i TRO

---

**Algorithm 7** TRO algoritam

---

**Input:** Neka zadan neusmjereni graf  $G = (V, E)$ .

**Output:** Vraća se ili tranzitivna orijentacija  $F$  grafa  $G$  ili poruka da uneseni graf nije graf usporedivosti.

```
1:  $i = 1$ ;  
2:  $E_i = E$ ;  
3:  $F = \emptyset$ ;  
4: Proizvoljno biramo  $x_i y_i \in E_i$ ;  
5: Numeriramo implikacijske klase  $B_i$  od  $E_i$  koje sadrže  $x_i y_i$ ;  
6: if  $B_i \cap B_i^{-1} = \emptyset$  then  
7:   Dodaj  $B_i$  u  $F$ .  
8: else return  $G$  nije graf usporedivosti.  
9: end if  
10:  $E_{i+1} := E_i - \widehat{B}_i$ ;  
11: if  $E_{i+1} = \emptyset$  then  
12:    $k = i$ ; return  $F$ ;  
13: else  
14:    $i += 1$ ;  
15:   Povratak na korak [4].  
16: end if
```

---

Algoritam može vraćati različite  $F$  ovisno o odabranoj shemi, ali može se pokazati da će broj iteracija  $k$  uvijek biti isti bez obzira na odabranu shemu. Ta činjenica slijedi direktno iz Golumbicevog teorema kojeg ćemo dokazati nakon idućeg pomoćnog teorema.

**Teorem 3.30** *Neka  $[e_1, \dots, e_k]$  dekompozicijska shema grafa  $G$ , te neka  $\pi$  permutacija brojeva  $1, \dots, k$ . Tada  $[e_{\pi(1)}, \dots, e_{\pi(k)}]$  također dekompozicijska*

### 3.2. Graf usporedivosti i TRO

shema od  $G$ .

**Dokaz.** Dokaz provodimo za  $k \geq 2$  (jer za  $k=1$  se nema što dokazati). Neka  $\widehat{B}_1 + \dots + \widehat{B}_k$   $G$ -dekompozicija sheme iz iskaza teorema. Fiksirajmo  $i < k$ . Neka  $E_i = \widehat{B}_i + \dots + \widehat{B}_k$ ,  $C_i$  implikacijska klasa od  $E_i$  koja sadrži  $e_{i-1}$ , te  $C_{i+1}$  implikacijska klasa od  $E_i - \widehat{C}_i$  koja sadrži  $e_i$ . Po Teoremu 3.28 ili  $B_{i+1} = C_i, B_i = C_{i+1}$  ili postoji implikacijska klasa  $A$  od  $E_i$  takva da  $\widehat{B}_{i+1} = \widehat{A} + \widehat{C}_i$  i  $\widehat{C}_{i+1} = \widehat{A} + \widehat{B}_i$ . Bilo koji od ovih rezultata povlači  $\widehat{B}_i + \widehat{B}_{i+1} = \widehat{C}_i + \widehat{C}_{i+1}$ . Tada je  $\widehat{B}_1 + \dots + \widehat{C}_i + \widehat{C}_{i+1} + \dots + \widehat{B}_k$   $G$ -dekompozicija od  $E$  sa shemom  $[e_1, \dots, e_{i+1}, e_i, \dots, e_k]$  ■

**Teorem 3.31 (Golumbic)** Neka  $G = (V, E)$  neusmjereni graf.

- (i) Svaka shema od  $G$  je jednake duljine.
- (ii) Svaka  $G$ -dekompozicija je jednake duljine.
- (iii) Ako su  $[e_1, \dots, e_k]$  i  $[f_1, \dots, f_k]$  sheme od  $G$ , onda za bilo koji  $e_i$  postoji  $f_j$  takav da  $[e_1, \dots, e_{i-1}, f_j, e_{i+1}, \dots, e_k]$  koji je također dekompozicijska shema od  $G$ .

**Dokaz.** Ako  $G$  ima implikacijsku klasu  $A$  za koju  $E = \widehat{A}$ , onda je bilo koja dekompozicijska shema duljine 1 i bilo koji brid može biti odabran kao dekompozicijska shema. Stoga pretpostavimo da teorem vrijedi za sve grafove koji imaju manje implikacijskih klasa od  $G$ . Neka tada  $[e_1, \dots, e_k]$  i  $[f_1, \dots, f_m]$ ,  $k, m \geq 2$ , dekompozicijske sheme od  $G$ . Odaberimo  $e_i$  takav da nije na prvoj poziciji (što smijemo po prošlom teoremu). Ako je  $E = \widehat{B}_1 + \dots + \widehat{B}_m$   $G$ -dekompozicijska koja je pridružena  $[f_1, \dots, f_m]$ , onda  $e_1 \in \widehat{B}_q$  za neki  $q$ . Tada  $[f_1, \dots, f_{q-1}, e_1, f_{q+1}, \dots, f_m]$  također dekompozicijska shema, a po Teoremu 3.30 to je i  $[e_1, f_1, \dots, f_{q-1}, f_{q+1}, \dots, f_m]$ .

Sada su i  $[e_2, \dots, e_i, \dots, e_k]$  i  $[f_1, \dots, f_{q-1}, f_{q+1}, \dots, f_m]$  dekompozicijske sheme od  $E - \widehat{C}$ , gdje je  $C$  implikacijska klasa od  $E$  koja sadrži  $e_1$ . Pošto



### 3.2. Graf usporedivosti i TRO

$E - \widehat{C}$  ima manje implikacijskih klasa nego  $E$ , duljine  $k-1$  i  $m-1$  su indukcijom jednake i postoji  $f_j$  koji može zamijeniti  $e_i$  u njegovoj dekompozicijskoj shemi.

Pošto sve odgovarajuće  $G$ -dekompozicije i dekompozicijske sheme imaju istu duljinu, možemo zaključiti da sve  $G$ -dekompozicije imaju jednaku duljinu. ■

Promotrimo sljedeću realizaciju prije spomenutog dekompozicijskog algoritma.

