

Содержание

ВВЕДЕНИЕ . . . . . 3

1. ПОИСК НЕСКОЛЬКИХ КРАТЧАЙШИХ ПУТЕЙ В ГРАФЕ . . . . . 4

1.1. Содержательное описание задачи . . . . . 4

1.2. Формальная постановка задачи . . . . . 4

2. РАЗРАБОТКА АЛГОРИТМА . . . . . 7

2.1. Разработка графического интерфейса пользователя . . . . . 7

2.2. Разработка структур данных . . . . . 8

2.3. Разработка структуры алгоритма . . . . . 9

2.4. Схема алгоритма . . . . . 9

3. РАЗРАБОТКА ПРОГРАММЫ . . . . . 12

3.1. Описание переменных и структур данных . . . . . 12

3.2. Описание функций . . . . . 12

4. ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЮ . . . . . 13

5. ТЕСТОВАЯ ЗАДАЧА . . . . . 14

5.1. Аналитическое решение и умозрительные результаты . . . . . 14

5.2. Решение, полученное с использованием разработанного ПО . . . . . 15

5.3. Выводы . . . . . 16

ЗАКЛЮЧЕНИЕ . . . . . 16

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ . . . . . 17

ПРИЛОЖЕНИЕ . . . . . 17

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл.	

					Вариант №3				
Изм	Лист	№ докум.	Подп.	Дата					
Разраб.		Белым А.А.			Пояснительная записка к лабораторной работе по курсу «Вычислительный практикум» по теме «Алгоритм Йена»		Лит.	Лист	Листов
Пров.		Ермаков А.С.						2	23
Н. контр.							ТулГУ гр. 220601		
Утв.									

## ВВЕДЕНИЕ

Многие структуры, представляющие практический интерес в математике и информатике, могут быть представлены графами.

Графы часто используются для нахождения кратчайшего пути из одной точки в другую при отсутствии прямой связи между точками, учитывая возможность прохождения маршрута через какие-либо другие точки.

В данной работе разбирается алгоритм поиска некоторого числа кратчайших путей в графе - алгоритм Йена и алгоритм поиска кратчайшего пути на графе без рёбер отрицательного веса - алгоритм Дейкстры, а также разработана программа, которая реализует этот алгоритм и визуализирует исходный граф. Отчёт содержит полный текст программы на языке Python, описание всех функций, инструкцию пользователю и тестовый пример.

Инов. № подл.	Подп. и дата	Взам. инв. №	Инов. № дубл.	Подп. и дата	Вариант №3					Лист
										3
Изм	Лист	№ докум.	Подп.	Дата						

# 1. ПОИСК НЕСКОЛЬКИХ КРАТЧАЙШИХ ПУТЕЙ В ГРАФЕ

## 1.1. Содержательное описание задачи

Дан граф без петель и без рёбер отрицательного веса. Требуется найти  $K$  кратчайших маршрутов без циклов из одной некоторой точки в другую.

Например, при  $K=1$  ищется самый кратчайший путь. При  $K=2$  ищется кратчайший путь, а также путь, отличающийся от кратчайшего на минимальное расстояние. Аналогично при  $K=3$  и т.д.

Если запрошенное количество невозможно отыскать, вывести максимальное количество маршрутов (без циклов).

## 1.2. Формальная постановка задачи

Пользователь задает граф и количество кратчайших путей, которое надо найти. Требуется с помощью алгоритмов Йена и Дейкстры найти указанное количество кратчайших путей, либо сколько есть, если заданное количество слишком большое.

В математической теории графов и информатике граф — это совокупность непустого множества вершин и множества пар вершин (связей между вершинами).

Объекты представляются как вершины, или узлы графа, а связи — как дуги, или рёбра. Для разных областей применения виды графов могут различаться направленностью, ограничениями на количество связей и дополнительными данными о вершинах или рёбрах.

Для представления связей между вершинами при программировании могут использоваться различные способы.

Первый способ задания графа (невзвешенного) это задать матрицу связности  $S$  размера  $n \times n$ , где  $n$  количество вершин графа, т.е. мощность множества  $V$ , при этом элемент  $s_{ij}=1$ , если существует ребро из  $i$ -ой в  $j$ -ую вершины и  $s_{ij}=0$ , если такого ребра нет. Нетрудно видеть, что матрица  $S$ - симметрична, если граф неориентиро-

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл.	

Изм	Лист	№ докум.	Подп.	Дата	Вариант №3	Лист
						4

ванный, и может быть не симметричный в противном случае. При этом полагаем, что  $s_i = 0$ , т.е. в графе нет петель.

Второй способ используется для задания взвешенного графа, т.е. графа каждому ребру которого соответствует некий параметр - вес. Для определения такого графа используется матрица весов  $W$  размер которой  $n \times n$ , где  $n$  количество вершин графа. При этом элемент  $w_{ij}$  равен весу ребра соединяющего  $i$ -ую и  $j$ -ую вершины. Если такого ребра нет, то  $w_{ij}$  полагаем равным бесконечности (на практике это максимальное число возможное на данном языке программирования). Этот способ задания используется например в алгоритмах поиска пути во взвешенном графе.

Алгоритм Йена предназначен для нахождения  $K$  путей минимальной длины во взвешенном графе соединяющих вершины  $u_1, u_2$ . Ищутся пути, которые не содержат петель.

Задача состоит в отыскании нескольких минимальных путей, поэтому возникает вопрос о том чтобы не получить путь содержащий петлю, в случае поиска одного пути минимального веса, это условие выполняется по необходимости, в данном же случае мы используем алгоритм Йена, позволяющий находить  $K$  кратчайших простых цепей.

Работа алгоритма начинается с нахождения кратчайшего пути, для этого будем использовать следующий алгоритм.(алгоритм Дейкстры).

