ЛЕКЦІЯ10

Розфарбування графів. Хроматичне число

Розфарбування графів. Хроматичне число

Задачі розфарбування вершин або ребер графа займають важливе місце в теорії графів.

До задачі побудови розфарбування графа зводиться цілий ряд практичних задач.

Одна з областей – складання розкладів.

- розкладу для освітніх закладів;
- розкладу в спорті;
- планування зустрічей, зборів, інтерв'ю;
- розкладу транспорту, у тому числі — авіатранспорту;
- розкладу для комунальних служб;
- інші.

Правильне розфарбування

Нехай $G=(V,E)$ — скінченний граф, а k — деяке натуральне число.

Вершинне розфарбування. Довільну функцію виду $f:V\to N_k$, де $N_k=\{1,2,...,k\}$ - множина кольорів,

називають *вершинним k-розфарбуванням*, або просто *k-розфарбуванням* графа G.

Правильне розфарбування. Розфарбування називають *правильним*, якщо кольори суміжних вершин не співпадають, тобто для всіх $(u,v)\in E$ справедливо $f(u)\neq f(v)$.

Розфарбований граф. Граф, для якого існує правильне k-розфарбування, називають *розфарбованим графом.*

Базовий принцип оптимізації розфарбування

Якщо функція f **не взаємно однозначна**, то при $|V| = k$ фактично може бути використано **менше**, ніж k кольорів.

Правильне розфарбування – це розбиття множини вершин

Правильне k-розфарбування можна розглядати як розбиття множини вершин V графа G на класи $V_1 \cup V_2 \cup ... \cup V_l = V$, де $l \leq k$, $V_i \neq \varnothing$, $i = 1, 2, ..., l$.

Кожний клас V_i — незалежна множина, а класи називають *кольоровими класами.*

Хроматичне число

Визначення. Мінімальне число k, при якому існує правильне k-розфарбування графа G, називають *хроматичним числом* цього графа і позначають $X(G)$.

Визначення. Якщо $X(G)=k$, то граф G називають *k-хроматичним.* Тобто його вершини можна розфарбувати k різними кольорами так, що у будь-якого ребра інцидентні вершини матимуть різний колір.

Визначення. Правильне k-розфарбування графа G при $k = X(G)$ називають *мінімальним.*

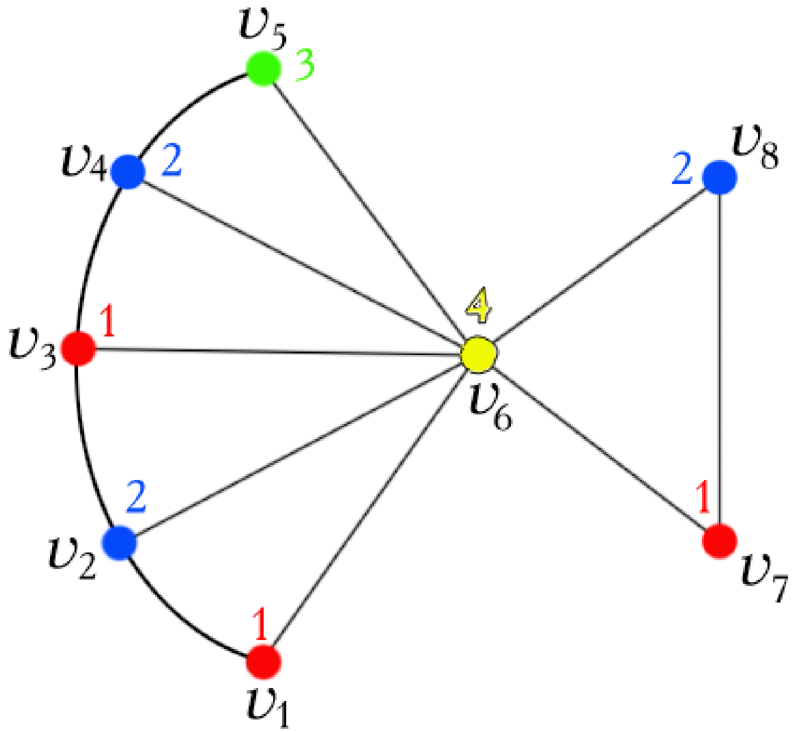
Визначення. Хроматичне число незв'язного графа дорівнює максимальному з хроматичних чисел його компонент зв'язності.

Приклад.

Розглянемо граф G, зображений на рисунку, на якому показано одне із правильних k-розфарбувань. Натуральними числами 1,2,3,4 позначені кольори відповідних вершин.

У цьому випадку кількість кольорів не є мінімальною.

Тому $X(G) < k$.

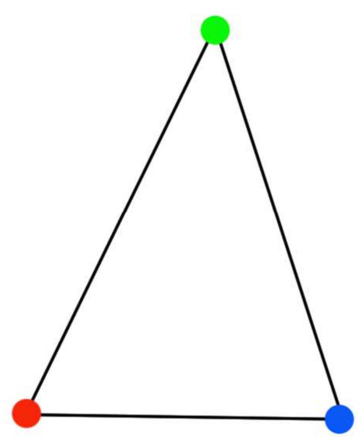


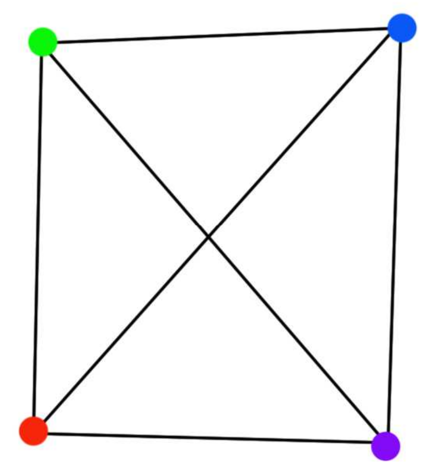
Хроматичні числа деяких графів

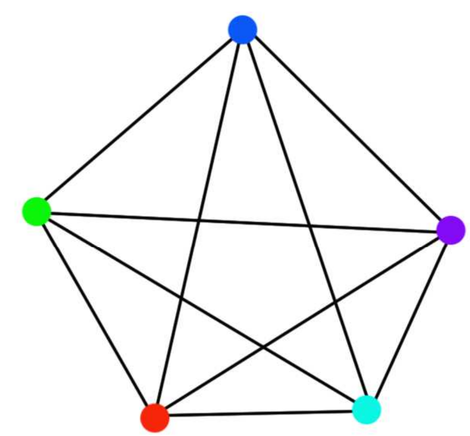
Для деяких простих графів неважко знайти хроматичні числа.

Приклади.

1. Повний граф K_n, що складається з n вершин, має хроматичне число $X_p(K_n) = n$

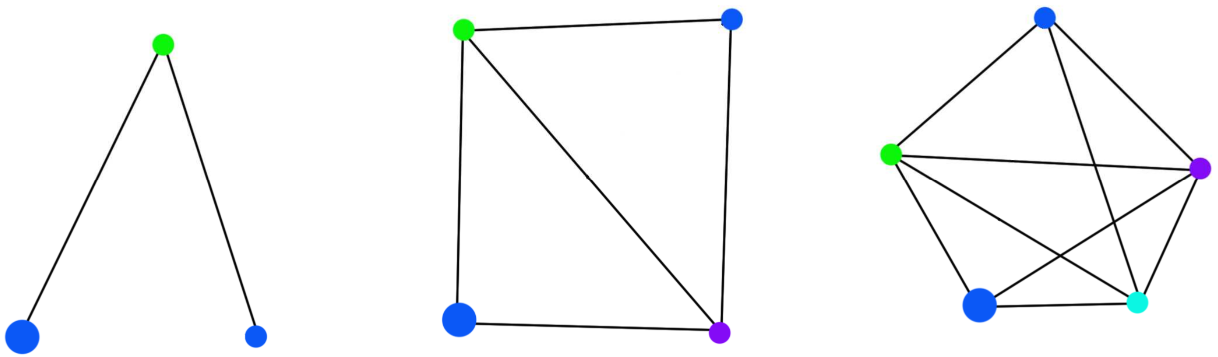




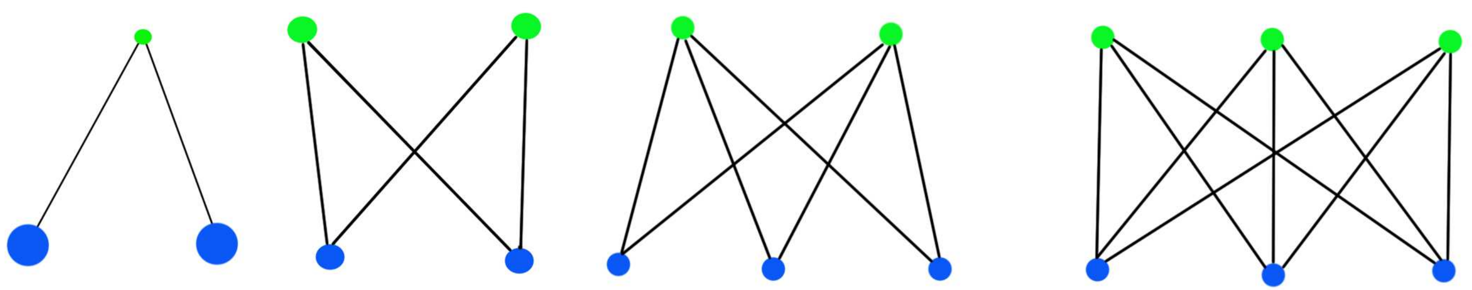


$X(K_2) = 2$ $X(K_3) = 3$ $X(K_4) = 4$ $X(K_5) = 5$

2. Повний граф $K_n - e$, який складається з n вершин з одним відсутнім ребром, має хроматичне число $X_p\left(K_n - e\right) = n - 1$



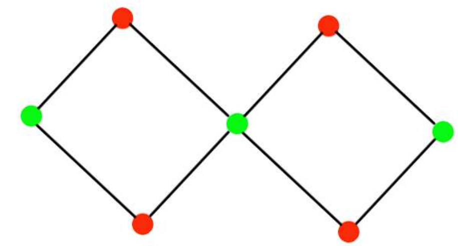
3. Повні дводольні графи $K_{m,n}$, що складаються з долей $|A| = m$ і $|B| = n$, мають хроматичне число $X_p\left(K_{m,n}\right) = 2$



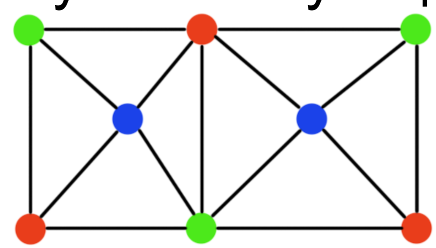
Теорема. Непустий граф є **біхроматичним** тоді й тільки тоді, коли він не має циклів непарної довжини.

