Федеральное государственное автономное образовательное учреждение

«Санкт-Петербургский государственный электротехнический университет

«ЛЭТИ» им. В.И. Ульянова (Ленина)»

Отчет

по практическому заданию

Выполнили:

Диогенова К.А.

Казачкова А.Д.

Факультет: КТИ

Группа №3304

Преподаватель:

Фирсов М.А.

Подпись преподавателя

Санкт-Петербург

2015

1. **Формулировка задания**

Разработать программу, визуализирующую алгоритм нахождения минимального остовного дерева алгоритмом Крускала, на языке Java с использованием библиотек “Swing”, “AWT”. Программа должна иметь простой в использовании, удобный интерфейс.

1. **Спецификация**
2. *Внешняя*

Граф задается в файле. Работа алгоритма будет продемонстрирована следующим образом:

1 шаг:

5

9

3

11

2 шаг:

9

5

11

3

3 шаг:

9

11

5

3

4 шаг:

5

3

9

11

5 шаг:

5

11

9

3

Т.е. выбираемое на каждом шаге ребро подсвечивается голубым цветом. Ребра уже вошедшие в МОД подсвечиваются красным. В итоге получится выделенное красным цветом минимальное остовное дерево.

Если очередное рассматриваемое ребро, при включении его в МОД, образует цикл, то выводится соответствующее сообщение и ребро становится черным.

Предполагается, что окно программы будет выглядеть следующим образом:

Алгоритм Краскала

Граф

Далее

Область отрисовки графа

Выбрать файл, из которого следует считать граф, можно, нажав «Граф.. Выбрать файл».

Для генерации графа необходимо нажать «Граф.. Генерация». Для просмотра ребер нужно нажать «Граф.. Показать ребра». Для перехода к следующему шагу алгоритма надо нажать «Далее».

1. *Внутренняя*

Граф должен представляться классом, расширяющим класс JPanel и реализующим класс потока, также должен содержать такие члены-данные:

* Массив ребер, вошедших в МОД
* Массив вершин
* Число вершин
* Число ребер
* Массив ребер
* Текущий поток

И следующие методы:

* Реализация самого алгоритма Краскала
* 2 конструктора с параметрами
* Отрисовка самого графа

1. *Формат входного файла*

Файл должен содержать последовательно количество вершин, количество ребер, номер вершины из, номер вершины в, стоимость ребра.

Пример:

*5*

*3*

*0 2 6*

*0 1 5*

*2 4 7*

1. **Использование структур данных**

* Граф, описанный в спецификации
* Массив булевских переменных размером число вершин на число вершин. Этот массив используется в алгоритме Краскала для определения подграфа в котором содержится вершина

1. **Распределение обязанностей**

|  |  |
| --- | --- |
| Диогенова К.А. | * Реализация алгоритма Краскала * Создание репозитория * Оформление отчета * Проведение тестирования |
| Казачкова А.Д. | * Визуализация алгоритма Краскала * Раелизация графа (структура графа и конструкторы) * Оформление отчета |

1. **План разработки**

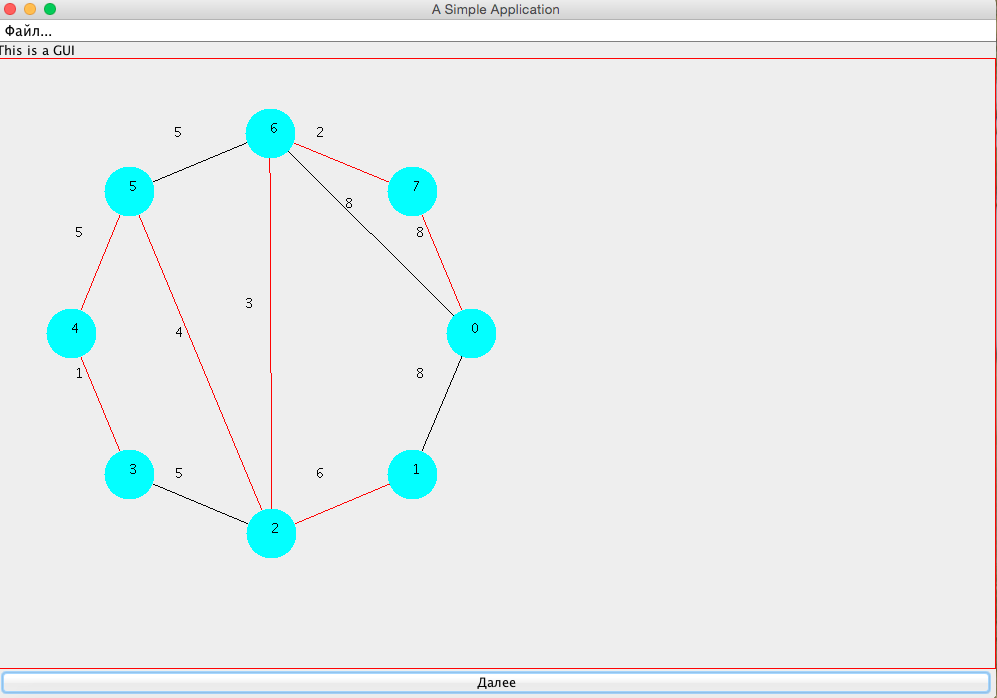
|  |  |
| --- | --- |
| 24.06 – 25.06 | * Оформление спецификации * Распределение обязанностей * Создание репозитория |
| 25.06 – 26.06 | * Написание прототипа программы (граф умеет отрисовываться и инициализироваться) |
| 27.06 – 29.06 | * Создание первой версии программы (рабочий алгоритм Краскала) |
| 29.06 – 1.07 | * Создание второй версии программы (визуализация алгоритма Краскала) |
| 2.07 | * Устранение возможных недочетов в программе и включение по возможности подсвечивания цикла графа |
| 3.07 | * Окончательная сдача финальной версии программы (включает при возможности подсветку циклов) |

**Уточнения ко второй версии, возникшие во время работы над первой:**

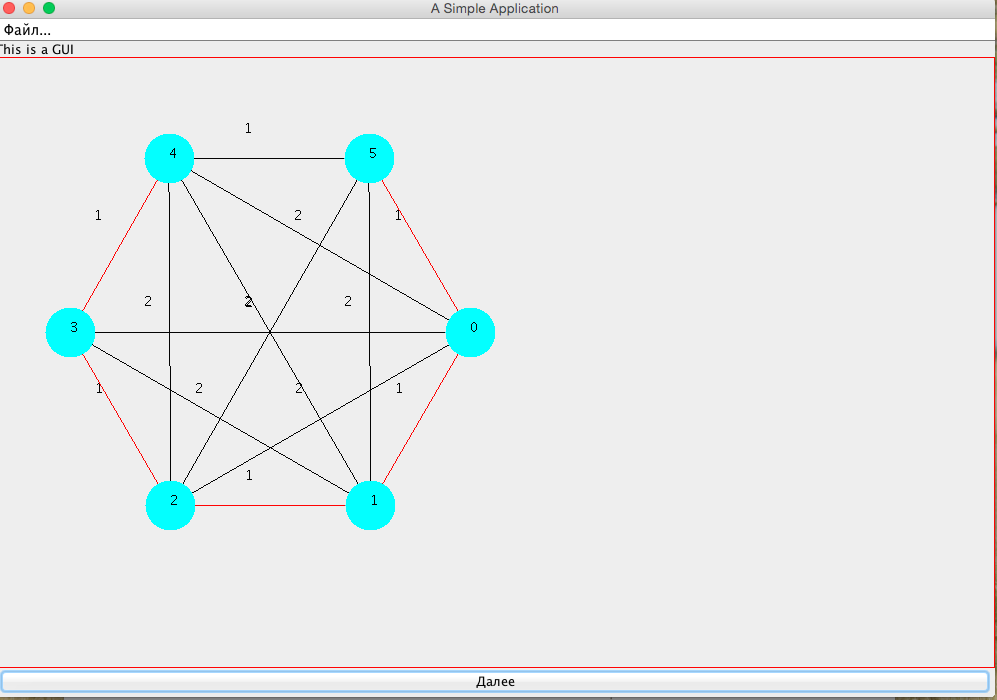
1. Улучшить формат входного файла;
2. Улучшить вывод веса ребер;
3. Загрузка файла и его выбор;
4. При большом числе изменять радиус вершины;
5. Генерация графа (количество вершин и процент ребер);
6. Добавить окно списка ребер.
7. **Тестирование первой версии программы (рабочий алгоритм Краскала)**

Красным выделено минимальное остовное дерево.

