Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ)

КАФЕДРА «Информационная безопасность» (ИУ8)

Отчёт

по лабораторной работе № 6
по дисциплине «Интеллектуальные технологии информационной безопасности»

Тема: «Алгоритмы кластерного анализа данных»

Вариант 1

Выполнил: Антипов И.С.,
студент группы ИУ8-63

Проверил: Волосова Н.К.,
преподаватель каф. ИУ8

г. Москва,
2021 г.

1. Цель работы

Исследовать применение основных алгоритмов кластерного анализа, включая их модификации, на примере различных типов данных.

2. Условие

Алгоритм: k-средних

$$S = \sum_{i=1}^{k} \sum_{x_j \in Y_i} \left(x_j - \mu_i\right)^2$$

где k – количество кластеров (задано заранее); Y_i – полученные кластеры, $i = 1, \dots, k;\ \mu_i$- центры масс $x_j \in Y_i$

Исходные кластеризуемые данные: Координатные точки на форме приложения

Расстояние: Евклида, Чебышева

Координатные точки: (20, -10); (1, 10); (43, -9); (25, 20); (15, -12); (-29, 21); (10, 4); (-30, 11); (-24, 8)

Координаты точек кластеров:

Евклид: (14, 6); (3, 0); (-10, 15)

Чебышев: (20, 8); (-3, 18)

3. Ход работы

Суть алгоритма заключается в следующем: на каждой итерации перевычисляется центр масс для каждого кластера, полученного на предыдущем шаге.

Затем кластеризуемые точки разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике.

Алгоритм завершается, когда на i-й итерации не изменяется центр масс кластеров.

Ниже представлен результат работы программы для расстояния Евклида (см. рис. 1 – 3):

