|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | |
| МИНОБРНАУКИ РОССИИ | | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **"МИРЭА - Российский технологический университет"**  **РТУ МИРЭА** | | | |
| **Институт искусственного интеллекта** | | |  |
| **Кафедра высшей математики** | | |  |

|  |  |
| --- | --- |
| **КУРСОВАЯ РАБОТА** | |
| **по дисциплине** | |
| **«**Алгоритмы и теория сложности**»** | |
| **Тема курсовой работы**  **«Алгоритм Дейкстры вычисления минимального расстояния»** | |
| Студент группы КМБО-03-22 | *…* |
| Руководитель курсовой работы | *Драгилева И.П.* |
|  |  |
|  |  |

|  |  |  |
| --- | --- | --- |
| Работа представлена к защите | «\_\_»\_\_\_\_\_\_\_2024 г. | *(подпись студента)* |
|  |  |  |
| «Допущен(ы) к защите» | «\_\_»\_\_\_\_\_\_\_2024 г. | *(подпись руководителя)* |

Москва – 2024

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **"МИРЭА - Российский технологический университет"**  **РТУ МИРЭА** | | | | **Институт искусственного интеллекта** | | | | **Кафедра высшей математики** | | | | | | |  |
|  | | **Утверждаю** | | |
|  | | Заведующий  кафедрой\_\_\_\_\_\_\_\_\_\_\_\_\_Шатина А.В. | | |
|  | | «\_2\_» \_декабря\_\_2024г. | | |
| **ЗАДАНИЕ** | | | | |
| **на выполнение курсовой работы** | | | | |
| **по** **дисциплине** «Алгоритмы и теория сложности» | | | | |
|  | | | | |
| Студент … Группа *КМБО-03-22.* | | | | |
|  | | | | |
| 1. **Тема: «Алгоритм Дейкстры вычисления минимального расстояния».** | | | | |
| **2. Исходные данные:** тестовые примеры и наборы данных для испытаний | | | | |
| **3**. **Перечень вопросов, подлежащих обработке, и обязательного графического материала:**  1) Алгоритм(ы) решения задачи на языке высокого уровня.  2) Получение и сравнение количественных результатов при различных исходных данных.  3) Асимптотические оценки временной сложности. | | | | |
|  | | | | |
| **4. Срок представления к защите курсовой работы:** **до** «20» декабря 2024 г. | | | | |
|  | | | | |
| Задание на курсовую  работу выдал | «1» октября 2024г. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | (*\_\_\_\_\_\_\_\_\_\_\_\_*) | |
| Задание на курсовую  работу получил | «\_\_\_»\_\_\_\_\_\_2024г. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | (\_\_\_\_\_\_\_\_\_\_\_\_) | |

**Оглавление**

[**Введение** 4](#_Toc183010217)

[**Постановка задачи** 5](#_Toc183010218)

[**Описание алгоритмов** 6](#_Toc183010219)

[**Доказательство корректности** 12](#_Toc183010220)

[**Асимптотическая оценка временной сложности алгоритмов** 14](#_Toc183010221)

[**Результаты запуска программы** 15](#_Toc183010222)

[**Заключение** 17](#_Toc183010223)

[**Список литературы** 18](#_Toc183010224)

[**Приложения** 19](#_Toc183010225)

## **Введение**

Алгоритмы поиска кратчайших путей играют важную роль в решении множества задач, связанных с графами. Одним из наиболее известных и часто используемых методов является алгоритм Дейкстры, предложенный Эдсгером Дейкстрой в 1956 году. Этот алгоритм находит кратчайший путь от одной вершины графа до всех остальных, если **веса рёбер неотрицательны**, что делает его важным инструментом в таких областях, как навигация, сетевое планирование и анализ транспортных систем.

Целью данной курсовой работы является изучение алгоритма Дейкстры и его реализаций. Будет рассмотрена наивная реализация алгоритма, основанная на последовательном переборе вершин, а также улучшенная версия с использованием структуры данных «двоичная куча», которая значительно ускоряет процесс выбора минимальной вершины. Эти подходы позволят продемонстрировать, как правильный выбор структуры данных может повлиять на эффективность алгоритма.

