Лекція 15

План лекції

1. **Базові відомості**
2. **Алгоритми прямого неявного перебору**
3. **Приклад алгоритму прямого неявного перебору**
4. **Евристичний алгоритм розфарбування**
5. **Приклад евристичного алгоритму розфарбування**
6. **Модифікований евристичний алгоритм розфарбування**
7. **Приклад модифікованого евристичного алгоритму розфарбування**
8. **Розфарбування графа методом Єршова.**
9. **Приклад розфарбування графа методом Єршова**
10. **Рекурсивна процедура послідовного розфарбування**
11. **Приклад роботи рекурсивної процедури послідовного розфарбування.**
12. **«Жадібний» алгоритм розфарбування.**
13. **Приклад роботи «жадібного» алгоритму розфарбування**
14.

Основні алгоритми розфарбування графів

1. Базові відомості.

Різноманітні завдання, що виникають при плануванні виробництва, складанні графіків огляду, зберіганні та транспортуванні товарів та ін., часто можуть бути представлені як задачі теорії графів, тісно пов'язані з так званим «завданням розфарбовування». Графи, що розглядаються в даній лабораторній роботі, є неорієнтованими і такими, що не мають петель.

Граф G називають r-*хроматичним*, якщо його вершини *можуть бути розфарбовані* з використанням r кольорів (фарб) так, що не знайдеться двох суміжних вершин одного кольору. Найменше число r, таке, що граф G є $r-$*хроматичним*, називається *хроматичним числом* графа G і позначається $\gamma(G)$. Завдання знаходження хроматичного числа графа називається *задачею про розфарбовування* (або *завданням розфарбовування*) графа. Відповідне цьому числу розфарбування вершин розбиває множину вершин графа на r підмножин, кожна з яких містить вершини одного кольору. Ці множини є незалежними, оскільки в межах однієї множини немає двох суміжних вершин.

Завдання знаходження хроматичного числа довільного графа стало предметом багатьох досліджень в кінці XIX і в XX столітті. З цього питання отримано багато цікавих результатів.

Хроматичне число графа не можна знайти, знаючи тільки кількість вершин і ребер графа. Недостатньо також знати степінь кожної вершини, щоб обчислити хроматичне число графа. При відомих величинах n (кількість вершин), m (кількість ребер) і $\deg(x_1),...,\deg(x_n)$ (степені вершин графа) можна отримати тільки верхню і нижню оцінки для хроматичного числа графа.

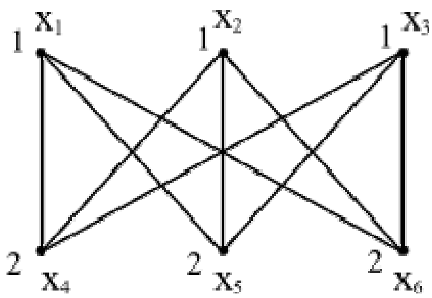


Рис. 5.1 – Дводольний біхроматичний граф Кеніга.

Приклад розфарбовування графа наведений на рисунку 5.1. Цей граф є однією із заборонених фігур, що використовуються для визначення планарності. Цифрами «1» і «2» позначені кольори вершин.

Максимальна кількість незалежних вершин графа□$\alpha(G)$□, що дорівнює потужності найбільшої множини попарно несуміжних вершин, збігається також з потужністю найбільшої множини вершин в G, які можуть бути пофарбовані в один колір, отже:

$$\gamma(G) \geq \left\lceil \frac{n}{\alpha(G)} \right\rceil, \tag{5.1}$$

де n - кількість вершин графа G, а $\lceil x \rceil$ позначає найбільше ціле число, яке не більше за x.

Ще одна нижня оцінка для $\gamma(G)$ може бути отримана наступним чином:

$$\gamma(G) \geq \frac{n^2}{n^2 - 2m}. \tag{5.2}$$

Верхня оцінка хроматичного числа може бути обчислена за формулою:

$$\gamma(G) \le 1 + \max_{x_j \in X}\left[d\left(x_j\right) + 1 \right]. \qquad (5.3)$$

Застосування оцінок для хроматичного числа значно звужує межі рішення. Для визначення оцінки хроматичного числа також можуть використовуватися інші топологічні характеристики графа, наприклад, властивість планарності.

Граф, який можна зобразити на площині так, що жодні два його ребра не перетинаються між собою, називається *планарним.*

Теорема про п'ять фарб. Кожен планарний граф можна розфарбувати за допомогою п'яти кольорів так, що будь-які дві суміжні вершини будуть пофарбовані в різні кольори, тобто якщо граф G - планарний, то $\gamma(G) \le 5$.

Гіпотеза про чотири фарби (недоведена). Кожен планарний граф можна розфарбувати за допомогою чотирьох кольорів так, що будь-які дві суміжні вершини будуть пофарбовані в різні кольори, тобто якщо граф G– планарний, то $\gamma(G) \le 4$.

У 1852 р. про гіпотезу чотирьох фарб говорилося в листуванні Огюста де Моргана з сером Вільямом Гамільтоном. З того часу ця «теорема» стала, поряд з теоремою Ферма, однією з найзнаменитіших невирішених задач в математиці.

Повний граф K_n завжди розфарбовується в n кольорів, тобто кількість кольорів дорівнює кількості його вершин.

2. АГОРИТМ ПРЯМОГО НЕЯВНОГО ПЕРЕБОРУ

Алгоритм прямого неявного перебору є найпростішим алгоритмом вершинного розфарбування графів. Цей алгоритм дозволяє реалізувати правельне розфарбування графа з вибором мінімальної в рамках даного алгоритму кількості фарб.

