Министерство образования Российской Федерации  
Пензенский государственный университет  
Кафедра «Вычислительная техника»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

К курсовому проектированию  
по курсу «Логика и основы алгоритмизации  
в инженерных задачах»

на тему «Реализация алгоритма Дейкстры»

Выполнил:

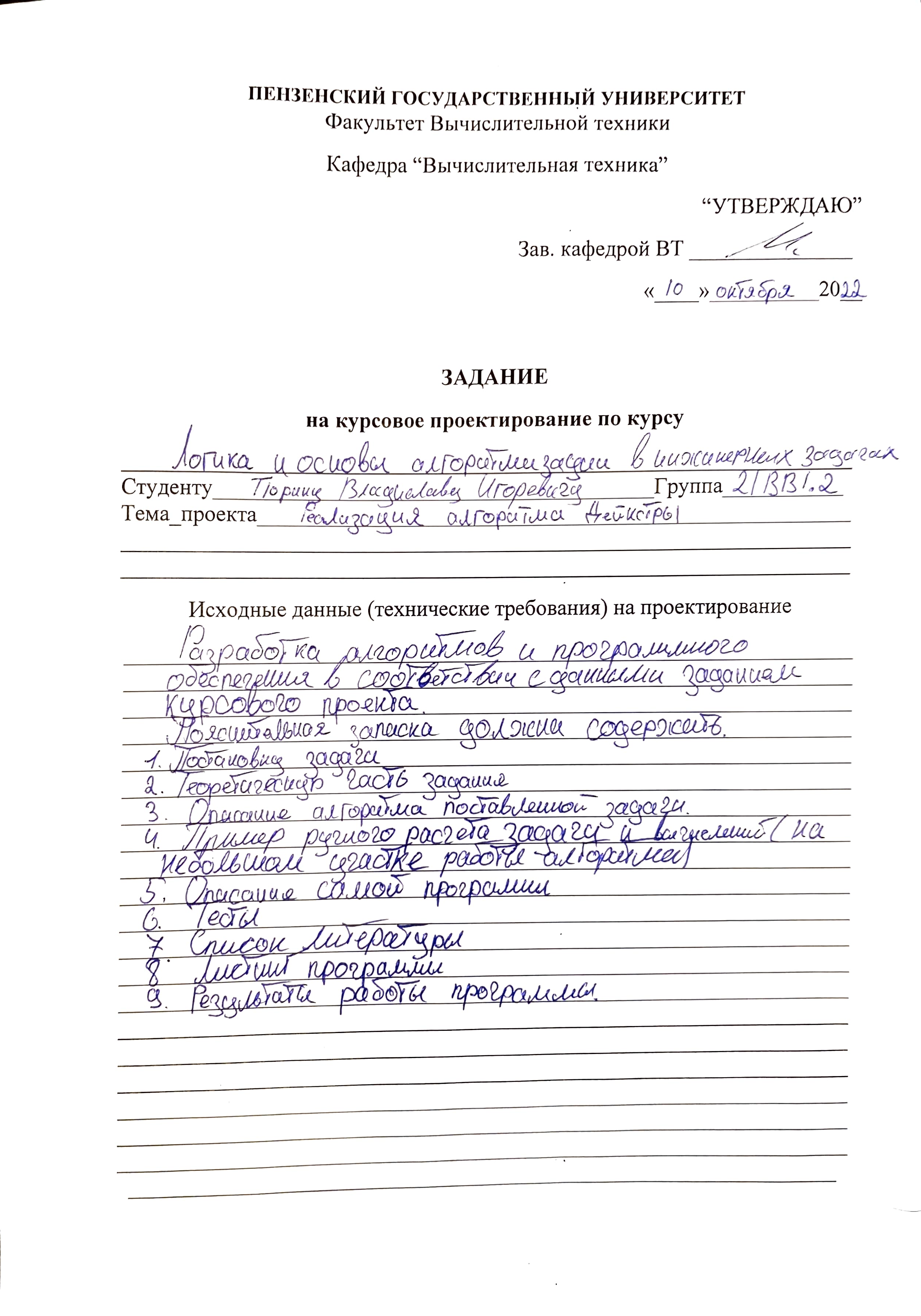
студент группы 21ВВ2

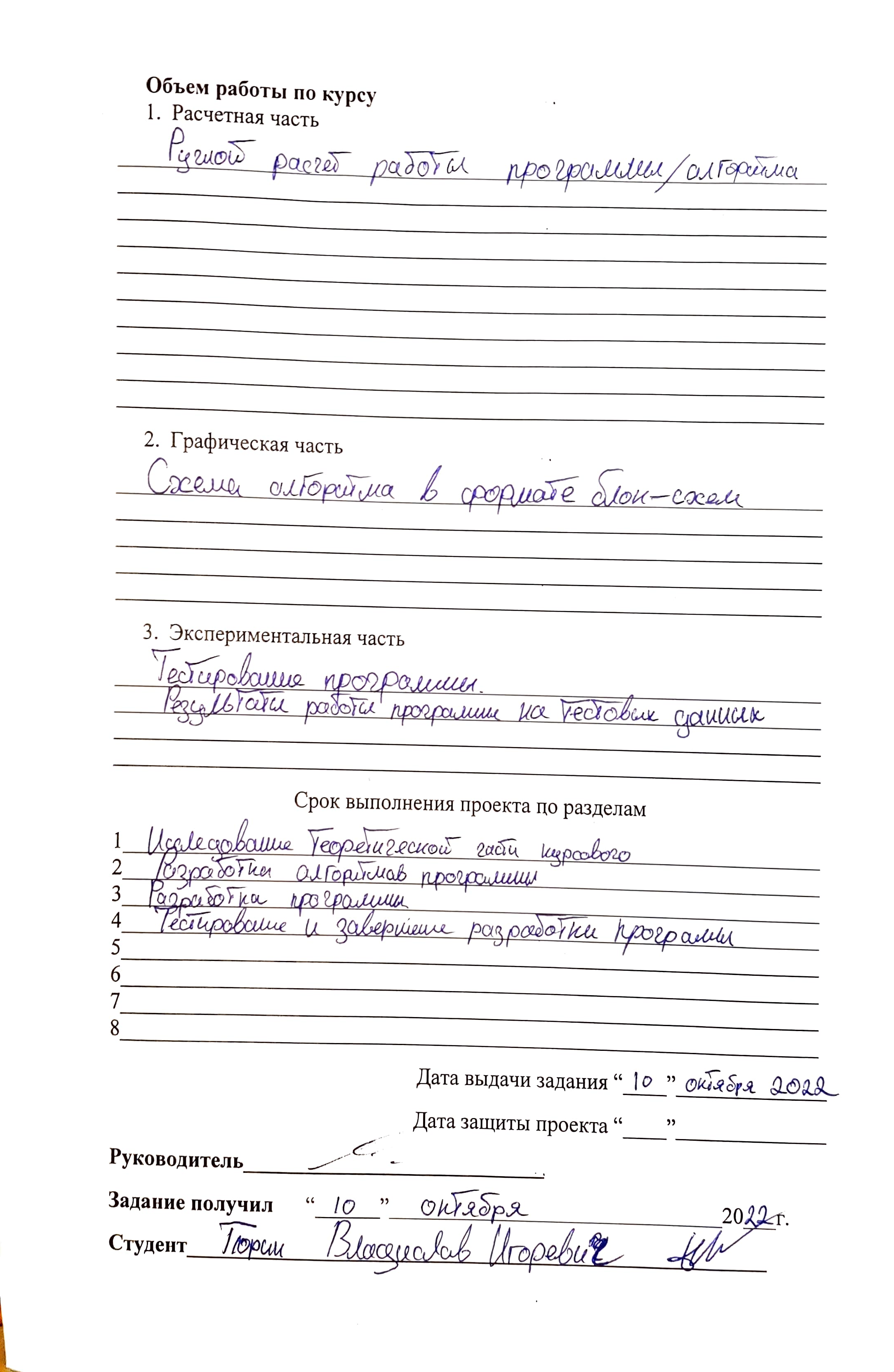
Тюрин В.И.

Принял:  
 Митрохин М.А.

Юрова О.В.

Пенза 2022





**Содержание**

[Введение 5](#_Toc121441839)

[Постановка задачи 6](#_Toc121441840)

[Теоретическая часть задания 7](#_Toc121441841)

[Описание алгоритма программы 8](#_Toc121441842)

[Описание программы 10](#_Toc121441843)

Введение

Алгоритм Дейкстры – алгоритм на графах, изобретенный нидерландским ученым Эдсгером Дейкстрой в 1959 году. Алгоритм находит кратчайшие пути от одной из вершин графа до всех остальных. Алгоритм работает только для графов без рёбер отрицательного веса. Алгоритм широко применяется в программировании, например, его используют протоколы маршрутизации OSPF и IS-IS.

В качестве среды разработки мною была выбрана среда PyCharm, язык программирования – Python.

Целью данной курсовой работы является разработка программы на языке Python, который является современным и широко используемым. Именно с его помощью в данной курсовом проекте реализуется алгоритм Дейкстры, осуществляющий поиск кратчайшего пути от одной вершины графа до всех остальных.

Постановка задачи

Требуется разработать программу, которая выявит кратчайшие пути от одной вершины графа до других, используя алгоритм Дейкстры.