## **Постановка задачи**

***Дано:***

Взвешенный неориентированный граф

Любому ребру соответствует **неотрицательное** число (вес, длина, стоимость).

***Найти:*** пути из во все вершины графа так, чтобы длины этих путей были минимальны.

***Требуется:***

Реализовать задачу поиска кратчайших путей с помощью алгоритма Дейкстры, используя:

* Наивный подход к вычислению
* Двоичную кучу для вычисления

***Необходимо:***

*Проверить решение задачи для двух наборах данных:*

* Тестовый пример (неориентированный граф с 8 вершинами)
* Усложненный набор данных (неориентированный граф с 200 вершинами)

***Ответить на вопрос:*** Каковы кратчайшие расстояния от вершины 1 до следующих десяти вершин (7,37,59,82,99,115,133,165,188,197)?

*Оценить время выполнения алгоритма* и привести *асимптотическую оценку* временной сложности разработанного решения, обосновав её.

## **Описание алгоритмов**

* 1. ***Наивный алгоритм Дейкстры.***

Данный алгоритм вычисляет длины кратчайших путей из заданной вершины s до всех остальных в неориентированном взвешенном графе , вес рёбер которого неотрицателен и определяется весовой функцией , не используя никаких оптимизационных методов.

***Описание работы алгоритма:***

*Инициализация.* Создаётся массив расстояний , где для каждой вершины записывается текущее кратчайшее расстояние от начальной вершины (изначально расстояния для всех вершин равны бесконечности, кроме начальной вершины, для которой расстояние равно нулю) и массив *used*, который отслеживает, обработана ли вершина.

*Основной цикл.*

* На каждой итерации основного цикла, пока есть необработанные вершины, выбирается вершина , которой на текущий момент соответствует минимальная оценка кратчайшего пути , и она помечается, как обработанная.
* Для всех рёбер, исходящих из вершины , пытаемся улучшить кратчайшие расстояния до соседних вершин. Если текущее расстояние меньше предыдущего, то производим релаксацию всех исходящих из неё рёбер.

*Результат.*

После завершения алгоритма массив содержит кратчайшие расстояния от вершины

до каждой из остальных вершин. Если для какой-то вершины значение осталось равным , значит, она недостижима из начальной вершины.

|  |
| --- |
| **Dijkstra**  **Вход:** неориентированный граф представленный в виде списков смежности, вершина s , длина для каждого .  **Постусловие:** для каждой вершины значение равно истинному кратчайшему расстоянию  **Выход:** массив содержит кратчайшие расстояния от вершины до каждой из остальных вершин. |
| *func dijkstra(s):*  for              // найдём вершину с минимальным расстоянием            // произведём релаксацию по всем рёбрам, исходящим из v |

***Псевдокод 1.*** Псевдокод наивного алгоритма.

***Продемонстрируем пример выполнения алгоритма Дейкстры на тестовом наборе данных.***

Тестовый набор данных задан в формате:

1 2,1 8,2

2 1,1 3,1

3 2,1 4,1

4 3,1 5,1

5 4,1 6,1

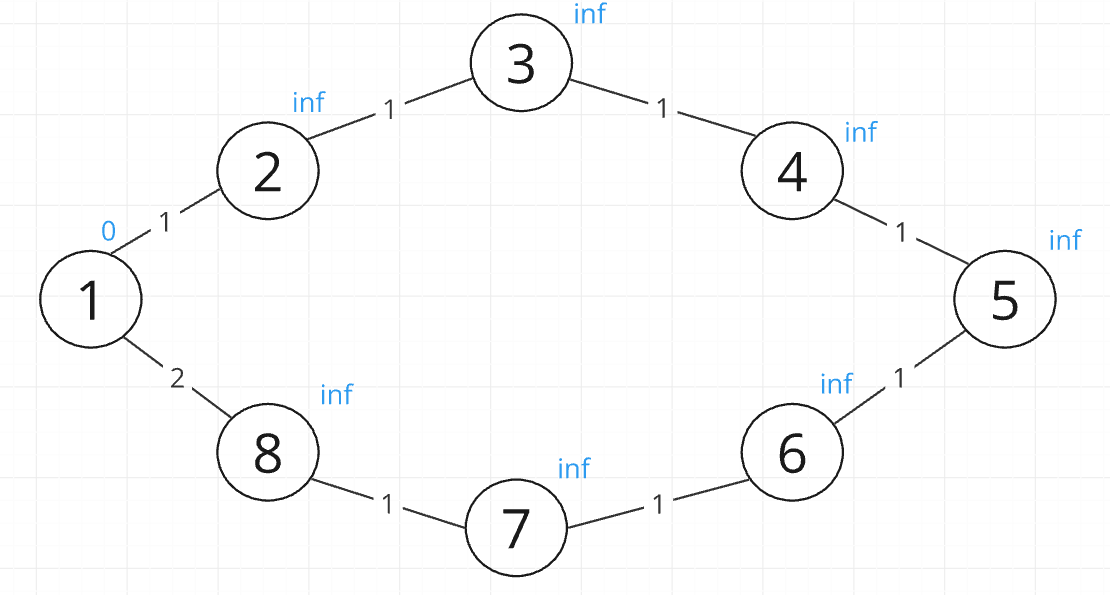
6 5,1 7,1

7 6,1 8,1

8 7,1 1,2

*В каждой строке указаны ребра, инцидентные данной вершине, а также их (неотрицательные) длины. Например, в шестой строке первой записью является «6», указывающая, что эта строка соответствует вершине 6. Следующая запись в этой строке «5,1» указывает, что между вершиной 6 и вершиной 5 существует ненаправленное ребро, имеющее длину 1. Вторая пара в этой строке указывает на вторую вершину, смежную с вершиной 6, и длину соответствующего ребра. Вершина 1 является начальной вершиной.*

Построим неориентированный граф, используя тестовые входные данные. На рисунке 1 синим шрифтом над вершинами написано кратчайшее расстояние до этой вершины в начале алгоритма; над ребрами написан вес каждого ребра.



***Рисунок 1.*** Взвешенный неориентированный граф на тестовых данных с кратчайшим расстоянием до каждой вершины от начальной (1).

Начинаем алгоритм с инициализации первой вершины.

На первом шаге (1 стрелка на рисунке 2) мы помечаем вершину 1, как отмеченную, и обновляем кратчайшие расстояния смежных вершин с 1, т.е. вершин 2 и 8, на веса ребер, ведущих к ним, т.е. на 1 и 2 соответственно.

На 2 шаге мы берем вершину с минимальным кратчайшим расстоянием (2) и обновляем кратчайшее расстояние смежной с ней вершиной (3), добавляя к кратчайшему расстоянию вершины 2 вес ребра между вершиной 2 и 3. И так далее.

Каждый шаг поиска алгоритмом Дейкстры на тестовом наборе данных изображен на рисунке 2.

|  |
| --- |
|  |

***Рисунок 2***. Пример выполнения алгоритма Дейкстры на тестовом наборе.

* 1. ***Алгоритм Дейкстры на двоичной куче.***

Алгоритм Дейкстры можно оптимизировать, если использовать для поиска кратчайших путей двоичную кучу. В библиотеках *Python* существует множество различных уже реализованных бинарных куч, однако, чтобы «жизнь мёдом не казалась», реализуем свой класс *BinaryHeap*, который будет задействован в нашем алгоритме.

***Описание класса BinaryHeap:***

*BinaryHeap* (двоичная куча) — это структура данных, представляющая собой почти полное бинарное дерево, хранящее элементы таким образом, чтобы минимальный (или максимальный) элемент всегда находился в корне.

Наша реализация использует массив для хранения дерева и поддерживает операции для работы с минимальной кучей.

*Основные компоненты и их задачи*

* *Хранение данных:*

*self.heap:* массив, где элементы хранятся в виде кортежей (вес, узел). Это позволяет сортировать узлы графа по их весам (ключам).