1-хроматичний граф – порожній граф.

2-хроматичний (біхроматичний) граф — дводольний непустий граф або граф тільки з парними циклами.



3-хроматичний граф — циклічний граф з непарним числом вершин у кожному з циклів.

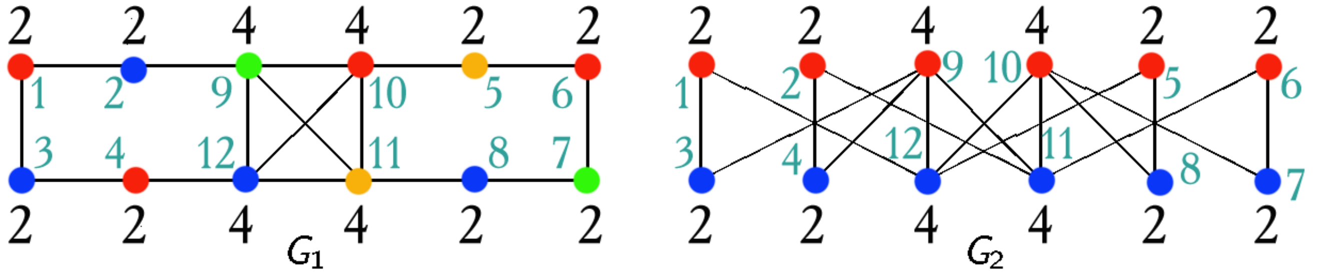


Властивість 1. Якщо граф має n вершин, то його хроматичне число не перевищує n.

Властивість 2. Якщо граф має підграф K_m, то його хроматичне число не менше, ніж m.

Хроматичне число й стандартні характеристики

У загальному випадку хроматичне число графа не можна точно обчислити, знаючи тільки його стандартні числові характеристики: *число вершин, ребер, компонент зв'язності, розподіл степенів вершин.*



Розглянемо графи G_1 й G_2. Кожний з них має 12 вершин, у тому числі 4 вершини зі степенем 4 і 8 вершин зі степенем 2, 16 ребер, одну компоненту зв'язності. Але, як видно з рисунка, $X(G_1)=4$, а $X(G_2)=2$, оскільки G_1 містить у якості підграфа граф K_4.

Оскільки граф G_2 — дводольний, маємо $X(G_2)=2$.

Тому надалі мова йтиме про оцінки, а не про точні значення хроматичного числа.

НИЖНІ ОЦІНКИ ХРОМАТИЧНОГО ЧИСЛА

Хроматичне число і щільність графа

Під нижніми оцінками хроматичного числа будемо розуміти нерівності виду $X(G) \geq c$, де c — деяка константа, що обчислюється на графі G.

Верхня оцінка хроматичного числа — це нерівності виду $X(G) \leq c$, де c має той же зміст, тобто є константою.

Визначення. Максимальне число вершин, що породжують повний підграф у графі G, називають ***щільністю*** G і позначають через $\omega(G)$.

Визначення. Повний підграф деякого графа G - це підграф, що складається з попарно суміжних вершин.

Перша нижня оцінка може застосовуватися у випадку, якщо підграфом деякого графа є повний підграф.

Перша нижня оцінка

Для довільного графа G справедлива нерівність $X(G) \geq \omega(G)$.

Хроматичне число і число незалежності графа

Визначення. Будь-яку множину попарно несуміжних вершин графа G називають *незалежною множиною.*

Визначення. Максимальне число вершин у незалежній множині називають *числом незалежності (внутрішньої стійкості)* графа G й позначають через $\beta(G)$.

Число незалежності графа — це поняття, протилежне за змістом поняттю щільності графа. Якщо G — звичайний граф, а $\overline{G}$ — його доповнення, то $\beta(G)=\omega(\overline{G})$.

Друга нижня оцінка

Для довільного графа G справедлива нерівність

$$X(G) \geq \frac{n(G)}{\beta(G)},$$

де $n(G)-$ *кількість вершин графа* G

Третя нижня оцінка хроматичного числа

Існують **нижні оцінки** хроматичного числа, які використовують тільки ті характеристики графа, що легко обчислюються. Наведемо без доведення одну з них.

Третя нижня оцінка

Якщо G – звичайний граф і

$n = n(G)-$ кількість вершин графа G,

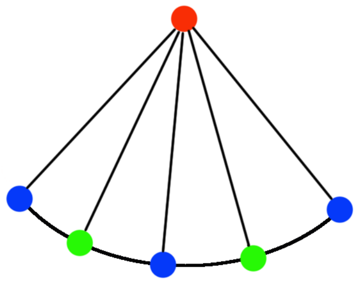
$m = m(G)$– кількість ребер графа G,

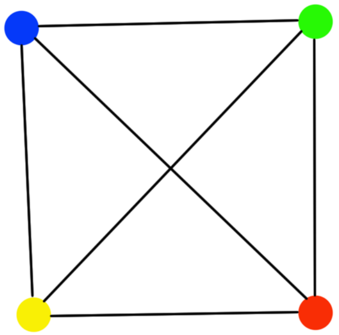
то хроматичне число $X(G) \geq \dfrac{n^2}{n^2 - 2m}$.

Легко зрозуміти, що в повному графі (як і в будь-якому звичайному графі) подвоєне число ребер менше квадрата числа вершин, і тому число, що стоїть в знаменнику в правій частині нерівності, завжди додатне.

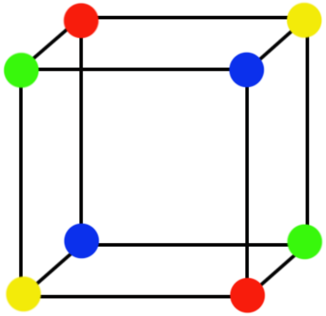
ВЕРХНІ ОЦІНКИ ХРОМАТИЧНОГО ЧИСЛА

Теорема 2. Для будь-якого графа G має місце нерівність $X(G) \le r+1$, де $r = \max\limits_{v \in V}\left(\deg(v)\right)$.





Наслідок. Будь-який кубічний граф розфарбовується за допомогою чотирьох кольорів.



Для певних графів справедлива теорема.

Теорема Брукса. *Якщо* G *— зв'язний неповний граф* і

$$r \geq 3,\ \text{де}\ r = \max_{v \in V}\left(\deg\left(v\right)\right),\ \text{то}\ X\left(G\right) \leq r\ .$$

Отже, обмеження для даної теореми:

1. Граф зв'язний неповний
2. $r \geq 3$

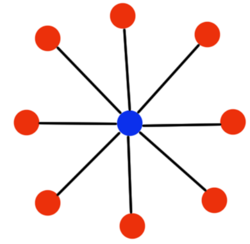
Верхня оцінка за кількістю ребер.

Нехай $G\left(V,E\right)$ - довільний зв'язний неорієнтований граф з m ребрами. Тоді

$$X\left(G\right) \leq \frac{1}{2} + \sqrt{2m + \frac{1}{2}}$$

Хоча обидві теореми й дають певну інформацію про хроматичне число графа, але їх оцінки досить неточні.

Розглянемо **граф-зірку** K_{1n},



Цей граф відповідає умові теореми Брукса.

Адже 1) $r = 8$, 2) граф зв'язний, $3)$ граф неповний.

За умовою тереми Брукса $X(G) \leq 8$, але насправді є **біхроматичним.** Отже така оцінка є неточною.

Трохи точнішою для такого графа є реберна оцінка.

$$X(G) \leq \frac{1}{2} + \sqrt{2 \cdot 8 + \frac{1}{4}} = 4.53$$

Ця ситуація значно спрощується, якщо обмежитися ***планарними графами***. У цьому випадку легко довести такий досить загальний і важливий факт.

Історично послідовно доведені теореми

Теорема про шість фарб. Для будь-якого планарного (ізоморфного плоскому (у якому ребра перетинаються лише у вершинах)) графа G має місце нерівність $X_p\left(G\right) \le 6$.

Більш детальний аналіз шляхів зниження верхньої границі хроматичного числа приводить до так званої *теореми про п'ять фарб.*

Теорема про п'ять фарб. Для всякого планарного графа G має місце нерівність $X_p\left(G\right) \le 5$.

Теорема про чотири фарби. Кожний планарный граф без петель і кратних ребер є не більш ніж 4-хроматичним.