Исходный граф в программе должен задаваться матрицей смежности, причем при генерации данных должны быть предусмотрены граничные условия. Программа должна работать так, чтобы пользователь вводил количество вершин для генерации матрицы смежности. После обработки данных на экран должна выводиться матрица смежности графа. Необходимо предусмотреть различные исходы работы алгоритма, чтобы программа не выдавала ошибок.

Устройство ввода – клавиатура и мышь.

Задания выполняются в соответствии с вариантом №13.

Теоретическая часть задания

Граф G (рисунок 1) задается множеством вершин X1, X2, ..., Xn. и множеством ребер, соединяющих между собой определенные вершины. Ребра из множества А ориентированы, что показывается линией, которая указывает достижимость данной вершины, граф с такими ребрами называется неориентированным.

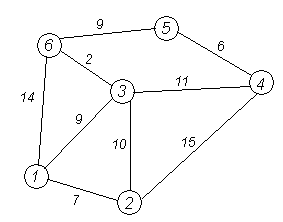


Рисунок 1 - Пример графа

При представлении графа матрицей смежности информация о ребрах графа хранится в квадратной матрице, где присутствие пути из одной вершины в другую обозначается единицей, иначе нулем. После этого по данной матрице строится еще одна матрица смежности, которая показывает вес дуги.

Каждой вершине из *V* сопоставим метку — минимальное известное расстояние от этой вершины до *a*. Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.

Описание алгоритма программы

Для программной реализации алгоритма понадобится один массив: T – массив, в который записываются минимальные пути. Словарь S, куда записываются посещенные вершины. Имеется граф G=(I, J). Каждая из вершин графа пока не занесена в словарь с посещенными вершинами.

В качестве исходного пункта выбирается вершина 1. В массиве T задается нулевое значение для начальной вершины. Пока не просмотрены все вершины выполняется цикл: с помощью функции getlink() и цикла for мы получаем все связанные вершины с текущей, и если вершина не записана в словарь S, то есть не посещена, то в переменную w записывается ее вес(сумма путей), если текущий вес меньше чем тот что уже записан, то в этом случае сохраняем новые значения связи. Далее выбирается вершина с минимальным весом с помощью функции arg\_min(), если данная вершина больше нуля, то мы ее добавляем во множество S. После завершения цикла в массиве T будут записаны кратчайшие пути от начальной вершины ко всем остальным.

Ниже представлен псевдокод алгоритма и функций getlink() и arg\_min().

getlink()

1. цикл: i и weight в enumerate(gg[v])
   1. если weigth>0, то
      1. Вернуть i

arg\_min()

1. amin = -1
2. m= максимальное значение из массива T
3. цикл: i и t в enumerate(T)
   1. если t < m & I не находится в S, то
      1. m = t
      2. amin = i
4. вернуть amin

Сам алгоритм

S = {v}

T[v] = 0

1. пока v не равно -1 делать:
   1. цикл: j в getlink(v,gg)
      1. Если j не в S, то
         1. w = T[v] + gg[v][j]
         2. если w < T[j], то
            1. T[j] = w
   2. v = arg\_min(T, S)
   3. Если v >= 0, то
      1. v добавить в S

Полный код можно увидеть в Приложении А.

Описание программы

Для написания данной программы использовался язык программирования Python. Это универсальный и современный язык программирования, который завоевал свою популярность у программистов, благодаря богатой стандартной библиотеке и комфортным порогом входа.

Проект был создан в виде консольного приложения.

Данная программа является многомодульной, поскольку состоит из нескольких функций: getlink, arg\_min, creatematrix, start, printm, print2file, print2secfile, printresult.

Работа начинается с выбора генерации графа. На выбор пользователю предлагается 2 варианта: Случайная генерация с вводом только количества вершин или прочитать матрицу смежности из файла, в который пользователь ввел ее заранее. Если пользователь выбрал случайную генерацию, то программа запросит ввести количество вершин, после чего программа сгенерирует граф заданного размера.

if wi==1 :  
 while True:  
 try:  
 n = int(input("Введите размер матрицы: "))  
 except:  
 print("Введите число")  
 else:  
 break  
 gg = creatematrix(n)  
 break

def creatematrix(value):  
 matrx = [[0] \* (value) for \_ in range(value)]  
 i = 0  
 while (i < len(matrx)):  
 j = i  
 while (j < len(matrx[i])):  
 if (i == j):  
 matrx[i][j] = 0  
 else:  
 matrx[i][j] = int((randint(0, 1)))  
 if (matrx[i][j] == 1):  
 matrx[i][j] = int((randint(0, 99)))  
 matrx[j][i] = matrx[i][j]  
 j += 1  
 i += 1  
 return matrx

Далее выводятся элементы массива, а именно матрица смежности.