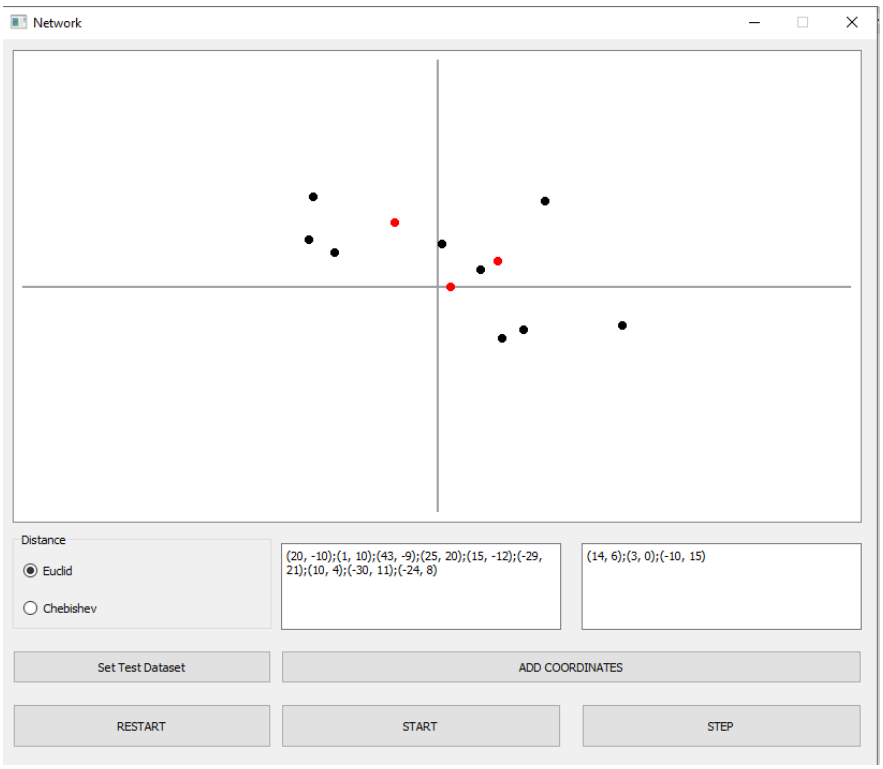

Рисунок 1 – Начальные данные

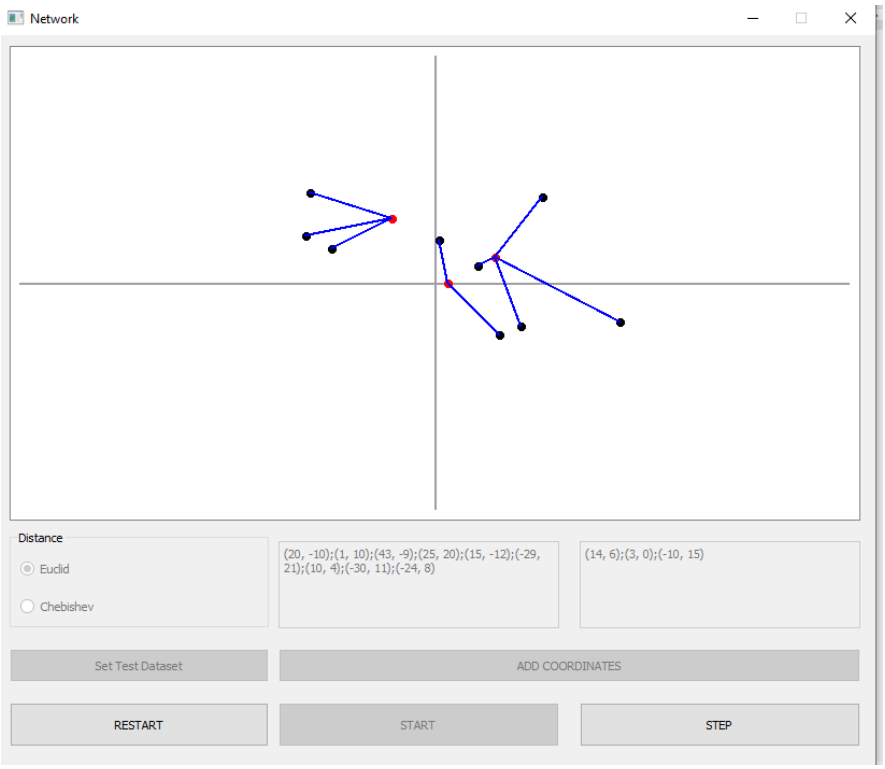

Рисунок 2 – Кластеризация на 1 шаге

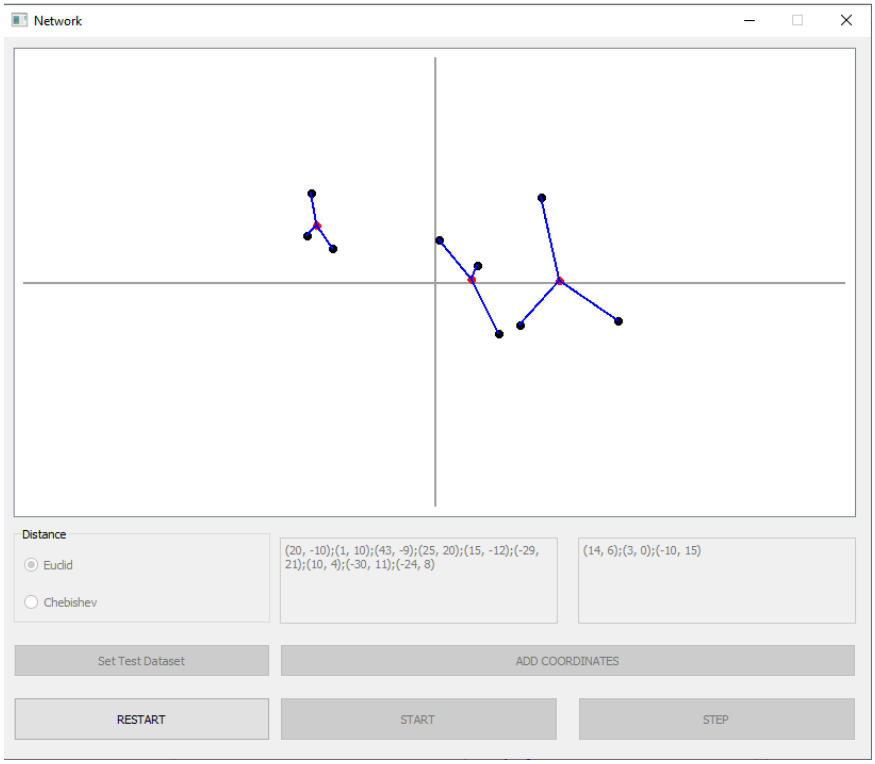

Рисунок 3 – Финальная кластеризация

Ниже представлен результат работы программы для расстояния Евклида (см. рис. 4 – 6):

