Лабораторная работа №7 по дисциплине «Типы и структуры данных»

Графы

Выполнил: Елгин Илья

Группа: ИУ7-34Б

**Задание**

Обработать графовую структуру в соответствии с указанным вариантом задания. Обосновать выбор необходимого алгоритма и выбор структуры для представления графов. Ввод данных – на усмотрение программиста. Результат выдать в графической форме.

В системе двусторонних дорог за проезд каждой дороги взимается некоторая пошлина. Найти путь из города A в город B с минимальной величиной S+P, где S - сумма длин дорог пути, а P - сумма пошлин проезжаемых дорог

**Входные данные**

Программа считывает данные о графе из текстового файла. Граф задается количеством вершин и двумя матрицами смежности, в первой указан вес ребер между вершинами (в ячейке (0, 1) вес ребра, связывающего 0 и 1 вершину), во второй матрице указана стоимость прохождение по ребру. Нулевые считаются отсутствующими. Матрица должна быть квадратной.

Выбор пункта меню:

МЕНЮ:

1) Показать граф

2) Показать входные данные

3) Найти путь с минимальным S+P

0) Выйти

Вершины между которыми нужно найти кратчайший путь

**Выходные данные**

Граф в виде PNG

Время работы

Память

Путь в виде последовательности вершин

Сумма S пути

Сумма P пути

**Аварийные ситуации**

* При неверных входных данных в файле (меньше данных чем нужно, больше данных чем нужно, не квадратные матрицы, отрицательное кол-во вершин, стоимость отличная от нуля при нулевом ребре. Программа выведет сообщение: «Содержимое файла некорректно!» и завершится с кодом ошибки.
* Попытка ввода несуществующего пункта меню. Программа выведет: «Нет данного пункта меню!».

**Внутренняя структура данных**

**typedef struct matrix\_t**

**{**

**unsigned int size;//Размер квадратной матрицы (количество вершин в графе)**

**int \*\*data;// матрица P -дорог**

**int \*\*cost;// матрица S -дорог**

**}matrix\_t;**

Граф представлен в виде матрицы смежности размера n \* n, где в (i, j) ячейке хранится 0, если ребра между вершинами нет (либо нет петли у вершины, если i = j ), и вес ребра в противном случае. Матрица смежности является более удобным способом хранения данных при обработке и заполнении. Недостатком выбранной реализации является большое количество требуемой памяти – хранятся также нули, которых при другой реализации можно выкинуть.

Структура для хранения графа в стандартном виде:

Struct graph

{

Int n;// номер вершины

Int \*cost;//массив весов рёбер

Struct graph \*\*g;// массив ссылок на вершины куда ведут ребра из данной вершины

}

Минусом данной реализации является необходимость проверки связности графа.

Также граф можно хранить как массив списков, что позволит избежать выделения памяти на пустые вершины и времени на обработку их.

**Функции**

int deiksrt(const matrix\_t \*matrix, int ifrom, int ito, int ver[], int \*k);

Вычисляет кратчайший путь между двумя вершинами в графе по методу декстры

Входные данные const matrix\_t \*matrix – граф, int ifrom, int ito – наяальная и конечная вершины, int ver[], int \*k – массив для записи пути и количество элементов в нём

Возвращает значение существует ли путь, выводит на экран путь и его стоимость, записывает путь в массив.

matrix\_t \*read\_matrix(FILE \*f);

Считывает граф в виде матриц из файла

Входные данные FILE \*f – файл.

Возвращает ссылку на граф.

void print\_matrix(const matrix\_t \*mtr);

Выводит граф в виде матриц.

Входные данные const matrix\_t \*mtr – граф.

Выводит на экран матрицы

void output\_graph\_to\_png(const matrix\_t \*a);

Выводит граф в виде png

Входные данные const matrix\_t \*a – граф в виде матриц.

Создаёт png графа.

void output\_got\_to\_png(const matrix\_t \*a, int mat[], int k\_mat, int ito);

Выводит граф в виде png с помеченным путём.

Входные данные const matrix\_t \*a – граф в виде матриц, , int mat[] , int k\_mat – матрица вершин пути и их количество, int ito – начальная точка пути.

.

Создаёт png графа с помеченным путём.

**Алгоритм**

Поиск кратчайшего пути алгоритмом Дейкстры:

Создание массива S кратчайших путей от заданной начальной вершины до вершины индекса массива.

