МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Тихоокеанский государственный университет»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

«Изучение и реализация алгоритмов дискретной математике.»

«Поиск кратчайшего пути с помощью алгоритма Беллмана–Форда.»

Текстовый документ курсовой работы

по дисциплине «Алгоритмы дискретной математики»

КР. 180008865.ТД

Выполнил студент Воронин В.В.

Факультет, группа ФКФН, ПО(аб) – 81

Руководитель работы Бахрушина Г.И.

Виза: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(доработать, к защите и т.д.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Хабаровск – 2020г.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc40689235)

[1 Постановка задачи (в соответствии с вариантом) 4](#_Toc40689236)

[2 Описание алгоритма. 5](#_Toc40689237)

[3 Описание программы 6](#_Toc40689238)

[3.1 Назначение программы 6](#_Toc40689239)

[3.2 Язык и среда программирования 6](#_Toc40689240)

[3.3 Описание структур данных и функций 6](#_Toc40689241)

[ЗАКЛЮЧЕНИЕ 8](#_Toc40689242)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 9](#_Toc40689243)

[ПРИЛОЖЕНИЕ А 10](#_Toc40689244)

[ПРИЛОЖЕНИЕ Б 22](#_Toc40689245)

## ВВЕДЕНИЕ

В ходе курсовой работы необходимо реализовать поиск матрицы достижимости ориентированного графа.

Матрица достижимости [простого](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C_%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%BE%D0%B2_%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D0%B8_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2#%D0%9F) ориентированного графа G=(V,A) — бинарная матрица замыкания по транзитивности отношения A (оно задаётся матрицей смежности графа). Таким образом, в матрице достижимости хранится информация о существовании путей между вершинами орграфа.

# Постановка задачи (в соответствии с вариантом)

Осуществить поиск матрицы достижимости для ориентированного графа. Пользователь строит граф, используя предложенные инструменты. Также реализовать возможность импорта и экспорта графа в текстовый файл. Программа должна выводить матрицы достижимости, смежности и инцидентности.

# Описание алгоритма.

Для хранения результатов работы алгоритма заведем одномерный массив d[]. В каждом его i-ом элементе будет храниться значение кратчайшего пути из вершины s до вершины i (если таковое имеется). Изначально, присвоим элементам массива d[] значения равные условной бесконечности (например, число заведомо большее суммы всех весов), а в элемент d[s] запишем нуль.

Так мы задействовали известную и необходимую информацию, а именно известно, что наилучший путь из вершины s в нее же саму равен 0, и необходимо предположить недоступность других вершин из s. По мере выполнения алгоритма, для некоторых из них, это условие окажется ложным, и вычисляться оптимальные стоимости путей до этих вершин из s.

Задан граф G=(V, E), n=|V|, а m=|E|. Обозначим смежные вершины этого графа символами v и u (vÎV и uÎV), а вес ребра (v, u) символом w. Иначе говоря, вес ребра, выходящего из вершины v и входящего в вершину u, будет равен w.

Тогда ключевая часть алгоритма Беллмана — Форда примет следующий вид:

Для i от 1 до n-1 выполнять  
Для j от 1 до m выполнять  
Если d[v] + w(v, u) < d[u] то  
d[u] < d[v] + w(v, u)

На каждом n-ом шаге осуществляются попытки улучшить значения элементов массива d[]: если сумма, составленная из веса ребра w(v, u) и веса хранящегося в элементе d[v], меньше веса d[u], то она присваивается последнему. /1-4/

# Описание программы

## 3.1 Назначение программы

Разработанная программа представляет собой визуальное приложение по созданию графа и нахождению кратчайшего пути по заданным вершинам. Функционирование программы и код функций представлены в приложении А и Б соответственно.

## 3.2 Язык и среда программирования

Программа была создана на языке Python 3.52 с использованием компилятора Pyinstaller /6/.