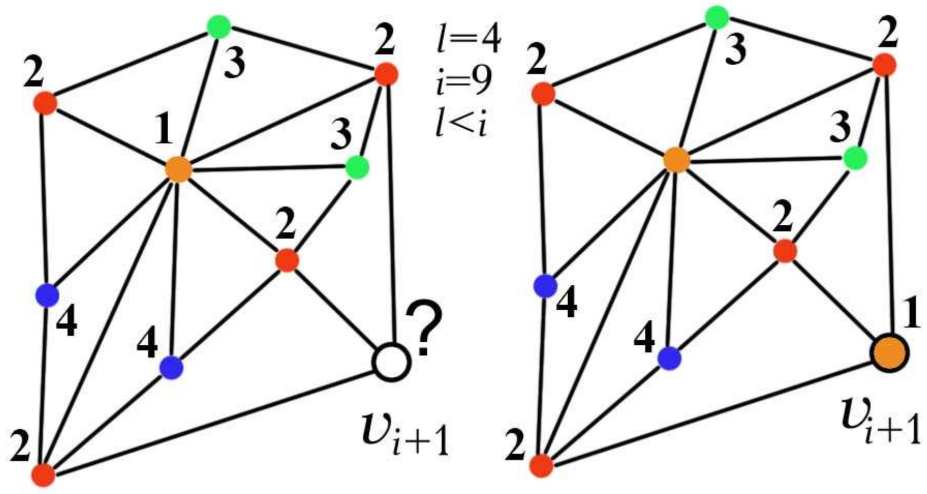
Теорема сформульована вперше Френсісом Гутрі (*англ.*) в 1852 році.Проблема чотирьох фарб залишалася невирішеною протягом багатьох років. Стверджується, що ця теорема була доведена за допомогою певних міркувань і комп'ютерної програми в 1976 році.

Алгоритми розфарбування

Алгоритм послідовного розфарбування

Якщо вершини $v_1, v_2, ..., v_i$ розфарбовані l кольорами $1, 2, ..., l;\ l \le i$, то новій довільно взятій вершині v_{i+1} припишемо мінімальний колір, не використаний при розфарбуванні суміжних з нею вершин.

Розфарбування, до якого приводить описаний алгоритм, називають *послідовним*.



Алгоритм прямого неявного перебору

```
def Color (i):
'''Функція вибору фарби для розфарбування вершини з номером i'''
  W = set();
  for j in range(i):
     if A[i][j] == 1:
       W.add(Colarr [j])
 '''Формування множини фарб, використаних для розфарбування прилеглих до вершини i вершин з номерами менше i'''
     while k not in W:
      k+=1
      return = j;
```

1. Введемо наступні структури даних:

Nmax - максимальна кількість вершин графа

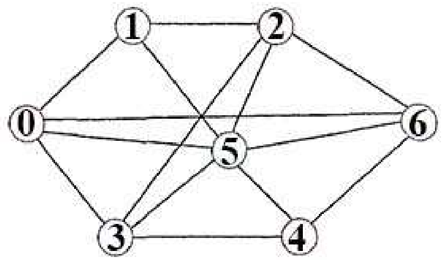
Colarr - список *номерів фарб для кожної вершини графа*

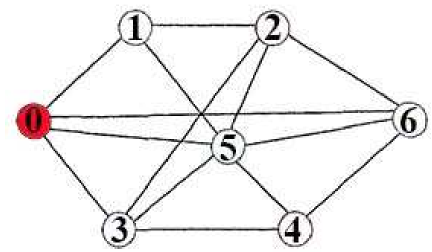
A - *матриця суміжності графа*

2. Виконуємо цикл по вершинах графа

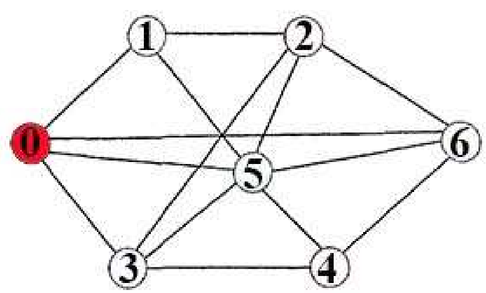
```
for i in range(Nmax):
  Colarr [i]=Color (i)
'''Виводимо результат розфарбування'''
```

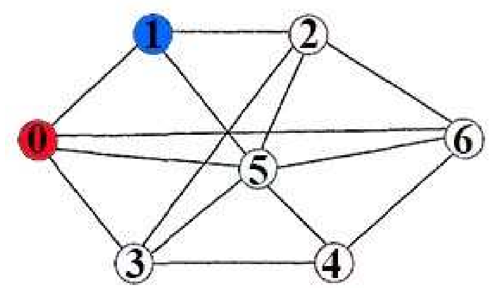
Крок1. Розглядаємо вершину 0. Множина розфарбованих суміжних вершин W порожня. Тому функція Color(0) повертає фарбу 0. Нехай колір 0- червоний.



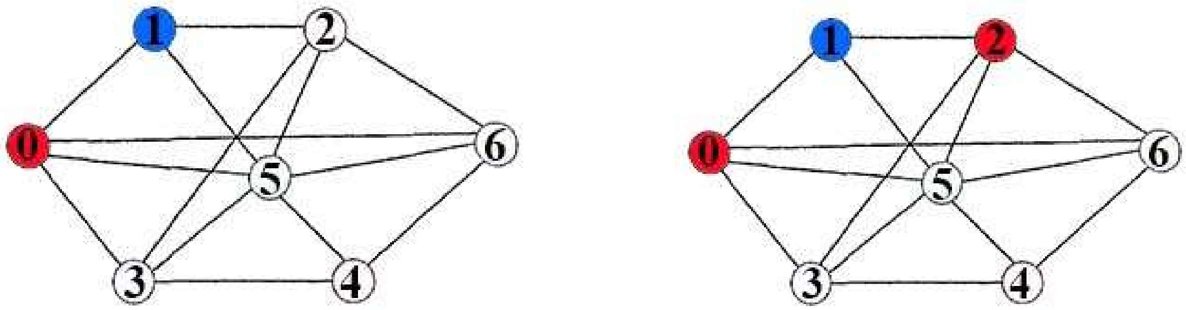


Крок2. Розглянемо вершину 1. Єдиною меншою за номером суміжною вершиною є вершина 0, яка уже червона. Тому множина W містить єдиний елемент 0. Функція Color(1) повертає наступну за номером фарбу 1 синього кольору.

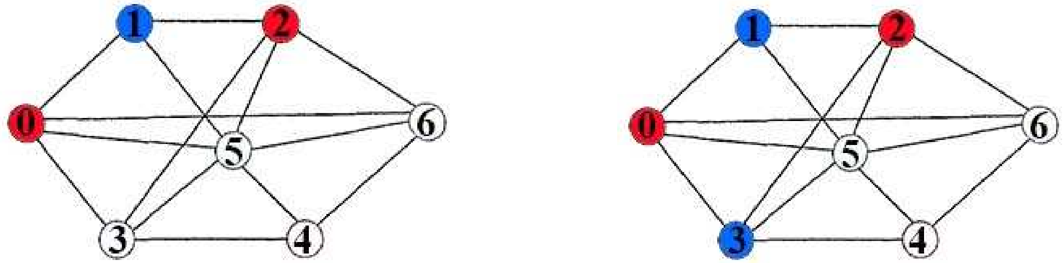




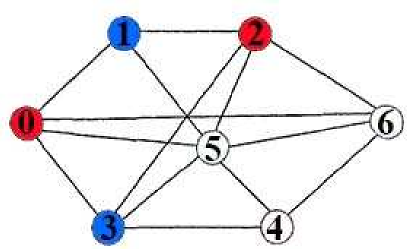
Крок3. Вершина 2 має єдину суміжну вершину 1 з меншим номером. Множина W містить єдиний елемент 1. Функція Color(2) повертає фарбу з номером 0 червоного кольору.

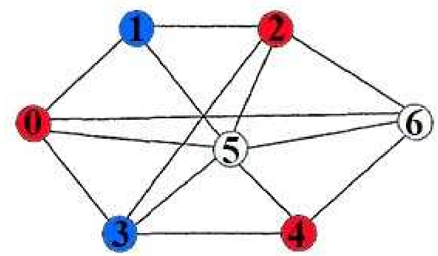


Крок4. Вершина 3 має дві суміжні вершини з меншими номерами:0 і 2. Оскільки обидві вершини розфарбовані в колір 0, то множина W містить єдиний елемент 0. Тому функція Color(3) повертає наступну за номером фарбу 2 синього кольору.

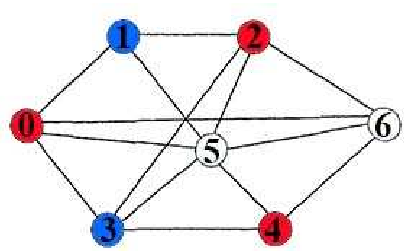


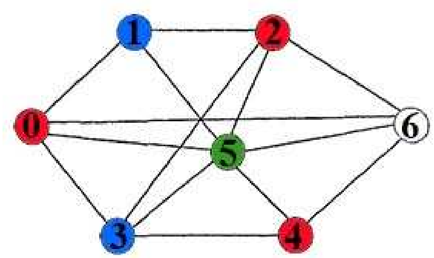
Крок5. Вершина 4 має єдину суміжну вершину з меншим номером. Це вершина 4. Множина W містить єдиний елемент 1. Тому функція Color(4) повертає фарбу з номером 0 червоного кольру.





