|  |  |  |  |
| --- | --- | --- | --- |
| Министерство образования и науки Российской Федерации | | | |
| федеральное государственное бюджетное образовательное учреждение  высшего образования | | | |
| «Иркутский государственный университет» | | | |
| (ФГБОУ ВО «ИГУ») | | | |
| Факультет бизнес-коммуникаций и информатики | | | |
| Кафедра естественнонаучных дисциплин | | | |
|  |  | | | |
|  | |  | | | | |
|  | |  | | | | |
|  | |  | | | | |
| **ОТЧЕТ ПО ОЗНАКОМИТЕЛЬНОЙ ПРАКТИКЕ**  **по направлению**  **09.03.03 «Прикладная информатика»**  **Профиль «Прикладная информатика (разработка программного обеспечения)»**  Быстрый алгоритм построения матрицы смежности для графового алгоритма | | | | | |
|  | | | | | |
|  | |  | | | |
|  | | | Студент 1 курса очного отделения, | | |
|  | | | Группа 14123 | | |
|  | | | \_\_\_\_\_\_\_\_\_\_ И.Л. Чикиндин | | |
|  | | |  | | |
|  | | | Руководитель: к.п.н., доцент  \_\_\_\_\_\_\_\_\_\_\_\_ М.А. Сокольская | | |
|  | | |  | | |
|  | | |  | | |
|  | | | Работа защищена: | | |
|  | | | «27» июня 2025 г. | | |
|  | | | С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_ | | |
|  | | |  | | |
|  | | |  | | |
| **Иркутск 2025** | | | | | |

|  |
| --- |
|  |

# Содержание

[Содержание 1](#_Toc137994492)

[Введение 2](#_Toc137994493)

[Глава 1 4](#_Toc137994494)

[1.1. Постановка задачи и выбор инструментов для её решения 4](#_Toc137994495)

1.2. Обзор алгоритмов……………………………………………………...5

[1.3. Проектирование функциональности приложения](#_Toc137994496) 5

[Глава 2 7](#_Toc137994497)

[2.1. Создание интерфейса приложения 6](#_Toc137994499)

[2.2. Разработка функционала приложения 5](#_Toc137994500)

[Заключение 5](#_Toc137994501)

[Список используемых источников 6](#_Toc137994502)

[Приложение 1. ЕЖЕДНЕВНЫЕ ЗАПИСИ СТУДЕНТА ПО ПРАКТИКЕ 7](#_Toc137994503)

[Приложение 2. МНЕНИЕ СТУДЕНТА ПО ИТОГАМ ПРАКТИКИ И ЕГО ПРЕДЛОЖЕНИЯ. 8](#_Toc137994504)

## 

# Введение

Место прохождения практики – 664033 г. Иркутск, ул. Лермонтова, 126

Практика по получению первичных умений и опыта профессиональной деятельности является частью учебно-воспитательного процесса и формирует профессиональные умения и навыки, необходимые для будущей практической деятельности.

Практика является частью учебно-воспитательного процесса и формирует первичные профессиональные умения и навыки, необходимые для будущей практической деятельности.

Основная **цель практики** – применение теоретических знаний в практической деятельности и выработка профессиональных навыков и умений в проектировании, подборе инструментов реализации, создании приложений различного назначения.

Виды (содержание), объемы и сроки прохождения практики определены образовательной программой направления, предусматривающей будущую профессиональную деятельность и возможности его профессиональной адаптации.

**Задачи ознакомительной практики**:

* формирование у обучающихся дизайн-мышления, заключающегося в умении использовать творческие приемы при решении инженерных задач;
* развитие понимания социальной значимости профессиональной деятельности через мотивацию и стремление к ее выполнению;
* отработка и закрепление методов и приемов самообучения для повышения профессиональной квалификации и мастерства;
* изучение методов и приемов использования современных информационно-коммуникационных технологий в профессиональной деятельности;
* практическая отработка и закрепление изученных ранее методов и приемов поиска и самостоятельного изучения литературы и электронно-образовательных ресурсов в сфере профессиональной деятельности;
* формирование умений и навыков по разработке, эксплуатации и сопровождению информационных систем и сервисов предприятий;
* формирование практических навыков участия в реализации профессиональных коммуникаций в рамках проектных групп (работа в коллективе);
* практическая отработка навыков и умения подготовки и проведения презентации результатов выполнения проекта и начального обучения пользователей информационных систем.

Объектом исследования является процесс создания приложения для поиска оптимального пути с помощью графов.

Предметом исследования является проектирование логики взаимодействия интерфейса и алгоритма, разработка приложения.