## 3.3 Описание структур данных и функций

**def BellmanFord(from\_, to\_, matrix) –** функция для вычисления кратчайшего пути с помощью алгоритма Беллмана–Форда.

**def minOfArray(lst) –** функция для нахождения минимального элемента в массиве.

**class Graph(QWidget) –** класс графа.

**def drawPoints(self, qp) –** функция для создания вершин графа.

**def arrows(self, qp) –** функция для создания дуг графа.

**def turn(self, x, y, alpha) –** функция для разворота точки на угол alpha.

**def drawFrame(self, qp, rec) –** функция для создания рамки по периметру поля графа.

**def drawText(self, qp, rec) –** функция для нумерации вершин и указания веса дуг.

**class Window(QtWidgets.QMainWindow) –** класс основного окна.

**def updateListArcs(self, comboBox) –** функция обновления выпадающих списков дуг.

**def updateListVex(self, comboBox) –** функция обновления списка вершин.

**def updateListArcs(self, comboBox) –** функция обновления выпадающих списков дуг

**def addVertex(self) –** функция добавления вершины в граф.

**def deleteVertex(self) –** функция удаления вершины из графа.

**def addArc(self) –** функция добавления дуги в граф.

**def deleteArc(self) –** функция удаления дуги из графа.

**def calculate(self) –** функция создания матрицы из наборов вершин и дуг, а затем вызов функции для вычисления кратчайшего пути с помощью алгоритма Беллмана–Форда.

# ЗАКЛЮЧЕНИЕ

Выполнение курсовой работы завершилось созданием программного продукта, который обладает функциональными возможностями, определенными заданием. Для разработки программного продукта был применен визуальный способ создания взвешенного ориентированного графа.

В приложении А представлен код программы. В приложении Б приведены скриншоты, полученные в результате тестирования программы.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Р. Хаггарти. Дискретная математика для программистов, 2003. — 384 с.

2. Файловый архив студентов — https://studfile.net/preview/2290655/page:13/

3. Ф. А. Новиков. Дискретная математика для программистов 3-е издание, 2008 — 384 с.

4. Джеймс Андерсон. Дискретная математика и комбинаторика, 2004 —959 с.

5. Акимов О.Е. Дискретная математика. Логика. Группы. Графы. 2001. — 352 с.

6. Марк Лутц. Изучаем Python, 4-е издание, 2011 — 1280 c.

# ПРИЛОЖЕНИЕ А

Код программы

Содержимое файла main.py

from mainInterface import Ui\_MainWindow

from graph import Graph

from BellmanFord import BellmanFord

from random import randint

import sys

from math import inf

from PyQt5 import QtWidgets

class Window(QtWidgets.QMainWindow):

def \_\_init\_\_(self, parent=None):

self.points = list()

self.arcs = list()

QtWidgets.QWidget.\_\_init\_\_(self, parent)

self.ui = Ui\_MainWindow()

self.ui.setupUi(self)

self.graph = Graph(self.points, self.arcs)

self.ui.verticalLayout\_2.addWidget(self.graph)

self.ui.pushButton.clicked.connect(self.addVertex)

self.ui.pushButton\_2.clicked.connect(self.addArc)

self.ui.pushButton\_3.clicked.connect(self.deleteVertex)

self.ui.pushButton\_4.clicked.connect(self.deleteArc)

self.ui.pushButton\_5.clicked.connect(self.calculate)

def updateListArcs(self, comboBox): # Обновление выпадающих списков дуг

comboBox.clear()

newArcs = list()

for arc in self.arcs:

newArcs.append(arc[:-1])

for firstVex in self.points:

for secondVex in self.points:

if firstVex != secondVex and [firstVex, secondVex] not in newArcs and [secondVex, firstVex] not in newArcs and type(firstVex) == int and type(secondVex) == int:

comboBox.addItem(str(firstVex) + "->" + str(secondVex))

def updateListVex(self, comboBox): # Обновление списка вершин

comboBox.clear()

comboBox.addItems([str(item) for item in self.points])

def addVertex(self):

if len(self.points) != 8:

# Добавление вершины в граф

for i in range(1, 9):

if i not in self.points:

self.points.append(i)

break

# Обновление выпадающих списков вершин

self.updateListVex(self.ui.comboBox\_4)

self.updateListVex(self.ui.comboBox\_5)

# Обновление выпадающих списков дуг

self.updateListArcs(self.ui.comboBox)

# Обновление графа

self.graph.update()

else:

self.ui.textBrowser.append("!Achieved by a high number of vertices!")

def deleteVertex(self):

if len(self.points):

# Удаление вершины из списка вершин

deleteVex = self.points[-1]

self.points.remove(deleteVex)

# Удаление дуг со смежной вершиной

delArcs = list()

for arc in self.arcs:

if deleteVex in arc:

delArcs.append(arc)

for arc in delArcs:

self.arcs.remove(arc)

# Обновление выпадающих списков вершин

self.updateListVex(self.ui.comboBox\_4)

self.updateListVex(self.ui.comboBox\_5)

# Обновление выпадающего списка дуг

self.updateListArcs(self.ui.comboBox)

self.ui.comboBox\_3.clear()

self.ui.comboBox\_3.addItems([str(item[0]) + "->" + str(item[1]) for item in self.arcs])

