Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки
Дискретна математика
Лабораторна робота №4
«Розфарбовування графа, алгоритми розфарбування»

Виконав:
студент групи ІВ-71
Мазан Ян Владиславович
Залікова книжка
№7109
Перевірив:
Саверченко Василь Григорович

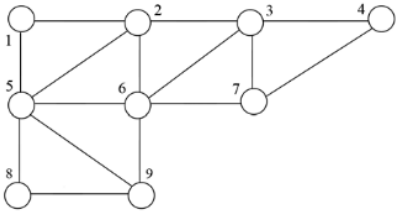
Київ-2018

Варіант виразу відповідно до індивідуального завдання:

I = 7109mod6 + 1 = 6

6

А) Виконати завдання 6 до лабораторної роботи.
Б) Програма повинна дозволяти розфарбування довільного графа.
В) Перевірити роботу програми на даному графі G.



Вивести у графічному режимі розфарбований граф, або включити у протокол розфарбований вручну граф за результатами роботи програми.

Теоретичні відомості:

Жадібний алгоритм розфарбовування:

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).

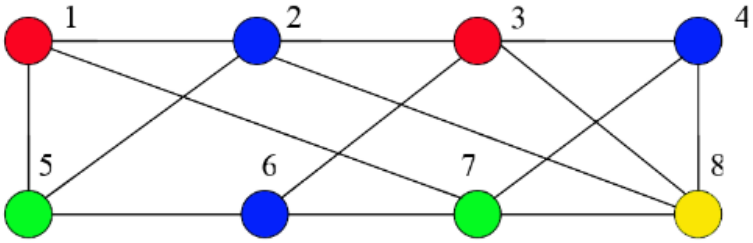
Наступна знайдена для розфарбування кольором 1 вершина повинна бути не суміжною з двома попередніми. Процес продовжується до того часу, поки всі можливості розфарбувати вершини кольором 1 будуть вичерпані.

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 цією фарбою.

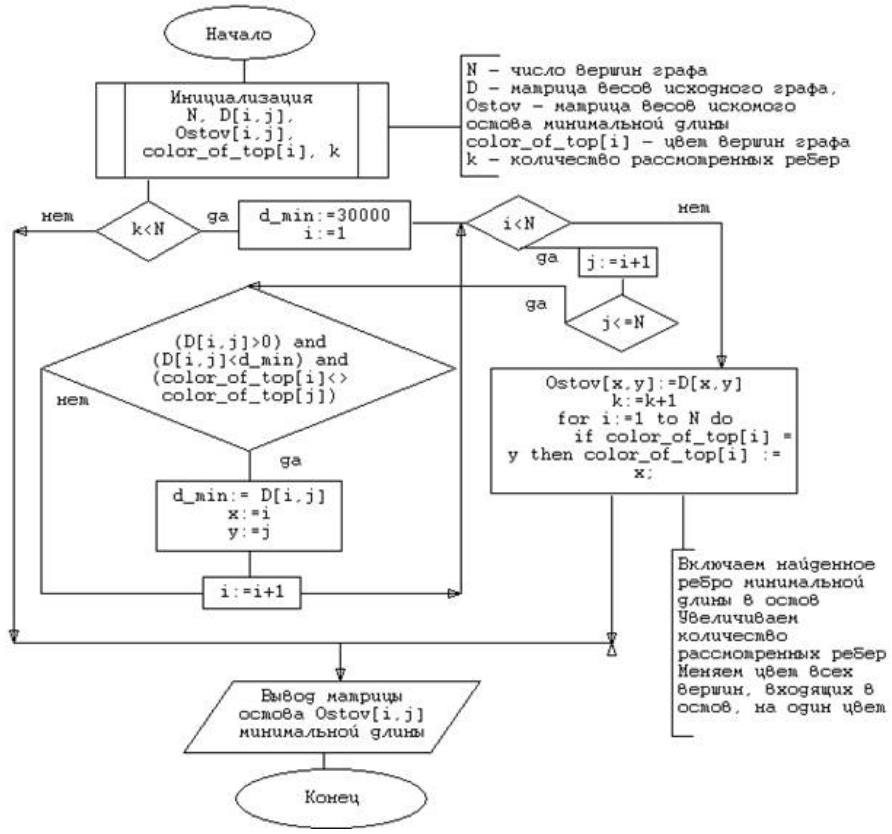
Перед вибором чергової фарби для розфарбування завжди перевіряємо, чи залишилися ще не розфарбовані вершини. Якщо такі вершини знайдено, то вибираємо чергову фарбу і продовжуємо процес розфарбування.