Структура отчета: введение, две главы, заключение, список используемых источников, приложения.

# Глава 1

## Постановка задачи и выбор инструментов для её решения

Задача практики звучит так: написать алгоритм построения матрицы смежности для графового алгоритма пересечения реки по данным ГИС и требований ГОСТов. Для разработки использовался язык программирования python и графовые алгоритмы.

Были определены такие этапы решения этой задачи:

1. Обсудить задачу с научным руководителем
2. Найти алгоритмы для построения оптимального маршрута
3. Выбрать инструменты для реализации программы
4. Написать интерфейс и логику для него
5. Написать логику алгоритма для нахождения пути
6. Записать оптимальные маршруты в матрицу смежности

Промежуточные результаты:

1. Было найдено шесть подходящих алгоритма: A\*, алгоритм SPFA, муравьиный алгоритм, алгоритм Флойда-Уоршелла, алгоритм Джонсона, алгоритм PCA
2. Было выбрано шесть библиотеки: numpy, matplotlib, tkinter, python-igraph, networkx
3. Разработан интерфейс с логикой
4. Для решения этого этапа были применены алгоритм А\* и алгоритм PCA
5. Программа записывает результат в виде матрицы смежности в csv файл

Было выбрано шесть инструментов, таких как numpy, tkinter, matplotlib, python-igraph, networkx.

Библиотека Numpy дает высокопроизводительные вычисления с массивами и матрицами. Библиотека tkinter — это простой графический инструмент для реализации интерфейсов. Библиотека matplotlib дает начертить матрицы больших размеров, чего tkinter не может сделать. Библиотека python-igraph позволяет использовать такой тип данных как граф и операции над ними. Библиотека networkx также поддерживает графы и операции над нами, но в добавок в ней есть готовый алгоритм А\*.

Весь код был написан в Visual Studio Code.

## Обзор алгоритмов

Для работы приложения было выбрано несколько алгоритмов. В этом параграфе опишем эти алгоритмы.

Алгоритм A\* — это модификация алгоритма Дейкстры, оптимизированная для единственной конечной точки. Алгоритм предназначен для поиска кратчайшего пути в графе с эвристикой. Сложность алгоритма O(bd), где b — это ветвление, а d — это глубина решения.

Алгоритм SPFA — это улучшенная версия алгоритма Беллмана-Форда для поиска кратчайших путей в графе, особенно эффективная в графах с ребрами отрицательного веса. Сложность алгоритма O(VE).

Муравьиный алгоритм — это алгоритм оптимизации, вдохновленный поведением муравьев при поиске пути от места к месту. Алгоритм предназначен для решения задач оптимизации. Сложность алгоритма

O(N\* K \* n2), где N — это итерации, K — муравьи, n — вершины.

Алгоритм Флойда-Уоршелла — это мощный универсальный инструмент для поиска кратчайших путей в графах. Алгоритм предназначен для поиска кратчайших путей между всеми парами вершин. Сложность алгоритма О(V3).

Алгоритм Джонсона — это алгоритм для нахождения кратчайшего пути между всеми парами вершин во взвешенном ориентированном графе с любыми весами ребер, но не имеющем отрицательных циклов. Этот алгоритм использует алгоритм Дейкстры. Сложность алгоритма O(V2logV+VE).

Алгоритм PCA — алгоритм обучения без учителя, используемый для понижения размерности и выявления наиболее информативных признаков в данных. С помощью этого алгоритма можно понять как, относительно реки строить путь. Сложность алгоритма O(N · d2).

## Проектирование функциональности приложения

Для разработки приложения применялось в основном функциональное проектирование. При таком методе программирования система разбивается на независимые, функционально завершенные модули с четко определенными интерфейсами взаимодействия. Проект поделен на три файла. В первом файле содержится интерфейс программы. Во втором файле содержится логика интерфейса. В третьем файле содержится логика работы алгоритма.

На вход программа принимает csv файл и стоимостный коэффициент. Файл должен содержать 3 столбца разделенные запятой. В первом столбце файл должен содержать координаты по x. Второй столбец должен содержать координаты по y. Третий столбец должен содержать 0 или 1, где 0 — это земля, а 1— это вода. Стоимостный коэффициент нужен для умножения всего водного пути на него.

На выходе программа дает csv файл с матрицей смежности, где вершины графа — это порядковый номер координаты из файла.

Диаграмма взаимодействия в проекте.

