МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по учебной практике

ТЕМА: «Алгоритм Борувки»

Студент гр. 9383 _____ Рыбников Р.А.

Студентка гр. 9383 _____ Сергиенкова А.А.

Студент гр. 9383 _____ Крейсманн К.В.

Руководитель _____ Ефремов М.А.

Санкт-Петербург

2021

ЗАДАНИЕ НА УЧЕБНУЮ ПРАКТИКУ

Студент Рыбников Р.А. группы 9383

Студентка Сергиенкова А.А. группы 9383

Студент Крейсманн К.В. группы 9383

Тема практики: Алгоритм Борувки

Задание на практику:

Командная разработка визуализации алгоритма на языке Java с графическим интерфейсом.

Алгоритм: Алгоритм Борувки.

Дата сдачи отчёта:

Дата защиты отчёта:

Студент гр. 9383	_____	Рыбников Р.А.
Студентка гр. 9383	_____	Сергиенкова А.А.
Студент гр. 9383	_____	Крейсманн К.В.
Руководитель	_____	Ефремов М.А.

АННОТАЦИЯ

Целью учебной практики является разработка графического приложения для нахождения минимального оставного дерева для заданного графа с помощью алгоритма Борувки.

Программа разрабатывается на языке Java, командой из трёх человек, каждый из которых имеет определённую специализацию.

SUMMARY

The aim of the training practice is to develop a graphical application for finding the minimum abandoned tree for a given graph using Boruvka's algorithm.

The program was developed in the Java language by a team of three people, each of whom has a specific specialization.

СОДЕРЖАНИЕ

Введение

1. Требования к программе……6

1.1. Исходные требования к программе……6

1.2. Уточнение требований после сдачи прототипа……..6

2. План разработки и распределение ролей в бригаде……7

2.1. План разработки……7

2.2. Распределение ролей в бригаде……...7

3. Особенности реализации……8

4. Тестирование……..11

4.1 План тестирование……..11

4.2 Тестирование с библиотекой Junit……..12

4.3 Демонстрация тестирования операций над графом……..14

Заключение……..21

ВВЕДЕНИЕ

Целью данной практической работы является разработка графического приложения, выполняющего визуализацию работы алгоритма Борувки. Пользователю программы должна быть предоставлена возможность самостоятельно задать входные данные для алгоритма с помощью графического интерфейса. Результат работы алгоритма должен иметь графическое отображение. Должна быть предоставлена возможность просмотра итогового результата алгоритма и просмотра хода его исполнения по шагам.

Разработка осуществляется на языке Java, командой из трёх человек, каждый из которых имеет определённую специализацию.

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Исходные требования к программе

Программа представляет собой визуализацию алгоритма Борувки, нахождения минимального остовного дерева для взвешенного неориентированного графа.

Требования к вводу исходных данных

Для задания графа будут реализованы несколько возможностей: задание графа посредством взаимодействия с графическими элементами; случайная генерация графа.

Требования к визуализации

Пользователю должно быть доступно графическое изображение графа, взаимодействие с ним средствами графического интерфейса, просмотр состояний графа на каждом шаге алгоритма и просмотр конечного результата.

Рисунок 1 – use-case диаграмма.

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

- Распределение ролей, составление диаграммы прецедентов программы, создана директория с исходным кодом.
  Срок выполнения: 02.07.2021
- Создание интерфейса, но пока не рабочего, проектирование классов программы.
  Срок выполнения: 04.07.2021
- Реализация случайной генерации графа, реализация алгоритма с отображением результата работы, составить план тестирования.
  Срок выполнения: 06.07.2021
- Сделан прототип программы в котором визуализируется как получение и отображение результата сразу, так и пошаговое выполнение алгоритма.
  Срок выполнения: 08.07.2021
- Проект полностью готов, программа корректно собирается.
  Срок выполнения: 10.07.2021

2.2. Распределение ролей в бригаде

Рыбников Р.А. – Реализация алгоритма, логики программы, документации и тестирования.

Сергиенкова А.А. – Работа с алгоритмом, документация, тестирование.

Крейсманн К.В – Работа с логикой взаимодействия с графическим интерфейсом, основная логика взаимодействия интерфейса с алгоритмом, тестирование.

3.ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1. Структуры данных

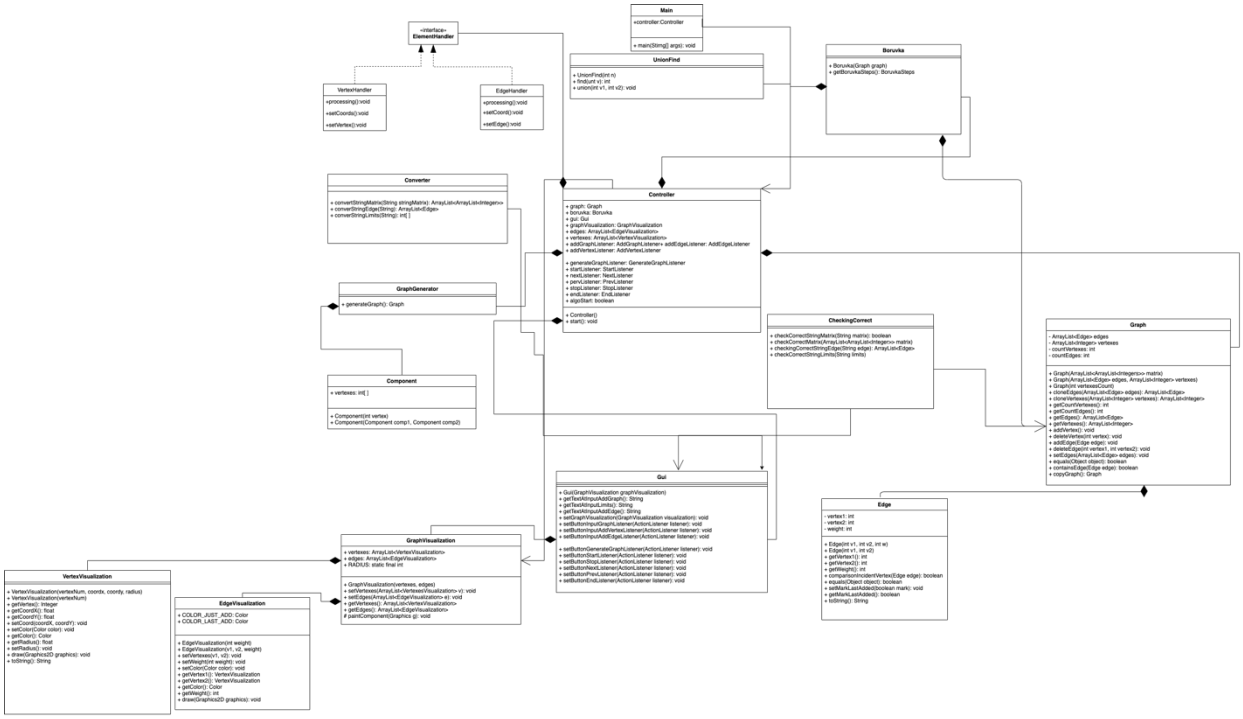

Рисунок 2 – UML диаграмма классов.

