MMAD - Úkol 2

Filip Ditrich

Unicorn University, Prague, Czech Republic
28. dubna 2024

✓ Úloha 1 1
Úloha 2 5
✓ Úloha 3 6
Úloha 4 7
✓ Úloha 5 8
Úloha 6 9
✓ Úloha 7 10
✓ Úloha 8 11
✓ Úloha 9 13
✓ Úloha 10. 15
✓ Úloha 11. 16
✓ Úloha 12. 18
✓ Úloha 13. 20
✓ Úloha 14. 22
✓ Úloha programovací 1 24
✓ Úloha programovací 2 25

✓ Úloha 1 (2 body)

Pro následující grafy určete minimální stupeň $\delta(G)$, maximální stupeň $\Delta(G)$, skóre grafu a barevnost pro následující grafy:

a. Cesta P_n
b. Kružnice C_n
c. Úplný graf K_n
d. Úplný bipartitní graf $K_{m,n}$
e. Papův graf ukázaný níže

- **Minimální stupeň** $\delta(G)$ je nejmenší stupeň vrcholu v grafu G.
- **Maximální stupeň** $\Delta(G)$ je největší stupeň vrcholu v grafu G.
- **Skóre grafu** je součet stupňů všech vrcholů v grafu G.
- **Barevnost** $\chi(G)$ je minimální počet barev potřebných k obarvení vrcholů grafu G tak, aby žádné dva sousední vrcholy neměly stejnou barvu.

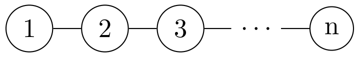
Řešení

a) Cesta P_n

Definice: Cesta P_n je graf, který má n vrcholů a $n-1$ hran.

- První a poslední vrchol mají stupeň 1, všechny ostatní vrcholy mají stupeň 2.
- Pro obarvení cesty P_n stačí vždy 2 barvy (jedna pro liché a druhá pro sudé vrcholy).

Poznámka: Bereme v potaz cestu P_n s $n \geq 2$ vrcholy.



Obrázek (1) — Cesta P_n

ODPOVĚĎ 1A

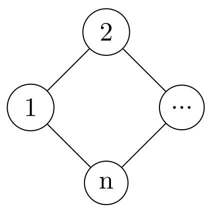
- $\delta(P_n) = 1$
- $\Delta(P_n) = 2$
- Skóre grafu $P_n = 2n - 2$
- Barevnost grafu $\chi(\mathrm{P}_n) = 2$

b) Kružnice C_n

Definice: Kružnice C_n má n vrcholů a každý vrchol je spojen s předchozím a následujícím vrcholem.

- Všechny vrcholy mají stupeň 2.
- Lichou kružnici C_n lze obarvit 3 barvami, sudou kružnici C_n lze pak obarvit 2 barvami střídavě.

Poznámka: Bereme v potaz kružnici C_n s $n \geq 3$ vrcholy.



Obrázek (2) — Kružnice C_n

ODPOVĚĎ 1B

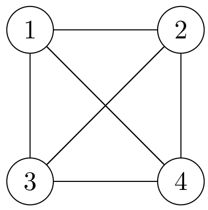
- $\delta(C_n) = 2$
- $\Delta(C_n) = 2$
- Skóre grafu $C_n = 2n$
- Barevnost grafu $\chi(\mathrm{C}_n) = \begin{cases} 2 & \text{pro sudé } n \\ 3 & \text{pro liché } n \end{cases}$

c) Úplný graf K_n

Definice: Úplný graf K_n má n vrcholů a každý vrchol je spojen s každým jiným vrcholem.

- Všechny vrcholy mají stupeň $n - 1$.
- Kvůli propojení všech vrcholů se všemi je nutné použít n barev.

Poznámka: Bereme v potaz úplný graf K_n s $n \geq 3$ vrcholy.



Obrázek (3) — Úplný graf K_4

ODPOVĚĎ 1C

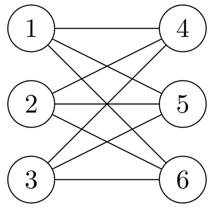
- $\delta(K_n) = n - 1$
- $\Delta(K_n) = n - 1$
- Skóre grafu $K_n = n(n - 1)$
- Barevnost grafu $\chi(\mathrm{K}_n) = n$

d) Úplný bipartitní graf $K_{m,n}$

Definice: Úplný bipartitní graf $K_{m,n}$ má m vrcholů v jedné partitě a n vrcholů v druhé partitě a každý vrchol z jedné partity je spojen s každým vrcholem z druhé partity.

- Všechny vrcholy z první partity mají stupeň n, všechny vrcholy z druhé partity mají stupeň m.
- Pro obarvení úplného bipartitního grafu $K_{m,n}$ stačí 2 barvy, jedna pro každou partitu.

Poznámka: Bereme v potaz úplný bipartitní graf $K_{m,n}$ s $m,n \geq 1$ vrcholy.



Obrázek (4) — Úplný bipartitní graf $K_{3,3}$

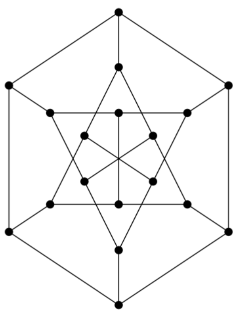
ODPOVĚĎ 1D

- $\delta(K_{m,n}) = m$
- $\Delta(K_{m,n}) = n$
- Skóre grafu $K_{m,n} = mn$
- Barevnost grafu $\chi(\mathrm{K}_{m,n}) = 2$

e) Papův graf

Definice: Na obrázku níže je zobrazen Papův graf s 18 vrcholy a 27 hranami, označme si jej jako $PG_{n,m}$, kde n je počet vrcholů a m je počet hran.

- Všechny vrcholy mají stupeň 3, jedná se tedy o kubický graf.
- Skóre grafu je $\{3, 3, \ldots, 3\}$, tedy $3 \cdot 18 = 54$.
- Každý vrchol je spojen s právě 3 dalšími vrcholy, tedy je možné graf obarvit 3 barvami.



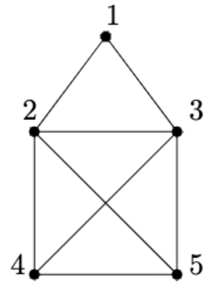
Obrázek (5) — Papův graf $PG_{18,27}$

ODPOVĚĎ 1E

- $\delta(PG_{18,27}) = 3$
- $\Delta(PG_{18,27}) = 3$
- Skóre grafu $PG_{18,27} = 54$
- Barevnost grafu $\chi(\mathrm{PG}_{18,27}) = 3$

Úloha 2 (2 body)

Máme graf $G = (V, E)$. Pro ukázku si představme následující graf H:



Obrázek (6) – Graf H

Definujme Laplacovu matici L jako:

$$L_{i,j} = \begin{cases} \deg(v_i) & \text{pro } i = j \\ -1 & \text{pro } i \neq j \text{ a } v_i \text{ je spojen s } v_j \\ 0 & \text{jinak} \end{cases}$$

Například pro ukázkový graf dostaneme Laplacovu matici:

$$L = \begin{pmatrix} 2 & -1 & -1 & 0 & 0 \\ 1 & 4 & -1 & -1 & -1 \\ -1 & -1 & 4 & -1 & -1 \\ 0 & -1 & -1 & 3 & -1 \\ 0 & -1 & -1 & -1 & 3 \end{pmatrix}$$

Mějme k-regulární graf $G = (V, E)$ velikosti $|V| = n$, tj. graf pro který víme, že $\deg(v_i) = k$ pro všechny vrcholy $v_i \in V$. Navíc víme, že vlastní čísla matice sousednosti jsou reálná v pořadí $\lambda_1(A) \geq \lambda_2(A) \geq \ldots \geq \lambda_n(A)$. Lze nějak obecně vyjádřit všechna vlastní čísla Laplacovy matice takového grafu?

