


| | |
|---|--|
|  | <p>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)</p> |
|---|--|

ФАКУЛЬТЕТ

Информатика и системы управления

КАФЕДРА

Программное обеспечение ЭВМ и информационные технологии

Отчет по лабораторной работе №8 **«Графы»**

Студент

Родинков Алексей Глебович

Группа

ИУ7 – 31БВ

Преподаватель

Силантьева Александра Васильевна

2023 год.

Оглавление

| | |
|------------------------------------|---|
| Описание условия задачи..... | 3 |
| Описание технического задания..... | 3 |
| Описание структуры данных..... | 3 |
| Описание алгоритма..... | 4 |
| Набор тестов..... | 5 |
| Оценка эффективности (такты)..... | 5 |
| Ответы на контрольные вопросы..... | 6 |

ОПИСАНИЕ УСЛОВИЯ ЗАДАЧИ

Цель работы: реализовать алгоритмы обработки графовых структур: поиск различных путей, проверка связности, построение остовых деревьев минимальной стоимости.

ОПИСАНИЕ ТЕХНИЧЕСКОГО ЗАДАНИЯ

Задана система двусторонних дорог.

Определить, можно ли, закрыв какие-нибудь три дороги, добиться того, чтобы из города А нельзя было попасть в город В.

Входные данные:

Система двусторонних дорог. (неорграф, текстовый файл)

Выходные данные:

Изображение графа. Ответ на поставленную задачу.
Сравнение работы алгоритмов.

Обращение к программе:

Запускается через терминал командой: `./app.exe`.

Аварийные ситуации:

1. Некорректные ввод номера меню.
2. Пустой файл.
3. Несуществующий граф

ОПИСАНИЕ СТРУКТУРЫ ДАННЫХ

```
typedef struct graph_t graph_t;

struct graph_t
{
    size_t lines; // количество уникальных строк
    size_t v;     // порядок графа (количество вершин)
    size_t e;     // размер графа (количество ребер)
    size_t maxamount; // количество выделенной памяти под массив указателей
    dict_t **heads; // массив указателей на голову списка
};
```

Дескриптор графа

```
typedef struct dict_t dict_t;

struct dict_t
{
    char *vertex_name; // название вершины
    int i;             // позиция в массиве смежности
    node_t *head;      // указатель на начало списка смежности
    node_t *tail;      // указатель на хвост для добавления в список смежности
};
```

Дескриптор списка графа

```
//
typedef struct node_t node_t;

struct node_t
{
    char *vertex_name; // название вершины
    int way_cost;      // стоимость прохода по пути
    node_t *next;      // указатель на следующий элемент
};

typedef struct dict_t dict_t;
```

Ребро и вершина графа

```
/// меню программы
enum menu
{
    EXIT,
    INIT,
    PRINT,
    FORD,
    BRUTEFORCE,
    VERTEX,
};
```

Структура главного меню.