Описание use-case диаграммы:

Пользователь, посредством взаимодействия с графическим интерфейсом, может совершать ряд действий:

1) Создание графа с помощью ввода матрицы смежности.
2) Создание графа с помощью его случайной генерации с параметрами : количество ребер, количество вершин, минимальный вес, максимальный вес.
3) Добавление вершины в граф при нажатии на кнопку.
4) Добавление ребра при вводе инцидентных ребру вершин и веса и нажатии кнопки.
5) Добавление ребра от выбранной вершины к другой вершине, при помощи нажатия на нужную вершину правой кнопкой мыши и выборе нужного пункта меню.
6) Изменение веса ребра при нажатии на него правой кнопкой мыши и выборе нужного пункта меню.

7) Удаление ребра при помощи нажатия на него правой кнопкой мыши и выборе нужного пункта меню.
8) Удаление вершины при помощи нажатия на нее правой кнопкой мыши и выборе нужного пункта меню.
9) Просмотреть вес ребра, нажав на него левой кнопкой мыши.
10) Начать работу алгоритма, нажав кнопку “START” и пошагово просматривать его ход с помощью стрелочек.
11) Перейти к результату алгоритма, нажав на кнопку “TO END”.
12) Завершить алгоритм, нажав на кнопку “STOP”.

Описание uml-диаграммы классов:

1) Класс Graph является реализацией графа. Граф хранится в виде набора ребер и набора вершин. В классе Graph реализованы функции для работы с ним.
2) Класс Edge является реализацией ребра.
3) Класс GraphGenerator генерирует по заданным параметрам связный граф. Для того чтобы граф был связный реализован вспомогательный класс Component, который позволяет хранить и объединять наборы вершин.
4) Класс Gui является реализацией графического интерфейса. В данном классе реализованы все необходимые элементы, через которые пользователь может общаться с приложением.
5) С помощью классов VertexVisualization и GraphVisualizatoin отрисовываются вершины и ребра.
6) Класс GraphVisualization является графическим представлением графа. GraphVisualization располагает вершины равномерно по окружности. Класс реализовывает интерфейс MouseListener и дает возможность прослушивать нажатия мыши пользователем.

7) Класс Controller является связующим звеном между графикой и логикой приложения. В классе добавлены ряд классов-слушателей, которые реагируют на события пользователя при нажатии кнопок.
8) Классы EdgeHandler и VertexHandler реализуют интерфейс ElementHandler и представляют из себя обработчиков нажатия пользователя по элементам графа (вершинам и ребрам).
9) Класс Boruvka является реализацией алгоритма. С помощью класса BoruvkaSteps реализован паттерн «Снимок», на каждой итерации алгоритма создается снимок и добавляется в BoruvkaSteps. Затем при нажатии пользователем стрелочек производится перемещение по снимкам.
10) Класс CheckingCorrect представляет различные функции для проверки корректности ввода.
11) Класс Converter представляет функции преобразования строкового ввода в необходимый формат.

4. Тестирование.

4.1. План тестирования.

- Тестирование операций над графом:
  - Базовый функционал
  - Добавление ребра в несуществующую вершину, добавление с отрицательным весом выдают исключение.
  - Удаление вершины влечёт за собой удаление связанных с ней рёбер.
- Тестирование алгоритма:
  - Базовый функционал.
  - Корректная обработка несвязных графов.

4.2 Тестирования с помощью библиотеки Junit.

Основное тестирование программы реализовано с использование библиотеки для модульного тестирования Junit. Предполагается тестирования основных методов, которые используется для работы алгоритма. Ниже приведено описание методов. Во всех тестах сравнивается ожидаемое значение и текущее значение. Если эти значения равны, то тестирование метода прошло успешно.

- Метод graphTest() – принимает на вход граф, инициализированный матрицей смежности. Результатом работы метода будет граф с определенными значениями.
- Метод getCountVertexes() – принимает на вход вершины графа. Результатом работы метода будет ответ исходя из проверки, что этот метод даст результат равный числу вершин графа.
- Метод getCountEdges() – принимает на вход рёбра графа. Результатом работы метода будет ответ исходя из проверки, что этот метод даст результат равный числу ребер графа.
- Метод addVertex() – принимает на вход количество вершин графа. Результатом работы метода будет ответ равны ли массивы вершин или нет.
- Метод deleteVertex() – принимает на вход количество вершин графа. Результатом работы методы будет ответ исходя из сравнения значений массивов до удаления и после.
- Метод addEdge() – принимает на вход количество ребер графа. Результатом работы метода будет результат сравнения количества ребер до и после.
- Метод deleteEdge() – принимает на вход количество ребер графа. Результатом работы метода будет результат сравнения количества ребер до и после.

Для тестирования самого алгоритма Борувки были созданы три метода, которые проверяют работу алгоритма, путем сравнения ожидаемого результата и текущего, на некоторых входных данных. Входные данные: матрица смежности и сумма ребер результирующего остовного дерева.

4.3 Демонстрация тестирования операций над графом.

В базовый функционал программы входит:

1. Ввод графа вручную через матрицу смежности.
2. Случайная генерация данных, на основе введенных данных(количество рёбер и вершин, минимальное и максимальное значение весов рёбер).
3. Добавление вершины, посредством нажатия на кнопку.
4. Добавление ребра, посредством нажатия на кнопку, при условии ввода вершин, для которых будет добавлено новое ребро.

Все эти возможности будут продемонстрированы на рисунках ниже.