Neka  $G = (V = \{v_1, \dots, v_n\}, E)$  neusmjereni graf. U algoritmu koristimo funkciju  $KLASA$  definiranu na sljedeći način:

$$KLASA(i, j) = \begin{cases} 0 & , \quad v_i v_j \notin E \\ k & , \quad v_i v_j \text{ zadan u } B_k \\ -k & , \quad v_i v_j \text{ zadan u } B_k^{-1} \\ \text{undefined} & , \quad v_i v_j \text{ još uvijek nije zadan nikome} \end{cases} \quad (3.1)$$

Uvedimo i varijablu  $FLAG$  koja će biti 0 ako je dani graf graf usporedivosti, a 1 inače. U slučaju da  $FLAG = 0$ , onda tranzitivnu orijentaciju od  $G$  dobivamo tako da kombiniramo sve bridove s pozitivnom vrijednošću koje dobivamo iz  $KLASA$ .

### 3.2. Graf usporedivosti i TRO

---

**Algorithm 8** Implementacija dekompozicijskog algoritma

---

**Input:** Neka zadan neusmjereni graf  $G = (V = \{v_1, \dots, v_n\}, E)$ , te neka vrijedi da  $j \in Adj(i)$  ako i samo ako  $v_i v_j \in E$ .

**Output:**  $G$ -dekompozicija iz *KLASA* i rezultat varijable *FLAG*.

```
1:  $k = 0$ ;  
2:  $FLAG = 0$ ;  
3: for svaki brid  $v_i v_j \in E$  do  
4:   if  $KLASA[i, j] = \text{undefined}$  then  
5:      $k = k + 1$ ;  
6:      $KLASA[i, j], KLASA[j, i] = k, -k$  ;  
7:      $PRETRAŽIVANJE(i, j)$ ;  
8:   end if  
9: end for
```

---

Prethodni algoritam nastavlja s radom dok svi bridovi nisu posječeni. U  $k$ -toj iteraciji, neposječeni brid smještamo u  $B_k$ , te anžuriramo *KLASA*. Nakon smještanja brida u  $B_k$ , rekursivno se dodavaju u  $B_k$  bridovi koji su u relaciji  $R$ .

Varijabla *FLAG* se mijenja iz 0 u 1 čim nađemo  $B_k$  takav da  $B_k \cap B_k^{-1} \neq \emptyset$ . Po TRO teoremu za  $G$  u tom trenu znamo da nije graf usporedivosti.

$v_i v_j \in E$  ako i samo ako  $KLASA[i, j] == k$  ili je nedefinirano do  $k$ -tog koraka.

Prikažimo još i funkciju *PRETRAŽIVANJE* koja se koristi u TRO algoritmu.

### 3.2. Graf usporedivosti i TRO

---

**Algorithm 9** PRETRAŽIVANJE

---

**Input:**  $(i, j)$  - numeracije vrhova iz brida  $v_i v_j$

```
1: for svaki  $l \in Adj(i)$  takve da  $[l \notin Adj(j)$  ili  $|KLASA[j, l]| < k]$  do
2:   if  $KLASA[i, l] == \text{undefined}$  then
3:      $KLASA[i, l], KLASA[l, i] = k, -k$  ;
4:      $PRETRAŽIVANJE(i, l)$ ;
5:   else
6:     if  $KLASA[i, l] == -k$  then
7:        $KLASA[i, l], FLAG = k, 1$ ;
8:        $PRETRAŽIVANJE(i, l)$ ;
9:     end if
10:  end if
11: end for
12: for svaki  $l \in Adj(j)$  takve da  $[l \notin Adj(i)$  ili  $|KLASA[i, l]| < k]$  do
13:  if  $KLASA[l, j] == \text{undefined}$  then
14:     $KLASA[l, j], KLASA[l, j] = k, -k$  ;
15:     $PRETRAŽIVANJE(l, j)$ ;
16:  else
17:    if  $KLASA[l, j] == -k$  then
18:       $KLASA[l, j], FLAG = k, 1$ ;
19:       $PRETRAŽIVANJE(l, j)$ ;
20:    end if
21:  end if
22: end for
```

---

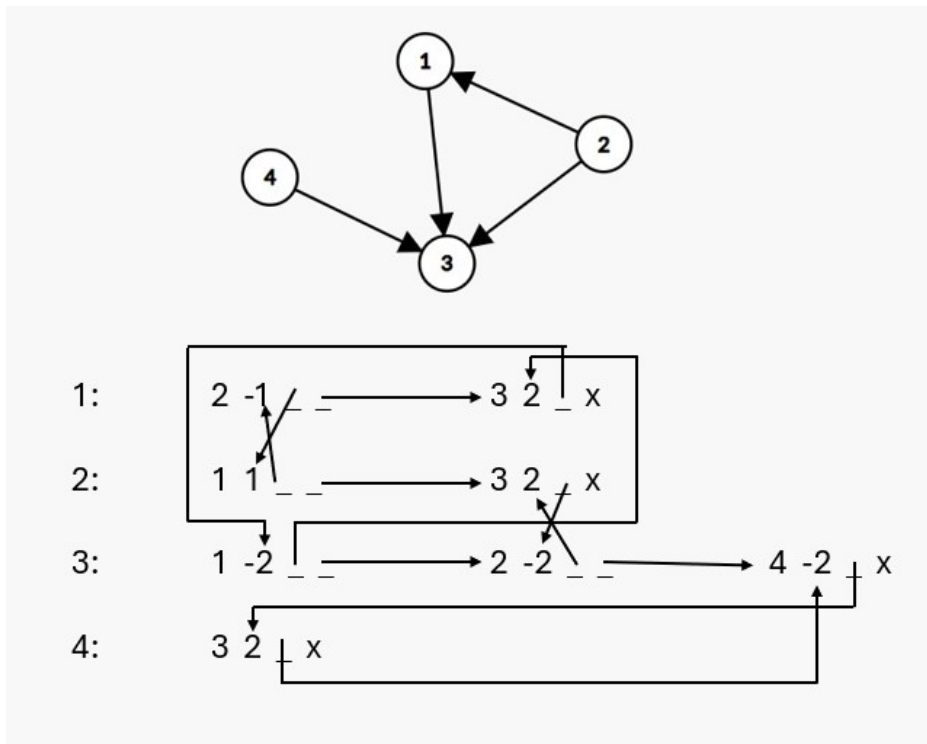
Promotrimo sada kompleksnost prikazanog TRO algoritma.