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

Результат розфарбування «жадібним» алгоритмом графа G показано на рисунку



Блок-схеми, які відповідають алгоритмам, що використані в лабораторній роботі



Роздруківка того фрагменту тексту програми, який написаний індивідуально

Файл GUI.py:

```
from tkinter import *
from tkinter import messagebox
import Graph_gen
import matplotlib.pyplot as plt
import networkx as nx
class window_1(Frame):
    def __init__(self,master = None):
        super().__init__(master)
        self.pack()
        self.def_widgets()
        self.packer()
    def def_widgets(self):
        self.caption = Label(self, text = "Введіть кількість вершин графа")
        self.entry = Entry(self)
        self.gen_button = Button(self, width = 30, height = 2, wraplength = 200,
                                 text = "Перейти до наступного етапу генерування графа",
command = self.generate_graph_window)
    def packer(self):
```

```
        self.caption.grid(row = 0, column = 0)
        self.entry.grid(row = 1, column = 0)
        self.gen_button.grid(row = 2, column = 0)
    def table(self,nodes_num):
        self.entry.destroy()
        self.gen_button.destroy()
        self.entry_labels_first_row = [Label(self, text = "n" + str(i)) for i in
range(nodes_num)]
        self.entry_labels_first_column = [Label(self, text = "n" + str(i)) for i in
range(nodes_num)]
        self.table_entries = [[Entry(self, width = 5) for i in range(nodes_num)] for j in
range(nodes_num)]
        for i in range(len(self.entry_labels_first_row)):
            self.entry_labels_first_row[i].grid(row = 0, column = i+1)
            self.entry_labels_first_column[i].grid(row = i+1, column = 0)
        for i in range(len(self.entry_labels_first_row)):
            for j in range(len(self.entry_labels_first_row)):
                self.table_entries[i][j].grid(row = i+1, column = j+1)
                self.table_entries[i][j].insert(END, "0")

        self.caption.config(text = "Матриця суміжності графа", justify = CENTER)
        self.graph_show = Button(self, width = 20, text = "Згенерувати граф",
command = self.draw)
        self.task_calculate = Button(self, width = 20, wraplength = 180,text =
"Розфарбувати граф", command = self.color_graph)
        self.caption.grid(row = nodes_num+2, column = 1, columnspan = nodes_num)
        self.graph_show.grid(row = nodes_num+3, column = 1, columnspan =
nodes_num)
        self.task_calculate.grid(row = nodes_num+4, column = 1, columnspan =
nodes_num)
    def generate_graph_window(self):
        if self.entry.get().isdigit():
            self.nodes_number = int(self.entry.get())
            self.table(self.nodes_number)
        else:
            messagebox.showinfo("Помилка", "Ви ввели ненатуральну кількість
вершин для графа")
    def graph_gen(self):
        self.Graph_Renji = Graph_gen.graph_generation(self.nodes_number)
        self.Graph_Renji.make_connections(self)
    def draw(self):
        self.graph_gen()
        self.Graph_Renji.draw()
    def color_graph(self):
        self.graph_gen()
```

