# Domination on Modular Product Graphs

Maks Rozman, Nea Lovrinčić, Rene Turk

31. januar 2025

# 1 Uvod

V projektni nalogi smo preučevali dominacijo (dominacijsko število) na modularnem produktu grafov. Modularni produkt smo označevali z znakom $\diamond$, dominacijsko število pa z $\gamma$. Kot je predvideno v navodilih, smo si delo razdelili na dva dela. Zaradi težav pri uporabi Sage-a na naših napravah (ter omejitev plačljive spletne storitve CoCalc), smo kodo pisali v Pythonu s pomočjo knjižnice `NetworkX`.

# 2 Definicije

Delali smo z neusmerjenimi grafi $G = (V, E)$, kjer je $V$ množica vozlišč, $E$ pa množica povezav grafa.

## 2.1 Dominacijska množica in dominacijsko število

**Definicija 2.1.** Dominacijska množica grafa $G = (V, E)$ je podmnožica $D \subseteq V$, za katero velja, da je vsako vozlišče, ki ni v $D$, sosed vsaj enemu vozlišču iz $D$. Dominacijsko število $\gamma(G)$ je moč najmanjše dominacijske množice grafa $G$.

## 2.2 Modularni produkt

**Definicija 2.2.** Modularni produkt $G \diamond H$ grafov $G$ in $H$ je graf z množico vozlišč $V(G) \times V(H)$, ki je unija kartezičnega produkta, direktnega produkta in direktnega produkta komplementov $G$ in $H$:

$$G \diamond H = G \square H \cup G \times H \cup \overline{G} \times \overline{H}.$$

Natančneje, vozlišči $(g, h)$ in $(g', h')$ grafa $G \diamond H$ sta sosednji, če:

1. $g = g'$ in $hh' \in E(H)$ ali $gg' \in E(G)$ in $h = h'$; **ali**
2. $gg' \in E(G)$ in $hh' \in E(H)$; **ali**
3. (ko $g \neq g'$ in $h \neq h'$) $gg' \notin E(G)$ in $hh' \notin E(H)$.

# 3 Prvi del projektne naloge

V prvem delu smo iskali grafe $G$, za katere velja $\gamma(G \diamond G) \geq \gamma(G) + 2$ .

## 3.1 Stohastično iskanje ustreznih grafov

Najprej smo v datoteko `funkcije.py` zapisali štiri funkcije, ki smo jih potem uporabljali za iskanje ustreznih grafov. in sicer:

- `modularni_produkt` za konstrukcijo grafa modularnega produkta dveh grafov,
- `dominacijsko_stevilo`, da smo dobili dominacijsko število grafa,
- `zmanjsaj_premer_na_2`, ki je danemu grafu dodajala povezave, dokler ni bil premer grafa enak 2,
- `stohasticno_iskanje_grafa`, ki je stohastično generirala grafe (dejansko smo na koncu imeli tri, ker smo poskusili tri nekoliko različne načine konstrukcije).

### 3.1.1 Modularni produkt grafov

Psevdokoda za konstrukcijo grafa modularnega produkta dveh grafov:

**Algorithm 1** Modularni produkt grafov

**function** ModularniProdukt(G, H)
    GH ← nov prazen graf
    GH_vozlišča ← $\{(g, h) \mid g \in V(G), h \in V(H)\}$ ▷ Vozlišča GH so kartezični produkt vozlišč G in H.
    Dodaj GH_vozlišča v GH
    **for all** $(g_1, h_1) \in$ GH_vozlišča **do** ▷ Inicializiramo dve for zanki za dodajanje povezav po treh pravilih modularnega produkta (glej definicijo 2.2).
        **for all** $(g_2, h_2) \in$ GH_vozlišča **do**
            **if** $g_1 = g_2$ in $(h_1, h_2) \in E(H)$ **then**
                Dodaj povezavo $((g_1, h_1), (g_2, h_2))$ v GH
            **else if** $(g_1, g_2) \in E(G)$ in $h_1 = h_2$ **then**
                Dodaj povezavo $((g_1, h_1), (g_2, h_2))$ v GH
            **else if** $(g_1, g_2) \in E(G)$ in $(h_1, h_2) \in E(H)$ **then**
                Dodaj povezavo $((g_1, h_1), (g_2, h_2))$ v GH
            **else if** $g_1 \neq g_2$ in $h_1 \neq h_2$ in $(g_1, g_2) \notin E(G)$ in $(h_1, h_2) \notin E(H)$ **then**
                Dodaj povezavo $((g_1, h_1), (g_2, h_2))$ v GH
            **end if**
        **end for**
    **end for**
    **return** GH ▷ Vrnemo graf GH modularnega produkta grafov G in H.
**end function**

### 3.1.2 Dominacijsko število

Sledi zapis linearnega programa za izračun dominacijskega števila grafa. Prva oblika te funkcije je bila "brute-force". Za vsako število $i$ od 1 do vključno $n$ smo generirali vse možne podmnožice vozlišč moči $i$. Ko je funkcija za dano podmnožico vozlišč ugotovila, da je dominacijska, je vrnila število $i$ (ki je avtomatsko dominacijsko število, saj je $i$ moč najmanjše dominacijske množice grafa). Funkcija je pravilno delovala, vendar je bila časovno zelo potratna ($O(2^n)$). Po posvetovanju s prof. Škrekovskim smo za računanje

dominacijskega števila vpeljali časovno veliko učinkovitejši linearni program (v Python funkcijo smo ga implementirali z orodjem PuLP):

$$x_v = \begin{cases} 1; & v \in V(G) \text{ je v dominacijski množici} \\ 0; & v \in V(G) \text{ ni v dominacijski množici} \end{cases}$$

Hočemo

$$\min \sum_{v \in V(G)} x_v,$$

pri pogoju

$$x_v + \sum_{u \in N(v)} x_u \geq 1, \quad \forall v \in V(G),$$

kjer je $N(v)$ množica vseh sosedov vozlišča $v$ v grafu $G$:

$$N(v) = \{u \in V(G) \mid (u, v) \in E(G)\}.$$

Končna vrednost dominacijskega števila je

$$\sum_{v \in V(G)} x_v.$$

### 3.1.3 Zmanjšaj premer na 2

Znano je, da če obstajajo grafi, ki zadovoljijo ta pogoj, mora biti njihov premer enak 2. Zato smo grafe, ki smo jih stohastično konstruirali, s pomočjo naslednje metode preuredili, da so imeli premer enak 2:

1. Če vhodni graf ni bil povezan, je funkcija vrnila napako. Zato smo najprej naključno dodajali povezave, dokler graf ni bil povezan.

