

ФАКУЛЬТЕТ: Информатика и системы управления

КАФЕДРА: Программное обеспечение ЭВМ и информационные технологии

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7**

**«**Графы**»**

Студент: Зайцева Алена Андреевна

Группа ИУ7 – 32Б

*2020 г.*

Оглавление

Оглавление

[1. Описание условия задачи 2](#_Toc58933515)

[2. Описание ТЗ 3](#_Toc58933516)

[1) Описание исходных данных: 3](#_Toc58933517)

[2) Описание результатов: 3](#_Toc58933518)

[3) Описание задачи, реализуемой программой: 3](#_Toc58933519)

[4) Способ обращения к программе: 3](#_Toc58933520)

[5) Описание возможных аварийных ситуаций и ошибок пользователя. 3](#_Toc58933521)

[3. Описание внутренних структур данных 3](#_Toc58933522)

[4. Описание алгоритма 4](#_Toc58933523)

[5. Набор тестов с указанием, что проверяется 4](#_Toc58933524)

[1) Негативные тесты на ввод пользователя 4](#_Toc58933525)

[2) Негативные тесты на функционал программы 5](#_Toc58933526)

[3) Позитивные тесты на функционал программы 5](#_Toc58933527)

[6. Оценка эффективности 8](#_Toc58933528)

[7. Ответы на вопросы 9](#_Toc58933529)

# Описание условия 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.

Найти минимальное (по количеству ребер) подмножество ребер, удаление которых превращает заданный связный граф в несвязный.

# Описание ТЗ

Внешняя спецификация:

## Описание исходных данных:

* Количество вершин в графе n (целое, >0)
* Ребра графа в формате “v1 v2” (петли не допускаются). V1 и v2 принимают целые значения от 0 до n-1

## Описание результатов:

* минимальное количество ребер, удаление которых превращает заданный связный граф в несвязный.
* Графическое изображение полученного несвязного графа, где зеленым цветом отмечены удаленные ребра
* Время выполнения программы в тактах процессора

## Описание задачи, реализуемой программой:

* Находит минимальное (по количеству ребер) подмножество ребер, удаление которых превращает заданный связный граф в несвязный и визуализирует результат

## Способ обращения к программе:

Для сборки исполняемого файла в папке есть файл makefile. Чтобы собрать исполняемый файл app.exe нужно написать команду “make app.exe”. Программа запускается из папки, содержащей исполняемый файл app.exe командой “./app.exe”.

## Описание возможных аварийных ситуаций и ошибок пользователя.

Возможные ошибки пользователя:

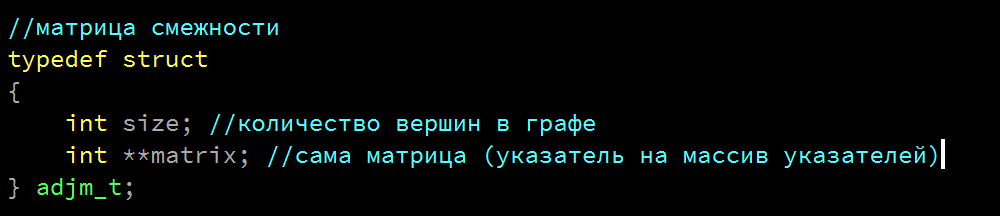
1. Ввод неверного количество вершин графа n (нецелое, или отрицательное, или нечисловое значение)
2. Ввод неверного ребра графа (одна из вершин - нецелое, или отрицательное, или нечисловое значение, или >= n, или ввод петли)
3. Ввод графа из 1 вершины
4. Ввод уже несвязного графа

Возможные аварийные ситуации:

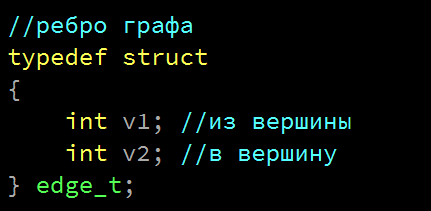
1. Программа не сможет выделить достаточное количество памяти для реализации задачи.