*self.positions:* словарь, где ключ — это узел графа, а значение — индекс этого узла в массиве *self.heap*. Это упрощает операции, такие как уменьшение ключа (*decrease\_key*).

* *Операции кучи:*

*Вставка (insert):* добавляет новый элемент в конец массива *heap* и выполняет операцию *sift\_up*, чтобы восстановить свойства кучи.

*Извлечение минимального (extract\_min):* удаляет корень кучи (минимальный элемент) и заменяет его последним элементом массива. Затем выполняется *sift\_down*, чтобы восстановить свойства кучи.

*Уменьшение ключа (decrease\_key):* обновляет значение веса для узла и выполняет *sift\_up*, чтобы переместить узел вверх до корректного положения.

*Проверка пустоты (is\_empty):* проверяет, есть ли элементы в куче.

* *Вспомогательные операции:*

*Просеивание вверх (sift\_up):* перемещает элемент вверх по дереву до тех пор, пока он не окажется на своем корректном месте (меньше родительского).

*Просеивание вниз (sift\_down):* перемещает элемент вниз по дереву, чтобы поддерживать свойства кучи.

*Обмен (swap):* меняет два элемента местами в массиве *heap* и обновляет их индексы в словаре *positions*.

***Описание работы алгоритма:***

*Инициализация.*

Создаем массив расстояний *dist*, в котором все расстояния инициализируются как бесконечные (), кроме исходной вершины (*src*), расстояние до которой равно 0.

Создаем объект двоичной кучи для хранения пар (расстояние, вершина) и добавляем исходную вершину с расстоянием 0.

*Основной цикл.*

1. Пока куча не пуста, извлекаем вершину с минимальным расстоянием, используя метод двоичной кучи – *extract\_min*.
2. Перебираются все вершины , соседние с (где ) и **обновляем минимальное расстояния до соседей:** если путь через до () короче текущего минимального расстояния , обновляем
3. Далее обновляем состояние в куче. Если вершина уже находится в куче, то вызываем метод *decrease\_key*, чтобы уменьшить ключ (расстояние) вершины до , иначе добавляем кучу с новым расстоянием .

*Результат.*

После завершения основного цикла массив *dist* содержит минимальные расстояния от исходной вершины *src* до всех остальных вершин графа.

|  |
| --- |
| **Dijkstra\_heap**  **Вход:** неориентированный граф представленный в виде списков смежности, вершина src , длина для каждого .  **Постусловие:** для каждой вершины значение равно истинному кратчайшему расстоянию  **Выход:** массив содержит кратчайшие расстояния от вершины до каждой из остальных вершин. |
| *func dijkstra\_heap(src):*              // найдём вершину с минимальным расстоянием      // произведём релаксацию по всем рёбрам, исходящим из u |

***Псевдокод 2.*** Псевдокод алгоритма Дейкстры на двоичной куче.

## **Доказательство корректности**

* 1. ***Наивный алгоритм Дейкстры.***

***Теорема.*** Пусть — ориентированный взвешенный граф, вес рёбер которого неотрицателен, — стартовая вершина. Тогда после выполнения алгоритма Дейкстры для всех , где — длина кратчайшего пути из вершины в вершину .

***Доказательство (через индукцию).***

Докажем по индукции, что в момент посещения любой вершины ,

*База индукции:* на первом шаге выбирается , для неё выполнено:

*Шаг индукции:*

Пусть для первых шагов алгоритм сработал верно и на шагу выбрана вершина . Докажем, что в этот момент .

Для начала отметим, что для любой вершины v, всегда выполняется (алгоритм не может найти путь короче, чем кратчайший из всех существующих).

Пусть *P* — кратчайший путь из *s* в *u*,

— первая непосещённая вершина на *P*,

*z* — предшествующая ей (следовательно, посещённая).

