Министерство образования и науки РФ

Федеральное государственное автономное образовательное учреждение высшего образования

«Омский государственный технический университет»

|  |  |
| --- | --- |
| Факультет (институт) | *Информационных технологий и компьютерных систем* |
|  |  |
| Кафедра | *Прикладная математика и фундаментальная информатика* |
|  |  |

**Расчетно-графическая работа**

|  |  |
| --- | --- |
| по дисциплине | ***Дискретная математика*** |
|  |  |
| на тему | Реализация алгоритма поиска кратчайшего пути |

Пояснительная записка

|  |  |
| --- | --- |
| **Шифр проекта** | 020-РГР-02.03.02-№ 3-ПЗ |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Студента** | | Буряк Варвары Николаевны | | | | | |
|  |  |  |  | | фамилия, имя, отчество полностью | | | | | |
|  |  |  | Курс | *1* |  | Группа | | ФИТ-232 | | |
|  |  |  |  |  |  | |  |  | |  |
|  | | | **Направление (специальность)** | | | | | ***02.03.02*** | | |
|  | | | *Фундаментальная информатика и информационные технологии* | | | | | | | |
|  |  |  | код, наименование | | | | | | | |
|  |  |  | Руководитель | | ***ст. преподаватель*** | | | | | |
|  |  |  | ученая степень, звание | | | | | |
|  |  |  | ***Федотова И.В.*** | | | | | | | |
|  |  |  | фамилия, инициалы | | | | | | | |
|  |  |  | Выполнил | | 22.05.2024 | | | | | |
|  |  |  | дата, подпись студента | | | | | |
|  |  |  | **Работа защищена с количеством баллов** | | | | | | | |
|  |  |  |  | | | | | |  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | дата, подпись руководителя |  |  |  |

Омск 2024

**Содержа**ние

[Введение 3](#_Toc22848)

Постановка задачи 4

[Теоретическая часть 5](#_Toc32058)

[Разработка кода 6](#_Toc16304)

[Результаты 8](#_Toc2637)

[Заключение 9](#_Toc23786)

[Список используемой литературы 1](#_Toc21612)0

# Введение

Дискретная математика играет важную роль в развитии и применении современных алгоритмов программирования. Её методы и теории находят широкое применение в разработке программного обеспечения, обеспечивая эффективность и надёжность вычислительных процессов. В этой расчётно-графической работе будет рассматриваться основной алгоритм дискретной математики, алгоритм Дейкстры, который позволяет найти кратчайший путь.

В данной работе этот алгоритм реализован на языке программирования Python.

Целью расчётно-графической работы является демонстрация применения алгоритмов дискретной математики для решения практических задач в программировании, что подчёркивает их значимость и универсальность в области информационных технологий.

# Постановка задачи

Фирме, осуществляющей перевозку скоропортящегося товара, дано задание на доставку товара из Ставрополя в Будённовск, при этом существует несколько путей, по которым возможно доставить товар. Расстояние между городом Ставрополь и селом К. – 26 километров, между г. Ставрополем и селом П. – 19 километров, между г. Ставрополем и селом Р. – 86 километров. Между сёлами К. и Д. – 16 километров, между сёлами К. и Л. – 66 километров. Между селом П. и городом Н. составляет 4 километра, между сёлами П. и В. – 51 километр. Между сёлами Д. и В. - 21 километр. Между городом Н. и селом М. – 21 километр. Между сёлами М. и Л. – 24 километра, между сёлами М. и В. – 34 километра. Между сёлами Л. и А. – 13 километров, между сёлами Л. и Ж. – 43 километра. Между сёлами А. и Б. 25 километров. Между сёлами Ж. и Р. – 31 километр, между сёлами Ж. и Б. – 44 километра. Между сёлами Б. и Р. 22 километра. Между сёлами В. и Ж. – 9 километров.

Необходимо найти самый короткий путь из Ставрополя в Буденновск.

**Формат выходных данных**

На экран выведите одно число – суммарную длина маршрута или –1, если добраться невозможно.

# Теоретическая часть

Алгоритм Дейкстры используется для нахождения кратчайшего пути от одной вершины до всех других вершин в графе с ненегативными весами рёбер.

**Математическое описание алгоритма**

Пусть задан граф с весами рёбер и выделенной вершиной-источником . Обозначим через кратчайшее расстояние от источника до вершины .

Пусть уже вычислены все расстояния, не превосходящие некоторого числа , то есть расстояния до вершин из множества . Пусть . Тогда , и лежит на кратчайшем пути от к .

Величины , где , называются *предполагаемыми расстояниями* и являются оценкой сверху для настоящих расстояний: .