### 3.2. Graf usporedivosti i TRO

Uzmimo da su susjednosti grafa spremljene u vezanu listu i podaci su poredani uzlazno.

Podatak  $Adj(i)$  reprezentira brid  $v_i v_j$  sa poljima koja sadrže:  $j$ ,  $KLASA[i, j]$ , pokazivač na  $KLASA[j, i]$ , te pokazivač na susjeda od  $Adj(i)$ . Za pohranu je potrebno  $O(|V| + |E|)$ .

Promotrimo još i vrijeme potrebno za pristup i mijenjanje funkcije  $KLASA$ . Za nalaženje  $KLASA[i, l]$  treba  $O(d_i)$  koraka, gdje  $d_i$  stupanj vrha  $v_i$ , kako bi prošli cijeli  $Adj(i)$ , osim ako privremeni pokazivač nije u susjedstvu. Tada referiranje na  $KLASA[i, l]$  ili  $KLASA[l, i]$  bi trebalo trajati fiksni broj koraka.



Slika 3.10: Primjer spremanja  $Adj(i)$  u TRO algoritmu na gore opisan način.

Preostaje još analizirati funkciju  $PRETRAŽIVANJE[i, j]$ . Dva privremena pokazivača paralelno prolaze kroz  $Adj(i)$  i  $Adj(j)$  i traže vrijednosti

### 3.2. Graf usporedivosti i TRO

$m$  koje zadovoljavaju *for* uvjete. Zbog sortiranosti listi i susjednih pokazivača, petlja se može izvesti u  $O(d_i + d_j)$  koraka. Analogno je kompleksnot od  $PRETRAŽIVANJE(i, j)$  jednaka  $O(d_i + d_j)$ .

U glavnom programu, privremeni pokazivač u *for* petlji prolazi kroz sve liste susjeda, što daje složenost od  $O(|E|)$ . Algoritam jednom poziva  $PRETRAŽIVANJE$  za svaki brid ili njegov inverz (oboje ako implikacijske klase nisu disjunktne), pa cijela složenost algoritma koji provjerava je li graf graf usporedivosti i pridružuje mu tranzitivnu orijentaciju će biti:

$$\sum_{v_i v_j \in E} (d_i + d_j) = 2 \sum_{i=1}^n d_i^2 \leq 2 \max \deg(G) \sum_{i=1}^n d_i = 2 \max \deg(G) |E|$$

tj. najviše  $O(\max \deg(G) |E|)$ .

# Poglavlje 4

## Permutacijski grafovi

**Definicija 4.1** *Permutacija* skupa  $S = \{s_1, \dots, s_n\}$  je bijekcija  $f : S \rightarrow S$ .

Broj permutacija skupa od  $n$  elemenata je  $n!$ .

Primjerice, promotrimo funkciju  $S : \{1, 2, 3\} \rightarrow \{1, 2, 3\}$ , za koju vrijedi  $S(1) = 2$ ,  $S(2) = 3$ ,  $S(3) = 1$ . Tada kažemo da je  $\{2, 3, 1\}$  permutacija skupa  $\{1, 2, 3\}$ . Laički, permutacije služe za prikazivanje elemenata u određenom redosljedu.

Permutacije imaju brojne primjene u stvarnome svijetu. Permutacije se često koriste u igrama, logici i biologiji. Mogu se koristiti i u računarstvu, primjerice u algoritmima pretraživanja, te kriptografiji (stvaranju lozinka).

Osim u standardnom obliku, permutacije se mogu reprezentirati uz pomoć grafova što može olakšati rad s njima. Intuitivnije je uočiti uzorke ako ih vizualiziramo, ali i sama svojstva grafa (npr. ima li ciklus) mogu dovesti do zaključaka vezanih za permutaciju. Naravno, kao i kod prethodnih algoritama, permutacijski grafovi mogu pomoći pri optimiziranju pojedinih postojećih algoritama i korisni su za sortiranje.

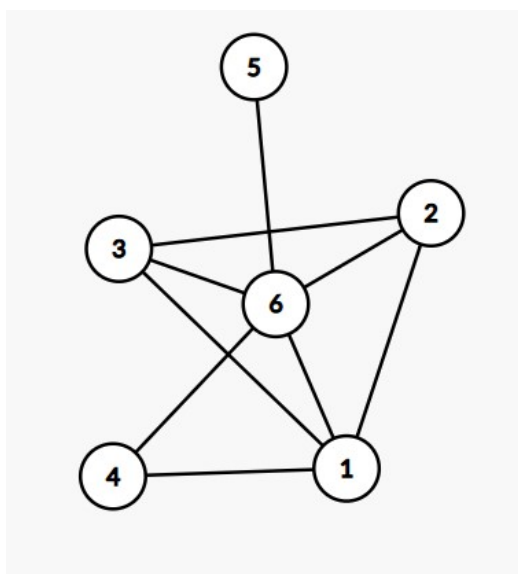
Inverzna permutacija  $\pi^{-1}$  od  $\pi$  je permutacija za koju vrijedi  $\pi(\pi^{-1}(i)) = i$  i  $\pi^{-1}(\pi(i)) = i$ .

**Primjer 4.2** Inverzna permutacija od permutacije  $[6\ 3\ 2\ 4\ 1\ 5]$  je  $[5\ 3\ 2\ 4\ 6\ 1]$ . Inverzna permutacija nam govori na kojoj poziciji u zadanoj permutaciji se nalazi element s trenutne pozicije.

**Definicija 4.3** Neka je  $\pi = [p_1 \dots p_n]$  proizvoljna permutacija skupa  $V = \{1, \dots, n\}$ . Neka je  $G[\pi] = (V, E)$  neusmjereni graf za koji vrijedi:

$$xy \in E \Leftrightarrow (x - y)(\pi^{-1}(x) - \pi^{-1}(y)) < 0.$$

Za graf  $G[\pi]$  kažemo da je **permutacijski**.



Slika 4.1: Graf permutacije  $[6\ 3\ 2\ 4\ 1\ 5]$ .

Komplement permutacijskog grafa je također permutacijski jer vrijedi  $G[\pi^p] = G[\pi]^C$  gdje  $\pi^p$  permutacija koju dobijemo kad "obrnemo" redosljed od  $\pi$  tj. ako  $\pi = [p_1\ p_2\ \dots\ p_n]$ , onda  $\pi^p = [p_n\ \dots\ p_2\ p_1]$

Također vrijedi da se permutacijski graf može tranzitivno orijentirati - to bismo dobili tako da po redu svaki element usmjerimo prema njegovom sljedbeniku. Štoviše, vrijedi i sljedeći teorem.