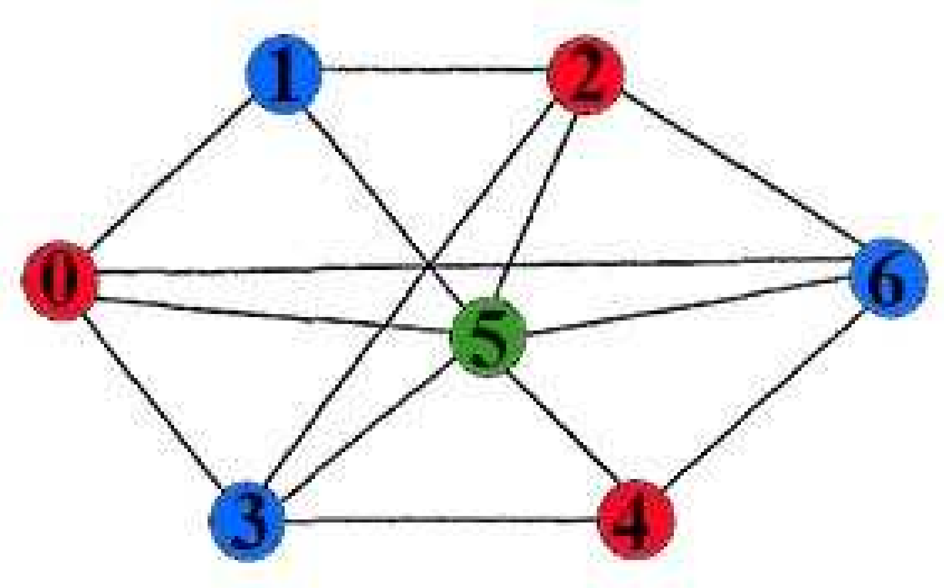
Крок6. Вершина 5 має такі суміжні вершини з меншими номерами:1,3,4 і 5. Ці вершини розфарбовані в колір 0 та колір 1. Отже, множина W містить два елементи: 0 і 1. Тому функція Color(5) повертає наступну за номером фарбу 2 зеленого кольору.





Крок7. Вершина 6 має такі суміжні вершини з меншими номерами:1,3, 5 і 6. Ці вершини розфарбовані в колір 0 та колір 2. Отже, множина W містить два елементи: 0 і 2. Тому функція Color(6) повертає фарбу 1 синього кольору.

В результаті роботи данного алгоритму одержуємо правильно розфарбований граф, що показаний на рисунку.



Рекурсивна процедура послідовного розфарбування

1. Фіксуємо порядок обходу вершин.
2. Ідемо по суміжних вершинах, використовуючи такий найменший колір, який не викличе конфліктів.
3. Якщо на черговому кроці колір вибрати не виходить, то «відкочуємось» до попередньої вершини й вибираємо для неї наступний колір, який не викличе конфліктів.

```
def visit(i):
  if i== n + 1:
    Print #Друк після розфарбування всіх вершин
  else # на початку color[i]=0-не розфарб.вершини
    for c in range(color[i],k):
      #k-кількість кольорів
       if (знайдено неконфліктний колір c):
          color[i]= c
          visit(i + 1)
       else
          visit(i)
visit(0) #Виклик рекурсивної процедури
```

«Жадібний» алгоритм розфарбування

Нехай дано зв'язний граф $G(V,E)$.

1. Задамо множину $monochrom := \varnothing$, куди будемо записувати всі вершини, які можна розфарбувати одним кольором.

2. Переглядаємо всі вершини й виконуємо наступний «жадібний» алгоритм

def Greedy:

  for (для кожної незафарбованої вершини $v \in V$):

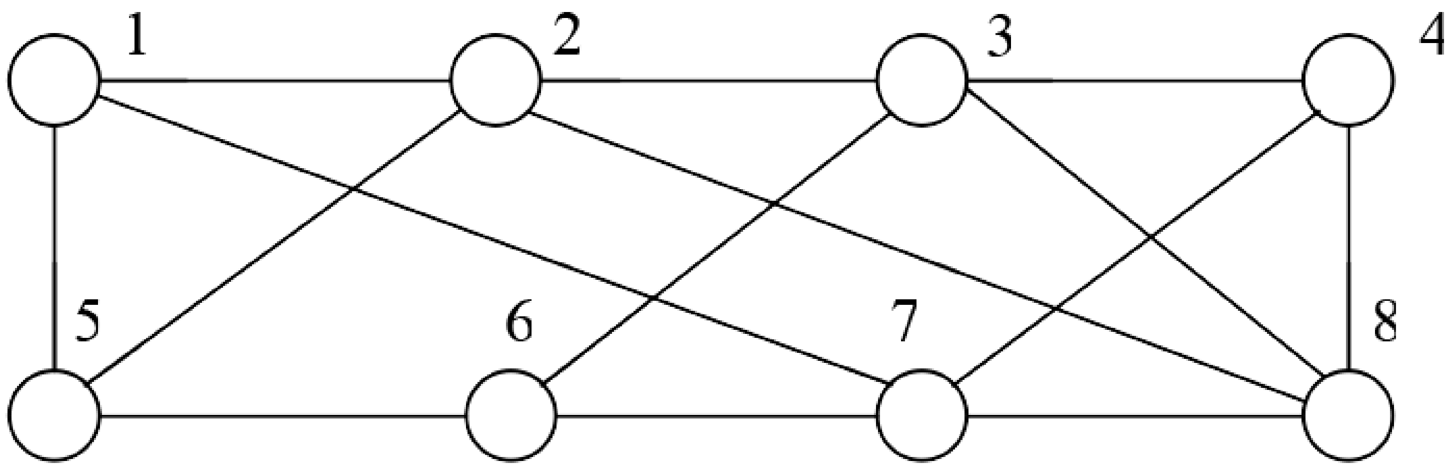
    If (v *не суміжна з вершинами з* $monochrom$):

      $color(v) =$ колір;

      $monochrom = monochrom \cup \{v\}$

ПРИКЛАД РОБОТИ «ЖАДІБНОГО» АЛГОРИТМУ РОЗФАРБУВАННЯ

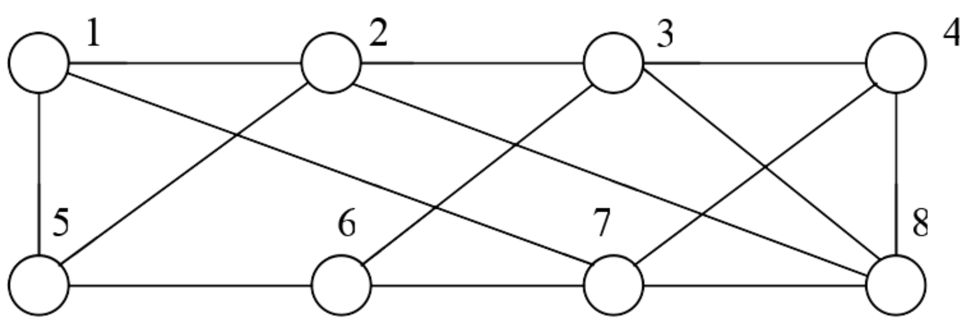
Розглянемо граф G:



Множину вершин графа $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$ потрібно розфарбувати з використанням «ЖАДІБНОГО» алгоритму розфарбування.

Сформуємо матрицю суміжності A :

$$A = \begin{array}{cc} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{matrix} & \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$



1. Знаходимо нерозфарбовану вершину і встановлюємо для неї новий колір.
2. Запускаємо процедуру «жадібного» розфарбування, яка розфарбовує в цей колір всі вершини, які тільки можливо.
3. Якщо не всі вершини розфарбовані, то переходимо до п.1.

Крок 1. Вибираємо вершину 1 і фарбуємо її в червоний колір 1.

Крок 1.1.

2-конфліктна, оскільки суміжна з 1

3-неконфліктна. Фарбуємо в червоний.

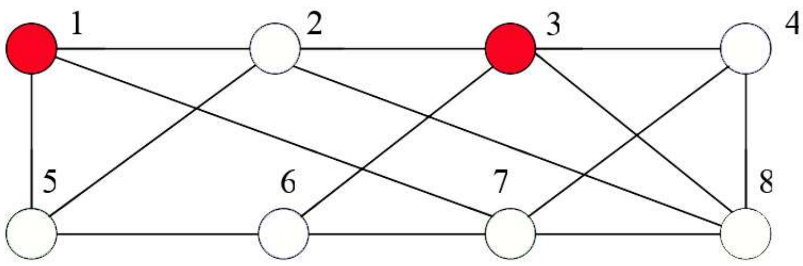
4- конфліктна, оскільки суміжна з 3.

5- конфліктна, оскільки суміжна з 1.

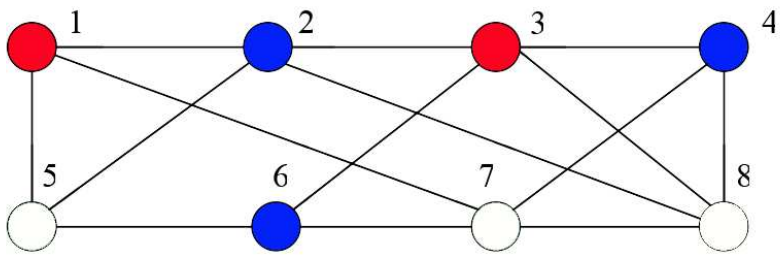
6- конфліктна, оскільки суміжна з 3.

7- конфліктна, оскільки суміжна з 1.