Создание массива P кратчайших путей от заданной начальной вершины до вершины данного индекса массива.

Создание массива списков для хранения кратчайших путей, состоящих из вершин от заданной начальной вершины.

В начале очередная вершина-это начальная вершина, далее она выбирается как вершина наименьшим S+P (S+P не = 0) из массивов, которая ещё не была выбрана, выбор продолжается до тех пор, пока очередная вершина не является конечной вершиной или подходящих вершин не осталось.

Для очередной вершины

для индексов, связанных с ней вершин (которые ещё не были выбраны)

если S+P выбранной вершины + S+P пути от выбранной вершины до данной вершины < S+P в массиве данной вершины или S+P данной вершины = 0 то

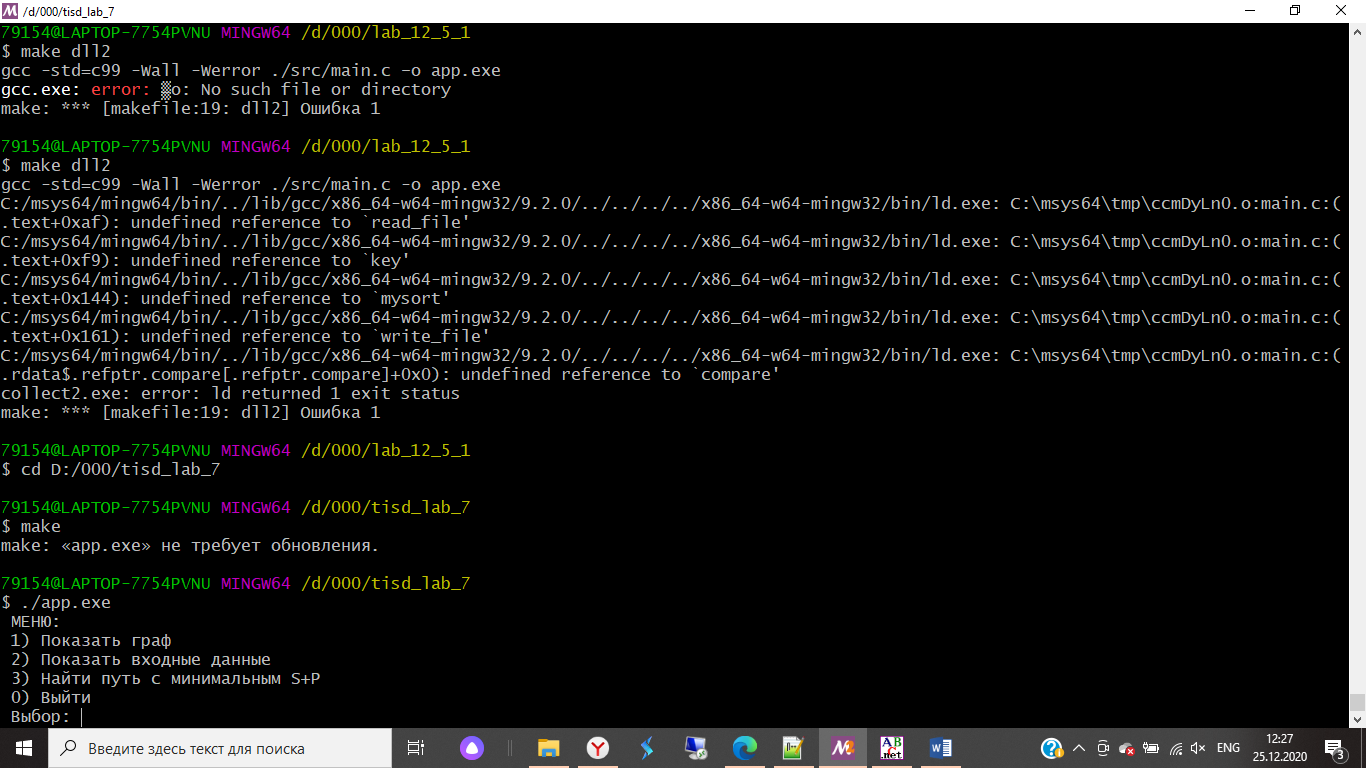
в массивы вносим S и P выбранной вершины + S и P пути от выбранной вершины