1. всем вершинам приписывается вес - вещественное число,  $d(i) = \infty$  для всех вершин кроме вершины с номером  $u_1$ , а  $d(u_1) = 0$ ; в качестве предыдущей вершины для всех вершин ставим начальную.

2. всем вершинам приписывается метка  $m(i) = 0$ ;

3. вершина  $u_1$  объявляется текущей -  $t = u_1$

4. для всех вершин  $u$  которых  $m(i) = 0$ , пересчитываем вес по формуле:

$d(i) := \min(d(i), d(t) + W[t, i])$ . Если вес через вершину  $t$  меньше, то запоминаем её как предыдущую для текущей вершины.

5. среди вершин для которых выполнено  $m(i) = 0$  ищем ту для которой  $d(i)$  минимальна, если минимум не найден, т.е. вес всех не "помеченных" вершин равен

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	случае мы используем алгоритм Йена, позволяющий находить К кратчайших простых цепей.	
					Работа алгоритма начинается с нахождения кратчайшего пути, для этого будем использовать следующий алгоритм.(алгоритм Дейкстры).	
					1. всем вершинам приписывается вес - вещественное число, $d(i)=inf$ для всех вершин кроме вершины с номером $u1$ , а $d(u1)=0$ ; в качестве предыдущей вершины для всех вершин ставим начальную.	
					2. всем вершинам приписывается метка $m(i)=0$ ;	
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	3. вершина $u1$ объявляется текущей - $t=u1$	
					4. для всех вершин $u$ которых $m(i)=0$ , пересчитываем вес по формуле: $d(i):=\min(d(i), d(t)+W[t,i])$ . Если вес через вершину $t$ меньше, то запоминаем её как предыдущую для текущей вершины.	
					5. среди вершин для которых выполнено $m(i)=0$ ищем $tu$ для которой $d(i)$ минимальна, если минимум не найден, т.е. вес всех не "помеченных" вершин равен	
Изм	Лист	№ докум.	Подп.	Дата	Вариант №3	Лист
						5



нения шагов с 3 по 6, в результирующий список. При расчете пути можно искать его вес только от вершины  $i$  до конечной вершины, и сложить его с весом первой части пути от начала до  $i$ , полученным на шаге 4. Если  $k = K$ , то алгоритм заканчивает работу, иначе увеличить  $k$  на единицу и вернуться к шагу 2. Если  $MinW$  равен бесконечности, то больше путей невозможно найти, выход.

## 2. РАЗРАБОТКА АЛГОРИТМА

### 2.1. Разработка графического интерфейса пользователя

Для задания матрицы смежности используется квадратная таблица. Пользователь задает размер матрицы, и может создать новую пустую матрицу, либо изменяет размеры с сохранением введенных данных с помощью соотв. кнопок. Так как матрица неориентированного графа симметрична, пользователь имеет возможность отразить правый верхний треугольник матрицы с помощью соотв. кнопки.

Пользователь также вводит начальную и конечную вершины, и количество путей для поиска.

Кнопка "Поиск"запускает алгоритм Йена для поиска кратчайших путей, которые выводятся в таблицу.

Пользователь может визуализировать граф с помощью специальной кнопки, а после нахождения путей выбрать в таблице какой-нибудь из них и также показать его на графе.

Создать панель меню со следующими разделами:

- 1) Файл. Содержит разделы: "Выход";
- 2) Правка. Содержит раздел "Создать пустую матрицу" , "Изменить размер матрицы" , "Сделать матрицу симметричной".
- 3) Запуск. Содержит раздел "Поиск путей".
- 4) Вид. Содержит "Показать граф" , "Показать путь".
- 5) Справка. Содержит раздел "Управление graphviz-x11" ,"Справка".

Инов. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	Пользователь также вводит начальную и конечную вершины, и количество путей для поиска.
					Кнопка "Поиск"запускает алгоритм Йена для поиска кратчайших путей, которые выводятся в таблицу.
					Пользователь может визуализировать граф с помощью специальной кнопки, а после нахождения путей выбрать в таблице какой-нибудь из них и также показать его на графе.
					Создать панель меню со следующими разделами:
					1) Файл. Содержит разделы: "Выход";
Инов. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	2) Правка. Содержит раздел "Создать пустую матрицу" , "Изменить размер матрицы" , "Сделать матрицу симметричной".
					3) Запуск. Содержит раздел "Поиск путей".
					4) Вид. Содержит "Показать граф" , "Показать путь".
					5) Справка. Содержит раздел "Управление graphviz-x11" ,"Справка".

Итак, внешний вид разработанного интерфейса представлен на рисунке 1.

