Министерство науки и высшего образования 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»
Кафедра «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №8
«Обработка графов»

по курсу «Типы и структуры данных»

Студент: Есин Денис Павлович

Группа: ИУ7-36Б

Студент _____ Есин Д. П.
подпись, дата

Преподаватель _____ Никульшина Т. А.
подпись, дата

Оценка _____

2022 г

Описание условия задачи

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

Указания к выполнению работы

При отладке программы необходимо проверить правильность ввода данных. Программа должна адекватно реагировать на неверный ввод, в том числе на «пустой» ввод. Необходимо проверять поиск несуществующих путей в графе, использовать различные варианты графов для проверки правильности работы программы.

При разработке интерфейса программы следует предусмотреть:

- указание типа, формата и диапазона вводимых данных;
- указание действий, производимых программой;
- наличие пояснений при выводе результата;
- вывод графов осуществить в графическом виде (или предложить иную визуализацию в виде графа);
- вывод времени выполнения программы и объема требуемой памяти при использовании различных структур для представления графа.

При тестировании программы необходимо:

- проверить правильность ввода и вывода данных (т.е. их соответствие требуемому типу и формату), обеспечить адекватную реакцию программы на неверный ввод данных;
- обеспечить вывод сообщений при отсутствии входных данных («пустой ввод»);
- проверить правильность выполнения операций;
- предусмотреть вывод сообщения при поиске несуществующих путей в графе.

Описание технического задания

Задана система двусторонних дорог. Для каждой пары городов найти длину кратчайшего пути между ними.

Входные данные:

1. **Файлы с данными:** файлы (в названии указаны размеры графов) с данными для заполнения графов, данные хранятся в матричном виде: если дорога между городами есть, то значение ячейки матрицы отлично от нуля и меньше 10000, если нет, то равно 10000, дорога из одного города в тот же город равна 0.
2. **Целое число, представляющее собой номер команды**: целое число в диапазоне от 0 до 17.

Выходные данные:

1. Граф дорог в графическом виде (выводится с помощью Graphviz).
2. Таблица минимальных расстояний между городами.

Функции программы:

0. Выход из программы
1. Заполнение графа из файла
2. Вывод графа в графическом виде с помощью Graphviz
3. Создание матрицы минимальных расстояний между городами
4. Вывод матрицы минимальных расстояний
5. Сравнение работы алгоритма для матричной и для списковой версий представления графа
6. Вывод меню

Обращение к программе:

Запуск программы производится из терминала.

Листинг 1: Пример запуска программы из коренной папки проекта

```
~/D/P/d/lab_8> ./app.exe
```

Аварийные ситуации:

10. Выбор неверного пункта меню:

    Вход — неверный пункт меню.

    Выход — сообщение об ошибке.

10. Попытка провести операцию с пустым графом.

    Вход — пустой граф.

    Выход — сообщение об ошибке.

Структуры данных

Граф представлен в двух видах.

В виде матрицы стоимостей:

```
typedef struct road_graph
{
        size_t total_cities;
        int **roads_matrix;
} road_graph_t;
```

total_cities — число вершин графа;

***roads_matrix* —** матрица стоимостей в виде двумерного массива.

И в виде связного списка:

```
typedef struct conn_city
{
        size_t ind;
        int dist;
        struct conn_city *next;
} conn_city_t;


typedef struct beg_city
{
        conn_city_t *next;
} beg_city_t;


typedef struct road_graph_list
{
        size_t total_cities;
        beg_city_t *cities;
} road_graph_list_t;
```

ind – номер города

dist – расстояние до города от того города, с которым он соединен

next – один из тех городов, с которым соединен данный город

total_cities – общее количество городов

cities – одномерный массив городов

Алгоритм

По мере ввода графа заполняется матрица стоимостей, которая является представлением графа. Значение 0 соответствует пути из вершины в себя же, значение, близкое к бесконечности (в идеале бесконечность, но, ввиду ограничений со стороны реализации, используется 10000). Для поиска кратчайших путей в графе используется алгоритм Флойда-Уоршолла.

