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)

Отчет

по лабораторной работе №7
по теме
«Графы»
Вариант 11.

Дисциплина: Типы и структуры данных

Студент ИУ7-31Б:
Косарев Алексей
Проверила:

Москва, 2021

1. 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.

Вариант 11:

Задана система двусторонних дорог. Определить, можно ли, закрыв какие-нибудь три дороги, добиться того, чтобы из города A нельзя было попасть в город B.

2. Описание ТЗ

- Описание исходных данных

Исходными данными являются числовые данные, введенные пользователем, для выбора пунктов меню, а также описание графа (связи вершин).

Меню:

1. Ввод данных о графе
2. Ввод данных о графе из файла
3. Вывод матрицы смежности графа и экспорт графа в DOT file
4. Выбор двух вершин для задания
5. Удаление трех дорог
6. Анализ времени и памяти

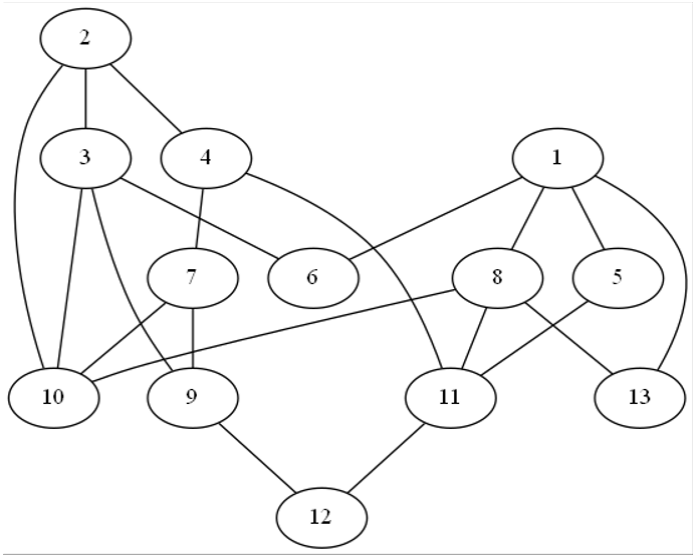
0. Выход из программы

- Описание выходных данных

- матрица смежности графа
- DOT файл с информацией о графе, из которого можно сделать png файл
- вывод результата поиска 3 дорог для удаления
- анализ работы программы по времени и анализ памяти

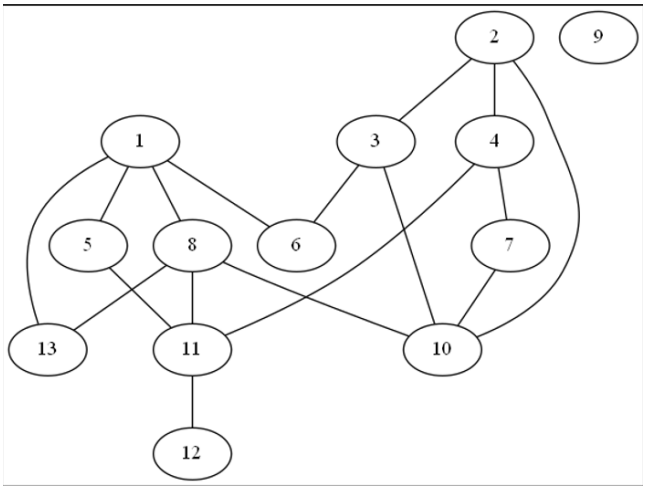
- Пример работы и расчеты

1)Граф:



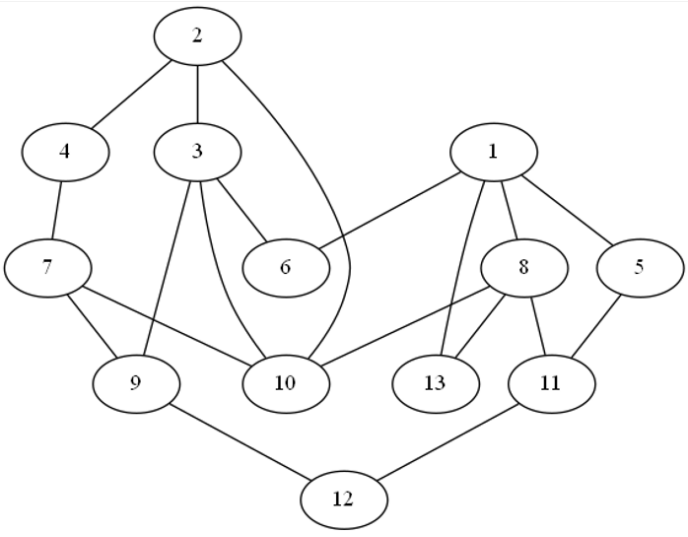
Варианты работы для данного графа:

- ```
  Input number of A node:  You need to delete these 3 roads:
  1                        3 -- 9
  Input number of B node:  7 -- 9
  9                        9 -- 12
  ```

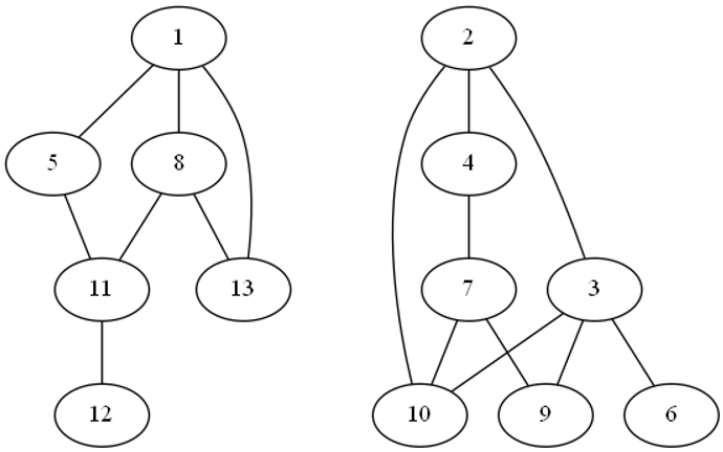



- ```
  Input number of A node:
  1
  Input number of B node:
  10                       Cannot find 3 roads to delete.
  ```

2) Граф:



Варианты работы для данного графа:

- ```
  Input number of A node: You need to delete these 3 roads:
  1                       1 -- 6
  Input number of B node: 8 -- 10
  10                      9 -- 12
  ```
  

- ```
  Input number of A node:
  8
  Input number of B node:
  1                       Cannot find 3 roads to delete.
  ```

3. Описание задачи, реализуемой в программе

Программа реализует ввод данных о графе (как из консоли, так и из файла), а именно - она запрашивает количество вершин графа и затем для каждой вершины - список смежных ей. Также необходимо ввести вершину А и вершину Б для которых и будет произведена основная операция поиска 3 дорог для удаления, отсутствие которых не позволит попасть из одного города в другой. Присутствует также возможность вывода графа в формате матрицы смежности и в графическом виде (экспорт в DOT файл).

Вариант реальной задачи, для которой можно использовать данную программу:
Между городами нужно отремонтировать дороги в срочном порядке. Строителям необходимо определить, какие три дороги они точно не должны перекрывать одновременно, чтобы не нарушать логистику между городами.

4. Способ обращения к программе

Обращение к программе происходит через консоль, путем запуска файла с расширением .exe (./main.exe).

5. Описание возможных аварийных ситуаций и ошибок пользователя

```
// Коды ошибок
#define OK 0                                // Нет ошибок
#define FILE_OPEN_ERR 4                     // Ошибка чтения файла
#define FILE_CLOSE_ERR 5                    // Ошибка закрытия файла
#define FILE_ERR 6                          // Ошибка размера и данных файла
#define INCORRECT_INPUT 7                   // Некорректный ввод
```

Также при некорректном вводе (например, вводе буквы вместо номера вершины или вводе несуществующего пункта меню) программа будет запрашивать ввод у пользователя, пока тот не введет корректное значение.