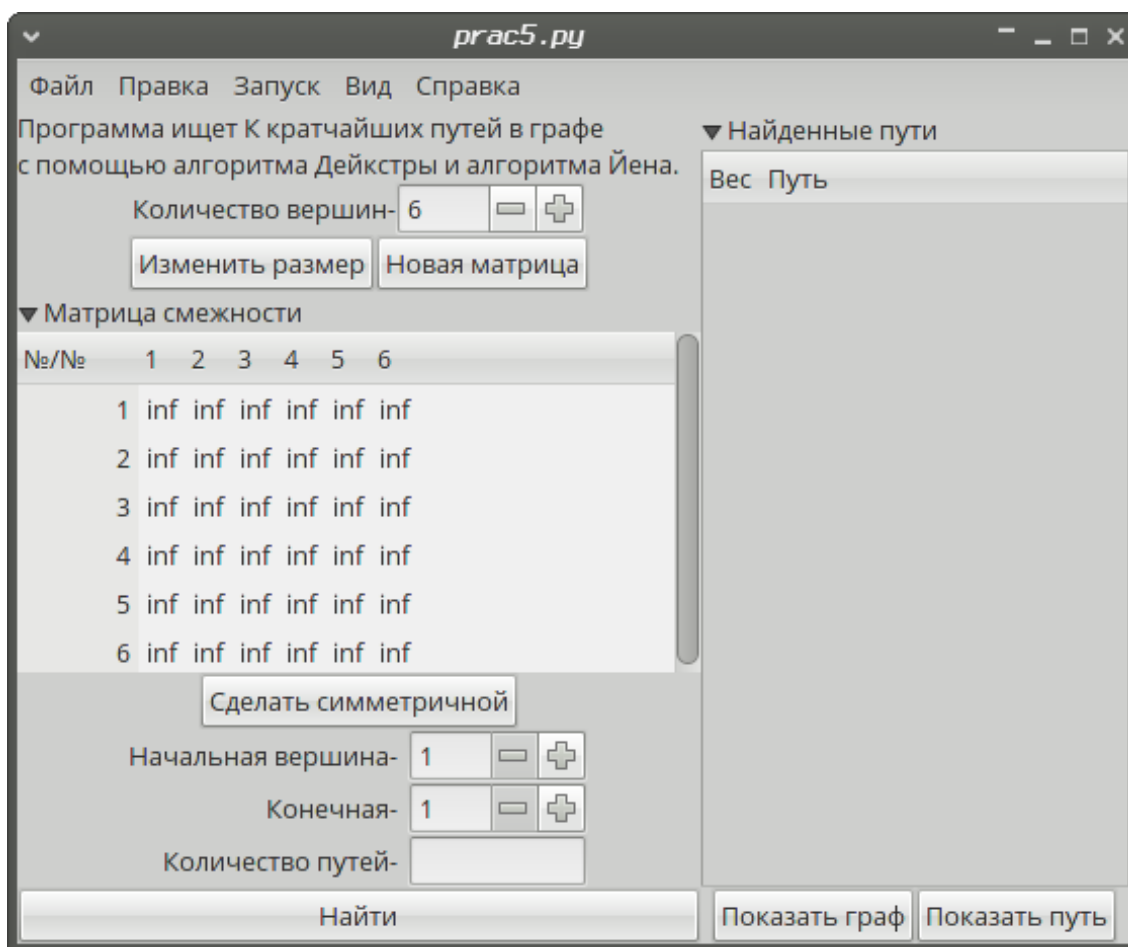


Рисунок 1 – Разработанный интерфейс программы

## 2.2. Разработка структур данных

В качестве данных будут использоваться следующие переменные:

matrix - матрица смежности заданного графа,

src,dst - начальная и конечная вершины путей,

k - количество путей для поиска.

Инов. № подл.	Подп. и дата	Взам. инв. №	Инов. № дубл.	Подп. и дата	Вариант №3					Лист
										8
Изм	Лист	№ докум.	Подп.	Дата						

### 2.3. Разработка структуры алгоритма

Основную программу можно разбить на три участка: считывание значений , нахождения путей и вывод полученных результатов.

- 1) Для нахождения нескольких кратчайших путей с помощью алгоритма Йе-на будет использоваться функция uen, принимающая в качестве параметров указанные в предыдущем разделе переменные, и возвращающая результат в виде списка найденных путей. Кроме того, для нахождения одного пути используется алгоритм Дейкстры, реализованной функцией dijkstra.
- 2) Подпрограмма ввода данных input\_data.
- 3) Подпрограмма вывода данных output\_data.

### 2.4. Схема алгоритма

На рисунке 2 представлена схема алгоритма Дейкстры - поиска кратчайшего пути в графе без ребер с отрицательным весом.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	пути в графе без ребер с отрицательным весом.					
					Вариант №3					Лист
										9
Изм	Лист	№ докум.	Подп.	Дата						