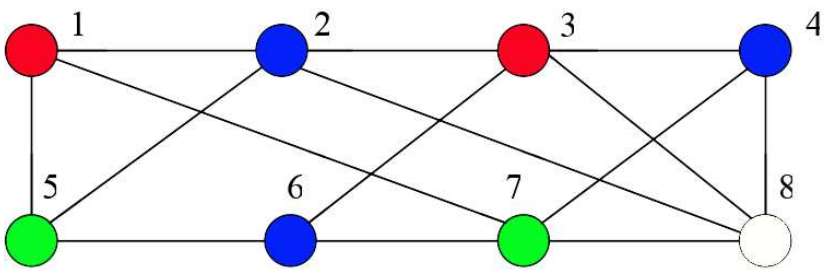
8- конфліктна, оскільки суміжна з 3.



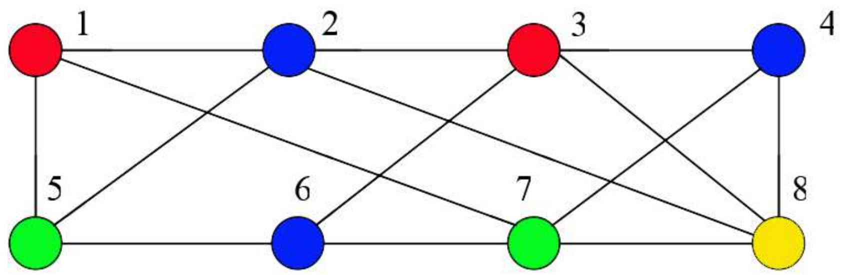
Крок 2. Вибираємо вершину 2 і фарбуємо її в синій колір 2.
Крок 2.1.
1-конфліктна, має колір.
3-конфліктна, має колір.
4- неконфліктна. Фарбуємо в синій.
5- конфліктна, оскільки суміжна з 2.
6- неконфліктна. Фарбуємо в синій.
7- конфліктна, оскільки суміжна з 4.
8- конфліктна, оскільки суміжна з 4.



Крок 3. Вибираємо вершину 5 і фарбуємо її в зелений колір 3.
Крок 3.1.
1-конфліктна, має колір.
2-конфліктна, має колір.
3-конфліктна, має колір.
4-конфліктна, має колір.
6-конфліктна, має колір.
7- неконфліктна. Фарбуємо в зелений.
8- конфліктна, оскільки суміжна з 7.



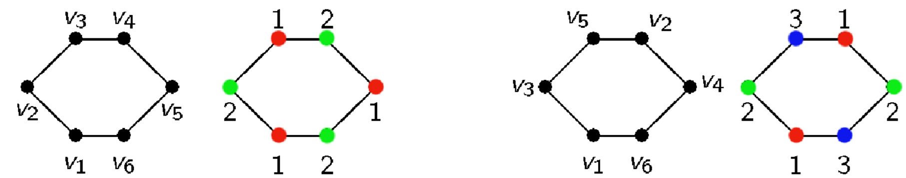
Крок 4. Вибираємо вершину 8 і фарбуємо її в жовтий колір 4.



1-конфліктна, має колір.
2-конфліктна, має колір.
3-конфліктна, має колір.
4-конфліктна, має колір.
5-конфліктна, має колір.
6-конфліктна, має колір.
7-конфліктна, має колір.
Кінець алгоритму.

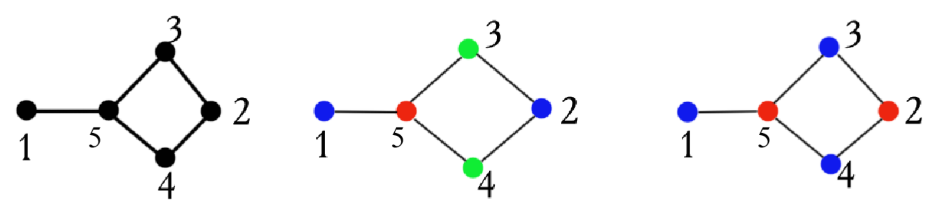
Результати роботи алгоритмів послідовного розфарбування

Отримане розфарбування завжди правильне, але не завжди оптимальне навіть для простих графів.



Воно суттєво залежить від того, у якому порядку вибираються вершини. На першому рисунку вийшов оптимальний результат (2 фарби), а на другому рисунку використано більше кольорів.

Нехай зафарбуємо вершину 1 у синій колір, а потім, пропустивши вершину 2, зафарбуємо в синій колір вершини 3 і 4. Тоді можна одержати 2 кольори.



Але "жадібний" алгоритм, ґрунтуючись на нумерації вершин, зафарбує в синій колір вершини 1 і 2, для розфарбування графа тепер потрібно 4 кольори.

Висновок. Спробувати розфарбовувати не за номерами

Алгоритм послідовного розфарбування з упорядкуванням множини вершин (Евристичний алгоритм)

1. Упорядкувати вершини по незростанню степеня.

2. Вибрати колір фарбування 1.

3. Розфарбувати першу вершину в колір 1.

4. Поки не пофарбовані всі вершини, повторювати п.4.1. - 4.2.:

4.1. Проходим по списку зверху вниз, розфарбувати в обраний колір всі вершини, які не суміжні з вершиною, яка уже пофарбована в цей колір.

4.2. Вибрати наступний колір.

4.3. Повернутися до першої в списку нерозфарбованої вершини, проходити по списку вниз та розфарбувати всі можливі вершини даним кольором.

{* СТРУКТУРА ДАНИХ ЕВРИСТИЧНОГО АЛГОРИТМУ *}

Nmax - *максимальна кількість вершин графа*

Colarr - список *номерів фарб для кожної вершини графа*

Degarr - список *степенів вершин*

Sortarr- список, *відсортований за зменшенням степенів*

A – двовимірний список (*матриця суміжності графа*)

Curcol - *поточний номер фарби*

n,i - службові змінні

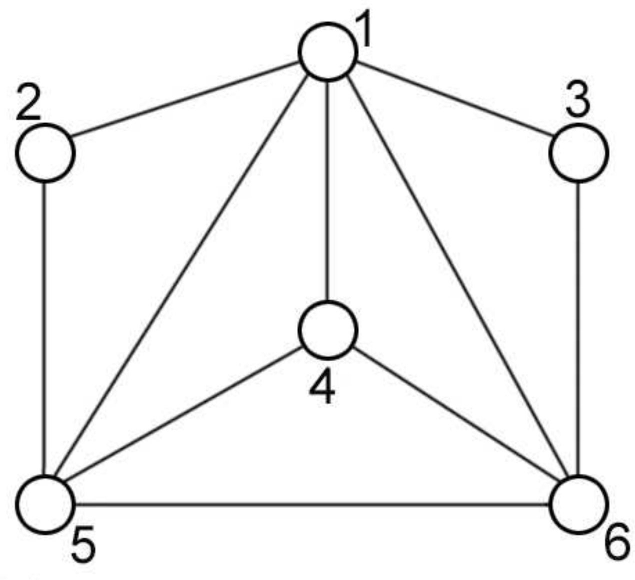
’’’ **п р о г р а м а**’’’

```
Curcol = 1 #початковий колір
Nmax=100

'''Вводимо матрицю суміжності графа'''

Degforming #Формування масиву степенів вершин

Sortnodes #Формування масиву відсортованих вершин Sortarr

for n  in range(1,Nmax+1):

   If Colarr [Sortarr [n]]== 0:

      Colarr [Sortarr [n]] = Curcol  #розфарбування вершини в списку

      Color (Sortarr [n]) #розфарбування суміжних до даної вершин

     Inc (Curcol);

'''Виводимо результат розфарбування'''
```

```
def Degforming:  # Функція формування масиву степенів вершин
  for i  in range(1, Nmax+1):
    Degarr [i] = 0
    Colarr [i] = 0;
    for j  in  range(1, Nmax+1 ):
      Degarr [i]+=A [i][j]
def Sortnodes: # Сортування вершин за степенями
  for k in range(1, Nmax):
     max = Degarr [k]
     c = k;
    for i in range(k + 1, Nmax+1):
       If Degarr [i]> max:
         max = Degarr [[i]
         c = i
    Degarr [c] = Degarr [[k];
    Degarr [k] = max;
    Sortarr [k] = c;
def Color (i);
'''Розфарбування поточним кольором НЕ суміжних з i вершин '''
   for j in range(1, Nmax+1):
     if A [j, i] == 0:
        If Colarr [j] == 0:
           Colarr [j] = Curcol
```

ПРИКЛАД ЕВРИСТИЧНОГО АЛГОРИТМУ РОЗФАРБУВАННЯ

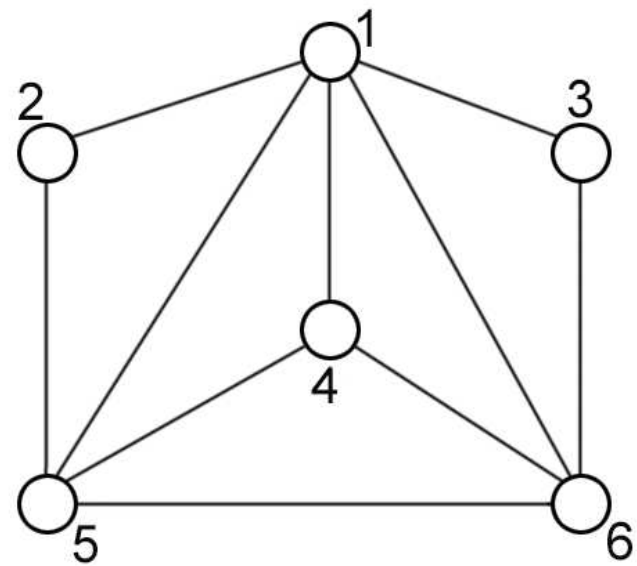
Дано граф G, зображений на рисунку.