Řešení

TODO

✓ Úloha 3 (2 body)

Určete minimální a maximální počet hran v grafu na n vrcholech s c komponentami.

Řešení

a) Postup pro nalezení minimálního počtu hran

- Izolované komponenty: Minimální počet hran nastává, když mají komponenty co nejméně hran. Extrémním případem je mít komponenty bez hran, tedy izolované vrcholy.
- Neizolované komponenty: Pro každou komponentu, která není jediný izolovaný vrchol, je minimální struktura vlastně strom. Strom s k vrcholy má $k - 1$ hran (minimum pro udržení grafu spojeného).
- Pokud je potřeba c komponent a předpokládáme že $c - 1$ komponent jsou jednotlivé izolované vrcholy a jedna komponenta obsahuje zbytek vrcholů, $n - (c - 1)$, tato poslední komponenta jako strom by měla $n - (c - 1) - 1 = n - c$ hran.
- Minimální počet hran je tedy $0 + (n - c) = n - c$.

Příklad na grafu s $n = 10$ vrcholy a $c = 3$ komponentami:

- Izolované komponenty: 2 izolované vrcholy, 1 komponenta s 8 vrcholy.
- Minimální počet hran: $10 - 3 = 7$.

ODPOVĚĎ 3A

Minimální počet v grafu na n vrcholech s c komponentami je $n - c$.

b) Postup pro nalezení maximálního počtu hran

- Maximální počet hran nastává, když každá komponenta je úplný graf (graf, kde jsou každé různé vrcholy spojeny jedinou hranou).
- Respektive postačí nám jedna komponenta jako úplný graf a zbytek komponent jako izolované vrcholy (tedy bez hran).
- Počet hran v úplném grafu na k vrcholech je $\binom{k}{2}$
- Náš úplný graf má $n - (c - 1)$ vrcholů, tedy počet vrcholů mínus počet ostatních izolovaných vrcholů (komponent).
- Maximální počet hran je tedy $\binom{n-(c-1)}{2}$.

Příklad na grafu s $n = 10$ vrcholy a $c = 3$ komponentami:

- Úplný graf s 8 $(10 - (3 - 1))$ vrcholy a 1 izolovaný vrchol.
- Maximální počet hran: $\binom{10-2}{2} = 28$.

ODPOVĚĎ 3B

Maximální počet v grafu na n vrcholech s c komponentami je $\binom{n-(c-1)}{2}$.

Úloha 4 (2 body)

Mějme následující funkci:

$$f(x, y) = 100(y - x^2)^2 + (1 - x)^2$$

Vypočtěte gradient $\nabla f(x)$ a Hessian $\nabla^2 f(x)$ a rozhodněte (a zdůvodněte) zda-li je v bodě $(1, 1)$ splněna 1. podmínka pro lokální minimizátor (nulovost gradientu) či zda-li je splněna i 2. podmínka pro Hessovu matici.

Řešení

TODO

✓ Úloha 5 (2 body)

Mějme množinu $\{1, 2, \ldots, n\}$. Určete, kolik je možné na této množině najít různých kružnic délky n? (jedná se tedy o počet průchodů, ale neorientovaného grafu).

Řešení

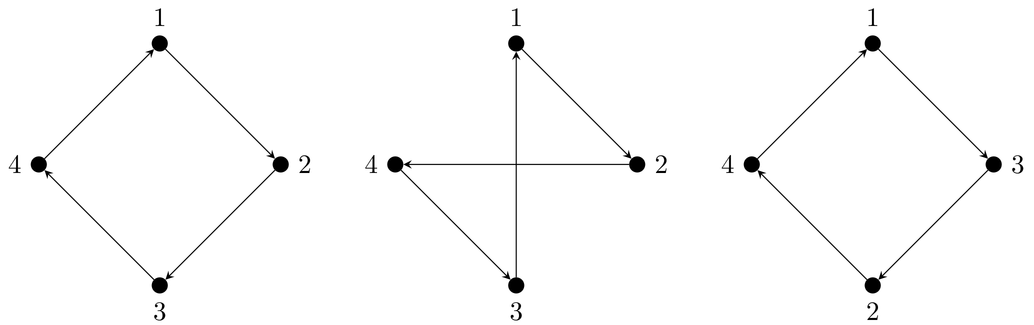
Problém můžeme řešit následovně:

- Krok 1: Seřadíme vrcholy kružnice do pořadí $1, 2, \ldots, n$. Takových sekvencí je $n!$.
- Krok 2: Zvolíme si jeden výchozí vrchol (symetrická rotace). Tím tedy získáme $(n-1)!$ unikátních sekvencí, ignorujeme-li rotace.
- Krok 3: Otočením sekvence získáme stejnou kružnici. Počet kružnic tedy musíme dělit dvěma (pro $n > 2$), protože každá sekvence a její zrcadlový obraz jsou v kružnici identické.
- Počet různých neorientovaných kružnic délky n (pro $n > 2$) je tedy: $\frac{(n-1)!}{2}$.

Ukázka: na příkladu množiny $\{1, 2, 3, 4\}$:
Dle definice výše víme, že počet různých neorientovaných kružnic délky 4 bude $\frac{(4-1)!}{2} = 3$.

A bude se jednat o tyto 3 kružnice:

1. $1 \to 2 \to 3 \to 4 \to 1$
2. $1 \to 2 \to 4 \to 3 \to 1$
3. $1 \to 3 \to 2 \to 4 \to 1$



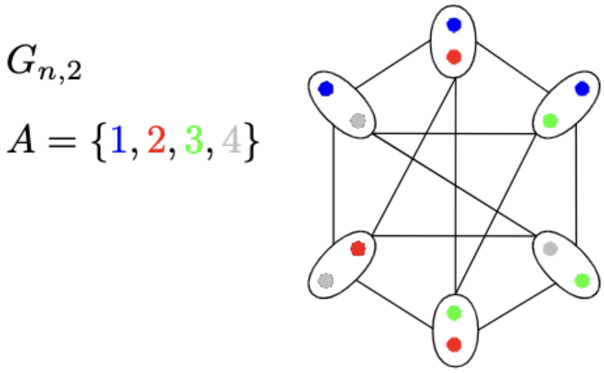
Obrázek (7) — Různé neorientované kružnice délky 4

ODPOVĚĎ

Pro množinu $\{1, 2, \ldots, n\}$ je možné najít $\frac{(n-1)!}{2}$ různých neorientovaných kružnic délky n.

Úloha 6 (2 body)

Mějme graf $G_{n,2} = (V, E)$ definovaný následovně. Množina vrcholů jsou všechny podmnožiny množiny $A = \{1, 2, \ldots, n\}$ o velikosti 2, tedy například $v_1 = \{1, 2\}, v_2 = \{2, 3\}, \ldots$. Hrany spojují ty vrcholy $v_i = \{a, b\}, v_2 = \{c, d\}$, které sdílí právě jeden prvek, tj. $a = b \neq c = d$. Příklad takového grafu je vidět na následujícím obrázku.