Поскольку путь *P* кратчайший, его часть, ведущая из *s* через *z* в *v*, тоже кратчайшая, следовательно

По предположению индукции, в момент посещения вершины *z* выполнялось следовательно, вершина *v* тогда получила метку не больше чем следовательно, . С другой стороны, поскольку сейчас мы выбрали вершину *u*, её метка минимальна среди непосещённых, то есть где второе неравенсто верно из-за ранее упомянутого определения вершины *v* в качестве первой непосещённой вершины на *P*, то есть вес пути до промежуточной вершины не превосходит веса пути до конечной вершины вследствие неотрицательности весовой функции. Комбинируя это с имеем , **что и** **требовалось доказать**.

Поскольку алгоритм заканчивает работу, когда все вершины посещены, в этот момент для всех *u*.

* 1. ***Алгоритм Дейкстры на двоичной куче.***

Корректность алгоритма Дейкстры на двоичной куче доказывается подобным образом, как и корректность наивного алгоритма.

***Теорема.*** Пусть — ориентированный взвешенный граф, вес рёбер которого неотрицателен, — стартовая вершина. Тогда после выполнения алгоритма Дейкстры с использованием двоичной кучи для всех , где — длина кратчайшего пути из вершины в вершину .

***Доказательство (через индукцию).***

Докажем по математической индукции, что на момент извлечения каждой вершины из кучи

*База индукции:*

1. На первом шаге из кучи извлекается начальная вершина с минимальным расстоянием .
2. , так как длина кратчайшего пути из вершины в саму себя равна .

*Следовательно, для вершины выполняется*

*Шаг индукции:*

Предположим, что для первых *n* шагов алгоритма утверждение верно: для каждой извлечённой вершины 𝑣 из кучи выполнено . Докажем, что оно остаётся верным на -м шаге, при извлечении очередной вершины 𝑢.

Пусть *P* — кратчайший путь из *s* в *u*,

— первая непосещённая вершина на *P*,

*z* — предшествующая ей (следовательно, посещённая).

1. По определению кратчайшего пути , его часть от 𝑠 до 𝑣 также является кратчайшей. Следовательно:
2. По предположению индукции, на момент посещения вершины z было выполнено Тогда в этот момент:

Таким образом, для 𝑣:

1. Теперь рассмотрим текущую вершину 𝑢, которая извлекается из кучи:

* 𝑢 выбрана из кучи с минимальным значением
* Это означает, что среди всех непосещённых вершин метка минимальна. Следовательно:

1. Комбинируя неравенства, получаем:
2. С другой стороны, алгоритм не переоценивает длину пути (это следует из неотрицательности весов рёбер). Поэтому:
3. Следовательно,

Таким образом, алгоритм корректно вычисляет кратчайшие пути из начальной вершины sss до всех остальных вершин графа, ч.т.д..

## **Асимптотическая оценка временной сложности алгоритмов**

В реализации алгоритма присутствует функция выбора вершины с минимальным значением и релаксация по всем рёбрам для данной вершины. Асимптотика работы зависит от реализации.

Пусть  — количество вершин в графе,  — количество рёбер в графе. На таблице 2 представлены асимптотические оценки временной сложности алгоритмов, с помощью которой можно их сравнить, приведены обоснования к ним.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Алгоритм | Время работы | | | Обоснование |
| Поиск минимума | Релаксация | Общее |
| Наивная реализация |  |  |  | Осуществляем поиск вершины раз с минимальной величиной среди непомеченных вершин и m раз проводим релаксацию за Для плотных графов () данная асимптотика является оптимальной. |
| Двоичная куча |  |  |  | Используя двоичную кучу можно выполнять операции извлечения минимума и обновления элемента за Тогда время работы алгоритма Дейкстры составит |

**Таблица 2.** Асимптотическая оценка сложности алгоритмов Дейкстры: наивной реализации и реализации на двоичной куче.

## **Результаты запуска программы**

Перейдем к оценке работы этих алгоритмов на тестовой и усложненной выборках и сравним время их выполнения.

*В тестовом наборе данных:* неориентированный граф с 8 вершинами.

*В усложненном наборе данных:* неориентированный граф с 200 вершинами.