# Описание внутренних структур данных

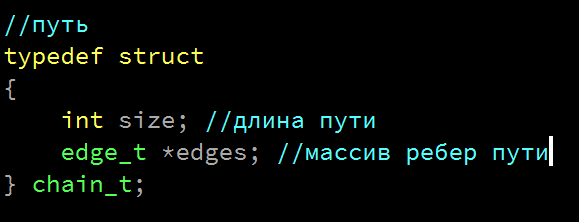
1. Матрица смежности



1. Ребро графа



1. Путь



Выбрана именно матрица смежности (а не список смежности), а также представление пути в виде массива ребер, так как это наиболее удобные и быстрые в обращении структуры данных.

## Описание алгоритма

Программа последовательно реализует следующие действия:

1. Вводится количество вершин графа и сам граф.
2. Граф проверяется на связность и, если он уже не является связным, выводится соответствующее сообщение и граф в исходном состоянии, а программа завершается.

Так как программа обрабатывает неориентированный граф, то для проверки его на связность достаточно выполнить поиск в глубину для любой вершины. Тогда, если в результате поиска была посещена каждая вершина графа, граф является связным, иначе – несвязным.

1. Программа последовательно пытается удалить из графа все комбинации по m ребер, где m принадлежит (1, 2, … n) так, чтобы граф стал несвязным.
2. Выводится минимальное количество ребер, которые нужно удалить для решения задачи, время выполнения программы и графическая визуализация полученного графа.

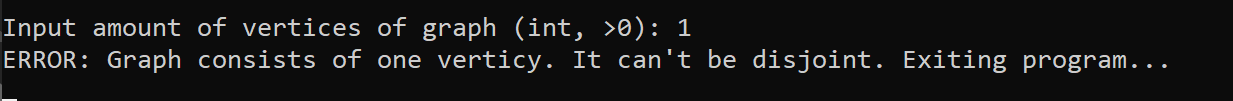
# Набор тестов с указанием, что проверяется

## Негативные тесты на ввод пользователя

|  |  |  |
| --- | --- | --- |
| Что проверяет | Ввод | Результат |
| Неверное количество вершин графа | 0 или -2 или а или .1 | ERROR: incorrect amount of vertices. Try again: |
| Неверное ребро графа (при n = 3) | (0, -1) или (1, 1) или (0, 3) или (0, .1) или (0, a) | ERROR: incorrect edge. Try again: |

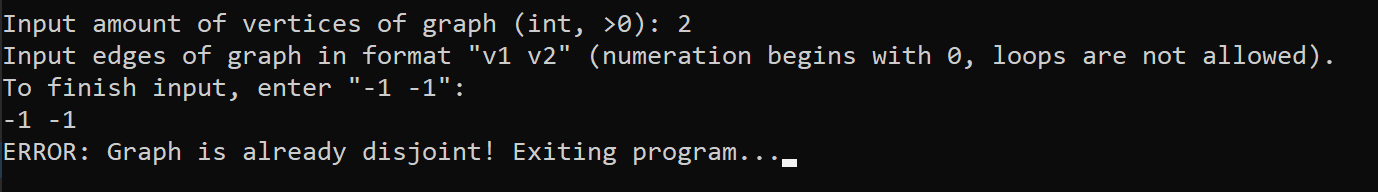
## Негативные тесты на функционал программы