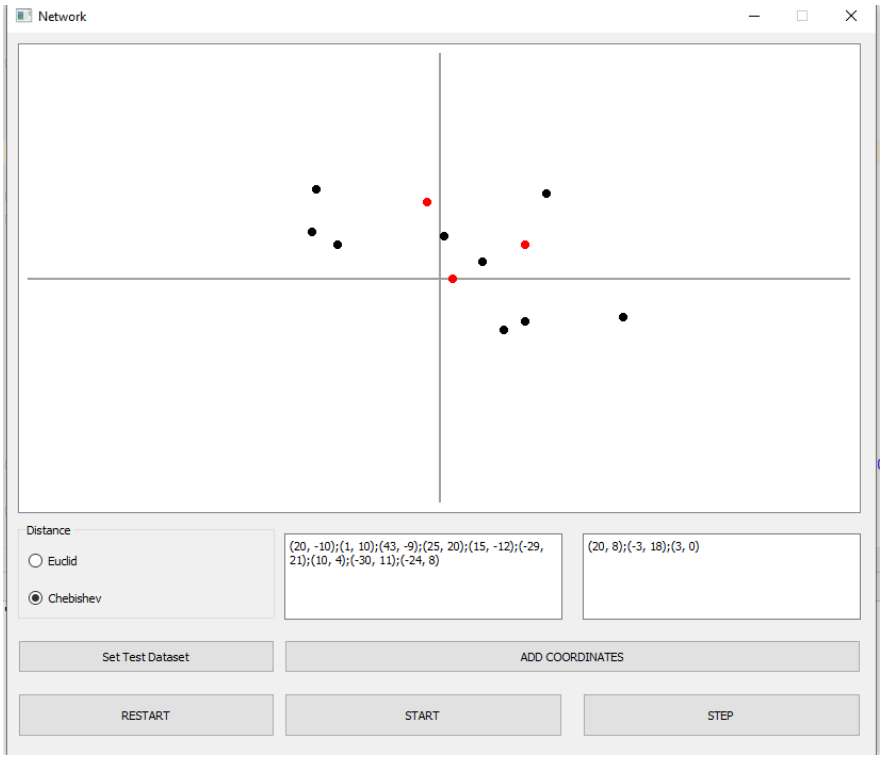

Рисунок 4 – Начальные данные

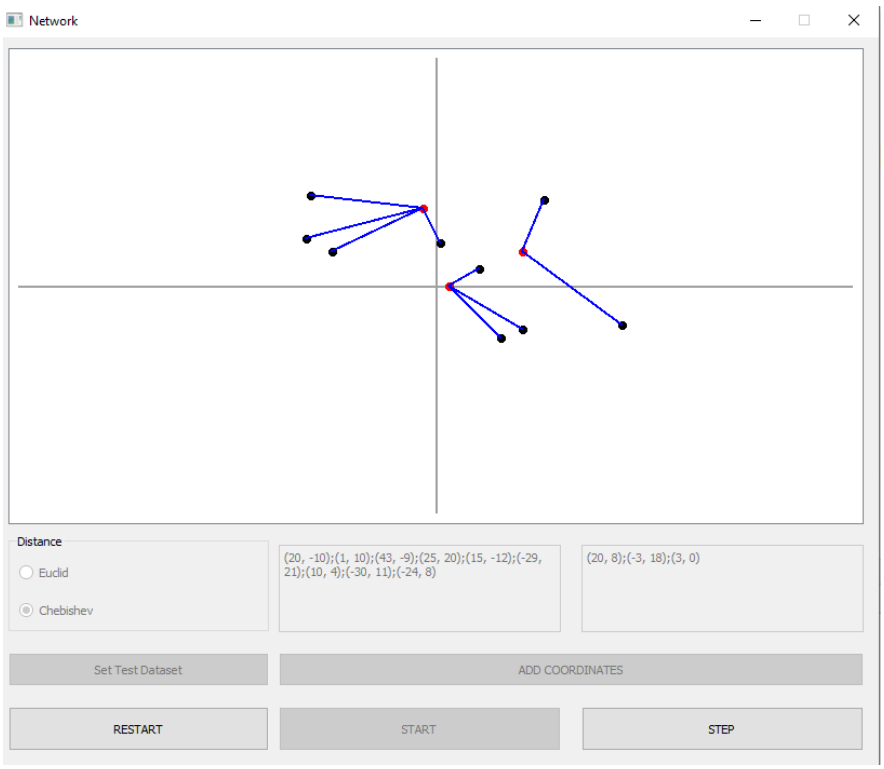

Рисунок 5 – Кластеризация на 1 шаге

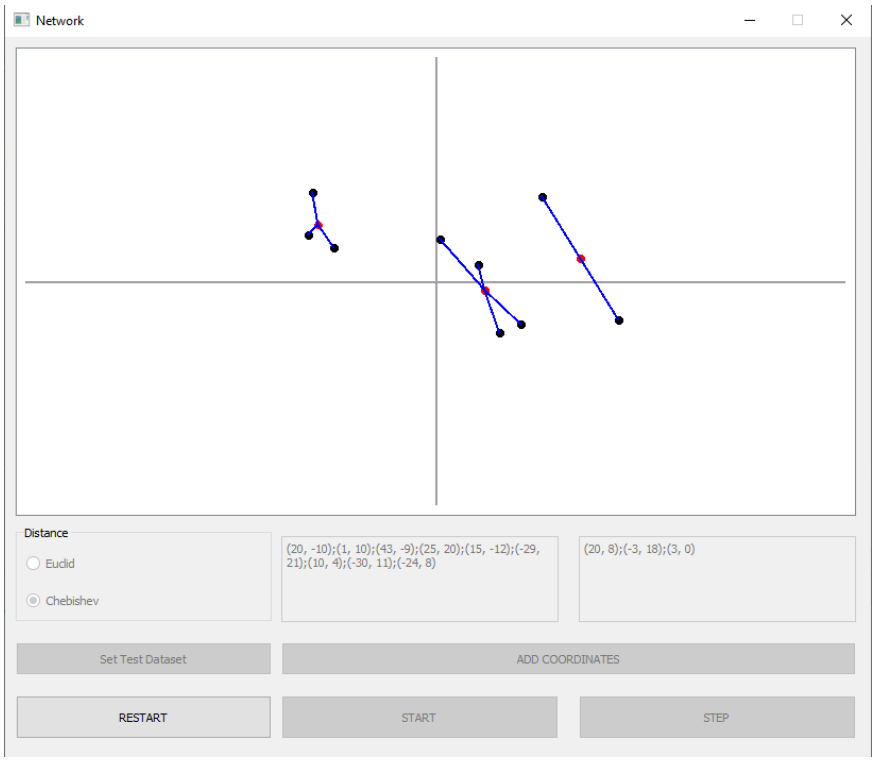

Рисунок 6 – Финальная кластеризация

Код программы приведен в Приложении А.

4. Выводы

В ходе выполнения лабораторной работы было произведена кластеризация с помощью метода k-средних, используя расстояние Евклида и Чебышева.