Введемо такі структури даних:

```
Const Nmax=100; {*максимальна кількість вершин графа*}

Type V=0..Nmax;

   TS=Set of V;

      TColArr = Array (1..Nmax) of V;

      TA = Array (1..Nmax, 1..Nmax) of Integer;
```

```
Var ColArr: TColArr; {*масив номерів фарб для кожної вершини графа*}
        A:TA; {*матриця суміжності графа*}

Function Color (i): Integer;
{*функція вибору фарби для розфарбування вершини з номером i *}
Var W:TS;
   j:Byte;
Begin
  W:=[];
  For j=1 to i-1 do if A[j,i]=1 then W:=W+[ColArr[j]];
 {*формування множини фарб, що використані для розфарбування суміжних до
  вершини i вершин з номерами меншими за i*}
 j:=0; {*змінну j далі використовуємо для вибору номера фарби*}
  Repeat
    Inc(j);
   Until NOT (j In W);
   Color:=j;
End;
Begin
 <Вводимо матрицю суміжності графа>
 {*цикл по вершинах графа*}
  For i=1 to Nmax do ColArr[i]:=Color(i);
  <Виводимо результат розфарбування>
End;
```

3. ПРИКЛАД АГОРИТМУ ПРЯМОГО НЕЯВНОГО ПЕРЕБОРУ

1. Розглянемо граф $G(V,E)$, який показаний на рис. 5.2.

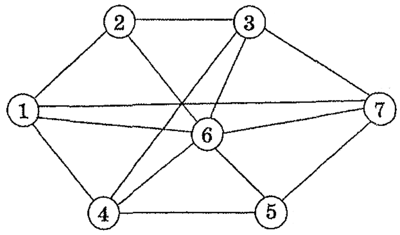


Рис.5.2

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

Сформуємо матрицю суміжності A :

$$A=\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

Крок1. Для вершини 1, відповідно до представленого вище алгоритму, множина розфарбованих суміжних вершин завжди є пустою. Тому функція Color(1) завжди повертатиме фарбу 1. Встановимо, що 1 кодує фарбу червоного кольору.

Крок2. Розглянемо вершину 2. Для цієї вершини єдиною меншою за номером суміжною вершиною є вершина 1. Ця вершина розфарбована червоним кольором. Тому множина W містить єдиний елемент 1. Тому функція Color(2) повертає наступну за номером фарбу 2 синього кольору.

Крок3. Вершина 3 має єдину суміжну вершину з меншим номером. Це вершина 2. Множина W містить єдиний елемент 2. Тому функція Color(3) повертає фарбу з номером 1 червоного кольру.

Крок4. Вершина 4 має дві суміжні вершини з меншими номерами:1 і 3. Оскільки обидві вершини розфарбовані в колір 1, то множина W містить єдиний елемент 1. Тому функція Color(4) повертає наступну за номером фарбу 2 синього кольору.

Крок5. Вершина 5 має єдину суміжну вершину з меншим номером. Це вершина 4. Множина W містить єдиний елемент 2. Тому функція Color(5) повертає фарбу з номером 1 червоного кольру.

Крок6. Вершина 6 має такі суміжні вершини з меншими номерами:1,3,4 і 5. Ці вершини розфарбовані в колір 1 та колір 2. Отже множина W містить два елементи: 1 і 2. Тому функція Color(6) повертає наступну за номером фарбу 3 зеленого кольору.

Крок7. Вершина 7 має такі суміжні вершини з меншими номерами:1,3, 5 і 6. Ці вершини розфарбовані в колір 1 та колір 3. Отже множина W містить два елементи: 1 і 2. Тому функція Color(7) повертає фарбу 2 синього кольору.

В результаті роботи данного алгоритму одержуємо правильно розфарбований граф, що показаний на рис. 5.3.

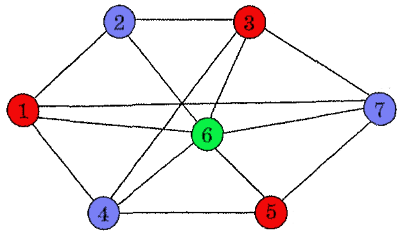


Рис. 5.3.

4. ЕВРИСТИЧНИЙ АЛГОРИТМ РОЗФАРБОВУВАННЯ

Точні методи розфарбовування графа складні для програмної реалізації. Однак існує багато евристичних процедур розфарбовування, які дозволяють знаходити хороші наближення для визначення хроматичного числа графа. Такі процедури також можуть з успіхом використовуватися при розфарбовуванні графів з великим числом вершин, де застосування точних методів не виправдане з огляду на високу трудомісткість обчислень.

З евристичних процедур розфарбовування слід зазначити послідовні методи, засновані на впорядкуванні множини вершин. В одному з найпростіших методів вершини спочатку розташовуються в порядку зменшення їх степенів. Перша вершина зафарбовується в колір 1, потім список вершин переглядається за зменшенням степенів, і в колір 1 зафарбовується кожна вершина, яка не є суміжною з вершинами, зафарбованими в той же колір. Потім повертаємося до першої в списку незафарбованої вершині, фарбуємо її в колір 2 і знову переглядаємо список вершин зверху вниз, зафарбовуючи в колір 2 будь-яку незафарбовану вершину, яка не з'єднана ребром з іншою, вже пофарбованою в колір 2 вершиною. Аналогічно діємо із кольорами 3, 4 і т. д., допоки не будуть пофарбовані всі вершини. Кількість використаних кольорів буде тоді наближеним значенням хроматичного числа графа.