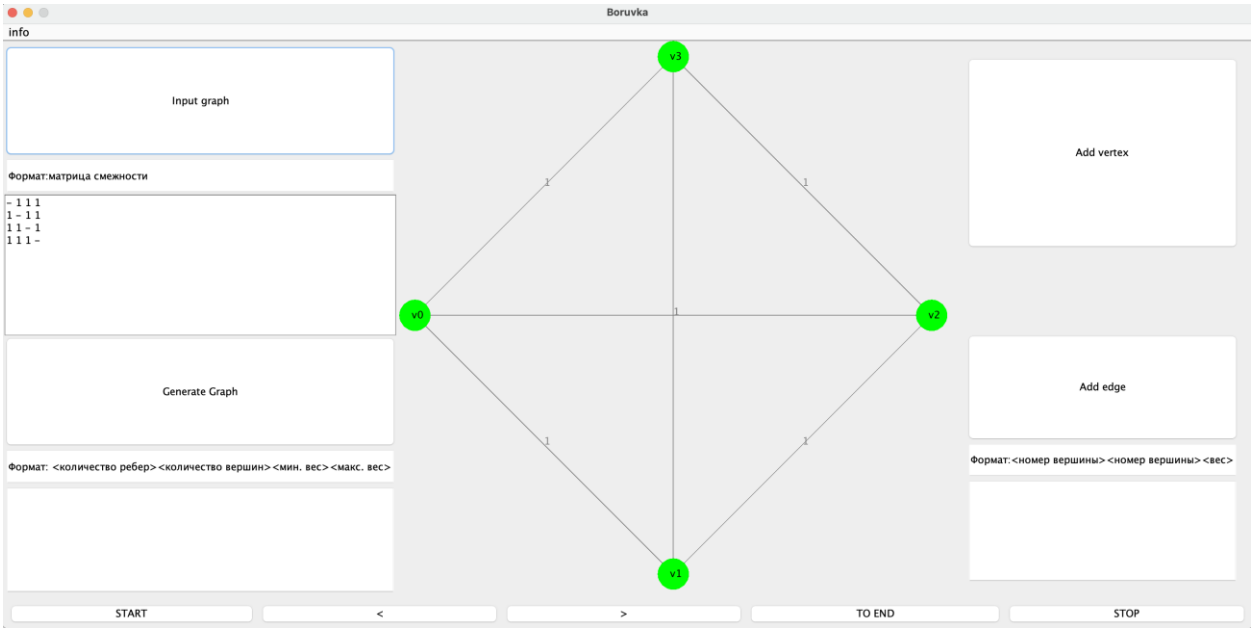

Рисунок 1 – Демонстрация ввода графа через матрицу смежности.

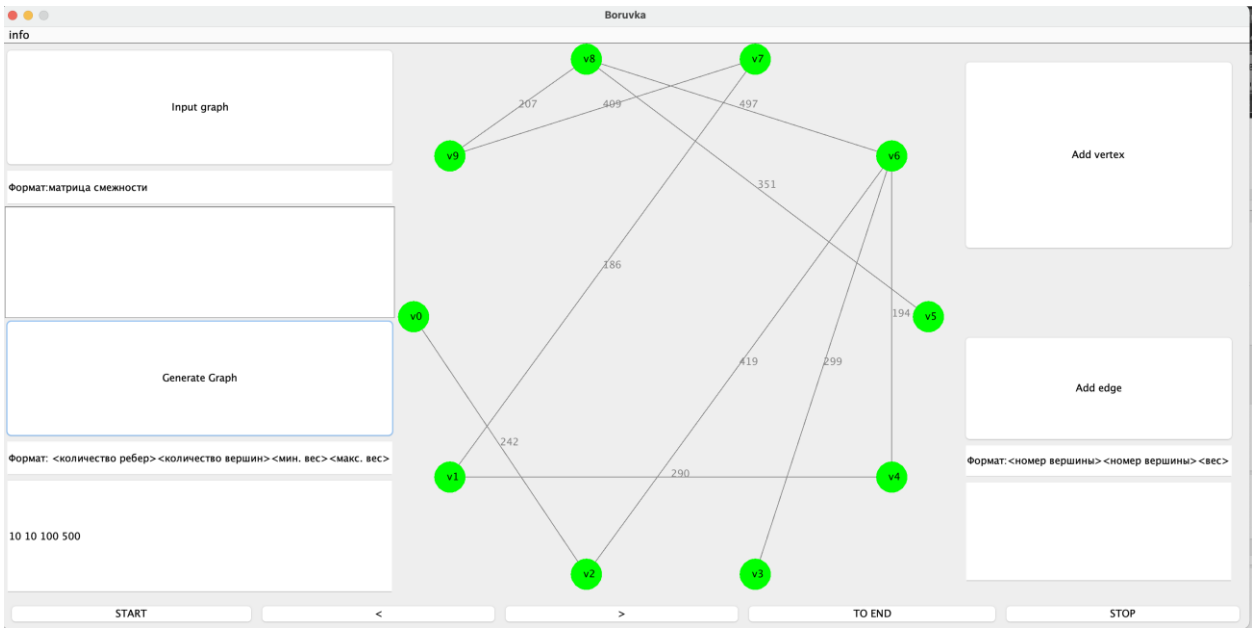

Рисунок 2 – Демонстрация случайной генерации графа.

