**Отчет лабораторной работы №7**

по курсу «Типы и структуры данных»

Графы

Отчёт выполнила:

Кондрашова Ольга

Группа ИУ7-35Б

Вариант 9

**Цель работы**:

Реализовать алгоритмы обработки графовых структур: поиск различных путей, проверку связности, построение остовых деревьев минимальной стоимости.

**Задание**:

Задана система двусторонних дорог. Найти множество городов, расстояние от которых до выделенного города (столицы) больше, чем Т.

**Входные данные:**

Программа считывает данные о графе из текстового файла.

Целое число – длина минимального пути между городом и столицей.

**Выходные данные:**

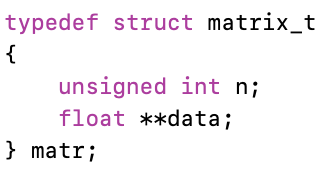
Граф, на котором показаны все дороги.

Граф, на котором показаны дороги, длина которых больше введенной пользователем величины.

**Обращение к программе:**Через консоль

**Внутренняя структура данных:**

Граф представлен в виде матрицы смежности размера n\*n, где в (i,j) ячейке хранится длина дороги из i в j.



**Описание задачи, реализуемой программой**

1. Построение графа на основе данных из текстового документа.
2. Нахождение путей, длина которых больше заданной величины.
3. Вывод матрицы дорог.

**Алгоритм**

1 – загрузить граф из файла.

2 – ввод минимальной длины пути Т.

3 – нахождение всех дорог, длина которых больше Т.

4 – вывод графов в графической форме.

**Функции**

int menu\_value(void)

Вход: void

Выход: пункт меню

float \*\*allocate\_matrix\_rows(int n)

Вход: размерность матрицы

Выход: память под матрицу

matr \*create\_matr(unsigned int n)

Вход: размерность матрицы

Выход: матрица

matr \*read\_matr(FILE \*f)

Вход: файл

Выход: считанная матрица

matr \*copy\_matr(const matr \*A)

Вход: матрица

Выход: скопированная матрица

void free\_matr(matr \*matrix)

Вход: матрица

Выход: освобождение памяти

void print\_matr(FILE \*f, const matr \*m)

Вход: файл, размерность матрицы

Выход: напечатанная матрица

void show\_matr\_gv(const matr \*a)

Вход: матрица

Выход: графическое изображение графа всех путей

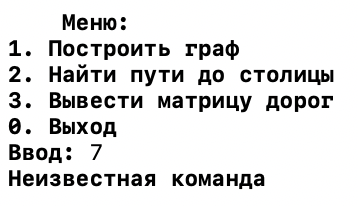
void show\_best\_gv(const matr \*b)

Вход: матрица

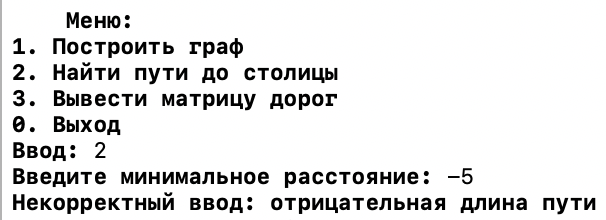
Выход: графическое изображение графа путей, удовлетворяющих условию

**Аварийные ситуации и ошибки пользователя**

1. Ввод несуществующей команды меню

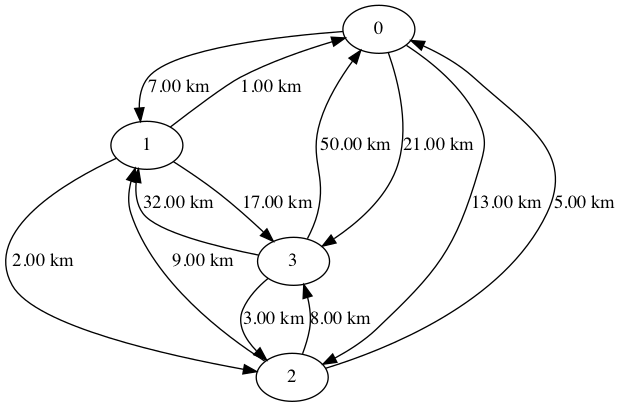


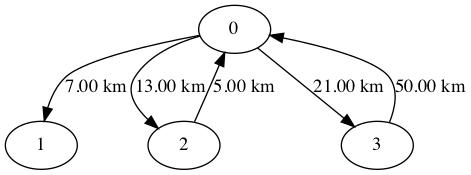
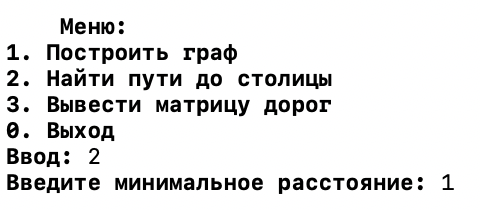
1. Ввод отрицательного значения длины пути