6. Описание внутренних структур данных

```
#define MAX 100                             // Максимальный размер матрицы
```

- Структура представления графа в виде матрицы:

```
typedef struct
{
   int matrix[MAX][MAX];  // Матрица смежности
   int n;                 // Количество городов
   int node_A;            // Город отправления
   int node_B;            // Город назначения
   int visited[MAX];      // Массив посещенных городов
} graph_t;
```

- Структура сохранения существующих дорог:

```
typedef struct
{
   int rows[MAX];         // Индекс города A
   int columns[MAX];      // Индекс города B (соответствующего городу A)
   int n;                 // Количество дорог
} roads_t;
```

7. Алгоритмы

Алгоритм обхода графа - поиск в глубину:

1. Заходим в какую-либо смежную вершину
2. Из этой вершины обходим все возможные пути до смежных вершин
3. Если таких путей нет или мы не достигли конечной вершины, то возвращаемся назад к вершине с несколькими исходящими ребрами и идем по другому пути
4. Алгоритм повторяется, пока не будут исследованы все вершины или не будет достигнута конечная вершина

8. Тесты

Положительные тесты:

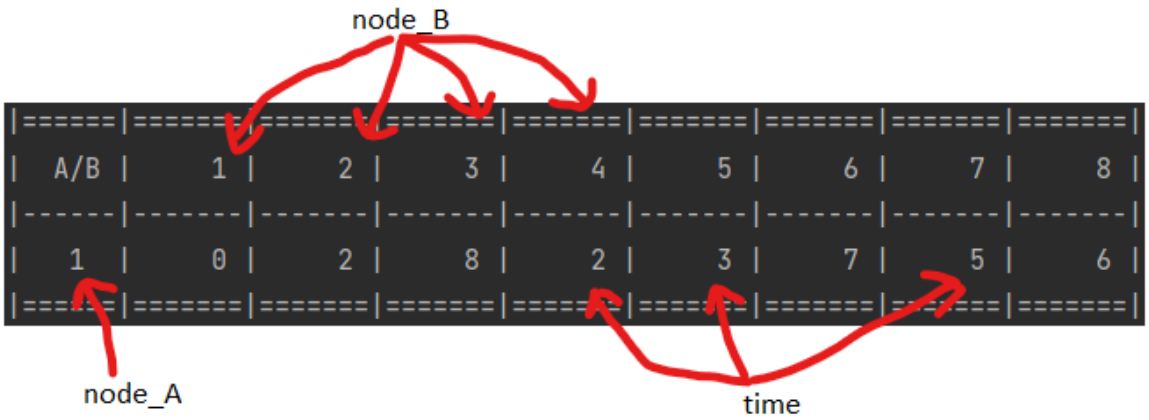
1. Граф из одной вершины
2. Граф из двух вершин
3. Граф из двух вершин
4. Граф, в котором для всех пар вершин нельзя найти три дороги для удаления
5. Граф, в котором для всех пар вершин можно найти три дороги для удаления
6. Граф, в котором есть пары вершин, для которых можно удалить три дороги и для которых невозможно это сделать

Негативные тесты:

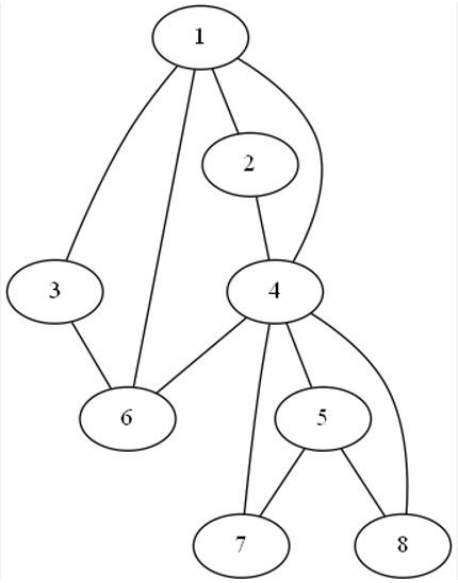
1. Некорректный выбор пункта меню (ввод буквы).
2. Некорректный выбор пункта меню (ввод несуществующего номера пункта).
3. Некорректный ввод номера вершины (ввод буквы).
4. Некорректный ввод номера вершины (ввод отрицательного значения).
5. Некорректный ввод номера вершины (ввод несуществующего значения).