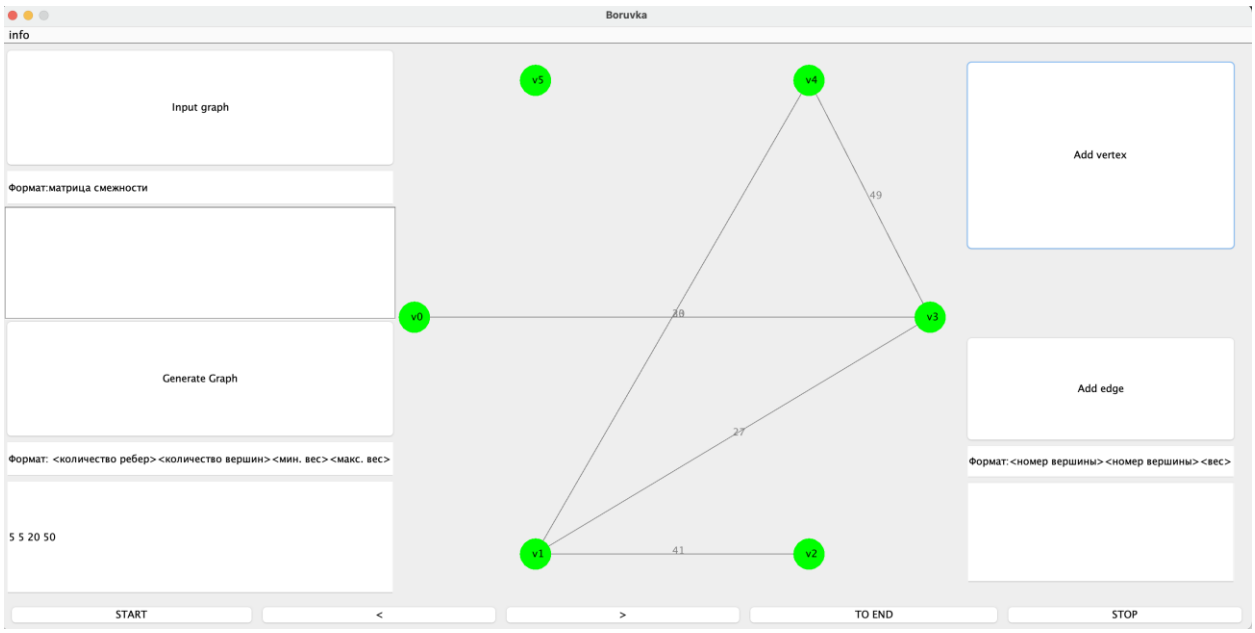

Рисунок 3 – Демонстрация добавления вершины в уже существующий граф.

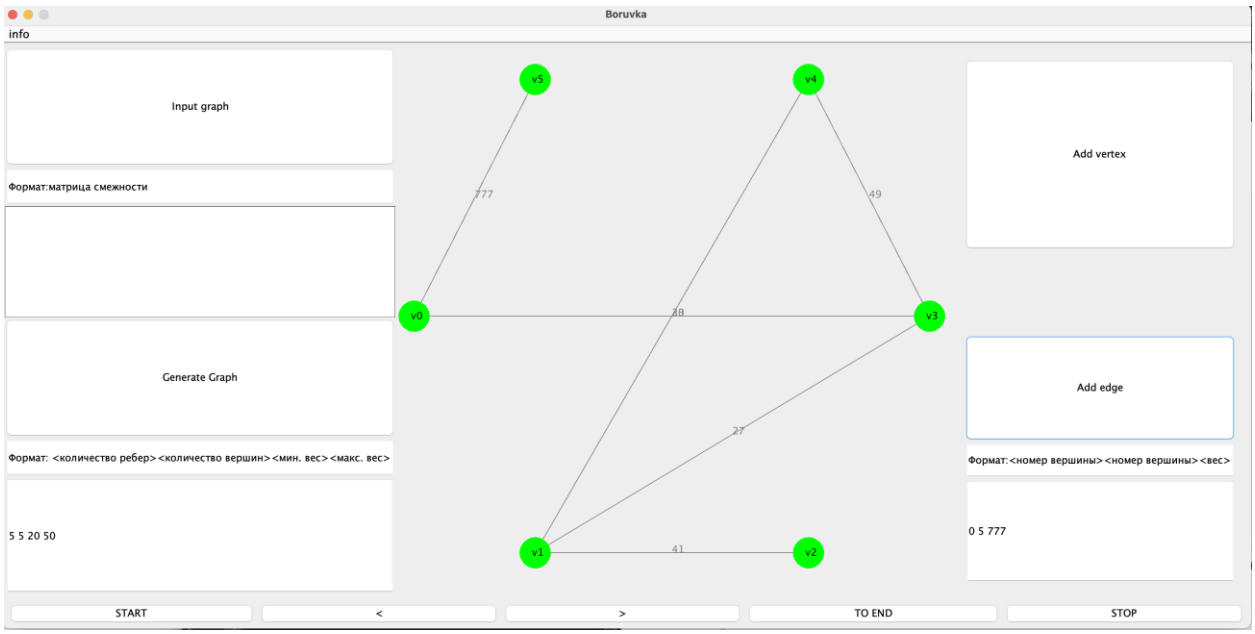

Рисунок 4 – Демонстрация добавления ребра в уже существующий граф.