Евристичний алгоритм розфарбовування вершин графа має наступний вигляд:

Крок 1. Сортувати вершини графа за степенями зменшення:

$$\deg(x_i) \geq \deg(x_j),\ \forall x_i, x_j \in G.$$

Встановити поточний колір $p := 1, i := 1$.

Крок 2. Вибрати чергову нерозфарбовану вершину зі списку і призначити їй новий колір $\mathrm{col}(x_i) := p;\ X = \{x_i\}$.

Крок 3. $i := i + 1$. Вибрати чергову не розфарбовану вершину x_i і перевірити умову суміжності: $x_i \cap \Gamma(X) = \varnothing$, де X - множина вершин, вже розфарбованих в колір p. Якщо вершина x_i не є суміжною з даними вершинами, то також присвоїти їй колір p: $\mathrm{col}(x_i) := p$.

Крок 4. Повторювати крок 3 до досягнення кінця списку $(i = n)$.

Крок 5. *Якщо* всі вершини графа розфарбовані, *то* – кінець алгоритму;

інакше: $p := p + 1;\ i := 1$. Повторити крок 2.

Для роботи алгоритму можна використовувати довільну структуру даних, яка однозначно задає граф.

Розглянемо роботу алгоритму на прикладі матриці суміжності A.

```
Const Nmax=100; {*максимальна кількість вершин графа*}
Type  TArr = Array (1..Nmax) of Byte;
        TA = Array (1..Nmax, 1..Nmax) of Byte;

Var ColArr: TArr; {*масив номерів фарб для кожної вершини графа*}
   DegArr: TArr {*масив степенів вершин*}
       SortArr:TArr;{*відсортований за зменшенням степенів масив вершин*}
       A:TA; {*матриця суміжності графа*}
    CurCol: Byte; {*поточний номер фарби*}
   n:Byte;

Procedure DegForming;{*Процедура формування масиву степенів вершин*}
Var i:Byte;
```

```
Begin
  For i:=1 to Nmax do
  begin
   DegArr[i]:=0; ColArr[i]:=0;
   For j:=1 to Nmax do
   DegArr[i]:= DegArr[i]+A[i,j];
  end;
End;
Procedure SortNodes; {*Сортування вершин за степенями*}
Var max,c,k,i:Byte;
Begin
 For k:=1 to Nmax-1 do
 begin
  max:=DegArr[k]; c:=k;
  For i:=k+1 to N do
  If DegArr[i] > max then
  begin
   max:= DegArr[ [i];
   c:=i;
  end;
  DegArr[c]:= DegArr[ [k];
  DegArr[k]:=max;
  SortArr[k]:=c;
 end;
End;
Procedure Color (i:Byte);
{*Розфарбування поточним кольром не суміжних з і вершин *}
Var j:Byte;
Begin
 For j=1 to Nmax do if A[j,i]=0 then
 begin
  If ColArr[j]=0 then ColArr[j]:=CurCol;
 end;
```

```
End;
Begin
 CurCol:=1;
 <Вводимо матрицю суміжності графа>
 DegForming; {*Формування масиву степенів вершин*}
 SortNodes;  {*Формування массиву відсортованих вершин SortArr*}
 For n:=1 to Nmax do
 begin
  If ColArr[SortArr[n]]=0 then
  begin
   ColArr[SortArr[n]]:=CurCol;
   Color(SortArr[n]);
   Inc(CurCol);
  end;
 end;
<Виводимо результат розфарбування>
end;
```

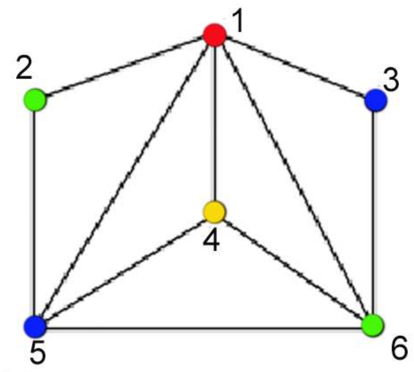
5. ПРИКЛАД ЕВРИСТИЧНОГО АЛГОРИТМУ РОЗФАРБУВАННЯ

Розфарбуємо граф G, зображений на рисунку 5.4. Проміжні дані для вирішення завдання будемо записувати в таблицю. Відсортуємо вершини графа за зменшенням їх степенів. У результаті отримуємо вектор відсортованих вершин $\mathrm{SortArr} = (1,5,6,4,2,3)$

Степені, що відповідають даним вершинам, утворюють другий вектор: $D = (5,4,4,3,2,2)$

У першому рядку таблиці запишемо вектор SortArr , у другому – степені відповідних вершин. Наступні рядки відображають вміст вектора розфарбування $\mathrm{ColArr}[\mathrm{SortArr}]$.

Таким чином, даний граф можна розфарбувати не менш ніж у чотири кольори, тобто $\gamma(G) = 4$.



Номери вершин SortArr	x_1	x_5	x_6	x_4	x_2	x_3
Степені вершин DegArr	5	4	4	3	2	2
$\mathrm{CurCol}=1$	1	-	-	-	-	-
$\mathrm{CurCol}=2$	1	2	-	-	-	2
$\mathrm{CurCol}=3$	1	2	3	-	3	2
$\mathrm{CurCol}=4$	1	2	3	4	3	2

Рис. 5.4.

6. МОДИФІКОВАНИЙ ЕВРИСТИЧНИЙ АЛГОРИТМ РОЗФАРБУВАННЯ

Попередні визначення

Визначення 1. Відносний ступінь – це ступінь нерозфарбованих вершин у нерозфарбованому підграфі даного графа.

