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» (ИУ)
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии» (ИУ7)

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №8

«ГРАФЫ»

по курсу:

«ТИПЫ И СТРУКТУРЫ ДАННЫХ»

Вариант: 9

Студент:
Авдейкина Валерия Павловна, группа ИУ7-33Б _____
(подпись, дата)

Руководители:
Преподаватель ИУ7
Силантьева Александра Васильевна _____
(подпись, дата)
Преподаватель ИУ7
Барышникова Марина Юрьевна _____
(подпись, дата)

Проверяющий:
Барышникова Марина Юрьевна _____
(подпись, дата)

Оценка: _____

2022 г.

Оглавление

Описание условий задачи

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

Вариантное задание (9): Задана система двусторонних дорог, где для любой пары городов есть соединяющий их путь. Найти город с минимальной суммой расстояний до остальных городов.

Техническое задание

1 Входные данные

- Входные данные при вводе номера выбранной опции меню:
  - Целые беззнаковые числа [0; 14]
- Входные данные для различных опций меню:

Номер опции	Название опции	Входные данные
0	Выход	-
1	Задать файл для обрабтки	Имя файла
2	Заполнить из файла таблицу смежности, списки	-
3	Вывести на экран таблицу смежности, списки	-
4	Показать граф (по таблице)	-
5	Показать граф (по спискам)	-
6	Найти город с минимальной суммой кратч. путей до остальных городов	-
7	Выполнить сравнение затрат	-
8	Вывести меню	-
9	Вывести правила ввода	-
10	Очистить все	-

Таблица 1: Входные данные опций меню

- Имя файла: символы, не более 255
- Ввод матрицы смежности и списков ребер осуществляется с помощью соответствующих файлов в папке data.

2 Выходные данные

- Выходные данные при вводе номера выбранной опции меню:
  - Выходные данные команды в случае корректного ввода
- Выходные данные для различных опций меню:

Номер опции	Название опции	Выходные данные
0	Выход	Сообщение об успехе / Сообщение об ошибке
1	Задать файл для обрабтки	Сообщение об успехе / Сообщение об ошибке
2	Заполнить из файла таблицу смежности, списки	Сообщение об успехе / Сообщение об ошибке
3	Вывести на экран таблицу смежности, списки	Данные / Сообщение об ошибке
4	Показать граф (по таблице)	Данные / Сообщение об ошибке
5	Показать граф (по спискам)	Данные / Сообщение об ошибке
6	Найти город с минимальной суммой кратч. путей до остальных городов	Данные / Сообщение об ошибке
7	Выполнить сравнение затрат	Данные / Сообщение об ошибке
8	Вывести меню	Меню
9	Вывести правила ввода	Правила ввода
10	Очистить все	Сообщение об успехе / Сообщение об ошибке

Таблица 2: Выходные данные опций меню

3 Задача, реализуемая программой

Программа реализует поиск вершины, сумма кратчайших путей от которой до всех остальных вершин минимальна, используя при этом графовую структуру.

4 Способ обращения к программе

Программа вызывается в командной строке без каких-либо аргументов. Ввод данных производится с клавиатуры после соответствующего приглашения к вводу («Команда...» / «Введите..»).

5 Возможные аварийные ситуации и ошибки со стороны пользователя

1. Некорректный ввод имени файла:
    1.1. Пустой ввод
    1.2. Недопустимые символы

1.3. Превышение допустимого количества символов *(поведение программы не определено)*

1.4. Имя несуществующего файла или директории

2. Некорректные данные в обрабатываемом файле / вводе с клавиатуры

3. Некорректный ввод номера опции:

3.1. Пустой ввод

3.2. Недопустимые символы (*поведение программы не определено*)

3.3. Недопустимый номер

Описание внутренних структур данных

В ходе работы были составлены следующие структуры:

1. Матрица смежности (стоимостей) — my_matrix_t, где:
   - data — двумерный массив стоимостей
   - rows — количество рядов (вершин, из которых сущетсвуют дороги)
   - cols — количество столбцов (вершин, в которые существуют дороги)

```
typedef int bool;
typedef struct my_matrix
{
   size_t **data;
   size_t rows;
   size_t cols;
} my_matrix_t;
```

Итоговый размер структуры: 20 байт.