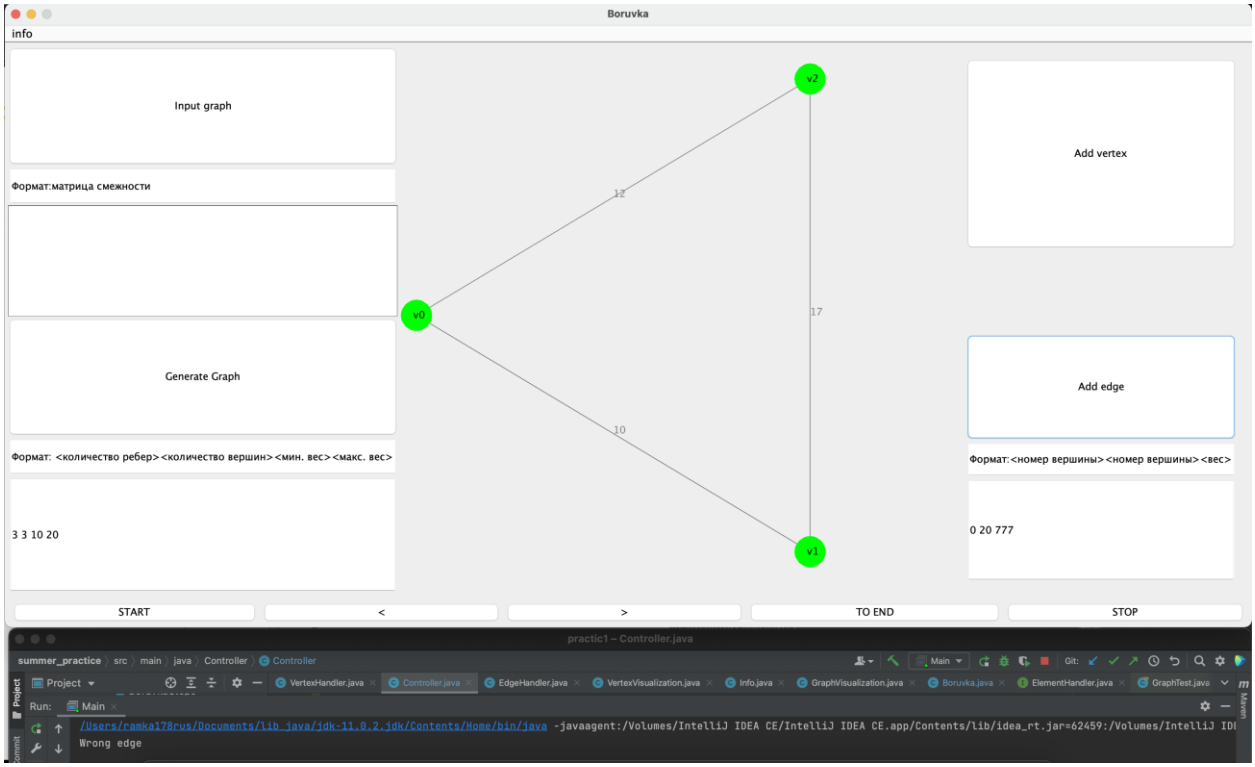

Рисунок 5 – Демонстрация добавления ребра в несуществующую вершину.

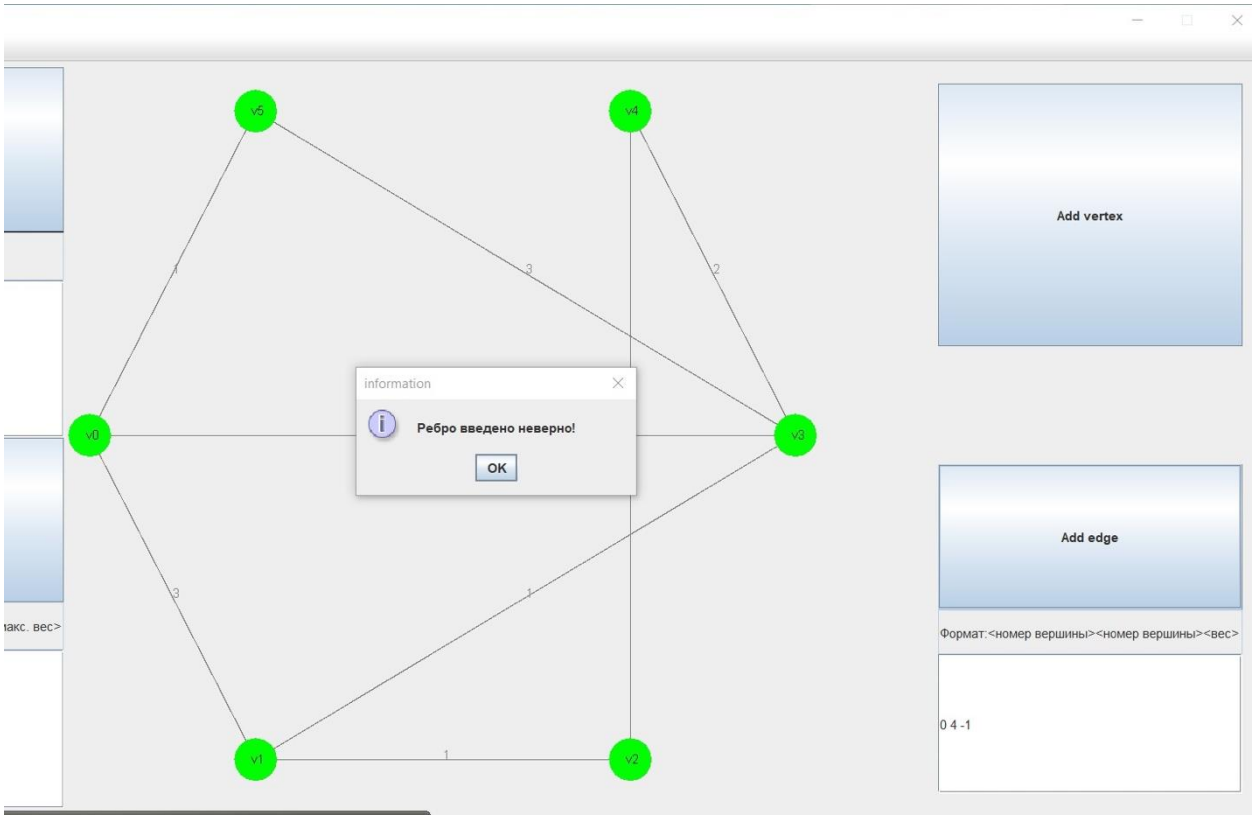

Рисунок 6 – Демонстрация добавления ребра с отрицательным весом.