Визначення 2. Двокроковий відносний ступінь – сума відносних степенів суміжних вершин у нерозфарбованому підграфі.

Проста модифікація описаної вище евристичної процедури базується на переупорядкуванні нерозфарбованих вершин по незростанню їх відносних степенів.

Дана модифікація полягає у тому, що якщо дві вершини мають однакові степені, то порядок таких вершин випадковий. Їх можна впорядкувати по двокрокових степенях. Двокроковий ступінь визначимо як суму відносних степенів суміжних вершин.

Крок1. Сортуємо вершини графа за степенями зменшення:

$$\deg(x_i) \geq \deg(x_j),\ \forall x_i, x_j \in G.$$

У випадку $\deg(x_i) = \deg(x_j),\ \forall x_i, x_j \in G$ розглянемо множини суміжності $\Gamma(x_i)$ і $\Gamma(x_j)$.

Сортуємо вершини за ознакою :

$$[\deg(x_{i1}) + \deg(x_{i2}) + \ldots + \deg(x_{ik})] \geq [\deg(x_{j1}) + \deg(x_{j2}) + \ldots + \deg(x_{jn})]\ ,$$

де $x_{i1}, x_{i2}, \ldots, x_{ik}$ - нерозфарбовані вршини з множини суміжності $\Gamma(x_i)$;

$x_{j1}, x_{j2}, ..., x_{jn}$ - нерозфарбовані вршини з множини суміжності $\Gamma(x_j)$;

Встановити поточний колір $p := 1, i := 1$.

Крок 2. Вибрати чергову нерозфарбовану вершину зі списку і призначити їй новий колір $\mathrm{col}(x_i) := p;\ X = \{x_i\}$.

Крок 3. $i := i + 1$. Вибрати чергову нерозфарбовану вершину x_i і перевірити умову суміжності: $x_i \cap \Gamma(X) = \varnothing$, де X - множина вершин, вже розфарбованих в колір p. Якщо вершина x_i не є суміжною з даними вершинами, то також присвоїти їй колір p: $\mathrm{col}(x_i) := p$.

Крок 4. Повторювати крок 3 до досягнення кінця списку $(i = n)$.

Крок 5. *Якщо* всі вершини графа розфарбовані, *то* – кінець алгоритму;

інакше: $p := p + 1;\ i := 1$. Повторити крок 2.

Даний алгоритм від попереднього відрізняється ускладненням процедури сортування SortNodes, яка при сортуванні вершин з однаковими степенями враховує двукроковий ступінь.

Як і в попередньому випадку, розглянемо роботу алгоритму на прикладі матриці суміжності A.

Const Nmax=100; {*максимальна кількість вершин графа*}

Type TArr = **Array** (1..Nmax) of Integer;

TA = **Array** (1..Nmax, 1..Nmax) of Byte;

Var ColArr: TArr; {*масив номерів фарб для кожної вершини графа*}

DegArr: TArr {*масив степенів вершин*}

SortArr:TArr;{*відсортований за зменшенням степенів масив вершин*}

A:TA; {*матриця суміжності графа*}

CurCol: Byte; {*поточний номер фарби*}

n:Byte;

Procedure DegForming;{*Процедура формування масиву степенів вершин*}

Var k:Byte;

```
Function DegCount(m:Byte):Integer;
Var Deg:Iteger;
Begin
 Deg:=0;
 For k:=1 to Nmax do Deg:= Deg+A[k,m];
 DegCount:=Deg;
End;
Begin
 For j:=1 to Nmax do
 begin
  ColArr[i]:=0;
  DegArr[j]:= DegCount(j)*100;
  For i:=1 to Nmax do
  If A[i,j]=1 then DegArr[i]:= DegArr[i]+DegCount(i);
 end;
End;


Procedure SortNodes; {*Сортування вершин за степенями*}
Var max,c,k,i:Byte;
Begin
 For k:=1 to Nmax-1 do
 begin
  max:=DegArr[k]; c:=k;
  For i:=k+1 to N do
  If DegArr[i] > max then
  begin
   max:= DegArr[ [i];
   c:=i;
  end;
  DegArr[c]:= DegArr[ [k];
  DegArr[k]:=max;
  SortArr[k]:=c;
 end;
```

```
End;
Procedure Color (i:Byte);
{*Розфарбування поточним кольром не суміжних з і вершин *}
Var j:Byte;
Begin
 For j=1 to Nmax do if A[j,i]=0 then
 begin
  If ColArr[j]=0 then ColArr[j]:=CurCol;
 end;
End;
Begin
 CurCol:=1;
 <Вводимо матрицю суміжності графа>
 DegForming; {*Формування масиву степенів вершин*}
 SortNodes;  {*Формування массиву відсортованих вершин SortArr*}
 For n:=1 to Nmax do
 begin
  If ColArr[SortArr[n]]=0 then
  begin
   ColArr[SortArr[n]]:=CurCol;
   Color(SortArr[n]);
   Inc(CurCol);
  end;
 end;
<Виводимо результат розфарбування>
end;
```

7. ПРИКЛАД МОДИФІКОВАНОГО ЕВРИСТИЧНОГО АЛГОРИТМУ РОЗФАРБУВАННЯ

Розфарбуємо граф G , зображений на рисунку 5.3

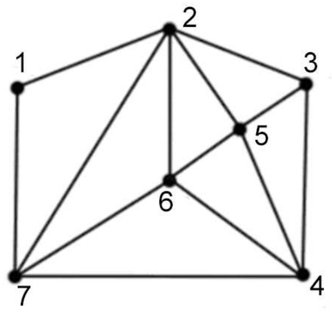


Рис. 5.5.