ОПИСАНИЕ АЛГОРИТМА

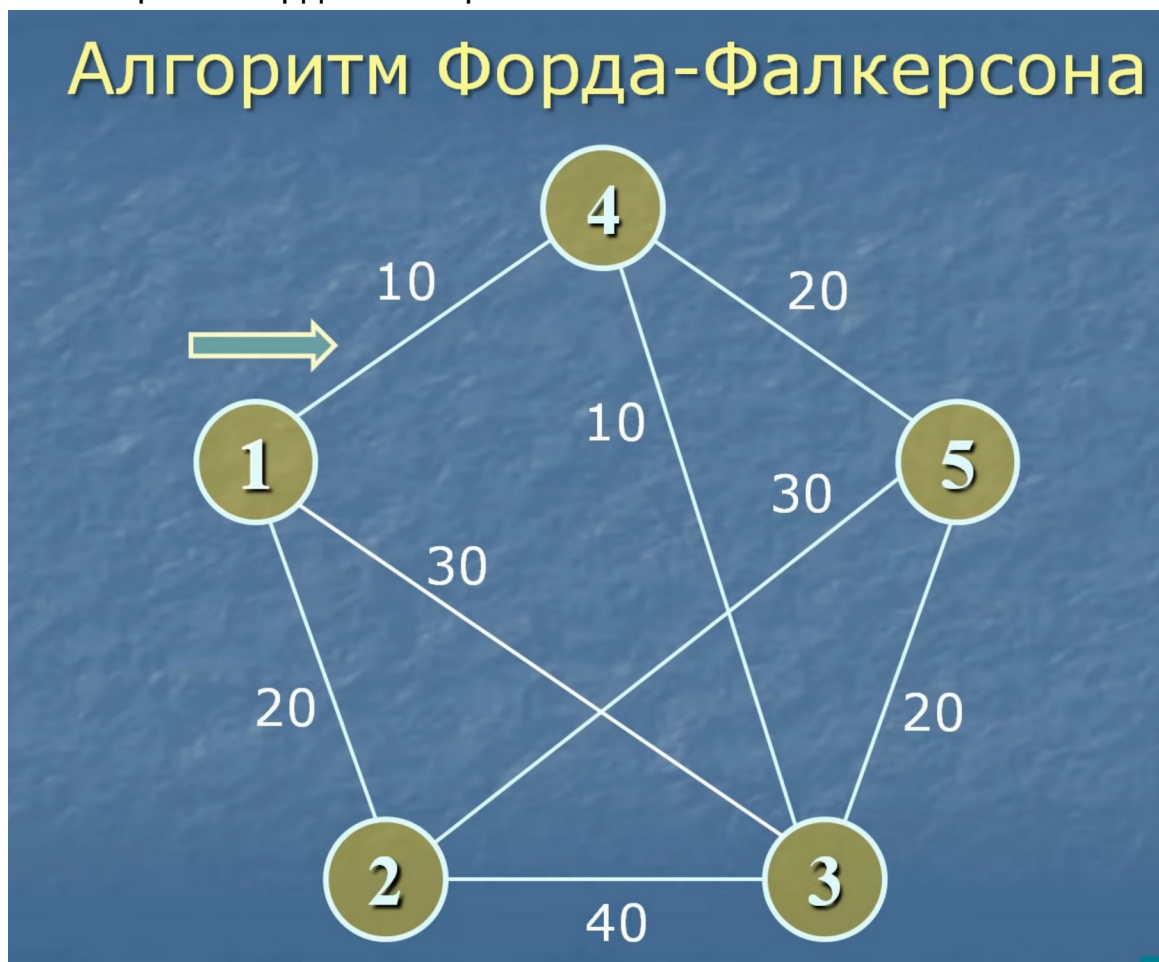
1. После запуска программы пользователю предлагается выбрать пункт меню.
2. После успешного ввода (
 - 2.1 Инициализация нового графа
 - 2.2 Вывод графа
 - 2.3 Алгоритм полного перебора
 - 2.4 Алгоритм Форда-Фалкерсона
 - 2.5 Добавление вершины графа

3. Ввод цифры 0 завершает работу программы

Алгоритм полного перебора графа:

- Запускается стандартный поиск в ширину, если в течение пути элемент графа имеет меньше 3 исходящих ребер, то алгоритм заканчивается и возвращает ответ на поставленный вопрос, в ином случае рекурсивно проходится по каждому следующему элементу графа, начиная с левого, проверяет наличие 3 исходящих ребер, а также окрашивает его для того, чтобы не заиклиться.

Алгоритм Форда-Фалкерсона:



- Заключается в поиске максимальное пропускной способности выбранных ребер (далее исток и сток)

1. В остаточной сети находим любой путь из источника в сток. Если такого пути нет, останавливаемся.
2. Пускаем через найденный путь (он называется **увеличивающим путём** или **увеличивающей цепью**) максимально возможный

поток:

1. На найденном пути в остаточной сети ищем ребро с минимальной пропускной способностью .
2. Для каждого ребра на найденном пути увеличиваем поток на Δ , а в противоположном ему — уменьшаем на Δ .
3. Модифицируем остаточную сеть. Для всех рёбер на найденном пути, а также для противоположных (антипараллельных) им рёбер, вычисляем новую пропускную способность. Если она стала ненулевой, добавляем ребро к остаточной сети, а если обнулилась, стираем его.

Возвращаемся на шаг 2.

Алгоритм добавления :

- Проходимся по списку смежности, если вершина отсутствует, тогда добавляем ее, если максимальный размер равен значению `maxamount`, тогда перевыделяем память, а затем создаем дескриптор этой вершины и инициализируем первый ее элемент, в ином случае создаем дескриптор этой вершины и инициализируем первый ее элемент.