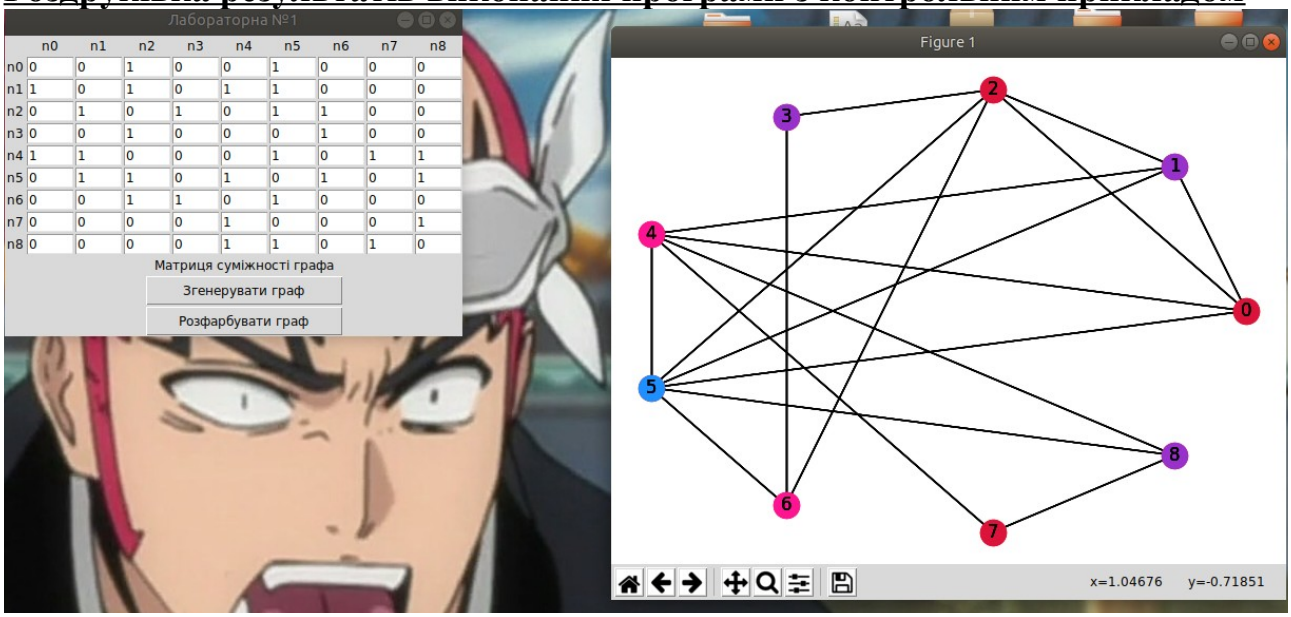
```
        self.Graph_Renji.algorythm()
```

Файл Graph_gen.py:

```
import networkx as nx
from tkinter import messagebox
import pylab as plt
class graph_generation():
    def __init__(self, nodes_num):
        self.nodes_num = nodes_num
    def make_connections(self,window):
        try:
            self.data_array = [[int(window.table_entries[i][j].get()) for i in
range(self.nodes_num)] for j in range(self.nodes_num)]
        except:
            messagebox.showinfo("Помилка", "Ви ввели нецілі числа в матрицю")
            return
        self.Graph = nx.Graph()
        for i in range(self.nodes_num):
            self.Graph.add_node(i)
        for i in range(self.nodes_num):
            for j in range(self.nodes_num):
                if self.data_array[i][j]==1:
                    self.Graph.add_edge(i, j)
    def draw(self):
        nx.draw(self.Graph, node_color = "#DC143C",with_labels=True)
        plt.show()
    def algorythm(self):
        def graphColoring():
            global colored
            colored=[[]]
            color=0
            stack=[]
            num2=num1=-1
            def loop(num):
                for t in colored[color]:
                    if self.data_array[t][num]==1:
                        return False
                return True
            for a in self.data_array:
                num1+=1
                if num1 not in stack:
                    if not loop(num1):
                        colored.append([])
                        color+=1
```

```
                    stack.append(num1)
                    colored[color].append(num1)
                num2=num1
                for b in a[num1:]:
                    if b==0 and num2 not in stack and loop(num2):
                        stack.append(num2)
                        colored[color].append(num2)
                    num2+=1
        return colored
    def make_graf2(nodes):
        list_color = ["#DC143C", "#9932CC", "#FF1493", "#1E90FF", "#4B0082",
"#B0C4DE", "#32CD32", "#191970"]
        nx.draw(self.Graph, pos = nx.shell_layout(self.Graph))
        for i in range(len(nodes)):
            nx.draw(self.Graph, pos = nx.shell_layout(self.Graph),node_color =
list_color[i],nodelist = nodes[i],with_labels=True)
        plt.show()
    make_graf2(graphColoring())
```

Роздруківка результатів виконання програми з контрольним прикладом



Аналіз результатів та висновки

Під час виконання даної лабораторної роботи я навчився навичкам роботи з Tkinter та Networkx у Python, вивчив способи розфарбовування графів. Під час виконання роботи виникали деякі проблеми із реалізацією алгоритму на Python.