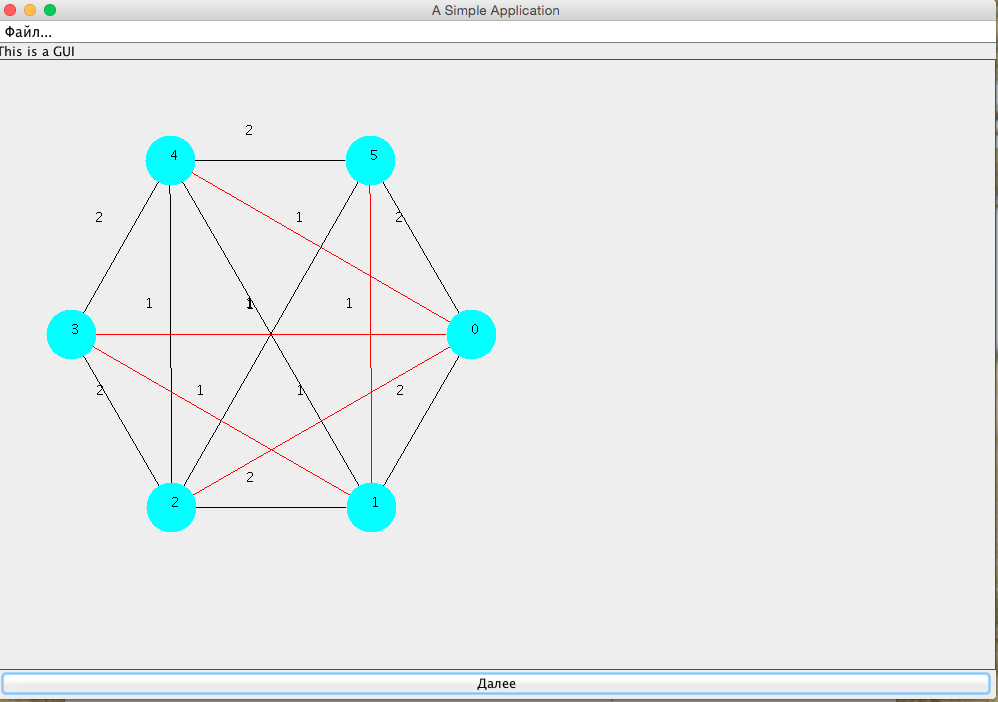
1. Пример с возможным циклом



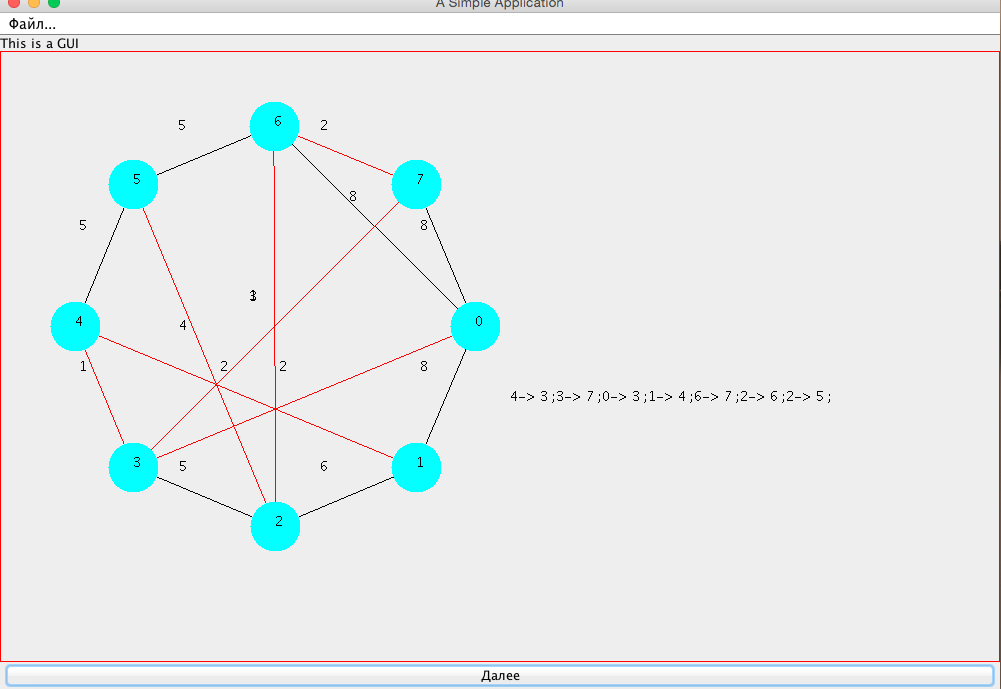
1. Пример с возможным циклом



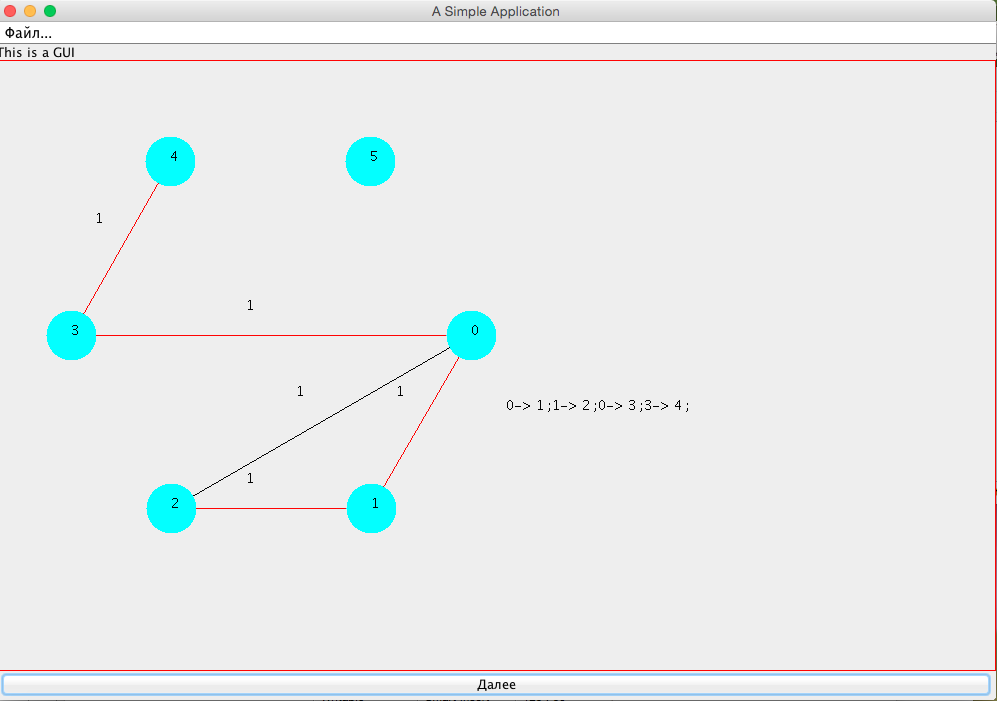
1. Пример с возможным циклом



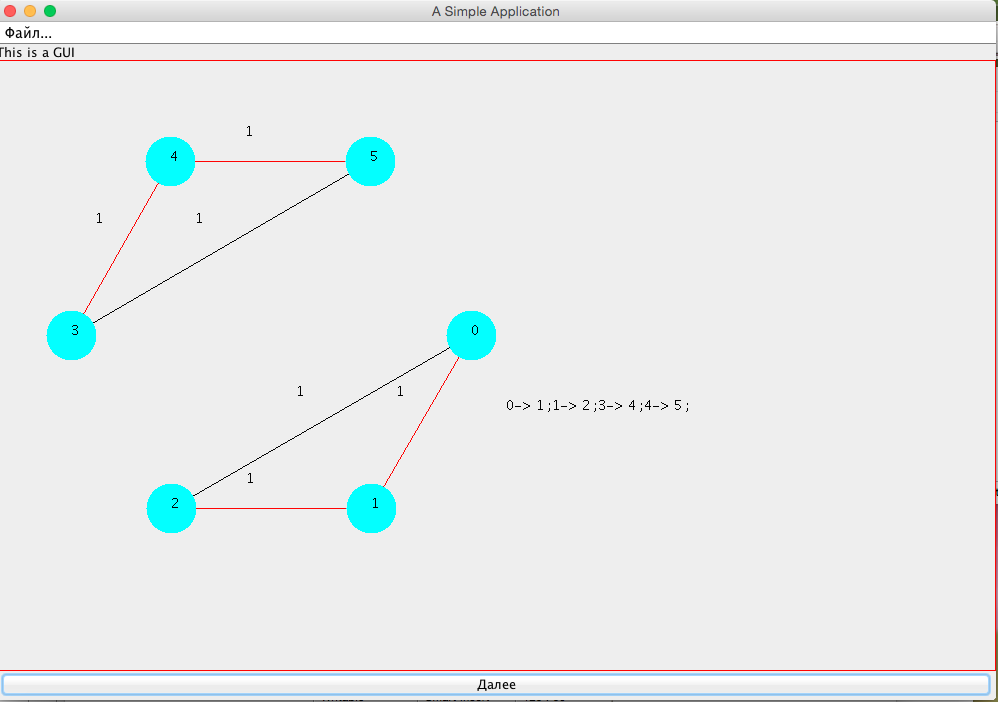
1. Пример с большим количеством ребер