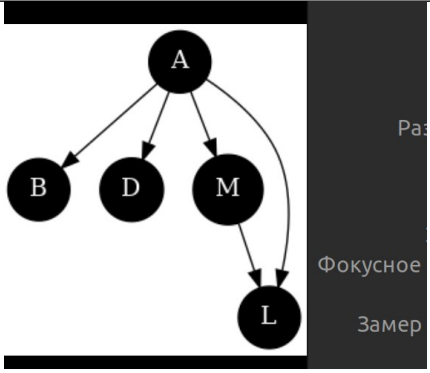
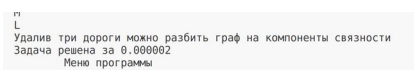
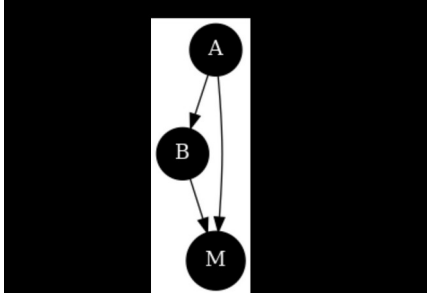
НАБОР ТЕСТОВ

(Негативные)

| № | Название теста | Пользовательский ввод | Вывод |
|---|-------------------|--|---------------------|
| 1 | Некорректный файл | (Файл содержит неправ. последователь.) | Ошибка чтения файла |

| | | | |
|---|---|-----------------------------------|---------------------------|
| 2 | Слишком большой граф | Граф больше 100 | Ошибка размера графа |
| 3 | Невозможно выделить память под элементы | Чрезмерное использование ресурсов | Ошибка выделения памяти ! |

(Позитивные)

| № | Название теста | Пользовательский ввод | Вывод |
|---|----------------------------|---|---|
| 1 | Добавление элемента | A L 10 |  |
| 2 | Алгоритм полного перебора | 4 |  |
| 4 | Инициализация нового графа | <pre> 1 choose -> 1 Введите название файла new.txt Успешно .. new.txt </pre> |  |

| | | | |
|---|---------------------------|---|---|
| 5 | Алгоритм Форда-Фалкерсона | 3 | <div> <div>PROBLEMS</div> <div>OUTPUT</div> <div>DEBUG CONSOLE</div> <div>TERMINAL</div> <div>PORTS</div> </div> <pre> 1 -> A 2 -> B 3 -> D 4 -> H 5 -> L Введите вершины для поиска (исток -> сток) 1 5 Удалив три дороги можно разбить граф на компоненты связности Задача решена за 0.000005 Меню программы </pre> |
|---|---------------------------|---|---|

ОЦЕНКА ЭФФЕКТИВНОСТИ (МС)

| | Полный перебор | Алгоритм Форда-Фалкерсона |
|------------------------|----------------|---------------------------|
| Полный граф (10) | 20 | 5 |
| Слабосвязный граф (10) | 10 | 3 |
| Полный граф 15 | 24 | 8 |
| Слабосвязный граф (15) | 16 | 4 |

Для оценки эффективности было проведено 100 расчётов и взято среднее время.

| | Список смежности (ребро) | Матрица смежности (ребро) |
|----------|--------------------------|---------------------------|
| Вставка | $\rightarrow O(1)$ | $O(N)$ |
| Удаление | $O(N)$ | $O(N)$ |
| Доступ | $O(N)$ | $O(N)$ |

ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое граф?

Граф – это конечное множество вершин и ребер.

2. Как представляются графы в памяти?

Графы представляются в памяти с помощью матриц смежности или с помощью списков смежности, также возможен симбиоз этих вариантов.

3. Какие операции возможны над графами?

Деление, окраска.

поиск кратчайшего пути от одной вершины к другой (если он есть);

поиск кратчайшего пути от одной вершины ко всем другим;

поиск кратчайших путей между всеми вершинами;

поиск эйлера пути (если он есть);

поиск гамильтонова пути (если он есть).

4. Какие способы обхода графов существуют?

Обход в глубину

Обход в ширину.

5. Где используются графовые структуры?

Графовые структуры используют при составлении дорог, сетей, при отслеживании позиции в играх, в комбинаторике.

6. Какие пути в графе Вы знаете?

Путь Эйлера, Гамильтонов путь.

7. Что такое каркасы графа?

Каркасом, или остовным деревом для этого графа называется связный подграф этого графа, содержащий все вершины графа и не имеющий циклов.

Вывод

Завершив лабораторную работу, я получил необходимые навыки работы с графами. В результате метод грубой силы, а именно перебор оказался в 2 раза медленнее, чем алгоритм Форда-Фалкерсона. Это произошло, потому что алгоритм Форда-Фалкерсона использует дополнительную памяти. Также на основании таблицы сравнения представлений графа оказалось, что Списки смежности работают быстрее, чем матрица смежности, однако визуальное представление матрицы смежности более простое, с ней удобнее (нагляднее работать).