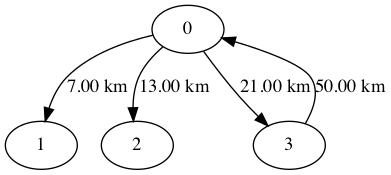
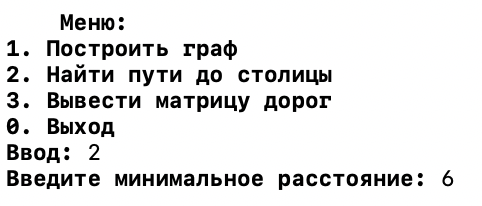


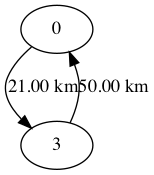
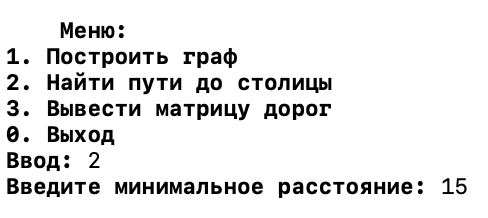
**Тесты**

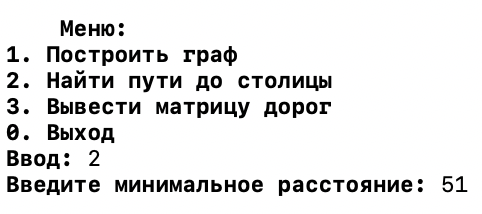
Основной граф







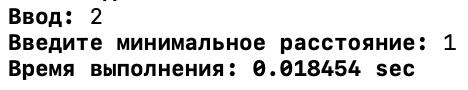
****

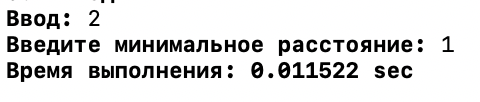
 (пустой граф)

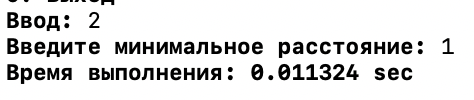
**Оценка эффективности**

Занимаемая память – 68 байт

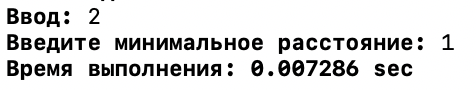
Оценка скорости:











Среднее значение – 0.0110368 секунд

**Вывод:**

Для реализации данной задачи применен алгоритм Дейкстры. Реализованный алгоритм находит все пути, которые больше заданного значения, от определённой вершины («столицы») до всех остальных «городов».

Алгоритм можно применить в жизни, например:

1. Имеется система дорог. Необходимо найти слишком длинные пути, чтобы исключить их из предполагаемого маршрута.
2. Имеется система платных дорог, где вес ребра обозначает цену проезда по данному пути. Нахождение дорог, стоимость проезда по которым превышает намеченный бюджет, поможет исключить их из маршрута.
3. В случае, если человек ставит перед собой цель проходить каждый день определенное расстояние и строит свой маршрут, исходя из этого.

**Ответы на вопросы**

1. Что такое граф?

Граф – конечное множество вершин и соединяющих их ребер; G = <V, E>. Если пары Е (ребра) имеют направление, то граф называется направленным; если ребро имеет вес, то граф называется взвешенным.

1. Как представляются графы в памяти?

Существуют различные методы представления графов в программе. Матрица смежности B(n\*n) – элемент b[i,j]=1, если существует ребро, связывающее вершины i и j, и =0, если ребра не существует.

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, 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 – поиск в глубину. Начиная с некоторой вершины v0, ищется ближайшая смежная ей вершина v, для которой в свою очередь осуществляется поиск в глубину 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. При просмотре используется стек.

Поиск в ширину – обработка вершины V осуществляется путём просмотра сразу всех «новых» соседей этой вершины, которые последоватеьно заносятся в очередь просмотра.

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.

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.