Obrázek (8) — Graf $G_{n,2}$

Pro takový obecný graf $G_{n,2}$ určete jaký bude jeho minimální a maximální stupeň vrcholu vyjádřeno jako funkce n . Také určete počet hran tohoto grafu, opět jako funkci n.

Řešení

TODO

✓ Úloha 7 (2 body)

Zdůvodněte, proč každá hrana vrcholově 2-souvislého grafu musí ležet na kružnici.

Řešení

Vrcholově 2-souvislý graf je graf, který zůstane souvislý i po odebrání libovolního vrcholu, ale po odebrání alespoň dvou vrcholů se může rozpadnout na více komponent vzájemně nespojených hranami, tedy nesouvislých.

- Ve vrcholově 2-souvislém grafu musí existovat alespoň dva nezávislé průchody mezi libovolnými dvěma vrcholy.
- Uvažujme libovolnou hranu uv ve vrcholově 2-souvislém grafu.
- Dle vlastností vrcholové 2-souvislosti existuje další cesta z vrcholu u do vrcholu v, která neobsahuje hranu uv.
- Existence této alternativní cesty mezi vrcholy u a v, spolu s hranou uv, tvoří kružnici.
- Kružnice je vytvořena cestou z vrcholu u do vrcholu v přes alternativní cestu a návratem do vrcholu u pomocí hrany uv.

ODPOVĚĎ

Každá hrana ve vrcholově 2-souvislém grafu musí ležet na kružnici, protože definice vrcholové 2-souvislosti zaručuje přítomnost alternativních cest mezi vrcholy.

✓ Úloha 8 (2 body)

Určete vrcholový a hranový stupeň grafu, neboli $\alpha(G)$ a $\kappa(G)$, pro následující grafy:

a. Cesta P_n

b. Kružnice C_n

c. Úplný graf K_n

d. Úplný bipartitní graf $K_{m,n}$

Řešení

- **Vrcholový stupeň** $\alpha(G)$ je minimální počet vrcholů, které je třeba odebrat, aby se graf rozpadl na více komponent.
- **Hranový stupeň** $\kappa(G)$ je minimální počet hran, které je třeba odebrat, aby se graf rozpadl na více komponent.

Cesta P_n

- Odebráním jakéhokoliv vnitřního vrcholu se cesta rozpadne na dvě komponenty. Vrcholový stupeň je tedy $\alpha(P_n) = 1$.
- Odebráním jakékoliv hrany se cesta rozpadne na dvě komponenty. Hranový stupeň je tedy $\kappa(P_n) = 1$.

Poznámka: Bereme v potaz cestu P_n s $n \geq 2$ vrcholy.

ODPOVĚĎ 8A

Pro cestu P_n platí $\alpha(P_n) = 1$ a $\kappa(P_n) = 1$.

Kružnice C_n

- Odebráním jakéhokoliv vrcholu se z kružnice stane cesta, z tvrzení výše víme že $\alpha(P_n) = 1$. Vrcholový stupeň je tedy $\alpha(C_n) = \alpha(P_{n-1}) + 1$.
- Obdobně odebráním jakýchkoliv 2 sousedních hran se kružnice rozpadne na cestu a jeden izolovaný vrchol. Hranový stupeň je tedy $\kappa(C_n) = 2$.

Poznámka: Bereme v potaz kružnici C_n s $n \geq 3$ vrcholy.

ODPOVĚĎ 8B

Pro kružnici C_n platí $\alpha(C_n) = 2$ a $\kappa(C_n) = 2$.

Úplný graf K_n

- Postupným odebíráním vrcholů zůstává graf stále souvislý, dokud neodebereme až $n-1$ vrcholů, pak zůstává pouze izolovaný vrchol. Vrcholový stupeň je tedy $\alpha(K_n) = n - 1$.
- Odebráním všech hran jednoho vrcholu, který má stupeň $n-1$, se graf rozpadne jednu jednu souvislou komponentu a izolovaný vrchol. Hranový stupeň je tedy $\kappa(K_n) = n - 1$.

Poznámka: Bereme v potaz úplný graf K_n s $n \geq 2$ vrcholy.

ODPOVĚĎ 8C

Pro úplný graf K_n platí $\alpha(K_n) = n - 1$ a $\kappa(K_n) = n - 1$.

Úplný bipartitní graf $K_{m,n}$

- Odebráním všech vrcholů jedné partity se graf rozpadne na několik izolovaných vrcholů (jelikož každý vrchol z jedné partity je spojen s každým vrcholem z druhé partity, ale nikoliv s vrcholem ze stejné partity). Lze tedy říci, že vrcholový stupeň je $\alpha(K_{m,n}) = \min(m, n)$.
- Odebráním všech hran spojujících vrcholy jedné partity se graf rozpadne na souvislou komponentu a několik izolovaných vrcholů. Pokud vybereme vrchol z větši parity, musíme odebrat pouze tolik hran, kolik vrcholů má menší parita, tedy opět hranový stupeň je tedy $\kappa(K_{m,n}) = \min(m, n)$.

ODPOVĚĎ 8D

Pro úplný bipartitní graf $K_{m,n}$ platí $\alpha(K_{m,n}) = \min(m, n)$ a $\kappa(K_{m,n}) = \min(m, n)$.

✓ Úloha 9 (2 body)

Vezměmě si grafy typu strom o fixní velikosti n. Rozhodněte a nakreslete, jaký strom o velikosti n má:

a. Největší hodnotu nezávislosti $\alpha(G)$

b. Nejmenší hodnotu nezávislosti $\alpha(G)$

c. Největší hodnotu vrcholového pokrytí $\beta(G)$

d. Nejmenší hodnotu vrcholového pokrytí $\beta(G)$

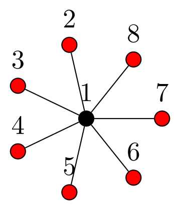
Řešení

- **Strom** je graf, který je souvislý a neobsahuje cykly (kružnice).
- **Nezávislost** $\alpha(G)$ je největší podmnožina vrcholů grafu G, která není spojená hranou.
- **Vrcholové pokrytí** $\beta(G)$ je nejmenší podmnožina vrcholů grafu G, která pokryje všechny hrany.

a) Největší hodnota nezávislosti $\alpha(G)$

K nalezení stromu s největší hodnotou nezávislosti $\alpha(G)$ je třeba nalézt strom, který má co nejvíce listů (vrcholů stupně 1). Nezávislá množina pak bude tvořena všemi listy stromu.

Takový strom může být například **hvězdicový strom**, pro který platí, že má jeden vrchol stupně $n-1$ a $n-1$ vrcholů stupně 1:



Obrázek (9) — Hvězdicový strom o velikosti $n = 8$

Dále si na tomto grafu můžeme povšimnout, že jeho vrcholové pokrytí $\beta(G)$ bude tvořeno pouze kořenem stromu, tedy $\beta(G) = 1$, k tomu se ale dostaneme v dalších bodech.

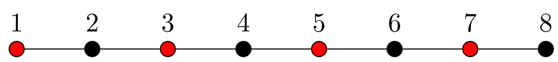
ODPOVĚĎ 9A

Největší hodnota nezávislosti $\alpha(G)$ je docílena na hvězdicovém stromu o velikosti n, kde $\alpha(G) = n - 1$.

b) Nejmenší hodnota nezávislosti $\alpha(G)$

Pro nalezení stromu s nejmenší hodnotou nezávislosti $\alpha(G)$ je třeba nalézt strom, který bude mít co nejvíce hran mezi vrcholy. V takovém stromu pak bude nezávislá množina tvořena každým druhým vrcholem.