2. Če je bil premer grafa večji od 2, je metoda najprej poiskala dve vozlišči na grafu, ki sta bili na največji razdalji. Med njima je nato naključno dodala povezavo.

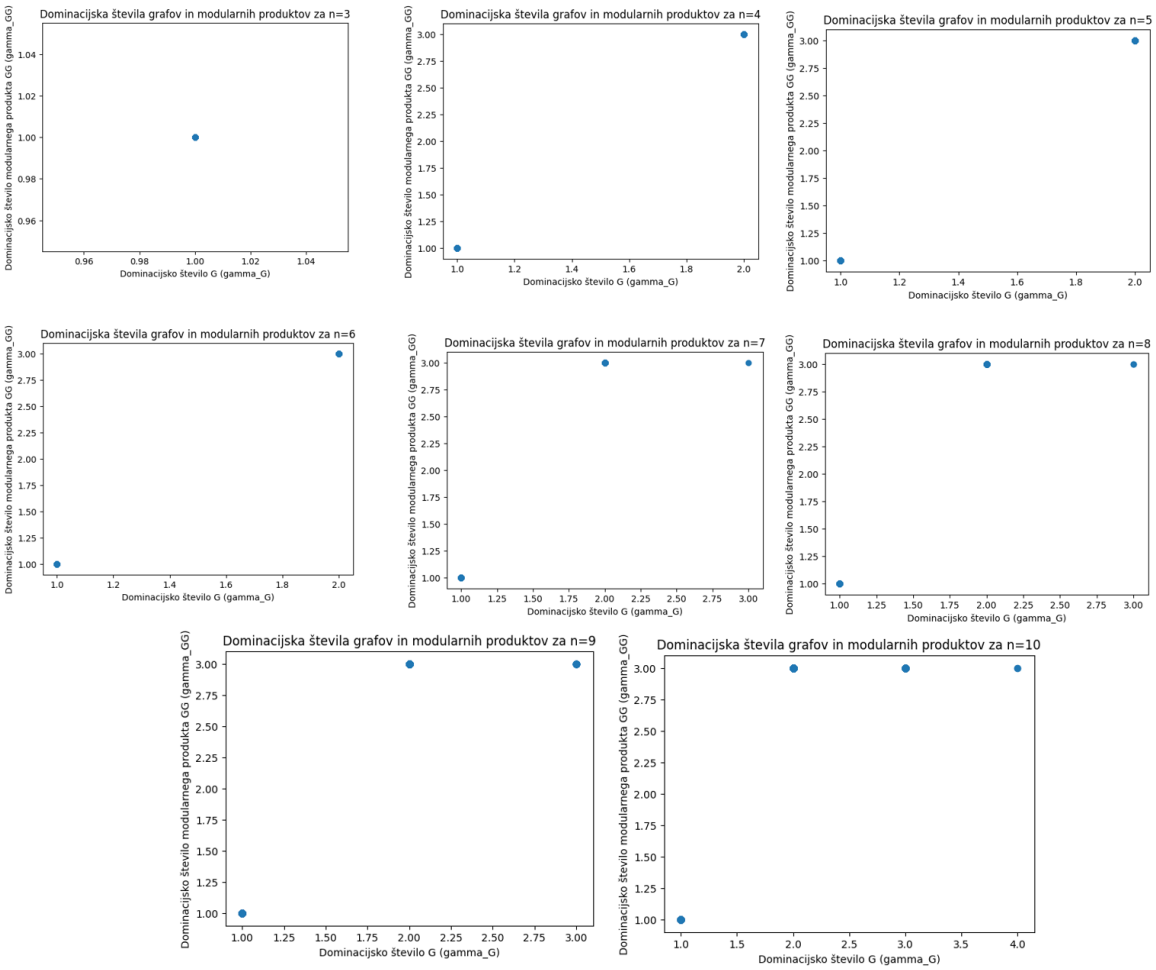
3. Nato je izbrala drugi dve vozlišči, ki sta bili na maksimalni povezavi, in tako naprej, dokler ni bil premer grafa enak 2.

### 3.1.4 Stohastična konstrukcija na tri načine

Grafe, ki ustrezajo pogoju, smo iskali stohastično na tri podobne načine. Za vsako število vozlišč $n$ smo konstruirali 1000 grafov. Načini so se razlikovali v tem, da smo pri prvem načinu vsak graf na novo konstruirali, kjer je bilo $n$ število vozlišč, število povezav pa je bilo naključno. Če graf ni bil povezan, smo iteracijo zanke preskočili. Pri drugem načinu smo konstruirali graf na $n$ vozliščih brez povezav, in nato naključno dodajali povezave, dokler graf ni bil povezan. Hoteli smo poskusiti, če na ta način morda dobimo drugačne grafe kot v prvem primeru. Pri obeh načinih smo potem enako uporabili funkcijo za zmanjšanje premera grafa na 2, in potem testirali, če je graf ustrezen. Pri tretjem načinu pa smo graf kot pri drugem načinu konstruirali in ga testirali, nismo pa v naslednji iteraciji naredili novega grafa, temveč smo staremu odstranili nekaj povezav, spet uporabili funkcijo za zmanjšanje premera na 2 in tako naprej.