**Teorem 4.4** *Neusmjereni graf  $G$  je permutacijski ako i samo ako su  $G$  i  $G^C$  grafovi usporedivosti.*

**Dokaz.**  $\Rightarrow$  Pretpostavimo da  $G \equiv G[\pi]$ . Tada je  $G$  graf usporedivosti jer  $G[\pi]$  ima tranzitivnu orijentaciju.  $G^C$  je također graf usporedivosti jer  $G^C \equiv G[\pi^p]$ .

$\Leftarrow$  Uzmimo da su  $(V, F_1)$  i  $(V, F_2)$  tranzitivne orijentacije od  $G = (V, E)$  i  $G^C = (V, E^C)$  redom. Pokažimo da je  $(V, F_1 + F_2)$  aciklična orijentacija potpunog grafa  $(V, E + E^C)$ . Pretpostavimo da  $F_1 + F_2$  ima ciklus  $v_0v_1 \dots v_l v_0$  minimalne duljine  $l$ . Ako je  $l$  veći od 3, onda se ciklus može skratiti uklanjanjem  $v_0v_2$  ili  $v_2v_0$ , pa dobivamo kontradikciju. Ako je  $l = 3$ , onda bar dva brida ciklusa se nalaze u istom  $F_i$  što povlači da  $F_i$  nije tranzitivan. Stoga je  $(V, F_1 + F_2)$  acikličan. Analogno je i  $(V, F_1^{-1} + F_2)$  acikličan.

Konstruirajmo još i permutaciju  $\pi$  takvu da  $G \equiv G[\pi]$ . Aciklična orijentacija potpunog grafa je tranzitivna i određuje jedinstven, linearni poredak vrhova. Promotrimo sljedeći postupak:

1) Labeliramo vrhove ovisno o redoslijedu u  $F_1 + F_2$ . Primjerice,  $L(x) = i$ , gdje je  $\text{indeg}(x) = i - 1$ .

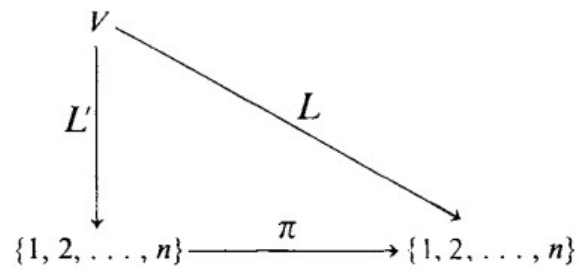
2) Labeliramo vrhove ovisno o redoslijedu u  $F_1^{-1} + F_2$ ,  $L'(x) = i$ , gdje je  $\text{indeg}(x) = i - 1$ .

(\*) Primijetimo da vrijedi  $xy \in E \iff [L(x) - L(y)][L'(x) - L'(y)] < 0$ , jer su bridovi u  $E$  promijenili orijentaciju između 1) i 2).

3) Definirajmo  $\pi$ : Za svaki vrh  $x$ , ako  $L(x) = i$ , onda  $\pi_i^{-1} = L'(x)$ . Sada po (\*),  $\pi$  je tražena permutacija, a  $L$  traženi izomorfizam.

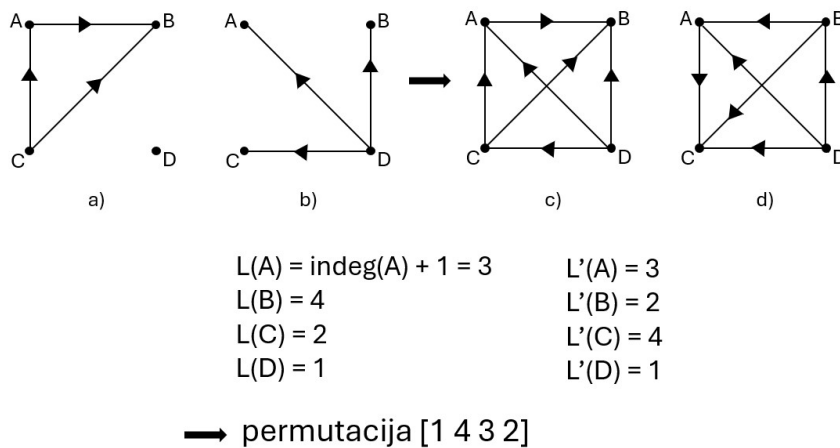
■





Slika 4.2: Odnos  $L$ ,  $L'$  i  $\pi$  iz Teorema 4 - slika iz *Algorithmic Graph Theory and Perfect Graphs* - Martin Charles Golumbic.

Prethodni algoritam govori da ako nađemo tranzitivnu orijentaciju, onda je graf permutacijski. Konstrukcija u dokazu nam pokazuje kako naći odgovarajuću permutaciju. Složenost ovog postupka je  $O(n^2)$ .



Slika 4.3: a)  $F_1$  b)  $F_2$  c)  $F_1 + F_2$  d)  $F_1^{-1} + F_2$

Slika 4.3 prikazuje konstrukcija permutacije [ 1 4 3 2 ] iz tranzitivnih orijentacija  $F_1$  i  $F_2$

Kako bismo grafu mogli pravilno pridružiti vrijednosti permutacije moramo uvesti funkciju koja će to omogućiti.

Za neusmjereni graf  $G = (V, E)$  uvedimo bijekciju  $L : V \rightarrow \{1, \dots, n\}$  koja označava vrhove. Funkcija  $L$  se naziva **labeliranje permutacije** ako postoji permutaciji  $\pi$  skupa od 1 do  $n$  takva da:

$$xy \in E \text{ ako i samo ako } [L(x) - L(y)][\pi^{-1}(L(x)) - \pi^{-1}(L(y))] < 0.$$

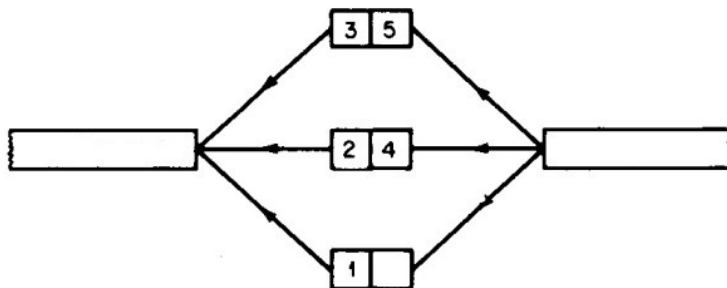
Permutacijski graf mora imati bar jedno labeliranje, te labeliranje ne mora biti jedinstveno.