Відсортуємо вершини графа за зменшенням їх степенів. У результаті отримуємо вектор відсортованих вершин $\text{SortArr} = (2,6,5,4,7,3,1)$

Вектор степенів відсортованих вершин має наступний вигляд: $D = (5,4,4,4,4,3,2)$

У першому рядку таблиці запишемо вектор SortArr, у другому – вектор D, а у третьому – вектор D^2 .

Четвертий рядок відповідає представленню степенів D та D^2 в масиві DegArr.

Наступні рядки відображають вміст вектора розфарбування $\text{col}\left(X^*\right)$.

Таким чином, даний граф можна розфарбувати не менш ніж у три кольори, тобто $\gamma(G) = 3$.

Номери вершин X^*	a_2	a_6	a_5	a_4	a_7	a_3	a_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
$\text{CurCol} = 1$	1	-	-	1	-	-	-
$\text{CurCol} = 2$	1	2	-	1	-	2	2
$\text{CurCol} = 3$	1	2	3	1	3	2	2

В результаті роботи модифікованого евристичного алгоритму одержимо розфарбований граф, показаний на рис. 5.6.

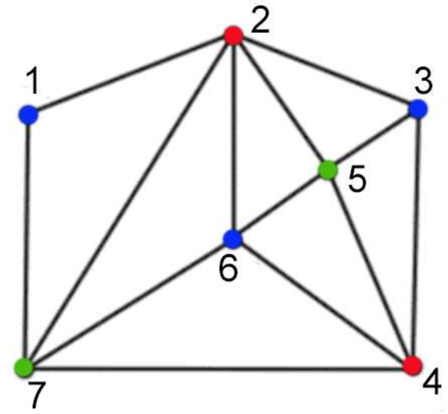


Рис. 5.6.

8. РОЗФАРБУВАННЯ ГРАФА МЕТОДОМ А.П. ЄРШОВА

Андрій Петрович Єршов (1931-1988 рр.), видатний вчений в області теоретичного програмування, вніс великий вклад у розвиток інформатики. Зокрема, він створив алгоритм розфарбування графа, що базується на оригінальній евристичній ідеї.

Введемо ряд визначень.

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

Всі вершини, що перебувають на відстані два від v, назвемо 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

Фарбування у фарбу α вершини v утворює навколо неї в $R_1(v)$ «мертву зону» для фарби α. Очевидно, при мінімальному розфарбуванні кожна фарба повинна розфарбувати максимальну можливу кількість вершин графа. Для цього необхідно, щоб мертві зони, хоча б частково, перекривалися між собою. Перекриття мертвих зон двох несуміжних вершин v_1 і v_2

досягається тільки тоді, коли одна з них перебуває в околі $R_2(v_1)$ від іншої, $v_2 \in R_2(v_1)$.

Таким чином, суть алгоритму полягає в тому, щоб на черговому кроці вибрати для розфарбування фарбою α вершину $v_2 \in R_2(v_1)$. Цей процес повторювати доти, поки фарбою α не будуть пофарбовані всі можливі вершини графа.

Графічно фарбування вершин v_1 і v_2 однією фарбою можна відобразити як «склеювання» цих вершин.

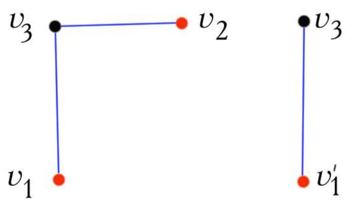


Рис. 14.1. Приклад об'єднання двох вершин: $v_1' := v_1 \cup v_2$

При цьому кількість вершин зменшується на одиницю у графі G, а також зменшується кількість ребер.

Алгоритм

1.Встановити $i := 0$.

2.Вибрати в графі G довільну незафарбовану вершину v.

3.Встановити $i := i + 1$.

4.Розфарбувати вершину v фарбою i.

5.Розфарбовувати фарбою i незабарвлені вершини графа G, вибираючи їх з $R_2(v)$, поки граф не перетвориться в граф-зірку відносно v .

6.Перевірити, чи залишилися незабарвлені 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.

Як випливає з алгоритму, після того, як у першу обрану вершину стягнуті всі можливі й граф перетворився в граф-зірку відносно цієї вершини, вибирається довільним чином друга вершина й процес повторюється доти, поки граф не перетвориться в повний.

Повний граф - це граф-зірка відносно будь-якої вершини. Хроматичне число повного графа дорівнює числу його вершин.

9. ПРИКЛАД РОЗФАРБУВАННЯ ГРАФА МЕТОДОМ А.П. ЄРШОВА

На мал.14.2 зображений граф G, на прикладі якого будемо розглядати роботу евристичного алгоритму Єршова.

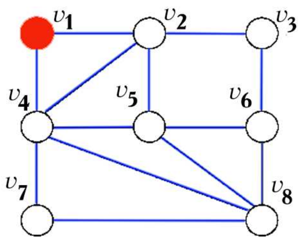


Рис.14.2. Початковий граф G

Виберемо довільну вершину, наприклад, v_1. Окіл 2-го порядку $R_2\left(v_1\right)=\left\{v_3,v_5,v_7,v_8\right\}$. Склеїмо вершину v_1 наприклад, з вершиною v_3: $v_1'=v_1\cup v_3$. Одержимо граф G_1 зображений на мал. 14.3.