Множину вершин графа $V = \{1, 2, 3, 4, 5, 6\}$ потрібно розфарбувати з використанням евристичного алгоритму розфарбування.

Сформуємо матрицю суміжності A :

$$A=\begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 2 & 1 & 0 & 0 & 0 & 1 & 0 \\ 3 & 1 & 0 & 0 & 0 & 0 & 1 \\ 4 & 1 & 0 & 0 & 0 & 1 & 1 \\ 5 & 1 & 1 & 0 & 1 & 0 & 1 \\ 6 & 1 & 0 & 1 & 1 & 1 & 0 \end{array}$$



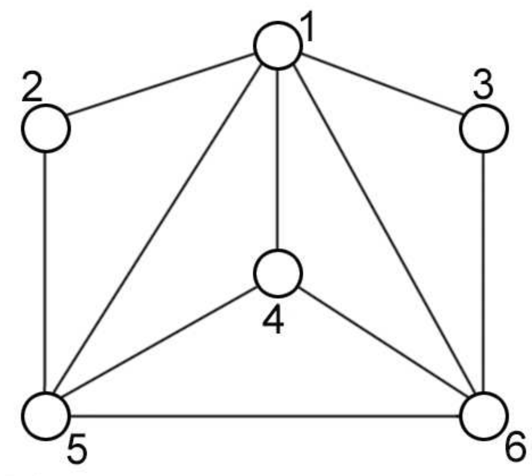
Відсортуємо вершини графа за зменшенням їх степенів. В результаті отримуємо вектор відсортованих вершин $\mathrm{SortArr}=(1,5,6,4,2,3)$

SortArr	1	5	6	4	2	3
DegArr	5	4	4	3	2	2

1. Рухаючись вздовж масиву *SortArr* першій знайденій нерозфарбованій вершині надаємо черговий новий колір.
2. Всім несуміжним зі знайденою вершинам надаємо цей же колір.

У першому рядку таблиці запишемо вектор SortArr , у другому – степені відповідних вершин. Наступні рядки відображають вміст вектора розфарбування.

Номери вершин SortArr		1	5	6	4	2	3
Степені вершин DegArr		5	4	4	3	2	2
$CurCol = 1$		1	-	-	-	-	-
$CurCol = 2$		1	2	-	-	-	2
$CurCol = 3$		1	2	3	-	3	2
$CurCol = 4$		1	2	3	4	3	2



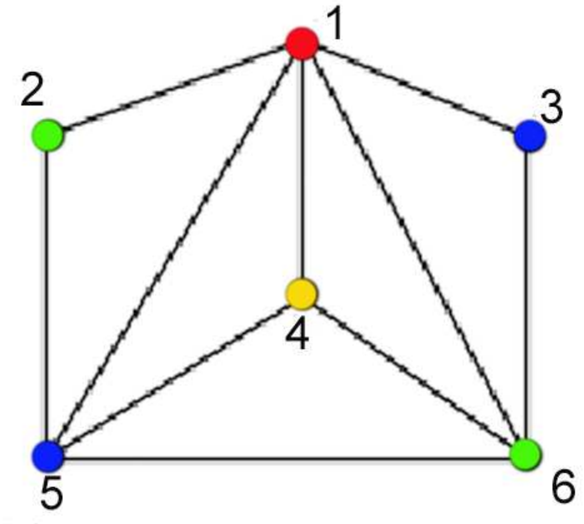
Крок 1. Першою в SortArr стоїть вершина 1, яку фарбуємо червоним кольором 1. Несуміжних з 1 немає.

Крок 2. Другою в SortArr стоїть вершина 5, яку фарбуємо синім кольором 2. Несуміжна з 5 вершина 3, яку процедура Color(5) фарбує також синім кольором 2.

Крок 3. Третьою в SortArr стоїть вершина 6, яку фарбуємо зеленим кольором 3. Несуміжна з 6 вершина 2, яку процедура Color(6) фарбує також зеленим кольором 3.

Крок 4. Четвертою в SortArr стоїть вершина 4, яку фарбуємо жовтим кольором 4. Всі несуміжні вершини з 4 вже розфарбовані

Номери вершин SortArr		1	5	6	4	2	3
Степені вершин DegArr		5	4	4	3	2	2
CurCol = 1		1	-	-	-	-	-
CurCol = 2		1	2	-	-	-	2
CurCol = 3		1	2	3	-	3	2
CurCol = 4		1	2	3	4	3	2



Модифікація алгоритму послідовного розфарбування

Визначення. Відносний степінь – це степінь нерозфарбованих вершин у нерозфарбованому підграфі початкового графа.
Визначення. Двокроковий відносний степінь – сума відносних степенів суміжних вершин у нерозфарбованому підграфі.

Проста модифікація описаної вище евристичної процедури полягає в переупорядкуванні нерозфарбованих вершин за незростанням їх відносних степенів.

Алгоритм відрізняється ускладненням процедури сортування Sortnodes, яка при сортуванні вершин з однаковими степенями враховує двокроковий степінь.

СТРУКТУРА ДАНИХ

Nmax- *максимальна кількість вершин графа*

Colarr - список *номерів фарб для кожної вершини графа*

Degarr - список *степенів вершин*

Sortarr – список, *відсортований по незростанню степенів*

A - *матриця суміжності графа*

Curcol - *поточний номер фарби*

n: службова змінна

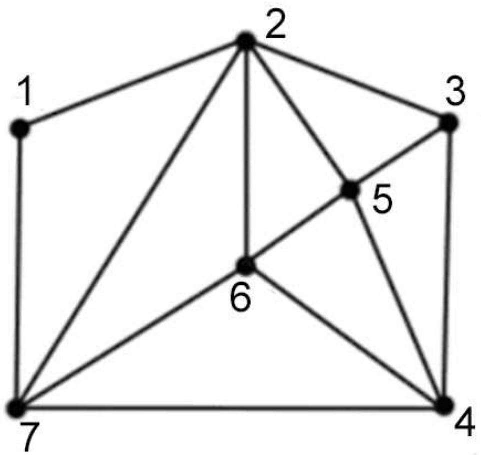
```
'''п р о г р а м а'''
Curcol = 1
Nmax=100
 '''Вводимо матрицю суміжності графа'''
 Degforming  #Формування масиву степенів вершин
 Sortnodes   #Формування масиву відсортованих вершин Sortarr
 for n in range(1, Nmax+1):
     If Colarr [Sortarr [n]] ==0:
        Colarr [Sortarr [n]] = Curcol
        Color (Sortarr [n])
        Inc (Curcol)
'''Виводимо результат розфарбування'''
```

```
def Degcount (m):
    Deg = 0;
    for k in range(1, Nmax+1):
        Deg = Deg + A [k] [m]
    return = Deg

def Degforming:
'''Процедура формування масиву степенів вершин'''
   for j  in range(1,Nmax+1):
      Colarr [i] = 0
      Degarr [j] = Degcount (j) * 100
    for i in range(1,Nmax+1):
        If A [i][ j] == 1 :
            Degarr [i]+=Degcount (i)
```

```
def Sortnodes: #Сортування вершин за степенями
  for k in range(1, Nmax):
      max = Degarr [k]
      c = k;
      for i in range(k + 1, N+1):
         if Degarr [i]> max then
            max = Degarr [i];
            c = i;
   Degarr [c] = Degarr [k];
   Degarr [k] = max;
   Sortarr [k] = c;


def Color (i):
  ''' Розфарбування поточним кольором НЕ суміжних з і вершин'''
  for j in range(1, Nmax+1):
       if A [j, i]== 0:
          If Colarr [j] == 0:
              Colarr [j] = Curcol;
```

ПРИКЛАД МОДИФІКОВАНОГО ЕВРИСТИЧНОГО АЛГОРИТМУ РОЗФАРБУВАННЯ

Дано граф G, зображений на рисунку



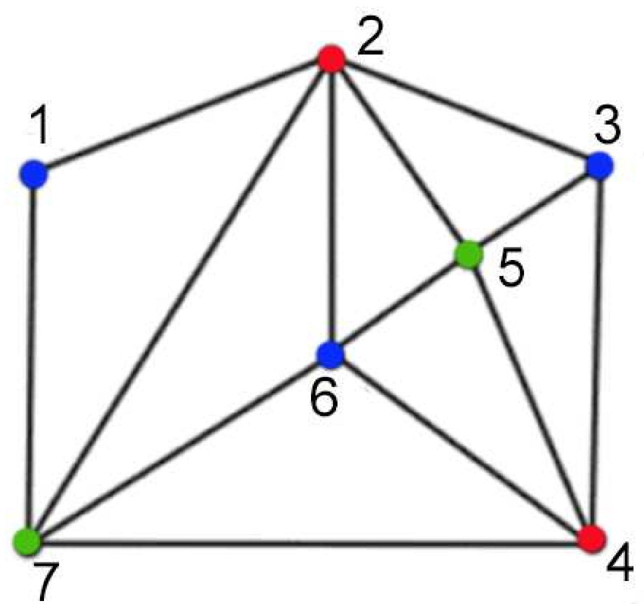
Номери вершин X^*	2	6	5	4	7	3	1
Степінь вершин D	5	4	4	4	4	3	2
Двокроковий степінь D^2	17	17	16	15	15	13	9
DegArr	517	417	416	415	415	313	209
$\mathrm{CurCol} = 1$	1	-	-	1	-	-	-
$\mathrm{CurCol} = 2$	1	2	-	1	-	2	2
$\mathrm{CurCol} = 3$	1	2	3	1	3	2	2

Множину вершин графа $V = \{1, 2, 3, 4, 5, 6, 7\}$ потрібно розфарбувати з використанням модифікованого евристичного алгоритму розфарбування.

