|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | **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

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 7

По дисциплине «Типы и структуре данных»

### Название Графы

### Студент Мансуров Владислав Михайлович

*фамилия, имя, отчество*

### Группа ИУ7-36Б

Тип лабораторной работы Учебная

### Название

предприятия

|  |  |  |
| --- | --- | --- |
| Студент |  | Мансуров В.М. |
| Преподаватель | *подпись, дата* | *фамилия, и.о.*  Силантьева А.В. |
|  | *подпись, дата* | *фамилия, и.о.* |

*2021 г.*

***Содержание***

### Условие задачи 4

### Техническое задание… 4

### Описание алгоритма 5

### Анализ алгоритмов… 13

### Контрольные вопросы… 17

### Заключение… 19

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

***Условия задачи***

***Вариант 4***

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

***Техническое задание***

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.

***Входные данные***

В качестве указания пункта меню – целое число.

В подпунктах – по 1-2 целых числа

***Выходные данные***

При выводе графа – вывод данных из графа, либо матрицы, либо список смежности, содержащие целые числа.

При выводе графа – вывод граф, графически.

Данные о путей и вершина, подходящих под условие задачи(см. выше)

Данные об эффективности.

***Описание алгоритма:***

В реализации программы были созданы две структуры для хранения Дерева и массивом смежности:

*Матрица смежности*

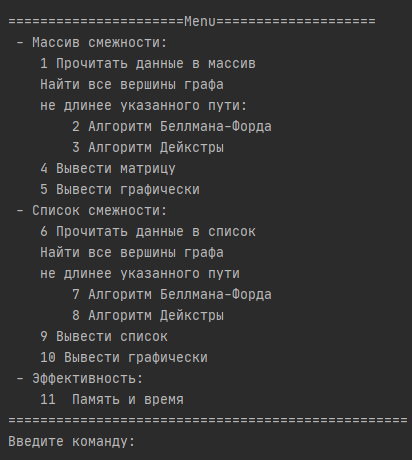
typedef struct {  
 size\_t n; // размер матрицы  
 int \*\*matrix; // матрица  
} st\_matrix;

*Список смежности*

typedef struct node {  
 int top; // вершина  
 int weight; // вес ребра  
 struct node \*next; // указатель на следующую вершину  
} st\_list;

typedef struct {  
 st\_list \*\*array; // массив списков  
 size\_t n; // размер массива  
}st\_array\_list;

Для взаимодействия с программой было создано консольное меню:



Название файл передается через командную строку, если оно корректно или такой файл существует и не пуст, то пользователю выводится меню для взаимодействие с программой.

При вводе корректного значения выполняются определенные действия или операции, при некорректном выводится в консоль сообщение об ошибке и предоставляется ввести пункт еще раз. Так происходит пока не будет введен корректное значение меню.

Для реализации программы, была использована статическая и динамическая типизация, поэтому были написаны функции выделения памяти и очищения памяти, чтобы при работе не возникали утечки памяти.

Если граф пуст – во всех пунктах кроме 1 и 6 – Выводится сообщение об этом.

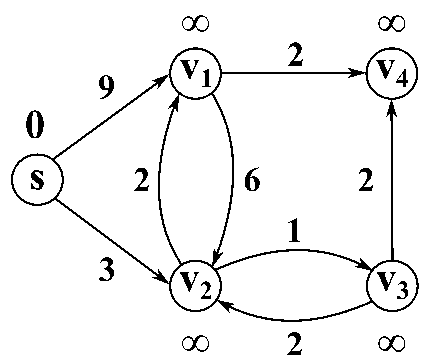
*Пункта 1 консольного меню:*

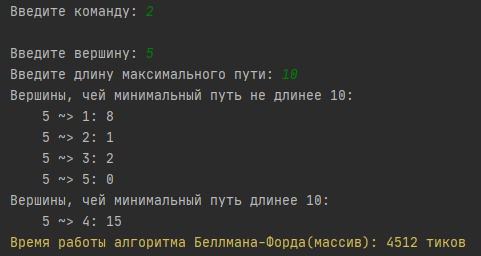
В этом пункте данные читаются из файл в бинарное дерево. При успешном чтении дерева, получаем сообщение об этом, при повторном чтении файл не считывается.

*Пункта 2 и 7 консольного меню:*

В этом пункте происходит поиск всех вершин, чей путь от заданной вершины не длиннее указанного пути, алгоритмом Беллмана-Форда.

Алгоритм Беллмана-Форда с асимптотикой O(VE). Алгоритм находит кратчайшие пути от одной вершины графа до всех остальных, учитывая как положительный, так и отрицательный вес ребер, дуг, петель графа.

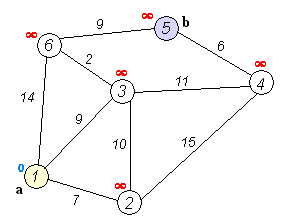


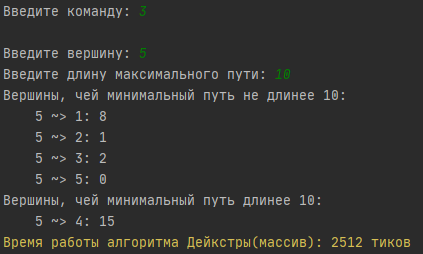


*Пункта 3 и 8 консольного меню:*

В этом пункте происходит поиск всех вершин, чей путь от заданной вершины не длиннее указанного пути, алгоритмом Дейкстры.

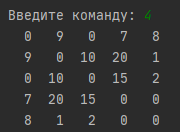
Алгоритм Дейкстры с асимптотикой O(Vlog(V)). Алгоритм находит кратчайшие пути от одной вершины графа до всех остальных, учитывая только вес ребер, дуг, петель графа.