Алгоритм Дейкстры на каждом шаге находит вершину с наименьшим предполагаемым расстоянием, помечает её как посещённую и обновляет предполагаемые расстояния для всех концов рёбер, исходящих из неё.

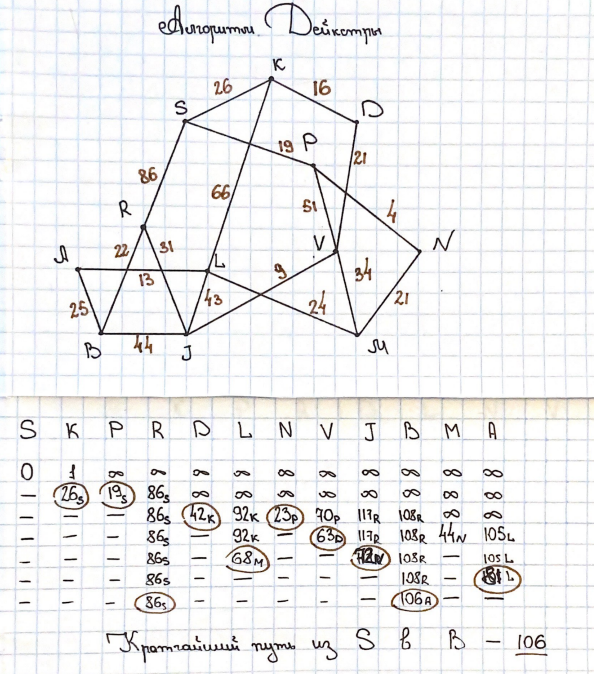


Рисунок 1 – Ручной подсчёт алгоритма

# Разработка кода

Этот код реализует программу, которая использует алгоритм Дейкстры для нахождения кратчайшего пути в графе. В графе описаны расстояния между городами. Программа предлагает пользователю несколько опций в главном меню: вычисление, об авторе программы, описание постановки задачи и выход. Пользователь выбирает пункт меню, после чего выполняется соответствующая функция.

import **heapq**

graph = {

    'С.': {'К.': 26, 'П.': 19, 'Р.': 86},

    'К.': {'Д.': 16, 'Л.': 66},

    'П.': {'С.': 19, 'Н.': 4, 'В.': 51},

    'Р.': {'Ж.': 31, 'Б.': 22},

    'Д.': {'В.': 21, 'К.': 16},

    'Л.': {'М.': 24, 'А.': 13, 'Ж.': 43, 'К.': 66},

    'Н.': {'М.': 21, 'П.': 4},

    'В.': {'М.': 34, 'Ж.': 9, 'Д.': 21, 'П.': 51},

    'Ж.': {'Б.': 44, 'Р.': 31, 'Л.': 43, 'В.': 9},

    'Б.': {'А.': 25, 'Р.': 22, 'Ж.': 44},

    'М.': {'Л.': 24, 'В.': 34, 'Н.': 21},

    'А.': {'Л.': 13, 'Б.': 25}

}

def **validate\_graph**(graph):

    for node, edges in graph.items():

        for neighbor, distance in edges.items():

            if not **isinstance**(distance, (**int**, **float**)) or distance <= 0:

**print**(f"Ошибка: некорректное расстояние от {node} до {neighbor}")

                return False

    return True

def **dijkstra**(graph, start, end):

    distances = {vertex: **float**('infinity') for vertex in graph}

    distances[start] = 0

    priority\_queue = [(0, start)]

    while priority\_queue:

        current\_distance, current\_vertex = **heapq**.**heappop**(priority\_queue)

        if current\_distance > distances[current\_vertex]:

            continue

        for neighbor, weight in graph[current\_vertex].items():

            if neighbor **not** in distances:

**print**(f"Ошибка: Вершина {neighbor} не найдена в графе.")

                return -1

            distance = current\_distance + weight

            if distance < distances[neighbor]:

                distances[neighbor] = distance

**heapq**.**heappush**(priority\_queue, (distance, neighbor))

    return distances[end] if distances[end] != **float**('infinity') else -1

def **about\_author**():

**print**("Программа разработана студенткой первого курса ОмГТУ группы ФИТ-232 Буряк Варварой Николаевной.")

def **problem\_description**():

**print**("Фирме, осуществляющей перевозку скоропортящегося товара, дано задание на доставку товара из Ставрополя в Будённовск. "

          "\nНеобходимо найти самый короткий путь с учетом расстояний между пунктами.")

def **main\_menu**():

    while True:

**print**("\nГлавное меню")

**print**("1. Вычисление")

**print**("2. Об авторе программы")

**print**("3. Описание постановки задачи")

**print**("4. Выход")

        choice = **input**("Выберите пункт меню: ")

        if choice == '1':

            if **validate\_graph**(graph):

                start = 'С.'

                end = 'Б.'

                shortest\_distance = **dijkstra**(graph, start, end)

                if shortest\_distance != -1:

**print**(f"Самый короткий путь из {start} в {end} составляет {shortest\_distance} километров.")

                else:

**print**(f"Невозможно добраться из {start} в {end}.")

            else:

**print**("Граф содержит ошибки.")

        elif choice == '2':

**about\_author**()

        elif choice == '3':

**problem\_description**()

        elif choice == '4':

**print**("Выход из программы.")

            break

        else:

**print**("Некорректный ввод. Пожалуйста, выберите пункт меню от 1 до 4.")

if \_\_name\_\_ == "\_\_main\_\_":

**main\_menu**()

# Результаты

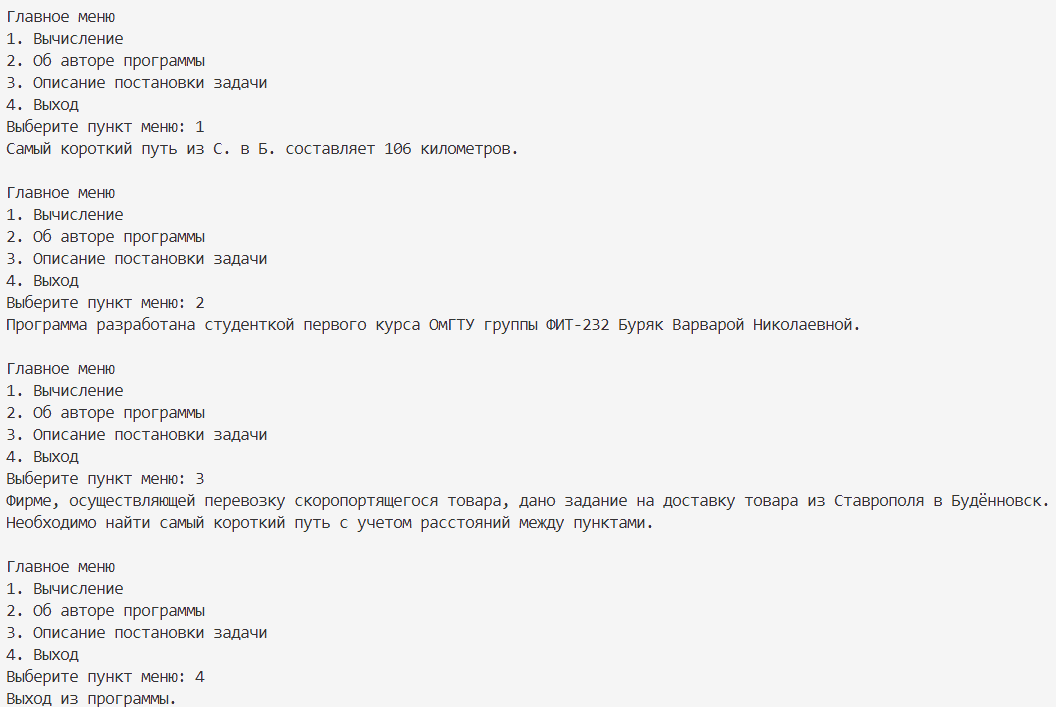


Рисунок 2 – Результат работы программы

# Заключение

В данной расчётно-графической работе был исследован алгоритм Дейкстры, применяемый для нахождения кратчайшего пути во взвешенном графе от одной вершины до всех остальных. Алгоритм показал эффективность и высокую точность в определении оптимального маршрута, основываясь на минимальной стоимости прохождения через вершины.

С помощью языка программирования Python была разработана программа, реализующая алгоритм Дейкстры. Полученные результаты подтвердили правильность работы алгоритма и его способность находить кратчайшие пути в графах различной сложности.

# Список используемой литературы

1. Алгоритм Дейкстры // Википедия URL: https://ru.wikipedia.org/wiki/Алгоритм\_Дейкстры (дата обращения: 22.05.2024).
2. Алгоритм Дейкстры: что это такое, как работает и где используется // Skillbox Media URL: https://skillbox.ru/media/code/algoritm-deykstry-chto-eto-takoe-kak-rabotaet-i-gde-ispolzuetsya/ (дата обращения: 22.05.2024).
3. Python 3.12.3 documentation // Python URL: https://docs.python.org/3/ (дата обращения: 22.05.2024).