1. Пример с изолированной вершиной

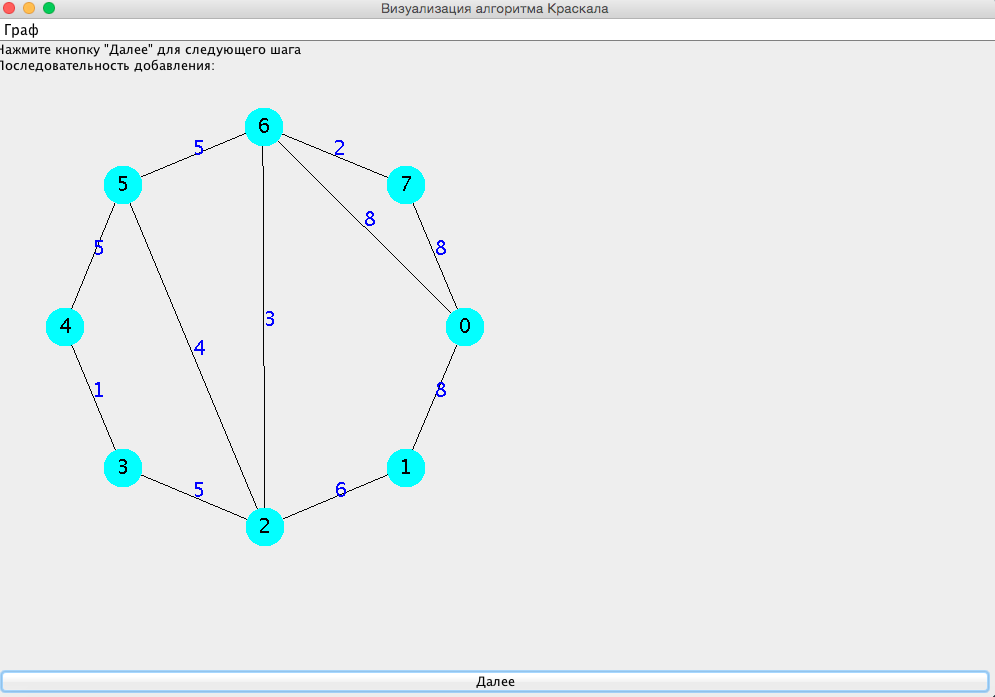


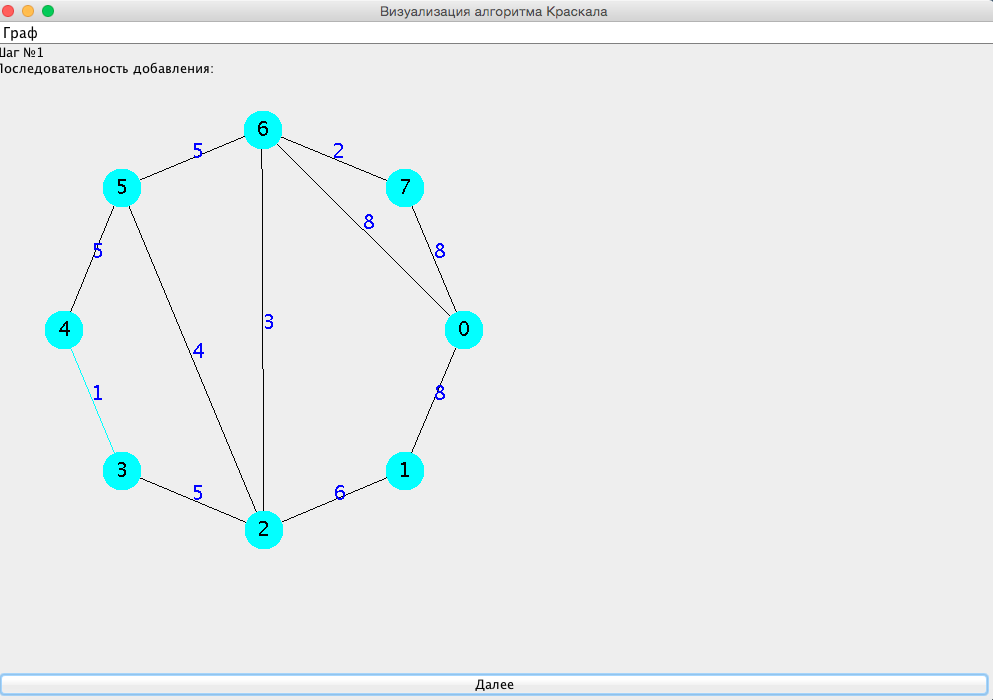
1. Пример графа в котором не все вершины соединены ребрами

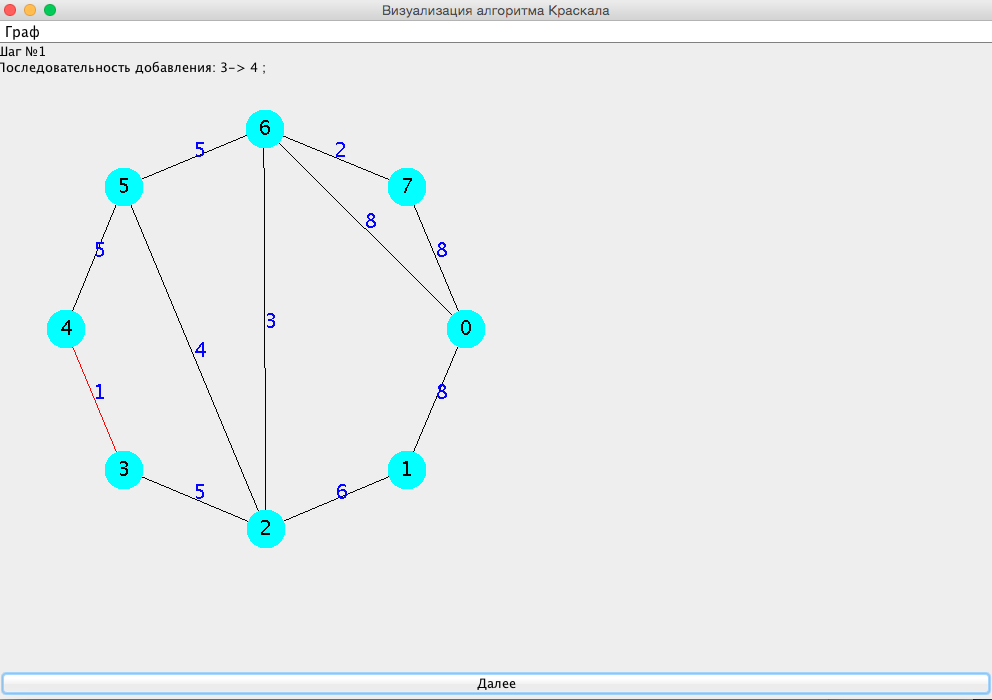


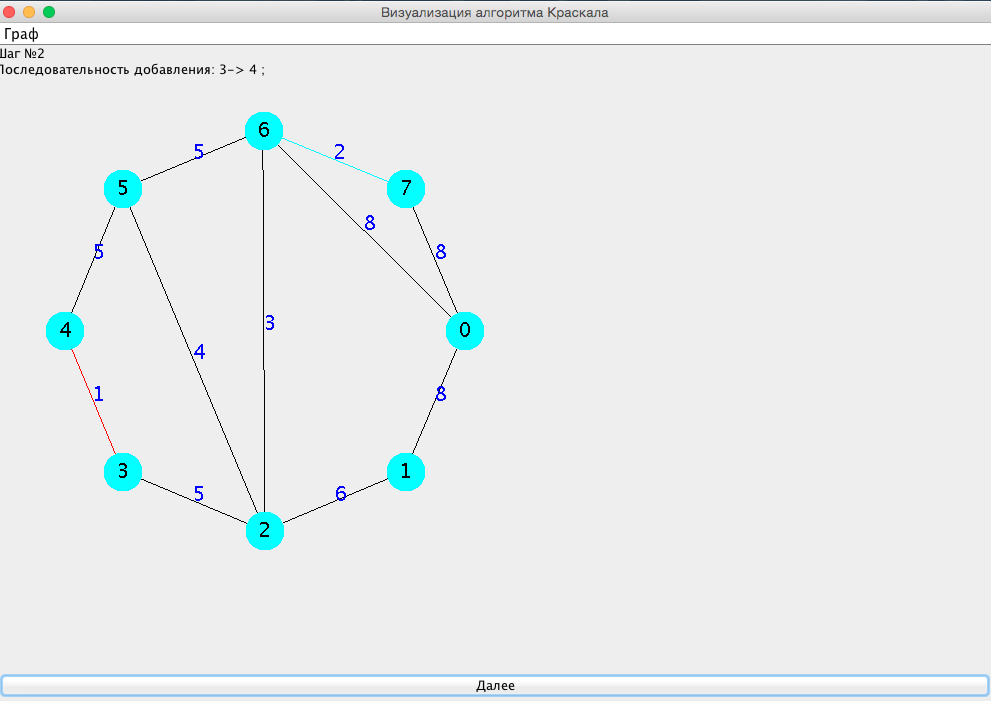
Как видим, алгоритм правильно находит МОД.