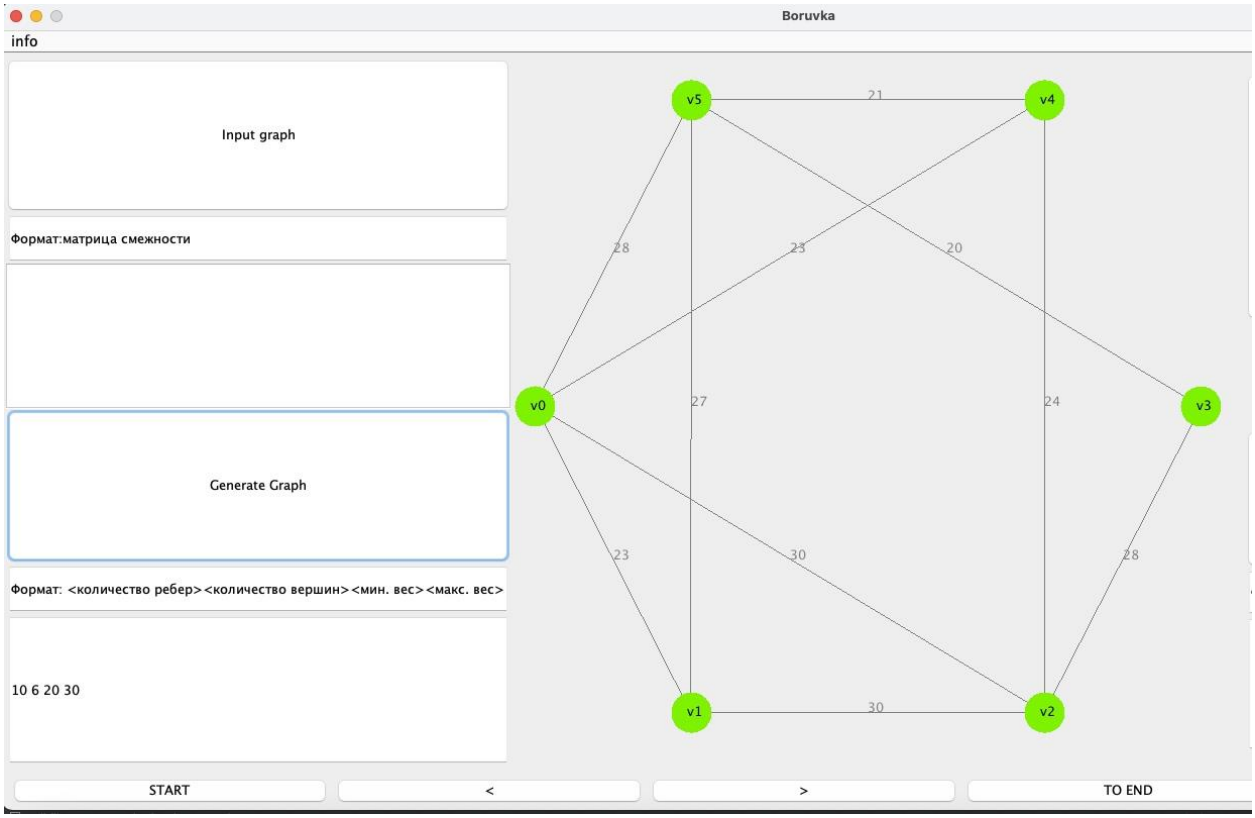

Рисунок 7 – Демонстрация ситуации до удаления вершины со связными рёбрами.

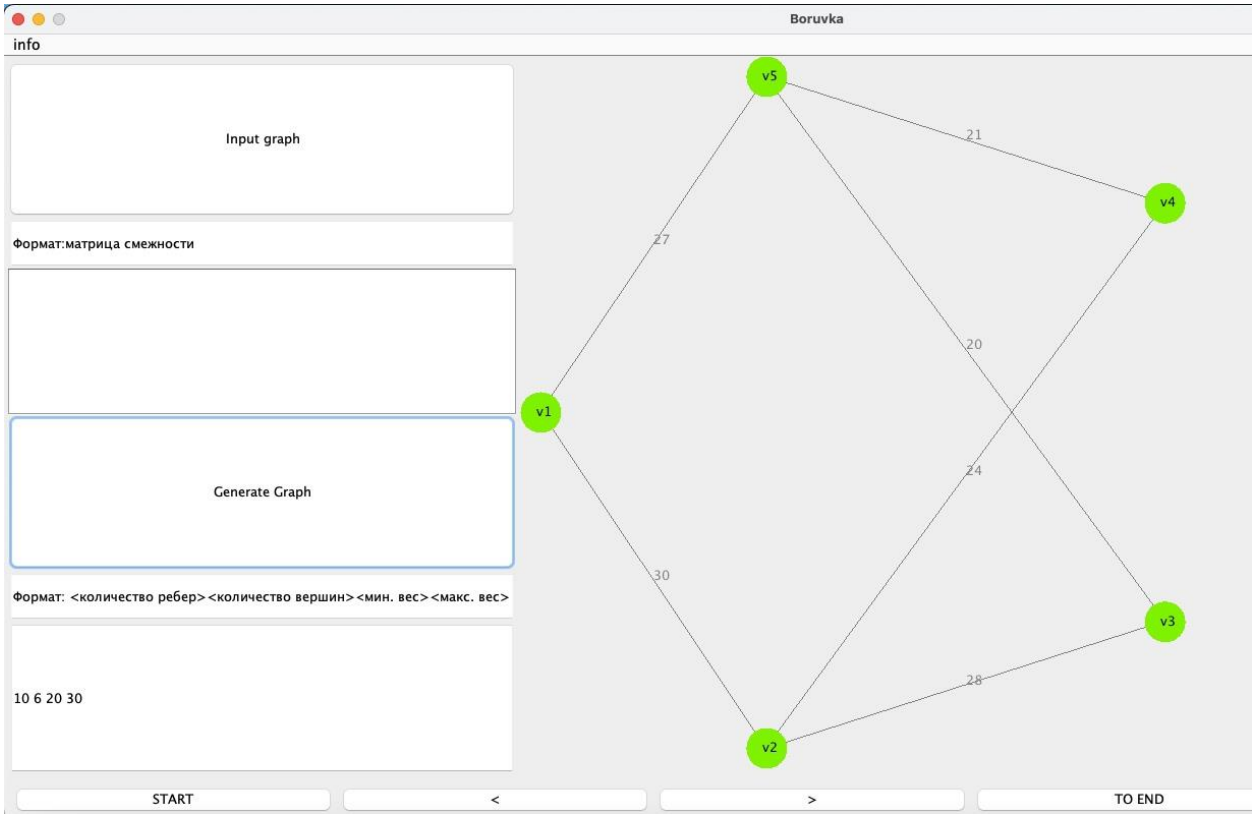

Рисунок 8 – Демонстрация ситуации после удаления вершины со связными рёбрами.

