

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение
высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Измерительно-вычислительные комплексы»

Курсовая работа

По дисциплине «Алгоритмы и структуры данных»

**Тема «Информационная система поиска пути в
навигационных сетках»**

Р.02069337.22/315-32

Листов 24

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

Руководитель разработки:

доцент каф. ИВК, к.т.н.,
доцент

Шишкин Вадим Викторович

« ____ » _____ 2023 г.

Исполнитель:

студентка гр. ИСТбд-23

Кузнецова Елизавета Юрьевна

« ____ » _____ 2023г.

2023

Содержание

Аннотация.....	1
Техническое задание.....	3
Пояснительная записка	7
Руководство программиста	12
Текст программы.....	18

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение
высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Измерительно-вычислительные комплексы»

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

на курсовую работу

по дисциплине «Алгоритмы и структуры данных»

Тема «Информационная система поиска пути в

навигационных сетках»

Р.02069337. 22/315-32 ТЗ-02

Листов 4

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

Исполнитель:

студентка гр. ИСТбд-23

Кузнецова Е.Ю.

« ____ » _____ 2023 г.

2023

Введение

Информационная система поиска пути в навигационных сетках. Приложение представляет собой поиск кратчайшего пути от одной вершины графа до другой. Нахождение пути происходит посредством сравнения длин путей графа.

Общая характеристика функциональных возможностей:

1. Регистрация/авторизация пользователей
2. Расчет кратчайшего пути

Интерфейс.

При запуске программы на экран выводится авторизация/регистрация, после авторизации окно закрывается и появляется предложение о выборе

документа для считывания. После выполнения алгоритма, выводится граф,

под которым написан кратчайший путь. В случае, если начальная и конечная вершины совпадают, отображается путь, равный 0.

1. Основания для разработки

Учебный план направления 09.03.02 «Информационные системы и технологии» и распоряжение по факультету «О закреплении тем курсовых работ (проектов) за студентами 2 курса ФИСТ направления 09.03.02 «Информационные системы и технологии» (профиль Информационные системы и технологии) по дисциплине «Алгоритмы и структуры данных» от 20.10.2023

2. Требования к программе или программному изделию

2.1. Функциональное назначение

Требуется разработать однопользовательское десктопное приложение по информационному поиску пути в навигационных сетках с графическим интерфейсом в среде Windows.

2.2 Требования к функциональным характеристикам

2.2.1 Требования к структуре приложения

Общая структура приложения должна быть разделена на 2 компонента. Первый отвечает за авторизацию/регистрацию и переход к началу.

Второй отвечает за сам расчет, отрисовку графа и проверку веса путей.

2.2.2 Требования к составу функций приложения

В приложении должны быть реализованы в графическом режиме следующие основные функции:

- регистрация/авторизация пользователя;
- отрисовка графа;
- информирование пользователя о результате.

2.2.2 Требования к организации информационного обеспечения, входных и выходных данных

В приложении должен быть реализован графический интерфейс вывода информации.

Отдельно выделены папки под заготовку вводимой информации, логинов и паролей и для самого кода. Логин и пароль пользователя должны вводиться с клавиатуры. Логин и пароли зарегистрированных пользователей должны храниться в отдельном файле или базе данных в зашифрованном виде.

2.3 Требования к надёжности

Поддержка непрерывной и стабильной работы компьютера.

2.4 Требования к информационной и программной совместимости

Рекомендуется к использованию на Windows 10/11.

При создании программы используются встроенные библиотеки “re”, “os”.

И сторонние библиотеки “tkinter 8.6.”, “matplotlib”, “network”.

Разработка ведётся в “PyCharm community edition 2022.3” на версии языка программирования Python 3.9.

2.5. Требования к маркировке и упаковке

Определяются заданием на курсовую работу.

2.6 Требования к транспортированию и хранению

2.6.1 Условия транспортирования

Требования к условиям транспортирования не предъявляются

2.6.2 Условия хранения

Диск CD-R должен храниться при комнатной температуре, в диапазоне от 20°C до 25°C. Рекомендуется хранить диск в условиях с относительной влажностью воздуха от 20% до 50%. Диск CD-R должен храниться в темном месте, защищенном от прямых солнечных лучей и других источников яркого света. Для предотвращения повреждения диска CD-R рекомендуется хранить его в специальных пластиковых коробках или футлярах, предназначенных для хранения CD-дисков.

2.6 3 Сроки хранения

Срок хранения – до июля 2024 года.

3. Требования к программной документации

Определяется заданием на курсовую работу.

4. Стадии и этапы разработки

Определяется заданием на курсовую работу.

5. Порядок контроля и приёмки

Определяется заданием на курсовую работу.