путь копируем из выбранной вершины добавляя в него саму выбранную вершину

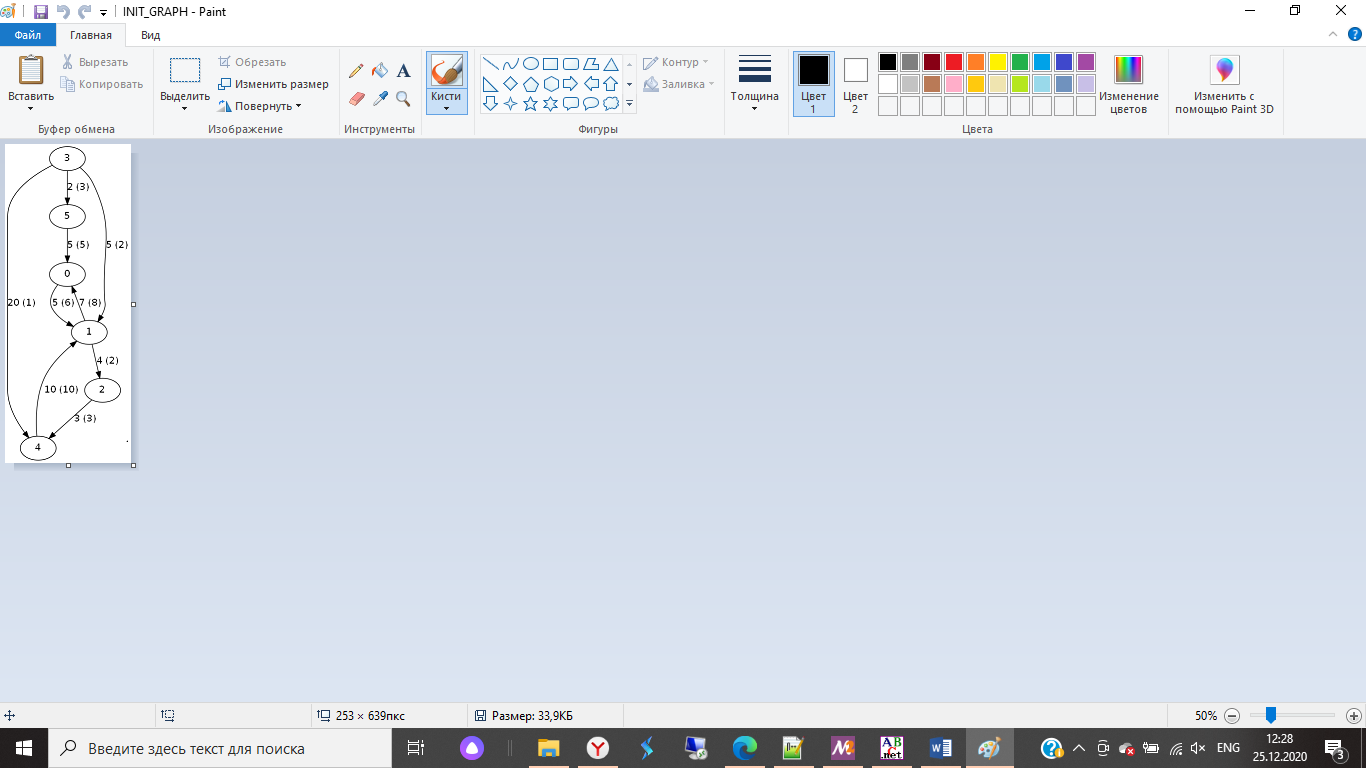
Если S или P для конечной точки = 0 то пути до данной точки не существует.

