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 ІО-64

Кобилюк А. Г.

Залікова книжка № 6411

Перевірив Новотарський М. А.

Київ

2017 р.

**Тема:** Розфарбовування графа, алгоритми розфарбування.

**Мета:** вивчення способів правильного розфарбовування графа

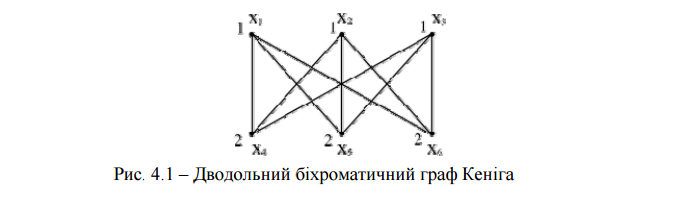
**Загальне завдання:** створити програму для правильного розфарбовування графа на основі одного з алгоритмів розфарбування

**Теоретичні відомості:**

**ОСНОВНІ ОЗНАЧЕННЯ**

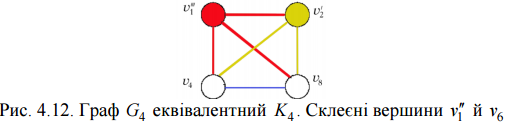
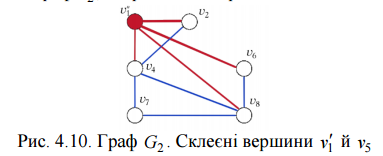
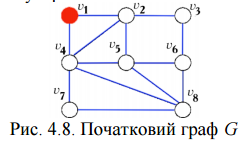
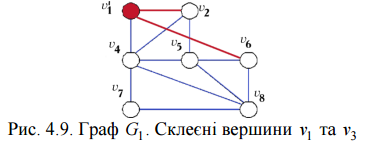
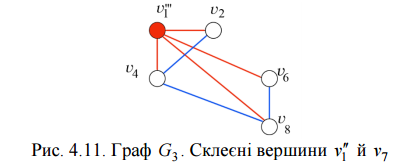
Різноманітні завдання, що виникають при плануванні виробництва, складанні графіків огляду, зберіганні та транспортуванні товарів та ін., часто можуть бути представлені як задачі теорії графів, тісно пов'язані з так званим «завданням розфарбовування». Графи, що розглядаються в даній лабораторній роботі, 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 γ (G).

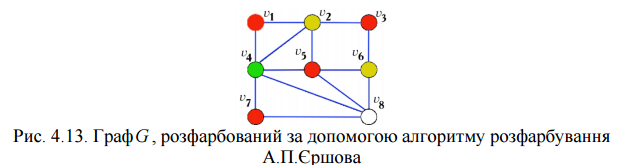
Завдання знаходження хроматичного числа графа називають задачею про розфарбовування (або завданням розфарбовування) графа. Відповідне цьому числу розфарбування вершин розбиває множину вершин графа на r підмножин, кожна з яких містить вершини одного кольору. Ці множини є незалежними, оскільки в межах однієї множини немає двох суміжних вершин. 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( x1 ),...,deg( xn ) (степені вершин графа) можна отримати тільки верхню і нижню оцінки для хроматичного числа графа.



Приклад розфарбовування графа наведений на рисунку 4.1. Цей граф є однією із заборонених фігур, що використовуються для визначення планарності. Цифрами «1» і «2» позначені кольори вершин. Максимальна кількість незалежних вершин графа α (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, де n - кількість вершин графа G , позначає найбільше ціле число, яке не більше за x . Ще одна нижня оцінка для γ (G) може бути отримана наступним чином: Верхня оцінка хроматичного числа може бути обчислена за формулою.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. 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 .

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. 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 Kn завжди розфарбовується в n кольорів, тобто кількість кольорів дорівнює кількості його вершин.

**ПРИКЛАД РОЗФАРБУВАННЯ ГРАФА МЕТОДОМ А. П. ЄРШОВА**



**Варіант для виконання лабораторної роботи:**

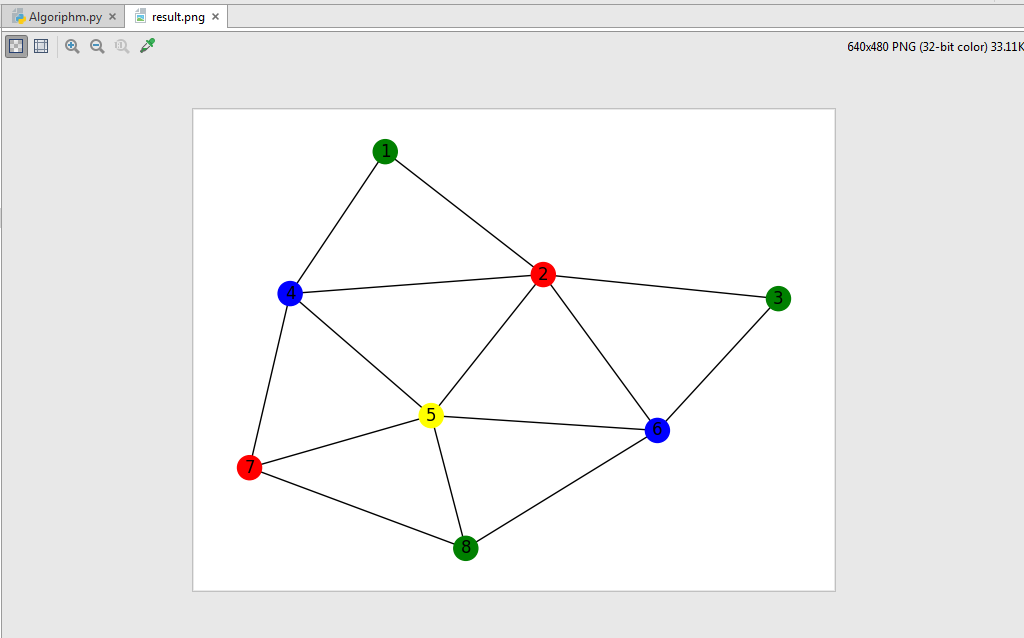
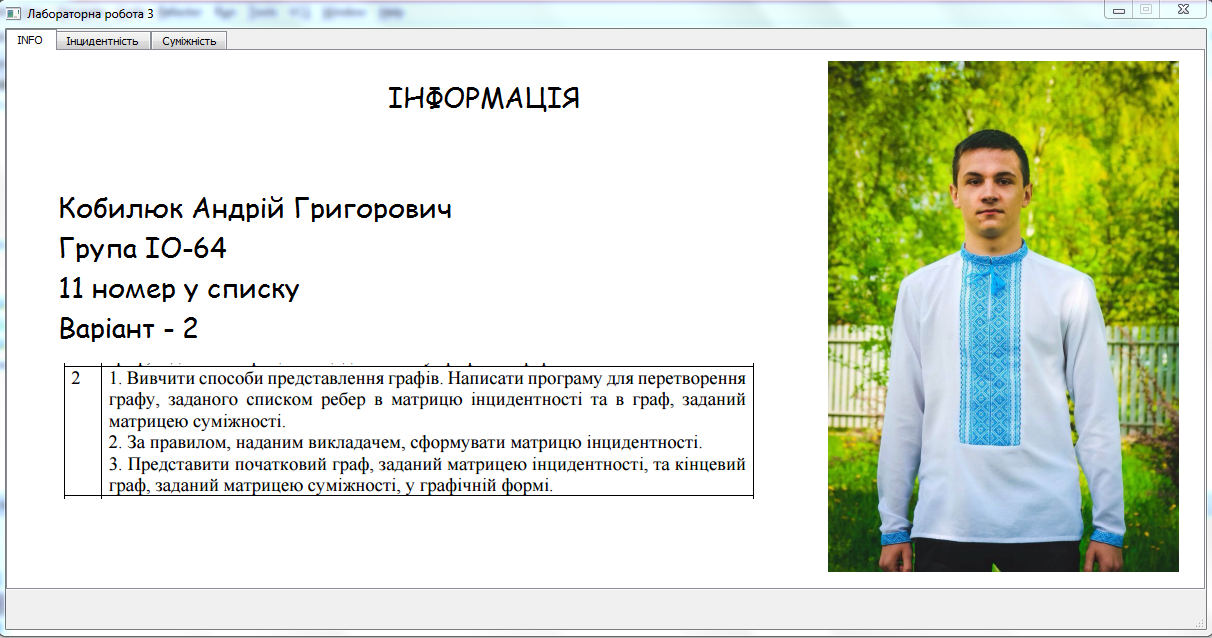
Номер варіанта I визначають як результат операції I = NZK mod 6+1, де NZK – номер залікової книжки. Номер варіанта відповідає номеру пункту завдання до лабораторної роботи.