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение
высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Измерительно-вычислительные комплексы»

Курсовая работа

По дисциплине «Алгоритмы и структуры данных»

**Тема «Информационная система поиска пути в
навигационных сетках»**

Пояснительная записка

Р.02069337. 22/315-32 ПЗ-02

Листов 5

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

Руководитель разработки:

доцент каф. ИВК, к.т.н.,
доцент

Шишкин Вадим Викторович

«_____» _____ 2023 г.

Исполнитель:

студентка гр. ИСТбд-23
Кузнецова Елизавета Юрьевна
«_____» _____ 2023г.

2023

Введение

Приложение «Информационная система поиска пути в навигационных сетках».

В работе выбраны такая структура данных, как массив. Структура данных — массив была выбрана потому, что эта структура имеет множество преимуществ:

1. Массивы обеспечивают произвольный доступ к элементам. Это ускоряет доступ к элементам по положению;
2. Массивы хранят несколько данных похожих типов с одним и тем же именем;
3. В массиве данные организованы таким образом, что ими легко и удобно манипулировать.

1. Проектная часть

1.1 Постановка задачи на разработку приложения

Определяется заданием на курсовую работу

1.2 Математические методы

В качестве математической модели для хранения смежной матрицы был выбран двумерный массив, он позволяет легко записать длины ребер, а также может быть легко изменен, что упрощает произведение расчетов. Каждый элемент двумерного массива является длиной определенного ребра.

1.3 Архитектура и алгоритмы

1.3.1. Архитектура



1. Регистрация/Авторизация.

Интерфейс, созданный с помощью библиотеки Tkinter позволяющий пользователю зарегистрировать новый аккаунт и пользоваться приложением с помощью уже существующего.

2. Алгоритм Дейкстры.

Расчёт кратчайшего пути.

3. Отрисовка основных виджетов.

Вывод на экран графа.

1.3.2.1. Алгоритм проверки регистрации

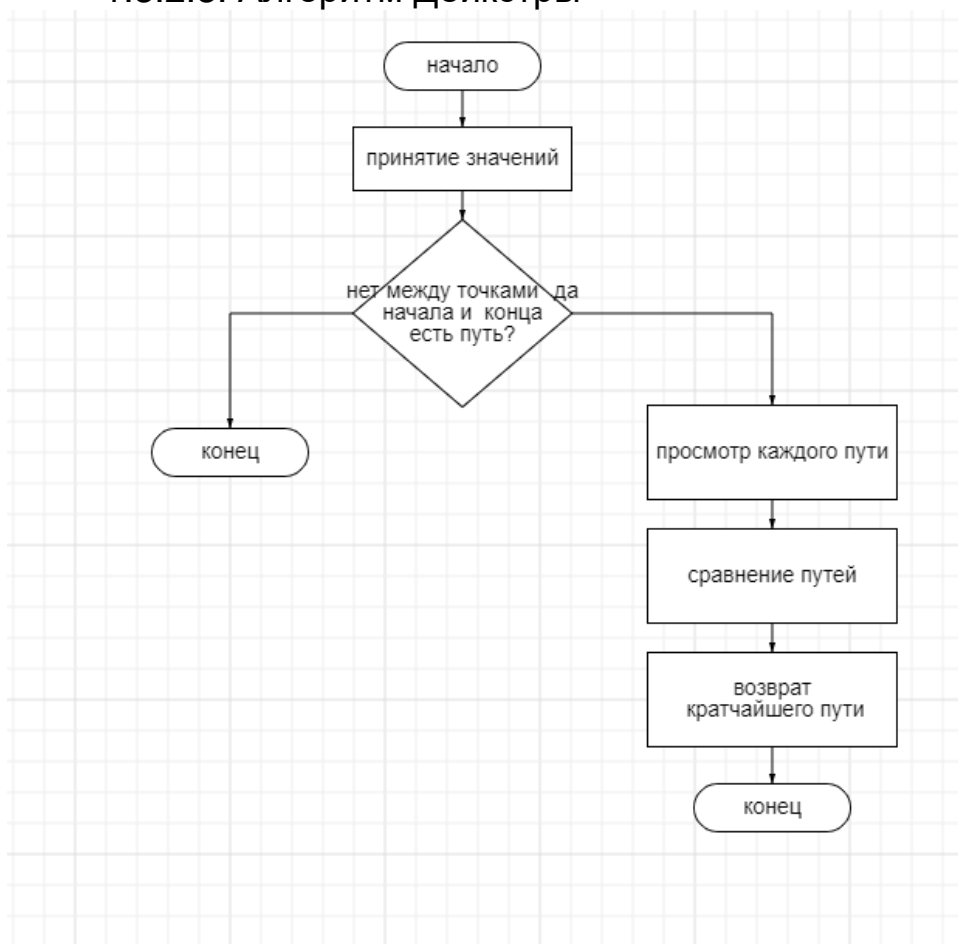
Данный алгоритм необходим для проверки регистрации пользователя. Алгоритм принимает логин и пароль, шифрует пароль, сверяет полученные данные с данными о зарегистрированных пользователях. Если пользователь с таким логином найден и пароль соответствует записанному в файл с зарегистрированными, то алгоритм позволяет выбрать файл, иначе выводит предупреждение о том, что данные некорректны.