### 3.1.5 Ugotovitve

Pri iskanju grafov žal nismo bili uspešni, saj kljub testiranju na velikem številu grafov (od 2 do vključno 22 vozlišč) nismo našli nobenega, ki bi ustrezal pogoju. S funkcijami, ki so delovale na isti način, kot funkcije za stohastično generiranje grafov, smo konstruirali 1000 grafov, za vsako število vozlišč od 3 do 10. Potem smo naredili prikaz, kjer smo za vsak graf $G$ izračunali njegovo dominacijsko število (x-os) ter dominacijsko število modularnega produkta $(G \diamond G)$ (y-os):




Vidimo lahko, da smo zajeli maksimalno možno dominacijsko število povezanega grafa za vsako število vozlišč (kar nam pove, da smo uspeli generirati med sabo dovolj različne grafe). Vidimo lahko tudi da je dominacijsko število modularnega produkta $(G \diamond G)$ kvečjemu za ena večje od dominacijskega števila $G$, za grafe, ki smo jih uspeli generirati.

## 3.2 Iskanje grafov, za katere velja $\gamma(G \diamond G) = \gamma(G) + 1$

Pri prejšnji točki smo ugotovili, da nismo dobili grafov, za katere bi veljalo $\gamma(G \diamond G) \geq \gamma(G) + 2$. Smo pa dobili grafe, za katere velja $\gamma(G \diamond G) = \gamma(G) + 1$. V tem podpoglavju bomo predstavili naše ugotovitve, glede grafov, za katere velja zgornji pogoj.

### 3.2.1 Način iskanja

Grafov, ki ustrezajo novemu pogoju, nismo več iskali stohastično, temveč na način, da smo za vsak možen povezan graf na $n$ vozliščih testirali, ali je ustrezen. To smo naredili s pomočjo funkcije `generiraj_vse_povezane_grafe`, ki nam je vrnila seznam vseh povezanih grafov na $n$ vozliščih. Njena slaba stran je izredno visoka računska zahtevnost ($O(2^{\frac{n(n-1)}{2}})$). Funkcija je zaradi tega uporabna le za majhne $n$, do vključno 6, za večje grafe postane računsko prezahtevna.
Za večje grafe tako nismo iskali več vseh povezanih grafov, ki ustrezajo pogoju. Dobili smo le nekaj ustreznih za določeno število vozlišč, na katerih smo potem preverjali naše ugotovitve (konkretno smo jih imeli 240 na 7 vozliščih, 19 na 8 vozliščih, 5 na 9 vozliščih, 4 na 10 vozliščih, 3 na 11 vozliščih, 5 na 12 vozliščih in 5 na 13 vozliščih → skupaj 334).

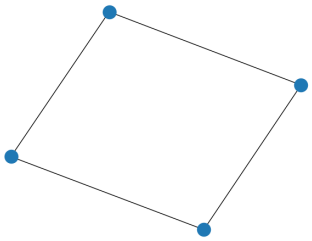
**Algorithm 2** Generiranje vseh povezanih grafov na $n$ vozliščih

```
1: function GenerirajVsePovezaneGrafe(n)
2:     vozlišča ← {1, ..., n}
3:     vse_možne_povezave ← vse možne povezave med pari vozlišč v polnem grafu
4:     povezani_grafi ← ∅
5:     for all k ∈ {n − 1, ..., |vse_možne_povezave|} do      ▷ Začnemo z minimalnim
       številom povezav (drevo), da je graf lahko povezan
6:         for all povezave ∈ vse kombinacije dolžine k iz vse_možne_povezave do
7:             G ← nov prazen graf
8:             Dodaj vozlišča v G                          ▷ Vozlišča so za vsak G enaka
9:             Dodaj povezave v G
10:            if G je povezan then
11:                Dodaj G v povezani_grafi
12:            end if
13:        end for
14:    end for
15:    return povezani_grafi
16: end function
```

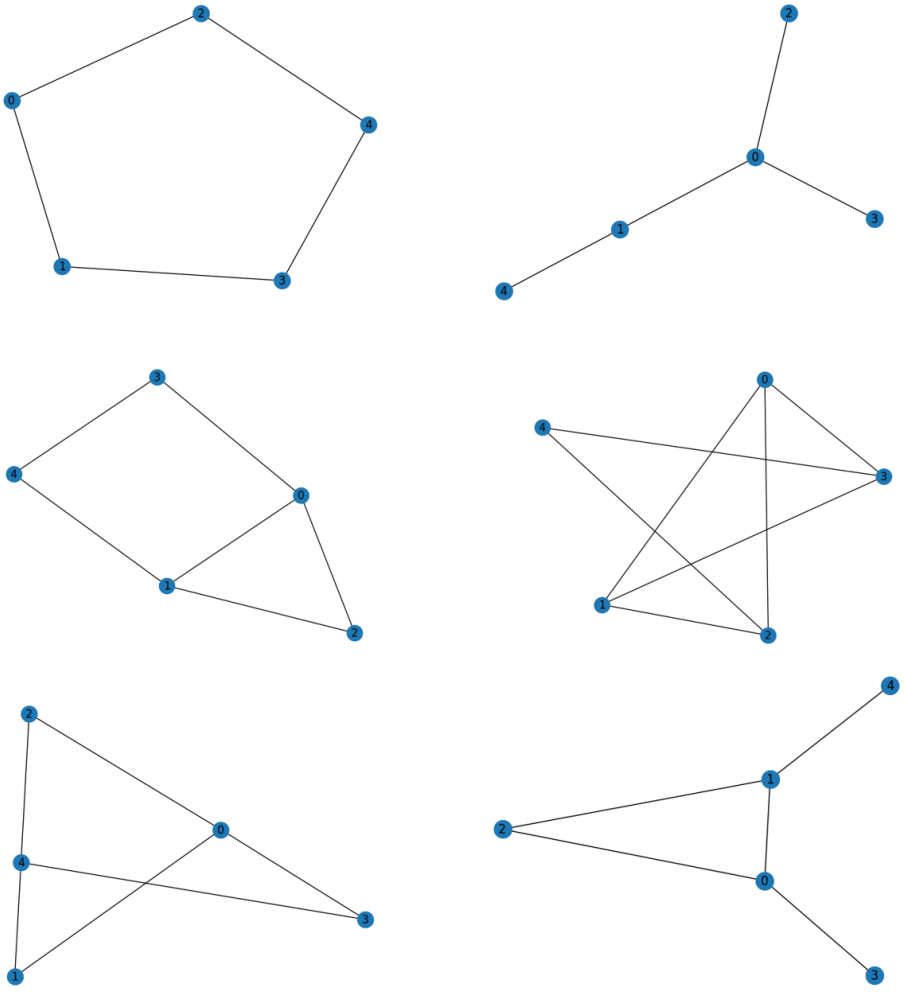
Za vsak povezan graf, ki smo ga s pomočjo prejšnje funkcije dobili, smo nato testirali, ali ustreza pogoju. Ko smo dobili seznam ustreznih grafov, smo s pomočjo še ene funkcije `vrni_neizomorfne_grafe` seznam skrajšali tako, da so bili v njem samo grafi, ki med seboj niso bili izomorfni (funkcija je v glavnem delovala s pomočjo `GraphMatcher(G,H).is_isomorphic()`, ki je vgrajena v knjižnico `NetworkX`).