**6411 mod 6 + 1 = 4**

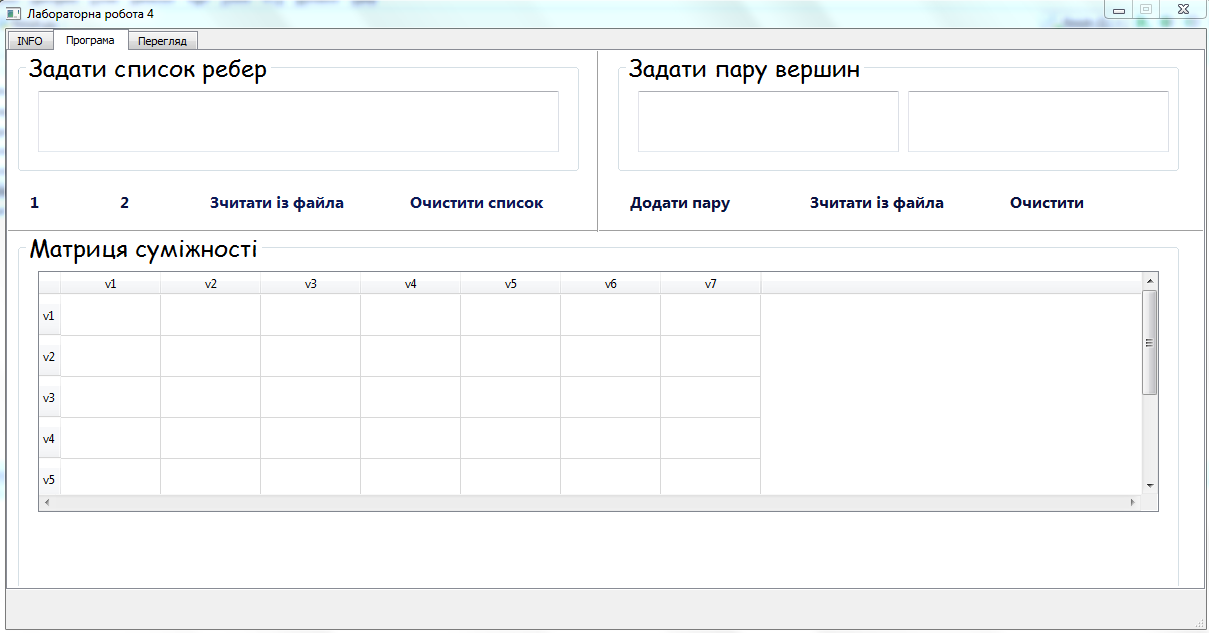
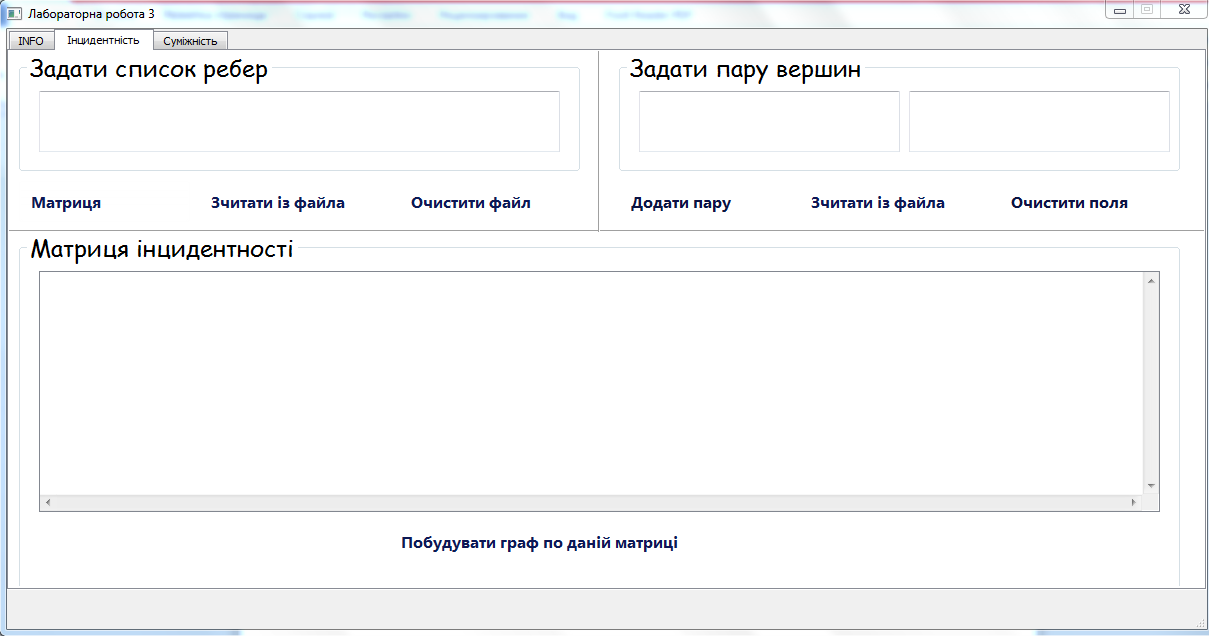
|  |  |
| --- | --- |
| Варіант | Опис варіанта |
| 4 | А) Виконати завдання 4 до лабораторної роботи.  Б) Програма повинна дозволяти розфарбування довільного графа.  В) Перевірити роботу програми на даному графі G .  Вивести у графічному режимі розфарбований граф або включити у протокол розфарбований вручну граф за результатами роботи програми. |