1.3.2.2. Алгоритм шифрования.



Данный алгоритм необходим для обеспечения безопасности данных пользователей. Алгоритм принимает на вход пароль пользователя, шифрует его с помощью ключа и возвращает зашифрованный пароль.

1.3.2.3. Алгоритм Дейкстры



Данный алгоритм необходим для вычисления кратчайшего пути. Алгоритм принимает значения из текстового документа и, обработав их, возвращает кратчайший путь между точками начала и конца.

1.4 Тестирование

Весь процесс тестирования проходил вручную, без привлечения специального ПО. На протяжении всего хода разработки, использовался метод белого ящика, так как в любое время имелся доступ ко всем компонентам программы. Всё тестирование выполнялось интуитивным методом, без подготовки специальных тестов.

На протяжении всего хода разработки, по мере добавления новых функций программы, использовалось системное тестирование новых функций, для устранения возникших в ходе написания ошибок. После положительных результатов тестирования функция считалась внедренной.

2. Источники, использованные при разработке

1. YouTube [Электронный ресурс]: Алгоритм Дейкстры (Dijkstra's algorithm) | Алгоритмы на Python – YouTube – URL:
https://www.youtube.com/watch?v=MCfjc_UIP1M;
2. Wikipedia [Электронный ресурс]: Алгоритм Дейкстры – URL:
https://ru.wikipedia.org/wiki/Алгоритм_Дейкстры.

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение
высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Измерительно-вычислительные комплексы»

Курсовая работа

По дисциплине «Алгоритмы и структуры данных»

**Тема «Информационная система поиска пути в
навигационных сетках»**

Руководство программиста

Р.02069337. 22/315-32 РП-01

Листов 6

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

Исполнитель:

студент гр. ИСТбд-23

Кузнецова Елизавета

Юрьевна

«_____» _____ 2023 г.

2023

1. Назначение и условия применения программы

1.1 Назначение и функции, выполняемые приложением

Приложение «Информационная система поиска пути в навигационных сетках» реализовано для поиска кратчайшего пути между точками. Приложение реализовано со следующими условиями:

Для считывания текстового документа, в нём должны быть соблюдены условия ввода:

Первая строка - количество точек в графе

Смежная матрица

На следующей, после матрицы, строке точка начала

На следующей строке, после точки начала, точка конца

Пример:

```
6
0 3 1 0 0 0
3 0 4 0 0 0
1 4 0 0 7 5
0 0 0 0 0 2
0 0 7 0 0 4
0 0 5 2 4 0
3
4
```

В приложении предоставляется возможность зарегистрироваться чтобы пользоваться приложением со своим аккаунтом, в приложении реализовано вычисление кратчайшего пути в графе от одной точки до другой. Выводится граф и кратчайший путь.

1.2 Условия, необходимые для использования приложения

Рекомендуется к использованию на Windows 10/11.

При создании программы используются встроенные библиотеки “re”, “os”.

И сторонние библиотеки “tkinter 8.6.”, “matplotlib”, “network”.

Разработка ведётся в “PyCharm community edition 2022.3” на версии языка программирования Python 3.9.

2. Характеристики программы

2.1 Характеристики приложения

Значимых строк кода 228

Структура данных одна – массив.

Использованные библиотеки:

“os” –библиотека функций для работы с операционной системой. Методы, включенные в неё, позволяют определять тип операционной системы, получать доступ к переменным окружения, управлять директориями и файлами