### 3.2.2 Ustrezni grafi

- Za $n = 2$ in $n = 3$ ne najdemo nobenega grafa, ki bi ustrezal pogoju;
- Za $n = 4$ dobimo, da je edini ustrezen graf:

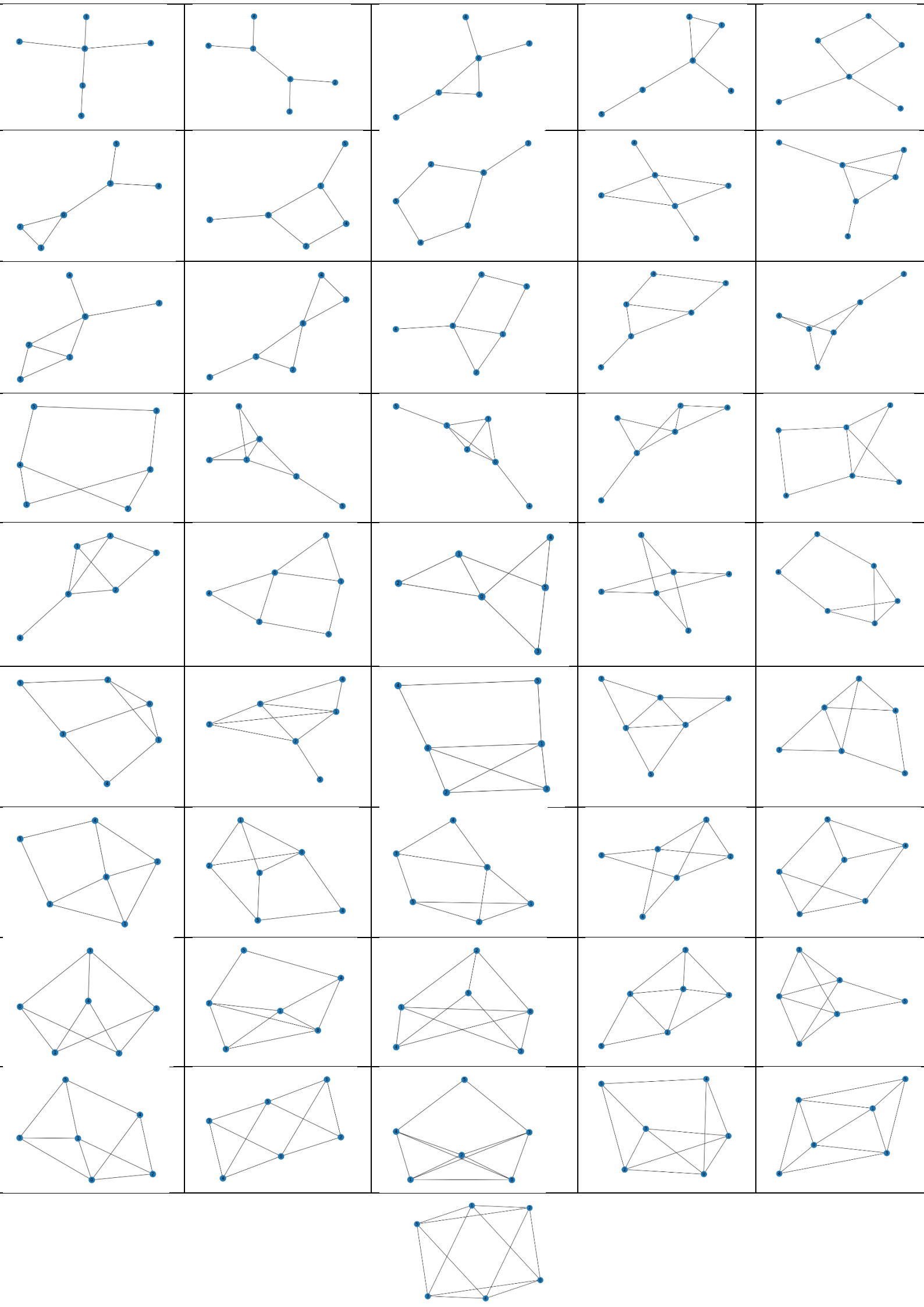


- Za $n = 5$, dobimo 6 grafov, ki ustrezajo pogoju:




Do vključno tu bi lahko sklepali, da bo $n$-kotnik vedno ustrezal. Vendar že za $n = 6$ to ne velja več, saj za 6-kotnik velja $\gamma(G \diamond G) = 2$ in $\gamma(G) = 2$. Podobno velja za 7-kotnik, le da je dominacijsko število v obeh primerih enako 3.

- Za $n = 6$ dobimo 46 ustreznih grafov:



Pri $n = 6$ smo naše iskanje končali, saj se je koda za $n = 7$ poganjala veliko preveč časa. Kot že napisano, za grafe od $n = 7$ do vključno $n = 13$ nismo generirali vseh povezanih grafov, temveč le nekaj ustreznih.