# Обновление графа

self.graph.update()

else:

self.ui.textBrowser.append("!No vertices!")

def addArc(self):

if self.ui.comboBox.currentText() and self.ui.lineEdit.text().strip("-").isdigit():

weight = self.ui.lineEdit.text()

# Очищение поля веса

self.ui.lineEdit.clear()

text = self.ui.comboBox.currentText().split("->")

# Добавление дуги

self.arcs.append([int(text[0]), int(text[1]), weight])

# Обновление выпадающих списков вершин

self.updateListVex(self.ui.comboBox\_4)

self.updateListVex(self.ui.comboBox\_5)

# Обновление выпадающих списков дуг

self.updateListArcs(self.ui.comboBox)

self.ui.comboBox\_3.clear()

self.ui.comboBox\_3.addItems([str(item[0]) + "->" + str(item[1]) for item in self.arcs])

# Обновление графа

self.graph.update()

else:

self.ui.textBrowser.append("!Unable to create an arc!")

def deleteArc(self):

if self.ui.comboBox\_3.currentText():

text = self.ui.comboBox\_3.currentText()

textList = text.split("->")

for arc in self.arcs:

if arc[0] == int(textList[0]) and arc[1] == int(textList[1]):

weight = arc[2]

break

# Удаление дуги

self.arcs.remove([int(textList[0]), int(textList[1]), weight])

# Обновление выпадающих списков дуг

self.updateListArcs(self.ui.comboBox)

self.ui.comboBox\_3.clear()

self.ui.comboBox\_3.addItems([str(item[0]) + "->" + str(item[1]) for item in self.arcs])

# Обновление графа

self.graph.update()

else:

self.ui.textBrowser.append("!No arcs!")

def calculate(self):

if self.ui.comboBox\_4.currentText() and self.ui.comboBox\_5.currentText():

from\_ = int(self.ui.comboBox\_4.currentText())

to\_ = int(self.ui.comboBox\_5.currentText())

count = len(self.points)

# Создание матрицы

matrix = list()

for i in range(0, count):

matrix.append(list())

for j in range(0, count):

matrix[i].append(inf)

# Обновление элементов матрицы

for arc in self.arcs:

matrix[arc[0]-1][arc[1]-1] = int(arc[2])

for i in range(0, count):

matrix[i][i] = 0

# Применение алгоритма

self.ui.textBrowser\_2.clear()

path = BellmanFord(from\_, to\_, matrix)

if path != inf:

self.ui.textBrowser\_2.append("Minimal path equals " + str(path))

else:

self.ui.textBrowser\_2.append("Missing path")

else:

self.ui.textBrowser.append("!Select vertices!")

if \_\_name\_\_ == '\_\_main\_\_':

app = QtWidgets.QApplication(sys.argv)

myapp = Window()

myapp.show()

sys.exit(app.exec\_())

*Содержимое файла graph.py*

from PyQt5.QtGui import QPainter, QPen, QFont

from PyQt5.QtCore import Qt

from PyQt5.QtWidgets import QWidget

from math import atan, sin, cos, pi, fabs

class Graph(QWidget):

def \_\_init\_\_(self, points, arcs, parent=None):

super().\_\_init\_\_()

self.points = points

self.arcs = arcs

self.show()

def paintEvent(self, event):

rec = event.rect()

self.vex = [[rec.width()/4, rec.height()/4], [3\*rec.width()/4, rec.height()/4], [rec.width()/4, 3\*rec.height()/4], [3\*rec.width()/4, 3\*rec.height()/4],\

[3\*rec.width()/16, 2\*rec.height()/4], [13\*rec.width()/16, 2\*rec.height()/4], [rec.width()/2, rec.height()/6], [rec.width()/2, 5\*rec.height()/6]]

qp = QPainter()

qp.begin(self)

self.drawFrame(qp, rec)

self.arrows(qp)

self.drawPoints(qp)

self.drawText(qp, rec)

qp.end()

def drawPoints(self, qp):

for point in self.points:

qp.setPen(QPen(Qt.red, 10))

qp.drawPoint(self.vex[point-1][0], self.vex[point-1][1])

def arrows(self, qp):

qp.setPen(QPen(Qt.blue, 5))

for arc in self.arcs:

x1 = self.vex[arc[0]-1][0]

x2 = self.vex[arc[1]-1][0]

y1 = self.vex[arc[0]-1][1]

y2 = self.vex[arc[1]-1][1]