2. Массив списков ребер для каждой вершины графа — list_t, где:
   - size — размер массива (количество вершин)
   - nodes — одномерный массив указателей на начала списков ребер list_data_t, где:
     - vertex — вершина, до которой существует путь
     - dist — путь до вершины vertix
     - next — указатель на следующий элемент списка

```
struct list_elem
{
   size_t dist;
   size_t vertex;
   list_data_t next;
};

typedef struct list_elem *list_data_t;

typedef struct list
{
   size_t size;
   list_data_t *nodes;
} list_t;
```

Итоговый размер структуры list_t: 12 байт, *list_data_t:* 16 байт

Алгоритм

Для реализации решения задачи было решено для каждой вершины графа использовать **алгоритм Дейкстры**, суммировать полученные кратчайшие расстояния и сравнивать между собой с поиском минимальной суммы.

Обоснование выбора алгоритма:

- Составным шагом задачи является поиск кратчайших путей от заданной вершины до всех остальных вершин графа
- Стоимость путей между городами не может быть отрицательной, следовательно, алгоритм Дейкстры будет работать корректно

Описание алгоритма:

Каждой вершине графа U сопоставляется метка — кратчайшее расстояние от SRC (исходной), для каждой вершины U имеется состояние S — TRUE или FALSE (старая/новая)

Инициализация: все вершины считаются новыми, массив меток заполнен «плюс»-бесконечностями, кроме метки исходной вершины — она равна нулю.

Основной цикл:

- Пока есть новые вершины U:
  - среди ребер (U,V), таких, что U — старая, V — новая, выбираем такое, что сумма кратчайшего пути от SRC до U и ребра (U,V) — минимальна
  - помечаем V старой
    - если полученная сумма меньше кратчайшего расстояния от SRC до V, заменяем это расстояние суммой

Массив меток (расстояний) будет результатом алгоритма.

Сложность алгоритма: $O(|V|^2)$ — простейший случай (используется обычный массив меток, проверяются все вершины)

В реализации решения задачи были использованы следующие основные функции:

Функция	Описание
`size_t *get_matrix_dijkstra(my_matrix_t graph_matrix, const size_t source_v)`	Возвращает массив кратчайших расстояний от вершины source_v с использованием алгортима Дейкстры. Граф представлен матрицей смежности graph_matrix
`int matrix_get_min_sum_city(my_matrix_t matrix, result_t *result)`	Возвращает код ошибки/успеха. Выполняет поиск вершины, сумма кратчайших расстояний от которой до остальных вершин графа минимальна. Записывает результат в переменную result. Граф представлен матрицей смежности matrix
`size_t get_arr_sum(const size_t size, size_t *arr)`	Возвращает сумму элементов массива arr длины size
`size_t get_min_dist_vertex(const size_t num_of_v, size_t *dist, bool *check_status)`	Возвращает индекс новой в соответствии с соответствующим элементом check_status вершины, расстояние из dist до которой минимально
`size_t *get_list_dijkstra(list_t graph_list, const size_t source_v)`	Возвращает массив кратчайших расстояний от вершины source_v с использованием алгортима Дейкстры. Граф представлен массивом списков ребер graph_list
`size_t get_dist_from_to(list_data_t *data, const size_t v, const size_t w)`	Возвращает расстояние от вершины v до вершины w с помощью списка ребер data вершины v
`int list_get_min_sum_city(list_t list, result_t *result)`	Возвращает код ошибки/успеха. Выполняет поиск вершины, сумма кратчайших расстояний от которой до остальных вершин графа минимальна. Записывает результат в переменную result. Граф представлен массивом списков ребер list

Таблица 3. Описание основных функций

Тестирование

- **Команда 1**

```
Команда (8 - меню, 9 - правила): 1
Ввод файла таблицы...
Введите имя файла: ./data/matrix/5.txt
Файл таблицы был успешно задан
Ввод файла списков...
Введите имя файла: ./data/list/5.txt
Файл списков был успешно задан
```

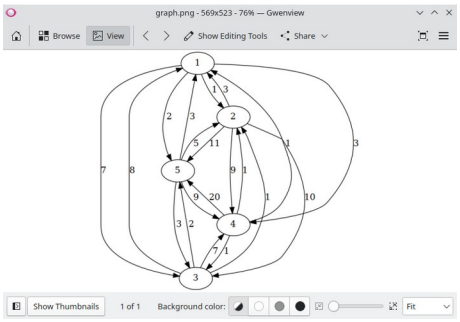
- **Команда 2**

```
Команда (8 - меню, 9 - правила): 2
Начало загрузки таблицы смежности...
Таблица смежности была успешно заполнена
Начало загрузки списков ребер...
Списки ребер был успешно заполнен
```