В графе всего одна вершина



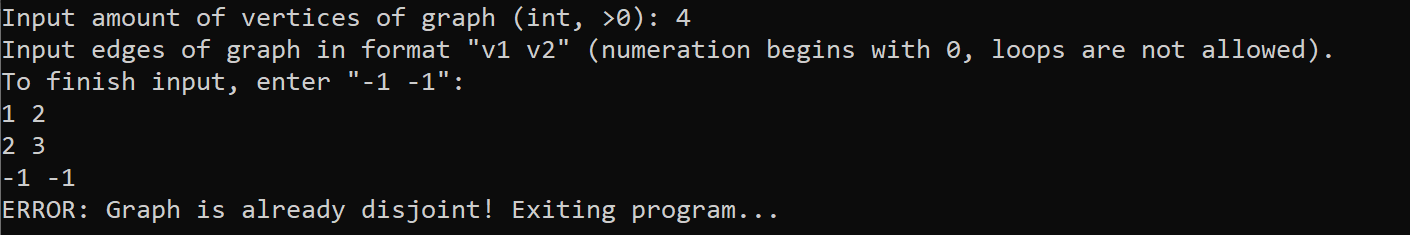
Граф уже несвязный

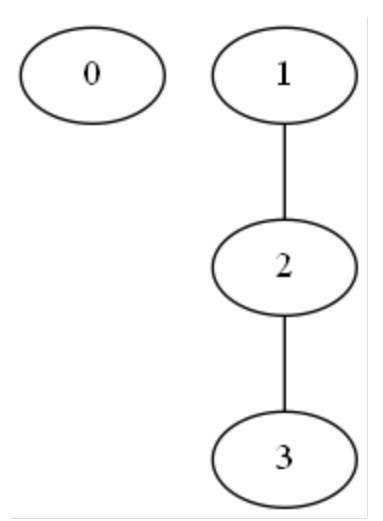
Вообще нет ребер





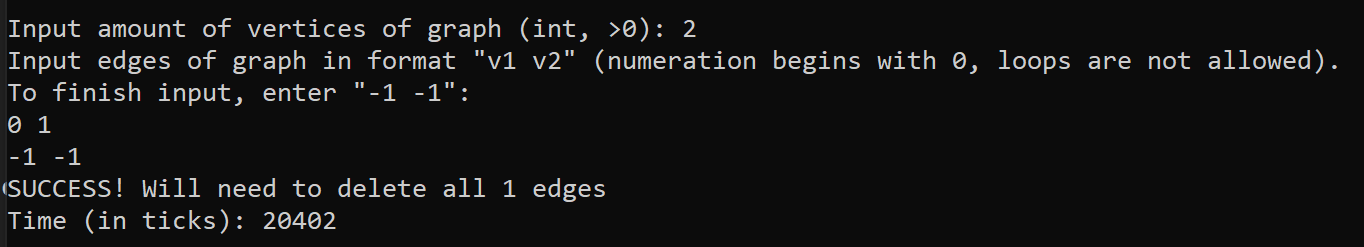
Есть некоторые ребра

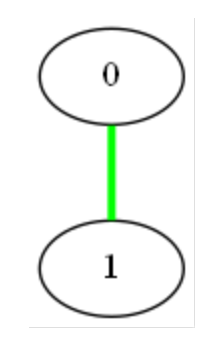




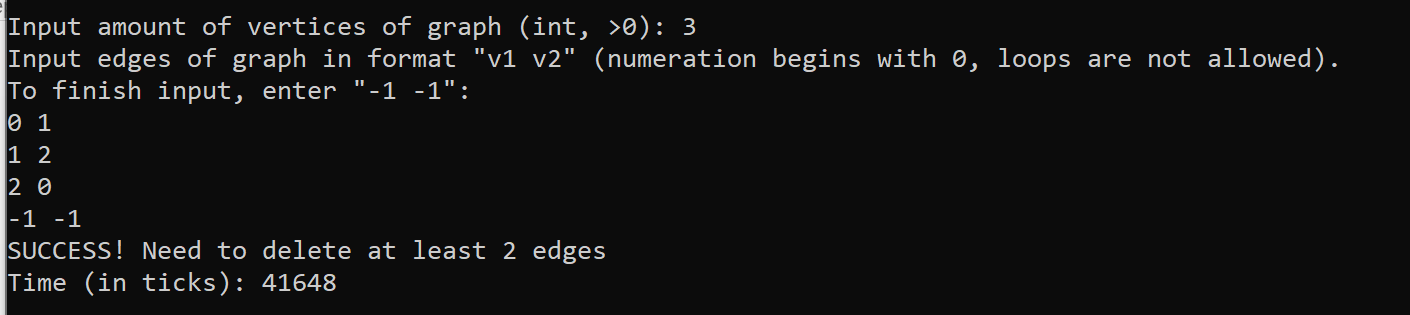
## Позитивные тесты на функционал программы