**Знімки екрану моєї роботи:**

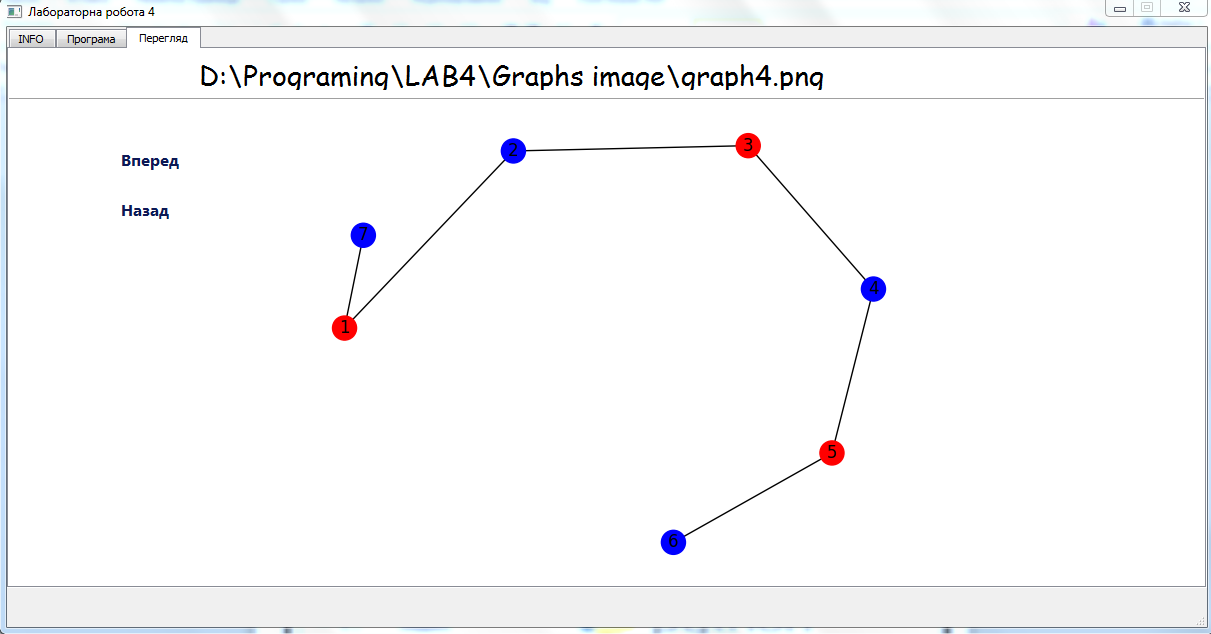
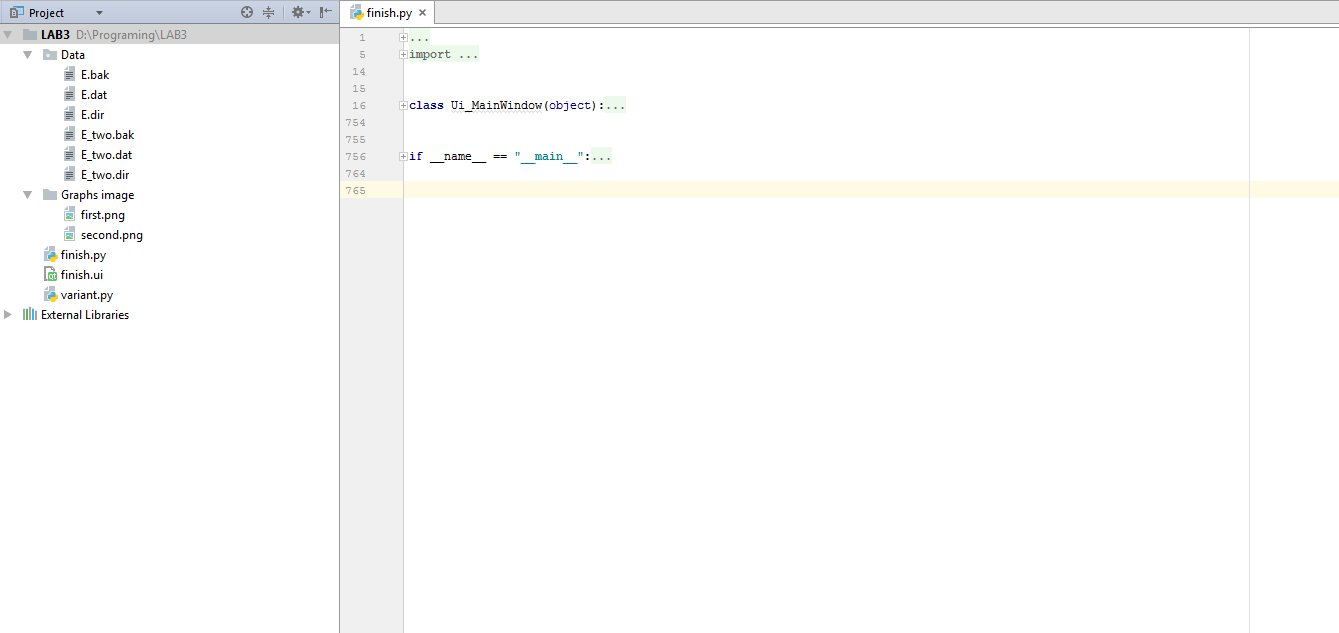
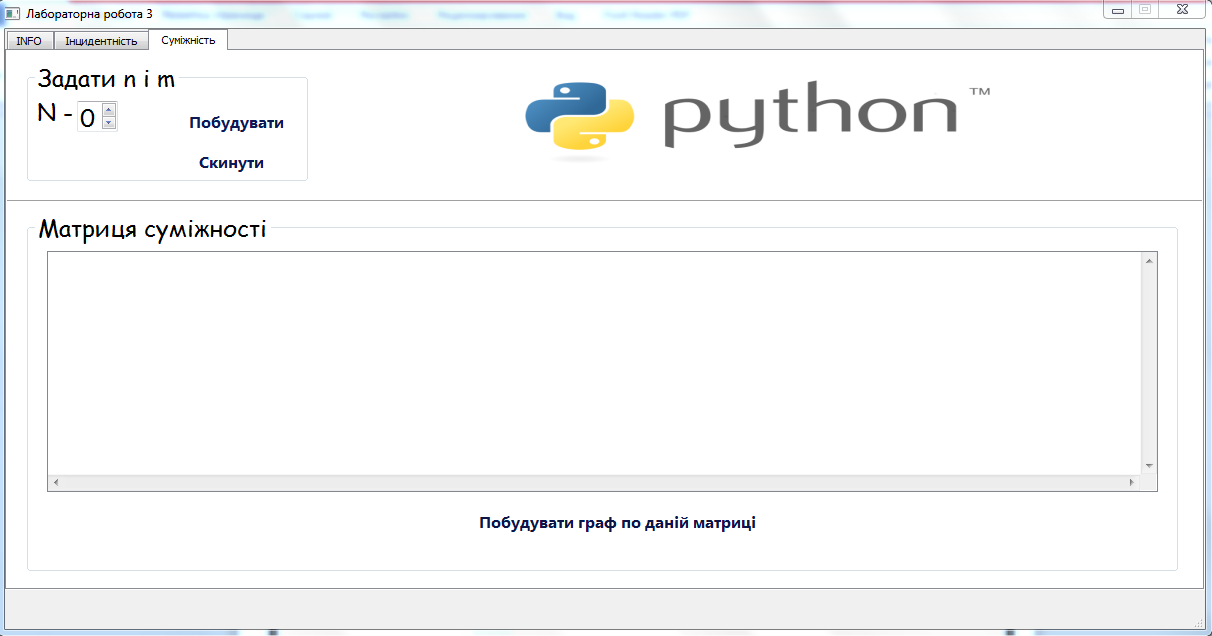
Розфарбований граф