**Работа программы**

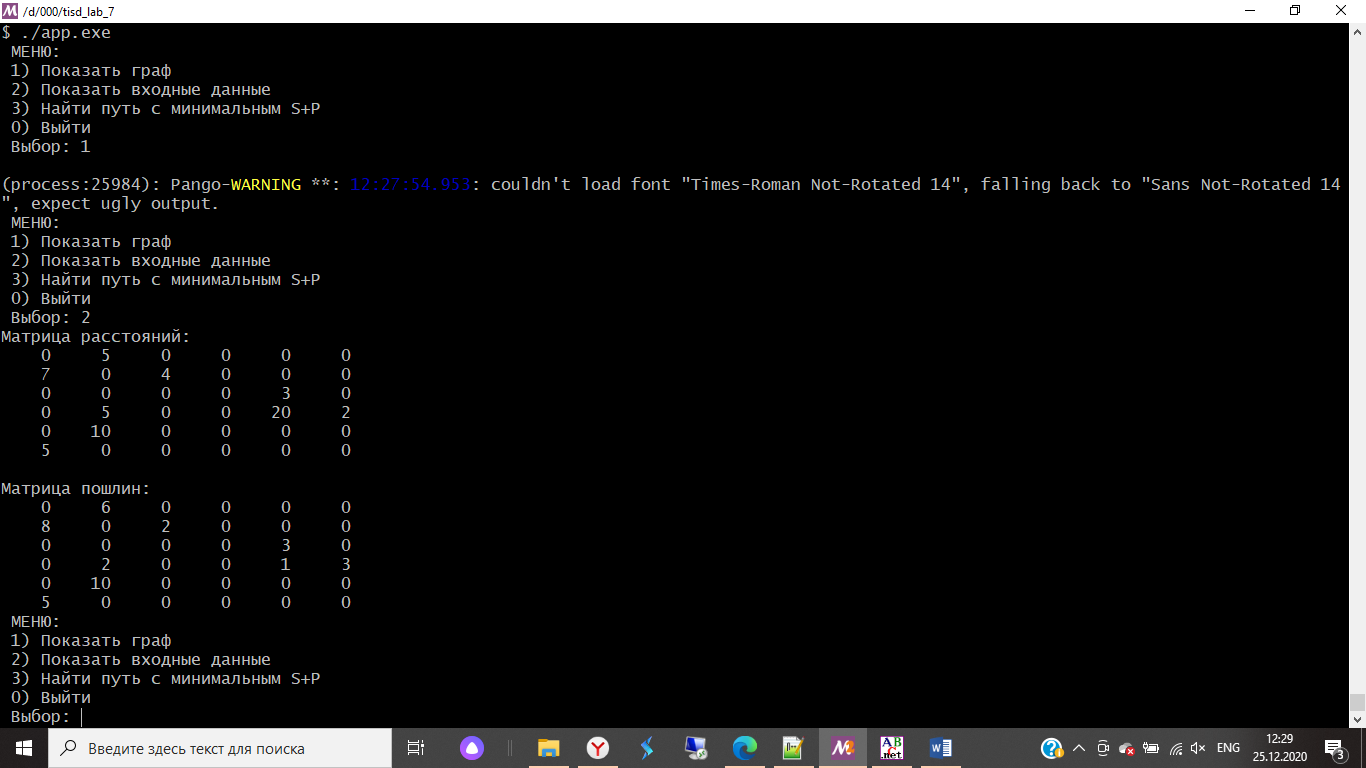
1. Меню



