|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **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**

«Графы»

**по курсу «Типы и структуры данных»**

Студент: Шимшир Эмирджан Османович

Группа: ИУ7-33Б

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Шимшир Э.О.

*подпись, дата фамилия, и.о.*

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Рыбкин Ю.А.

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Условие 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.

Найти все вершины графа, к которым от заданной вершины можно добраться по пути не длиннее А.

**Техническое задание**

Входные данные

Для корректной работы программы нужно загрузить данные из файла или ввести их с клавиатуры.

Выходные данные

Вывод графа и результатов измерения времени и поиска вершин.

Задачи, реализуемые программой

1. Загрузка данных их файла
2. Вывод графа с клавиатуры
3. Вывод графа в виде png картинки
4. Вывод матрицы графа
5. Поиск вершин графа, к которым от заданной вершины можно добраться по пути не длиннее А
6. Вывод сравнительной характеристики времени и памяти

Допущения

При вводе необходимо вводить только целые числа

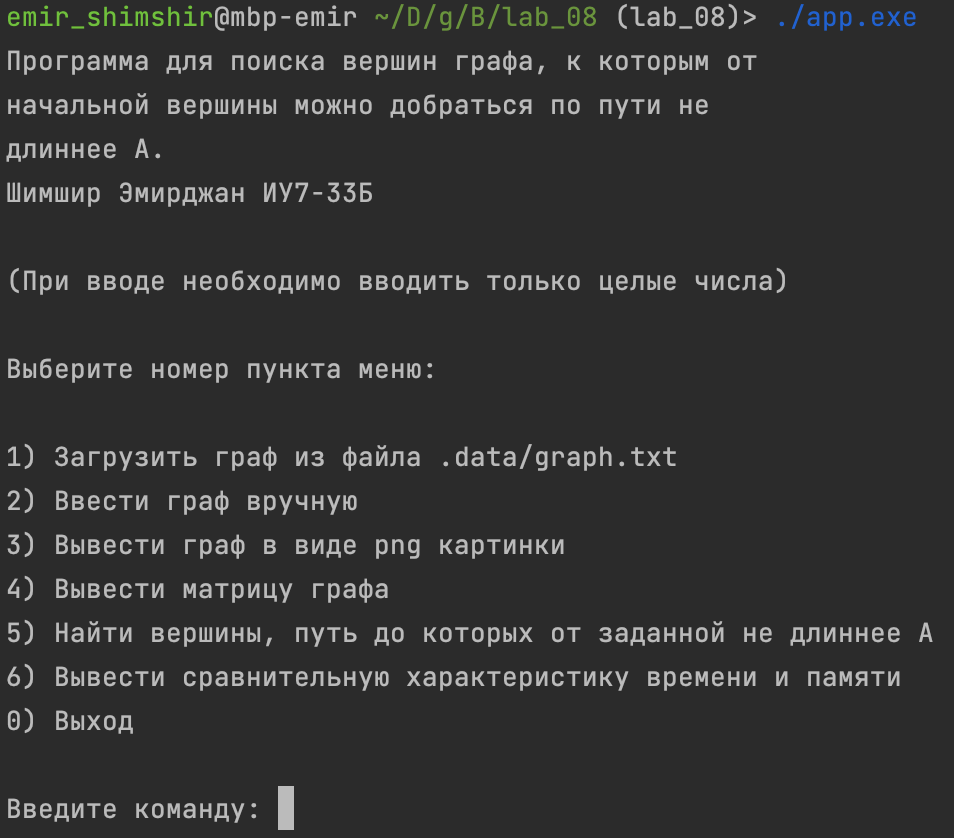
Описание внутренних структур данных

Структура для хранения матрицы смежности:

typedef struct  
{  
 int count;  
 int \*\*data;  
} matrix\_t;

*int count –* количество вершин графа  
*int \*\*data* - матрица смежности

Память под описанные выше структуры данных выделяется динамически методом массива указателей на строки

Описание меню и функций программы

1.