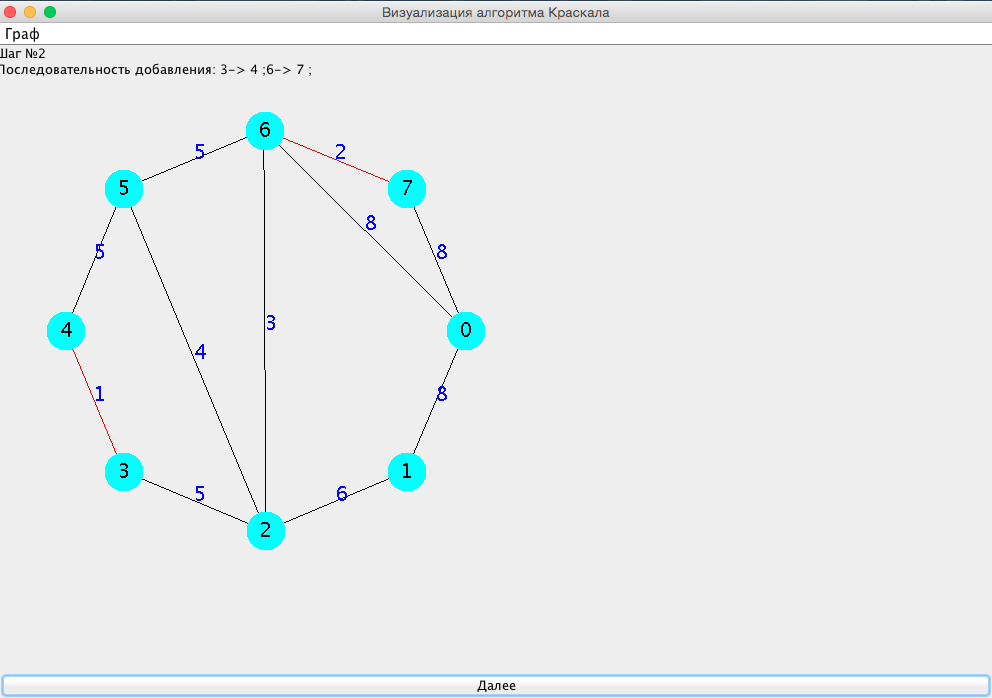
1. **Тестирование визуализации работы алгоритма**