В рамках реализации алгоритмов Дейкстры для удобства обработки графа из исходных данных создается матрица смежности.

*Матрица смежности* — это представление графа в виде двумерного массива, где элемент (𝑖, 𝑗) обозначает вес ребра между вершинами 𝑖 и 𝑗. Если ребро отсутствует, значение элемента равно нулю (пример преобразования входных данных можно увидеть в таблице 1 на примере тестового файла).

В таблице 2 показано на примере тестового файла, как преобразуются входные данные в матрицу смежности.

|  |  |
| --- | --- |
| Формат входных данных тестового файла | Соответствующая матрица смежности |
| 1 2,1 8,2  2 1,1 3,1  3 2,1 4,1  4 3,1 5,1  5 4,1 6,1  6 5,1 7,1  7 6,1 8,1  8 7,1 1,2 | [[0, 1, 0, 0, 0, 0, 0, 2],  [1, 0, 1, 0, 0, 0, 0, 0],  [0, 1, 0, 1, 0, 0, 0, 0],  [0, 0, 1, 0, 1, 0, 0, 0],  [0, 0, 0, 1, 0, 1, 0, 0],  [0, 0, 0, 0, 1, 0, 1, 0],  [0, 0, 0, 0, 0, 1, 0, 1],  [2, 0, 0, 0, 0, 0, 1, 0]] |

***Таблица 2.*** Преобразование входных данных тестового файла в матрицу смежности.

В таблице 3 приведено сравнение описанных алгоритмов по времени выполнения.

|  |  |  |
| --- | --- | --- |
| Алгоритм | Время выполнения на тестовом наборе данных | Время выполнения на усложненном наборе данных |
| Наивная реализация | < 0.00001 с. | 0.0109 с. |
| Двоичная куча | < 0.00001 с. | 0.0013 c. |

***Таблица 3.*** Сравнение алгоритмов Дейкстры для вычисления кратчайших расстояний от заданной вершины до остальных в неориентированном графе.

Оба алгоритма справились со своей задачей и дали такие ответы:

* На тестовом наборе данных: [0, 1, 2, 3, 4, 4, 3, 2]
* На усложненном наборе данных: [0, 2971, 2644, 3056, 2525, 2818, 2599, 1875, 745, 3205, 1551, 2906, 2394, 1803, 2942, 1837, 3111, 2284, 1044, 2351, 3630, 4028, 2650, 3653, 2249, 2150, 1222, 2090, 3540, 2303, 3455, 3004, 2551, 2656, 998, 2236, 2610, 3548, 1851, 4091, 2732, 2040, 3312, 2142, 3438, 2937, 2979, 2757, 2437, 3152, 2503, 2817, 2420, 3369, 2862, 2609, 2857, 3668, 2947, 2592, 1676, 2573, 2498, 2047, 826, 3393, 2535, 4636, 3650, 743, 1265, 1539, 3007, 4286, 2720, 3220, 2298, 2795, 2806, 982, 2976, 2052, 3997, 2656, 1193, 2461, 1608, 3046, 3261, 2018, 2786, 647, 3542, 3415, 2186, 2398, 4248, 3515, 2367, 2970, 3536, 2478, 1826, 2551, 3368, 2303, 2540, 1169, 3140, 2317, 2535, 1759, 1899, 508, 2399, 3513, 2597, 2176, 1090, 2328, 2818, 1306, 2805, 2057, 2618, 1694, 3285, 1203, 676, 1820, 1445, 2468, 2029, 1257, 1533, 2417, 3599, 2494, 4101, 546, 1889, 2616, 2141, 2359, 648, 2682, 3464, 2873, 3109, 2183, 4159, 1832, 2080, 1831, 2001, 3013, 2143, 1376, 1627, 2403, 4772, 2556, 2124, 1693, 2442, 3814, 2630, 2038, 2776, 1365, 3929, 1990, 2069, 3558, 1432, 2279, 3829, 2435, 3691, 3027, 2345, 3807, 2145, 2703, 2884, 3806, 1151, 2505, 2340, 2596, 4123, 1737, 3136, 1073, 1707, 2417, 3068, 1724, 815, 2060]

***Также необходимо было ответить на вопрос:*** Каковы кратчайшие расстояния от вершины 1 до следующих десяти вершин (7,37,59,82,99,115,133,165,188,197)?