1. Пользователь загружает граф из файла ./data/graph.txt

2. Пользователь вводит граф вручную

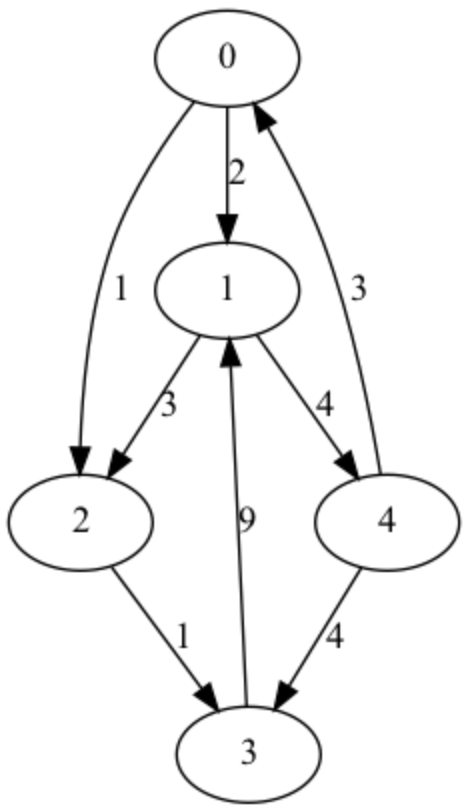
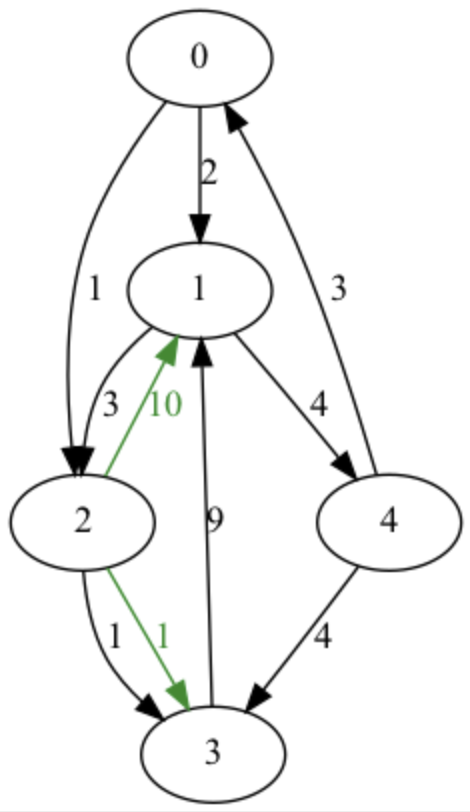
3. Пользователь выводит граф в виде png картинки

4. Пользователь выводит матрицу графа

5. Пользователь находит вершины графа, к которым от заданной вершины можно добраться по пути не длиннее А

6. Пользователь выводит сравнительной характеристики времени и памяти

Вывод графа в виде *.png* картинки (graphviz):

Описание функций

/\*  
\* Функция создает матрицу смежности  
\*  
\* Принимает количество вершин  
\*/  
matrix\_t \*create\_matrix(int count);

/\*  
\* Функция выделяет память для матрицы смежности  
\*  
\* Принимает количество вершин  
\*/  
int \*\*allocate\_matrix(int count);

/\*  
\* Функция освобождает память для матрицы смежности  
\*  
\* Принимает количество вершин и матрицу  
\*/  
void free\_matrix(int \*\*data, int count);

/\*  
\* Функция для вызова алгоритма Дейкстры  
\*  
\* Принимает матрицу, массив расстояний до вершин, начальная вершина  
\*/  
void Dijkstra(matrix\_t \*graph, int distance[graph->count], int top);

Описание алгоритма и исследование полученных результатов

Для представления графа в программе используется матрица смежности. Данные для заполнения матрицы смежности считываются из файла. Далее по полученной матрице создается файл .gv, содержащий представление графа на языке DOT. Данный файл открывается с помощью программы graphviz и создается png картинка.

У пользователя запрашивается номер вершины, от которой производится расчет кратчайших путей до каждой другой вершины с помощью алгоритма Дейкстры. Временная сложность алгоритма – **O()**. Также запрашивается максимальный путь А. Алгоритм Дейкстры основан на выборе для включения в путь всякий раз той вершины, которая имеет наименьшую оценку кратчайшего пути (по весам ребер), то есть наименьший путь до этой вершины из всех возможных путей, которые были рассмотрены ранее. Далее вершины, расстояние между которыми меньше, чем А, выводятся на экран, связи между вершинами записываются в файл с расширением .gv на языке DOT для визуализации и создается png картинка.Полученные результаты

Результаты измерения времени при поиске кратчайших путей от выбранной вершины до каждой другой с помощью алгоритма Дейкстры. При проведении экспериментов программа была скомпилирована без оптимизаций (**-O0**), внешние задачи отсутствовали.