Takový strom je například **cesta** P_n, kde nezávislá množina bude tvořena každým druhým vrcholem:



Obrázek (10) — Cesta o velikosti $n = 8$

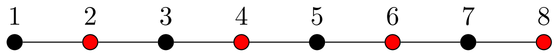
Dále si na tomto grafu můžeme povšimnout, že jeho vrcholové pokrytí $\beta(G)$ bude také tvořeno každým druhým vrcholem, tedy $\beta(G) = \frac{n}{2}$.

ODPOVĚĎ 9B

Nejmenší hodnota nezávislosti $\alpha(G)$ je docílena na cestě o velikosti n, kde $\alpha(G) = \frac{n}{2}$.

c) Největší hodnota vrcholového pokrytí $\beta(G)$

Jak jsme si již ukázali v předchozích bodech, největší hodnota vrcholového pokrytí $\beta(G)$ bude docílena na cestě P_n, kde $\beta(G) = \frac{n}{2}$.



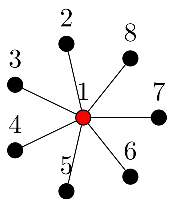
Obrázek (11) — Cesta o velikosti $n = 8$

ODPOVĚĎ 9C

Největší hodnota vrcholového pokrytí $\beta(G)$ je docílena na cestě o velikosti n, kde $\beta(G) = \frac{n}{2}$.

d) Nejmenší hodnota vrcholového pokrytí $\beta(G)$

Naopak opět se vracíme k hvězdicovému stromu, kde jsme dále zjistili, že nejmenší hodnota vrcholového pokrytí $\beta(G)$ bude docílena na hvězdicovém stromu, kde $\beta(G) = 1$ a to díky kořeni stromu, který pokryje všechny hrany.



Obrázek (12) — Hvězdicový strom o velikosti $n = 8$

ODPOVĚĎ 9D

Nejmenší hodnota vrcholového pokrytí $\beta(G)$ je docílena na hvězdicovém stromu o velikosti n, kde $\beta(G) = 1$.

✓ Úloha 10 (2 body)

Ukažte proč pro každý kubický graf G, t.j. takový, že všechny stupně vrcholů jsou 3, platí, že stupeň vrcholové i hranové souvislosti se rovnají, tj. $\alpha(G) = \kappa(G)$. *Hint: Pokuste se rozebrat případy pro různé vrcholové stupně souvislosti.*

Řešení

Víme, že pro každý graf $G = (V, E)$ platí *Whitneyho nerovnost*:

$$\kappa(G) \leq \alpha(G) \leq \delta(G)$$

kde $\delta(G)$ je minimální stupeň vrcholu v grafu G. Pro kubický graf tedy platí $\delta(G) = 3$.

Odebírání hran a vrcholů v kubickém grafu

- Odebráním hrany se sníží stupeň dvou vrcholů z 3 na 2, ale graf zůstává souvislý.
- Odebráním vrcholu se sníží stupeň tří hran z 3 na 2, což může vést k rozpadu grafu.
- Zatím můžeme pozorovat, že odebrání vrcholu může být více kritické než odebrání hrany.

Případy pro různé vrcholové stupně souvislosti

- **Případ 1:** Pokud $\alpha(G) = 1$, graf obsahuje most, který po odebrání rozdělí graf na dvě komponenty. Odebráním jednoho vrcholu se graf rozpadne, tedy $\kappa(G) = 1$ také.
- **Případ 2:** Pokud $\alpha(G) > 1$, odebrání jednoho vrcholu nevede k rozpadu grafu, což naznačuje vyšší odolnost, tedy $\kappa(G)$ může být 2 nebo 3.
  - Vzhledem k tomu, že kubické grafy mají vrcholy a hrany těsně propojeny kvůli jejich uniformnímu stupni, odebrání minimálního počtu vrcholů obvykle znamená odebrání i minimálního počtu hran.
- **Případ 3:** Pro $\alpha(G) = 2$ nebo $\alpha(G) = 3$ bude kubický graf vyžadovat podobně minimální počet odebraných vrcholů k rozdělení, což znamená, že $\kappa(G)$ bude obvykle odpovídat $\alpha(G)$.

ODPOVĚĎ

Díky stejnému stupni vrcholů v kubickém grafu G platí, že stupeň vrcholové i hranové souvislosti jsou stejné, tj. $\alpha(G) = \kappa(G)$.

✓ Úloha 11 (2 body)

Pokuste se navrhnout Turingův stroj pro rozpoznání, že neorientovaný graf má izolovaný vrchol.
Hint: Graf uložte na pásku jako matici sousednosti (nezapomeňte na oddělovače řádků) a v ní pomocí pravidel nalezněte takový vrchol.

Řešení

Tento Turingův stroj kontroluje, zda je v neorientovaném grafu izolovaný vrchol, který je reprezentován jako řádek v matici sousednosti se samými 0.

Příklad reprezentace grafu na pásce

- Graf je reprezentován jako matice sousednosti, kde $A_{ij} = 1$ pokud existuje hrana mezi vrcholy i a j.
- Abeceda obsahuje znaky $1, 0$ a speciální symbol $\hookleftarrow$ jako oddělovač řádků společně se standardními znaky $\triangleright$ pro označení počátečního stavu a $\star$ pro prázdné pole.
- Příklad matice, se kterou budeme pracovat: 001 $\hookleftarrow$ 011 $\hookleftarrow$ 000 $\hookleftarrow$.
- Tu lze reprezentovat na pásce Turingova stroje následovně:

$\cdots$	$\star$	$\star$	$\star$	$\triangleright$	0	0	1	$\hookleftarrow$	0	1	1	$\hookleftarrow$	0	0	0	$\hookleftarrow$	$\star$	$\star$	$\star$	$\cdots$

Obrázek (13) — Reprezentace grafu na pásce

Konfigurace Turingova stroje

- **Množina stavů** K: Obsahuje počáteční stav s_0, stavy pro čtení řádků s_r, stav pro kontrolu řádku s_c, stav přijetí s_{ano} a stav odmítnutí s_{ne}.
- **Vstupní abeceda** Σ: $\{0, 1, \star, \hookleftarrow, \triangleright\}$.
- **Abeceda pásky** Γ: $\{0, 1, \hookleftarrow, \triangleright, X\}$, kde X označuje již zkontrolované prvky.
- **Přechodová funkce** δ.
- **Počáteční stav**: s_0.
- **Koncové stavy**: $s_{\text{ano}}, s_{\text{ne}}$.

Přechodová funkce

Tabulka přechodů pro Turingův stroj je následující:

Stav	Čtení	Zápis	Pohyb	Další stav
s_0	$\triangleright$	$\triangleright$	$\rightarrow$	s_r

Stav	Čtení	Zápis	Pohyb	Další stav
s_r	0	0	$\rightarrow$	s_r
s_r	1	X	$\rightarrow$	s_c
s_r	$\hookleftarrow$	$\hookleftarrow$	$\rightarrow$	s_{ano}
s_r	$\star$	$\star$	$\rightarrow$	s_{ano}

Stav	**Čtení**	**Zápis**	**Pohyb**	**Další stav**
s_c	1	X	$\rightarrow$	s_{ne}
s_c	0	0	$\rightarrow$	s_r
s_c	$\hookleftarrow$	$\hookleftarrow$	$\rightarrow$	s_r
s_c	$\star$	$\star$	$\rightarrow$	s_r

Popis přechodů

- Stroj začíná ve stavu s_0 a přejde do stavu s_r po přečtení počátečního stavu $\triangleright$.
- Ve stavu s_r stroj pokračuje ve stavu s_r po přečtení 0. Po přečtení 1 přejde do stavu s_c.
- Stroj přejde do stavu s_{ano} po přečtení oddělovače řádku $\hookleftarrow$ nebo $\star$.
- Ve stavu s_c stroj přejde do stavu s_{ne} po přečtení 1. Po přečtení 0, X nebo oddělovače řádku $\hookleftarrow$ přejde zpět do stavu s_r.
- Stroj přijme vstup, pokud nalezne řádek s samými 0 nebo pokud dojde na konec pásky ve stavu s_r po přečtení samých 0.