# Отрисовка основной линии

qp.drawLine(x1, y1, x2, y2)

# Отрисовка указателя на конце линии

if x2 != x1:

k = (y2-y1)/(x2-x1)

alpha = atan(k) \* 180 / pi

else:

alpha = 90 \* (y2-y1)/abs(y2-y1)

delta1 = 14\*cos(30/180\*pi)

delta2 = 14\*sin(30/180\*pi)

if x1 != x2:

k1 = (x2-x1)/fabs(x2-x1)

else:

k1 = 1

if y1 != y2:

k2 = (y2-y1)/fabs(y2-y1)

else:

k2 = 1

point1 = self.turn(-1 \* k1 \* delta1, k2 \* delta2, alpha)

point2 = self.turn(-1 \* k1 \* delta1, -1 \* k2 \* delta2, alpha)

if x1 != x2:

if x1 > x2:

middleX = (x1-x2) / 2 + x2

else:

middleX = (x2-x1) / 2 + x1

else:

middleX = x1

if y1 != y2:

if y1 > y2:

middleY = (y1-y2) / 2 + y2

else:

middleY = (y2-y1) /2 + y1

else:

middleY = y1

qp.drawLine(middleX, middleY, middleX + point1[0], middleY + point1[1])

qp.drawLine(point2[0] + middleX, point2[1] + middleY, middleX, middleY)

def turn(self, x, y, alpha):

new\_x = x \* cos(alpha/180\*pi) - y \* sin(alpha/180\*pi)

new\_y = x \* sin(alpha/180\*pi) + y \* cos(alpha/180\*pi)

return new\_x, new\_y

def drawFrame(self, qp, rec):

qp.setPen(QPen(Qt.gray, 1))

qp.drawRect(9, 9, rec.width()-18, rec.height()-18)

def drawText(self, qp, rec):

middleScreenX = rec.width()/2

middleScreenY = rec.height()/2

# Нумерация точек

qp.setPen(QPen(Qt.black))

qp.setFont(QFont('Decorative', 15))

for point in self.points:

x = self.vex[point-1][0]

y = self.vex[point-1][1]

if x < middleScreenX:

znakX = -1

space = 1

else:

znakX = 1

space = 2

if y < middleScreenY:

znakY = -1

space = 1

else:

znakY = 1

space = 2

if x == middleScreenX:

deltaX = -5

else:

deltaX = znakX \* 12 \* space

if y == middleScreenY:

deltaY = 5

else:

deltaY = znakY \* 15 \* space

qp.drawText(x + deltaX, y + deltaY, str(point))

# Вес дуг графа

qp.setPen(QPen(Qt.darkMagenta))

qp.setFont(QFont('Decorative', 10))

for arc in self.arcs:

x1 = self.vex[arc[0]-1][0]

x2 = self.vex[arc[1]-1][0]

y1 = self.vex[arc[0]-1][1]

y2 = self.vex[arc[1]-1][1]

if x2 != x1:

k = (y2-y1)/(x2-x1)

alpha = atan(k) \* 180 / pi

else:

alpha = 90 \* (y2-y1)/abs(y2-y1)

if x1 != x2:

if x1 > x2:

middleX = (x1-x2) / 2 + x2

else:

middleX = (x2-x1) / 2 + x1

else:

middleX = x1

if y1 != y2:

if y1 > y2:

middleY = (y1-y2) / 2 + y2

else:

middleY = (y2-y1) /2 + y1

else:

middleY = y1

deltaX = 25\*cos(90/180\*pi)

deltaY = 25\*sin(90/180\*pi)

if x1 != x2:

k1 = (x2-x1)/fabs(x2-x1)

else:

k1 = 1

if y1 != y2:

k2 = (y2-y1)/fabs(y2-y1)

else:

k2 = 1

point = self.turn(-k1 \* deltaX, -k2 \* deltaY, alpha)

qp.drawText(middleX + point[0], middleY + point[1], str(arc[2]))

*Содержимое файла mainInterface.py*

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_MainWindow(object):

def setupUi(self, MainWindow):

MainWindow.setObjectName("MainWindow")

MainWindow.resize(800, 784)

MainWindow.setMinimumSize(QtCore.QSize(800, 784))