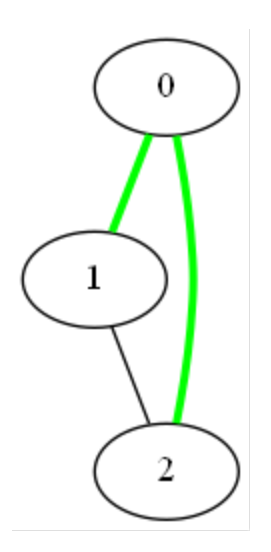
Придется удалить все ребра



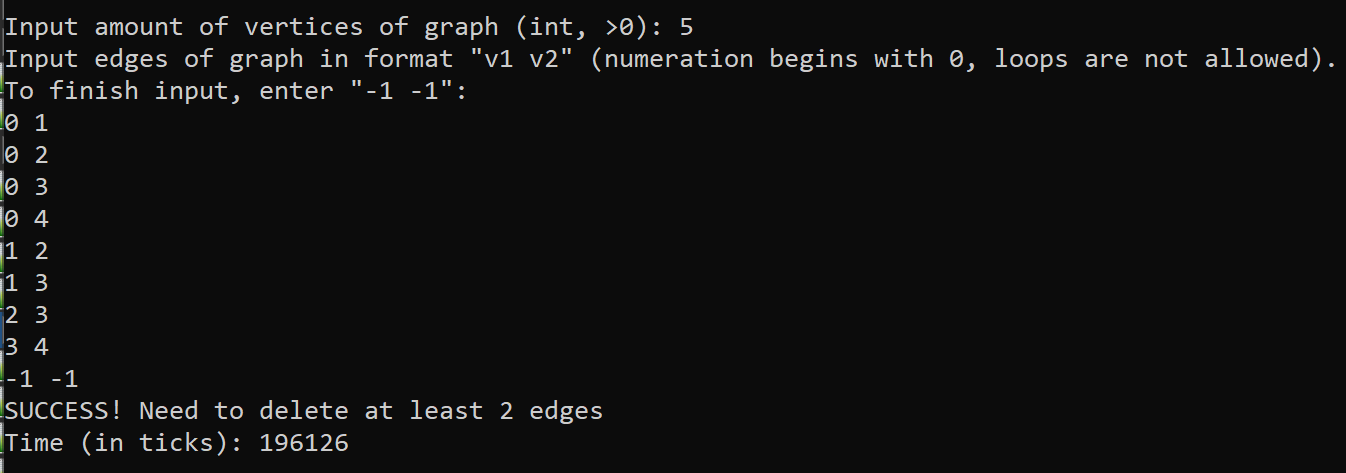


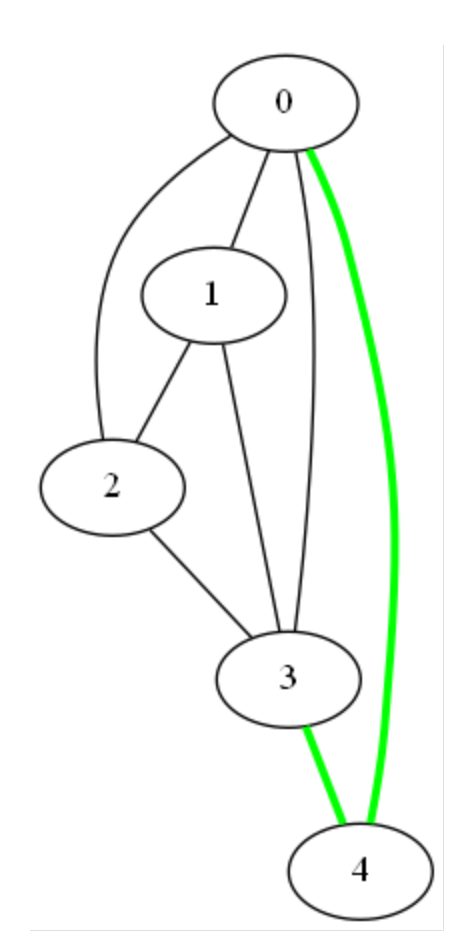
Нужно удалить 2 ребра из 3

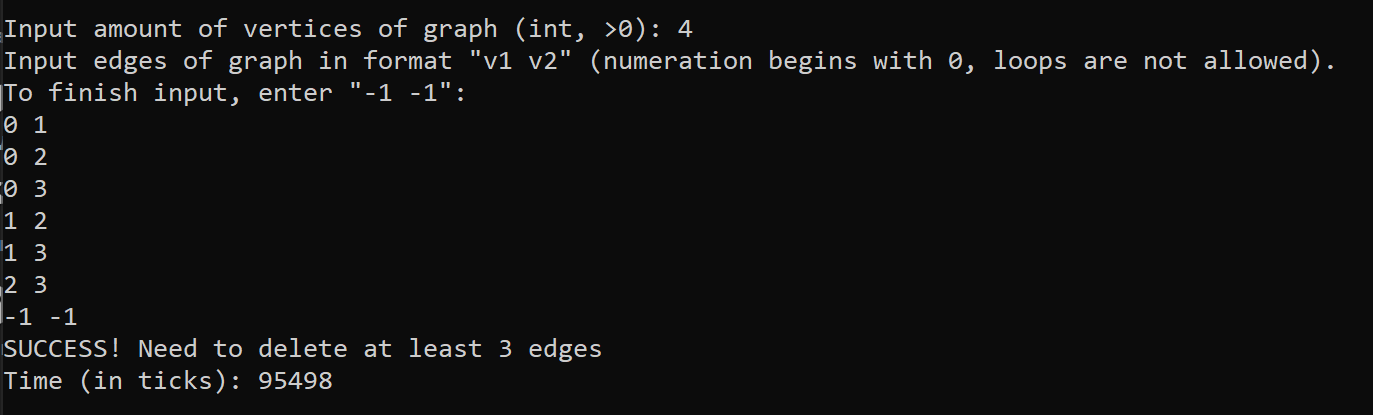


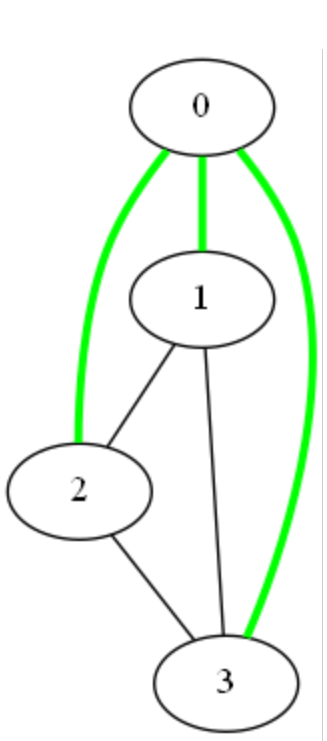


Более сложные графы









# Оценка эффективности

**Время выполнения задачи** (в тактах процессора, все вершины связаны «по кругу»)

|  |  |
| --- | --- |
| Количество элементов | Время |
| 5 | 105.112 |
| 10 | 199.434 |
| 50 | Не хватает памяти |

**Объем занимаемой памяти** (байт, только для матрицы смежностей)

16 + 8\*n + 4\*n\*n

|  |  |
| --- | --- |
| Количество вершин | Память |
| 5 | 156 |
| 10 | 496 |
| 50 | 10416 |

# Ответы на вопросы

1. Что такое граф?

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

1. 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. У неориентированных графов матрица смежности всегда симметрична.

Во многих случаях удобнее представлять граф в виде так называемого списка смежностей. Список смежностей содержит для каждой вершины из множества вершин V список тех вершин, которые непосредственно связаны с этой вершиной. Каждый элемент (ZAP[u]) списка смежностей является записью, содержащей данную вершину и указатель на следующую запись в списке (для последней записи в списке этот указатель – пустой). Входы в списки смежностей для каждой вершины графа хранятся в таблице (массиве) (BEG [u])

1. 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).

