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-91

Степанюк Роман Вікторович

Залікова книжка № ІВ-9127

Перевірив Новотарський М.А.

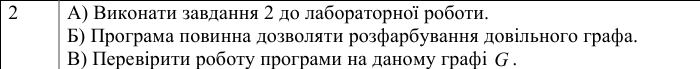
Київ – 2020

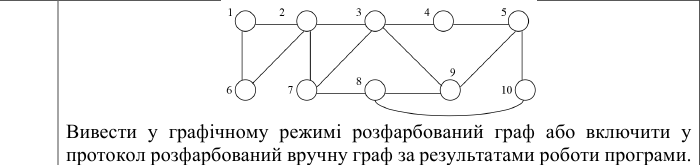
**Мета роботи:** вивчення способів правильного розфарбовування графа.

**Завдання:** створити програму для правильного розфарбовування графа на основі одного з алгоритмів розфарбування.

**Варіант = 9127 mod 6 +1 = 2**

**Завдання за варіантом:**





**Реалізація евристичного алгоритму:**

def color\_graph():

    global colors

    n = int(node\_entry.get())

    def getkey(item):

        return item[1]

    node\_degree = []

    for node in G.nodes():

        node\_degree.append([node, G.degree(node)])

        node\_degree.sort(key = getkey, reverse = True)

    def coloring():

        def colorcheck(node, color):

            p = True

            for i in list(G[node]):

                if ((matrix[i][int(node)] == 1) and (node\_color.get(i, NONE) ==  colors[color])):

                    p = False

            return p

        curcolor = 0

        painted = []

        node\_color = {}

        for node, degree in node\_degree:

            neighbours\_node = list(G[node])

            if node not in painted:

                node\_color.update({node: colors[curcolor]})

                painted.append(node)

            for i in G.nodes():

                if (i not in neighbours\_node):

                    if (i not in painted):

                        if (colorcheck(i, curcolor) == True):

                            node\_color.update({i: colors[curcolor]})

                            painted.append(i)

                else: pass

            curcolor += 1

        return node\_color

    coloring = coloring()

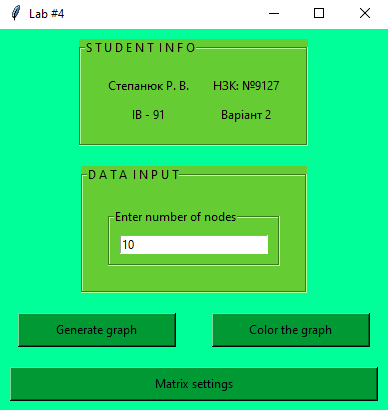
        node\_colormap = []

    for i in range(n):

        node\_colormap.append(coloring[i])

**Контрольний приклад:**

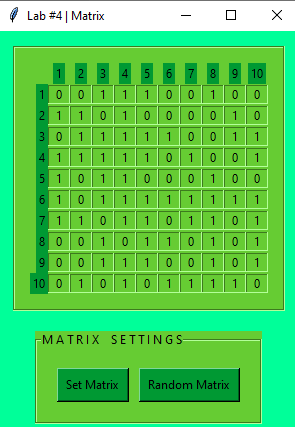
1) Вводимо кількість вершин(для прикладу взято N = 10)



2) Перейдемо до налаштувань матриці суміжності за допомогою кнопки «Matrix settings» та сформуємо випадкову матрицю. Після цього підтвердимо її кнопкою «Set matrix».

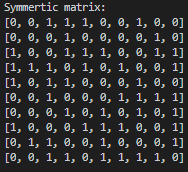
Також, можна вручну задавати ребра натискаючи на відповідні віджети «entry» в якому зберігаються значення елементів матриці

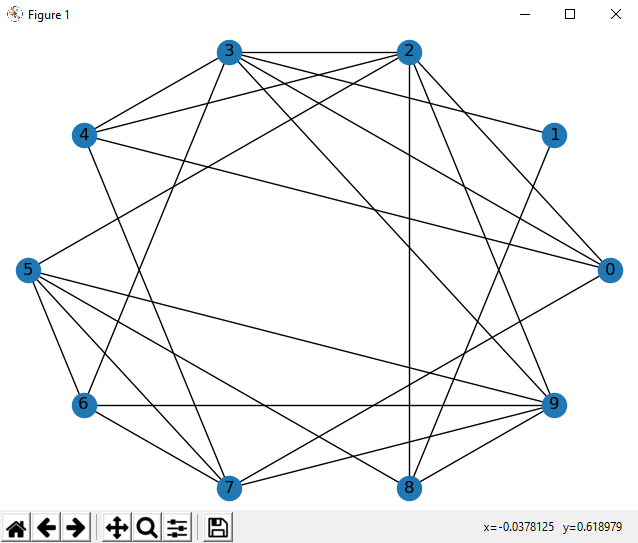
!ВАЖЛИВО: Треба завжди натискати кнопку «Set matrix», інакше значення не збережеться



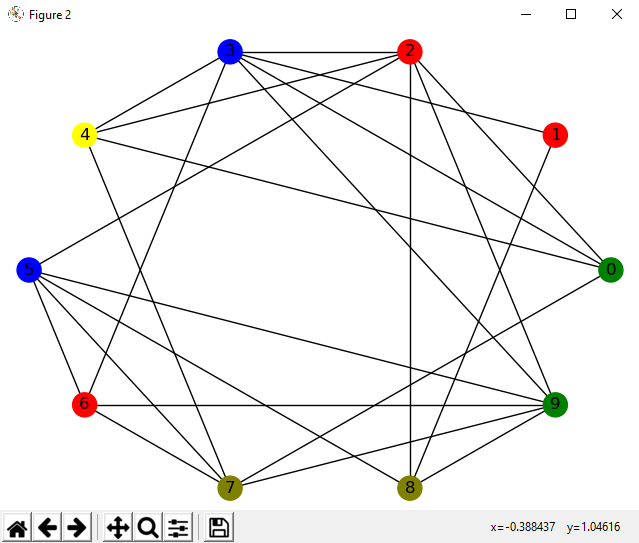
3) Повернемось до головного вікна та натиснемо кнопку «Generate graph»

Після цього випадково згенерована верхньотрикутна матриця буде симетрично відображена, а також, всі елементи головної діагоналі отримають значення «0». В результаті буде отримана симетрична матриця з якої і буде будуватись ациклічниий граф





4) Для того, щоб розфарбувати отриманий граф у головному вікні необхідно натиснути кнопку «Color the graph»



**Висновок:**

Під час виконання цієї лабораторної роботи виникла лише одна проблема.

Алгоритм не перевіряв умову того, чи розфарбовані суміжні вершини і тому розфарбування виконувалось неправильно. Проблема була вирішена успішно за допомогою функції яка проходила по всіх суміжних вершинах і перевіряла їхній колір.