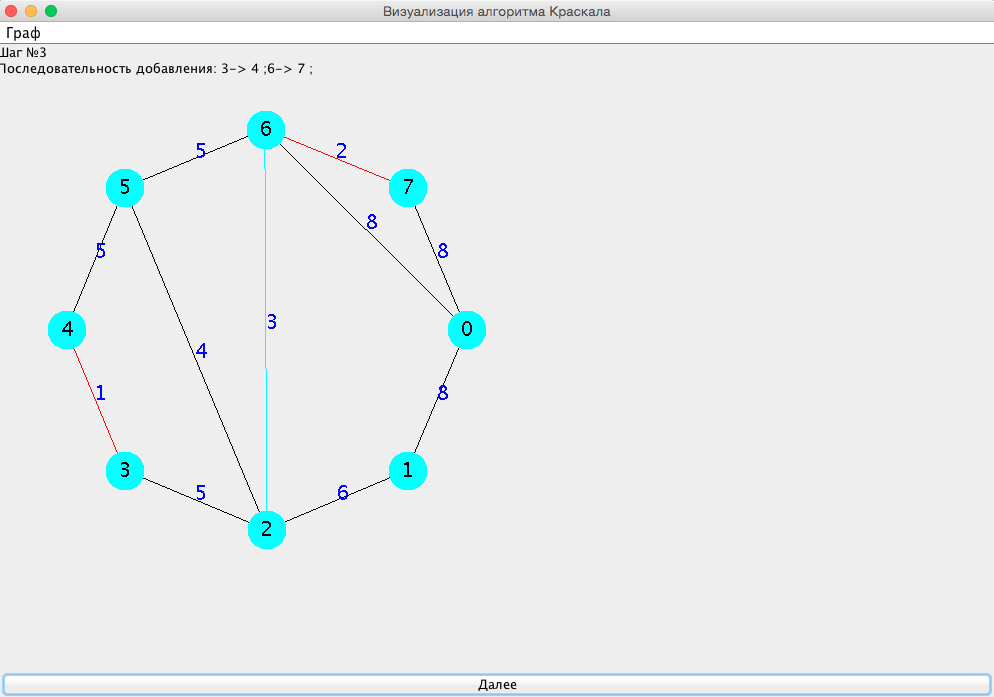
****

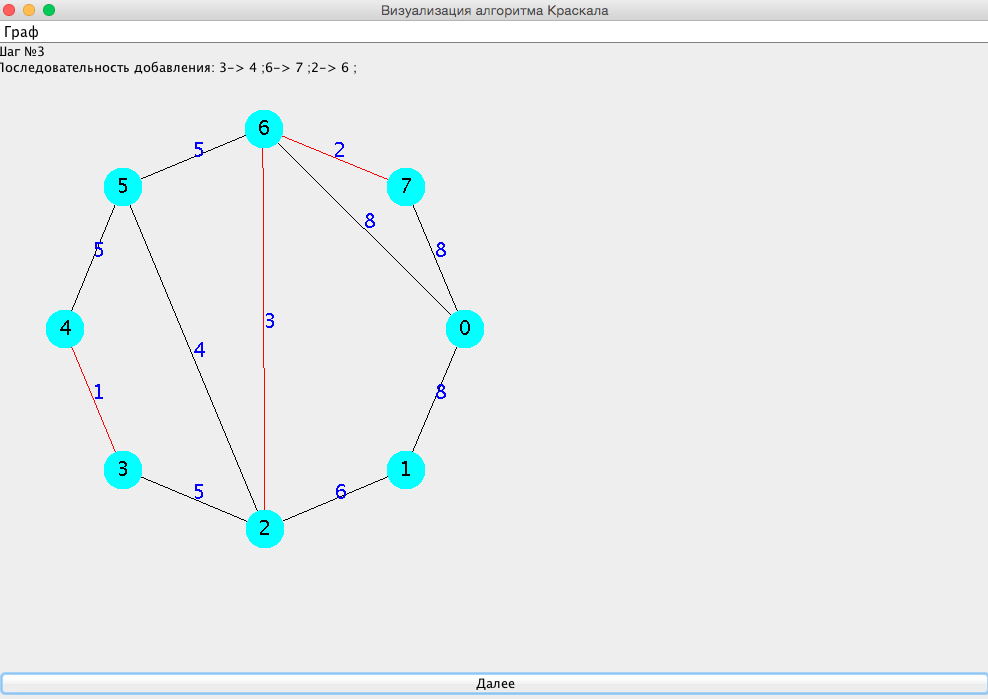
****

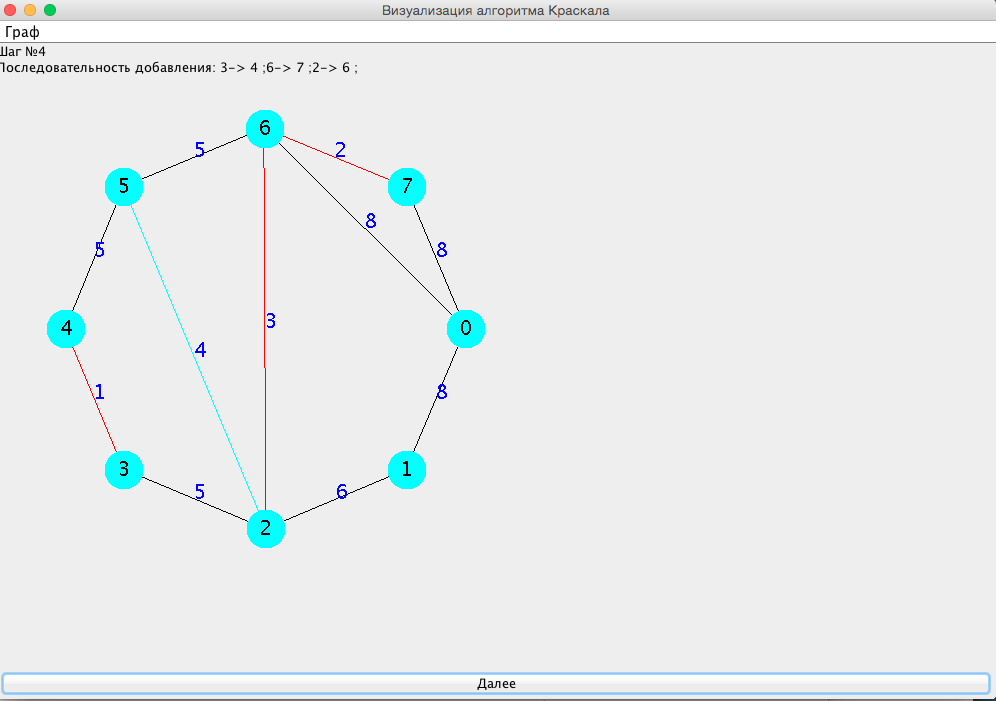
****

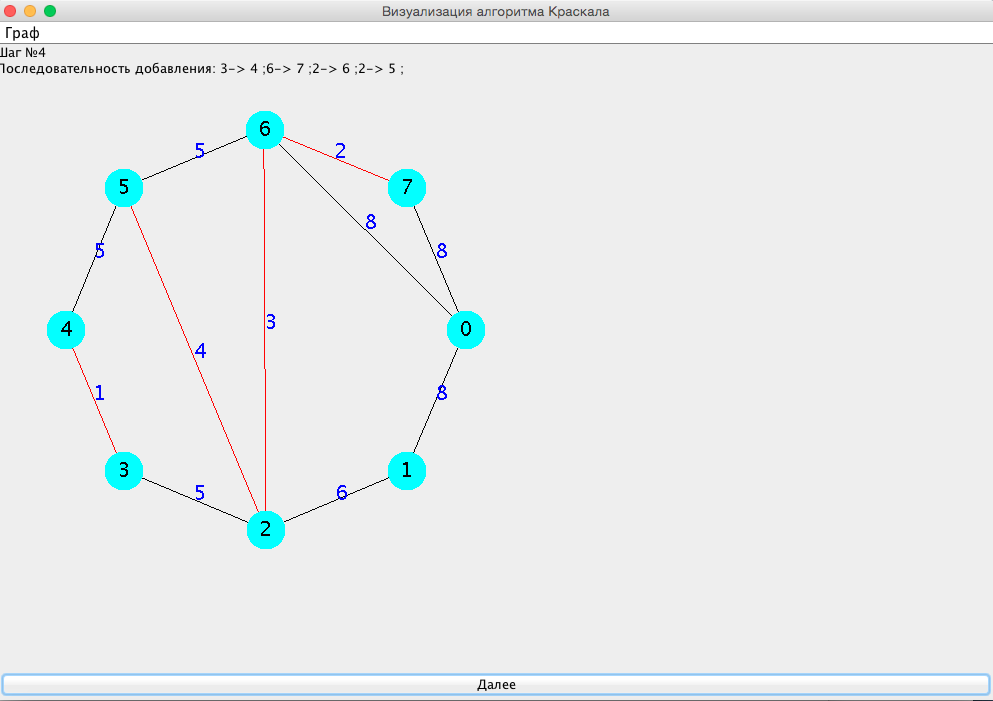
****

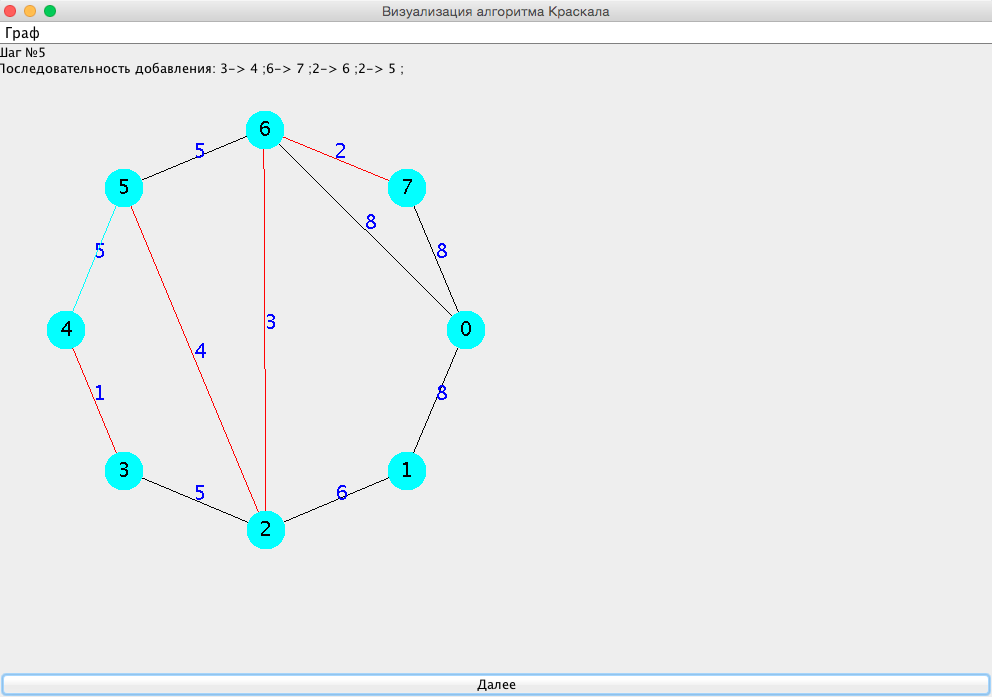
****

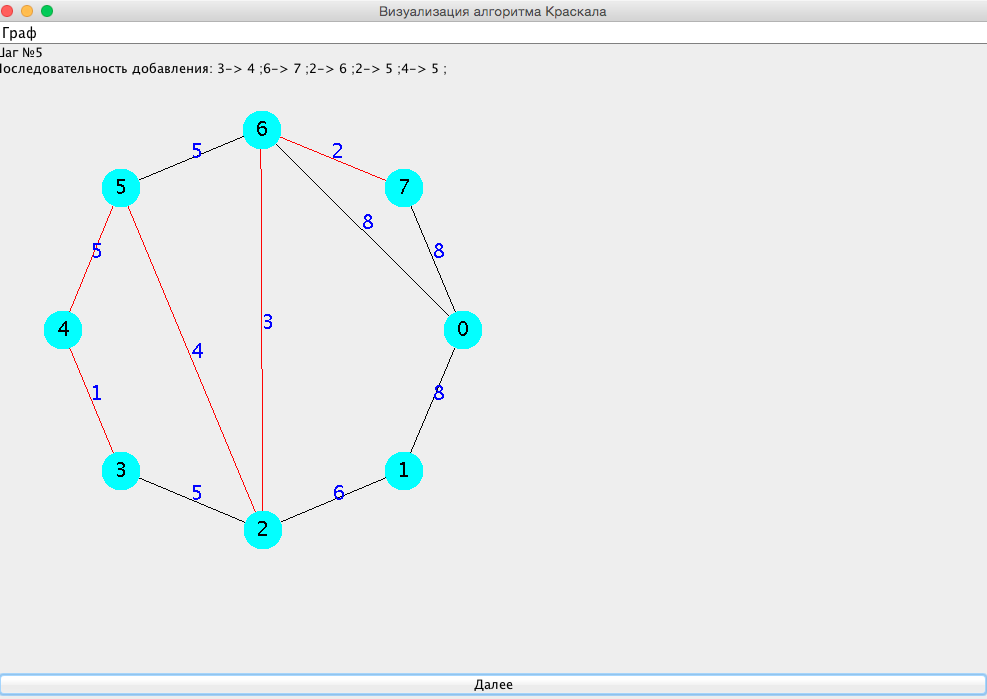
****

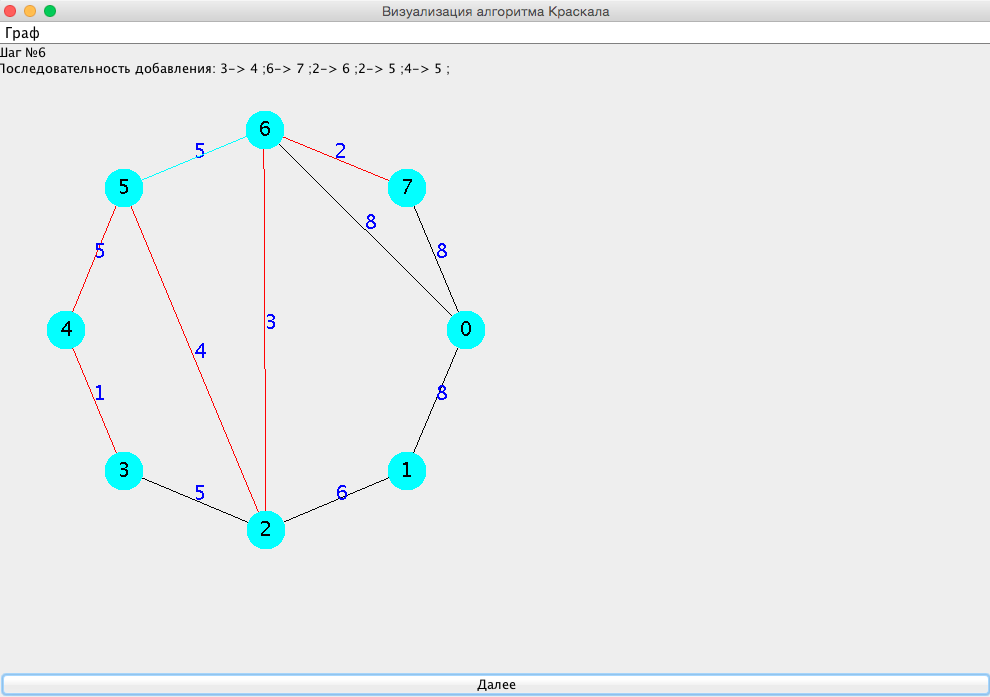
****

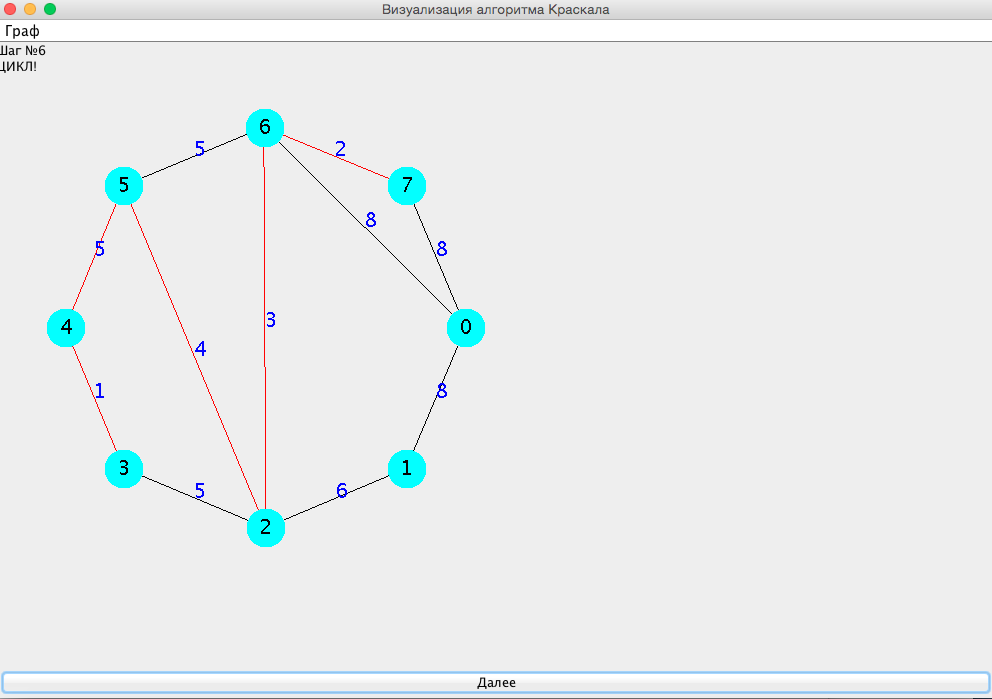
****

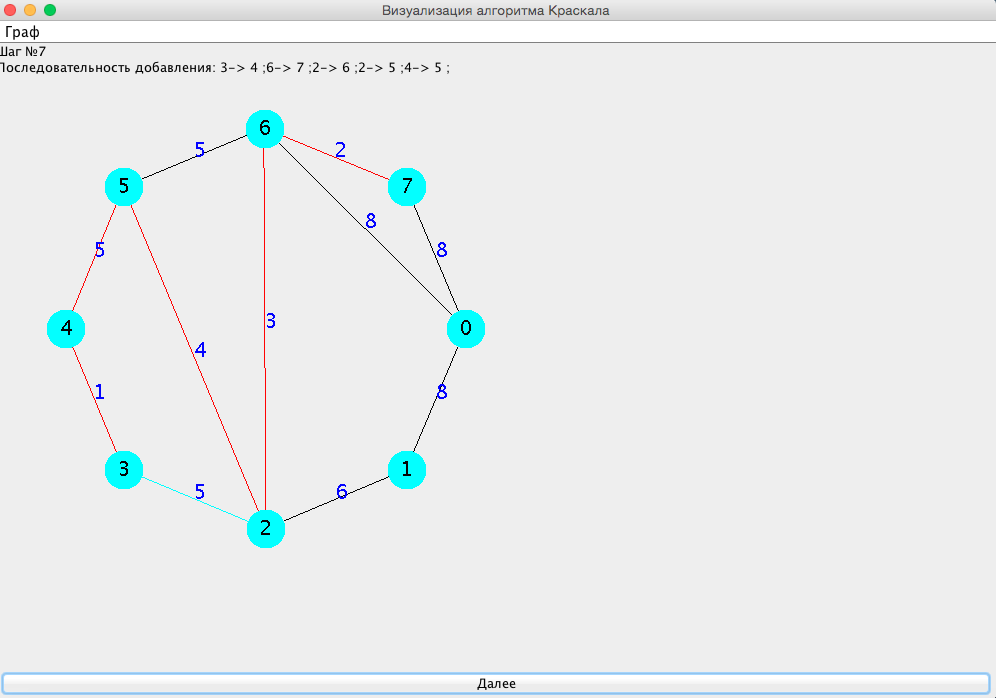