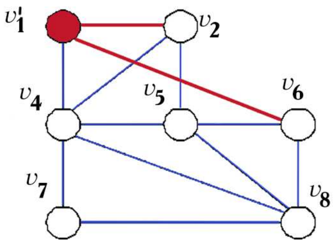


Рис.14.3. Граф G_1. Склеєні вершини v_1 та v_3

Розглянемо тепер граф G_1. 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: $v_1'':=v_1'\cup v_5$. Одержимо граф G_2, зображений на мал. 14.4.

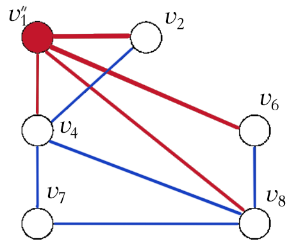


Рис.14.4. Граф G_2. Склеєні вершини v_1' й v_5

У графі G_2 визначимо 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: $v_1'''=v_1''\cup v_7$. Одержимо граф G_3, зображений на мал. 14.5. Цей граф є зіркою відносно вершини v_1'''.

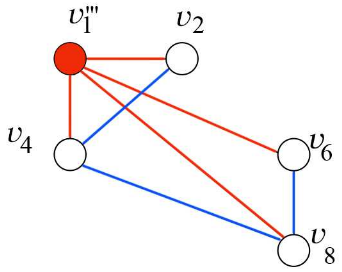


Рис.14.5. Граф G_3. Склеєні вершини v_1'' й v_7

У графі G_3 виберемо, наприклад, вершину v_2 для фарбування другою фарбою. 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: $v_2'=v_2\cup v_6$. Одержимо граф G_4, зображений на мал.14.6.

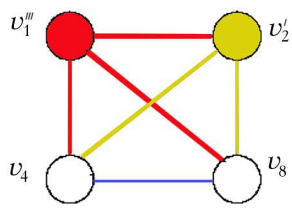


Рис.14.6. Граф G_4 еквівалентний K_4. Склеєні вершини v_1'' й v_6

Граф G4 є повним четиривершинним графом K_4 . Отже, граф G_4 на мал.14.6 розфарбовується в чотири кольори. У перший (червоний) колір розфарбовується вершина v_1 й склеєні з нею вершини: v_3, v_5 і v_7. У другий (жовтий) колір розфарбовується вершина v_2 й склеєна з нею вершина v_6. У третій (зелений) колір розфарбовується вершина v_4 й у четвертий (білий) колір розфарбовується вершина v_8.

У підсумку одержуємо правильно розфарбований граф, показаний на мал.14.7.

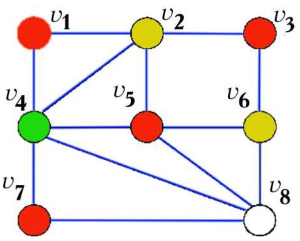


Рис.14.7.Граф G, розфарбований за допомогою алгоритму розфарбування А.П.Єршова

Програмна реалізація

Розглянемо матрицю суміжності графа G, представленого на рис. 14.2.

$$A=\begin{array}{c|cccccccc} & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 \\ \hline v_1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ v_2 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ v_3 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ v_4 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ v_5 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ v_6 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ v_7 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ v_8 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{array}$$

Операції склеювання відповідних вершин графа, які описні вище, виконуються на матриці суміжності шляхом виконання операції диз'юнкції між цими вершинами.

Наприклад, розглянемо склеювання вершин 1 та 3.

Результуюча матриця сумажності містить новий рядок та стовбець для вершини v_1', але не містить v_1 та v_3.

$$A'=\begin{array}{c|ccccccc} & v_1' & v_2 & v_4 & v_5 & v_6 & v_7 & v_8 \\ \hline v_1' & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ v_2 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ v_4 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ v_5 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ v_6 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ v_7 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ v_8 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{array}.$$

Фінальна матриця суміжності має вигляд:

$$A''' = \begin{array}{c c} & \begin{array}{cccc} v_1''' & v_2' & v_4 & v_8 \end{array} \\ \begin{array}{c} v_1''' \\ v_2' \\ v_4 \\ v_8 \end{array} & \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \end{array},$$

що відповідає матриці повного графа K_4 .

```
Const n=10; {*максимальна кількість вершин графа*}
Type V=0..n;
    R2S=Set of V;
        TA = Array (1..n, 1..n) of Integer;


Var   A:TA; {*матриця суміжності графа*}
   MainNode:Byte;


Procedure Glue(master,slave:Byte);
{*Процедура склеювання вершин *}
Begin
 {*Склеювання рядка master і рядка slave*}
 For i=1 to n do A[i,master]:= A[i,master] OR A[i,slave];
{*Склеювання стовбця master і стовбця slave*}
 For j=1 to n do A[master,j]:= A[master,j] OR A[slave,j];
End;

Procedure Reduce(master,slave:Byte);
{*Процедура видалення стовбця і рядка в матриці суміжності*}
Begin
 For i:=1 to n do
  For j:=1 to n-1 do
   {*Видалення рядка slave*}
   If (j≥slave) then A[i,j]:= A[i,j+1];
 For j:=1 to n-1 do
  For i:=1 to n-1 do
   {*Видалення стовбця slave*}
   If (i≥slave) then A[i,j]:= A[i+1,j];
 n:=n-1;
End;
Function Check_K:Byte;
{*Процедура перевірки наявності повного графа*}
Var Gh:Byte;
```

```
Begin
 {*У повному графі всі елементи матриці, крім діагональних, повинні бути одиничними*}
 Ch:=0;
 For i:=1 to n do
 For j:=1 to n do if (i≠ j ) AND (A[i,j]=0) then Ch:=j;
 Check_K:=Ch;
End;

Procrdure R2(master:Byte);
{*Процедура формування околу 2-го порядка*}
Begin
 For j:=1 to n do
 begin
  If (j≠master) AND (A[master,j]=1) then
  begin
     {*Вибрали суміжну вершину до master*}
   For i=1 to n do
   begin
    If (i≠master) AND (A[j,i]=1) then
    {* Вибрали  вершину околу 2-го порядку  до master*}
    R2S:=R2S+[i];  {*Додали вершину до множини вершин околу*}
   end;
  end;
 end;
End;

Begin
 MainNode:=1; {*Вибираємо першу вершину*}
 K_finded:=false; {*Ознака закінчення алгоритму*}
 While K_finded do
 begin
  R2S:=[]; {*Очистка множини околу 2-го порядку*}
  R2(MainNode); {*Формуємо окіл 2-го порядку для MainNode*}
  For k=1 to n do {*Цикл по вершинах графа*}
  begin
   If k in R2S then
   begin {*Вершина належить околу другого порядку*}
     {*Склеювання вершини MainNode з вершиною околу k *}
    Glue(MainNode,k);
    {*Модифікація матриці суміжності після склеювання вершин *}
    Reduce(MainNode,k);
   end;
  end;
  MainNode:= Check_K(MainNode);
```

```
  If MainNode=0 then K_finded:=true;
 end;
End;
```