Тесты

Таблица 1: Негативные тесты

№	Тип теста	Описание	Входные данные	Результат
1	Негативный	Тест меню: неправильный пункт	999	Ошибка
2	Негативный	Тест меню: неправильный пункт	sdf	Ошибка
3	Негативный	Попытка выполнить операцию над графом без предварительного его заполнения	-	Ошибка

Таблица 2: Позитивные тесты

№	Тип теста	Описание теста	Входные данные	Результат
1	Позитивный	Тест меню	2	Вывод графа в виде графа с помощью graphviz
2	Позитивный	Тест меню	4	Вывод результатов поиска минимальных путей в графе.

Оценка эффективности

Таблица, в которой содержатся результаты измерений времени выполнения алгоритма Флойда-Уоршалла и результаты замера размера матриц для каждой из

форм (матричной и списковой) представления графа для разного количества городов.

Время измеряется в миллисекундах, память в байтах. Каждый тест запускается 12500 раз (при выключенной оптимизации компилятора), поэтому погрешности измерений практически нулевые.

Количество городов	Предмет измерения	Граф в виде матрицы	Граф в виде списка
5	Время	0.001280	0.001760
	Память	148	336
10	Время	0.005520	0.008720
	Память	488	1144
15	Время	0.017440	0.025120
	Память	1028	2480
20	Время	0.040320	0.056320
	Память	1768	5352
25	Время	0.075760	0.101280
	Память	2708	7408
30	Время	0.130000	0.172800
	Память	3848	9752
35	Время	0.205360	0.266000
	Память	5188	15168
40	Время	0.309200	0.396240
	Память	6728	18136
45	Время	0.416720	0.538320
	Память	8468	23360
50	Время	0.578720	0.743280
	Память	10408	30168

Количество городов	Предмет сравнения	Результаты графа в виде матрицы по отношению к результатам графа в виде списка
5	Время	0.727273
	Память	0.440476
10	Время	0.633028
	Память	0.426573
15	Время	0.694268
	Память	0.414516
20	Время	0.715909
	Память	0.330344
25	Время	0.748025
	Память	0.365551
30	Время	0.752315
	Память	0.394586
35	Время	0.772030
	Память	0.342036
40	Время	0.780335
	Память	0.370975
45	Время	0.774112
	Память	0.362500
50	Время	0.778603
	Память	0.345001

Контрольные вопросы

1. Что такое граф?

Граф – конечное множество вершин и соединяющих их ребер; **G = (V, E)**. Если пары **E** (ребра) имеют направление, то граф называется ориентированным, в ином случае считается, что ребра двунаправлены, тогда граф считается неориентированным; если ребро имеет вес, то граф называется взвешенным.

2. Как представляются графы в памяти?

С помощью матрицы смежностей (стоимостей) или списков смежностей.

3. Какие операции возможны над графами?

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

4. Какие способы обхода графов существуют?
Обход в ширину, обход в глубину

5. Где используются графовые структуры?

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

6. Какие пути в графе Вы знаете?

Эйлеров путь, простой путь, Гамильтонов путь, сложный путь.

7. Что такое каркасы графа?

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

Вывод

Я научился реализовывать хранение и обработку графов, изучил алгоритм Флойда-Уоршалла.

Были проведены измерения с целью выяснить, имеется ли преимущество в использовании списка смежностей вместо матрицы смежностей. В виду того, что алгоритм Флойда-Уоршалла подразумевает работу с матрицей, список необходимо было переводить в матрицу, в результате чего для каждого случая списковое преставление проигрывало матричному. К тому же, список занимает больше места в памяти.

В связи с вышеперечисленным, список не рекомендуется в качестве структуры для хранения графа в случаях, если алгоритмы обработки графа требуют работы с матрицами.