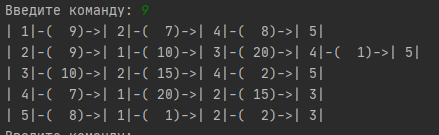




*Пункта 4 и 9 консольного меню:*

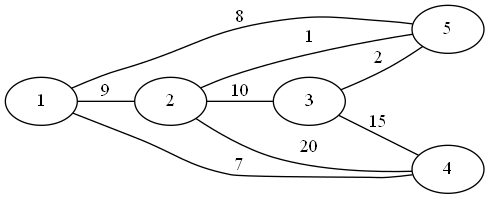
Вывод в виде матрица или списка.





*Пункта 5 и 10 консольного меню:*

Вывод графа графически.



*Пункт 0 консольного меню* Выход из программы и очищение памяти дерева и хеш-таблицы

Функции для работы с матрицей

//функция устанавливает значения структуры по умолчанию

void default\_matrix\_graf(st\_matrix \*graf);

// очищение матрицы

void free\_matrix(int \*\*\*matrix, size\_t n);

//выделение памяти под матрицу  
int \*\*allocate\_matrix(size\_t n, size\_t m);

// чтение из файла в матрицу

int read\_file\_matrix(FILE \*f, st\_matrix \*graf);

// алгоритмы нахождение путей, работающие с матрицей

int algoritm\_ballman\_fold\_matrix(const st\_matrix \*graf, int start\_top, int \*distance);  
int algoritm\_dijkstra\_matrix(const st\_matrix \*graf, int start\_top, int \*distance);

// вывод матрицы

void print\_graphviz\_matrix(FILE \*graph\_visual, st\_matrix \*matrix) ;  
void print\_matrix(const st\_matrix \*graf);

Функции работы с списком

//функция устанавливает значения структуры по умолчанию

void default\_list\_graf(st\_array\_list \*graf);

// выделение памяти под массив списков

st\_list \*\*alocate\_array\_list(size\_t n);

// создание записи списка  
st\_list \*top\_create(int top, int weight);

// добавление записи в список  
int list\_push(st\_list \*\*head, int top, int weight);

// очищение списка  
void free\_list(st\_list \*\*head);

//очищение массив списка  
void free\_array\_list(st\_array\_list \*graf);

// чтение файла в список смежности

int read\_file\_list(FILE \*f, st\_array\_list \*graf);

// алгоритмы нахождение путей, работающие с списком смежности

int algoritm\_ballman\_fold\_list(const st\_array\_list \*graf, int start\_top, int \*distance);  
int algoritm\_dijkstra\_list(const st\_array\_list \*graf, int start\_top, int \*distance);

// вывод списка смежности

void print\_graphviz\_list(FILE \*graph\_visual, const st\_array\_list \*graf);  
void print\_array\_list(const st\_array\_list \*graf);

***Аварийные выходы из программы***

Выходом из программы является некорректное чтение файла, несуществующий файл или пустой файл.

При взаимодействии с меню – ошибка выделения памяти под дерево или хеш-таблицу.

Выходов аварийных нет, в том случае, если программа не будет принудительна закрыта.

Выход осуществляется только пунктом меню «0»

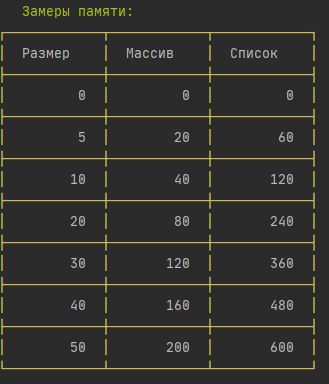
В ошибочных случаях выводится сообщение об ошибке и ввод повторяется, пока программа не получит корректные данные.

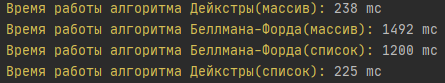
***Анализ результатов эффективности***

*Запускается пунктом меню – 11*

*Замеры по памяти:*

Замерив память работы от 10 – 50 элементов, где хранить матрицу смежности эффективнее, чем хранить список смежности (50-70%)





Используя два алгоритма поиска кратчайшего пути от заданной вершины, были получены результаты, по которым видно, что список смежности работает быстрее матрица смежности (30-50%).

При сравнение работы двух алгоритмов, алгоритм Дейкстры работает намного быстрее, чем алгоритм Беллмана-Форда, и для решение моей задачи подходит полностью, так он быстрее и в структурах не используется отрицательный вес ребер.

***Контрольные вопросы***

**Что такое граф?**

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

**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.

Пример:

Схемы авиалиний, схемы дорог

**Какие пути в графе Вы знаете?**

Эйлеров путь, непростой путь, гамильтонов путь.

**Что такое каркасы графа?**

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

***Вывод***

Для решения поставленной задачи был реализован алгоритм Форда-Беллмана и Дейкстры для поиска кратчайших путей в графе. Выбор обоснован тем, что вес рёбер может быть в том числе отрицательным (но программа не работает с таким весом), при этом, например, в алгоритме Дейкстры это не разрешено. Кроме того, алгоритм обнаруживает отрицательный цикл (цикл, сумма весов рёбер которого отрицательна, называется отрицательным циклом.)

Так же были рассмотрены матрица смежности и список смежности, где по памяти эффективнее работать с матрицей, а по времени с списком.

Если в случае, когда нам нужно скорость работы более нужнее для программ, то используем список смежности, а уже если не важно на время работы программы, но при этом памяти программы хотелось бы оптимизировать, то использовать матрицу.