10. РЕКУРСИВНА ПРОЦЕДЕРА ПОСЛІДОВНГО РОЗФАРБУВАННЯ

1. Фіксуємо порядок обходу вершин.

2. Ідемо по вершинах, використовуючи такий найменший колір, який не викличе конфліктів.

3. Якщо вже використаний колір вибрати не виходить, то тільки тоді вводимо новий колір.

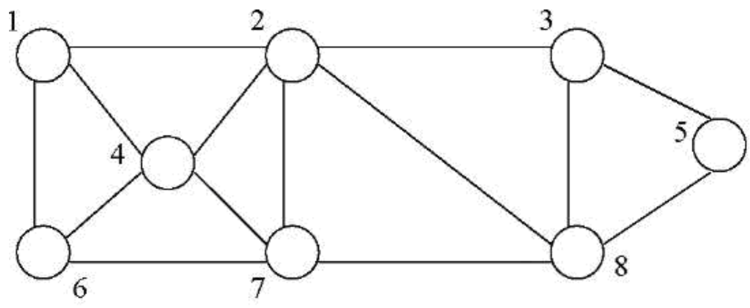
У процедурі використовується рекурсивний виклик процедури фарбування наступної вершини у випадку успішного фарбування попередньої вершини.

```
Const n=10;
     Cmax=10;
Type
        TA = Array (1..n, 1..n) of Byte;
        TArr = Array (1..n) of Byte;
Var i:Byte;
    color:TArr;
     A:TA;
     C:Byte;


procedure visit(i:Byte);
 Function Nicecolor:Boolean;
{*Функція пошуку неконфліктної фарби*}
 Var CN:Boolean;
     j:integer;
 Begin
  CN:=true;
  For j=1 to n do
```

```
 If (A[j,i]=1) AND (color[j]=c) then CN:=false;
 {*Знайдена суміжна вершин до вершини і. Ця вершина має поточний колір с .
     Тоді колір с є конфліктним*}
End;
begin
 if i = n + 1 then Print else
{*Якщо всі вершини розфарбовано, то виводимо результат*}
 begin
  If color[i]=0 then {*Якщо поточна вершина не рофарбована*}
  begin
   for c:=color[i]+1 to Cmax do
   if Nicecolor then
   begin
    color[i]:=c;
     {*Якщо неконфліктний, то розфарбовуємо вершину і фарбою с*}
    visit(i+1);
      {Рекурсивно викликаємо процедуру для розфарбування наступної вершини}
   end;
  end;
 end;
end;


Begin
 i:=1;
 visit(i);
End;
```

11. ПРИКЛАД РОБОТИ РЕКУРСИВНОЇ ПРОЦЕДУРИ ПОСЛІДОВНОГО РОЗФАРБУВАННЯ

Розглянемо граф G та застосуємо рекурсивну процедуру для його розфарбування.



Матриця суміжності A має такий вигляд

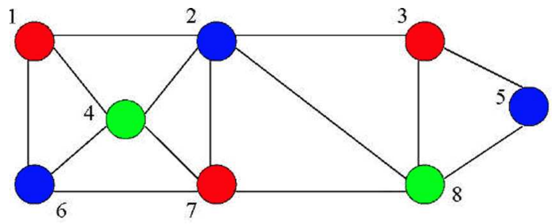
$$A=\begin{array}{cc} & \begin{array}{cccccccc}1&2&3&4&5&6&7&8\end{array}\\ \begin{array}{c}1\\2\\3\\4\\5\\6\\7\\8\end{array} & \begin{pmatrix}0&1&0&1&0&1&0&0\\1&0&1&1&0&0&1&1\\0&1&0&0&1&0&0&1\\1&1&0&0&0&1&1&0\\0&0&1&0&0&0&0&1\\1&0&0&1&0&0&1&0\\0&1&0&1&0&1&0&1\\0&1&1&0&1&0&1&0\end{pmatrix}\end{array}$$

Робота алгоритму зведена в таблицю

Перший стовбець містить виклики процедури Visit, а решта стовбців показує яка фарба була принята, а яка відхилена.

	Червоний	Синій	Зелений
Visit(1)	+		
Visit(2)	-	+	
Visit(3)	+		
Visit(4)	-	-	+
Visit(5)	-	+	
Visit(6)	-	+	
Visit(7)	+		
Visit(8)	-	-	+

Розфарбований граф має вигляд:



12. «ЖАДІБНИЙ » АЛГОРИТМ РОЗФАРБУВАННЯ

Нехай дано зв'язний граф $G(V,E)$.