Время измеряется в тактах, память – в байтах.

|  |  |  |
| --- | --- | --- |
| **Количество\*** | **Время** | **Память** |
| 10 | 5 | 88 |
| 50 | 36 | 408 |
| 100 | 119 | 808 |
| 500 | 2114 | 4008 |

**Тестирование**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Название теста** | **Пользовательский ввод** | **Результат** |
| 1 | Некорректный ввод команды | gdgfd | Неверная команда |
| 2 | Неверное количество вершин или вес ребра (пункт 2) | -4  fsdfsd | Неверный элемент |
| 3 | Вывод графа, вывод матрицы графа или поиск в несозданном графе (пункт 3/4/5) | Пустой ввод | Данные еще не были загружены |
| 4 | Верный пункт меню |  | Действие выполнено |
| 6 | Ввод максимально возможной длины, меньше, чем самый маленький путь (пункт 5) |  | Вывод путей и их длин.  Пути не отображаются |
| 7 | Ввод максимально возможной длины, большей, чем самый большой путь (пункт 5) |  | Вывод путей и их длин  Пути отображаются |
| 8 | Файла, который необходимо открыть, не существует (пункт 1/4/6) |  | Ошибка открытия файла |

Контрольные вопросы

***1. Что такое граф?***

Граф – это конечное множество вершин и ребер, соединяющих их, G = < V,E >, где V – конечное непустое множество вершин; Е – множество ребер (пар вершин).

Если пары Е (ребра) имеют направление, то граф называется ориентированным (*орграф*), если иначе - неориентированный (*неорграф*). Если в пары Е входят только различные вершины, то в графе нет петель. Если ребро графа имеет вес, то граф называется *взвешенным*.

Неорграф называется *связным,* если существует путь из каждой вершины в любую другую.

***2. Как представляются графы в памяти?***

Граф в памяти представляется в виде матрицы смежности или списка смежности.

*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.

***3. Какие операции возможны над графами?***

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 эйлерова пути (если он есть);

поиск гамильтонова пути (если он есть).

***4. Какие способы обхода графов существуют?***

*Обход в ширину (BFS – Breadth First Search)* - обработка вершины V осуществляется путём просмотра сразу всех «новых» соседей этой вершины, которые последовательно заносятся в очередь просмотра.

*Обход в глубину (DFS – Depth First Search)* - 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.

***5. Где используются графовые структуры?***

Графовые структуры могут использоваться в задачах, где между элементами могут быть установлены произвольные связи. Распространенное применение — решение задачах о путях.

***6. Какие пути в графе Вы знаете?***

*Эйлеровый путь* - путь в графе, проходящий через каждое ребро ровно один раз. Если путь проходит по некоторым вершинам несколько раз – он называется *непростым*, иначе – *простым*.

*Гамильтонов путь* - путь, проходящий через каждую вершину ровно один раз.

***7. 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.

Вывод

Для решения данной задачи был выбран способ хранения графа в виде матрицы смежности. Плюсы данного метода заключаются в возможности сразу визуально понять, насколько плотно связан граф, а также в легкости перевода на язык DOT связей вершин, что позволяет быстро получить графическое изображение графа.

Также для решения задачи использовался алгоритм Дейкстры – достаточно быстрый, особенно при небольших размерах графа (что было выявлено экспериментально), и простой метод нахождения кратчайших путей между вершинами. Алгоритм Дейкстры основан на выборе для включения в путь всякий раз той вершины, которая имеет наименьшую оценку кратчайшего пути (по весам ребер), то есть наименьший путь до этой вершины из всех возможных путей, которые были рассмотрены ранее.

Графы полезны при решении задач о путях или об иных связях и зависимостях. Они помогают при отображении и визуализации связей, что иногда является ключевым для быстрого решения задач.