Приложение А. Исходный код программы

Файл main.py

```
import math
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QPushButton, QGraphicsView, QRadioButton
import sys
import itertools

TEST_COORDS = "(20, -10);(1, 10);(43, -9);(25, 20);(15, -12);(-29, 21);(10,
4);(-30, 11);(-24, 8)"
EUCLID_TEST_CENTERS = "(14, 6);(3, 0);(-10, 15)"
CHEBISHEV_TEST_CENTERS = "(20, 8);(-3, 18);(3, 0)"

class Ui_Form(object):
    def __init__(self, obj):
        super().__init__()
        self.setupUi(obj)
        self.retranslateUi(obj)
        self.graphicsView = QGraphicsView(obj)
        self.graphicsView.setGeometry(QtCore.QRect(10, 10, 791, 441))
        self.graphicsView.setObjectName("graphicsView")
        self.scene = QtWidgets.QGraphicsScene()
        self.graphicsView.setScene(self.scene)

        pen = QtGui.QPen(QtCore.Qt.GlobalColor.gray)
        for i in range(-1 * self.graphicsView.height() // 2 + 10,
self.graphicsView.height() // 2 - 10):
            r1 = QtCore.QRectF(QtCore.QPointF(0, i), QtCore.QSizeF(1, 1))
            self.scene.addRect(r1, pen)


        for i in range(-1 * self.graphicsView.width() // 2 + 10,
self.graphicsView.width() // 2 - 10):
            r2 = QtCore.QRectF(QtCore.QPointF(i, 0), QtCore.QSizeF(1, 1))
            self.scene.addRect(r2, pen)


        self.coordsContainer = []
        self.centersContainer = []
        self.clastersContainer = []
        self.distance = None

    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(815, 678)

        self.startPushButton = QPushButton(Form)

self.startPushButton.clicked.connect(self.startPushButton_button_clicked)
        self.startPushButton.setGeometry(QtCore.QRect(260, 620, 261, 41))
        self.startPushButton.setObjectName("startPushButton")

        self.coordsTextBox = QtWidgets.QPlainTextEdit(Form)
        self.coordsTextBox.setGeometry(QtCore.QRect(260, 470, 261, 81))
        self.coordsTextBox.setObjectName("coordsTextBox")

        self.CentersTextBox = QtWidgets.QPlainTextEdit(Form)
        self.CentersTextBox.setGeometry(QtCore.QRect(540, 470, 261, 81))
        self.CentersTextBox.setObjectName("CentersTextBox")
```

```
        self.addCordsPushButton = QPushButton(Form)

self.addCordsPushButton.clicked.connect(self.addCordsPushButton_button_clicked)
        self.addCordsPushButton.setGeometry(QtCore.QRect(260, 570, 541, 31))
        self.addCordsPushButton.setObjectName("addCordsPushButton")

        self.groupBox = QtWidgets.QGroupBox(Form)
        self.groupBox.setGeometry(QtCore.QRect(10, 460, 241, 91))
        self.groupBox.setObjectName("groupBox")

        self.euclidRadioButton = QRadioButton(self.groupBox)
        self.euclidRadioButton.toggled.connect(self.euclidRadioButton_clicked)
        self.euclidRadioButton.setGeometry(QtCore.QRect(10, 20, 221, 31))
        self.euclidRadioButton.setObjectName("euclidRadioButton")

        self.chebishevRadioButton = QRadioButton(self.groupBox)

self.chebishevRadioButton.toggled.connect(self.chebishevRadioButton_clicked)
        self.chebishevRadioButton.setGeometry(QtCore.QRect(10, 50, 221, 41))
        self.chebishevRadioButton.setObjectName("chebishevRadioButton")

        self.stepPushButton = QPushButton(Form)
        self.stepPushButton.clicked.connect(self.stepPushButton_button_clicked)
        self.stepPushButton.setGeometry(QtCore.QRect(540, 620, 261, 41))
        self.stepPushButton.setObjectName("stepPushButton")

        self.restartPushButton = QPushButton(Form)

self.restartPushButton.clicked.connect(self.restartPushButton_button_clicked)
        self.restartPushButton.setGeometry(QtCore.QRect(10, 620, 241, 41))
        self.restartPushButton.setObjectName("restartPushButton")

        self.testPushButton = QPushButton(Form)
        self.testPushButton.clicked.connect(self.testPushButton_button_clicked)
        self.testPushButton.setGeometry(QtCore.QRect(10, 570, 241, 31))
        self.testPushButton.setObjectName("testPushButton")

        self.retranslateUi(Form)
        QtCore.QMetaObject.connectSlotsByName(Form)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        self.startPushButton.setText(_translate("Form", "START"))
        self.addCordsPushButton.setText(_translate("Form", "ADD COORDINATES"))
        self.groupBox.setTitle(_translate("Form", "Distance"))
        self.euclidRadioButton.setText(_translate("Form", "Euclid"))
        self.chebishevRadioButton.setText(_translate("Form", "Chebishev"))
        self.stepPushButton.setText(_translate("Form", "STEP"))
        self.restartPushButton.setText(_translate("Form", "RESTART"))