Инов. № подл.	Подп. и дата	Взам. инв. №	Инов. № дубл.	Подп. и дата

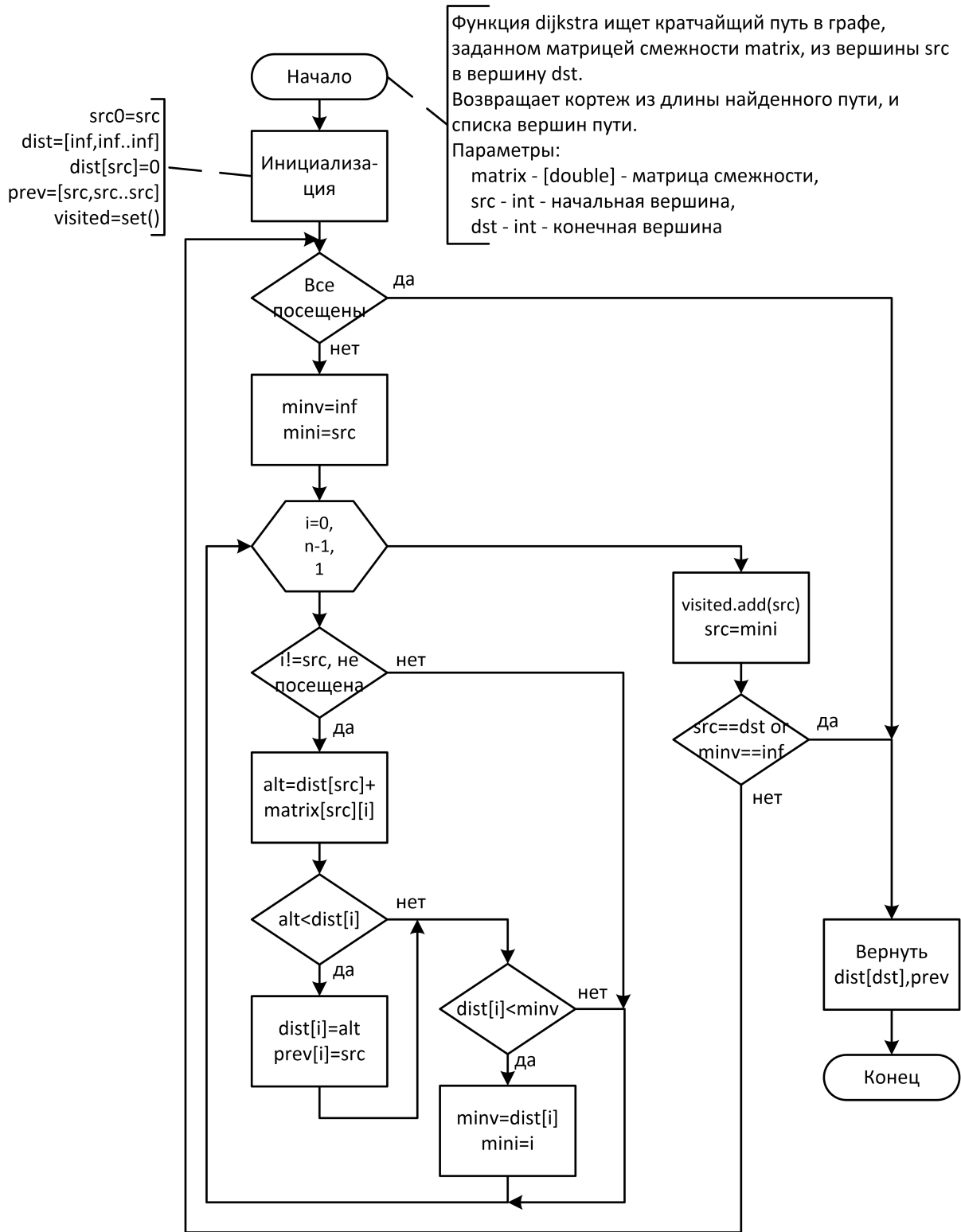


Рисунок 2 – Схема алгоритма Дейкстры

На рисунке 3 представлена схема алгоритма Йена - поиска К путей в графе без ребер с отрицательным весом.

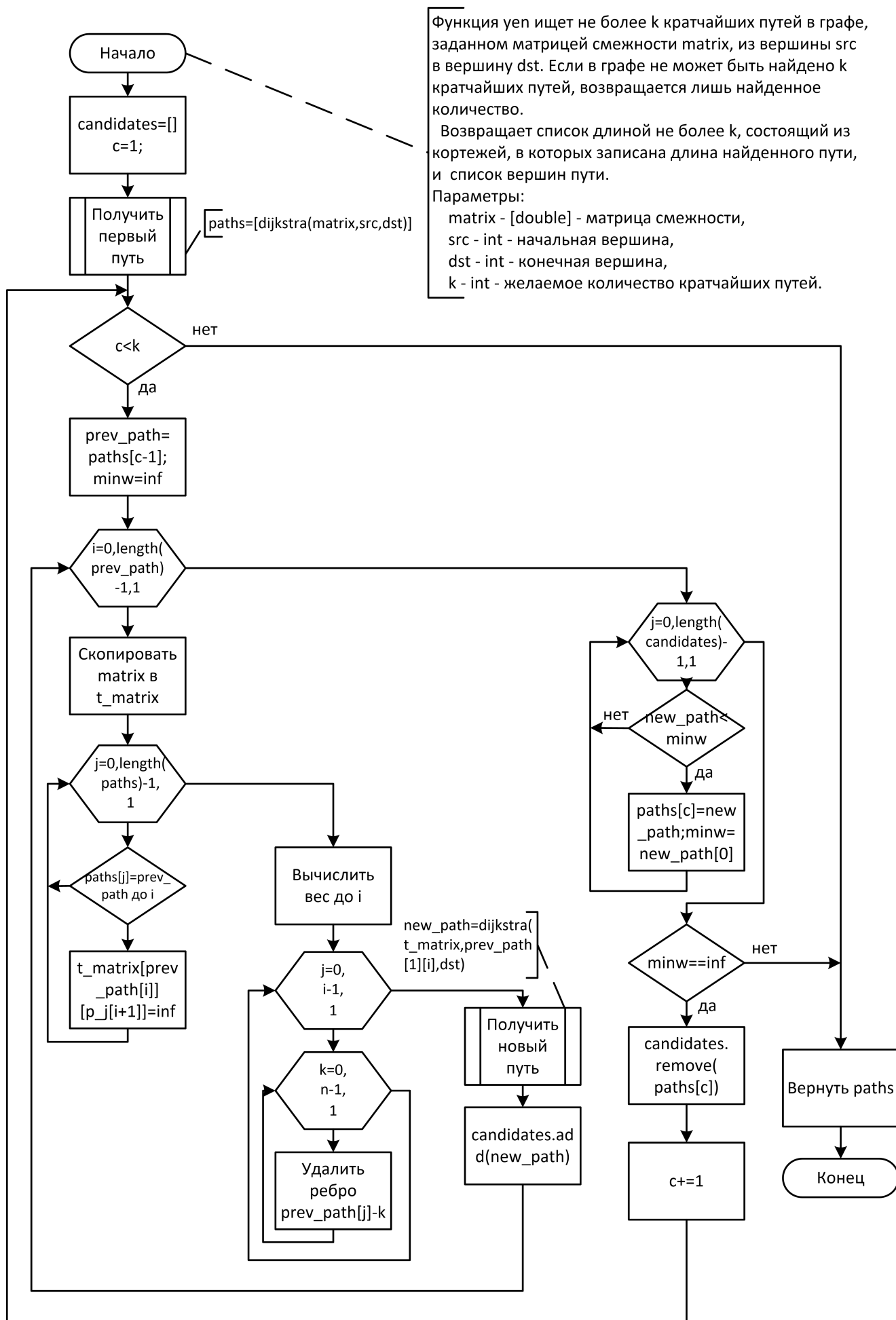


Рисунок 3 – Схема алгоритма алгоритма Йена

Вариант №3

Лист

11

### 3. РАЗРАБОТКА ПРОГРАММЫ

### 3.1. Описание переменных и структур данных

В качестве данных будут использоваться следующие переменные:

matrix - `[[float]]` - матрица смежности заданного графа,

src,dst - int - начальная и конечная вершины путей,

k - int - количество путей для поиска.