### 3.2.3 Ugotovitve

Za dobljene ustrezne grafe smo iskali morebitne podobnosti pri naslednjih lastnostih: dvodelnost, prisotnost Eulerjevega cikla, premer, radij, kromatično število, število vseh klik in povprečna dolžina poti. Dobili smo ujemanje pri radiju, kjer je pri vseh 334 grafih ta bil enak 2. Radij grafa je najmanjša ekscentričnost med vsemi vozlišči, kjer je ekscentričnost vozlišča najdaljša najkrajša pot od tega vozlišča do katerega koli drugega vozlišča v grafu. Za vse druge preiskovane lastnosti grafov nismo ugotovili nič konkretnega (vrednosti so se razlikovale).

### 3.2.4 Zaključek

S stohastično konstrukcijo velikega števila grafov nismo uspeli najti nobenega, ki bi ustrezal pogoju $\gamma(G \diamond G) \geq \gamma(G) + 2$. Smo pa proučevali grafe, za katere velja pogoj $\gamma(G \diamond G) = \gamma(G) + 1$ in ugotovili, da je prav za vseh 334 proučevanih grafov veljalo, da je njihov radij enak 2.

# 4 Drugi del projektne naloge

## 4.1 Uvod

V tej nalogi smo preverjali veljavnost neenakosti $\gamma(G \diamond H) \leq 4$ za modularni produkt grafa $G$ s premerom vsaj 3 in grafa $H$ tipa (SB). Pri tem smo uporabili podatke in funkcije, ki so jih razvile skupine 20 in 21. Cilj je bil najti pare grafov $G$ in $H$, pri katerih se doseže zgornja meja $\gamma(G \diamond H) = 4$, ter preveriti, ali obstajajo pari, kjer neenakost ne drži.

## 4.2 Metodologija

Za generiranje grafov smo uporabili naslednje pristope:

- Graf $G$ je bil naključno generiran z uporabo modela Erdős–Rényi, pri čemer smo preverili, da je povezan in da ima premer vsaj 3.
- Graf $H$ je bil generiran kot graf tipa (SB) s pomočjo funkcije `generate_random_sb()`.
- Izračunan je bil modularni produkt $G \diamond H$.
- Dominacijsko število $\gamma(G \diamond H)$ je bilo izračunano s funkcijo `dominacijsko_stevilo()`.
- Za vsako kombinacijo velikosti grafov (do vključno 20 vozlišč) je bilo izvedenih 1000 iteracij.

## 4.3 Rezultati

Po prvotno izvedenih simulacijah so bili dobljeni naslednji rezultati:

- Najden ni bil noben par grafov $G$ in $H$, za katerega bi neenakost $\gamma(G \diamond H) \leq 4$ ne držala.
- Prav tako ni bil najden noben par, za katerega bi držala enakost $\gamma(G \diamond H) = 4$.
- Vsi generirani primeri so potrjevali domnevo, da $\gamma(G \diamond H) \leq 4$.

## 4.4 Iskanje grafov, za katere velja $\gamma(G \diamond H) \leq 3$

Glede na to, da grafov, ki bi ustrezali neenakosti $\gamma(G \diamond H) \leq 4$ nismo našli, smo se odločili, da zožimo neenakost in pogledamo, ali dobimo kak graf, ki bi ustrezal neenakosti $\gamma(G \diamond H) \leq 3$ oz. enakosti $\gamma(G \diamond H) = 3$ .

Začeli smo s preverjanjem kombinacij števila vozlišč za grafa $G$ in $H$. Za kombinacije (prvo število je število vozlišč za $G$, drugo pa za $H$), kjer smo preverjali kombinacije do 8 vozlišč, smo dobili, da za:

- (5,5)
- (5,6)
- (6,5)
- (5,7)
- (7,5)
- (5,8) ter
- (8,5)

ne obstajajo ustrezni pari grafov, ki bi zadoščali $\gamma(G \diamond H) = 3$ . Iz tega sklepamo, da ne obstaja par, kjer ima eden izmed $G$ ali $H$ 5 vozlišč ali manj, ki zadošča enakosti.

### 4.4.1 Kombinacija (6,6)

Za kombinacijo (6,6), ob naključnem izbiranju grafov $G$ in $H$, smo dobili 119 primernih parov izmed 1000 iteracij, ki ustrezajo $\gamma(G \diamond H) = 3$ .