        self.testPushButton.setText(_translate("Form", "Set Test Dataset"))

    def drawLineToDot(self):
        pen = QtGui.QPen(QtCore.Qt.GlobalColor.blue)
        brush = QtGui.QBrush(QtCore.Qt.GlobalColor.blue)
        pen.setWidth(2)
        pen.setColor(QtCore.Qt.GlobalColor.blue)

        for i in range(len(self.clastersContainer)):
            for j in self.clastersContainer[i]:
                self.scene.addLine(QtCore.QLineF(4* j[0], -4* j[1],
```

```
4*self.centersContainer[i][0], -4*self.centersContainer[i][1]), pen)


    def stringParser(self, coords, centers):
        coords_l = None
        centers_l = None
        try:
            coords_string_array = coords.split(';')
            centers_string_array = centers.split(';')
            coords_l = []
            centers_l = []

            for i in coords_string_array:
                l = [float(k) for k in i.strip('()').split(',')]
                coords_l.append(l)

            for i in centers_string_array:
                l = [float(k) for k in i.strip('()').split(',')]
                centers_l.append(l)
        except:
            self.CentersTextBox.clear()
            self.coordsTextBox.clear()
            msg = QtWidgets.QMessageBox()
            msg.setIcon(QtWidgets.QMessageBox.Critical)
            msg.setText("Format Error")
            msg.setInformativeText('Follow the format!')
            msg.setWindowTitle("Error")
            msg.setStyleSheet("QLabel{font-size: 20px;}")
            msg.exec_()
            pass
        return coords_l, centers_l

    def add_coordinates_to_GraphView(self):
        pen = QtGui.QPen(QtCore.Qt.GlobalColor.black)
        brush = QtGui.QBrush(QtCore.Qt.GlobalColor.black)
        side = 4
        for i in self.coordsContainer:
            self.scene.addEllipse(i[0] * side - 3, -1* i[1] * side - 3, 7, 7,
pen, brush)

        pen = QtGui.QPen(QtCore.Qt.GlobalColor.red)
        brush = QtGui.QBrush(QtCore.Qt.GlobalColor.red)

        for i in self.centersContainer:
            self.scene.addEllipse(i[0] * side - 3, -1* i[1] * side - 3, 7, 7,
pen, brush)

    def addCordsPushButton_button_clicked(self):
        coordinates = self.coordsTextBox.toPlainText()
        centers = self.CentersTextBox.toPlainText()
        if coordinates == '' or centers == '':
            msg = QtWidgets.QMessageBox()
            msg.setIcon(QtWidgets.QMessageBox.Critical)
            msg.setText("Data Empty")
            msg.setInformativeText('Please, enter coords')
            msg.setWindowTitle("Error")
            msg.setStyleSheet("QLabel{font-size: 20px;}")
            msg.exec_()
            return

        coordinates_l, centers_l = self.stringParser(coordinates, centers)
```

```
    if coordinates_l is not None and centers_l is not None:
        co = self.coordsContainer.copy()
        ce = self.centersContainer.copy()

        co += coordinates_l.copy()
        ce += centers_l.copy()

        co.sort()
        ce.sort()

        co_new = list(num for num, _ in itertools.groupby(co))
        ce_new = list(num for num, _ in itertools.groupby(ce))

        self.centersContainer = ce_new.copy()
        self.coordsContainer = co_new.copy()

        print(self.centersContainer)
        print(self.coordsContainer)

        self.add_coordinates_to_GraphView()


def startPushButton_button_clicked(self):
    if self.coordsContainer == [] or self.centersContainer == []:
        msg = QtWidgets.QMessageBox()
        msg.setIcon(QtWidgets.QMessageBox.Critical)
        msg.setText("Data Empty")
        msg.setInformativeText('Please, enter coords')
        msg.setWindowTitle("Error")
        msg.setStyleSheet("QLabel{font-size: 20px;}")
        msg.exec_()
        return
    self.chebishevRadioButton.setEnabled(False)
    self.euclidRadioButton.setEnabled(False)
    self.addCordsPushButton.setEnabled(False)
    self.coordsTextBox.setEnabled(False)
    self.CentersTextBox.setEnabled(False)
    self.startPushButton.setEnabled(False)
    self.testPushButton.setEnabled(False)

    if self.distance == 'E':

        for _ in range(len(self.centersContainer)):
            self.clastersContainer.append([])

        for i in self.coordsContainer:
            range_l = []
            for c in self.centersContainer:
                range_l.append(math.sqrt((i[0]-c[0])**2 + (i[1]-c[1])**2))

            minindex = range_l.index(min(range_l))
            self.clastersContainer[minindex].append(i)
        self.drawLineToDot()
    elif self.distance == 'H':
        for _ in range(len(self.centersContainer)):