Друге вікно

Третє вікно

**Висновок:**

Отже, виконуючи дану лабораторну роботу я закріпив знання про основні поняття про графи , повторів теорію щодо їх створення. В лабораторній роботі я розробив алгоритм для задання й відображення графів у графічній формі, для кращого сприйняття створив простий інтерфейс. Опрацював деякі лекції із кyрсy програмування за минyлий семестр (графи та робота з файлами).

КОД ПРОГРАМИ

*# -\*- coding: utf-8 -\*-  
  
#Все, що потрібно для PyQT5***from** PyQt5 **import** QtCore, QtGui, QtWidgets  
**from** PyQt5.QtWidgets **import** QMessageBox  
  
**from** networkx.drawing.nx\_pydot **import** write\_dot  
**import** matplotlib.pyplot **as** plt  
**import** networkx **as** nx  
**from** networkx.drawing.layout **import** shell\_layout,\  
 circular\_layout,spectral\_layout,spring\_layout,random\_layout  
**import** variant,copy, shelve, random  
**import** os.path  
  
**class** Ui\_MainWindow(object):  
  
 **def** setupUi(self, MainWindow):  
 MainWindow.setObjectName(**"MainWindow"**)  
 MainWindow.resize(1200, 600)  
 MainWindow.setMinimumSize(QtCore.QSize(1200, 600))  
 MainWindow.setMaximumSize(QtCore.QSize(1200, 600))  
 icon = QtGui.QIcon()  
 icon.addPixmap(QtGui.QPixmap(**"C:/Programing/LAB2/Icons/bank128.ico"**), QtGui.QIcon.Normal, QtGui.QIcon.Off)  
 MainWindow.setWindowIcon(icon)  
 self.centralwidget = QtWidgets.QWidget(MainWindow)  
 self.centralwidget.setObjectName(**"centralwidget"**)  
 self.tabWidget = QtWidgets.QTabWidget(self.centralwidget)  
 self.tabWidget.setGeometry(QtCore.QRect(0, 0, 1201, 561))  
 self.tabWidget.setObjectName(**"tabWidget"**)  
 self.tab = QtWidgets.QWidget()  
 self.tab.setObjectName(**"tab"**)  
 self.label = QtWidgets.QLabel(self.tab)  
 self.label.setGeometry(QtCore.QRect(820, 10, 351, 511))  
 self.label.setText(**""**)  
 self.label.setPixmap(QtGui.QPixmap(**"C:/Users/Sanverik/Pictures/XR5O9i7XFvs (1).jpg"**))  
 self.label.setScaledContents(**True**)  
 self.label.setObjectName(**"label"**)  
 self.label\_2 = QtWidgets.QLabel(self.tab)  
 self.label\_2.setGeometry(QtCore.QRect(380, 30, 201, 31))  
 font = QtGui.QFont()  
 font.setFamily(**"Comic Sans MS"**)  
 font.setPointSize(21)  
 self.label\_2.setFont(font)  
 self.label\_2.setObjectName(**"label\_2"**)  
 self.label\_3 = QtWidgets.QLabel(self.tab)  
 self.label\_3.setGeometry(QtCore.QRect(50, 90, 411, 31))  
 font = QtGui.QFont()  
 font.setFamily(**"Comic Sans MS"**)  
 font.setPointSize(21)  
 self.label\_3.setFont(font)  
 self.label\_3.setObjectName(**"label\_3"**)  
 self.label\_4 = QtWidgets.QLabel(self.tab)  
 self.label\_4.setGeometry(QtCore.QRect(50, 130, 341, 31))  
 font = QtGui.QFont()  
 font.setFamily(**"Comic Sans MS"**)  
 font.setPointSize(21)  
 self.label\_4.setFont(font)  
 self.label\_4.setObjectName(**"label\_4"**)  
 self.label\_5 = QtWidgets.QLabel(self.tab)  
 self.label\_5.setGeometry(QtCore.QRect(50, 170, 341, 31))  
 font = QtGui.QFont()  
 font.setFamily(**"Comic Sans MS"**)  
 font.setPointSize(21)  
 self.label\_5.setFont(font)  
 self.label\_5.setObjectName(**"label\_5"**)  
 self.label\_6 = QtWidgets.QLabel(self.tab)  
 self.label\_6.setGeometry(QtCore.QRect(50, 210, 341, 31))  
 font = QtGui.QFont()  
 font.setFamily(**"Comic Sans MS"**)  
 font.setPointSize(21)  
 self.label\_6.setFont(font)  
 self.label\_6.setObjectName(**"label\_6"**)  
 self.label\_9 = QtWidgets.QLabel(self.tab)  
 self.label\_9.setGeometry(QtCore.QRect(50, 250, 711, 81))  
 font = QtGui.QFont()  
 font.setFamily(**"Comic Sans MS"**)  
 font.setPointSize(21)  
 self.label\_9.setFont(font)  
 self.label\_9.setText(**""**)  
 self.label\_9.setPixmap(QtGui.QPixmap(**"C:/Users/Sanverik/Pictures/36.PNG"**))  
 self.label\_9.setObjectName(**"label\_9"**)  
 self.label\_10 = QtWidgets.QLabel(self.tab)  
 self.label\_10.setGeometry(QtCore.QRect(50, 300, 711, 171))  
 font = QtGui.QFont()  
 font.setFamily(**"Comic Sans MS"**)  
 font.setPointSize(21)  
 self.label\_10.setFont(font)  
 self.label\_10.setText(**""**)  
 self.label\_10.setPixmap(QtGui.QPixmap(**"C:/Users/Sanverik/Pictures/37.PNG"**))  
 self.label\_10.setObjectName(**"label\_10"**)  
 self.label\_15 = QtWidgets.QLabel(self.tab)  
 self.label\_15.setGeometry(QtCore.QRect(50, 470, 711, 61))  
 font = QtGui.QFont()  
 font.setFamily(**"Comic Sans MS"**)  
 font.setPointSize(21)  
 self.label\_15.setFont(font)  
 self.label\_15.setText(**""**)  
 self.label\_15.setPixmap(QtGui.QPixmap(**"C:/Users/Sanverik/Pictures/38.PNG"**))  
 self.label\_15.setObjectName(**"label\_15"**)  
 self.tabWidget.addTab(self.tab, **""**)  
 self.tab\_2 = QtWidgets.QWidget()  
 self.tab\_2.setObjectName(**"tab\_2"**)  
 self.line = QtWidgets.QFrame(self.tab\_2)  
 self.line.setGeometry(QtCore.QRect(-3, 170, 1211, 20))  
 self.line.setFrameShape(QtWidgets.QFrame.HLine)  
 self.line.setFrameShadow(QtWidgets.QFrame.Sunken)  
 self.line.setObjectName(**"line"**)  
 self.line\_2 = QtWidgets.QFrame(self.tab\_2)  
 self.line\_2.setGeometry(QtCore.QRect(580, -10, 20, 191))  
 self.line\_2.setFrameShape(QtWidgets.QFrame.VLine)  
 self.line\_2.setFrameShadow(QtWidgets.QFrame.Sunken)  
 self.line\_2.setObjectName(**"line\_2"**)  
 self.commandLinkButton\_11 = QtWidgets.QCommandLinkButton(self.tab\_2)  
 self.commandLinkButton\_11.setGeometry(QtCore.QRect(610, 130, 171, 41))  
 font = QtGui.QFont()  
 font.setFamily(**"Segoe UI"**)  
 font.setPointSize(12)  
 font.setBold(**True**)  
 font.setWeight(75)  
 self.commandLinkButton\_11.setFont(font)  
 icon1 = QtGui.QIcon()  
 icon1.addPixmap(QtGui.QPixmap(**"C:/Icons/disk (1).ico"**), QtGui.QIcon.Normal, QtGui.QIcon.Off)  
 self.commandLinkButton\_11.setIcon(icon1)  
 self.commandLinkButton\_11.setObjectName(**"commandLinkButton\_11"**)  
 self.commandLinkButton\_12 = QtWidgets.QCommandLinkButton(self.tab\_2)  
 self.commandLinkButton\_12.setGeometry(QtCore.QRect(990, 130, 181, 41))  
 font = QtGui.QFont()  
 font.setFamily(**"Segoe UI"**)  
 font.setPointSize(12)  
 font.setBold(**True**)  
 font.setWeight(75)  
 self.commandLinkButton\_12.setFont(font)  
 icon2 = QtGui.QIcon()  
 icon2.addPixmap(QtGui.QPixmap(**"C:/Icons/trash.png"**), QtGui.QIcon.Normal, QtGui.QIcon.Off)  
 self.commandLinkButton\_12.setIcon(icon2)  
 self.commandLinkButton\_12.setObjectName(**"commandLinkButton\_12"**)  
 self.commandLinkButton\_13 = QtWidgets.QCommandLinkButton(self.tab\_2)  
 self.commandLinkButton\_13.setGeometry(QtCore.QRect(790, 130, 181, 41))  
 font = QtGui.QFont()  
 font.setFamily(**"Segoe UI"**)  
 font.setPointSize(12)  