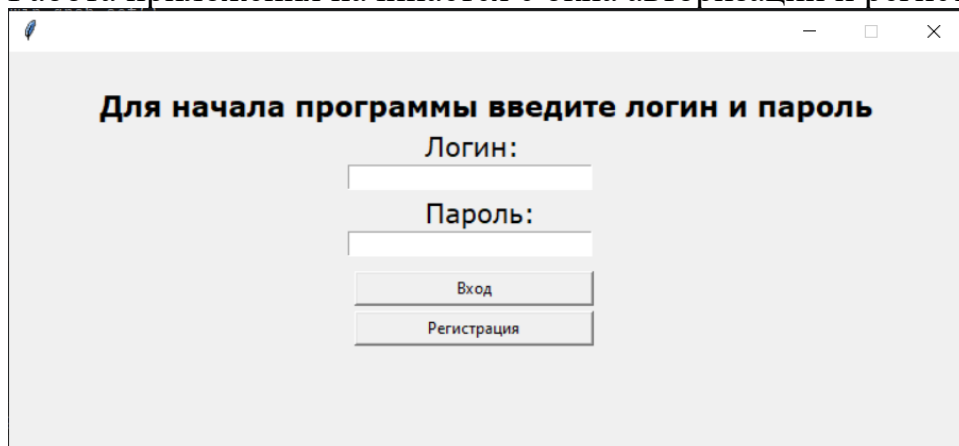
“tkinter 3.6” - библиотека для разработки графического интерфейса на языке Python. Методы, включенные в неё, позволяют создавать окна, размещать на них виджеты, настраивать параметры окна и виджетов.

“re” - это модуль языка программирования Python, предназначенный для работы со строками и реализующий регулярные выражения.

“matplotlib” - библиотека Python для создания визуализации данных.

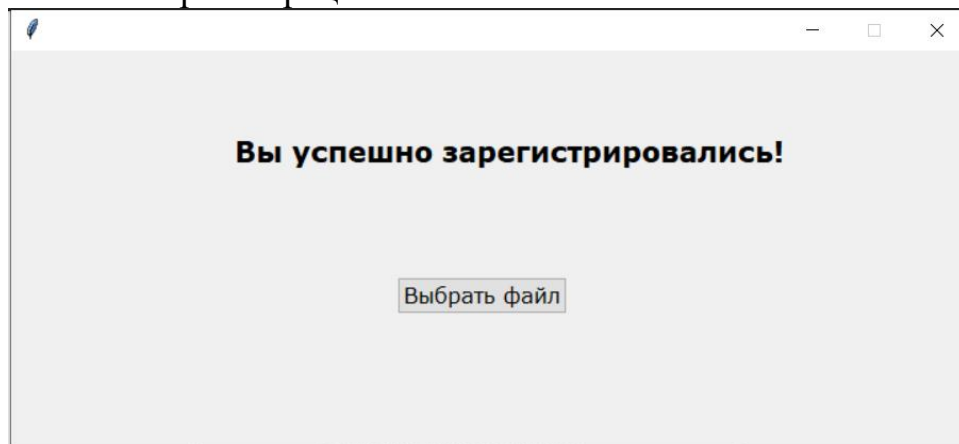
“network” - программный пакет на языке Python для создания, управления и изучения структуры, динамики и функционирования сложных сетей. Он используется для изучения больших сложных сетей, представленных в виде графов с узлами и ребрами.

Работа приложения начинается с окна авторизации и регистрации



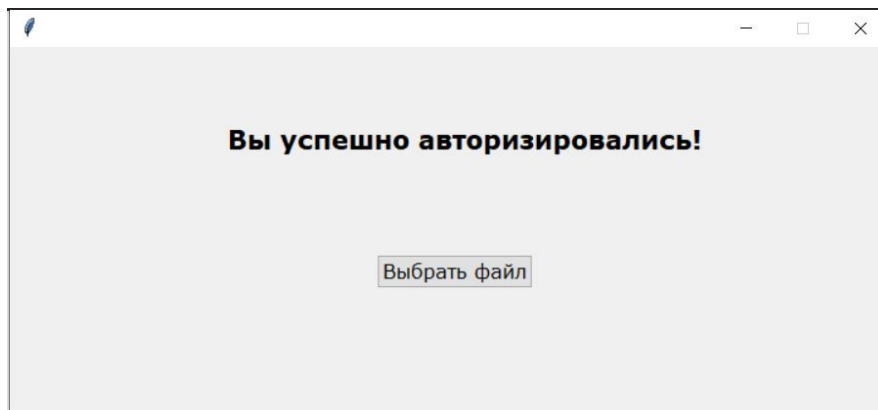
The screenshot shows a window titled "Для начала программы введите логин и пароль" (To start the program, enter login and password). It contains two input fields labeled "Логин:" (Login) and "Пароль:" (Password). Below these fields are two buttons: "Вход" (Login) and "Регистрация" (Registration).