Ukázka běhu Turingova stroje

Vstup: $\triangleright 001 \hookleftarrow 011 \hookleftarrow 000 \hookleftarrow$ – viz obrázek 13.

1. Stroj začíná ve stavu s_0 s hlavou nad symbolem $\triangleright$.
2. Přečte $\triangleright$, zapíše $\triangleright$, posune se doprava a přejde do stavu s_r.
3. Hlava je nyní nad prvním 0 za oddělovačem řádku. Přečte 0, zapíše 0, posune se doprava a zůstává ve stavu s_r.
4. Přečte druhé 0, zapíše 0, posune se doprava a zůstává ve stavu s_r.
5. Přečte 1. Protože stroj byl navržen tak, aby odmítl, pokud najde 1 v řádku, přejde do stavu s_c.
6. Přečte oddělovač řádku $\hookleftarrow$, zapíše $\hookleftarrow$, posune se doprava. Protože byla v tomto řádku 1, stroj nezastaví ani nepřijme, ale pokračuje ve stavu s_r pro kontrolu dalšího řádku.
7. Přečte další 0, zapíše 0, posune se doprava a zůstává ve stavu s_r.
8. Přečte 1, přejde do stavu s_c, zapíše 1 a posune se doprava.
9. Přečte druhé 1, zapíše 1, posune se doprava a zůstává ve stavu s_c.
10. Přečte oddělovač řádku $\hookleftarrow$, zapíše $\hookleftarrow$, posune se doprava a přejde zpět do stavu s_r pro kontrolu dalšího řádku.
11. Přečte 0, zapíše 0, posune se doprava a zůstává ve stavu s_r.
12. Přečte druhé 0, zapíše 0, posune se doprava a zůstává ve stavu s_r.
13. Přečte třetí 0. Nyní je to klíčové, protože je to poslední číslice v řádku a dalším symbolem je $\hookleftarrow$. Stroj se musí přepnout do stavu přijetí, protože našel řádek pouze s 0, což naznačuje izolovaný vrchol.
14. Přečte $\hookleftarrow$, zapíše $\hookleftarrow$ a přejde do stavu s_{ano} podle poslední aktualizace v přechodové tabulce. To je proto, že detekoval řádek bez 1, což naznačuje izolovaný vrchol.
15. Stroj zastaví ve stavu s_{ano}, když úspěšně nalezl izolovaný vrchol.

✓ Úloha 12 (2 body)

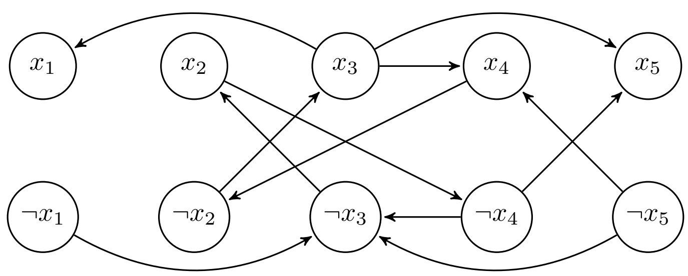
Využijte vysvětlení proč platí polynomialita 2-SAT a nakreslete graf odpovídající následující formuli a otestujte a případně ukažte, zda-li je splněna. *Pozn.: Graf na kreslete, i když budete schopni splnitelnost rozhodnout jinak.*

$$f(x_1, x_2, x_3, x_4, x_5) = (x_1 \vee \overline{x_3}) \wedge (x_2 \vee x_3) \wedge (\overline{x_3} \vee x_4) \wedge (\overline{x_2} \vee \overline{x_4}) \wedge (x_4 \vee x_5) \wedge (\overline{x_3} \vee x_5)$$

Řešení

Nejprve sestavíme graf na literálech a klauzuích, tak že:

- Vrcholy: $V = \{x_1, \ldots, x_5, \neg x_1, \ldots, \neg x_5\}$
- Hrany: Pro $\forall$ klauzuli $(a \vee b)$ přidáme hrany mezi $(\neg a, b)$ a $(\neg b, a)$



Obrázek (14) – Graf sestaený z literálů a klauzulí

Následně nalezneme silně souvislé komponenty (kvasikomponenty) pomocí Kosarajova algoritmu (dvojitý průchod DFS). Z tohoto algoritmu jsme nalezli 4 kvasikomponenty:

1. $G_1 = \{\neg x_1\}$
2. $G_2 = \{\neg x_5, x_4, \neg x_2, x_3\}$
3. $G_3 = \{\neg x_3, x_2, \neg x_4, x_5\}$
4. $G_4 = \{x_1\}$

Nalezneme průchody mezi kvasikomponentami:

1. $G_1 \to G_3$
2. $G_2 \to G_3$ a $G_2 \to G_4$

a acyklicky je očíslujeme:

1. $c(G_1) = 1$
2. $c(G_2) = 2$
3. $c(G_4) = 3$
4. $c(G_3) = 4$

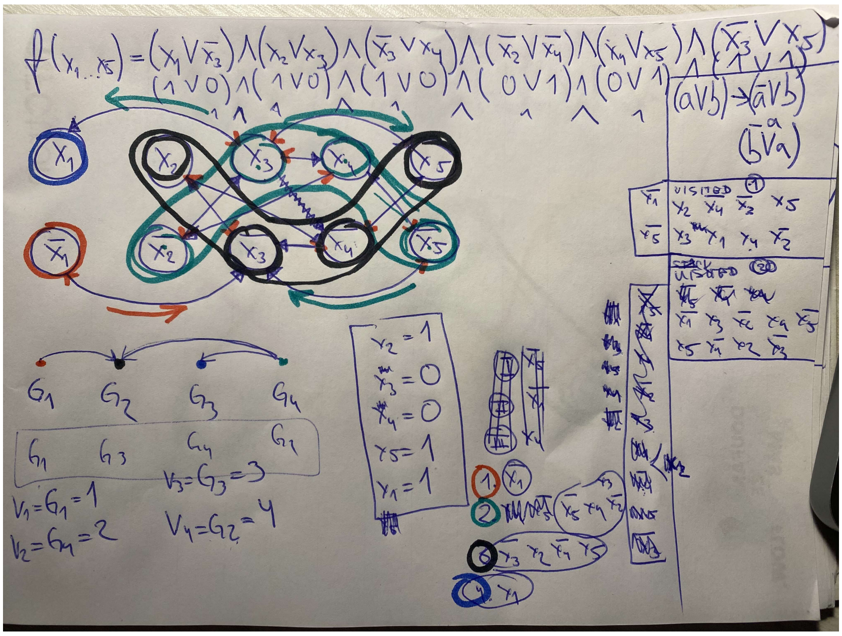
poté jdeme od nejvyšší komponenty G_2 a sbíráme informace o literálech z grafu:

- v G_3 jsou $x_2, x_5, \neg x_3, \neg x_4$ a tedy $x_2 = x_5 = 1$ a $x_3 = x_4 = 0$
- v G_4 je $x_1 = 1$ a tedy $x_1 = 1$

Pak pro tyto hodnoty ověříme formuli:

$$f(x_1, x_2, x_3, x_4, x_5) = (1 \vee 0) \wedge (1 \vee 0) \wedge (1 \vee 0) \wedge (0 \vee 1) \wedge (0 \vee 1) \wedge (1 \vee 1) = 1$$

Podrobnější postup je vidět na obrázku 15 níže:



Obrázek (15) — Postup nalezení silně souvislých komponent a ověření formule

ODPOVĚĎ

Formule je splněna s hodnotami $x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 0, x_5 = 1$.