***Ответ на данный вопроc:***

* Минимальное расстояние до вершины 7: 2599
* Минимальное расстояние до вершины 37: 2610
* Минимальное расстояние до вершины 59: 2947
* Минимальное расстояние до вершины 82: 2052
* Минимальное расстояние до вершины 99: 2367
* Минимальное расстояние до вершины 115: 2399
* Минимальное расстояние до вершины 133: 2029
* Минимальное расстояние до вершины 165: 2442
* Минимальное расстояние до вершины 188: 2505
* Минимальное расстояние до вершины 197: 3068

## **Заключение**

В ходе курсовой работы были рассмотрены 2 алгоритма решения задачи о поиске кратчайших расстояний от начальной вершины до остальных во взвешенном неориентированном графе:

* Наивный алгоритм Дейкстры (сложность )
* Алгоритм Дейкстры на двоичной куче (сложность ).

Мной были доказаны корректности данных алгоритмов и выведена их асимптотическая оценка.

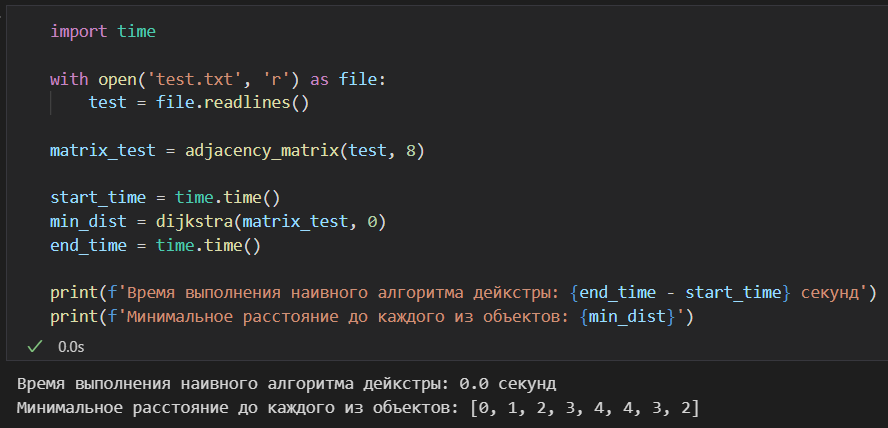
Алгоритмы были протестированы на тестовом и усложненном наборе данных, где лучшим по времени выполнения оказался ***алгоритм Дейкстры на двоичной куче,*** выполнив задачу на усложненном за 0.0013 c..