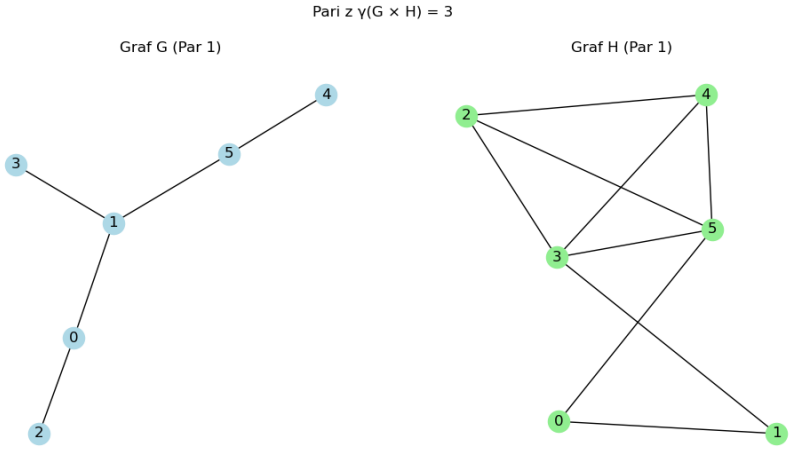



Figure 1: Primer 1

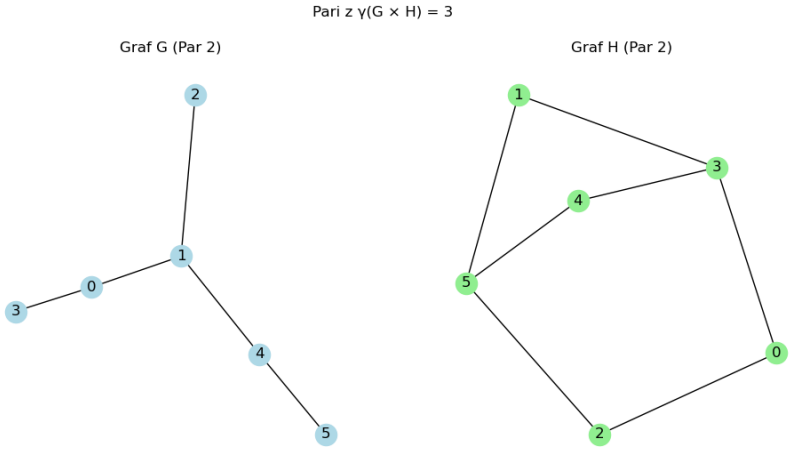



Figure 2: Primer 2

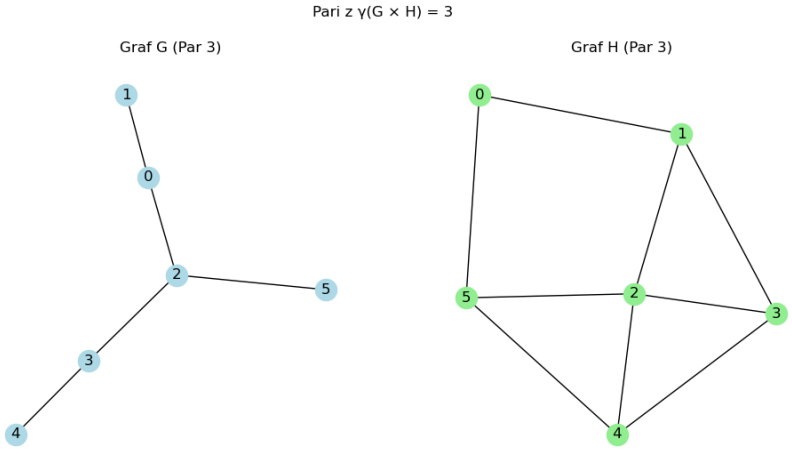



Figure 3: Primer 3

### 4.4.2 Kombinacija (6,7)

Za kombinacijo (6,7), ob naključnem izbiranju grafov $G$ in $H$, smo dobili 106 primernih parov izmed 1000 iteracij, ki ustrezajo $\gamma(G \diamond H) = 3$ .

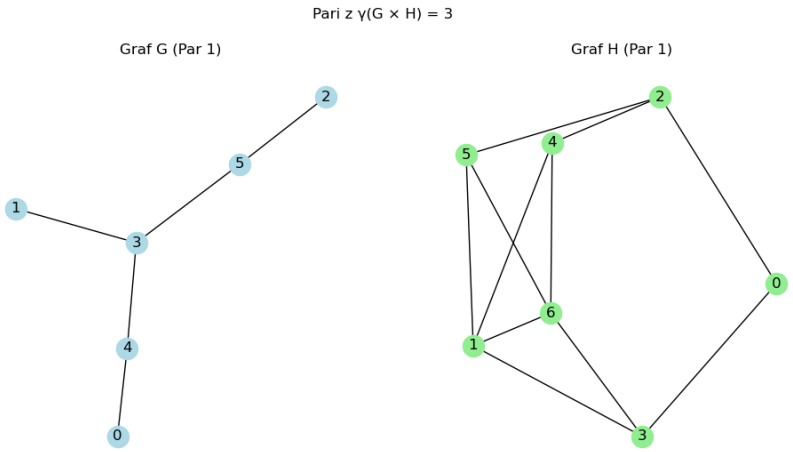



Figure 4: Primer 1

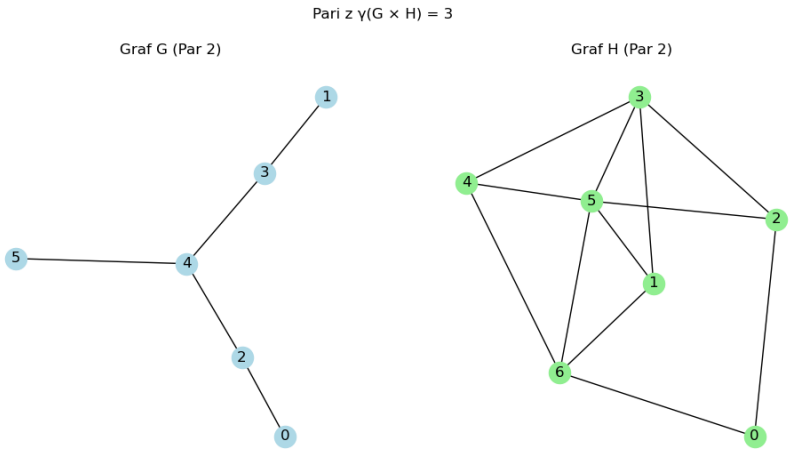



Figure 5: Primer 2

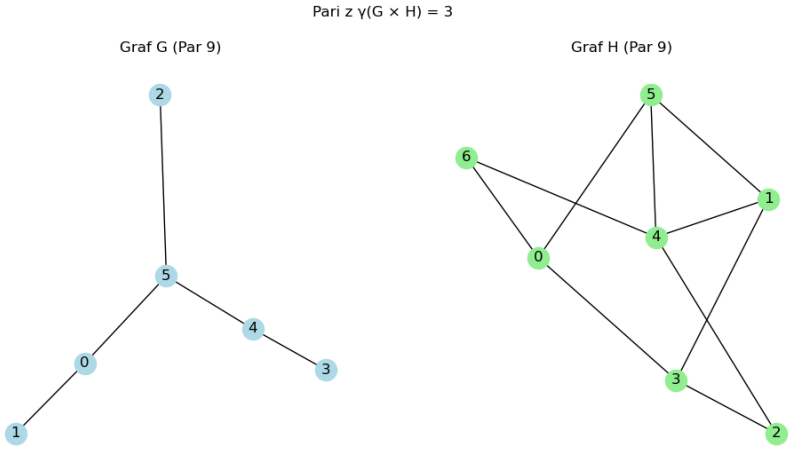



Figure 6: Primer 3

### 4.4.3 Kombinacija (7,6)

Za kombinacijo (7,6), ob naključnem izbiranju grafov $G$ in $H$, smo dobili 360 primernih parov izmed 1000 iteracij, ki ustrezajo $\gamma(G \diamond H) = 3$ .