 font.setBold(**True**)  
 font.setWeight(75)  
 self.commandLinkButton\_13.setFont(font)  
 icon3 = QtGui.QIcon()  
 icon3.addPixmap(QtGui.QPixmap(**"C:/Icons/download.ico"**), QtGui.QIcon.Normal, QtGui.QIcon.Off)  
 self.commandLinkButton\_13.setIcon(icon3)  
 self.commandLinkButton\_13.setObjectName(**"commandLinkButton\_13"**)  
 self.groupBox\_6 = QtWidgets.QGroupBox(self.tab\_2)  
 self.groupBox\_6.setGeometry(QtCore.QRect(10, 0, 561, 121))  
 font = QtGui.QFont()  
 font.setFamily(**"Comic Sans MS"**)  
 font.setPointSize(18)  
 self.groupBox\_6.setFont(font)  
 self.groupBox\_6.setObjectName(**"groupBox\_6"**)  
 self.textEdit = QtWidgets.QTextEdit(self.groupBox\_6)  
 self.textEdit.setGeometry(QtCore.QRect(20, 40, 521, 61))  
 self.textEdit.setObjectName(**"textEdit"**)  
 self.groupBox\_13 = QtWidgets.QGroupBox(self.tab\_2)  
 self.groupBox\_13.setGeometry(QtCore.QRect(610, 0, 561, 121))  
 font = QtGui.QFont()  
 font.setFamily(**"Comic Sans MS"**)  
 font.setPointSize(18)  
 self.groupBox\_13.setFont(font)  
 self.groupBox\_13.setObjectName(**"groupBox\_13"**)  
 self.textEdit\_2 = QtWidgets.QTextEdit(self.groupBox\_13)  
 self.textEdit\_2.setGeometry(QtCore.QRect(20, 40, 261, 61))  
 self.textEdit\_2.setObjectName(**"textEdit\_2"**)  
 self.textEdit\_3 = QtWidgets.QTextEdit(self.groupBox\_13)  
 self.textEdit\_3.setGeometry(QtCore.QRect(290, 40, 261, 61))  
 self.textEdit\_3.setObjectName(**"textEdit\_3"**)  
 self.commandLinkButton\_14 = QtWidgets.QCommandLinkButton(self.tab\_2)  
 self.commandLinkButton\_14.setGeometry(QtCore.QRect(10, 130, 81, 41))  
 font = QtGui.QFont()  
 font.setFamily(**"Segoe UI"**)  
 font.setPointSize(12)  
 font.setBold(**True**)  
 font.setWeight(75)  
 self.commandLinkButton\_14.setFont(font)  
 self.commandLinkButton\_14.setIcon(icon1)  
 self.commandLinkButton\_14.setObjectName(**"commandLinkButton\_14"**)  
 self.commandLinkButton\_15 = QtWidgets.QCommandLinkButton(self.tab\_2)  
 self.commandLinkButton\_15.setGeometry(QtCore.QRect(190, 130, 181, 41))  
 font = QtGui.QFont()  
 font.setFamily(**"Segoe UI"**)  
 font.setPointSize(12)  
 font.setBold(**True**)  
 font.setWeight(75)  
 self.commandLinkButton\_15.setFont(font)  
 self.commandLinkButton\_15.setIcon(icon3)  
 self.commandLinkButton\_15.setObjectName(**"commandLinkButton\_15"**)  
 self.commandLinkButton\_16 = QtWidgets.QCommandLinkButton(self.tab\_2)  
 self.commandLinkButton\_16.setGeometry(QtCore.QRect(390, 130, 181, 41))  
 font = QtGui.QFont()  
 font.setFamily(**"Segoe UI"**)  
 font.setPointSize(12)  
 font.setBold(**True**)  
 font.setWeight(75)  
 self.commandLinkButton\_16.setFont(font)  
 self.commandLinkButton\_16.setIcon(icon2)  
 self.commandLinkButton\_16.setObjectName(**"commandLinkButton\_16"**)  
 self.groupBox\_14 = QtWidgets.QGroupBox(self.tab\_2)  
 self.groupBox\_14.setGeometry(QtCore.QRect(10, 180, 1161, 361))  
 font = QtGui.QFont()  
 font.setFamily(**"Comic Sans MS"**)  
 font.setPointSize(18)  
 self.groupBox\_14.setFont(font)  
 self.groupBox\_14.setObjectName(**"groupBox\_14"**)  
 self.tableWidget = QtWidgets.QTableWidget(self.groupBox\_14)  
 self.tableWidget.setGeometry(QtCore.QRect(20, 40, 1121, 241))  
 self.tableWidget.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)  
 self.tableWidget.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)  
 self.tableWidget.setDragEnabled(**True**)  
 self.tableWidget.setShowGrid(**True**)  
 self.tableWidget.setRowCount(10)  
 self.tableWidget.setObjectName(**"tableWidget"**)  
 self.tableWidget.setColumnCount(10)  
 font = QtGui.QFont()  
 font.setFamily(**"Segoe UI"**)  
 font.setPointSize(12)  
 font.setBold(**True**)  
 font.setWeight(75)  
 icon4 = QtGui.QIcon()  
 icon4.addPixmap(QtGui.QPixmap(**"C:/Programing/LAB2/Icons/big.ico"**), QtGui.QIcon.Normal, QtGui.QIcon.Off)  
 self.commandLinkButton\_19 = QtWidgets.QCommandLinkButton(self.tab\_2)  
 self.commandLinkButton\_19.setGeometry(QtCore.QRect(100, 130, 81, 41))  
 font = QtGui.QFont()  
 font.setFamily(**"Segoe UI"**)  
 font.setPointSize(12)  
 font.setBold(**True**)  
 font.setWeight(75)  
 self.commandLinkButton\_19.setFont(font)  
 self.commandLinkButton\_19.setIcon(icon1)  
 self.commandLinkButton\_19.setObjectName(**"commandLinkButton\_19"**)  
 self.tabWidget.addTab(self.tab\_2, **""**)  
 self.tab\_9 = QtWidgets.QWidget()  
 self.tab\_9.setObjectName(**"tab\_9"**)  
 self.line\_4 = QtWidgets.QFrame(self.tab\_9)  
 self.line\_4.setGeometry(QtCore.QRect(-10, 40, 1211, 20))  
 self.line\_4.setFrameShape(QtWidgets.QFrame.HLine)  
 self.line\_4.setFrameShadow(QtWidgets.QFrame.Sunken)  
 self.line\_4.setObjectName(**"line\_4"**)  
 self.label\_7 = QtWidgets.QLabel(self.tab\_9)  
 self.label\_7.setGeometry(QtCore.QRect(280, 60, 911, 471))  
 self.label\_7.setFont(font)  
 self.label\_7.setObjectName(**"label\_7"**)  
 font = QtGui.QFont()  
 font.setFamily(**"Comic Sans MS"**)  
 font.setPointSize(21)  
 self.commandLinkButton\_17 = QtWidgets.QCommandLinkButton(self.tab\_9)  
 self.commandLinkButton\_17.setGeometry(QtCore.QRect(100, 90, 171, 41))  
 font = QtGui.QFont()  
 font.setFamily(**"Segoe UI"**)  
 font.setPointSize(12)  
 font.setBold(**True**)  
 font.setWeight(75)  
 self.commandLinkButton\_17.setFont(font)  
 self.commandLinkButton\_17.setIcon(icon1)  
 self.commandLinkButton\_17.setObjectName(**"commandLinkButton\_17"**)  
 self.commandLinkButton\_18 = QtWidgets.QCommandLinkButton(self.tab\_9)  
 self.commandLinkButton\_18.setGeometry(QtCore.QRect(100, 140, 171, 41))  
 font = QtGui.QFont()  
 font.setFamily(**"Segoe UI"**)  
 font.setPointSize(12)  
 font.setBold(**True**)  
 font.setWeight(75)  
 self.commandLinkButton\_18.setFont(font)  
 self.commandLinkButton\_18.setIcon(icon1)  
 self.commandLinkButton\_18.setObjectName(**"commandLinkButton\_18"**)  
 self.label\_8 = QtWidgets.QLabel(self.tab\_9)  
 self.label\_8.setGeometry(QtCore.QRect(190, 10, 731, 31))  
 font = QtGui.QFont()  
 font.setFamily(**"Comic Sans MS"**)  
 font.setPointSize(21)  
 self.label\_8.setFont(font)  
 self.label\_8.setObjectName(**"label\_8"**)  
 self.tabWidget.addTab(self.tab\_9, **""**)  
 MainWindow.setCentralWidget(self.centralwidget)  
 self.menubar = QtWidgets.QMenuBar(MainWindow)  
 self.menubar.setGeometry(QtCore.QRect(0, 0, 1200, 21))  
 self.menubar.setObjectName(**"menubar"**)  
 MainWindow.setMenuBar(self.menubar)  
 self.statusbar = QtWidgets.QStatusBar(MainWindow)  
 self.statusbar.setObjectName(**"statusbar"**)  
 MainWindow.setStatusBar(self.statusbar)  
  