MainWindow.setMaximumSize(QtCore.QSize(800, 784))

self.centralwidget = QtWidgets.QWidget(MainWindow)

self.centralwidget.setObjectName("centralwidget")

self.pushButton = QtWidgets.QPushButton(self.centralwidget)

self.pushButton.setGeometry(QtCore.QRect(530, 550, 91, 41))

self.pushButton.setObjectName("pushButton")

self.pushButton\_2 = QtWidgets.QPushButton(self.centralwidget)

self.pushButton\_2.setGeometry(QtCore.QRect(690, 648, 93, 41))

self.pushButton\_2.setObjectName("pushButton\_2")

self.label = QtWidgets.QLabel(self.centralwidget)

self.label.setGeometry(QtCore.QRect(30, 630, 41, 31))

font = QtGui.QFont()

font.setPointSize(14)

self.label.setFont(font)

self.label.setLineWidth(1)

self.label.setObjectName("label")

self.pushButton\_3 = QtWidgets.QPushButton(self.centralwidget)

self.pushButton\_3.setGeometry(QtCore.QRect(640, 550, 91, 41))

self.pushButton\_3.setObjectName("pushButton\_3")

self.pushButton\_4 = QtWidgets.QPushButton(self.centralwidget)

self.pushButton\_4.setGeometry(QtCore.QRect(690, 708, 91, 41))

self.pushButton\_4.setObjectName("pushButton\_4")

self.comboBox = QtWidgets.QComboBox(self.centralwidget)

self.comboBox.setGeometry(QtCore.QRect(600, 658, 73, 22))

self.comboBox.setObjectName("comboBox")

self.comboBox\_3 = QtWidgets.QComboBox(self.centralwidget)

self.comboBox\_3.setEnabled(True)

self.comboBox\_3.setGeometry(QtCore.QRect(600, 718, 73, 22))

self.comboBox\_3.setDuplicatesEnabled(False)

self.comboBox\_3.setObjectName("comboBox\_3")

self.textBrowser = QtWidgets.QTextBrowser(self.centralwidget)

self.textBrowser.setGeometry(QtCore.QRect(20, 670, 481, 71))

self.textBrowser.setObjectName("textBrowser")

self.lineEdit = QtWidgets.QLineEdit(self.centralwidget)

self.lineEdit.setGeometry(QtCore.QRect(532, 658, 51, 22))

self.lineEdit.setObjectName("lineEdit")

self.label\_3 = QtWidgets.QLabel(self.centralwidget)

self.label\_3.setGeometry(QtCore.QRect(630, 638, 21, 16))

self.label\_3.setObjectName("label\_3")

self.label\_4 = QtWidgets.QLabel(self.centralwidget)

self.label\_4.setGeometry(QtCore.QRect(630, 698, 21, 16))

self.label\_4.setObjectName("label\_4")

self.label\_5 = QtWidgets.QLabel(self.centralwidget)

self.label\_5.setGeometry(QtCore.QRect(540, 638, 41, 16))

self.label\_5.setObjectName("label\_5")

self.verticalLayoutWidget = QtWidgets.QWidget(self.centralwidget)

self.verticalLayoutWidget.setGeometry(QtCore.QRect(0, 0, 800, 510))

self.verticalLayoutWidget.setObjectName("verticalLayoutWidget")

self.verticalLayout\_2 = QtWidgets.QVBoxLayout(self.verticalLayoutWidget)

self.verticalLayout\_2.setContentsMargins(0, 0, 0, 0)

self.verticalLayout\_2.setObjectName("verticalLayout\_2")

self.comboBox\_4 = QtWidgets.QComboBox(self.centralwidget)

self.comboBox\_4.setGeometry(QtCore.QRect(160, 615, 73, 22))

self.comboBox\_4.setObjectName("comboBox\_4")

self.comboBox\_5 = QtWidgets.QComboBox(self.centralwidget)

self.comboBox\_5.setGeometry(QtCore.QRect(240, 615, 73, 22))

self.comboBox\_5.setObjectName("comboBox\_5")

self.pushButton\_5 = QtWidgets.QPushButton(self.centralwidget)

self.pushButton\_5.setGeometry(QtCore.QRect(320, 605, 91, 41))

self.pushButton\_5.setObjectName("pushButton\_5")

self.label\_6 = QtWidgets.QLabel(self.centralwidget)

self.label\_6.setGeometry(QtCore.QRect(182, 595, 31, 16))

self.label\_6.setObjectName("label\_6")

self.label\_7 = QtWidgets.QLabel(self.centralwidget)

self.label\_7.setGeometry(QtCore.QRect(270, 595, 16, 16))

self.label\_7.setObjectName("label\_7")

self.textBrowser\_2 = QtWidgets.QTextBrowser(self.centralwidget)

self.textBrowser\_2.setGeometry(QtCore.QRect(160, 555, 256, 31))