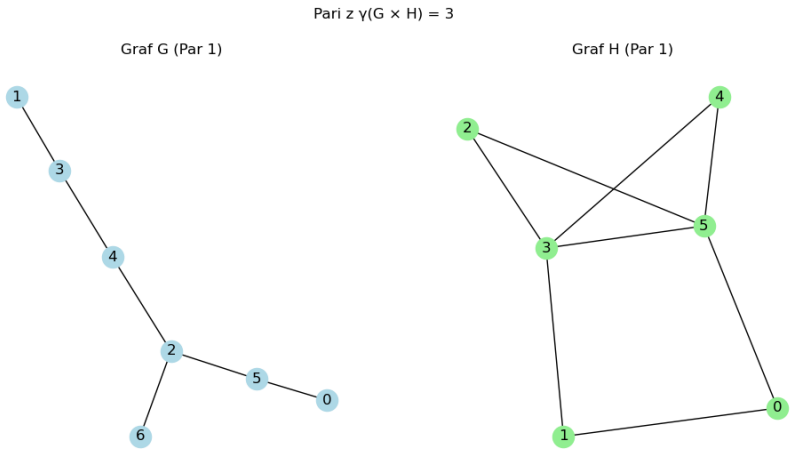



Figure 7: Primer 1

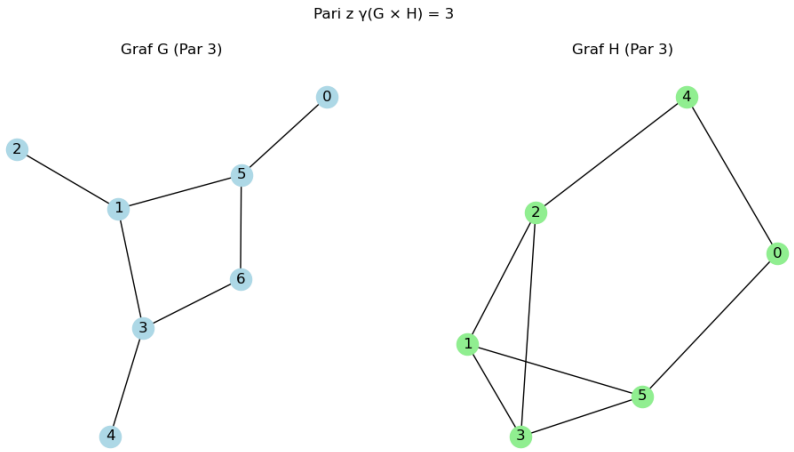



Figure 8: Primer 2

### 4.4.4 Kombinacija (7,7)

Za kombinacijo (7,7), ob naključnem izbiranju grafov $G$ in $H$, smo dobili 372 primernih parov izmed 1000 iteracij, ki ustrezajo $\gamma(G \diamond H) = 3$ .

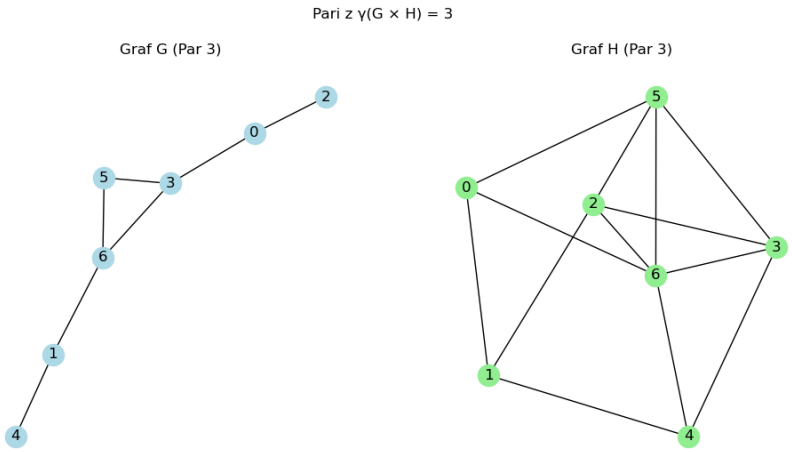



Figure 9: Primer 1

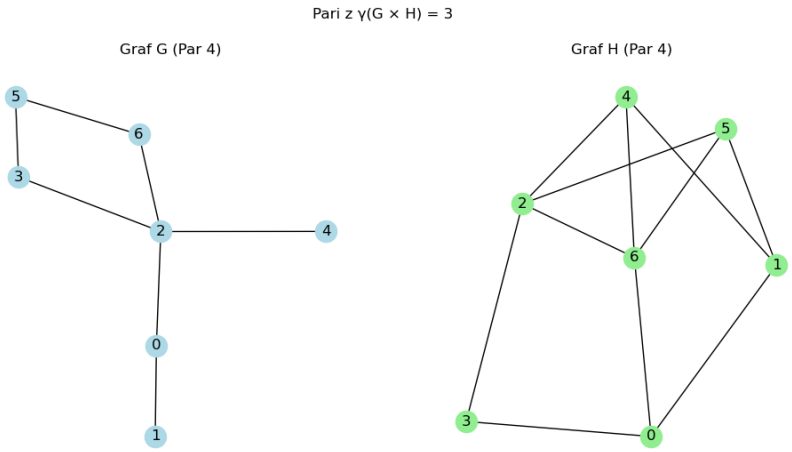



Figure 10: Primer 2

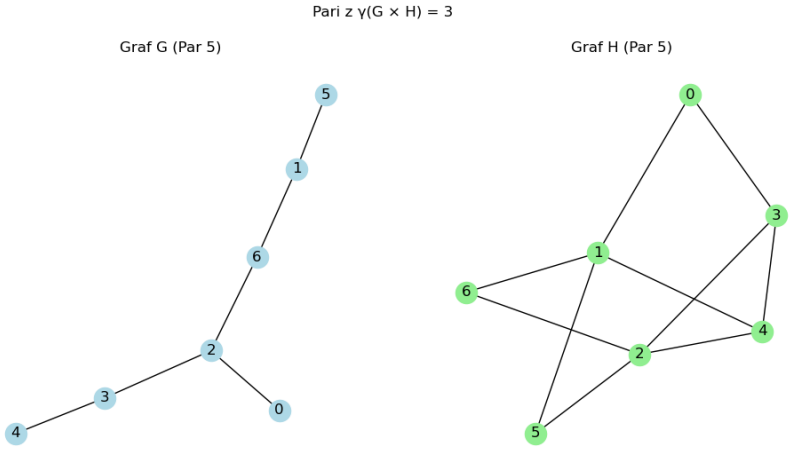



Figure 11: Primer 3

### 4.4.5 Kombinacija (7,8)

Za kombinacijo (7,8), ob naključnem izbiranju grafov $G$ in $H$, smo dobili 348 primernih parov izmed 1000 iteracij, ki ustrezajo $\gamma(G \diamond H) = 3$ .