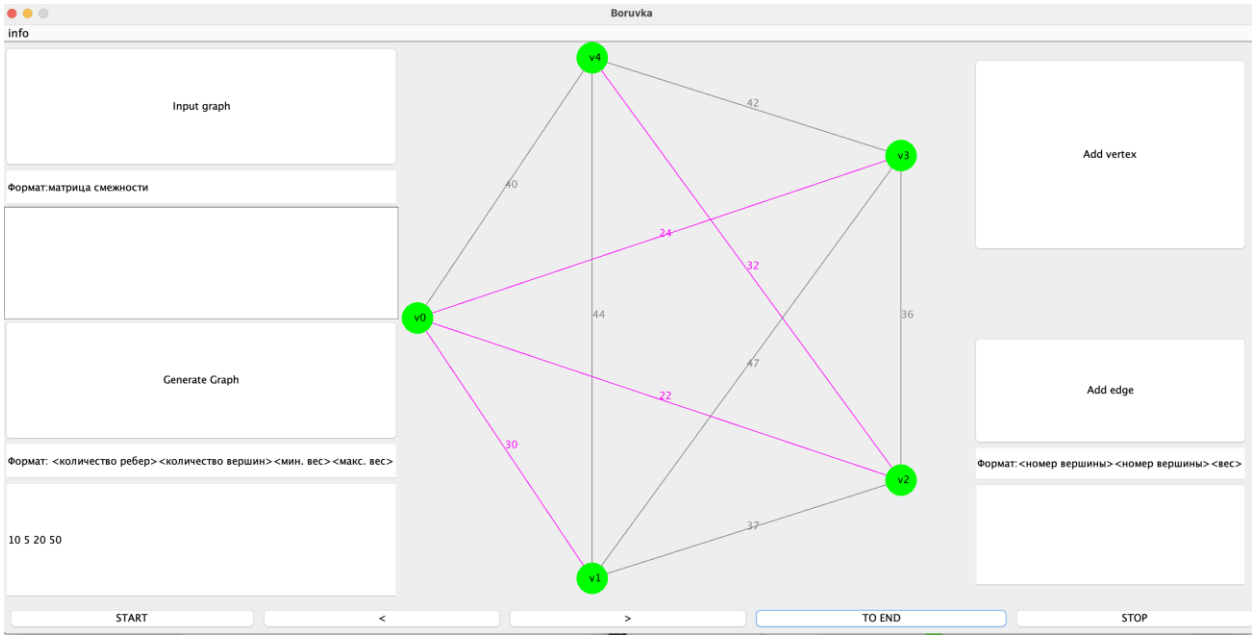

Рисунок 9 – Демонстрация работы не итеративного алгоритма.

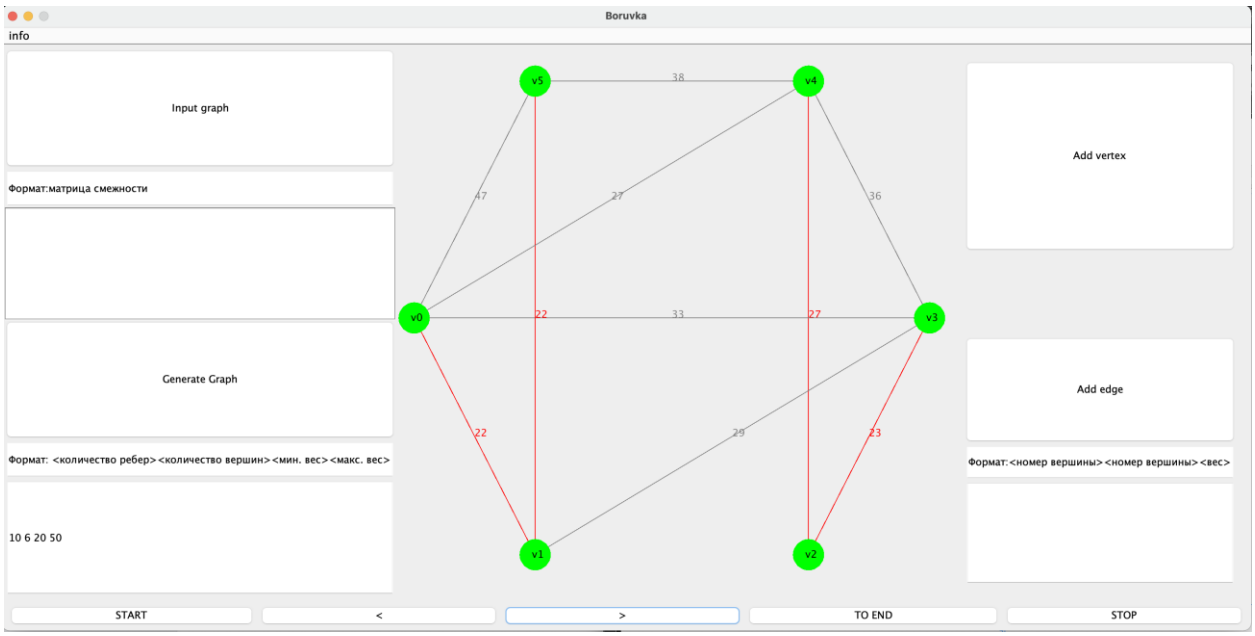

Рисунок 10.1. – Демонстрация работы итеративного алгоритма.

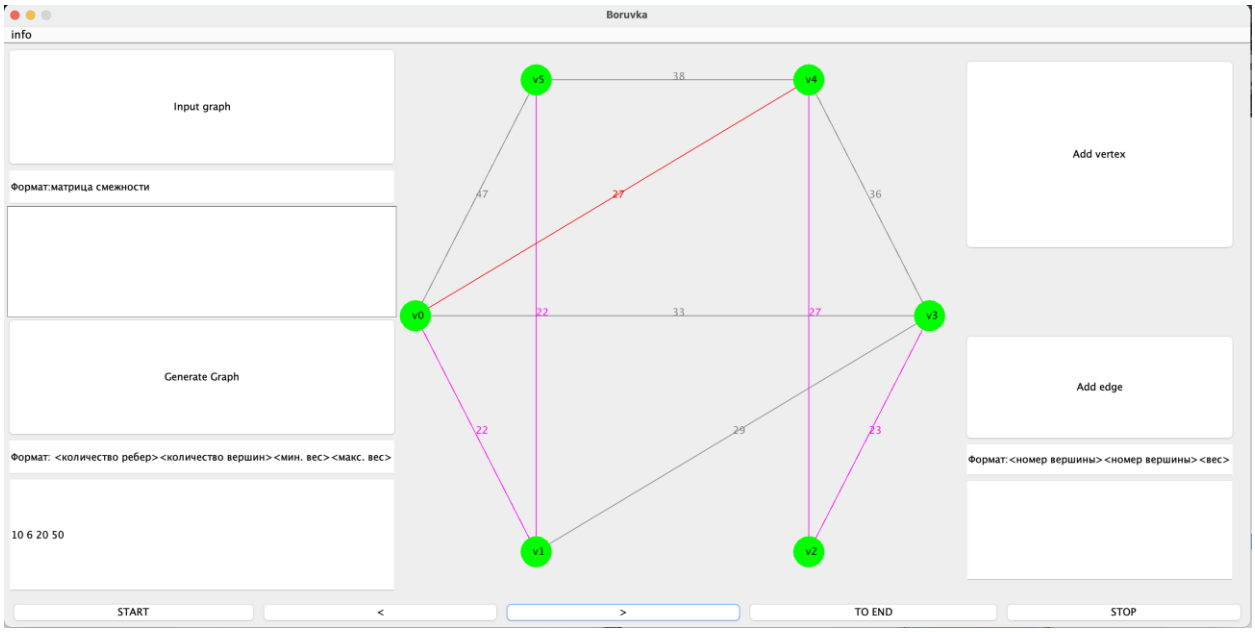

Рисунок 10.2. – Демонстрация работы итеративного алгоритма.