 self.retranslateUi(MainWindow)  
 self.tabWidget.setCurrentIndex(1)  
 QtCore.QMetaObject.connectSlotsByName(MainWindow)  
  
 **def** retranslateUi(self, MainWindow):  
 \_translate = QtCore.QCoreApplication.translate  
 MainWindow.setWindowTitle(\_translate(**"MainWindow"**, **"Лабораторна робота 4"**))  
 self.label\_2.setText(\_translate(**"MainWindow"**, **"ІНФОРМАЦІЯ"**))  
 self.label\_3.setText(\_translate(**"MainWindow"**, **"Кобилюк Андрій Григорович"**))  
 self.label\_4.setText(\_translate(**"MainWindow"**, **"Група ІО-64"**))  
 self.label\_5.setText(\_translate(**"MainWindow"**, **"11 номер у списку"**))  
  
 self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab), \_translate(**"MainWindow"**, **"INFO"**))  
 self.commandLinkButton\_11.setText(\_translate(**"MainWindow"**, **"Додати пару"**))  
 self.commandLinkButton\_12.setText(\_translate(**"MainWindow"**, **"Очистити"**))  
 self.commandLinkButton\_13.setText(\_translate(**"MainWindow"**, **"Зчитати із файла"**))  
 self.groupBox\_6.setTitle(\_translate(**"MainWindow"**, **"Задати список ребер"**))  
 self.groupBox\_13.setTitle(\_translate(**"MainWindow"**, **"Задати пару вершин"**))  
 self.commandLinkButton\_14.setText(\_translate(**"MainWindow"**, **"1"**))  
 self.commandLinkButton\_15.setText(\_translate(**"MainWindow"**, **"Зчитати із файла"**))  
 self.commandLinkButton\_16.setText(\_translate(**"MainWindow"**, **"Очистити список"**))  
 self.groupBox\_14.setTitle(\_translate(**"MainWindow"**, **"Матриця суміжності"**))  
 self.commandLinkButton\_19.setText(\_translate(**"MainWindow"**, **"2"**))  
 self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab\_2), \_translate(**"MainWindow"**, **"Програма"**))  
 self.commandLinkButton\_17.setText(\_translate(**"MainWindow"**, **"Вперед"**))  
 self.commandLinkButton\_18.setText(\_translate(**"MainWindow"**, **"Назад"**))  
 self.label\_8.setText(\_translate(**"MainWindow"**, **"ФАЙЛ"**))  
 self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab\_9), \_translate(**"MainWindow"**, **"Перегляд"**))  
  
 self.my\_part()  
  
