Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления (ИУ)

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии (ИУ7)

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7
«Графы»

Студент, группа **Дегтярев А., ИУ7-33Б**

2020 г.

Описание условия задачи

Задана система двусторонних дорог. Найти множество городов, расстояние от которых до выделенного города (столицы) больше, чем T. Результат выдать в графической форме.

Техническое задание

Входные данные

Имя файла с неориентированным графом, номер вершины-столицы, параметр T, тип структуры графа.

В первой строке файла с графом находится число — количество вершин. Затем идут тройки чисел — описание дуги графа. Первые 2 числа — номера вершин, 3-е число — длина дуги.

Выходные данные

Вывести список вершин, расстояние от которых до выделенной вершины-столицы больше, чем T.

Функция программы

Программа измеряет эффективность структур данных, позволяет найти минимальные расстояния от всех вершин до выделенной.

Обращение к программе

1. В папке с исходным кодом в консоли нужно вызвать команду:

make run args="ФАЙЛ СТОЛИЦА ПАРАМЕТР_T ТИП"

ФАЙЛ - имя файла с описанием графа.

СТОЛИЦА - номер вершины, к которой нужно считать путь.

ПАРАМЕТР_T - длина пути, с которого вершина попадает в выходной список.

ТИП - тип структуры графа. Принимает значения "m" (матрица) или "l" (список).

Аварийные ситуации

- Неправильные аргументы программы
- Нет файла с графом
- Неправильный формат данных в файле

Структуры данных

```
typedef struct
{
    int *data;
    int vertices_n;
} graph_matrix_t;
```

graph_matrix_t — структура графа на матрице.

Поля структуры:

- *data* — квадратная матрица длин ребер
- *vertices_n* — количество вершин

Эффективность структуры по памяти: $O(n^2)$, где *n* — количество вершин.

```
typedef struct list_s list_t;

struct list_s
{
    int vertex;
    int weight;
    list_t *next;
};

typedef struct
{
    list_t **data;
    int vertices_n;
} graph_list_t;
```

list_t — структура списка смежностей

Поля структуры:

- *vertex* — вершина, в которую ведет дуга
- *weight* — длина дуги
- *next* - указатель на следующий элемент списка

Эффективность структуры по памяти: $O(n)$, где *n* — количество дуг

graph_list_t — структура графа на списке.

Поля структуры:

- *data* — массив указателей на список дуг, по которым можно выйти из вершины
- *vertices_n* — количество вершин

Эффективность структуры по памяти: $O(n+m)$, где *n* — количество вершин, m — количество дуг.

```
typedef struct {
    void **buf;
    void **end;
    size_t size;
} stack_t;
```

stack_t — стек, реализованный массивом.

Поля структуры:

- *buf* — массив указателей на данные, которые были добавлены в стек
- *end* — указатель на элемент, следующий за последним добавленным указателем
- *size* — размер массива *buf*

Эффективность структуры по памяти: $O(size)$, где *size* — наибольшая степень 2-ки, которая не меньше количества элементов массива.

Алгоритм

1. Считать граф из файла
2. Найти расстояния от каждой вершины графа до вершины-столицы с использованием алгоритма Дейкстры.
    - Дополнительная память — $O(n^2)$, время $O(n^2)$
3. Вывести вершины, длина до которых больше указанной
4. Записать граф в файл в dot формате, для последующего его отображения с использованием библиотеки Graphviz

Тесты

Группа	Входные данные	Выходные данные
Аргументы	Неправильное количество аргументов	Неправильные аргументы
Файл	Файл не существует	Неправильный ввод
	Файл пустой	Неправильный ввод
	Отрицательное количество вершин	Неправильный ввод
	Отрицательная вершина или расстояние в дуге	Неправильный ввод