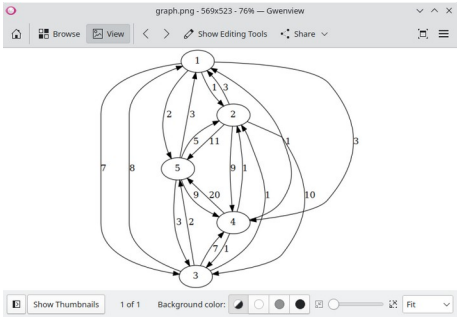
- **Команда 3**

```
Команда (8 - меню, 9 - правила): 3
Текущая таблица смежности:

      1   2   3   4   5
  1:  0   1   7   3   2
  2:  3   0  10   9  11
  3:  8   1   0   7   2
  4:  1   1   1   0  20
  5:  3   5   3   9   0
Текущие списки ребер:

1
-> 2: 1
-> 3: 7
-> 4: 3
-> 5: 2
2
-> 1: 3
-> 3: 10
-> 4: 9
-> 5: 11
3
-> 1: 8
-> 2: 1
-> 4: 7
-> 5: 2
4
-> 1: 1
-> 2: 1
-> 3: 1
-> 5: 20
5
-> 1: 3
-> 2: 5
-> 3: 3
-> 4: 9
```

- **Команда 4**



- **Команда 5**



- **Команда 6**

```
Команда (8 - меню, 9 - правила): 6
Начат поиск в таблице смежности...
Результаты по таблице
Номер: 4
Сумма: 6
Начат поиск в списках ребер...
Результаты по спискам
Номер: 4
Сумма: 6
```

- **Команда 7**

```
Команда (8 - меню, 9 - правила): 7

КОЛИЧЕСТВО ГОРОДОВ: 5
КОЛИЧЕСТВО ИТЕРАЦИЙ: 500

СРЕДНЕЕ ВРЕМЯ ПОИСКА ГОРОДА С МИН. СУММОЙ ПУТЕЙ ДО ВСЕХ ОСТАЛЬНЫХ
С использованием матрицы смежности: 564 мкс
С использованием списков ребер: 786 мкс

ЗАНИМАЕМЫЙ ОБЪЕМ ПАМЯТИ
С использованием матрицы смежности: 216 Б
С использованием списков ребер: 408 Б
```

- **Команда 10**

```
Команда (8 - меню, 9 - правила): 10
Данные были успешно очищены
```

Оценка эффективности

Измерение времени проводилось в одной плоскости — сравнивался поиск требуемого элемента в двух разных структурах. Количество итераций: 500.

Количество вершин графа	Тип хранения графа	Среднее время поиска, мкс	Размер структуры, Б
3	Матрица смежности	**282**	88
	Массив списков ребер	**294**	104
5	Матрица смежности	**558**	216
	Массив списков ребер	**790**	408

Из полученных результатов можно сделать следующие выводы:

- При очень малых размерах графа в обработке разных структур разница почти незаметна
- При увеличении размера графа обработка матрицы смежности выбранным алгоритмом становится явно эффективнее по времени
- Память, занимаемая массивом списков ребер, намного больше, чем память, занимаемая матрицей смежности, в любом случае, следовательно, использовать представление списками неэффективно по памяти в случае, когда в каждой паре вершин имеется путь

Контрольные вопросы

1 Что такое граф?

Абстрактный тип данных, предназначенный для реализации понятий неориентированного графа и ориентированного графа из области теории графов в математике. В общем случае: граф — совокупность множества вершин и множества ребер.

2 Как представляются графы в памяти?

Графы могут быть представлены двумя способами: матрицей смежности, с помощью списков ребер для каждой вершины.

3 Какие операции возможны над графами?

- Поиск кратчайшего пути от одной вершины к другой
- Поиск кратчайшего пути от одной вершины к остальным
- Поиск кратчайших путей между всеми вершинами
- Поиск эйлерова пути
- Поиск гамильтонова пути

4 Какие способы обхода графов существуют?

Существует два способа обхода графа: поиск (обход) в глубину, поиск (обход) в ширину.

5 Где используются графовые структуры?

Наибольшей популярностью графовые графовые пользуются при исследовании коммуникационных сетей, систем информатики, химических и генетических структур, электрических цепей и других систем сетевой структуры.

6 Какие пути в графе Вы знаете?

- Эйлеров путь — произвольный путь, проходящий через каждое ребро графа ровно один раз
- Гамильтонов путь — произвольный путь, проходящий через каждую вершину графа ровно один раз

7 Что такое каркасы графа?

Каркасом, или остовным деревом для графа называется связный подграф этого графа, содержащий все его вершины и не имеющий циклов. Количество ребер в каркасе связного графа всегда на единицу меньше количества вершин графа.

Выводы

В ходе работы были изучены детали обработки различных графовых структур с помощью различных алгоритмов, а также выяснено, что если для каждой пары вершин графа существует двухсторонняя дорога, выгоднее использовать представление в виде матрицы достижимости.