            self.clastersContainer.append([])

        for i in self.coordsContainer:
            range_l = []
            for c in self.centersContainer:
                range_l.append(max(abs(i[0]-c[0]) , abs(i[1]-c[1])))
```

```
                minindex = range_l.index(min(range_l))
                self.clastersContainer[minindex].append(i)
            self.drawLineToDot()


    def stepPushButton_button_clicked(self):
        if self.centersContainer is None or self.coordsContainer is None:
            msg = QtWidgets.QMessageBox()
            msg.setIcon(QtWidgets.QMessageBox.Critical)
            msg.setText("Empty Error")
            msg.setInformativeText('Not enough dots!')
            msg.setWindowTitle("Error")
            msg.setStyleSheet("QLabel{font-size: 20px;}")
            msg.exec_()
            return

        claster_backup = self.clastersContainer.copy()
        new_centers = []
        for i in self.clastersContainer:
            new_x, new_y = 0, 0
            for k in i:
                new_x += k[0]
                new_y += k[1]
            new_x /= len(i)
            new_y /= len(i)
            new_centers.append([new_x, new_y])

        self.centersContainer = new_centers.copy()
        self.redrow(False)
        self.add_coordinates_to_GraphView()
        self.clastersContainer.clear()
        for _ in range(len(self.centersContainer)):
            self.clastersContainer.append([])

        for i in self.coordsContainer:
            range_l = []
            for c in new_centers:
                range_l.append(math.sqrt((i[0] - c[0]) ** 2 + (i[1] - c[1]) **
2))

            minindex = range_l.index(min(range_l))
            self.clastersContainer[minindex].append(i)
        self.drawLineToDot()
        new_back_clasters = self.clastersContainer.copy()

        if claster_backup == new_back_clasters:
            self.stepPushButton.setEnabled(False)


    def redrow(self, full):
        self.scene.clear()
        pen = QtGui.QPen(QtCore.Qt.GlobalColor.gray)
        for i in range(-1 * self.graphicsView.height() // 2 + 10,
self.graphicsView.height() // 2 - 10):
            r1 = QtCore.QRectF(QtCore.QPointF(0, i), QtCore.QSizeF(1, 1))
            self.scene.addRect(r1, pen)

        for i in range(-1 * self.graphicsView.width() // 2 + 10,
self.graphicsView.width() // 2 - 10):
            r2 = QtCore.QRectF(QtCore.QPointF(i, 0), QtCore.QSizeF(1, 1))
            self.scene.addRect(r2, pen)
        if not full:
```

```
            pen2 = QtGui.QPen(QtCore.Qt.GlobalColor.black)
            brush2 = QtGui.QBrush(QtCore.Qt.GlobalColor.black)

            side = 4
            for i in self.coordsContainer:
                self.scene.addEllipse(i[0] * side - 3, -1 * i[1] * side - 3, 7,
7, pen2, brush2)

    def restartPushButton_button_clicked(self):
        self.chebishevRadioButton.setEnabled(True)
        self.euclidRadioButton.setEnabled(True)
        self.addCordsPushButton.setEnabled(True)
        self.coordsTextBox.setEnabled(True)
        self.CentersTextBox.setEnabled(True)
        self.testPushButton.setEnabled(True)
        self.startPushButton.setEnabled(True)
        self.stepPushButton.setEnabled(True)
        self.redrow(True)

        self.coordsContainer.clear()
        self.centersContainer.clear()
        self.clastersContainer.clear()

    def testPushButton_button_clicked(self):
        self.coordsTextBox.setPlainText(TEST_COORDS)
        if self.distance == 'E':
            self.CentersTextBox.setPlainText(EUCLID_TEST_CENTERS)
        elif self.distance == 'H':
            self.CentersTextBox.setPlainText(CHEBISHEV_TEST_CENTERS)
        else:
            self.coordsTextBox.clear()
            msg = QtWidgets.QMessageBox()
            msg.setIcon(QtWidgets.QMessageBox.Critical)
            msg.setText("Distance not set")
            msg.setInformativeText('Please, pick the distance')
            msg.setWindowTitle("Error")
            msg.setStyleSheet("QLabel{font-size: 20px;}")
            msg.exec_()
            pass

    def euclidRadioButton_clicked(self):
        if self.euclidRadioButton.isChecked():
            self.distance = 'E'

    def chebishevRadioButton_clicked(self):
        if self.chebishevRadioButton.isChecked():
            self.distance = 'H'


if __name__ == '__main__':
    app = QtWidgets.QApplication(sys.argv)
    widget = QtWidgets.QWidget()
    app2 = Ui_Form(widget)
    widget.setWindowTitle("Network")
    widget.setFixedWidth(815)
    widget.setFixedHeight(678)
    widget.show()

    exit(app.exec_())
```