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

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

Procedure Greedy

For (для кожної незафарбованої вершини $v \in V$) **do**

If v *не суміжна з вершинами з* $monochrom$ **then**

begin

$color(v) :=$ колір*;*

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

End;

Розглянемо детальніше програмну реалізацію даного алгоритму за умови, що для представлення графа використовують матрицю суміжності.

Const N=10; {*максимальна кількість вершин графа*}

Type V=0..N;

TS=**Set** of V;

TColArr = **Array** (1..N) of V;

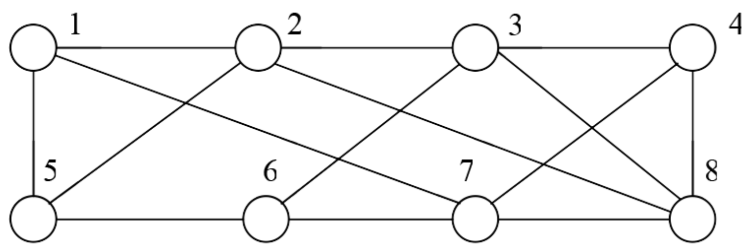
TA = **Array** (1..N, 1..N) of Integer;

```
Var ColArr: TColArr; {*масив номерів фарб для кожної вершини графа*}
        A:TA; {*матриця суміжності графа*}
        Color:Byte;
    AllColored:Boolean;
    k:Byte;
Procedure Avid(i:Integer);
{*функція вибору фарби для розфарбування вершини з номером і *}
Var W:TS;
   j:Byte;
function Check(i):Boolean; {*Перевірка кольору суміжних вершин*}
var Ch:Boolean;
begin
 Ch:=true;
 For j=1 to n do
 If (A[j,i]=1)then {*Якщо вершина j суміжна з тою, що підлягає перевірці *}
 If (j in W)then Ch:=false;
 {*Якщо вершина j розфарбована поточним кольором *}
 Check:= Ch;
end;
Begin
 Inc(Color); {*Встановлюємо новий колір*}
 W:=[]; {*Очищаємо множину одноколірних вершин*}
 ColArr[i]:=Color; {*Розфарбовуємо першу вершину новою фарбою*}
 W:=W+[i]; {*Доповнюємо множину одноколірних вершин вершиною і*}
{*Перевіряємо інші вершини на можливість розфарбування цією фарбою *}
 For k:=1 to n do if ColArr[k]=0 then
 If Check(k)then begin ColArr[k]:=Color; W:=W+[k];end;
End;
```

```
Begin {*Головний цикл*}
 <Вводимо матрицю суміжності графа>
 {*цикл по вершинах графа*}
  Color:=0;
  AllColored:=false; {*Ознака того, що всі вершини розфарбовано*}
  While not AllColored do
  Begin
   AllColored:=true;
   For i=1 to N do If ColArr[i]=0 then
   begin
    {*Знайшли не розфарбовану вершину*}
    AllColored:=false;
    Avid(i); {*процедура жадібного розфарбування*}
   end;
  End;
  <Виводимо результат розфарбування>
End;
```

13. ПРИКЛАД РОБОТИ «ЖАДІБНОГО» АЛГОРИТМУ РОЗФАРБУВАННЯ

Розглянемо граф G та застосуєм до нього «жадібний» алгоритм розфарбування.



Матриця суміжності A має такий вигляд

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

Для початку розфарбування вибираємо вершину з номером 1 та розфарбовуємо її в колір 1 (червоний).

Далі відбувається пошук несуміжної вершини з вершиною 1. Якщо така вершина знайдена, то вона також розфарбовується в колір 1(червоний).

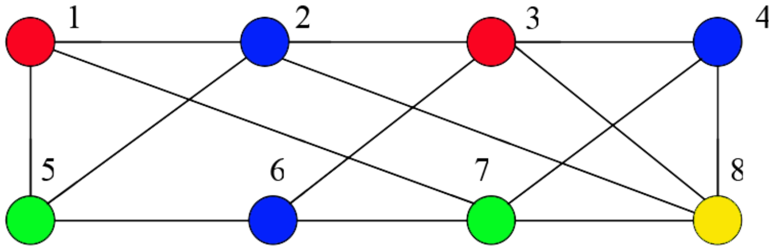
Наступна знайдена для розфорбування кольором 1 вершина повинна бути не суміжною з двома попередніми. Процес продовжується до того часу, поки всі можливості розфарбувати вершини кольором 1 будуть вичерпані.

Після цього, вибираємо фарбу кольору 2 (синя) і розфарбовуємо нею вершину з мінімальним номером, яка є не розфарбованою до цього часу. Наступна підходяща для розфарбування фарбою 2 вершина повинна бути не суміжною з вершиною, яка була розфарбована кольором 2 (синій) на попередньому кроці. Процес розфарбування фарбою 2 також продовжується до того часу, поки не будуть вичерпані всі можливості розфарбування вершин цією фарбою.

Перед вибором чергової фарби для розфарбування завжди перевіряємо, чи залишилися ще не розфарбовані вершини. Якщо такі вершини знайдено, то вибираємо чергову фарбу і продовжуємо процес розфарбування.

Якщо ж всі вершини графа розфарбовано, то процес розфарбування жадібним алгоритмом закінчується.

Результат розфарбування «жадібним» алгоритмом графа G показано на рисунку


1
2
3
4
5
6
7
8