### 3.2. Описание функций

1. `yen(matrix,src,dst,k)`

Функция `уеп` ищет не более  $k$  кратчайших путей в графе, заданном матрицей смежности `matrix`, из вершины `src` в вершину `dst`. Если в графе не может быть найдено  $k$  кратчайших путей, возвращается лишь найденное количество.

Возвращает список длиной не более  $k$ , состоящий из кортежей, в которых записана длина найденного пути, и список вершин пути. Параметры функции представлены в таблице 1 :

Таблица 1 – Параметры функции поиска нескольких путей в графе

имя	тип	предназначение
matrix	[double]	матрица смежности,
src	int	начальная вершина,
dst	int	конечная вершина,
k	int	желаемое количество кратчайших путей.

## 2. dijkstra(matrix,src,dst)

Функция `dijkstra` ищет кратчайший путь в графе, заданном матрицей смежности `matrix`, из вершины `src` в вершину `dst`.

Возвращает кортеж из длины найденного пути, и списка вершин пути. Параметры функции представлены в таблице 2 :

Таблица 2 – Параметры функции поиска одного пути в графе

имя	тип	предназначение
matrix	[double]	матрица смежности,
src	int	начальная вершина,
dst	int	конечная вершина

3. input\_data(self)

Подпрограмма ввода исходных данных.

4. output\_data(self,paths)

Подпрограмма вывода результатов.

5. on\_run\_click(self,button,data=None)

Подпрограмма считывания данных, поиска путей и вывода результатов.

6. on\_show\_graph(self,button,data=None):

Подпрограмма построения графа.

7. on\_show\_path(self,button,data=None):

Подпрограмма построения графа и пути.

## 4. ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЮ

Данная программа находит несколько кратчайших путей в графе с помощью алгоритмов Дейкстры и Йена.

Программа не требует установки. Для её запуска необходимо открыть файл prас4.py. Внимание: для работы приложения на компьютере должен быть установлен Python 3, GTK+3, GObject-introspection, Gnuplot и graphviz.

Инов. № подл.	Подп. и дата	Взам. инв. №	Инов. № дубл.	Подп. и дата	<div>Вариант №3</div>					Лист
										13
Изм	Лист	№ докум.	Подп.	Дата						

Для работы необходимо задать граф с помощью матрицы смежности. Сначала установите размер матрицы. Вы можете создать новую пустую матрицу указанного размера, либо изменить размер с сохранением уже введенных данных. После этого можно приступать к заполнению матрицы смежности с помощью таблицы. Если исходный граф не является ориентированным, можно заполнить только верхний правый треугольник матрицы, и нажать кнопку "Сделать симметричной так как матрица смежности неориентированного графа является симметричной.

После заполнения матрицы смежности укажите начальную и конечную вершину путей, а также их количество. Можно также визуализировать граф с помощью graphviz, нажав специальную кнопку. После этого нажмите кнопку "Поиск".

Найденные пути выводятся в специальную таблицу. Если невозможно найти указанное количество путей, будет выведено найденное количество.

Вы можете выбрать интересующий путь в таблице и визуализировать его на графе, нажав кнопку "Показать путь".

## 5. ТЕСТОВАЯ ЗАДАЧА

### 5.1. Аналитическое решение и умозрительные результаты

Рассмотрим граф на рисунке 4 .

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>5. ТЕСТОВАЯ ЗАДАЧА</div> <div>5.1. Аналитическое решение и умозрительные результаты</div> <div>Рассмотрим граф на рисунке 4 .</div>					
Изм	Лист	№ докум.	Подп.	Дата	<div>Вариант №3</div>					Лист
										14

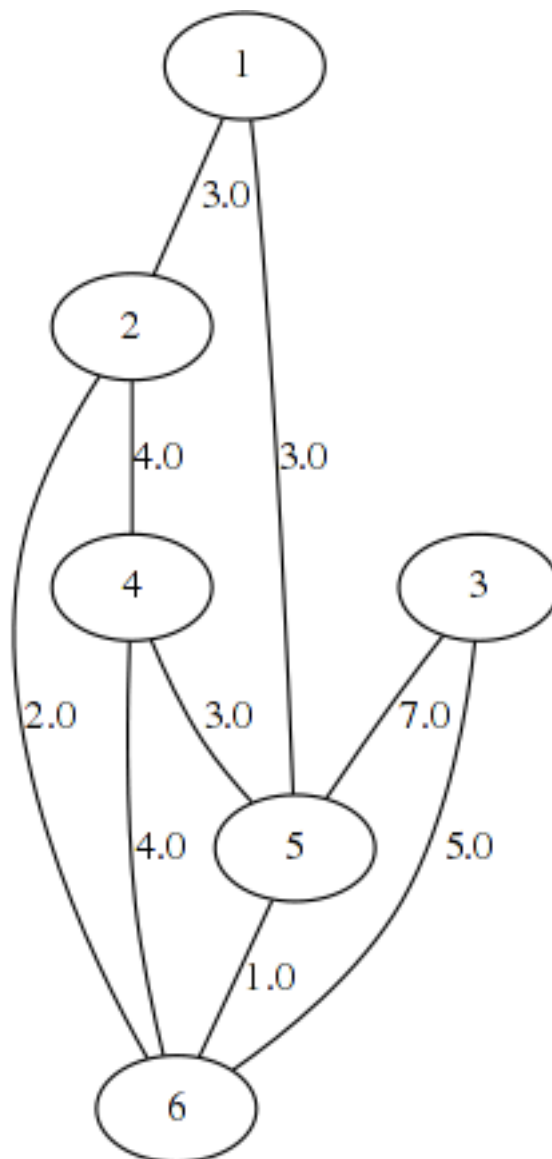


Рисунок 4 – Исходный граф

Найдем 5 путей из вершины 1 в вершину 6.