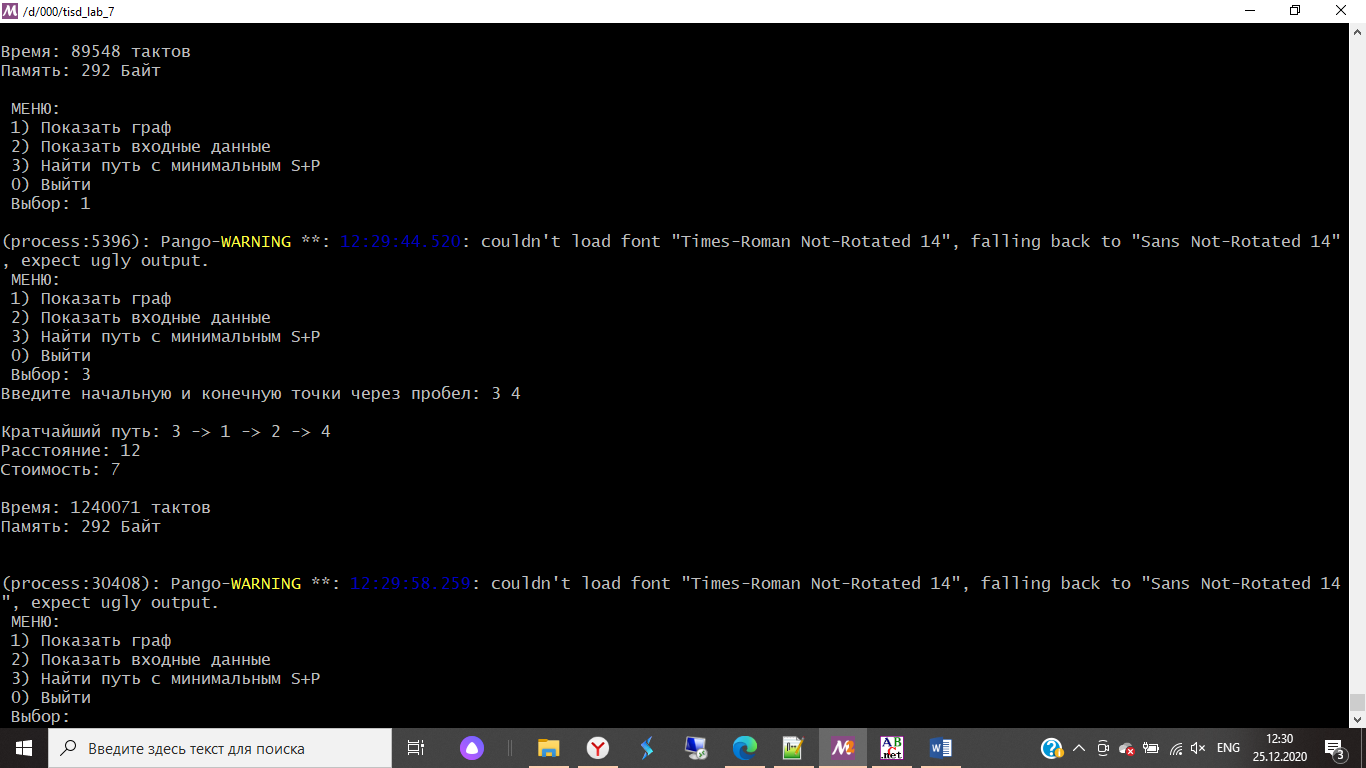
1. Вывести граф

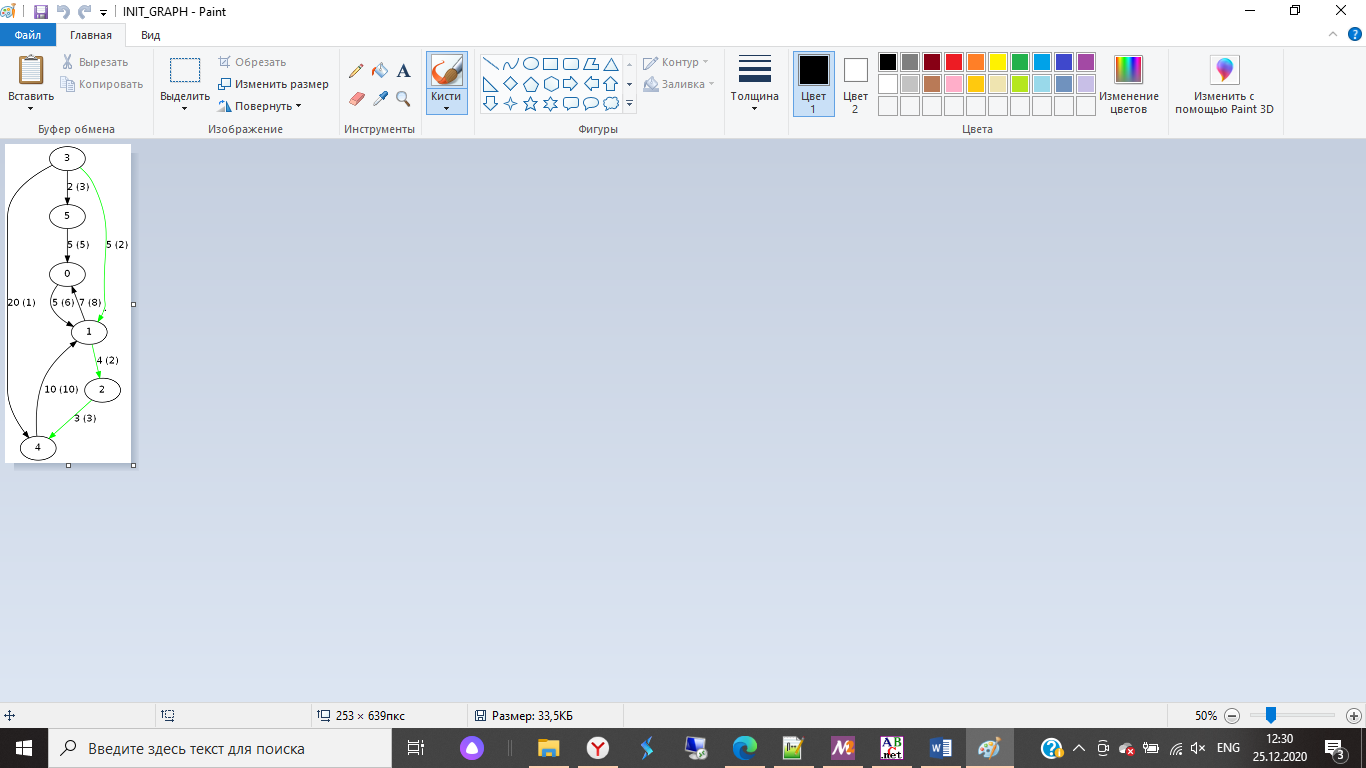