✓ Úloha 13 (2 body)

Na základě vysvětlení převodu SAT na IND nakreslete graf odpovídající následující formuli a otestujte splnitelnost formule nalezením nezávislé množiny. *Pozn.: Graf nakreslete a zhodnoťte, i když budete schopni splnitelnost rozhodnout jinak.*

$$f(x_1, x_2, x_3, x_4, x_5) = (x_2 \vee \overline{x_3} \vee x_4) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4) \wedge (x_1 \vee \overline{x_2} \vee x_3 \vee \overline{x_4}) \wedge (x_4 \vee x_5)$$

Řešení

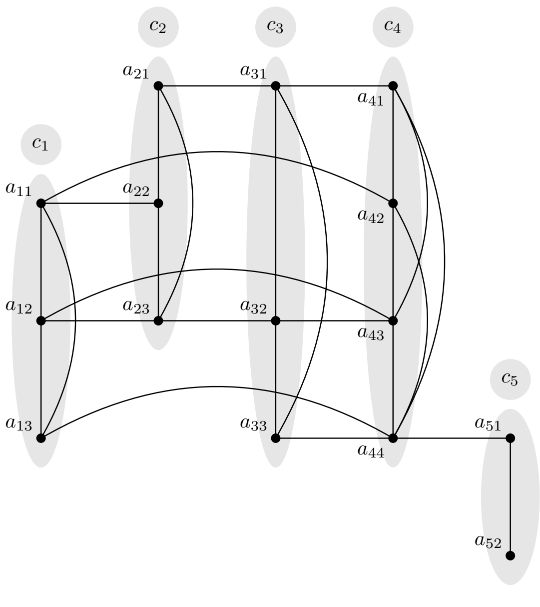
Nejprve sestavíme graf na literálech a klauzích, tak že:

- Vrcholy budou indexované jako literály
- A hrany budou spojovat:
  - vrcholy odpovídající literálům stejné klauzule
  - vrcholy odpovídající stejné proměnné, ale negaci

Formuli si nyní rozepíšeme do literálů a klauzulí:

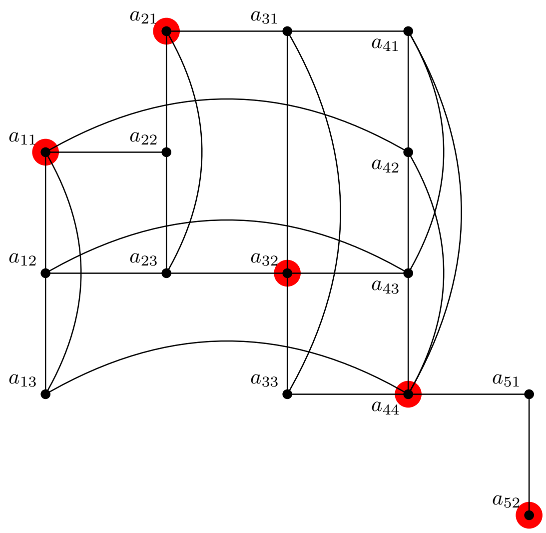
$$f(x_1, x_2, \ldots, x_5) = \underbrace{(a_{11} \vee a_{12} \vee a_{13})}_{c_1} \wedge \underbrace{(a_{21} \vee a_{22} \vee a_{23})}_{c_2} \wedge \underbrace{(a_{31} \vee a_{32} \vee a_{33})}_{c_3} \wedge \underbrace{(a_{41} \vee a_{42} \vee a_{43} \vee a_{44})}_{c_4} \wedge \underbrace{(a_{51} \vee a_{52})}_{c_5}$$

Pro tu nyní sestrojíme graf:



Obrázek (16) – Graf sestaený z literálů a klauzulí

Nyní musíme v každé klauzuli nalézt literál, který bude roven 1, a tím pádem bude tato klauzule splněna. Tedy musíme najít nezávislou množinu v grafu:



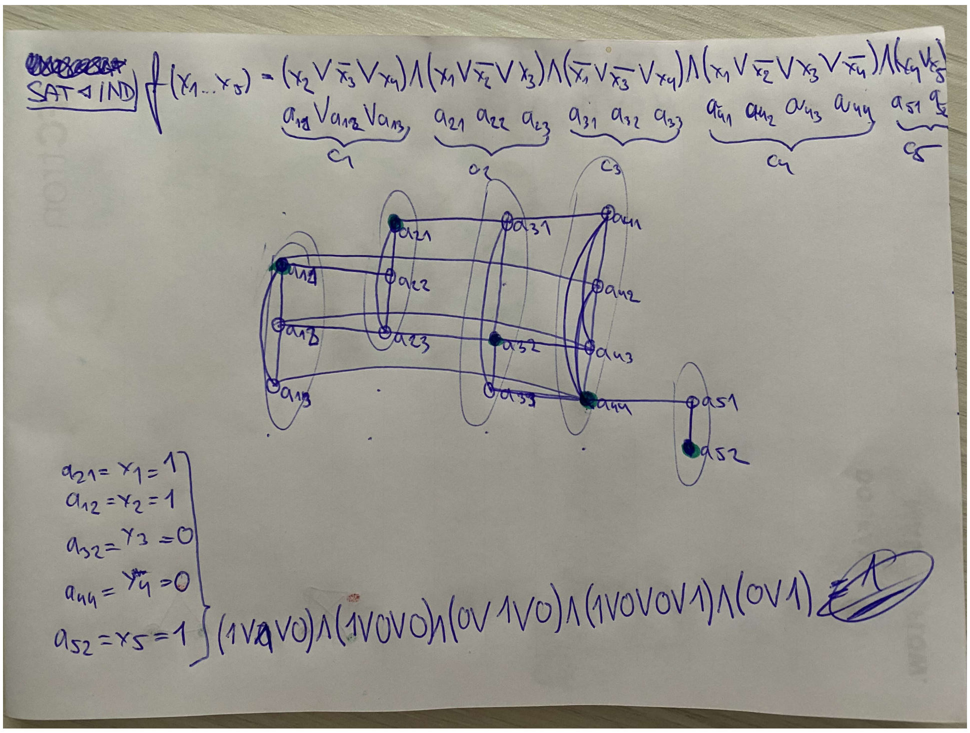
Obrázek (17) — Množina nezávislých vrcholů v grafu

Pro tuto množinu $a_{12}, a_{21}, a_{32}, a_{44}, a_{52}$ zvolíme literály rovny jedné:

$$f(x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 0, x_5 = 1) = (1 \vee 1 \vee 0) \wedge (1 \vee 0 \vee 0) \wedge (0 \vee 1 \vee 0) \wedge (1 \vee 0 \vee 0 \vee 1) \wedge (0 \vee 1) = 1$$

Formule je rovna 1 a tedy je splněna se zvolenými hodnotami literálů.