Успешная регистрация

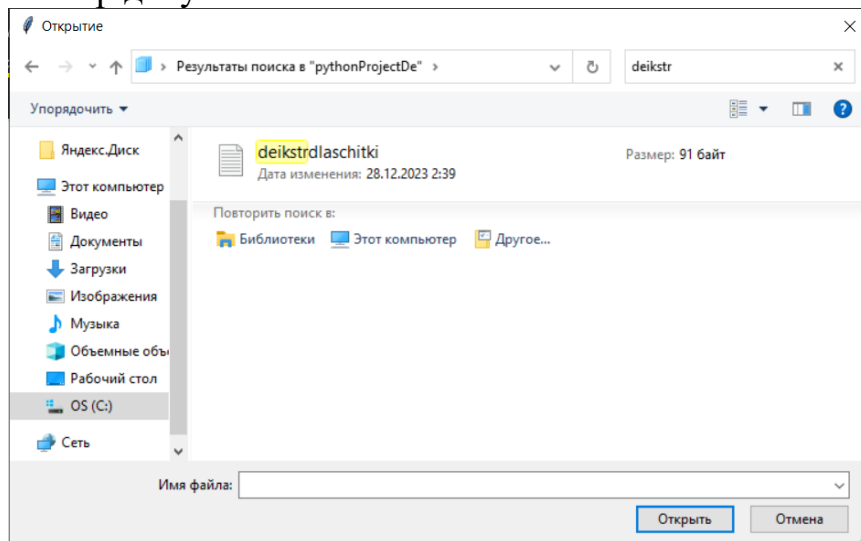


The screenshot shows a window titled "Вы успешно зарегистрировались!" (You have successfully registered!). It contains a single button labeled "Выбрать файл" (Select file).

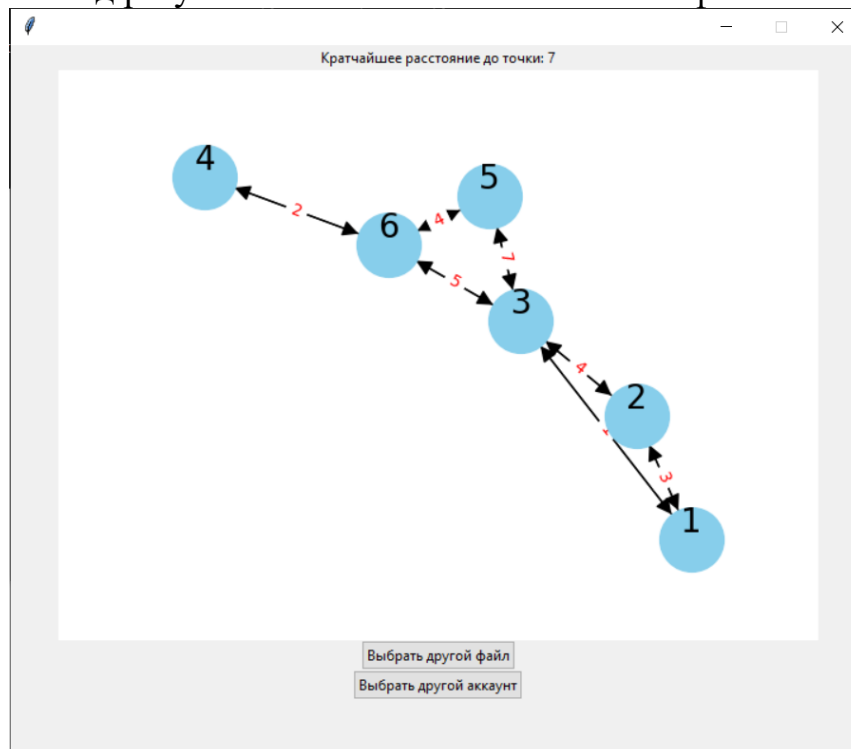
Успешная авторизация



Выбор документа



Вывод результата и возможность сменить файл или аккаунт



2.2 Особенности реализации приложения

В программе используется структура данных массив, эта структура была выбрана ввиду удобной навигации, он позволяет легко записать длины ребер, а также может быть легко изменен, что упрощает произведение расчетов. Каждый элемент двумерного массива является длиной определенного ребра.

3. Обращение к программе

Класс GraphApp содержит в себе следующие методы:

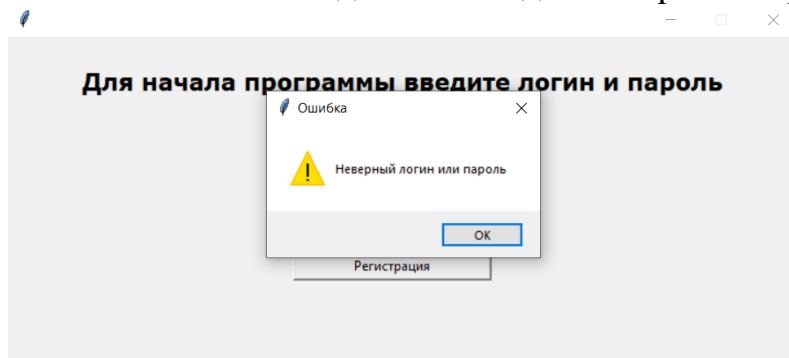
- 1) Метод `create_login_widgets`
Создает кнопки, надписи и поля ввода на окне
- 2) Метод `authenticate_user`
Авторизация пользователя, проверяет наличие логина и соответствующего ему пароля в файле с записанными данными пользователей
- 3) Метод `register_user`
Регистрация пользователя, проверяет наличие логина в файле с записанными данными пользователей и если пользователь с таким логином ещё не зарегистрирован, то вносит данные о пользователе.
- 4) Метод `create_file_selection_widgets`
Принятие файла, считывание информации оттуда. Если точки совпадают или не имеют пути между собой, выдает соответствующее сообщение.
- 5) Метод `read_file`
Считывание данных с файла и преобразование их в массив.
- 6) Метод `convert_to_int`
Возвращает значение массива, преобразованное в число.
- 7) Метод `dijkstra_algorithm`
Проводится анализ данных в массиве. Возвращается значение самого короткого пути между точками начала и конца.
- 8) Метод `go_from`
Определяется минимальный путь до каждой точки в смежной матрице.
- 9) Метод `create_graph`
Создается окно вывода графа и результата. Рисуются граф.

Алгоритмы, используемые в программе:

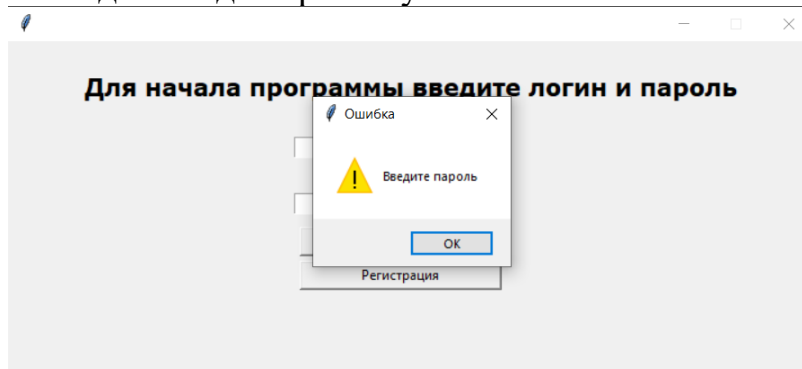
- 1) `codinpass`
Шифрование пароля пользователя, сдвигает каждый символ пароля с помощью ключа и возвращает зашифрованный пароль
- 2) `open_file`
Открывает файл с данными о регистрации пользователей, если не находит его, то создаёт новый файл
- 3) `dismiss`
Получает окно и позволяет перехватить управление
- 4) `main`
Создание рабочего окна и привязка данных.

4. Сообщения

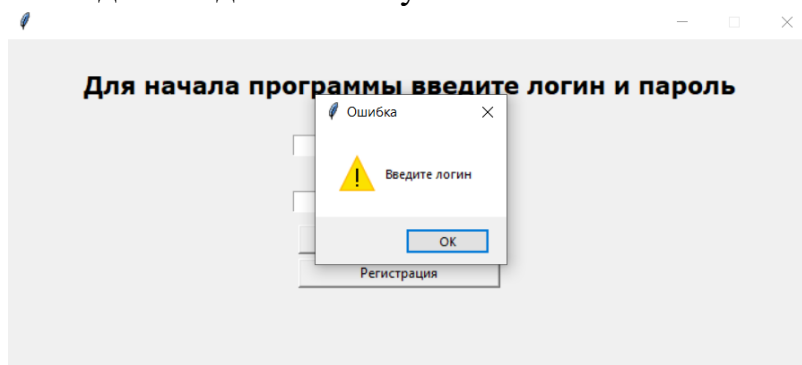
Пользователь не найден или введен неверный пароль



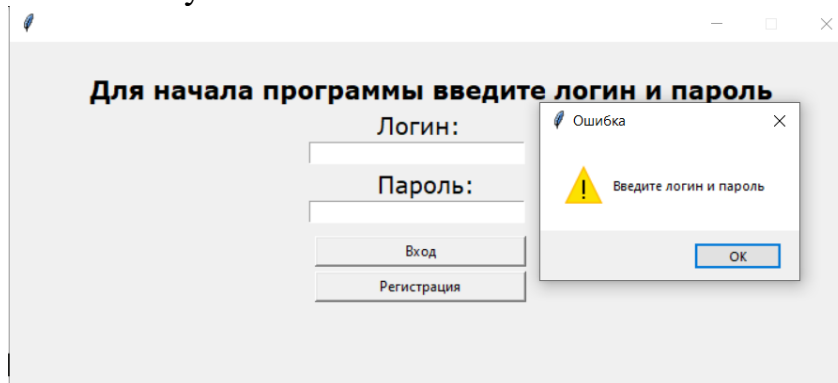
Поле для ввода пароля пустое



Поле для ввода логина пустое



Оба поля пустые:



Текст программы

```
import re
from tkinter import filedialog, messagebox
import matplotlib.pyplot as plt
import networkx as nx
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from tkinter import ttk
from tkinter import *
import os

def dismiss(win):
    win.grab_release()
    win.destroy()
def password_code(password):
    key = 3
    coded_password = ""
    for i in password:
        coded_password_temp = chr(ord(i) + key)
        coded_password += coded_password_temp
        key = -key + 1
    return coded_password

def open_file():
    try:
        text = open("useripass.txt", "r+")
        return text
    except FileNotFoundError:
        try:
            text = open("useripass.txt", "w")
            text.close()
            text = open("useripass.txt", "r+")
            return text
        except FileNotFoundError:
            text = open("useripass.txt", "r+")
            return text

class GraphApp:
    def __init__(self, root):
        self.root = root
        self.create_login_widgets()
        self.users = {}
```

```

button_style = ttk.Style()
button_style.configure("my.TButton", font="Verdana 12")

def create_login_widgets(self):
    self.label = Label(text="Для начала программы введите логин и пароль",
font="Verdana 16 bold")

    self.username_label = Label(self.root, text="Логин:", font="Verdana 15")
    self.username_entry = Entry(self.root, width=30, justify="center")

    self.password_label = Label(self.root, text="Пароль:", font="Verdana 15")
    self.password_entry = Entry(self.root, width=30, justify="center", show="*")

    self.register_button = Button(self.root, text="Регистрация",
command=self.register_user)

    self.login_button = Button(self.root, text="Вход",
command=self.authenticate_user)

    self.label.place(x=65, y=25)
    self.username_label.place(x=310, y=55)
    self.username_entry.place(x=255, y=85)
    self.password_label.place(x=310, y=105)
    self.password_entry.place(x=255, y=135)
    self.login_button.place(x=260, y=165, width=180)
    self.register_button.place(x=260, y=195, width=180)

def register_user(self):
    login = self.username_entry.get()
    password_raw = self.password_entry.get()
    password = password_code(password_raw)

    if len(login) == 0 and len(password) == 0:
        messagebox.showwarning(title="Ошибка", message="Введите желаемые
логин и пароль")

    elif len(login) == 0 and len(password) != 0:
        messagebox.showwarning(title="Ошибка", message="Введите логин")

    elif len(login) != 0 and len(password) == 0:
        messagebox.showwarning(title="Ошибка", message="Введите пароль")

    else:
        file = open_file()

```

```

temp = file.readline()[:-1].split(' ')

while True:
    if temp != [""]:
        self.users[temp[0]] = temp[1]
        temp = file.readline()[:-1].split(' ')
    else:
        break

flag_reg = False

for i in self.users.items():
    l, p = i
    if login == l:
        flag_reg = True

if not flag_reg:
    file = open_file()
    file.seek(0, os.SEEK_END)
    file.write(f'{login} {password}\n')
    file.close()

    for widget in self.root.winfo_children():
        widget.destroy()

    Label(self.root, text=f"Вы успешно зарегистрировались!",
           font="Verdana 16 bold").place(x=165, y=60)
    button = ttk.Button(self.root, text="Выбрать файл",
style="my.TButton", command=self.create_file_selection_widgets)
    button.place(x=290, y=170)
else:
    messagebox.showwarning(title="Ошибка", message="Такой аккаунт
уже существует")

def authenticate_user(self):
    login = self.username_entry.get()
    password_raw = self.password_entry.get()
    password = password_code(password_raw)

    if len(login) == 0 and len(password) == 0:
        messagebox.showwarning(title="Ошибка", message="Введите логин и
пароль")

    elif len(login) == 0 and len(password) != 0:
        messagebox.showwarning(title="Ошибка", message="Введите логин")