9. Временная эффективность и затраты памяти

Таблицы устроены следующим образом:



1) Граф:



Время поиска в глубину

TIME OF DFS

A/B	1	2	3	4	5	6	7	8
1	0	2	8	2	3	7	5	6

A/B	1	2	3	4	5	6	7	8
2	1	0	2	5	6	3	7	8

A/B	1	2	3	4	5	6	7	8
3	1	2	1	3	4	8	6	7

A/B	1	2	3	4	5	6	7	8
4	2	3	4	1	14	5	15	16

A/B	1	2	3	4	5	6	7	8
5	3	4	5	2	1	6	22	23

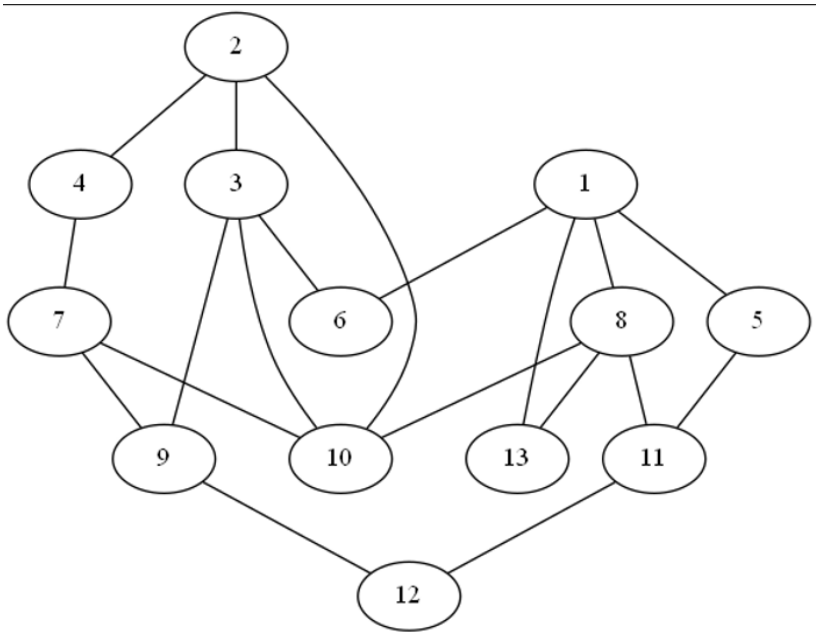
A/B	1	2	3	4	5	6	7	8
6	2	3	15	4	5	1	6	7

A/B	1	2	3	4	5	6	7	8
7	3	4	5	2	15	6	1	16

Время поиска дорог для удаления

TIME OF FINDING ROADS TO DELETE								
A/B	1	2	3	4	5	6	7	8
1	1	36	46	55	85	393	99	117
A/B	1	2	3	4	5	6	7	8
2	14	1	20	8	11	17	13	15
A/B	1	2	3	4	5	6	7	8
3	15	38	1	16	22	11	27	29
A/B	1	2	3	4	5	6	7	8
4	71	23	37	1	1879	78	441	513
A/B	1	2	3	4	5	6	7	8
5	81	58	47	637	1	84	481	696
A/B	1	2	3	4	5	6	7	8
6	212	38	48	73	97	1	110	127
A/B	1	2	3	4	5	6	7	8
7	88	40	52	155	406	93	1	497

2) Граф:



Время поиска в глубину

TIME OF DFS

A/B	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	7	9	16	2	11	14	5	12	6	3	18	62
A/B	1	2	3	4	5	6	7	8	9	10	11	12	13
2	5	1	2	16	6	4	14	10	17	12	8	19	21
A/B	1	2	3	4	5	6	7	8	9	10	11	12	13
3	14	2	1	4	12	16	5	18	6	19	10	8	21
A/B	1	2	3	4	5	6	7	8	9	10	11	12	13
4	6	2	4	1	8	5	16	12	18	14	10	20	21
A/B	1	2	3	4	5	6	7	8	9	10	11	12	13
5	2	6	5	8	1	4	10	17	12	19	16	14	20
A/B	1	2	3	4	5	6	7	8	9	10	11	12	13
6	2	10	12	17	4	1	16	6	14	8	5	19	61
A/B	1	2	3	4	5	6	7	8	9	10	11	12	13
7	8	4	5	2	10	6	1	14	20	16	12	19	18