Крок 1. SortArr[1]=2 фарбуємо червоним кольором 1. Несуміжна вершина 4. Також фарбуємо червоним.

Крок 2. SortArr[2]=6 фарбуємо синім кольором 2. Несуміжні 1 та 3 фарбуємо також синім кольором 2.

Крок 3. SortArr[3]=5 фарбуємо зеленим кольором 3. Несуміжна вершина 7. Фарбуємо також зеленим кольором 3.



Номери вершин X^*	2	6	5	4	7	3	1
Степінь вершин D	5	4	4	4	4	3	2
Двокроковий степінь D^2	17	17	16	15	15	13	9
DegArr	517	417	416	415	415	313	209
$\mathrm{CurCol}=1$	1	-	-	1	-	-	-
$\mathrm{CurCol}=2$	1	2	-	1	-	2	2
$\mathrm{CurCol}=3$	1	2	3	1	3	2	2

Розфарбування графа за методом А. П. Єршова

Андрій Петрович Єршов (1931–1988 рр.), визначний учений в області теоретичного програмування, зробив великий внесок у розвиток інформатики в нашій країні. Ним створений алгоритм розфарбування графа, що вирізняється оригінальною евристичною ідеєю.

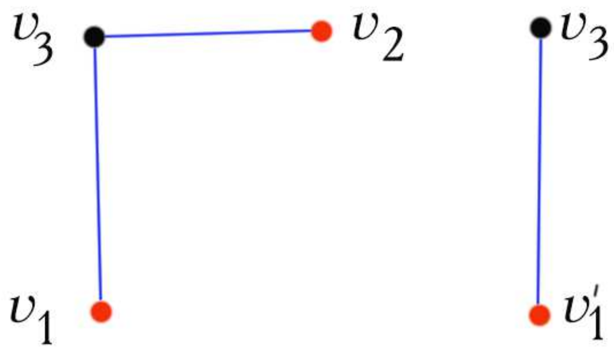
Введемо ряд визначень.

Окіл 1-го порядку. Для даної вершини $v \in V$ графа $G(V,E)$ всі суміжні з нею вершини називають околом 1-го порядку — $R_1(v)$.

Окіл 2-го порядку. Усі вершини, які перебувають на відстані два від v, називають околом 2-го порядку — $R_2(v)$.

Граф $G(V,E)$, у якого для вершини $v \in V$ всі інші вершини належать околу $R_1(v)$, назвемо граф-зіркою відносно вершини v.

Ідея алгоритму полягає у послідовному склеюванні вершин, що входять в окіл другого порядку.



Приклад об'єднання двох вершин: $v_1' := v_1 \cup v_2$

Графічно фарбування вершин v_1 і v_2 у один колір можна відобразити як «склеювання» цих вершин.

Отже, «склеювання» зменшує на одиницю кількість вершин у графі G і зменшує кількість ребер.

Алгоритм А. П. Єршова

1.Встановити $i := 0$.

2.Вибрати в графі G довільну нерозфарбовану вершину v.

3.Встановити $i := i + 1$.

4.Розфарбувати вершину v у колір i.

5.Розфарбовувати у колір i нерозфарбовані вершини графа G, вибираючи їх з $R_2(v)$ та склеюючи їх з вершиною v , поки граф не перетвориться в граф-зірку відносно v.

6.Перевірити, чи залишилися нерозфарбовані вершини в графі G. Якщо так, то перейти до п. 2, інакше – до п. 7.

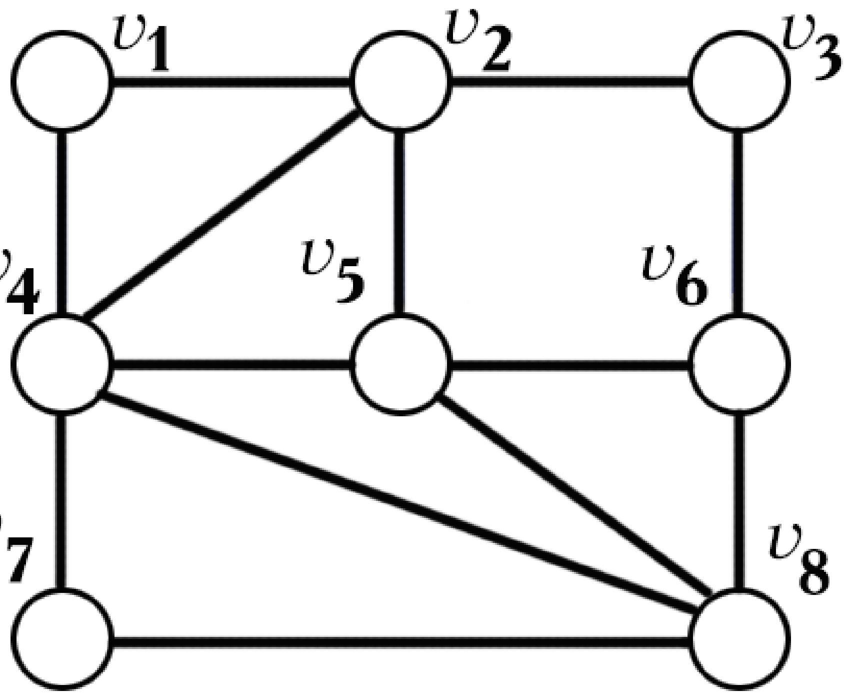
7.Отриманий повний граф K_i. Хроматичне число графа $X(K_i) = i$.

Кінець алгоритму.

Приклад розфарбування методом А. П. Єршова

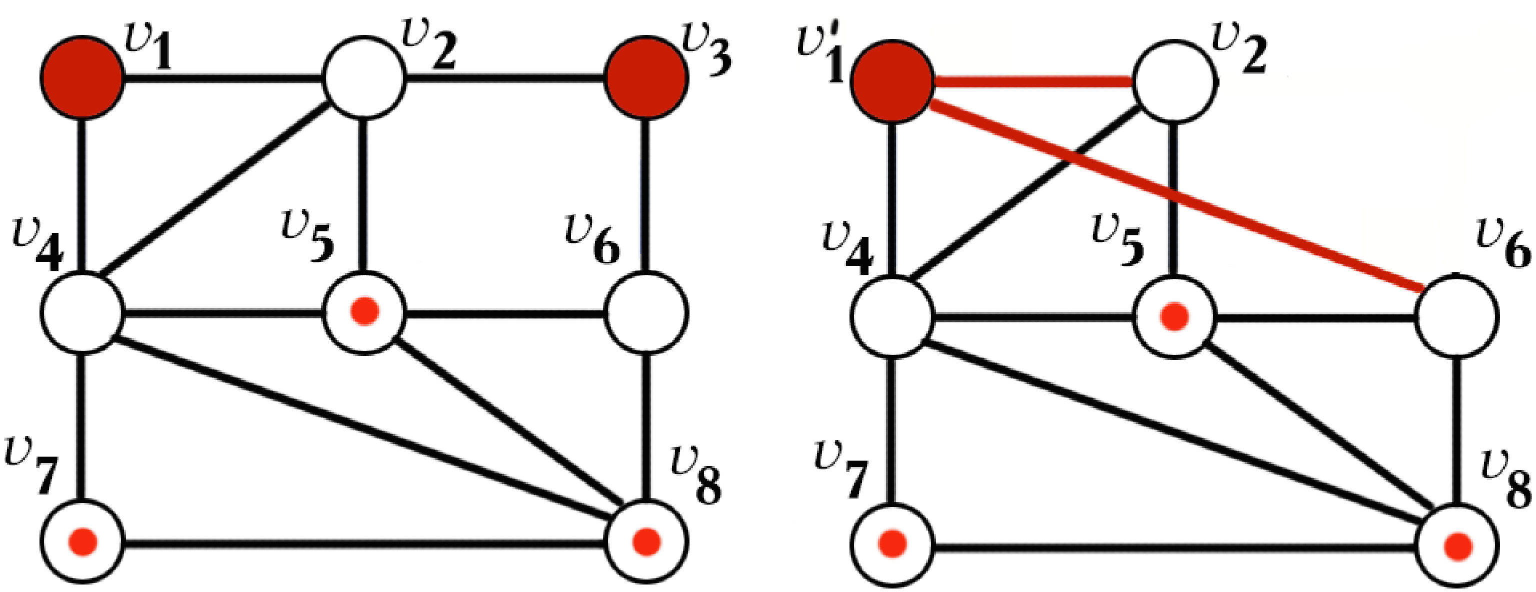
ПРИКЛАД РОЗФАРБУВАННЯ ГРАФА ЗА МЕТОДОМ А. П. ЄРШОВА

Розглянемо граф G:

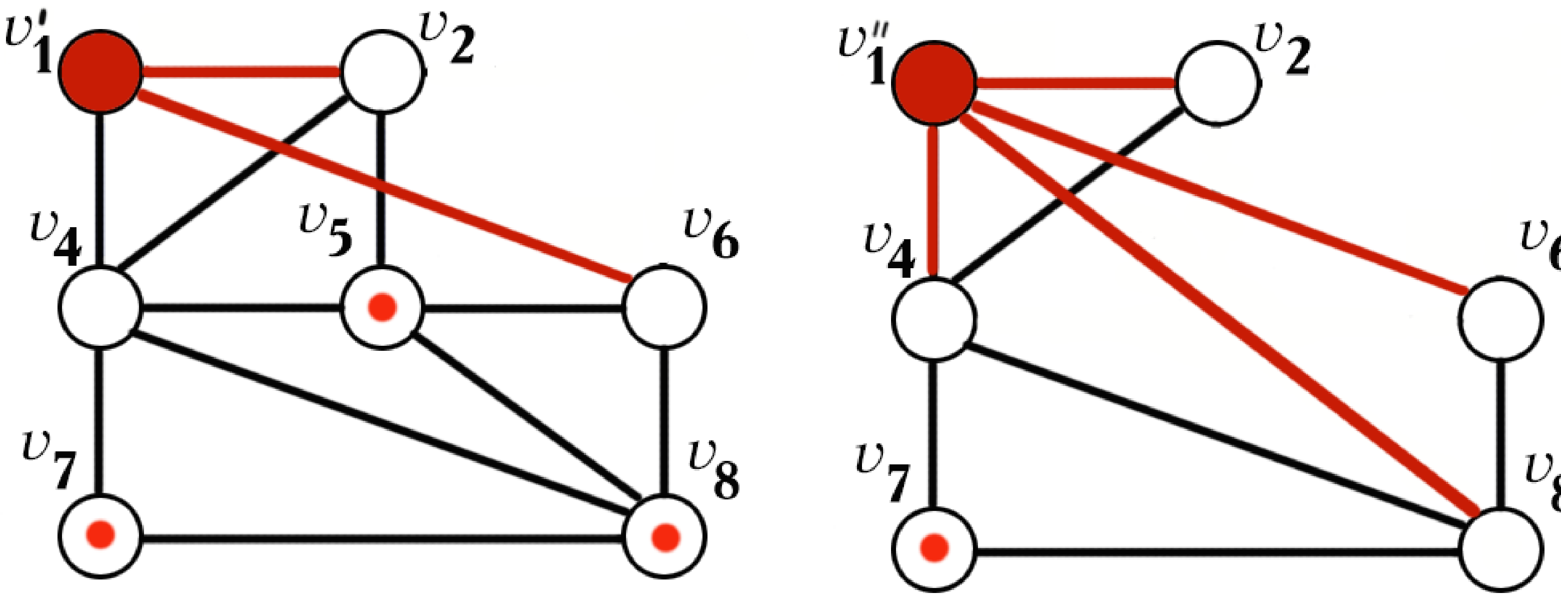


Крок 1. Виберемо довільну вершину, наприклад, v_1. Окіл 2-го порядку $R_2(v_1) = \{v_3, v_5, v_7, v_8\}$.

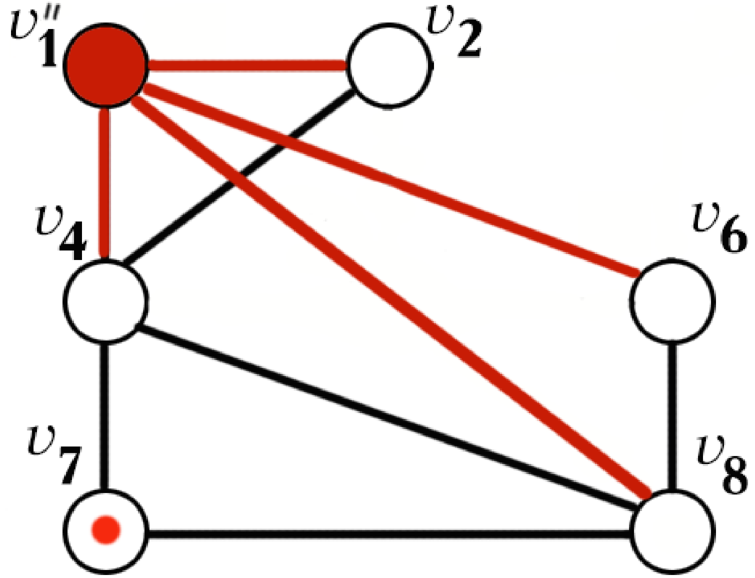
Склеїмо вершину v_1, наприклад, з вершиною v_3: $v_1' = v_1 \cup v_3$. Одержимо граф G_1 зображений на рисунку

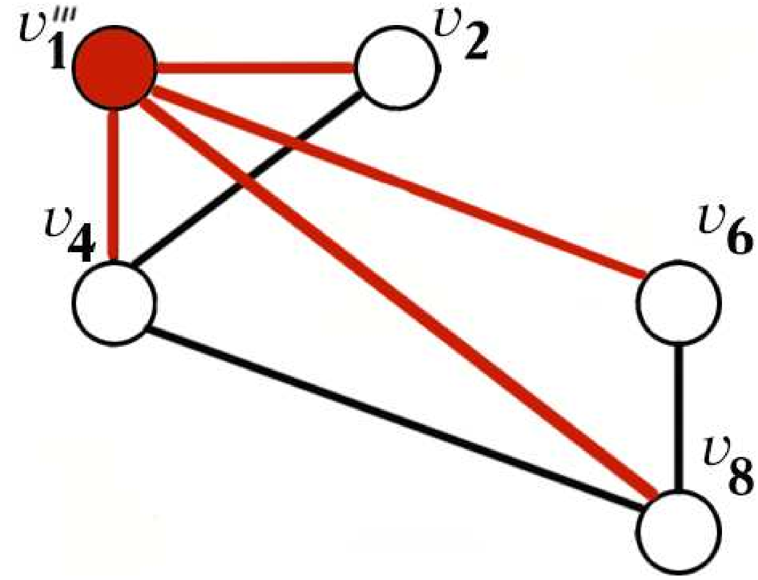


Крок 2. Окіл другого порядку вершини v_1' визначається множиною $R_2\left(v_1'\right)=\left\{v_5.v_7,v_8\right\}$. Склеїмо вершину v_1', наприклад, з вершиною v_5: $v_1'' := v_1' \cup v_5$.

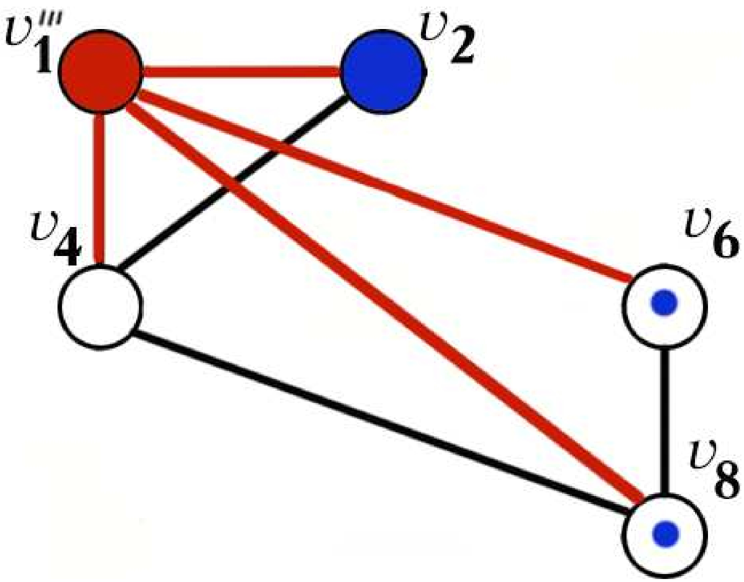


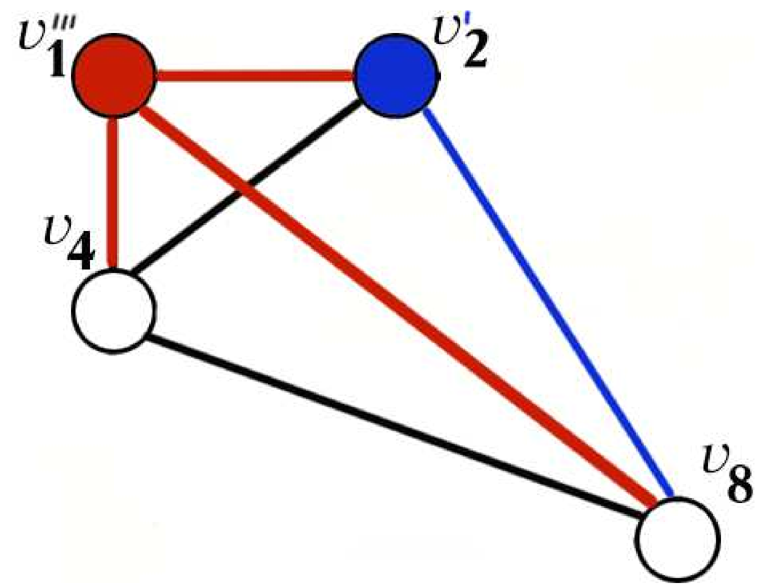
Крок 3. Окіл другого порядку для вершини v_1'' : $R_2\left(v_1''\right)=\left\{v_7\right\}$. Склеїмо вершину v_1'' з вершиною v_7: $v_1'''=v_1''\cup v_7$.





Крок 4. Окіл 2-го порядку $R_2\left(v_2\right)=\left\{v_6, v_8\right\}$. Склеїмо вершину v_2, наприклад, з вершиною v_6: $v_2'' = v_2 \cup v_6$.





Результат

У підсумку одержуємо правильно розфарбований граф, показаний на рисунку.