

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

КАТЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Дискретна математика

Лабораторна робота №4

«Розфарбовування графа, алгоритми розфарбування»

Виконав:
студент групи ІО-32
Крадожон М. Р.
Номер у списку групи: 16
Перевірив:
Пономаренко А. М.

Лабораторна робота №4

Тема: «Розфарбовування графа, алгоритми розфарбування».

Мета: вивчення способів правильного розфарбовування графа. основних алгоритмів на графах.

Загальне завдання: створити програму для правильного розфарбовування графа на основі одного з алгоритмів розфарбування.

Теоретичні основи:

Граф G задають множиною точок або вершин x_1, x_2, \dots, x_n (яку позначають через X) і множиною ліній або ребер a_1, a_2, \dots , (яку позначають символом A), що з'єднують між собою всі або частину цих точок. Таким чином, граф G повністю задають (і позначають) парою (X, A) .

Петлею називають дугу, початкова та кінцева вершини якої збігаються. Одним з найбільш важливих понять теорії графів є дерево. Неорієнтованим деревом називають зв'язний граф, що не має циклів.

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, називають хроматичним числом графа G і позначають $\chi(G)$. Завдання знаходження хроматичного числа графа називають 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.

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, називають планарним.

Алгоритм прямого неявного перебору

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

Крок 1. 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.

Крок 2. 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.

Крок 3. 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 кольору.

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 розташовують в порядку зменшення їх степенів. Першу вершину зафарбовують в колір 1, потім список вершин переглядають за зменшенням степенів, і в колір 1 зафарбовують кожну вершину, яка не є суміжною з вершинами, зафарбованими в той же колір. Потім повертаються до першої в списку незафарбованої вершині, фарбують її в колір 2 і знову переглядають 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 діють із кольорами 3, 4 і т. д., допоки не будуть пофарбовані всі вершини. Кількість використаних кольорів буде тоді наближеним значенням хроматичного числа графа.

Модифікований евристичний алгоритм розфарбування

Визначення 1. Відносий степінь – це степінь нерозфарбованих вершин у нерозфарбованому підграфі даного графа.

Визначення 2. Двокроковий відносний степінь – 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 степінь визначимо як суму відносних степенів суміжних вершин.

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.

Фарбування у фарбу a вершини v утворює навколо неї в $R(v)$ «мертву зону» для фарби a . Очевидно, при мінімальному розфарбуванні кожна фарба повинна розфарбувати максимально 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 $R(v)$ від іншої. Таким чином, суть алгоритму полягає в тому, щоб на черговому кроці вибрати для розфарбування фарбою a вершину R . Цей процес повторювати доти, поки фарбою a не будуть пофарбовані всі можливі вершини графа. Графічно фарбування вершин v_1 і v_2 однією фарбою можна відобразити як «склеювання» цих вершин.

Рекурсивна процедура послідовного розфарбування

1. Фіксуємо порядок обходу вершин.
2. Ідемо по вершинах, використовуючи такий найменший колір, який не викликає конфліктів.
3. Якщо вже використаний колір вибрати неможливо, то тільки тоді вводимо новий колір. У процедурі використовується рекурсивний виклик процедури фарбування наступної вершини у випадку успішного фарбування попередньої вершини.

«Жадібний» алгоритм розфарбування

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

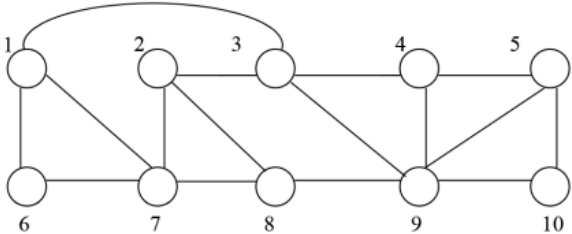
Далі відбувається пошук несуміжної вершини з вершиною 1. Якщо така вершина знайдена, то вона також розфарбовується в колір 1 (червоний).

Наступна знайдена для розфарбування 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 вершина, яка підходить для розфарбування фарбою 2, 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 цієї 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.

Індивідуальне завдання: Варіант 1:

№	Опис варіанта
1	<p>А) Виконати завдання 1 до лабораторної роботи. Б) Програма повинна дозволяти розфарбування довільного графа. В) Перевірити роботу програми на даному графі G.</p>  <p>Вивести у графічному режимі розфарбований граф або включити у протокол розфарбований вручну граф за результатами роботи програми.</p>