## 5.2. Решение, полученное с использованием разработанного ПО

Ниже на рисунке 5 представлен пример работы программы поиска К кратчайших путей в графе.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата




Рисунок 4 – Исходный граф

Найдем 5 путей из вершины 1 в вершину 6.

**5.2. Решение, полученное с использованием разработанного ПО**

Ниже на рисунке 5 представлен пример работы программы поиска К кратчайших путей в графе.

					Вариант №3	Лист
						15
Изм	Лист	№ докум.	Подп.	Дата		

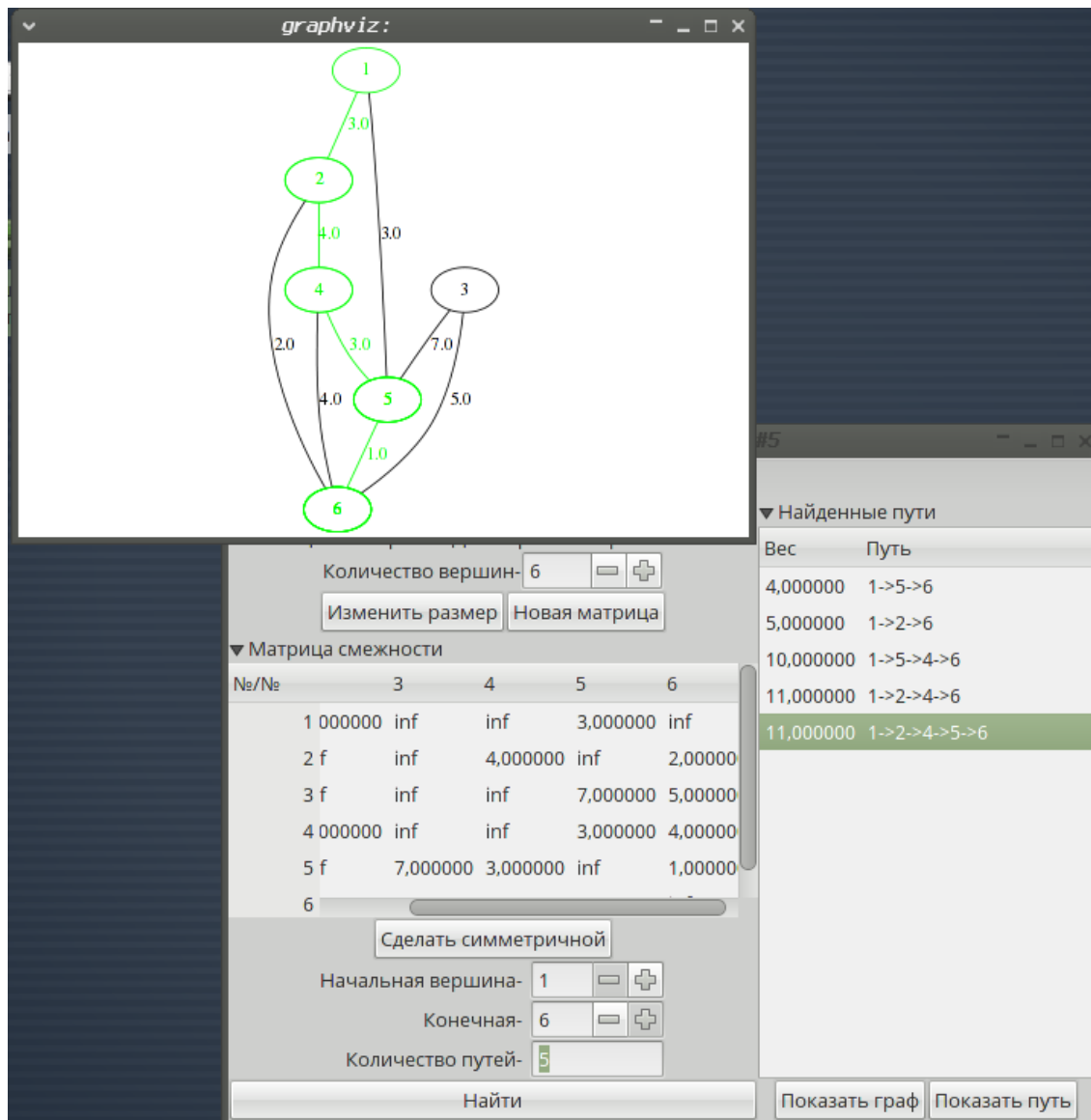


Рисунок 5 – Пример работы программы поиска кратчайших путей

### 5.3. Выводы

С помощью алгоритмов Дейкстры и Йена можно найти несколько кратчайших путей в графе. Программа graphviz позволяет визуализировать граф найденные пути.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Алгоритм Йена позволяет найти несколько кратчайших путей на графе. Для этого необходим способ поиска одного кратчайшего пути в графе, например, алгоритм Дейкстры, который ищет пути в графе без рёбер отрицательного веса.

1. <http://python.org>
2. <http://www.gtk.org>
3. <http://ru.wikipedia.org>
4. <http://en.wikipedia.org>

Далее приводится текст программы поиска нескольких кратчайших путей в графе, написанной на Python 3.

```
def dijkstra(matrix,src,dst):
    '''
    Функция dijkstra ищет кратчайший путь в графе,
    заданном матрицей смежности matrix,
```



из вершины *src* в вершину *dst*.  
Возвращает кортеж из длины найденного пути, и  
списка вершин пути.