Promotrimo kako bismo provodili sortiranje permutacije koristeći redove.

Red je način spremanja podataka koji radi po principu *first-in-first-out* tj. FIFO. Točnije, ono što smo stavili prije u red mora prije i izaći.

Koristit ćemo paralelno  $k$  redova na način da gledamo svaki element koji želimo sortirati i stavljamo ga u privremeni red. Na kraju elemente stavljamo u sortirani red na način da po veličini skidamo elemente s privremenih redova. Promotrimo na primjeru.

**Primjer 4.5** Neka  $\pi = [3, 2, 5, 1, 4]$ . Krećemo od prvog elementa 3 i stavljamo ga na prvi red  $Q_1$ . Sljedeći element, 2, ne smijemo staviti u  $Q_1$  jer ne bi više mogli doći do njega, pa ga stavljamo u  $Q_2$ . 5 smijemo staviti na  $Q_1$ , ali 1 ne smijemo staviti ni u  $Q_1$ , ni u  $Q_2$ , pa ga stavljamo u novi  $Q_3$ . 4 ne može u  $Q_1$ , ali može u  $Q_2$ , pa ga stavljamo u njega i nakon toga sortiramo elemente. U ovom slučaju su nam trebala 3 privremena reda za sortiranje.



Iz primjera možemo uočiti da sadržaj svakog reda  $Q_i$  mora biti uzlazno

sortiran kako bi imali točan redoslijed pri rezultatu. Također nije bitno hoćemo li prvo popuniti sve pomoćne redove prije slaganja rezultata ili ćemo istovremeno i namještati finalni rezultat.

Prethodni primjer je obavljen sa tzv. *kanonskom strategijom sortiranja* koja funkcionira na način da vrijednosti iz permutacija stavljamo na prvi red koji je dostupan.

Promotrimo razlog stavljanja brojeva u različite redove u kontekstu grafova. Brojevi se pojavljuju u suprotnom poretku od  $\pi$ , pa ako su vrhovi  $v$  i  $w$  susjedni u  $G[\pi]$ , onda će biti stavljeni u različite redove.

Predstavimo algoritam za bojanje permutacijskog grafa. Dokažimo prvo da postoji veza sortiranja permutacija i bojanja grafa.

**Propozicija 4.6** *Neka  $\pi = [p_1, \dots, p_n]$  permutacija prirodnih brojeva od 1 do  $n$ . Postoji jedan-na-jedan veza između pravilnog  $k$ -bojanja of  $G[\pi]$  i uspješnog sortiranja permutacije s  $k$  paralelnih redova.*

**Dokaz.** Pretpostavimo da svaki red  $Q_i$  boji brojeve sa svojom bojom. Postupkom sortiranja s  $k$  redova opisanim ranije, susjedni vrhovi  $v$  i  $w$  iz  $G[\pi]$  će biti obojani različitim bojama.

Sada provedimo sljedeći postupak nad pravilnim bojanjem permutacijskog grafa s  $k$  boja: ako je boju vrha  $x$  označimo sa  $b$ , spremimo  $x$  u  $Q_b$ .

Pretpostavimo da će ovo rezultirati neuspješnim sortiranjem. Tada mora postojati neki red  $Q_c$  takav da su vrhovi  $x$  i  $y$  spremljeni u krivom redoslijedu. Ali tada  $x$  i  $y$  moraju biti i u obrnutom redoslijedu i u  $\pi$ . Tada su  $x$  i  $y$  susjedi u permutacijskom grafu, ali su obojani istom bojom (jer su u istom redu), pa je to kontradikcija. Sortiranje je uspješno.

Stoga, postoji bijekcija između pravilnog  $k$ -bojanja i uspješnog sortiranja permutacije s  $k$  paralelnih redova. ■

**Korolar 4.7** Za permutaciju  $\pi$  sljedeći brojevi su jednaki:

- (i)  $\chi(G[\pi])$
- (ii) duljina najduljeg silaznog podniza od  $\pi$
- (iii) minimalni broj redova potrebnih za sortiranje  $\pi$

**Dokaz.**  $(i) \iff (iii)$  Direktno iz ropozicije 4.6.

$(i) \iff (ii)$  Iz činjenice da najdulji podniz možemo poistovjetiti s maksimalnom klikom od  $G[\pi]$  koja je duljine  $\chi(G[\pi])$  zbog savršenosti grafa. ■

Iz kanonske strategije sortiranja dobivamo sljedeći algoritam za kanonsko bojanje permutacijskog grafa - algoritam vraća minimalno bojanje.

---

**Algorithm 10** Kanonsko bojanje permutacije

---

**Input:** Permutacija  $\pi = [p_1, \dots, p_n]$  bojeva od 1 do  $n$

**Output:** Bojanje vrhova grafa  $G[\pi]$  i kromatski broj grafa

- 1:  $k = 0$ ;
  - 2: **for**  $j = 1 : n$  **do**
  - 3:      $i =$  prvi dopustivi red;
  - 4:      $BOJA[p_j] = i$ ;
  - 5:      $ZADNJI[i] = p_j$ ;
  - 6:      $k = \max\{k, i\}$ ;
  - 7: **end for**
  - 8:  $\chi = k$ ;
- 

Prethodni teorem radi na način da u  $j$ -toj iteraciji  $p_j$  stavi u red  $Q_i$  sa najmanjim indeksom  $i$  takvim da  $p_j$  veći ili jednak zadnjem elementu iz  $Q_i$ . Ne trebamo spremati cijeli  $Q_i$ , pa u niz  $ZADNJI[i]$  spremamo zadnji element iz  $Q_i$ .  $k$  je brojač koji pamti broj iskorištenih boja tj. redova.

Sljedeći teorem dokazuje točnost prethodnog algoritma.