****

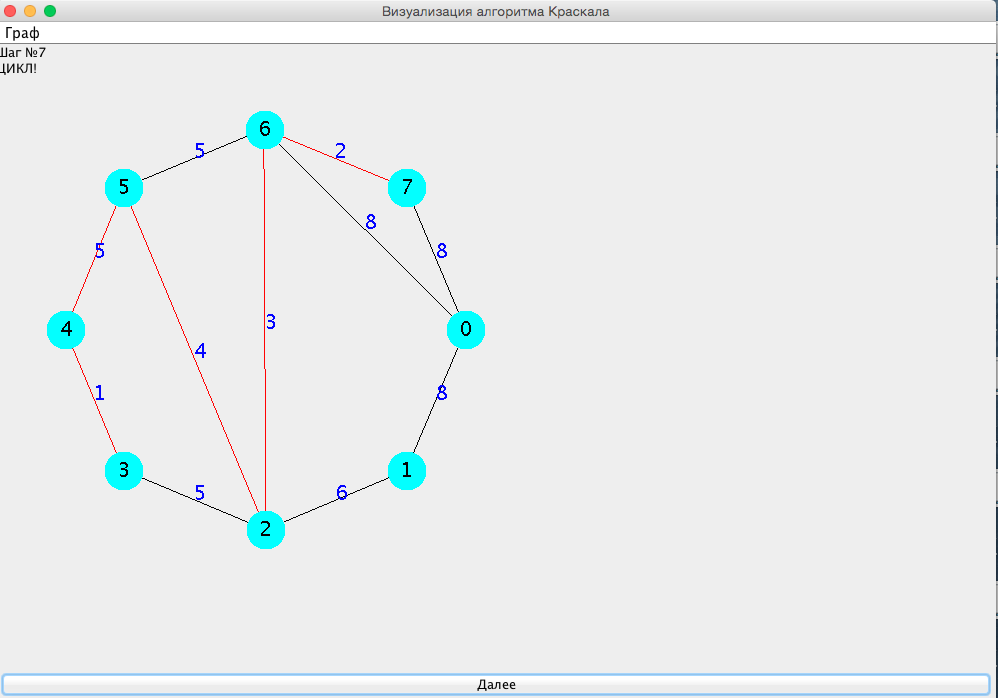
****

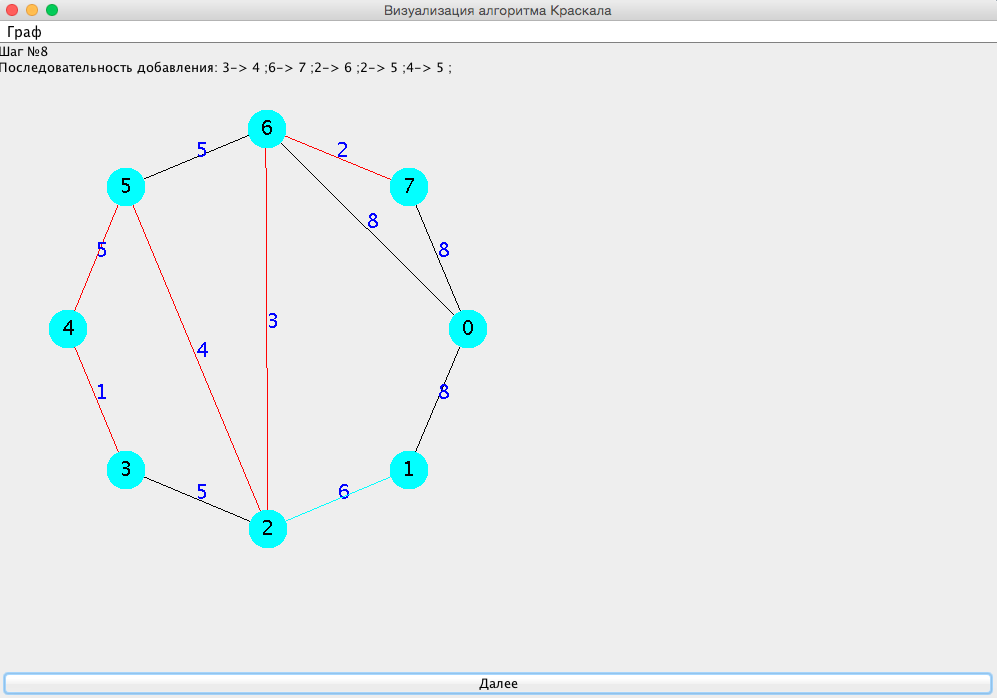
****

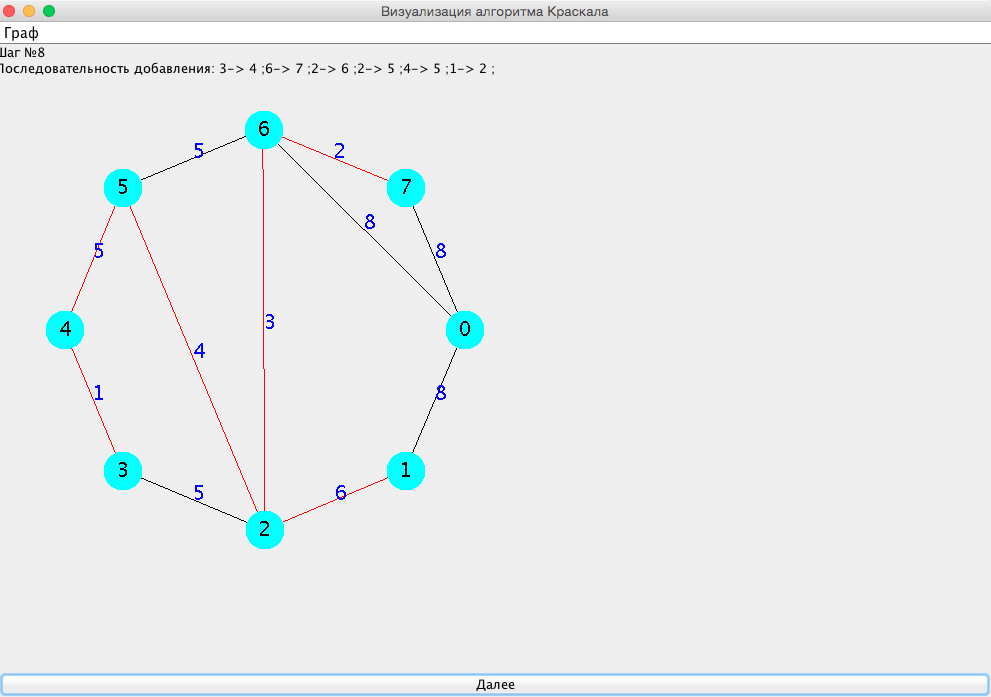
****

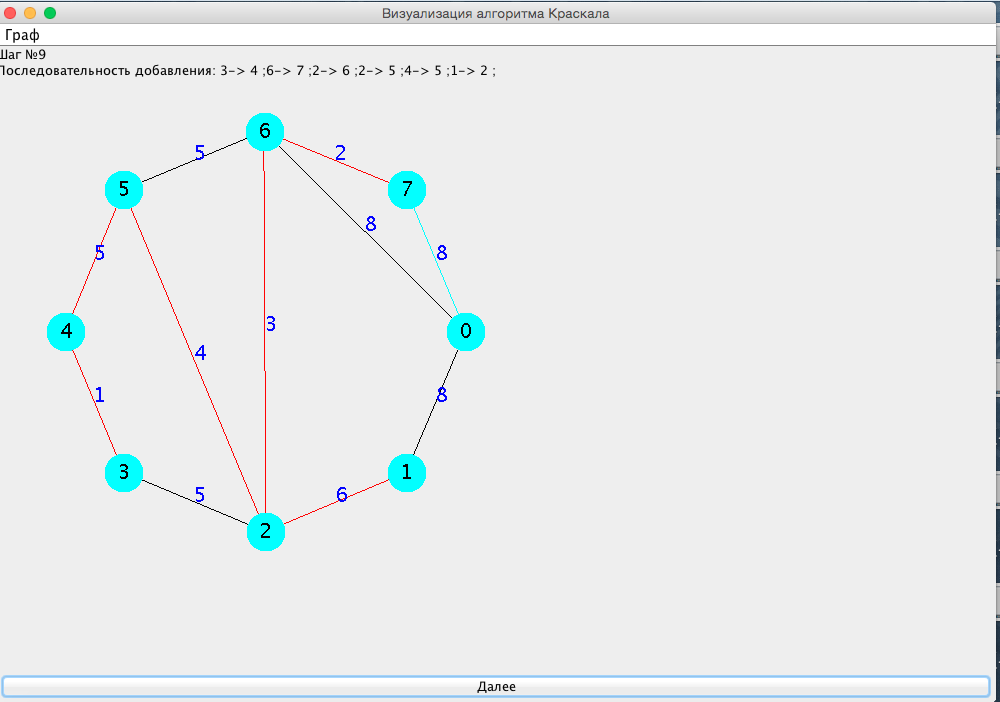
****

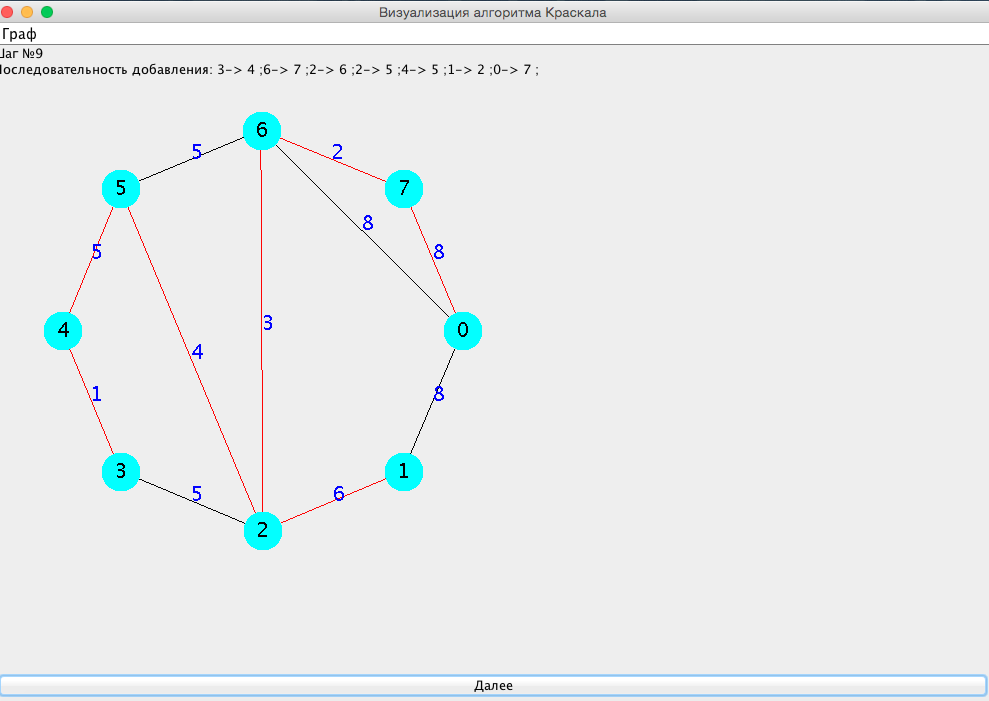
****

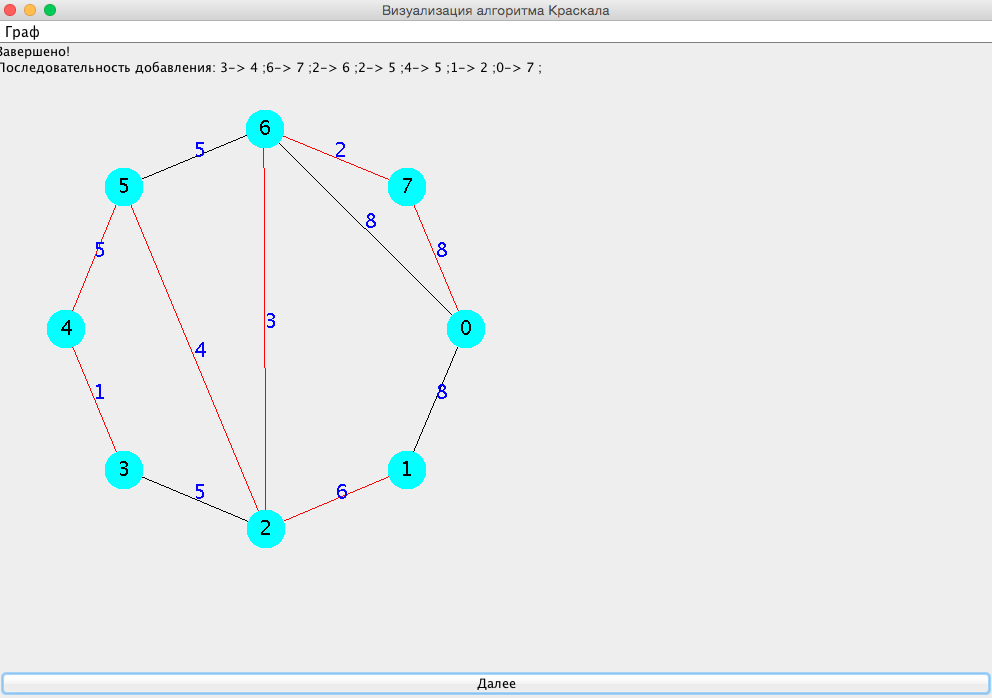
****

****

****

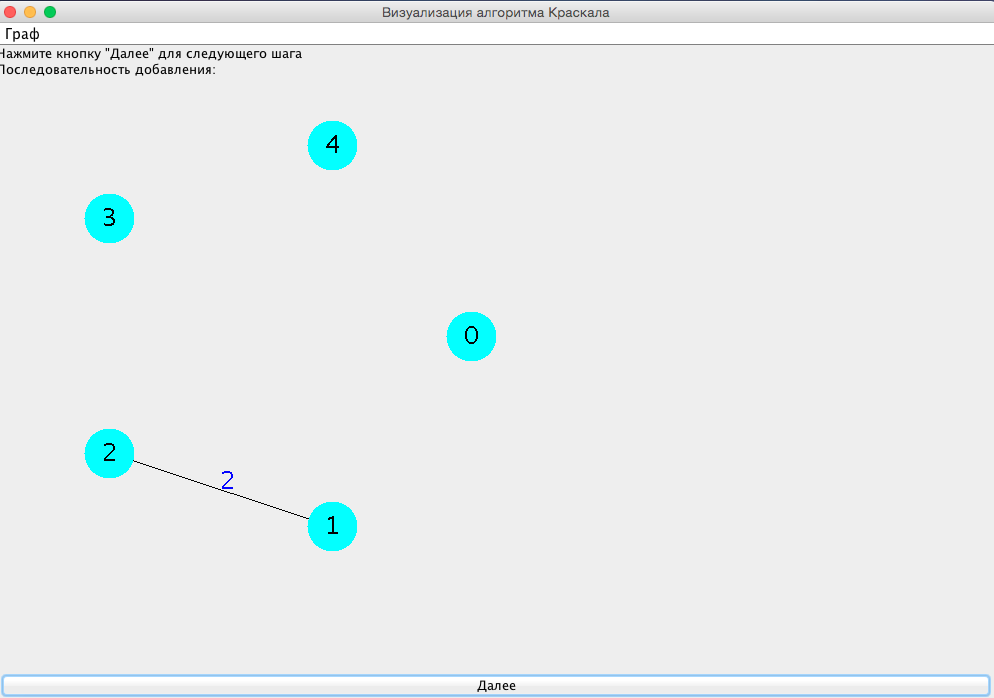
****

****

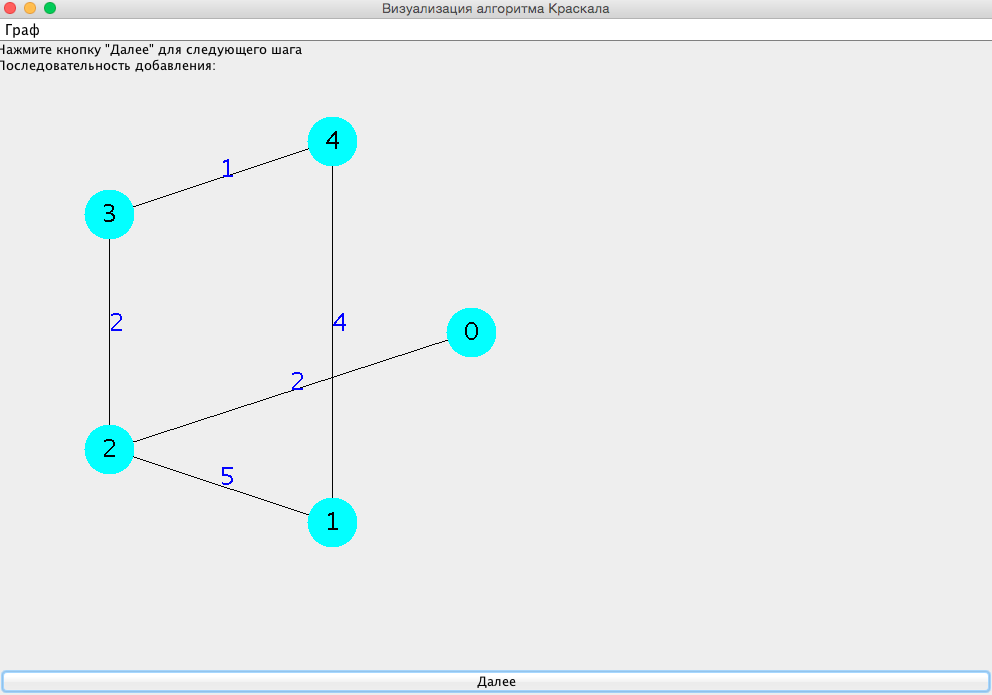