Роздруківка коду:

```
import math
import networkx as nx
import matplotlib.pyplot as plt
```

```
from tkinter import *
from matplotlib.backends.backend_tkagg import *
```

```
from networkx import Graph
from sencitive_data import senc_data
```

```
class GraphApp:
```

```
    def __init__(self):
        self.array, self.empty, self.copy_array, self.edges, self.list_of_colors, self.color_of_edge, self.root,
self.colored_graph, self.colored_graph_start, self.edges, self.sum, self.size, self.seed = [], [], [], [], ['pink',
'darkorange', 'darkviolet', 'chartreuse', 'cyan', 'crimson', 'brown', 'blueviolet', 'pink'], [], Tk(), [], [], {}, 0, 0, 0
        self.root.title("Window 1")
        self.root.geometry("1400x700")
        self.root.configure(bg='white')
        self.root.resizable(False, False)
        self.create_button("Побудувати з поля", 20, 2, self.read_fild, 20, 370)
        self.create_button("Побудувати з файлу", 20, 2, self.read_file, 238, 370)
        self.create_label(f"{senc_data['name']} \n ІО-{senc_data['group']} \nНомер у списку групи:
{senc_data['number_in_list']} Варіант: 1", 10, 450)
        self.create_label("Суміжність", 850, 0)
        self.create_label("Задайте граф матрицею суміжності", 60, 0)
        self.labelsym, self.labelinc, self.labeledict = self.create_label("", 750, 30), self.create_label("", 720, 30),
self.create_label("", 1220, 30)
        self.text = Text(width=45, height=15, bg="white", fg='black', wrap=WORD, font=("Garamond", 15))
        self.text.place(x=20, y=30)

    def run(self): self.root.mainloop()

    def create_button(self, text, width, height, command, x, y):
        button = Button(self.root, text=text, width=width, height=height, command=command, font=("Garamond",
12))
        button.place(x=x, y=y)
        return button

    def create_label(self, text, x, y):
        label = Label(self.root, text=text, justify=LEFT, font=("Garamond", 15), background="white")
        label.place(x=x, y=y)
        return label

    def read_fild(self):
        self.seed += 1
        self.clear()
        s: str = self.text.get('1.0', END)
        self.convert_text(s)
        self.start_work()

    def read_file(self):
        try:
            with open("file1.txt", "r", encoding="cp1251") as f: s = f.read()
            self.seed += 1
            self.clear()
            self.convert_text(s)
            self.start_work()
        except FileNotFoundError: print("File not found.")
```

```

def convert_text(self, s: str):
    s = s.replace(" ", "").replace("\n", "")
    if s:
        split = s.split(",")
        length = int(math.sqrt(len(split)))
        self.array = [[int(split[length * i + j]) for j in range(length)] for i in range(length)]

def start_work(self):
    self.make_array()
    self.print_matrix()
    self.set_colors()
    self.make_canvas(self.build_adjacent(), 450, 270, self.colored_graph_start)
    self.make_canvas(self.build_adjacent(), 900, 270, self.colored_graph)
    self.create_label("Початковий граф", 550, 290)
    self.create_label("Розфарбований граф", 1000, 290)

def make_array(self):
    self.size, self.copy_array, self.sum = len(self.array), [list(row) for row in self.array], sum(self.array[i][j] for i in
range(self.size) for j in range(i, self.size) if self.array[i][j] > 0)
    self.directed(self.sum, self.size)

def directed(self, weight: int, height: int):
    empty, a = [[0 for _ in range(weight)] for _ in range(height)], 0
    for i in range(height):
        for j in range(height):
            while self.copy_array[i][j] > 0 and a < weight:
                if i != j:
                    empty[i][a] += 1
                    empty[j][a] -= 1
                else: empty[i][a] += 2
                a += 1
            self.copy_array[i][j] -= 1

def build_adjacent(self) -> Graph: return nx.DiGraph([(i + 1, j + 1) for i in range(self.size) for j in range(i, self.size)
if self.array[i][j] == 1])

def set_colors(self):
    self.color_of_edge, self.colored_graph, self.colored_graph_start = [0 for i in range(len(self.array))], [0 for i in
range(len(self.array))], ['springgreen' for i in range(len(self.array))]
    for i in range(len(self.array)):
        self.color_of_edge[i] = self.color(i)
        self.colored_graph[i] = self.list_of_colors[self.color_of_edge[i]]

def color(self, i):
    w, curcol = {0}, 0
    w.update(self.color_of_edge[j] for j in range(i) if self.array[j][i] > 0)
    while True:
        curcol += 1
        if curcol not in w: break
    return curcol