Параметры:

*matrix* – [double] – матрица смежности,  
*src* – int – начальная вершина,  
*dst* – int – конечная вершина  
'''

```
src0=src
dist=[float("inf") for i in matrix]
dist[src]=0
prev=[src for i in matrix]
visited=set()
if src==dst:
    return (0,[src])
while len(visited)!=len(matrix):
    minv=float("inf");mini=src
    for i in range(0,len(matrix[src])):
        if i not in visited and i!=src:
            alt=dist[src]+matrix[src][i]
            if alt<dist[i]:
                dist[i]=alt
                prev[i]=src
            if dist[i]<minv:
                minv=dist[i]
                mini=i
    visited.add(src)
    src=mini
    if src==dst or minv==inf:
        break
#print (prev,src0,dst)
path=get_path(prev,src0,dst)
return dist[dst],path
```

```
def get_path(prev,src,dst):
    #print (src,dst)
    path=[dst]
    while dst!=src and dst!=prev[dst]:
        #print (prev[dst])
        path.append(prev[dst])
        dst=prev[dst]
    path.reverse()
    return path
```

```
def check_matrix(m1,m2):
    for i in range(0,len(m1)):
        for j in range(0,len(m1[i])):
            if m1[i][j]!=m2[i][j]:
                break
        else:
            continue
        break
    else:
        return True
    return False
def equal_paths(p1,p2,i):
    if i>=len(p2):
        print(0)
```

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	Вариант №3					Лист
										18
Изм.	Лист	№ докум.	Подп.	Дата						

```

        return False;
    for n in range(0,i+1):
        if p1[i]!=p2[i]:
            break
    else:
        return True
    return False

def calc_weigth(matrix,p):
    w=0
    for i in range(0,len(p)-1):
        w+=matrix[p[i]][p[i+1]]
    return w

def yen(matrix,src,dst,k):
    '''
    Функция yen ищет не более k кратчайших путей в графе,
    заданном матрицей смежности matrix,
    из вершины src в вершину dst. Если в графе не может быть найдено k
    кратчайших путей, возвращается лишь найденное количество.
    Возвращает список длиной не более k, состоящий из кортежей,
    в которых записана длина найденного пути, и
    список вершин пути.
    Параметры:
    matrix – [double] – матрица смежности,
    src – int – начальная вершина,
    dst – int – конечная вершина,
    k – int – желаемое количество кратчайших путей.
    '''
    paths=[dijkstra(matrix,src,dst)]
    candidates=[]
    c=1;
    while c<k:
        prev_path=paths[c-1]

        minw=inf
        paths.append((inf,None));
        for i in range(0,len(prev_path[1])-1):
            t_matrix=deepcopy(matrix)

            for p_j in paths[:c-1]:

                if equal_paths(prev_path[1],p_j[1],i):
                    t_matrix[prev_path[1][i]][p_j[1][i+1]]=inf

            r_w=calc_weigth(matrix,prev_path[1][0:i+1])
            for j in prev_path[1][0:i]:
                for m in range(0,len(t_matrix[j])):
                    pass
                    t_matrix[j][m]=inf
                    t_matrix[m][j]=inf
            new_p=dijkstra(t_matrix,prev_path[1][i],dst)
            new_path=(new_p[0]+r_w#calc_weigth(t_matrix,prev_path[1][0:i]+new_p[1])
                    ,prev_path[1][0:i]+new_p[1]);

            candidates.append(new_path)
        for new_path in candidates:
            if new_path[0]<minw:
                paths[c]=new_path

```

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Вариант №3</div>					Лист
										19
Изм	Лист	№ докум.	Подп.	Дата						

```

        minw=new_path[0]
    if minw==inf:
        break
    else:
        while paths[c] in candidates:
            candidates.remove(paths[c])
        c+=1
    return paths

def check_symmetry(matrix):
    for i in range(0,len(matrix)):
        for j in range(0,len(matrix[i])):
            if matrix[i][j]!=matrix[j][i]:
                break
        else:
            continue
        break
    else:
        return True
    return False

def mat_to_dot(matrix,path=None):
    a=deepcopy(matrix)
    if check_symmetry(matrix):
        graph_type='graph '
        sep='—'
    else:
        graph_type='digraph '
        sep='->'
    dot_p=""
    for i in range(0,len(matrix)):
        if path and i+1 in path:

            dot_p+=str(i+1)+"[color=green,fontcolor=green]\n"
        else:
            dot_p+=str(i+1)+'\n'

    for i in range(0,len(matrix)):
        if graph_type=='graph ':
            beg=i+1
        else:
            beg=0
        for j in range(beg,len(matrix[i])):
            if matrix[i][j]!=inf and i!=j:
                x,y=0,0
                if path and (i+1) in path and (j+1) in path:
                    x,y=path.index(i+1),path.index(j+1)
                if abs(x-y)==1:
                    dot_p+=(str(path[x])+sep+str(path[y])+
                        '[color=green,fontcolor=green,label='+
                        str(matrix[path[x]-1][path[y]-1])+"]\n")
                else:
                    dot_p+=str(i+1)+sep+str(j+1)+"[label="+str(matrix[i][j])+"]\n"
    dot_p=graph_type+"G{\n"+dot_p+"}"
    return dot_p

class Application(Gtk.Builder):
    def __init__(self,ui_filename):
        self.clipped=False