self.textBrowser\_2.setObjectName("textBrowser\_2")

self.label\_8 = QtWidgets.QLabel(self.centralwidget)

self.label\_8.setGeometry(QtCore.QRect(260, 525, 61, 31))

font = QtGui.QFont()

font.setPointSize(14)

self.label\_8.setFont(font)

self.label\_8.setLineWidth(1)

self.label\_8.setObjectName("label\_8")

self.line\_2 = QtWidgets.QFrame(self.centralwidget)

self.line\_2.setGeometry(QtCore.QRect(130, 514, 20, 141))

self.line\_2.setFrameShape(QtWidgets.QFrame.VLine)

self.line\_2.setFrameShadow(QtWidgets.QFrame.Sunken)

self.line\_2.setObjectName("line\_2")

self.line\_3 = QtWidgets.QFrame(self.centralwidget)

self.line\_3.setGeometry(QtCore.QRect(140, 505, 300, 16))

self.line\_3.setFrameShape(QtWidgets.QFrame.HLine)

self.line\_3.setFrameShadow(QtWidgets.QFrame.Sunken)

self.line\_3.setObjectName("line\_3")

self.line\_4 = QtWidgets.QFrame(self.centralwidget)

self.line\_4.setGeometry(QtCore.QRect(430, 514, 20, 141))

self.line\_4.setFrameShape(QtWidgets.QFrame.VLine)

self.line\_4.setFrameShadow(QtWidgets.QFrame.Sunken)

self.line\_4.setObjectName("line\_4")

self.line\_5 = QtWidgets.QFrame(self.centralwidget)

self.line\_5.setGeometry(QtCore.QRect(140, 650, 300, 16))

self.line\_5.setFrameShape(QtWidgets.QFrame.HLine)

self.line\_5.setFrameShadow(QtWidgets.QFrame.Sunken)

self.line\_5.setObjectName("line\_5")

self.line = QtWidgets.QFrame(self.centralwidget)

self.line.setGeometry(QtCore.QRect(520, 530, 221, 20))

self.line.setFrameShape(QtWidgets.QFrame.HLine)

self.line.setFrameShadow(QtWidgets.QFrame.Sunken)

self.line.setObjectName("line")

self.line\_6 = QtWidgets.QFrame(self.centralwidget)

self.line\_6.setGeometry(QtCore.QRect(520, 590, 221, 20))

self.line\_6.setFrameShape(QtWidgets.QFrame.HLine)

self.line\_6.setFrameShadow(QtWidgets.QFrame.Sunken)

self.line\_6.setObjectName("line\_6")

self.line\_7 = QtWidgets.QFrame(self.centralwidget)

self.line\_7.setGeometry(QtCore.QRect(520, 540, 3, 61))

self.line\_7.setFrameShape(QtWidgets.QFrame.VLine)

self.line\_7.setFrameShadow(QtWidgets.QFrame.Sunken)

self.line\_7.setObjectName("line\_7")

self.line\_8 = QtWidgets.QFrame(self.centralwidget)

self.line\_8.setGeometry(QtCore.QRect(740, 540, 3, 61))

self.line\_8.setFrameShape(QtWidgets.QFrame.VLine)

self.line\_8.setFrameShadow(QtWidgets.QFrame.Sunken)

self.line\_8.setObjectName("line\_8")

self.line\_9 = QtWidgets.QFrame(self.centralwidget)

self.line\_9.setGeometry(QtCore.QRect(510, 638, 20, 121))

self.line\_9.setFrameShape(QtWidgets.QFrame.VLine)

self.line\_9.setFrameShadow(QtWidgets.QFrame.Sunken)

self.line\_9.setObjectName("line\_9")

self.line\_10 = QtWidgets.QFrame(self.centralwidget)

self.line\_10.setGeometry(QtCore.QRect(780, 638, 20, 121))

self.line\_10.setFrameShape(QtWidgets.QFrame.VLine)

self.line\_10.setFrameShadow(QtWidgets.QFrame.Sunken)

self.line\_10.setObjectName("line\_10")

self.line\_11 = QtWidgets.QFrame(self.centralwidget)

self.line\_11.setGeometry(QtCore.QRect(520, 630, 270, 16))

self.line\_11.setFrameShape(QtWidgets.QFrame.HLine)

self.line\_11.setFrameShadow(QtWidgets.QFrame.Sunken)

self.line\_11.setObjectName("line\_11")

self.line\_12 = QtWidgets.QFrame(self.centralwidget)

self.line\_12.setGeometry(QtCore.QRect(520, 750, 271, 16))

self.line\_12.setFrameShape(QtWidgets.QFrame.HLine)

self.line\_12.setFrameShadow(QtWidgets.QFrame.Sunken)

self.line\_12.setObjectName("line\_12")