1. Вывод входных данных

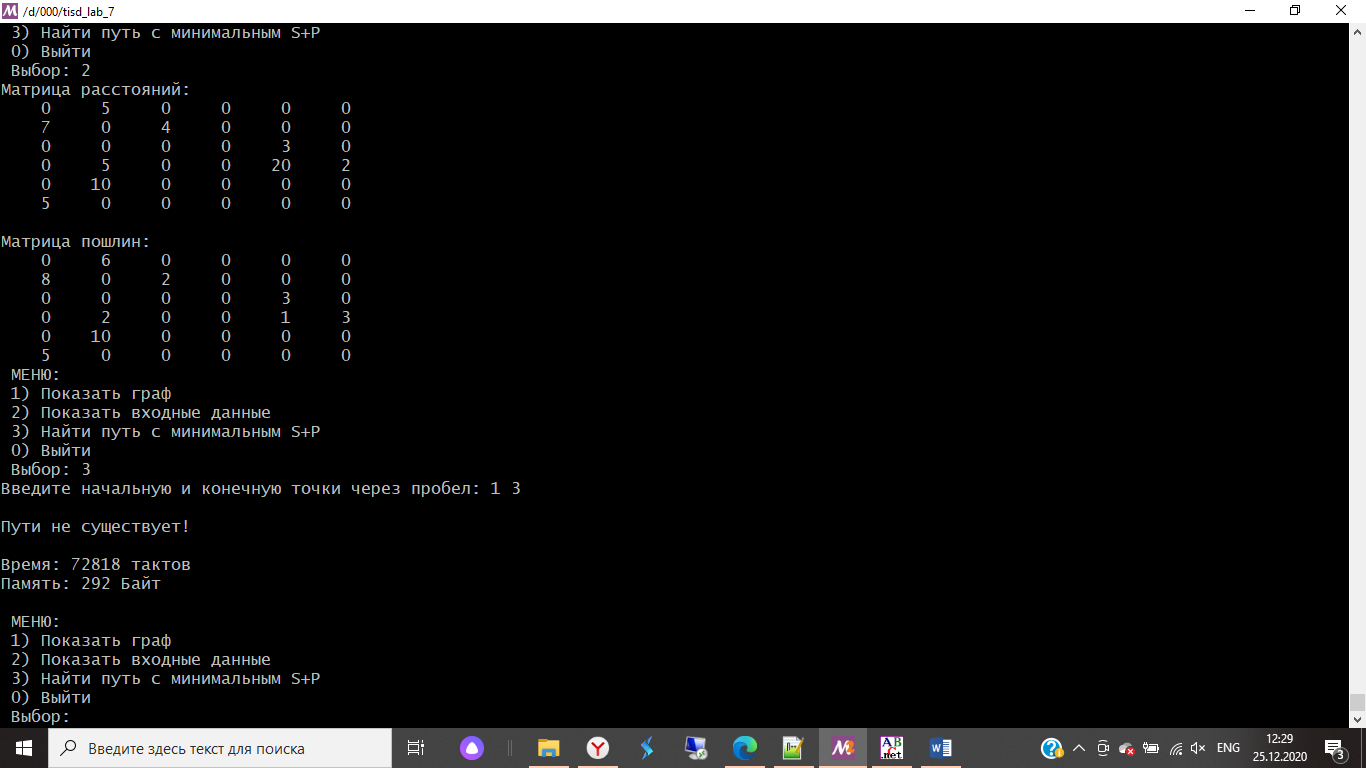


1. Вывод кратчайшего пути.