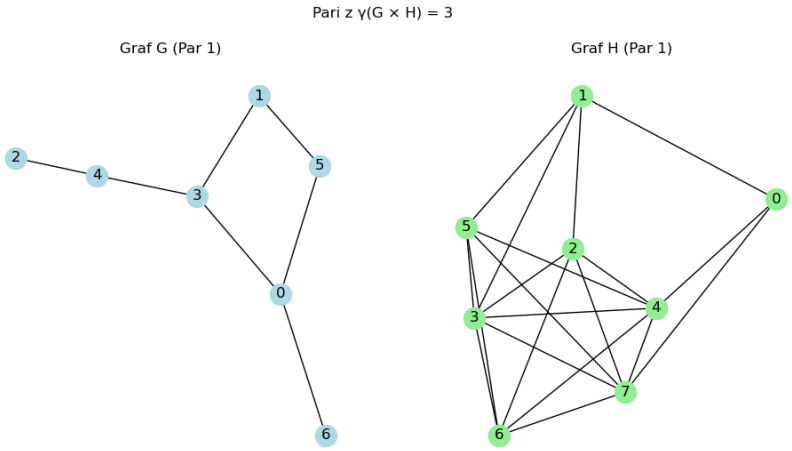



Figure 12: Primer 1

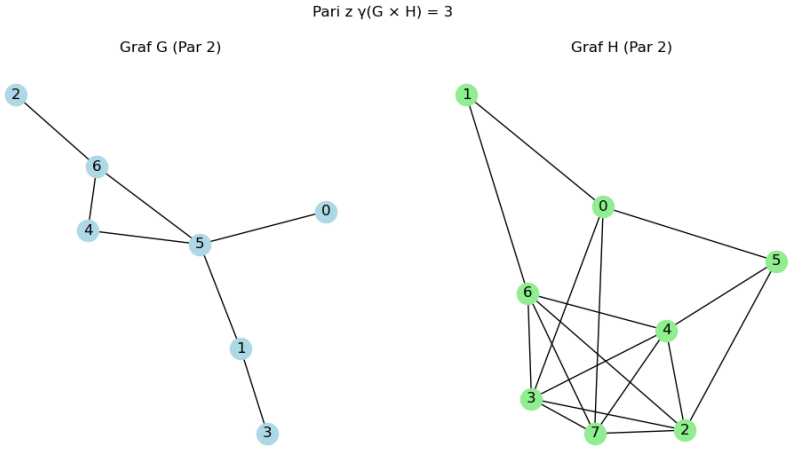



Figure 13: Primer 2

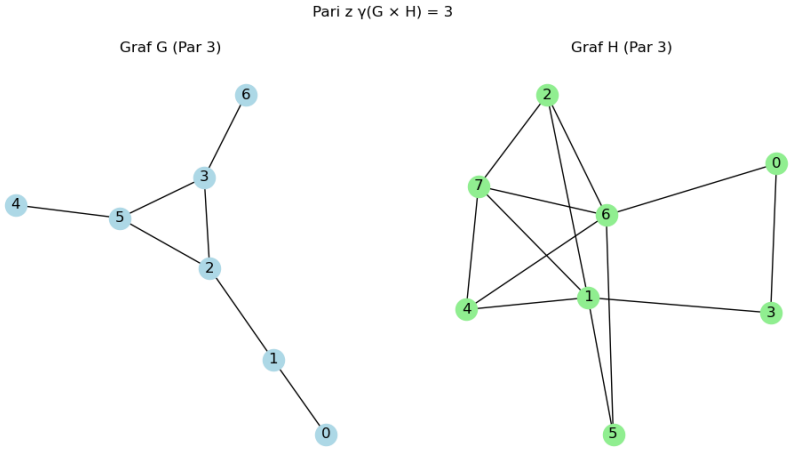



Figure 14: Primer 3

Enako smo izvedli še za kombinacije: (6,8), (8,6), (8,7) in (8,8) ter našli še kar nekaj grafov, ki ustrezajo neenakosti $\gamma(G \diamond H) \leq 3$. Kakšnih posebnih skupnih lastnosti med pari grafov, ki ustrezajo enakosti nisem opazila.

## 4.5 Zaključek

S pomočjo funkcij drugih skupin smo uspešno preverili neenakost $\gamma(G \diamond H) \leq 4$ za različne pare grafov. Rezultati kažejo, da v nobenem primeru dominacijsko število modularnega produkta ni preseglo vrednosti 4, kar potrjuje domnevo. Prav tako nismo našli primera, kjer bi bila dosežena zgornja meja, kar nakazuje možnost, da bi se dalo neenakost $\gamma(G \diamond H) \leq 4$ dodatno zožiti. S preverjanjem zožane neenakosti $\gamma(G \diamond H) \leq 3$ , smo še bolj prepričani, da lahko neenakost zožimo, saj tudi v tem primeru, nismo dobili para grafov, za katera ne bi veljala neenakost. Nadaljnje delo in raziskovanje bi lahko vključevalo povečanje števila iteracij ali analizo posebnih razredov grafov, pri katerih bi morda lahko dosegli $\gamma(G \diamond H) = 4$ ali dokončno potrdili $\gamma(G \diamond H) \leq 3$.