MainWindow.setCentralWidget(self.centralwidget)

self.retranslateUi(MainWindow)

QtCore.QMetaObject.connectSlotsByName(MainWindow)

def retranslateUi(self, MainWindow):

\_translate = QtCore.QCoreApplication.translate

MainWindow.setWindowTitle(\_translate("MainWindow", "Bellman-Ford"))

self.pushButton.setText(\_translate("MainWindow", "Add vertex"))

self.pushButton\_2.setText(\_translate("MainWindow", "Add arc"))

self.label.setText(\_translate("MainWindow", "info"))

self.pushButton\_3.setText(\_translate("MainWindow", "Delete vertex"))

self.pushButton\_4.setText(\_translate("MainWindow", "Delete arc"))

self.label\_3.setText(\_translate("MainWindow", "arc"))

self.label\_4.setText(\_translate("MainWindow", "arc"))

self.label\_5.setText(\_translate("MainWindow", "weight"))

self.pushButton\_5.setText(\_translate("MainWindow", "Calculate"))

self.label\_6.setText(\_translate("MainWindow", "from"))

self.label\_7.setText(\_translate("MainWindow", "to"))

self.label\_8.setText(\_translate("MainWindow", "result"))

Содержимое файла BellmanFord.py:

from math import inf

def BellmanFord(from\_, to\_, matrix):

vertices = list() # список длин путей до вершины

labels = list()

verticesCount = len(matrix) # количество вершин

for i in range(verticesCount):

vertices.append(inf)

vertices[from\_-1] = 0;

for k in range(1, verticesCount):

for i in range(verticesCount):

for j in range(verticesCount):

labels.append(vertices[j] + matrix[j][i])

vertices[i] = minOfArray(labels)

labels.clear()

return vertices[to\_-1]

def minOfArray(lst):

min = inf;

for i in lst:

if i < min:

min = i;

return min;

# ПРИЛОЖЕНИЕ Б

Функционирование программы

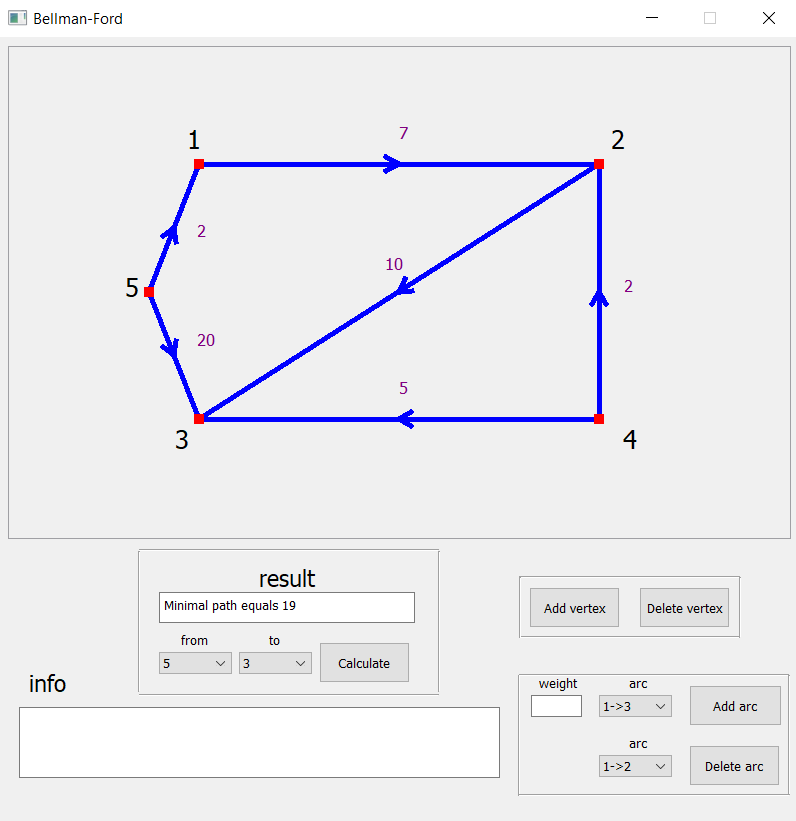


Рисунок Б.1 – Расчет кратчайшего пути от вершины 5 до вершины 3.