Оценка эффективности

Время в тактах

Количество вершин	Процент ребер	Граф на матрице			Граф на списке		
		Чтение из файла	Расчёт расстояний	Запись в файл	Чтение из файла	Расчёт расстояний	Запись в файл
10	25	33924	19255	409690	36879	38299	575764
10	50	43503	180913	1046481	35474	310330	1134697
10	75	54327	310581	518598	43692	182146	378106
100	25	740616	297788836	1674161	2493662	361829812	507386
100	50	2362997	906507824	1644530	4561964	2210577739	1181811
100	75	4040255	1730487943	2145230	10007010	5377751221	683969
250	25	7655758	12956272764	8960831	21961702	31429467362	1972958
250	50	9755637	39230565682	9606992	91084743	231030077295	3175954
250	75	12547018	60836270532	14287409	209452772	699699599597	4769135

Количество вершин	Процент ребер	Размер граф на матрице в байтах	Размер графа на списке в байтах
10	25	400	608
10	50	400	1160
10	75	400	1688
100	25	40000	60200
100	50	40000	119600
100	75	40000	179000
250	25	250000	375488
250	50	250000	749000
250	75	250000	1122488

Функции

```
graph_list_t *create_graph_list(int n);
    Создаёт граф на списке с n вершинами

void free_graph_list(graph_list_t *g);
    Удаляет граф на списке из памяти

int graph_list_set(graph_list_t *g, int v_from, int v_to, int weight);
    Устанавливает длину дуги в графе на списке

int graph_list_get(graph_list_t *g, int v_from, int v_to);
    Получить длину дуги в графе на списке

int graph_list_get_next(graph_list_t *g, int v_from, int v_last);
    Взять следующую за v_last вершину в графе на списке, в которую можно попасть
из v_from


graph_matrix_t *create_graph_matrix(int n);
    Создаёт граф на матрице с n вершинами

void free_graph_matrix(graph_matrix_t *g);
    Удаляет граф на матрице из памяти

int graph_matrix_set(graph_matrix_t *g, int v_from, int v_to, int weight);
    Устанавливает длину дуги в графе на матрице

int graph_matrix_get(graph_matrix_t *g, int v_from, int v_to);
    Получить длину дуги в графе на матрице

int graph_matrix_get_next(graph_matrix_t *g, int v_from, int v_last);
    Взять следующую за v_last вершину в графе на матрице, в которую можно по-
пасть из v_from


int stack_push_vertex(stack_t *s, int vertex, int distance);
    Положить в стек вершину с расстоянием до нее

int stack_pop_vertex(stack_t *s, int *vertex, int *distance);
    Взять из стек вершину с расстоянием до нее

void print_result_verteces(int *distances, int n, int param_t);
    Вывести в консоль вершины, расстояние до которых больше param_t

int process_with_graph_matrix(char *filename, int capital, int param_t);
    Выполнить задачу с использованием графа на матрице

int process_with_graph_list(char *filename, int capital, int param_t);
    Выполнить задачу с использованием графа на списке
```

Вывод

Граф на матрице быстрее в 2 раза графа на списке и в среднем в 3 раза эффективнее по памяти графа на списках, так как матрица предоставляет прямой доступ к данным. Алгоритм Дейкстры подходит к задачам поиска кратчайшего пути в графе с неотрицательными весами. Его эффективность по времени $O(n^2)$, n — количество вершин.

Контрольные вопросы

Что такое граф?

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

Как представляются графы в памяти?

С помощью матрицы смежности или списков смежности.

Какие операции возможны над графами?

Обход вершин, поиск различных путей, исключение и включение вершин.

Какие способы обхода графов существуют?

Обход в ширину и обход в глубину.

Где используются графовые структуры?

Графовые структуры могут использоваться в задачах, в которых между элементами могут быть установлены произвольные связи, необязательно иерархические.

Какие пути в графе Вы знаете?

Эйлеров путь, гамильтонов путь.

Что такое каркасы графа?

Каркас графа – дерево, в которое входят все вершины графа, и некоторые (необязательно все) его рёбра.