Podrobnější postup je vidět na obrázku 18 níže:



Obrázek (18) — Postup nalezení nezávislé množiny a ověření formule

ODPOVĚĎ

Formule je splněna s hodnotami $x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 0, x_5 = 1$.

✓ Úloha 14 (2 body)

Na základě vysvětlení převodu 3-SAT na 3-COL nakreslete graf odpovídající následující formuli a otestujte barevnost grafu.

$$f(x_1, x_2, x_3, x_4, x_5) = (x_1 \vee x_2 \vee \overline{x_4}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3} \vee x_4) \wedge (x_1 \vee \overline{x_2} \vee x_3)$$

Řešení

Nejprve si lze všimnout:

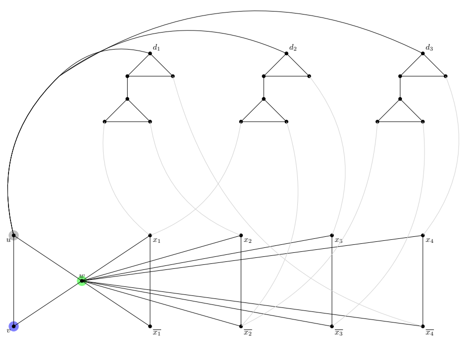
- Ve formuli není zahrnuta proměnná x_5, a tedy ji můžeme ignorovat.
- Klauzule $(x_1 \vee \overline{x_2} \vee x_3)$ je duplicitní a můžeme tedy jednu z nich odstranit.
- *Poznámka: Originální zadání jsem stáhnul a pročítal 5× a opravdu tak formule je a nebyla to chyba při přepisu. Budeme tedy dále předpokládat, že se jedná o záměr.*

Dále tedy pracujeme s upravenou formulí:

$$f(x_1, x_2, x_3, x_4) = \underbrace{(x_1 \vee x_2 \vee \overline{x_4})}_{c_1} \wedge \underbrace{(x_1 \vee \overline{x_2} \vee x_3)}_{c_2} \wedge \underbrace{(\overline{x_2} \vee \overline{x_3} \vee x_4)}_{c_3}$$

Nejprve si sestrojíme graf následovně:

- Vrcholy pro každou proměnnou x_i vytvoříme dvojici vrcholů x_1,$\overline{x_i}$.
- Přidáme vrcholy u, v, w
- Přidáme hrany $\{x_i$,$\overline{x_i}\}$ pro každou proměnnou
- Přidáme hrany $\{w$,$x_i\}$ a $\{w$,$\overline{x_i}\}$ pro každou proměnnou
- Přidáme hrany mezi vrcholy u, v, w
- Pro každou klauzuli $c1, c2, c3$ přidáme pomocný graf G (sloužící jako logická brána) s vrcholy d_i a a_i, b_i, c_i
- Vrcholy a_i, b_i, c_i ztotožníme s vrcholy x_i z klauzule
- Všechny vrcholy d_i spojíme hranou s vrcholem u
- A inicalizujeme obarvení vrcholů u, v, w na 3 barvy:
  - u - Barva 0 (šedá)
  - v - Barva 1 (modrá)
  - w - Barva 2 (zelená)

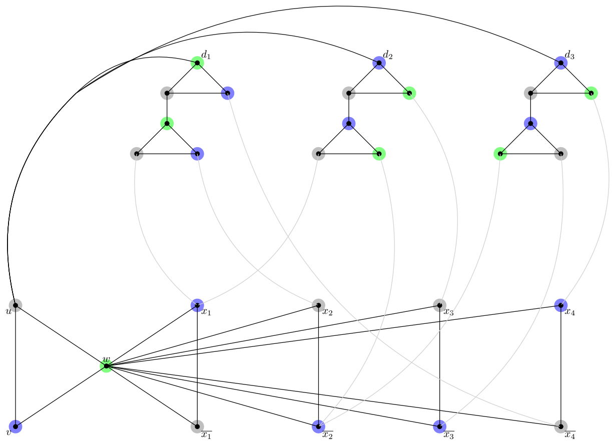


Obrázek (19) — Sestrojený graf pro formuli

Poté se pokusíme obarvit graf, tak že:

- Vrcholy d_i musí být obarveny tak, aby byly různé od barvy 0 (šedé).
- Vrcholy x_i a $\overline{x_i}$ musí být obarveny tak, aby od sebe různé, barvami 0 a 1
- Pokud v každé klauzuli existují vrcholy d_i takové, že jsou různé od barvy 0, pak je formule splnitelná a graf je 3-obarvitelný.

Takové obarvení může být násludující:



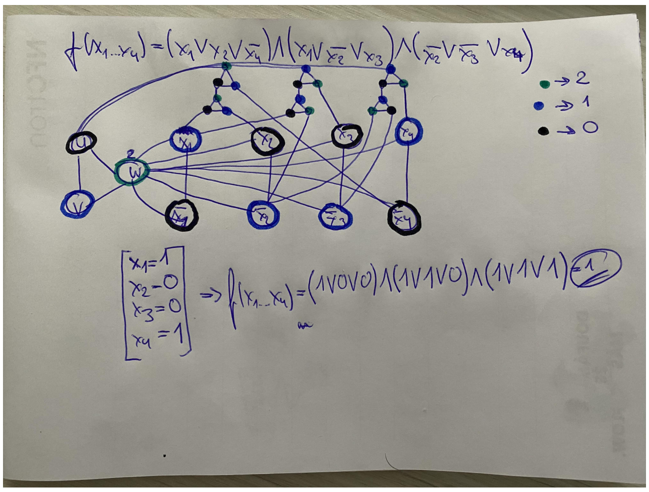
Obrázek (20) — Obarvený graf pro formuli

Dosazením do formule zjistíme, že všechny klauzule jsou splněny:

$$f(x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 1) = (1 \vee 0 \vee 0) \wedge (1 \vee 1 \vee 0) \wedge (1 \vee 1 \vee 1) = 1$$

Obarvení jsme v grafu našli a tedy je formule splnitelná a graf je 3-obarvitelný.

Podrobnější postup je vidět na obrázku 21 níže:



Obrázek (21) — Postup obarvení grafu

ODPOVĚĎ

Formule je splněna a graf je 3-obarvitelný pro $x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 1$.

✓ Úloha programovací 1 (4 body)

Naprogramujte algoritmus pro testování isomorfismu grafů hrubou silou a otestujte to na pár příkladech grafů s využitím knihovní funkce pro testování isomorfismu.

Řešení

Isomorfismus lze testovat hrubou silou, kde se všechny možné permutace uzlů grafu G porovnají s uzly grafu H.

Vytvoříme funkci `isomorphism(G, H)`, která otestuje isomorfismus grafů G a H následovně:

- Pokud mají grafy různý počet uzlů, vrátí `False`.
- Pro všechny permutace uzlů grafu G:
  - Vytvoří mapování uzlů grafu G na uzly grafu H.
  - Pokud všechny uzly grafu G mají svůj ekvivalent v grafu H a všechny hrany zůstanou zachovány, vrátí `True`.
- Pokud žádná permutace nevyhovuje, vrátí `False`.