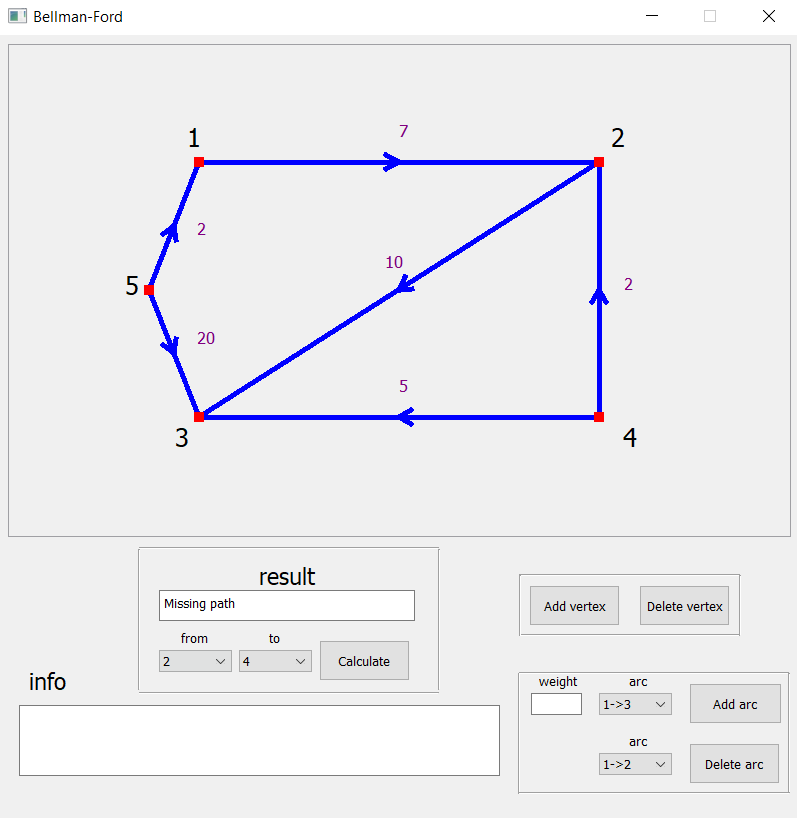


Рисунок Б.2 – Расчет несуществующего пути от вершины 2 до вершины 4.

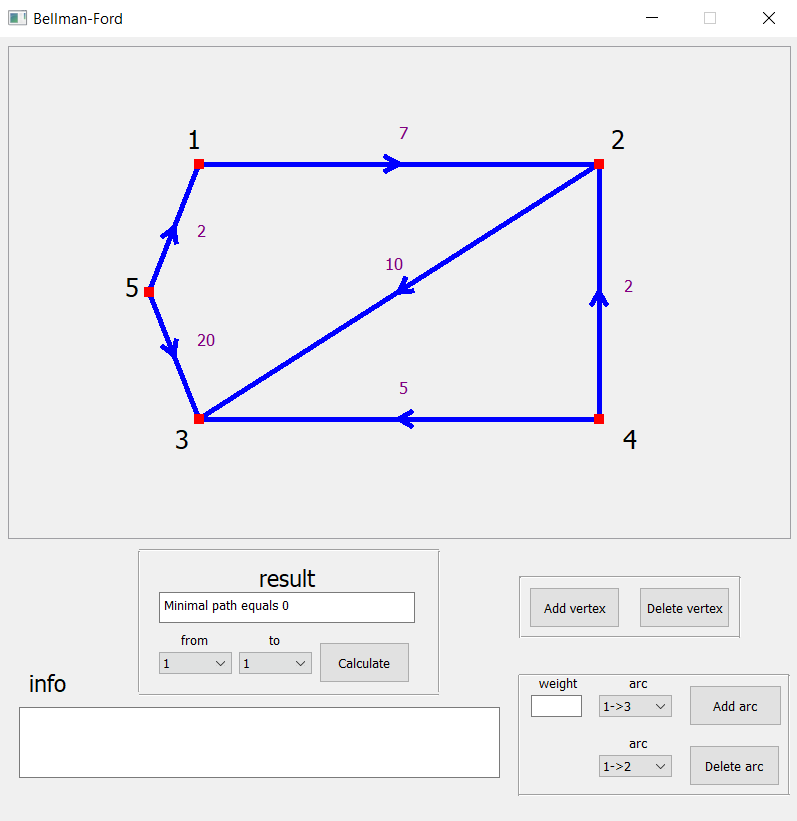


Рисунок Б.3 – Расчет кратчайшего пути от вершины 1 до вершины 1.