1. Какие способы обхода графов существуют?

Один из основных методов проектирования графовых алгоритмов – это поиск (или обход графа) в глубину (depth first search, DFS), при котором, начиная с произвольной вершины v0, ищется ближайшая смежная вершина v, для которой, в свою очередь, осуществляется поиск в глубину (т.е. снова ищется ближайшая, смежная с ней вершина) 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. Иными словами, поиск в глубину из вершины v основан на поиске в глубину из всех новых вершин, смежных с вершиной v

Другой способ - поиск в ширину (breadth first search, BFS). Обработка вершины v осуществляется путем просмотра сразу всех новых соседей этой вершины. При этом полученный путь является кратчайшим путем из одной вершины в другую.

1. Где используются грифовые структуры?

Графовые структуры могут использоваться в задачах, в которых между элементами могут быть установлены произвольные связи, необязательно иерархические.

Например, типичное применение остовых деревьев минимальной стоимости - это построение коммуникационных линий между городами, где стоимости ребер – это стоимость коммуникационных сетей.

1. Какие пути в графе Вы знаете?

Простой путь, сложный путь, Эйлеров путь, Гамильтонов путь.

1. Что такое каркасы графа?

Остовное дерево или каркас (скелет) графа – это подграф, который:  
1) содержит все вершины графа,  
2) является деревом (Дерево — это связный ациклический граф)