1. Кратчайшего пути не существует.



**Вывод**

Для реализации задачи был выбран алгоритм Дейкстры, так как в данной задаче нет отрицательных значений веса ребер (алгоритм Беллмана-Форда работает с подобными случаями), и нам требуется найти кратчайшие пути из одной вершины во все другие вершины (а не кратчайшие пути между всеми вершинами, как в алгоритме Флойда-Уоршалла ). Хранение массива в виде матрицы позволяет более удобно работать с графом, но является менее эффективным по памяти и времени чем хранение графа в обычном виде или граф в виде массива списков.

**Контрольные вопросы**

1. Что такое граф?

Граф – конечное множество вершин и соединяющих их ребер; G = <V, E>. V – вершины графа, E – ребра. Если пары Е (ребра) имеют направление, то граф называется ориентированным; если ребро имеет вес, то граф называется взвешенным. В моем задании мы имеем дело с ориентированным взвешенным графом.

1. Как представляются графы в памяти?

Существуют различные методы представления графов в программе.

* + - * 1. Матрица смежности B(n \* n) – элемент b[i, j] = вес ребра, если существует ребро, связывающее вершины i и j, и = 0, если ребра не существует.
        2. Список смежностей – содержит для каждой вершины из множества вершин V список тех вершин, которые непосредственно связаны с ней. Входы в списки смежностей могут храниться в отдельной таблице (в массиве), либо же каждая вершина может хранить свой список смежностей.

1. Какие операции возможны над графами?

Основные операции над графами: обход вершин и поиск различных путей: кратчайшего пути от вершины к вершине; кратчайшего пути от вершины ко всем остальным; кратчайших путей от каждой вершины к каждой; поиск эйлерова пути и гамильтонова пути, если таковые есть в графе.

1. Какие способы обхода графов существуют?

Один из основных методов проектирования графовых алгоритмов – это поиск (или обход графа) в глубину (depth first search, DFS). При поиске в глубину, начиная с произвольной вершины v0, ищется ближайшая смежная вершина v, для которой осуществляется поиск в глубину (т.е. снова ищется ближайшая смежная с ней вершина) до тех пор, пока не встретится ранее просмотренная вершина, или не закончится список смежности вершины v (то есть вершина полностью обработана). Если нет новых вершин, смежных с v, то вершина v считается использованной, идет возврат в вершину, из которой попали в вершину v, и процесс продолжается до тех пор, пока не получим v = v0. Иными словами, поиск в глубину из вершины v основан на поиске в глубину из всех новых вершин, смежных с вершиной v. Путь, полученный методом поиска в глубину, в общем случае не является кратчайшим путем из вершины v в вершину u. Это общий недостаток поиска в глубину.

Указанного недостатка лишен другой метод обхода графа – поиск в ширину (breadth first search, BFS). Обработка вершины v осуществляется путем просмотра сразу всех новых соседей этой вершины. При этом полученный путь является кратчайшим путем из одной вершины в другую.

1. Где используются графовые структуры?

Графовые структуры могут использоваться в задачах, в которых между элементами могут быть установлены произвольные связи, необязательно иерархические. Наиболее распространенным является использование графов при решении различных задач о путях, будь то построение коммуникационных линий между городами или прокладка маршрута на игровом поле.

1. Какие пути в графе Вы знаете?

Путь в графе, проходящий через каждое ребро ровно один раз, называется эйлеровым путём; путь может проходить по некоторым вершинам несколько раз – в этом случае он является непростым.

Путь, проходящий через каждую вершину ровно один раз, называется гамильтоновым путем. Как эйлеров, так и гамильтонов путь могут не существовать в некоторых графах.

Условие существования эйлерова пути доподлинно известно, гамильтонова – нет.

1. Что такое каркасы графа?

Каркас графа – дерево, в которое входят все вершины графа, и некоторые (не обязательно все) его рёбра. Для построения каркасов графа используются алгоритмы Крускала и Прима.