****

Как видим, визуализация алгоритма работает правильно.

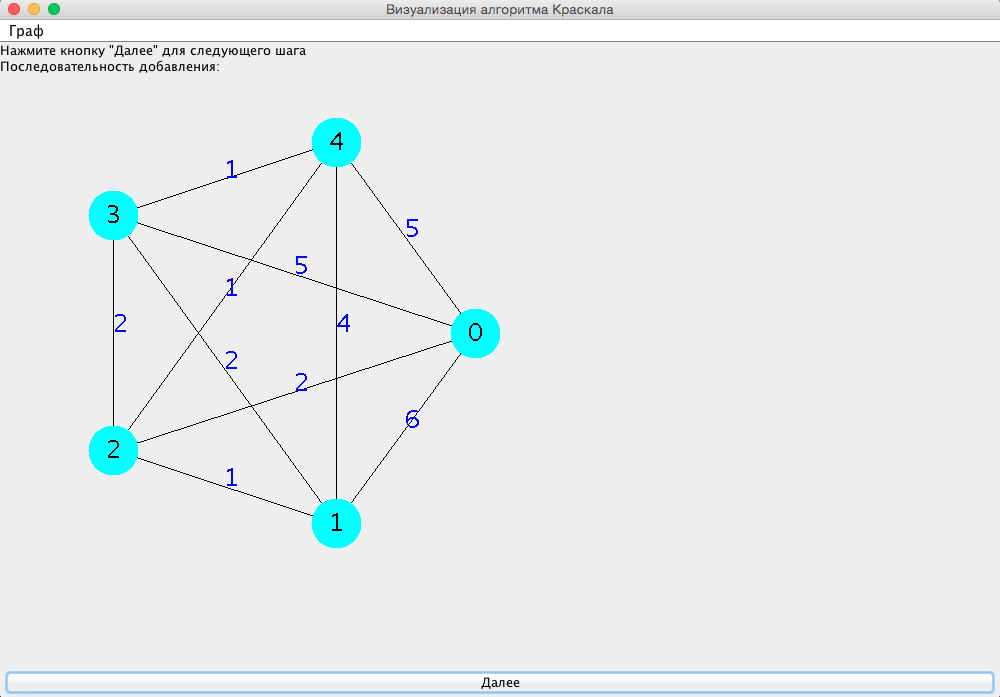
1. **Тестирование генерации**
2. Генерация графа из 5 вершин и 10 процентом ребер

****

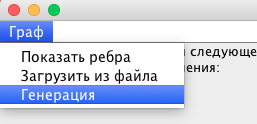
1. Генерация графа из 5 вершин и 50 процентов ребер



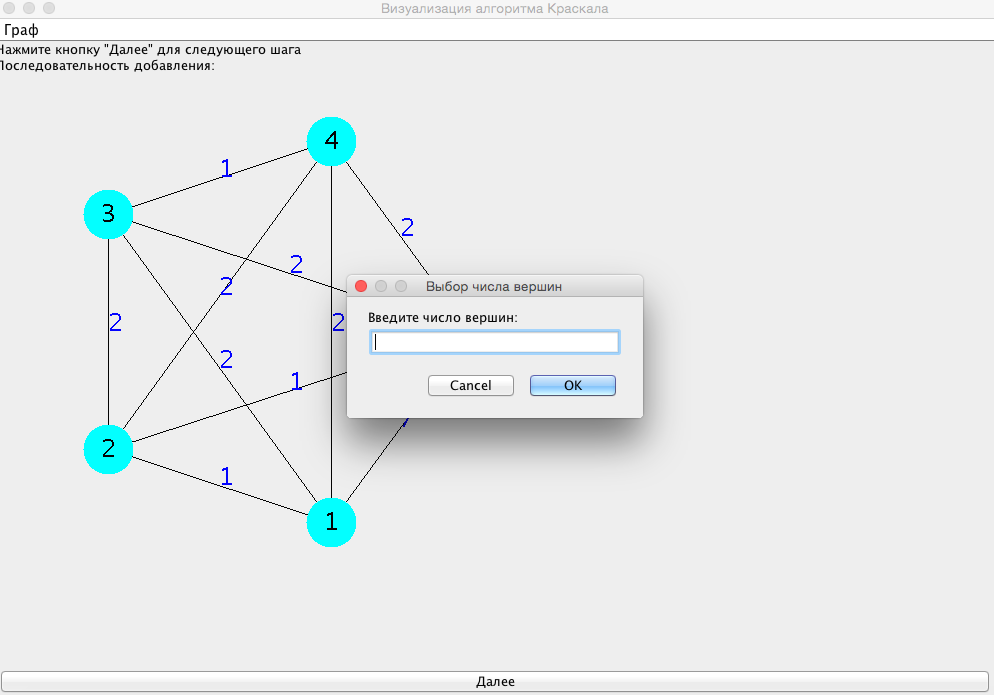
1. Генерация графа из 5 вершин и 100 процентов ребер