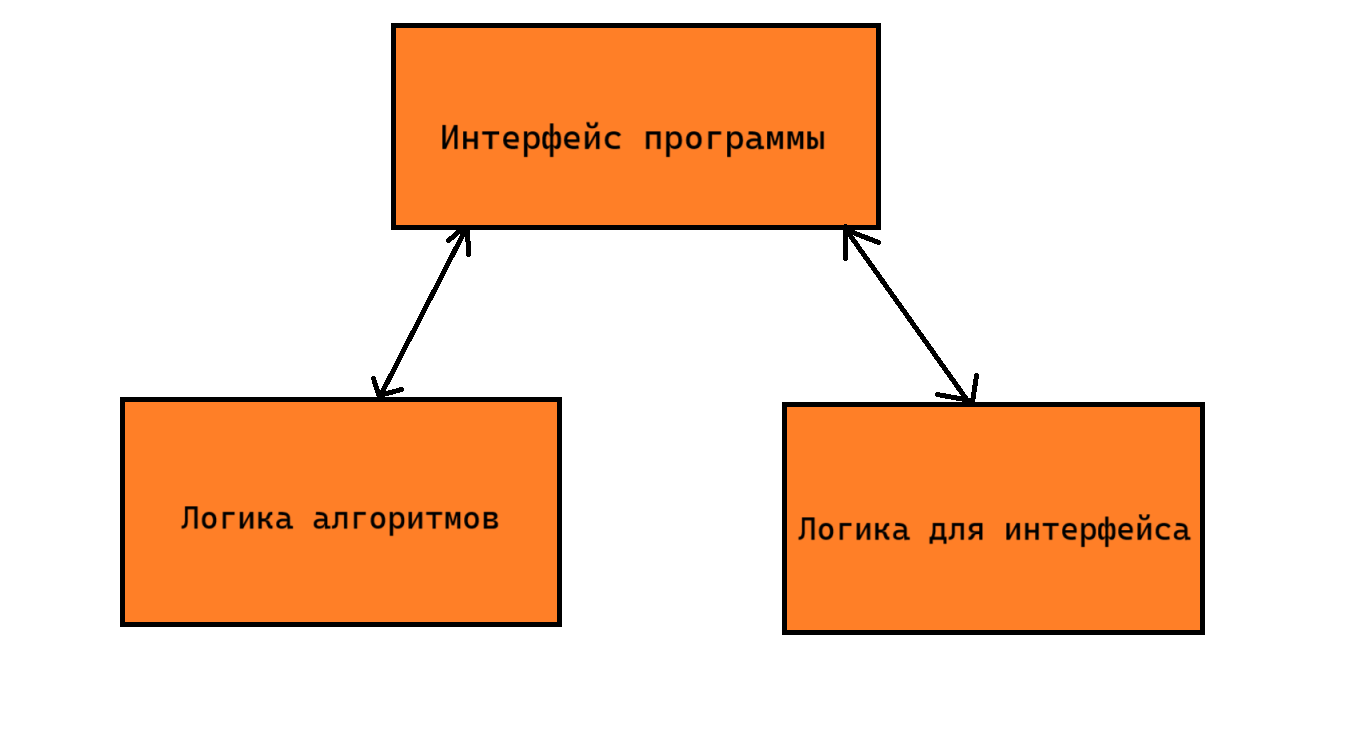


Рис 1. Диаграмма взаимодействия

Функции в интерфейсе программы:

1. Функция для получения данных от пользователя в интерфейсе
2. Функция для вызова второго окна с матрицей
3. Класс для матрицы

Функции в логике интерфейса

1. Функция для получения шага сетки матрицы и размерности матрицы
2. Функция для создания numpy матрицы, где элементы матрицы это 0 или 1 из входного файла
3. Функция для получения соседей у выбранных точек в матрице
4. Функция для получения двух точек, которые пользователь сам выбрал на матрице из интерфейса

Функции для алгоритма:

1. Алгоритм А\*
2. Алгоритм PCA
3. Функция для расчета стоимости
4. Функция для нахождения соседей
5. Функция для нахождения эвристики

Прототипы интерфейса:

1. Окно для сбора данных у пользователя



Рис 2. Первое окно интерфейса

1. Окно с матрицей из входного файла

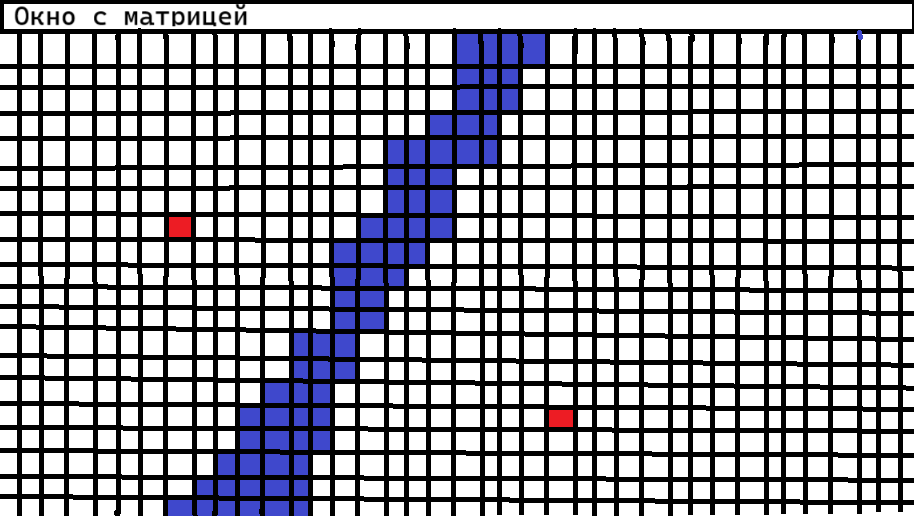


Рис 3. Окно с матрицей

# Глава 2



## Создание интерфейса приложения

Интерфейс написан на библиотеке tkinter. Cначала было разработано окно, где пользователь вводит стоимостный коэффициент и название файла, который находится в одной директории с файлами программы. Дальше нужно было разработать окно, где пользователь может выбрать стартовую точку и конечную точку.

Второе окно с матрицей должно было быть написано на функциях, но позже было решено написать его на ООП. Тем самым ушло меньше времени на тесты, код стал чище. Первая версия этого окна была написано полностью на tkinter. И при тестах этот подход показал себя ужасно. Матрицу 800х800 блоков tkinter не мог отрисовать из-за чего окно с матрицей не появлялось. Было понятно, что одной библиотеки tkinter было не достаточно. За подсказкой было решено обратиться к научному руководителю. Получив подсказку от научного руководителя, была добавлена библиотека matplotlib для отрисовки матрицы с помощью функций FigureCanvasTkAgg(), get\_tk\_widget(), draw().

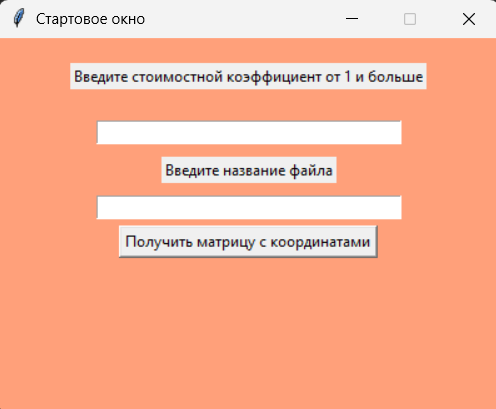


Рис 4. Стартовое окно программы



Рис 5. Окно с матрицей

## Разработка функционала приложения

Параллельно с разработкой интерфейса также писалась логика для интерфейса. Сначала была написана функция для получения шага между клетками сетки и размерности матрицы. Потом было решено написать функцию, которая брала данные из файла и формировала их в numpy матрицу.

После тестов интерфейса и логики для него, было решено начать писать алгоритмы для программы. Но для начала нужно было понять, как работают эти алгоритмы. Чтобы разобраться с алгоритмом А\* пришлось вручную порешать задачи.