**Teorem 4.8** *Neka  $\pi$  permutacija brojeva od 1 do  $n$ . Kanonsko bojanje od  $G[\pi]$  iz Algoritma 10 je minimalno.*

**Dokaz.** Bojanje je očito pravilno, pa preostaje još pokazati  $\chi = \chi(G[\pi])$ . Dovoljno je pokazati (po prošlom korolaru) da  $\pi$  ima padajući podniz duljine  $\chi$ .

Definirajmo funkciju  $f$  na sljedeći način: Ako  $BOJA[p_j] = i$  ( $\geq 2$ ), onda  $p_{f(j)} = ZADNJI[i-1]$  za vrijeme  $j$ -te iteracije. Tada  $p_{f(j)} > p_j$  i  $f(j) < j$  jer je činjenica da je  $p_{f(j)}$  u  $Q_{i-1}$  uzrokovala da stavimo  $p_j$  u  $Q_i$ . Tada  $p_{j_1}, \dots, p_{j_\chi}$  gdje  $BOJA[p_{j_\chi}] = \chi$ , te  $p_{j_{i-1}} = p_{f(j_i)}$ ,  $i = \chi, \chi - 1, \dots, 2$  je tražen silazeći podniz. ■

Složenost algoritma je  $O(n \log n)$  i ne ovisi o broju bridova od  $G$ .

# Poglavlje 5

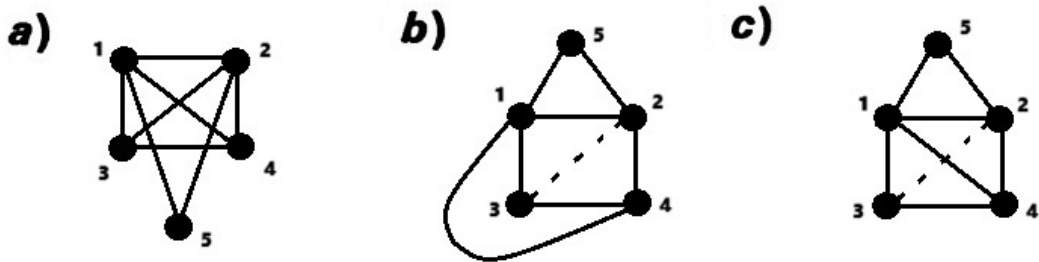
## Planarni grafovi

Planarnost je još jedno bitno svojstvo u teoriji grafova, a njihova najčešća uloga u primjeni je pitanje postoji li mogućnost organiziranja bridova bez bespotrebnog presijecanja. Ovo je, primjerice, korisno u organizaciji prometa, te u telekomunikacijama. Kako rasporediti antene koje su povezane, bez da se signali poklapaju? Planarnost je također interesantna pri dizajniranju igara, te u geografiji. Primjerice, u kreiranju mapa je koristan tzv. Teorem o četiri boje koji govori da se svaki planarni graf može obojati u četiri boje.

Podsjetimo se iz Uvoda da se graf naziva planarnim ako se može nacrtati u ravnini bez presijecanja bridova. Takvo smještenje se naziva planarno ili ravninski graf.

Zbog ovih razloga, korisno je imati algoritam koji provjerava je li dani graf planaran o čemu ćemo govoriti u ovoj sekciji. Prije algoritma, uvedimo dva pojma koja će nam biti potrebna za daljnji rad.

**Definicija 5.1** *Za planarni smještaj  $\tilde{H}$  podgrafa  $H$  grafa  $G$  kažemo da je  $G$ -prihvatljiv ako postoji planarni smještaj  $\tilde{G}$  takav da  $\tilde{H} \subseteq \tilde{G}$ .*



Slika 5.1: a)  $G$  b)  $G$ -prihvatan graf c) nije  $G$ -prihvatan graf

**Definicija 5.2 Blok** je maksimalni povezani podgraf grafa  $G$  koji nema reznih vrhova.

Sa  $\tilde{G}$  ćemo označavati planarno smještenje grafa.

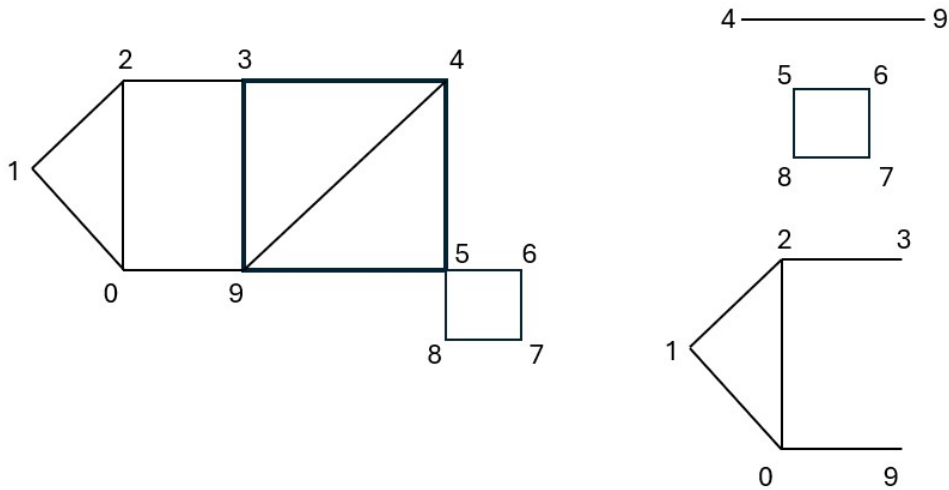
Graf je planaran ako i samo ako mu je svaki blok planaran.

**Definicija 5.3** Neka je  $G_1 = (V_1, E_1)$  podgraf grafa  $G = (V, E)$ . **Komadom** grafa  $G$  u odnosu na  $G_1$  nazivamo jedno od sljedećeg:

a) brid  $uv \in E$  takav da  $uv \notin E_1$  i  $u, v \in V$ .

b) povezana komponenta od  $G - G_1$  u uniji bridovima iz  $E$  takvim da su vrhovi iz dobivene komponente susjedni s vrhovima iz  $V_1$ .