Время поиска дорог для удаления

TIME OF FINDING ROADS TO DELETE

A/B	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	2439	2528	961	421	304	3858	11053	3825	3328	2819	2671	2512
A/B	1	2	3	4	5	6	7	8	9	10	11	12	13
2	3493	1	2467	1245	371	233	6920	3560	6697	7032	2604	2791	1051
A/B	1	2	3	4	5	6	7	8	9	10	11	12	13
3	4505	2868	1	677	311	390	6247	4322	11660	14083	2856	2472	1209
A/B	1	2	3	4	5	6	7	8	9	10	11	12	13
4	904	364	450	1	429	304	1078	1029	1165	1067	1350	1517	1358
A/B	1	2	3	4	5	6	7	8	9	10	11	12	13
5	163	131	156	210	2	298	201	66	203	115	53	246	252
A/B	1	2	3	4	5	6	7	8	9	10	11	12	13
6	129	46	31	56	113	2	57	114	68	109	102	97	132
A/B	1	2	3	4	5	6	7	8	9	10	11	12	13
7	5198	4367	6199	651	399	305	2	5247	12059	11879	3147	3563	1202

10. Вывод

В процессе выполнения данной лабораторной работы я научился работать с графами.

Для представления графа в памяти я использовал матрицу, потому что:

1) обработка матрицы требует меньше времени, чем обработка списка и производится она проще;
2) в алгоритме часто приходится проверять, связаны ли между собой две вершины;

В то же время матрицу не совсем рационально использовать с точки зрения экономии памяти, кроме тех случаев, когда число вершин невелико или число ребер довольно большое.

Для обхода графа я использовал алгоритм поиска в глубину. Мой выбор обусловлен тем, что в моей задаче мне нужно определять, есть ли хотя бы какой-нибудь путь из одного города в другой, то есть мне не обязательно искать самый короткий путь. Также преимуществом этого подхода является то, что если вершина, до которой ищется путь, находится далеко от заданной, то при данном методе путь найдется быстрее, чем при поиске в ширину (если эта вершина находится на пути первых порядковых вершин – лексикографически первый путь). Для реализации алгоритма поиска в глубину используется рекурсия, что удобно.

Время выполнения DFS составляет O(V + E), где V — общее количество вершин. E — общее количество ребер.

11. Ответы на контрольные вопросы

1. Что такое граф?

   Граф - это совокупность двух конечных множеств: множества точек и множества линий, попарно соединяющих некоторые из этих точек.
   Множество точек называетсервя вершинами (узлами) графа. Множество линий, соединяющих вершины графа, называются ребрами (дугами) графа.

2. Как представляются графы в памяти?

   Графы в памяти могут представляться различным способом. Один из видов представления графов – это матрица смежности B(n*n); В этой матрице элемент b[i,j]=1, если ребро, связывающее вершины Vi и Vj существует и b[i,j]=0, если ребра нет.
   Второй вид представления графов - это список смежностей. Список смежностей содержит для каждой вершины из множества вершин V список тех вершин, которые непосредственно связаны с этой вершиной.

3. Какие операции возможны над графами?

   • поиск кратчайшего пути от одной вершины к другой;
   • поиск кратчайшего пути от одной вершины ко всем другим;
   • поиск кратчайших путей между всеми вершинами;
   • поиск эйлерова пути;
   • поиск гамильтонова пути;

4. Какие способы обхода графов существуют?

   Двумя основными алгоритмами обхода графа являются поиск в глубину (Depth-First Search, DFS) и поиск в ширину (Breadth-First Search, BFS).
   Алгоритм DFS описан выше в отчете.
   BFS работает по принципу уровней, вместо того, чтобы двигаться по определенному пути до конца, BFS предполагает движение вперед по всем вершинам на одном уровне за раз.

5. Где используются графовые структуры?

   Типичное применение графовых структур, а именно остовных деревьев минимальной стоимости – это построение коммуникационных линий между городами, где стоимости ребер – это стоимость коммуникационных сетей.

6. Какие пути в графе Вы знаете?

   Эйлеров путь - произвольный путь в графе, проходящий через каждое ребро графа точно один раз.

   Гамильтонов путь - путь в графе, проходящий в точности один раз через каждую вершину графа (а не каждое ребро).

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. Количество ребер в каркасе связного графа всегда на единицу меньше количества вершин графа.