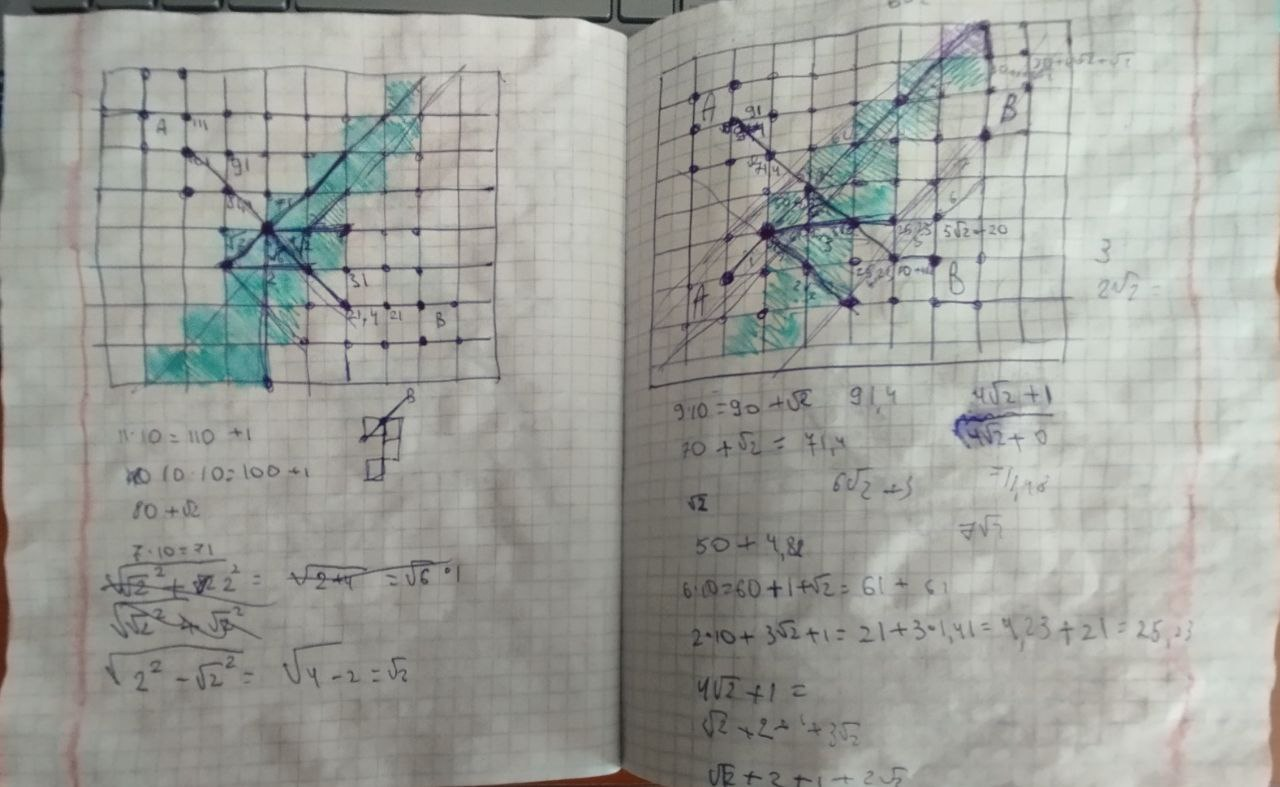


Рис 6. Решение задач с алгоритмом А\*

Разобравшись, как они работают, можно было приступать к разработке. Алгоритмы были написаны в виде функций. Алгоритм А\* был разбит на несколько функций: функция алгоритма, функция для нахождения эвристики для вершин графа, функция для расчета стоимости вершины графа. Функция алгоритма А\* возвращает путь из вершин по координатам, общую стоимость пути и возможные вершины, с которыми можно было соединится.

Алгоритм PCA был разработан в одной функции. Он оказался простым в разработке, т.к. функции для написания этого алгоритма уже были в библиотеке numpy. Функция этого алгоритма возвращает направление для перпендикулярного пути относительно реки.

Программа принимает csv файл и стоимостный коэффициент. Результат в виде матрицы смежности записывается в csv файл.

Программа запускается из файла с интерфейсом. Появляется окно с двумя полями ввода, где в первом поле ввода пользователь должен вести стоимостный коэффициент, а во втором поле ввода пользователь должен вести названия csv файла, расположенного в папке с программой. После ввода данных и проверки на их корректность. Дальше вызывается функция matrix\_window() для отрисовки второго окна интерфейса с матрицей. Полученные данные от пользователя идут в функцию process\_data(). Эта функция расположена в файле с логикой интерфейса. Возвращает она длину шага между вершинами графа, количество элементов для матрицы. Дальше вызывается класс FullScrollableMatrix() в него передается окно интерфейса, функция create\_np\_mas() для создания numpy матрицы, размеры сетки и название файла.

Класс строит матрицу. После постройки матрицы пользователь может выбирать ставить точки пути. Пользователь должен выбрать 2 точки, чтобы программа могла перейти к следующему шагу. Когда пользователь выбирает 2 точки в матрице, вызывается метод check\_red\_blocks(). Он блокирует взаимодействие с окном, пока будет происходить расчет пути. После блокировки окна вызывается функция a\_star(). Она расположена в файле с алгоритмами. В эту функцию передают координаты, выбранные пользователем. Алгоритм А\* ищет короткие путь от первой точки до второй точки. Если Алгоритм А\* сталкивается с элементом матрицы, который равен 1, то вызывается функция adaptive\_PCA(), в нее мы передаем всю numpy матрицу, минимальный размер для анализа, максимальный размер для анализа, минимальное кол-во точек воды для анализа. Функция возвращает вектор направления пути по воде.