```

```

elif len(login) != 0 and len(password) == 0:
    messagebox.showwarning(title="Ошибка", message="Введите пароль")

else:
    file = open_file()
    a = file.readline()[:-1].split(" ")

    while True:
        if a != [""]:
            self.users[a[0]] = a[1]
            a = file.readline()[:-1].split(" ")
        else:
            break

    flag_reg = False
    for i in self.users.items():
        login_check, password_check = i
        if login == login_check and password == password_check:
            flag_reg = True
            break

    if flag_reg:
        for widget in self.root.winfo_children():
            widget.destroy()

        Label(self.root, text="Вы успешно авторизовались!", font="Verdana
16 bold").place(x=175, y=60)
        button = tk.Button(self.root, text="Выбрать файл",
style="my.TButton", command=self.create_file_selection_widgets)
        button.place(x=300, y=170)

    else:
        messagebox.showwarning(title="Ошибка", message="Неверный логин
или пароль")

def create_file_selection_widgets(self):
    file_path = filedialog.askopenfilename()
    if file_path:
        data = self.read_file(file_path)
        start_node = self.convert_to_int(data[self.convert_to_int(data[0]) + 1])
        if start_node == (self.convert_to_int(data[self.convert_to_int(data[0]) +
2])):
            messagebox.showwarning(title="Ошибка", message="Начальная и
конечная вершины совпадают\n Выберите другой файл")
        else:

```

```

shortest_paths = self.dijkstra_algorithm(data, start_node)
if shortest_paths == "Путь не существует":
    messagebox.showwarning(title="Ошибка", message="Между
указанными точками не существует пути\n Выберите другой файл")
else:
    self.create_graph(data, shortest_paths)

def read_file(self, file_path):
    file = open(file_path, "r")
    values = file.read().split("\n")
    data = []
    for key in values:
        value = re.findall(r"[-+]?[d*]\.d+|\d+", key)
        if value:
            data.append(value)
    return data

def convert_to_int(self, value):
    return int("".join(map(str, value)))

def dijkstra_algorithm(self, data, start_node):
    n = self.convert_to_int(data[0])
    W = [[int(i) for i in row] for row in data[1:n + 1]]
    start = self.convert_to_int(data[n + 1]) - 1
    finish = self.convert_to_int(data[n + 2]) - 1
    INF = 1e8
    visited = [False] * n
    dist = [INF] * n
    dist[start] = 0
    while False in visited:
        u = self.go_from(dist, visited)
        for v in range(n):
            if W[u][v] != 0 and not visited[v]:
                dist[v] = min(dist[v], dist[u] + W[u][v])
        visited[u] = True

    if INF in dist:
        return "Путь не существует"
    else:
        return dist[finish]

def go_from(self, dist, visited):
    index = 0
    dist_min = float('inf')
    for i in range(len(dist)):

```

```

        if dist[i] < dist_min and not visited[i]:
            dist_min = dist[i]
            index = i
    return index

def create_graph(self, data, shortest_paths):
    G = nx.DiGraph()
    n = self.convert_to_int(data[0])
    edges = []
    for i in range(n):
        for j in range(n):
            if self.convert_to_int(data[i + 1][j]) != 0:
                edges.append((i, j, self.convert_to_int(data[i + 1][j])))
    G.add_weighted_edges_from(edges)

    pos = nx.spring_layout(G, seed=10)
    labels = { }
    for i in range(n):
        labels[i] = str(i+1)

    edge_labels = {(i, j): str(G[i][j]['weight']) for i, j in G.edges}

    fig, ax = plt.subplots()
    nx.draw(G, pos, with_labels=False, node_color='skyblue', node_size=1500,
arrowsize=20, ax=ax)
    nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels,
font_color='red', ax=ax)
    nx.draw_networkx_labels(G, pos, labels, font_size=20, font_color='black',
ax=ax,
                        verticalalignment='bottom')

    win = Toplevel()
    w = win.winfo_screenwidth()
    h = win.winfo_screenheight()
    w = w // 2 # середина экрана
    h = h // 2
    w = w - 360 # смещение от середины
    h = h - 300
    win.geometry(f'720x600+{w}+{h}')
    win.title("")
    win.resizable(False, False)
    win.protocol('WM_DELETE_WINDOW', lambda: dismiss(win))
    win.grab_set()

    shortest_path_label = Label(win, text=f"Кратчайшее расстояние до точки:

```

```

{shortest_paths}")
    shortest_path_label.pack()
    canvas = FigureCanvasTkAgg(fig, master=win)
    canvas.draw()
    canvas.get_tk_widget().pack()
    ttk.Button(win, text="Выбрать другой файл", command=lambda:
(win.destroy())).pack()
    ttk.Button(win, text="Выбрать другой аккаунт", command=lambda:
(win.destroy(), self.root.destroy(), main())).pack()

```

```

def main():
    root = Tk()
    root.title("")
    w = root.winfo_screenwidth()
    h = root.winfo_screenheight()
    w = w // 2 # середина экрана
    h = h // 2
    w = w - 360 # смещение от середины
    h = h - 150
    root.geometry(f'720x300+{w}+{h}')
    root.resizable(False, False)
    app = GraphApp(root)
    root.mainloop()

```

```

if __name__ == "__main__":
    main()

```