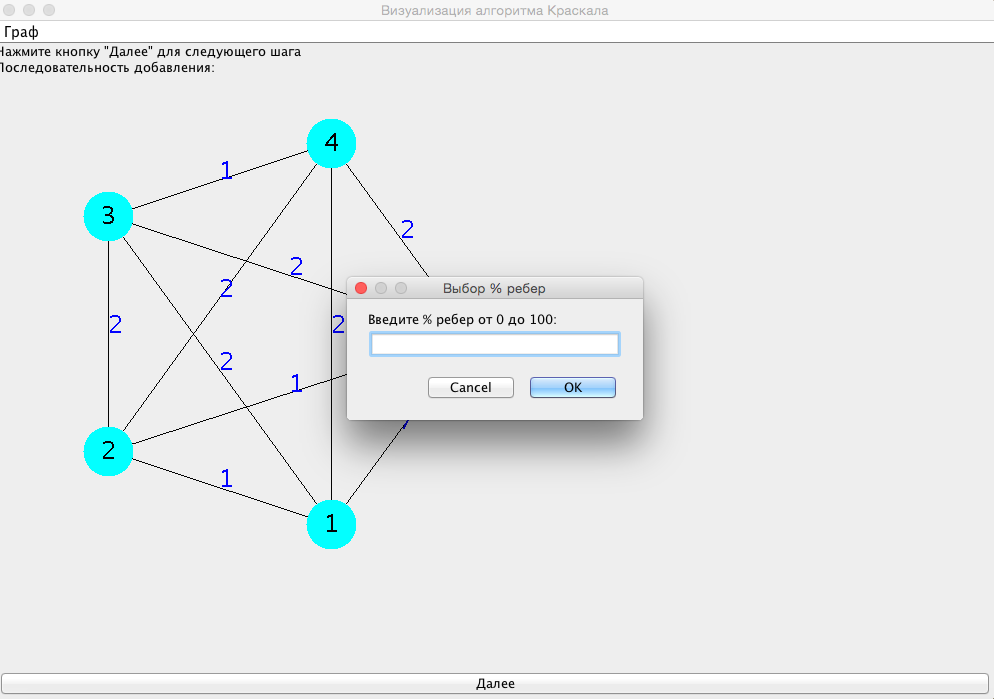
1. Кнопка генерации



1. Окно генерации «Выбор числа вершин»



1. Окно генерации «Выбор процента ребер»



1. Вывод:

Генерация происходит верно, так как в графе на 5 вершин максимальное число ребер 10, соответственно, каждые 10% ребер составляют одно ребро. Как видим, ровно нужное число ребер.

1. **Заключение**

Проект выполнен успешно. Учтены все требования.