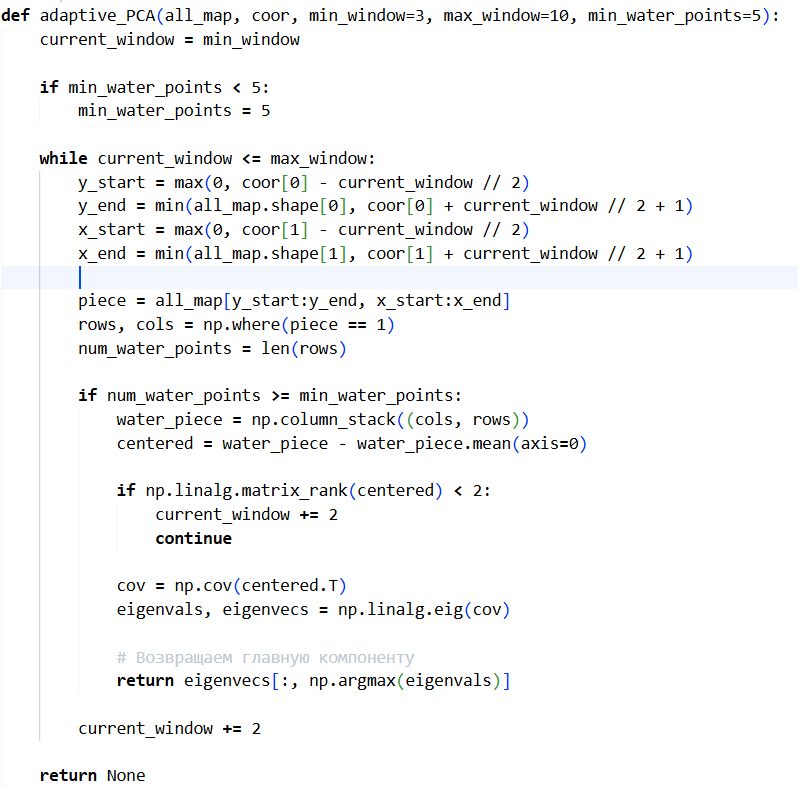


Рис 7. Функция adaptive\_PCA

Функция a\_star() работает до тех пор, пока не найдет вторую точку, выбранную пользователем. Как только находится эта точка. Функция формирует итоговый путь в лист из кортежей координат. Вызывается функция adj\_matrix(). Эта функция формирует csv файл с матрицей смежности и сохраняет его в папке с программой.

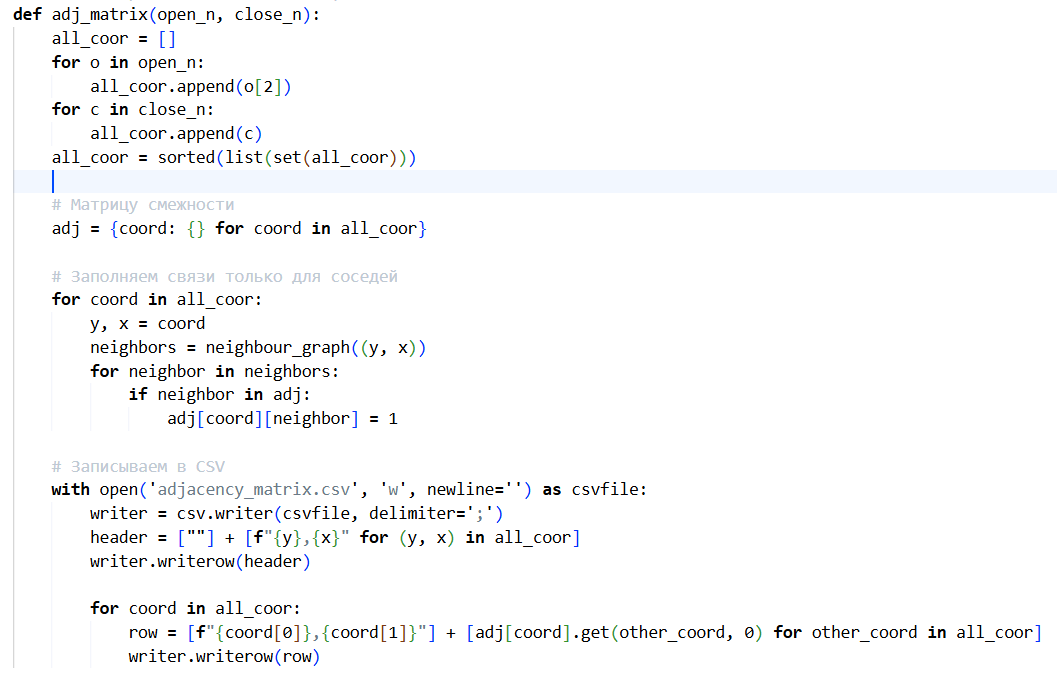


Рис 8. Функция adj\_matrix

Функция a\_star() возвращает лист из картежей координат от первой точки пользователя до второй точки пользователя, общую стоимость пути, возможные вершины в пути и время работы в секундах.