***Алгоритм Дейкстры на двоичной куче превзошел наивный алгоритм Дейкстры примерно в 10 раз.***

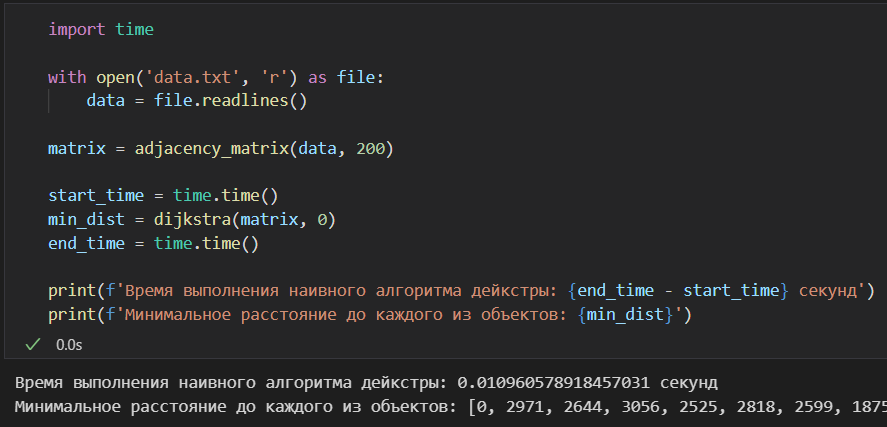
## **Список литературы**

1. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих. - 3-e изд. - СПб: Питер, 2024. - 288 с.
2. Рафгарден Т. Совершенный алгоритм. Основы. - 2-e изд. - СПб: Питер, 2023. - 256 с.
3. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн Алгоритмы: построение и анализ — 2-е изд. — М.: «Вильямс», 2007. — с. 459. — ISBN 5-8489-0857-4
4. Скиена С.С. Алгоритмы. Руководство по разработке. - 2-е изд. - М.: БХБ, 2018. - 720 с.

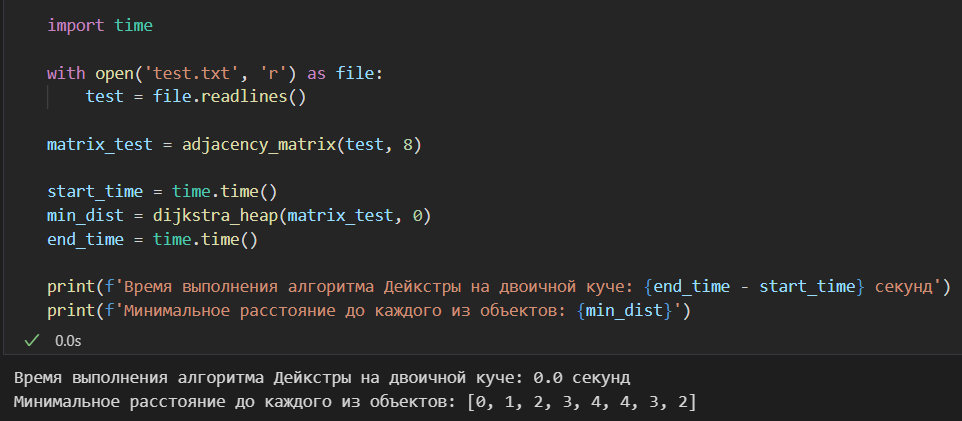
## **Приложения**



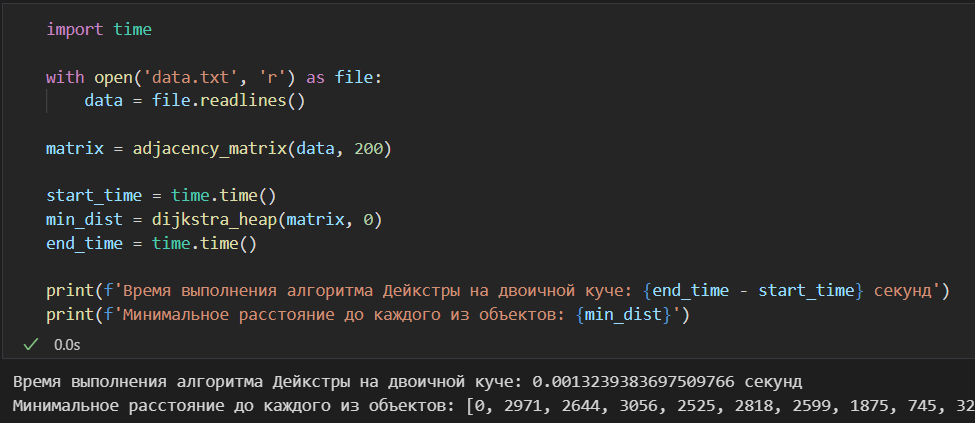
**Рисунок 3.** Результат и время выполнения наивного алгоритма Дейкстры на тестовых данных.



**Рисунок 4.** Результат и время выполнения наивного алгоритма Дейкстры на усложненных данных.



**Рисунок 5.** Результат и время выполнения алгоритма Дейкстры на двоичной куче на тестовых данных.



**Рисунок 6.** Результат и время выполнения алгоритма Дейкстры на двоичной куче на усложненных данных.