Implementačně je algoritmus následující:

```python
import networkx as nx
import itertools

def isomorphism(G, H):
    if len(G.nodes) != len(H.nodes):
        return False

    # získání všech permutací uzlů grafu G
    perms = itertools.permutations(list(G.nodes))
    for perm in perms:
        # vytvoření mapování uzlů grafu G na uzly grafu H
        # pokud všechny uzly grafu G mají svůj ekvivalent v grafu H a všechny hrany zůstanou zachovány,
        ↪ vrátí True
        mapping = dict(zip(G.nodes, perm))
        has_all_nodes = all([mapping[u] in H.nodes for u in G.nodes])
        has_all_edges = all([mapping[u] in H.nodes for u in G.nodes])
        if has_all_nodes and has_all_edges:
            return True
        return False
```

Poté již zbývá jen funkci výše otestovat na několika příkladech grafů a porovnat výsledky s knihovní funkcí pro testování isomorfismu:

```python
G = nx.generators.small.cycle_graph(5)
H = nx.complement(nx.generators.small.cycle_graph(5))

my_isom = isomorphism(G, H)
nx_isom = nx.is_isomorphic(G, nx.complement(H))
print(f"Vlastní: {my_isom}\nNetworkX: {nx_isom}\nÚspěch: {'Ano' if my_isom == nx_isom else 'Ne'}")
```

Úplný zdrojový kód je k nalezení v souboru ukol-2-k1.py.

✓ Úloha programovací 2 (4 body)

Realizujte hrubou silou nalezení největší nezávislé množiny daného grafu a následně otestujte, že je množina nezávislá. Následně se pokuste vylepšit řešení procházení množinami použitím sousedů vrcholu.

Pokud chceme testovat procházení, nabízí se, ne nutně, řešení pomocí nějakého rekurzivního přístupu.

Řešení

- **Maximální (Maximal) nezávislá množina** je taková množina uzlů, kde žádné dva uzly nejsou spojeny hranou a nelze přidat další uzel, aby zůstala nezávislá.
- **Největší (Maximum) nezávislá množina** je taková množina, která má největší počet uzlů.

Pro řešení naimplementujeme nejdříve funkci na zjištění nezávislosti množiny `is_independent_set(G, ind_set)`:

```
def is_independent_set(G, ind_set):
    """
    Množina je nezávislá, pokud žádné dva uzly nejsou spojeny hranou.
    :param G: Graf G
    :param ind_set: Množina uzlů
    :return: True pokud je množina nezávislá, jinak False
    """
    # pro všechny uzly v množině
    for node in ind_set:
        # projdeme všechny sousedy uzlu
        for neighbor in G.neighbors(node):
            # pokud je soused také v množině, množina není nezávislá
            if neighbor in ind_set:
                return False
    return True
```

Tato funkce nám umožní testovat nezávislost množiny uzlů v grafu. Následně můžeme implementovat funkci pro nalezení největší nezávislé množiny grafu `get_max_independent_set(G)` pomocí hrubé síly:

```
def get_max_independent_set(G):
    """
    Nalezení největší nezávislé množiny grafu hrubou silou.
    :param G: Graf G
    :return: Největší nezávislá množina grafu G
    """
    # inicializace (prázdné) maximální nezávislé množiny
    max_ind_set = set()
    # pro všechny možné velikosti množin
    for i in range(1, len(G.nodes) + 1):
        # pro všechny možné kombinace uzlů
        for ind_set in itertools.combinations(G.nodes, i):
            # pokud je množina nezávislá a má větší počet uzlů než dosavadní maximální množina
            if is_independent_set(G, ind_set) and len(ind_set) > len(max_ind_set):
                # nastavíme novou maximální množinu
                max_ind_set = set(ind_set)
    return max_ind_set
```

Pro otestování nyní zavoláme naši funkci `get_max_independent_set(G)` a porovnáme oproti výsledku z knihovny `networkx`:

```
G = nx.generators.small.petersen_graph()
# brute force
brute_max_ind_set = log_perf(get_max_independent_set)(G)
print(f"[brute] Set: {brute_max_ind_set}, Length: {len(brute_max_ind_set)}, Is independent:
↪ {is_independent_set(G, brute_max_ind_set)}")
# confirm via networkx
nx_max_ind_set = log_perf(nx.algorithms.approximation.maximum_independent_set)(G)
print(f"[nx] Set: {nx_max_ind_set}, Length: {len(nx_max_ind_set)}, Is independent:
↪ {is_independent_set(G, nx_max_ind_set)}")
# assert
assert len(brute_max_ind_set) == len(nx_max_ind_set)
# >>> [get_max_independent_set] took 0.36 ms
# >>> [brute] Set: {0, 8, 2, 9}, Length: 4, Is independent: True
# >>> [maximum_independent_set] took 1.28 ms
# >>> [nx] Set: {8, 9, 2, 0}, Length: 4, Is independent: True
```

Lze vidět, že naše funkce pro nalezení největší nezávislé množiny grafu funguje správně a vrací stejný výsledek jako funkce z knihovny `networkx`. Dále si také můžeme všimnout že průměrná doba běhu naší funkce je $0.36ms$ oproti $1.28ms$ funkce z knihovní funkce.

Nyní se pokusme naši funkci vylepšit pomocí rekurzivního přístupu:

```
def recursive_independent_set(G, current_set, nodes_remaining):
    """
    :param G:
    :param current_set:
    :param nodes_remaining:
    :return:
    """
    # pokud již nejsou žádné uzly k procházení, vrátíme aktuální množinu
    if not nodes_remaining:
        return current_set

    # inicializace maximální množiny
    max_set = current_set
    # pro všechny uzly, které ještě nebyly zpracovány
    for node in nodes_remaining:
        # všechny sousedy uzlu
        neighbors = G.neighbors(node)
        # pokud žádný soused není v aktuální množině
        if all(neighbor not in current_set for neighbor in neighbors):
            # vytvoříme novou množinu s uzlem
            new_set = current_set.union({node})
            # rekurzivně zavoláme funkci pro další uzly
            remaining = nodes_remaining.difference(new_set).difference(set(G.neighbors(node)))
            candidate_set = recursive_independent_set(G, new_set, remaining)
            # pokud je nová množina větší než dosavadní maximální množina
            if len(candidate_set) > len(max_set):
                # nastavíme novou maximální množinu
                max_set = candidate_set

    return max_set
```

Otestováním této funkce zjistíme, že výsledek je sice správný a tedy stejný jako u předchozích funkcí, ale doba běhu je výrazně delší, což je způsobeno rekurzivním přístupem:

```
            # improved with neighbors
            recursive_max_ind_set = log_perf(recursive_independent_set)(G, set(), set(G.nodes))
            print(f"[recursive] Set: {recursive_max_ind_set}, Length: {len(recursive_max_ind_set)}, Is independent: {is_independent_set(G, recursive_max_ind_set)}")
            # confirm via networkx
            nx_max_ind_set = log_perf(nx.algorithms.approximation.maximum_independent_set)(G)
            print(f"[nx] Set: {nx_max_ind_set}, Length: {len(nx_max_ind_set)}, Is independent: {is_independent_set(G, nx_max_ind_set)}")
            # assert
            assert len(brute_max_ind_set) == len(nx_max_ind_set)
            # >>> [recursive_independent_set] took 0.43 ms
            # >>> [recursive] Set: {0, 8, 2, 9}, Length: 4, Is independent: True
            # >>> [maximum_independent_set] took 1.28 ms
            # >>> [nx] Set: {8, 9, 2, 0}, Length: 4, Is independent: True
```

Úplný zdrojový kód je k nalezení v souboru ukol-2-k2.py.