После из полученного листа с координатами строится красный путь на матрице в интерфейсе.

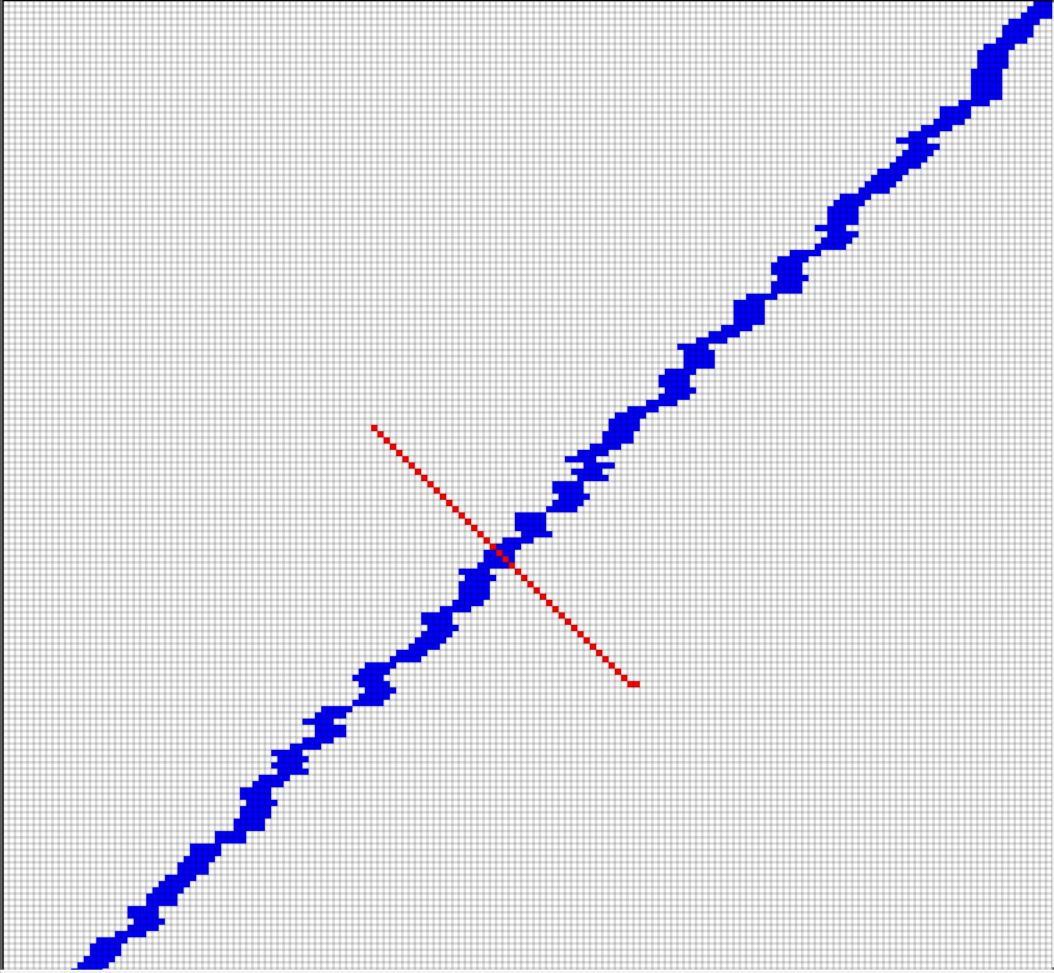


Рис 9. Пример отображения пути на матрице

# Заключение

В ходе практики была разработана программа для поиска оптимального маршрута с помощью графового алгоритма с логикой и интерфейсом.

Во время практики был изучен алгоритм РСА и алгоритм А\*. Также были изучены дополнительные методы библиотеки matplotlib.

Программа не идеальна и нуждается в доработке. Поэтому в дальнейшем планируется добавить распараллеливание для алгоритмов и доработать работу алгоритмов для “озер” в матрице.