 **def** my\_part(self):  
 *#My variant* self.label\_6.setText(**"Варіант - "**+str(variant.variant(6411)))  
  
 *# Кліки кнопок першого вікна* self.commandLinkButton\_19.clicked.connect(self.see\_graph\_two)  
 self.commandLinkButton\_15.clicked.connect(self.load1)  
 self.commandLinkButton\_16.clicked.connect(self.clear\_one)  
 self.commandLinkButton\_11.clicked.connect(self.add\_tuple)  
 self.commandLinkButton\_12.clicked.connect(self.clear\_tuple)  
 self.commandLinkButton\_13.clicked.connect(self.load\_two)  
 self.commandLinkButton\_14.clicked.connect(self.see\_graph\_one)  
  
 *# Кліки кнопок другого вікна* self.commandLinkButton\_17.clicked.connect(self.next)  
 self.commandLinkButton\_18.clicked.connect(self.prev)  
  
 db = shelve.open(**r"D:\Programing\LAB4\Data\Counter"**)  
 db[**"Counter"**] = -1  
  
 db.close()  
  
 self.makeMatrix()  
  
 **def** clear\_one(self):  
 **def** renew():  
 db = shelve.open(**r"D:\Programing\LAB4\Data\E"**)  
 db[**"E"**] = set()  
 db.close()  
  
 self.textEdit.setPlainText(**""**)  
 renew()  
 self.tableWidget.setColumnCount(0)  
 self.tableWidget.setRowCount(0)  
 self.makeMatrix()  
  
 **def** see\_graph\_one(self):  
  
 **def** coloring(node, color):  
 **for** neighbor **in** G.neighbors(node):  
 color\_of\_neighbor = colors\_of\_nodes.get(neighbor, **None**)  
 **if** color\_of\_neighbor == color:  
 **return False  
  
 return True  
  
 def** get\_color\_for\_node(node):  
 **for** color **in** colors:  
 **if** coloring(node, color):  
 **return** color  
  
 **def** load():  
 db = shelve.open(**r"D:\Programing\LAB4\Data\E"**)  
 E = db[**"E"**]  
 db.close()  
 **return** E  
  
 **def** toNumericEdges(V, E):  
 tmp\_list = list()  
 **for** i **in** E:  
 tmp\_list.append(tuple((V.index(i[0]), V.index(i[1]))))  
 **return** tmp\_list  
  
 **def** getV(E):  
 tmp\_set = set()  
 **for** i **in** E:  
 tmp\_set.add(i[0])  
 tmp\_set.add(i[1])  
 **return** sorted(list(tmp\_set))  
  
 **def** correct(E):  
 result = list()  
 **for** i **in** E:  
 result.append(tuple((i[0]+1,i[1]+1)))  
 **return** result  
  
 E = load()  
 V = getV(E)  
 E = toNumericEdges(V,E)  
  
 G = nx.Graph()  
  
 **for** i **in** correct(E):  
 G.add\_edge(\*i)  
  
 colors = [**'Red'**, **'Blue'**, **'Green'**, **'Yellow'**, **'Pink'**, **'Orange'**, **'Gray'**, **'Purple'**, **'Brown'**, **'Navy'**]  
  
 colors\_of\_nodes = {}  
  
 pos = nx.spring\_layout(G) *# positions for all nodes* **for** node **in** G.nodes():  
 colors\_of\_nodes[node] = get\_color\_for\_node(node)  
  
 color\_map = list(colors\_of\_nodes.values())  
 nx.draw(G, node\_color=color\_map, with\_labels=**True**)  
  
 plt.axis(**'off'**)  
 path = **r"D:\Programing\LAB4\Graphs image\image"**+str(random.randint(0,1000))+**".png"** plt.savefig(os.path.normpath(path)) *# save as png* plt.show() *# display* **def** load1(self):  
 db = shelve.open(**r"D:\Programing\LAB4\Data\E"**)  
 E = sorted(list(db[**"E"**]))  
 db.close()  
 s = **""  
 for** i **in** E:  
 s+=**" "** s+=i[0]  
 s+=**" "** s+=i[1]  
 tmp = s.split(**" "**)  
 **while "" in** tmp:  
 tmp.remove(**""**)  
 s = **" "  
 for** i **in** range(0, len(tmp), 2):  
 s += str(tuple((tmp[i], tmp[i + 1])))  
 **if** i != (len(tmp) - 2):  
 s += **","** self.textEdit.setPlainText(s)  
  
 **def** clear\_tuple(self):  
 self.textEdit.setPlainText(**""**)  
 self.textEdit\_2.setPlainText(**""**)  
 self.textEdit\_3.setPlainText(**""**)  
  
 **def** add\_tuple(self):  
 **def** choose\_one\_b(s):  
 tmp\_list = s.split(**" "**)  
 **while "" in** tmp\_list:  
 **try**:  
 tmp\_list.remove(**""**)  
 **except** Exception:  
 **pass  
 if** len(tmp\_list) == 1:  
 **return True  
 else**:  
 **return False  
  
 def** choose\_one(s):  
 tmp\_list = s.split(**" "**)  
 **while "" in** tmp\_list:  
 **try**:  
 tmp\_list.remove(**""**)  
 **except** Exception:  
 **pass  
 return** tmp\_list[0]  
  
 **def** load():  
 db = shelve.open(**r"D:\Programing\LAB4\Data\E"**)  
 E = sorted(list(db[**"E"**]))  
 db.close()  
 s = **""  
 for** i **in** E:  
 s += **" "** s += i[0]  
 s += **" "** s += i[1]  
 **return** s  
  
 **def** save\_own(object):  
 db = shelve.open(**r"D:\Programing\LAB4\Data\E\_two"**)  
 obj1 = object  
 db[**"E\_two"**] = obj1  
 db.close()  
  
 **def** renew(tuple1):  
 db = shelve.open(**r"D:\Programing\LAB4\Data\E"**)  
 E = db[**"E"**]  
 E.add(tuple1)  
 db[**"E"**] = set(sorted(list(E)))  
 db.close()  
  