```

```
def make_canvas(self, G: Graph, x: int, y: int, c: list):
    size = int(math.sqrt(len(G.nodes)) ** 1.15)
    f = plt.Figure(figsize=(size, size), dpi=400 / size)
    a = f.add_subplot(111)
    nx.draw(G, ax=a, with_labels=True, node_color=c, arrowstyle='-', pos=nx.spring_layout(G, seed=self.seed))
    canvas = FigureCanvasTkAgg(f, master=self.root)
    canvas.get_tk_widget().place(x=x, y=y)

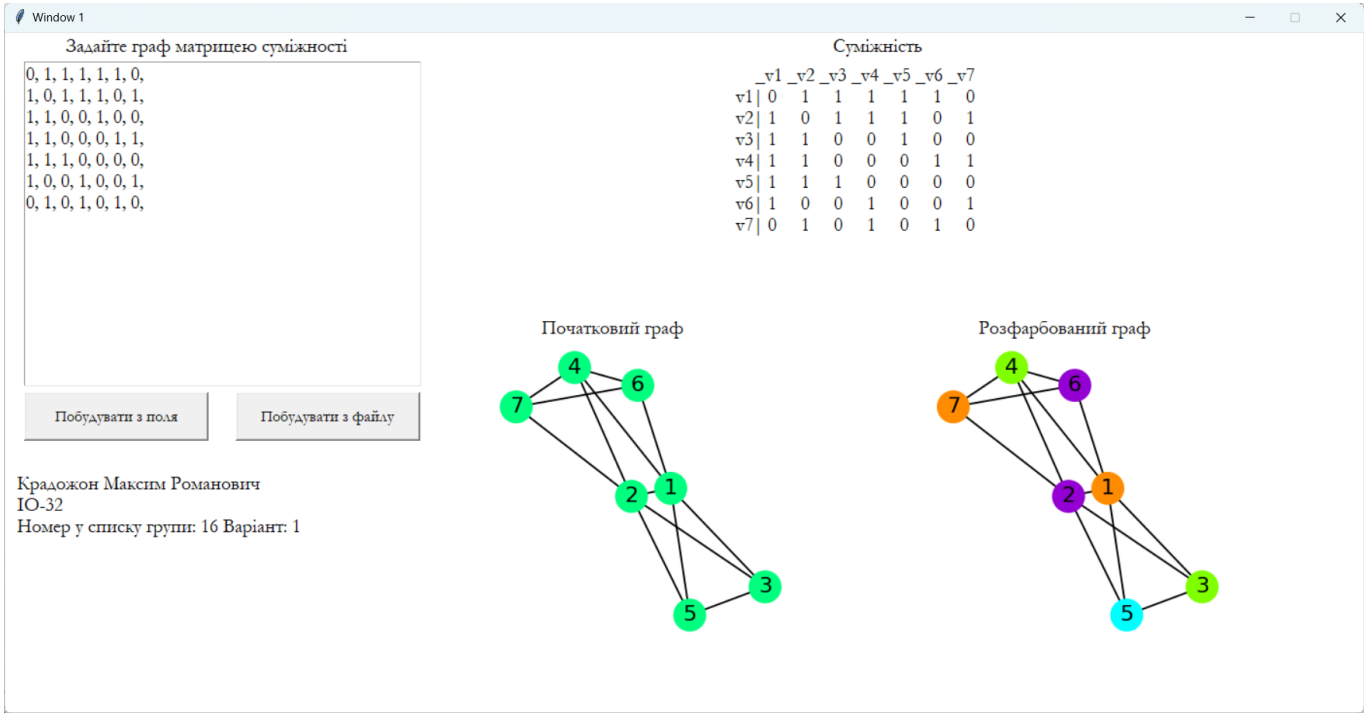
def print_matrix(self):
    s = "  " + " ".join(f"_v{i+1}" for i in range(self.size)) + "\n"
    s += "\n".join(f"v{i+1}| {' '.join(str(self.array[i][j]) for j in range(self.size))}" for i in range(self.size))
    self.labelsym.configure(text=s)

def clear(self):
    for attr in ['array', 'empty', 'copy_array', 'edges']:
        getattr(self, attr).clear()
    self.sum = self.size = 0

app = GraphApp()
app.run()
```

Скриншоти:

Для довільного графа:



Для графа, який дано за умовою:

Window 1

Задайте граф матрицею суміжності

Суміжність

	v1	v2	v3	v4	v5	v6	v7	v8	v9	v10
v1	0	0	1	0	0	1	1	0	0	0
v2	0	0	1	0	0	0	1	1	0	0
v3	1	1	0	1	0	0	0	0	1	0
v4	0	0	1	0	1	0	0	0	1	0
v5	0	0	0	1	0	0	0	0	1	1
v6	1	0	0	0	0	0	1	0	0	0
v7	1	1	0	0	0	1	0	1	0	0
v8	0	1	0	0	0	0	1	0	1	0
v9	0	0	1	1	1	0	0	0	0	1
v10	0	0	0	0	1	0	0	0	1	0

Побудувати з поля

Побудувати з файлу

Крадожон Максим Романович
ІО-32
Номер у списку групи: 16 Варіант: 1

Початковий граф

Розфарбований граф

file1.txt M

file1.txt

```
...
1 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
2 0, 0, 1, 0, 0, 0, 1, 1, 0, 0,
3 1, 1, 0, 1, 0, 0, 0, 0, 1, 0,
4 0, 0, 1, 0, 1, 0, 0, 0, 1, 0,
5 0, 0, 0, 1, 0, 0, 0, 0, 1, 1,
6 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
7 1, 1, 0, 0, 0, 1, 0, 1, 0, 0,
8 0, 1, 0, 0, 0, 0, 1, 0, 1, 0,
9 0, 0, 1, 1, 1, 0, 0, 0, 0, 1,
10 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
```

Висновок: Виконавши цю лабораторну роботу, я зміг здобути відповідні навички в розфарбовуванні графа, алгоритмах розфарбування. Під час виконання лабораторної роботи проблем не виникало, а складність була в структуруванні коду та приведенні його до більш гарного вигляду.