```

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	Вариант №3					Лист
										20
Изм.	Лист	№ докум.	Подп.	Дата						

```

        Gtk.Builder.__init__(self)
        self.add_from_file(ui_filename)
        self.connect_signals(self)
        self.graphviz=None
    def _graphviz_init(self):
        args=[]
        args.append('dot')
        args.append('-T')
        args.append('x11')
        return subp.Popen(args ,shell=False,stdin=subp.PIPE);
    def show_msg(self,msg):
        md=Gtk.MessageDialog(None, Gtk.DialogFlags.MODAL,
                               Gtk.MessageType.WARNING, Gtk.ButtonsType.OK, msg);
        md.run ();
        md.destroy();
    def show(self,form_name):
        window = self.get_object(form_name)
        window.show()
        Gtk.main()
    def on_window_destroy( self,widget, data=None):
        self.get_object('window1').hide()
        if self.graphviz and self.graphviz.poll()==None:
            self.graphviz.terminate()
            print("killed")
            if self.graphviz.poll()==None:
                self.graphviz.kill()
        Gtk.main_quit()

    def adj_changed(*args):
        print("changed")
    def clear_table(self,table):
        model=table.get_model()
        if model:
            model.clear()
        c=table.get_column(0)
        while c:
            table.remove_column(c)
            c=table.get_column(0)

    def on_resize_click(self,widget,data=None):
        table=self.get_object("treeview1")
        self.clear_table(table)
        self.update_tables()
        n=round(self.get_object("adjustment1").get_value())
        for adj in ((self.get_object(i) for i in('adjustment2','adjustment3'))):
            adj.set_property('upper',n)
        table=self.get_object("treeview2")
        table.get_model().clear()

    def resize_click(self,widget,data=None):
        n=round(self.get_object("adjustment1").get_value())
        table="treeview1"
        mat=[[float('inf') for j in range(0,n)] for i in range(0,n)]
        model=self.get_object(table).get_model()
        i=0;j=0;
        for x in model:
            i+=1;j=0
            if i>n:
                break;

```

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	Вариант №3					Лист
										21
Изм	Лист	№ докум.	Подп.	Дата						

```

        for y in x:
            j+=1
            if j>n:
                break
            mat[i-1][j-1]=y
model=self.get_object('liststore3')
model.clear()
for i in range(0,n):
    model.append([i+1])
self.clear_table(self.get_object(table))
self.create_table(table,2,float,mat,True)
for adj in ((self.get_object(i) for i in('adjustment2','adjustment3'))):
    adj.set_property('upper',n)
table=self.get_object("treeview2")
table.get_model().clear()

def update_tables(self):
    n=round(self.get_object("adjustment1").get_value())
    empty_mat=[[float('inf') for j in range(0,n)] for i in range(0,n)]
    self.create_table("treeview1",2,float,empty_mat,True)
    model=self.get_object('liststore3')
    model.clear()
    for i in range(0,n):
        model.append([i+1])
def create_table(self,name,dim,el_type,data,editable=False):
    table=self.get_object(name)
    model=table.get_model()
    self.clear_table(table)
    if dim==1:
        m=len(data)
    elif dim==2:
        m=len(data[0])
    model=Gtk.ListStore(*(el_type for j in range(0,m)))
    if dim==1:
        model.append(data)
    elif dim==2:
        for i in data:
            model.append(i)
    table.set_model(model)
    for i in range(0,m):
        rend = Gtk.CellRendererText()
        if editable:
            rend.set_property("editable",True)
            if el_type==float:
                rend.connect("edited",self.on_cell_edit_f,(model,i))
        col = Gtk.TreeViewColumn(str(i+1), rend)
        col.add_attribute(rend,"text",i)
        table.append_column(col)

def on_cell_edit_f(self, widget, path, text,data=None):
    model,col=data
    model[path][col]=atof(text)

def input_data(self):
    matrix=[[j for j in i] for i in self.get_object("treeview1").get_model()]
    src=round(self.get_object("adjustment2").get_value())-1
    dst=round(self.get_object("adjustment3").get_value())-1
    k=int(self.get_object("entry1").get_text())
    return matrix,src,dst,k

```

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Вариант №3</div>					Лист
										22
Изм	Лист	№ докум.	Подп.	Дата						

```

def output_data(self,paths):
    model=self.get_object("treeview2").get_model()
    model.clear()
    for w,p in paths:
        if p and w!=inf:
            model.append([w,"->".join((str(i+1) for i in p))])
def on_run_click(self,widget,data=None):
    matrix,src,dst,k=self.input_data()
    paths=yen(matrix,src,dst,k)
    self.output_data(paths)

def on_graph_show(self,widget,data=None):
    if self.graphviz and self.graphviz.poll()==None:
        self.graphviz.terminate()
        if self.graphviz.poll()==None:
            self.graphviz.kill()
    graphviz=self._graphviz_init()
    self.graphviz=graphviz
    matrix=[[j for j in i] for i in self.get_object("treeview1").get_model()]
    dot=mat_to_dot(matrix)
    graphviz.stdin.write(dot.encode()+b"\n")
    graphviz.stdin.close()
def on_path_show(self,widget,data=None):
    if self.graphviz and self.graphviz.poll()==None:
        self.graphviz.terminate()
        if self.graphviz.poll()==None:
            self.graphviz.kill()
    graphviz=self._graphviz_init()
    self.graphviz=graphviz
    matrix=[[j for j in i] for i in self.get_object("treeview1").get_model()]
    model=self.get_object("treeview2").get_model()
    selection=self.get_object("treeview2").get_selection()
    print(selection.get_selected_rows()[1])
    path=[int(i) for i in model[selection.get_selected_rows()[1][0]][1].split("->")]
    dot=mat_to_dot(matrix,path)
    #print(dot.encode()+b"\n\x04")
    #print(self.graphviz)
    #print(dir(self.graphviz))
    graphviz.stdin.write(dot.encode()+b"\n")

    graphviz.stdin.close()
def make_symmetric(self,widget,data=None):
    print("ok")
    table=self.get_object("treeview1")
    matrix=[[j for j in i] for i in table.get_model()]
    for i in range(0,len(matrix)):
        for j in range(i+1,len(matrix[i])):
            matrix[j][i]=matrix[i][j]
    self.clear_table(table)
    self.create_table("treeview1",2,float,matrix,True)

inf=float("inf")
app=Application("prac5.ui")
app.show("window1")

```

Подп. и дата		Инв. № дубл.		Взам. инв. №		Подп. и дата		Инв. № подл.	
Изм	Лист	№ докум.	Подп.	Дата	Вариант №3				Лист
									23