Za svaki komad  $B$ , zajednički vrhovi od  $B$  i  $G_1$  nazivaju se **mjestima kontakta** od  $B$ . Ako komad ima dva ili više mjesta kontakta, onda se naziva **mostom**. Ova definicija mosta se razlikuje od definicije mosta spomenute u uvodu, ali primijetimo da je ova definicija proširenje definicije kabela iz uvoda.



**Primjer 5.4** Promotrimo graf  $G$  na Slici 5.4 lijevo, te njegov podebljano označen podgraf  $G_1 = \{\{3,4\}, \{4,5\}, \{5,9\}, \{3,9\}\}$ . Desno na slici su prikazana 3 komada od  $G$  u odnosu na  $G_1$ :

- brid  $\{4,9\}$  sa mjestima kontakta 4 i 9 (svojstvo a) iz definicije)
- podgraf  $\{\{5,6\}, \{6,7\}, \{7,8\}, \{5,8\}\}$  sa vrhom kontakta 5 (svojstvo b) iz definicije)
- podgraf  $\{\{1,2\}, \{2,3\}, \{0,2\}, \{0,1\}, \{0,9\}\}$  sa vrhovima kontakta 3 i 9 (svojstvo b) iz definicije

Prvi i treći komad su mostovi jer imaju  $\geq 2$  mjesta kontakta.

**Primjer 5.5** Promotrimo graf na Slici 5.1 i uzmimo da je  $G_1 = 12 - 24 - 24 - 13$  (ciklus) promatrani podgraf. Njegovi komadi su sljedeći:

- 1)  $12 - 15 - 25$  sa vrhovima kontakta 1 i 2
- 1)  $14$  sa vrhovima kontakta 1 i 4
- 1)  $23$  sa vrhovima kontakta 2 i 3



### 5.1. Testiranje planarnosti grafa

*Primijetimo da su 2) i 3) kabeli.*

## 5.1 Testiranje planarnosti grafa

Uvedimo sad algoritam koji provjerava planarnost zadanog grafa. Neki postupci mogu olakšati algoritam. Primjerice, zasebno provodimo algoritam na različitim komponentama povezanosti, uklanjamo višestruke bridove prije provedbe algoritma, razdvojimo graf na blokove ako je to moguće ...

Napomenimo da algoritam provjere planarnosti nije jedinstven i prikazan algoritam je implementacija algoritma za provjeru planarnosti. Algoritam prima graf koji je blok.

Neka je  $G_i$  podgraf grafa  $G$ . Uvedimo da je  $F(B, \tilde{G}_i)$  skup stranica od  $\tilde{G}_i$  u koje možemo smjestiti  $B$ .

Algoritam je implementiran na sljedeći način:

### 5.1. Testiranje planarnosti grafa

---

**Algorithm 11** Algoritam za provjeru planarnosti

---

**Input:** Graf  $G$  koji je blok.

**Output:** Poruka o planarnosti grafa  $G$ .

```
1: Nađi ciklus  $C$  u grafu  $G$ 
2:  $i = 1$ ;
3:  $G_1 = C$ ;
4:  $\tilde{G}_1 = C$ ;
5:  $f = 2$ ;
6:  $SMJESTIV = \text{true}$ ;
7: while  $f \neq |E| - n + 2$  i  $SMJESTIV == \text{true}$  do
8:   Nađi svaki most  $B$  od  $G$  u odnosu na  $G_i$  i pridruži mu  $F(B, \tilde{G}_i)$ 
9:   if  $F(B, \tilde{G}_i) = \emptyset$ , za neki  $B$  then
10:      $SMJESTIV = \text{false}$ ;
11:     return " $G$  nije planaran";
12:   end if
```

---

### 5.1. Testiranje planarnosti grafa

---

**Algorithm 12** Algoritam za provjeru planarnosti (nastavak)

---

```
13:   if SMJESTIV == true then
14:       if  $|F(B, \tilde{G}_i)| == 1$ , za neki  $B$  then
15:            $F = F(B, \tilde{G}_i)$ ;
16:       else
17:           Neka  $B$  neki most i  $F$  neka strana za koju vrijedi  $F \in F(B, \tilde{G}_i)$ 
18:       end if
19:       Nađi put  $P_i \subseteq B$  koji povezuje dvije točke kontakta iz  $B$  sa  $G_i$ 
20:        $G_{i+1} = G_i + P_i$ ;
21:       Nađi planarno smještenje  $\tilde{G}_{i+1}$  od  $G_{i+1}$  tako da nacrtamo  $P_i$  u
           stranici  $F$  od  $\tilde{G}_i$ ;
22:        $i += 1$ ;
23:        $f += 1$ ;
24:       if  $f = |E| - n + 2$  then
25:           return "G je planaran";
26:       end if
27:   end if
28: end while
```

---

Algoritam pronalazi  $G_1, G_2, \dots (G_i \subset G_{i+1})$  i njihova planarna smještenja  $\tilde{G}_1, \tilde{G}_2, \dots$

Algoritam provjerava planarnost tako da provjerava  $G$ -prihvatljivost svakog  $\tilde{G}_i$ . Ako su svi  $\tilde{G}_i$   $G$ -prihvatljiv, onda će graf biti planaran i  $\tilde{G}_{|E|-n+1}$  će biti njegovo planarno smještenje. Ako nađemo most  $B$  takav da  $F(B, \tilde{G}_i) = \emptyset$ , graf nije planaran.

Algoritam prvo pronalazi ciklus  $C$  koji mora postojati jer je  $G$  blok.  $G_1$  je taj ciklus i onda ujedno i planaran. *SMJESTIV* je varijabla koja pamti smjestivost grafa i zadana vrijednost joj je true. *SMJESTIV* prelazi u false

### 5.1. Testiranje planarnosti grafa

ako nađemo most  $B$  takav da  $F(B, \tilde{G}_i) = \emptyset$ .  $f$  je varijabla koja pamti broj strana od  $\tilde{G}_i$ .

Svaka iteracija *while* petlje konstruira novi  $\tilde{G}_{i+1}$  tako da iz trenutnog  $\tilde{G}_i$  nađe skup svih mostova u odnosu na  $G_i$ , te za svaki takav most  $B$  nađe njegov skup  $F(B, \tilde{G}_i)$ . Ako postoji  $B$  koji je smješten samo u jednoj stranici  $F$  od  $\tilde{G}_i$ , onda  $\tilde{G}_{i+1}$  konstruiramo tako da nacrtamo put između vrhova kontakta  $B$  u  $F$  (ako takav most ne postoji onda uzmemo put između točaka kontakta bilo kojeg mosta). Put dijeli  $F$  na dva dijela i povećava se  $f$  ( broj strana ).