  
 **if** choose\_one\_b(self.textEdit\_2.toPlainText()) **and** choose\_one\_b(self.textEdit\_3.toPlainText()):  
 s1 = choose\_one(self.textEdit\_2.toPlainText())  
 s2 = choose\_one(self.textEdit\_3.toPlainText())  
 save\_own(tuple((s1,s2)))  
 E = load()+ **" "**+ s1 +**" "**+ s2  
 renew(tuple((s1,s2)))  
 tmp = E.split(**" "**)  
 **while "" in** tmp:  
 tmp.remove(**""**)  
 s = **" "  
 for** i **in** range(0, len(tmp), 2):  
 s += str(tuple((tmp[i], tmp[i + 1])))  
 **if** i != (len(tmp) - 2):  
 s += **","** self.textEdit.setPlainText(s)  
 self.textEdit\_2.setPlainText(**""**)  
 self.textEdit\_3.setPlainText(**""**)  
 self.load1()  
 **else**:  
 reply = QMessageBox.critical(MainWindow, **'Помилка'**, **"Виберіть ОДИН елемент в кожній клітинці"**, QMessageBox.Ok)  
  
 **def** load\_two(self):  
 db = shelve.open(**r"D:\Programing\LAB4\Data\E\_two"**)  
 E = db[**"E\_two"**]  
 db.close()  
 self.textEdit\_2.setPlainText(E[0])  
 self.textEdit\_3.setPlainText(E[1])  
  
 **def** makeMatrix(self):  
 n = 7  
 self.tableWidget.setColumnCount(n)  
 self.tableWidget.setRowCount(n)  
  
 *# STRUCTURE* **for** i **in** range(n):  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget.setVerticalHeaderItem(i, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget.setHorizontalHeaderItem(i, item)  
  
 *# TEXT* **for** i **in** range(n):  
 \_translate = QtCore.QCoreApplication.translate  
 item = self.tableWidget.verticalHeaderItem(i)  
 item.setText(\_translate(**"MainWindow"**, **"v"** + str(i + 1)))  
 item = self.tableWidget.horizontalHeaderItem(i)  
 item.setText(\_translate(**"MainWindow"**, **"v"** + str(i + 1)))  
  
 *# VALUE CELLS* **for** i **in** range(n):  
 **for** j **in** range(n):  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget.setItem(i, j, item)  
  
 **def** see\_graph\_two(self):  
  
 **def** getEmtyNxN(n):  
 result = list()  
 tmp\_list = []  
 **for** i **in** range(n):  
 result.append(copy.deepcopy(tmp\_list))  
 **for** j **in** range(n):  
 result[i].append(copy.deepcopy(tmp\_list))  
 **return** result  
  
 **def** check(matrix):  
 tmp = matrix  
 **for** i **in** range(len(tmp)):  
 **for** j **in** range(len(tmp)):  
 **if** tmp[i][j] == **""**:  
 tmp[i][j] = 0  
 **else**:  
 **try**:  
 s = tmp[i][j]  
 tmp[i][j] = int(s)-int(s)+1  
 **except** Exception:  
 tmp[i][j] = 1  
 **return** tmp  
  
 **def** coloring(node, color):  
 **for** neighbor **in** G.neighbors(node):  
 color\_of\_neighbor = colors\_of\_nodes.get(neighbor, **None**)  
 **if** color\_of\_neighbor == color:  
 **return False  
  
 return True  
  
 def** get\_color\_for\_node(node):  
 **for** color **in** colors:  
 **if** coloring(node, color):  
 **return** color  
  
 n = 7  
  
 matrix = getEmtyNxN(n)  
  
 **for** i **in** range(n):  
 **for** j **in** range(n):  
 matrix[i][j] = self.tableWidget.item(i,j).text()  
  
 matrix = check(matrix)  
 G = nx.Graph()  
  
 **for** i **in** range(len(matrix)):  
 **for** j **in** range(len(matrix)):  
 **if** matrix[i][j] == 0:  
 **pass  
 else**:  
 G.add\_edge(i+1, j+1)  
  
 colors = [**'Red'**, **'Blue'**, **'Green'**, **'Yellow'**, **'Pink'**, **'Orange'**, **'Gray'**, **'Purple'**, **'Brown'**, **'Navy'**]  
  
 colors\_of\_nodes = {}  
  
 **for** node **in** G.nodes():  
 colors\_of\_nodes[node] = get\_color\_for\_node(node)  
  
 color\_map = list(colors\_of\_nodes.values())  
 nx.draw(G, node\_color=color\_map, with\_labels=**True**)  
  
 plt.axis(**'off'**)  
 path = **r"D:\Programing\LAB4\Graphs image\image"**+str(random.randint(0,1000))+**".png"** plt.savefig(os.path.normpath(path)) *# save as png* plt.show() *# display* **def** next(self):  
  
 **def** loadCounter():  
 db = shelve.open(**r"D:\Programing\LAB4\Data\Counter"**)  
 counter = db[**"Counter"**]  
 db.close()  
 **return** counter  
  
 **def** renewCounter(counter):  
 db = shelve.open(**r"D:\Programing\LAB4\Data\Counter"**)  
 db[**"Counter"**] = counter  
 db.close()  
  
 list\_dir = os.listdir(**r"D:\Programing\LAB4\Graphs image"**)  
 i = loadCounter()  
 **if** len(list\_dir)-2<i:  
 **pass  
 else**:  
 i +=1  
 renewCounter(i)  
 self.label\_8.setText(str(os.path.normpath(**"D:/Programing/LAB4/Graphs image/"**+str(list\_dir[i]))))  
 self.label\_7.setText(**""**)  
 self.label\_7.setPixmap(QtGui.QPixmap(os.path.normpath(**"D:/Programing/LAB4/Graphs image/"**+str(list\_dir[i]))))  
  
 **def** prev(self):  
  
 **def** loadCounter():  
 db = shelve.open(**r"D:\Programing\LAB4\Data\Counter"**)  
 counter = db[**"Counter"**]  
 db.close()  
 **return** counter  
  
 **def** renewCounter(counter):  
 db = shelve.open(**r"D:\Programing\LAB4\Data\Counter"**)  
 db[**"Counter"**] = counter  
 db.close()  
  
 list\_dir = os.listdir(**r"D:\Programing\LAB4\Graphs image"**)  
 i = loadCounter()  
  
 **if** i<=0:  
 **pass  
 else**:  
 i -=1  
 renewCounter(i)  
 self.label\_8.setText(str(os.path.normpath(**"D:/Programing/LAB4/Graphs image/"** + str(list\_dir[i]))))  
 self.label\_7.setText(**""**)  
 self.label\_7.setPixmap(QtGui.QPixmap(os.path.normpath(**"D:/Programing/LAB4/Graphs image/"**+str(list\_dir[i]))))  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 **import** sys  
 app = QtWidgets.QApplication(sys.argv)  
 MainWindow = QtWidgets.QMainWindow()  
 ui = Ui\_MainWindow()  
 ui.setupUi(MainWindow)  
 MainWindow.show()  
 sys.exit(app.exec\_())