# Список используемых источников

1. “Как работает метод главных компонент (PCA) на простом примере [habr] URL: [https://habr.com/ru/articles/304214/](%20%20https://habr.com/ru/articles/304214/) (дата обращения 21.06.2025)”
2. “ Введение в алгоритм A\* [habr] URL: <https://habr.com/ru/articles/331192/> (дата обращения 20.06.2025)”
3. “Алгоритм поиска пути A\* [skypro]

URL: <https://sky.pro/wiki/python/algoritm-poiska-puti-a/> (дата обращения 20.06.2025)”

# Приложение 1. ЕЖЕДНЕВНЫЕ ЗАПИСИ СТУДЕНТА ПО ПРАКТИКЕ

|  |  |  |
| --- | --- | --- |
| Дата | Описание работы, выполненной студентом | Отметка руководителя практики от организации |
| 16.06.2025 | Выбрал тему для практики. Обсудил задание с научным руководителем. Построил примерный план действий. Узнал, какие есть алгоритмы для быстрого поиска оптимального пути. |  |
| 17.06.2025 | Пришел на лекцию “Параллельное программирование”. Собрал простой интерфейс для программы. Написал функцию для нахождения шага сетки. |  |
| 18.06.2025 | Понял, как реализовать логику для интерфейса. Делал интерфейс. Столкнулся с проблемой, что tkinter не может обработать матрицу 800х800 блоков. Смотрел, как работает многопоточность, многопроцессорность и ассинхронность в python. |  |
| 19.06.2025 | Спросил у научного руководителя, как решать проблему с матрицей. Он подсказал использовать matplotlib для работы. Делал матрицу для интерфейса с использованием библиотеки matplotlib и tkinter. Писал первую главу к документации. |  |
| 20.06.2025 | Пришел на консультацию. Закончил основную работу над интерфейсом. Переписывал первую главу. Консультировался с научным руководителем. Подробно узнавал, как работают выбранные алгоритмы. |  |
| 21.06.2025 | Продолжал узнавать, как работают алгоритмы. Выбрал алгоритм А\*. Дописывал первую главу. Вручную решал задачи с алгоритмом А\*, чтобы лучше понять алгоритм. Изучал, как работает алгоритм PCA. |  |
| 23.06.2025 | Писал код для алгоритма А\*. Пришел на консультацию. Написал алгоритм А\*. Писал код для алгоритма PCA. |  |
| 24.06.2025 | Написал алгоритм PCA и встроил его в алгоритм А\*. Писал вторую главу к отчету. Собирал программу воедино. Начал писал функцию для печати смежной матрицы |  |
| 25.06.2025 | Сходил на консультацию. Оптимизировал код алгоритма. Делал тесты для “озер” и “рек”. Дописывал отчет. |  |
| 26.06.2025 | Сходил на консультацию. Доделал сохранение в csv файл. Записал видео работы программ. |  |
| 27.06.2025 | Защита практики. |  |

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

# Приложение 2. МНЕНИЕ СТУДЕНТА ПО ИТОГАМ ПРАКТИКИ И ЕГО ПРЕДЛОЖЕНИЯ.

За время практики я пережил семь стадий принятия.

Первая стадия — шок. Когда нам стали известны задачи, я впал в шок, потому что я не понимал, что нужно делать в них и как их реализовывать. Хочешь, не хочешь, но пришлось что-то выбрать. Я выбрал задачу “Быстрый алгоритм построения матрицы смежности для графового алгоритма”, потому что в голове было небольшое понимание, что делать нужно.

Вторая стадия — отрицание. Когда я думал, как я буду её реализовывать, появлялись мысли, что её решить невозможно. Собравшись с силами, я сел за ноутбук и начал писать код.

Третья стадия — гнев. Сначала я писал интерфейс и уже на нем был разгневан, потому что ничего не получалось.

Четвертая стадия — торг. Я пообещал, что если я сдам задачу и не пойду на пересдачу, то пойду в ресторан.

Пятая стадия — депрессия. Когда пришло время реализовывать алгоритмы, я ничего не понимал и это очень сильно било по морали.

Шестая стадия — испытание. После долгого разбора алгоритмов я понял что к чему, и после этого стало проще.

Седьмая стадия — принятие. После всех страданий код писался проще так, как появилось понимание, не идеальное, но хоть какое-то.

Благодаря практике я задумался, кем мне быть в it сфере и что для этого нужно предпринять.

Я считаю нужно давать задачи проще. А лучше просто реализовать хакатон, как в начале учебного года, между дизайнерами и разработчиками. Такой вход в профессиональную деятельность будет плавным. С моей стороны неопытного студента, практика на данный момент выглядит так, как будто нас выбросили на мороз после небольшой подготовки. Я не любитель такого подхода.