Iz Teorema 1.15 znamo da planarni  $G$  ima  $|E| - n + 2$  strana što je kriterij zaustavljanja algoritma.

**Teorem 5.6** *Algoritam za provjeru planarnosti je ispravan.*

**Dokaz.** Trebamo pokazati da za svaki graf iz niza  $\tilde{G}_1, \dots, \tilde{G}_{|E|-n+1}$ , ako je  $G$  planaran, onda je i  $G$ -prihvatljiv. Dokaz provodimo indukcijom.

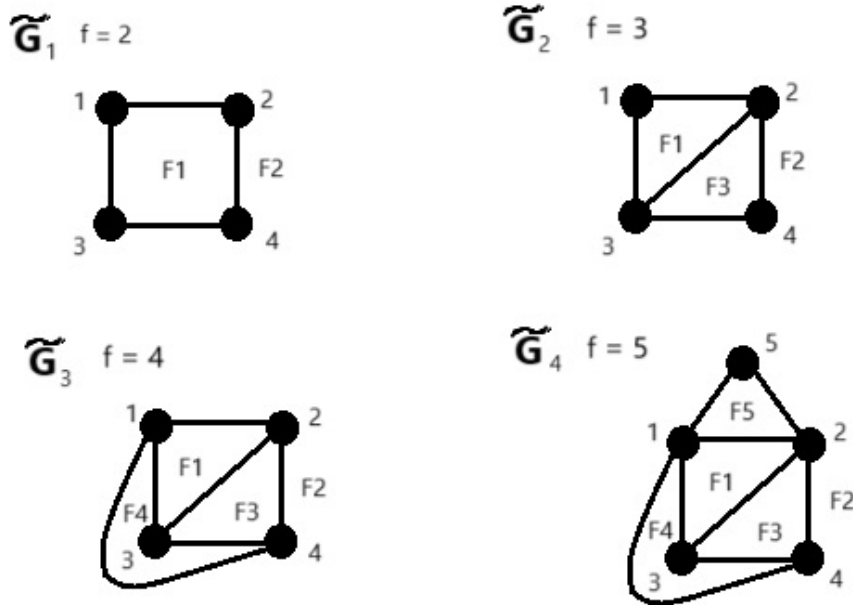
Ako je  $G$  planaran,  $\tilde{G}_1$  je sigurno  $G$ -prihvatljiv. Pretpostavimo da je  $\tilde{G}_i$ ,  $1 \leq i \leq k < |E| - n + 1$ ,  $G$ -prihvatljiv. Pokažimo da je i  $\tilde{G}_{k+1}$  također  $G$ -prihvatljiv. Neka su  $B$  i  $F$  definirani kao i u algoritmu. Neka  $\tilde{G}$  planarno smještenje od  $G$  takvo da  $\tilde{G}_k \subset \tilde{G}$ .

Ako  $|F(B, \tilde{G}_k)| = 1$ , onda po konstrukciji algoritma vrijedi  $\tilde{G}_{k+1} \subseteq \tilde{G}$ .

Stoga pretpostavimo  $|F(B, \tilde{G}_k)| > 1$  i pretpostavimo da  $B$  nije smješten u  $F$  od  $\tilde{G}$ , nego u nekoj drugoj stranici  $F^*$ . Sada je  $G$  blok i  $G_k$  takav da ima bar dva vrha kontakta, pa je smješten samo u dvije strane. Iz ovoga slijedi da svaki most s mjestima kontakta na granici između strana  $F$  i  $F^*$  može biti sam u  $F$  ili u  $F^*$ . Iz ovoga slijedi da postoji još jedno planarno smještenje od  $G$  u kojem će  $B$  biti u  $F$  ako se nalazi u  $F^*$  u  $\tilde{G}$ , te je u  $F^*$  ako u  $F$  u  $\tilde{G}$ .

$\tilde{G}_{k+1}$  dobiven iz algoritma je  $G$ -prihvatljiv jer  $\tilde{G}_{k+1} \subseteq \tilde{G}$ . ■

### 5.1. Testiranje planarnosti grafa



Slika 5.2: Primjer provedbe algoritma planarnosti na grafu  $G$  sa Slike 5.1 a).

**Primjer 5.7** Promotrimo primjer provedbe algoritma sa Slike 5.2. Odaberemo ciklus  $12 - 24 - 34 - 13$ .

Za  $\tilde{G}_1$ ,  $f = 2$ , biramo most  $\{2, 3\}$  u  $F_1 \Rightarrow$  dodajemo put  $2 - 3$ .

Za  $\tilde{G}_2$ ,  $f = 3$ , biramo most  $\{1, 4\}$  u  $F_2 \Rightarrow$  dodajemo put  $1 - 4$ .

Za  $\tilde{G}_3$ ,  $f = 4$ , biramo most  $\{1, 5\}, \{5, 2\}$  u  $F_2 \Rightarrow$  dodajemo put  $1 - 5 - 2$ .

Za  $\tilde{G}_4$ ,  $f = 5$ , dobivamo planarni prikaz grafa sa Slike 5.1.

Algoritam se može implementirati u polinomijalnom vremenu. *while* petlja se izvršava najviše  $|E| - n + 1$  puta.

# Literatura

- [1] Golumbic Martin Charles, *Annals of Discrete Mathematics: Algorithmic Graph Theory and Perfect Graphs*, Caesarea Rothschild Institute, University of Haifa, Haifa, Israel 2004
- [2] Gibbons Alan, *Algorithmic graph theory*, Cambridge University Press, Cambridge ; New York 1985.
- [3] Newman M. E. J., *Networks An Introduction*, Oxford University Press Inc., New York 2010.
- [4] Golemac Anka, *Teorija Grafova*, Prirodoslovno-matematički fakultet, Split 2022.
- [5] *GeeksForGeeks* website (<https://www.geeksforgeeks.org/>) - zadnje posjećena 16.8.2024.
- [6] Perić Jurica, *Složenost algoritama*, Prirodoslovno-matematički fakultet, Split 2023.