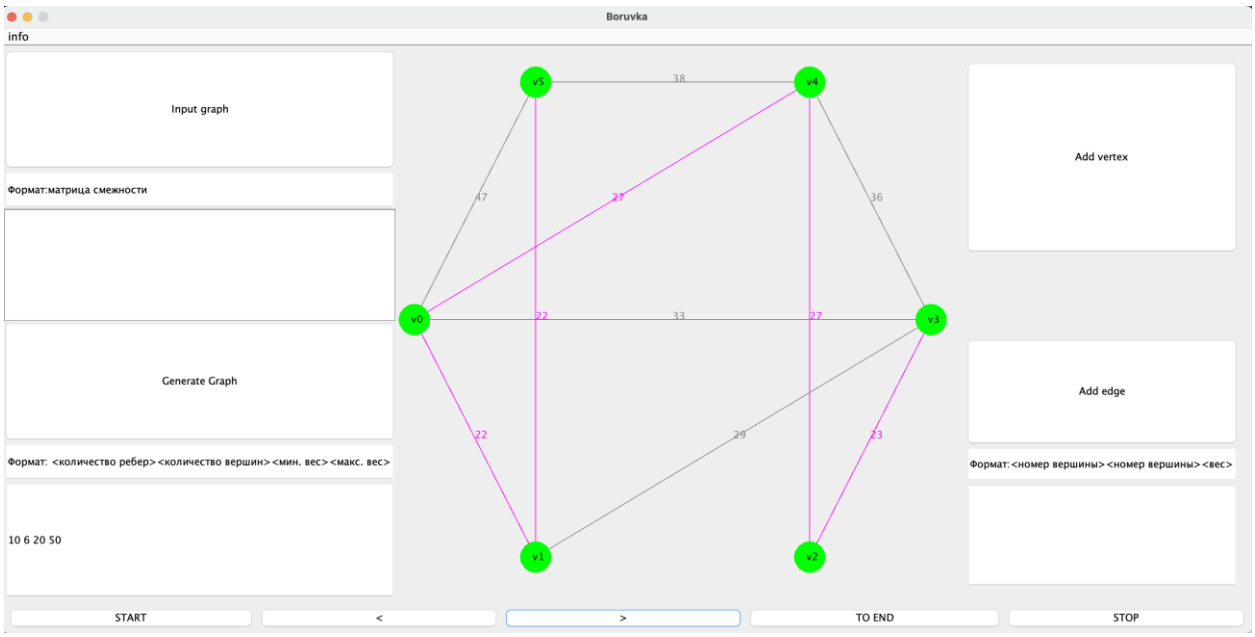

Рисунок 10.3. – Демонстрация работы итеративного алгоритма (конец).

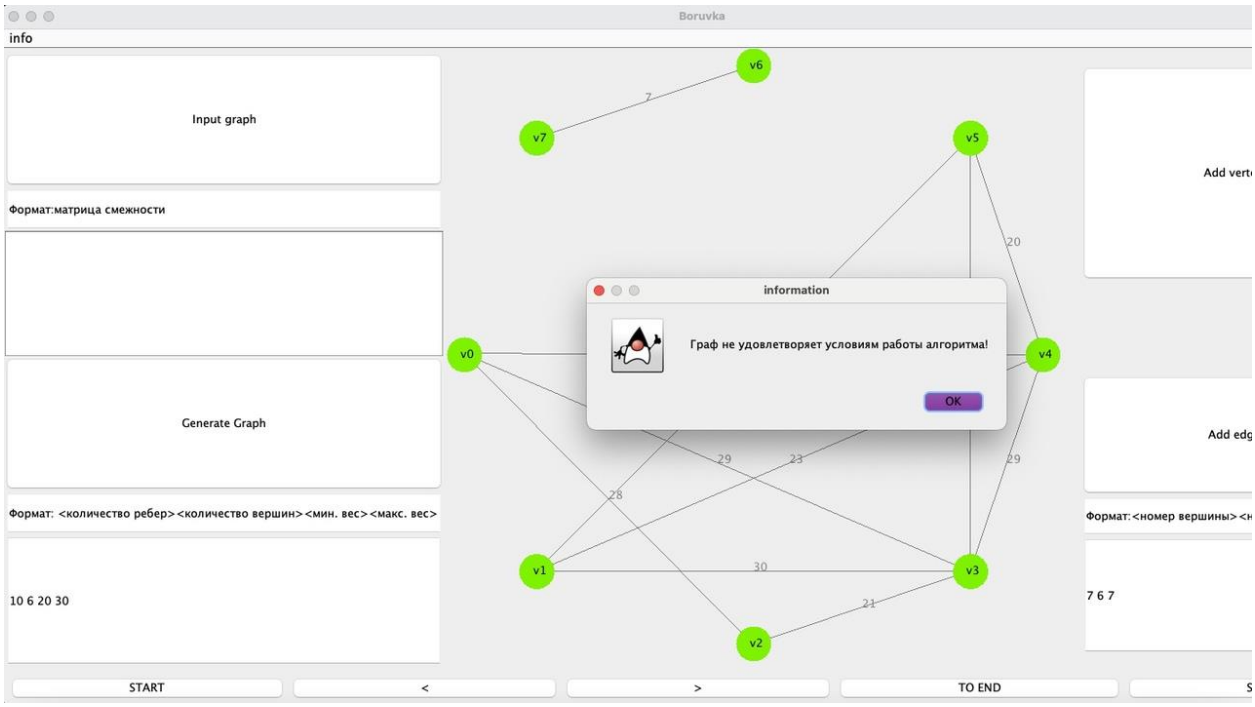

Рисунок 11 – Демонстрация обработки несвязного графа.

ЗАКЛЮЧЕНИЕ

В результате выполнения практической работы, командой была разработана программа с графическим интерфейсом для визуализации алгоритма Борувки.

В программе присутствует реализована возможность ввод графа вручную, а так же автоматически.

Присутствует возможность взаимодействия с элементами графа (удаление, добавление), а так же предусмотрена возможность выполнения алгоритма итеративно, а так же получения конечного ответа сразу.

Обработаны особые случаи, на которых программа может повести себя некорректно.

Для всего функционала программы, в её интерфейсе присутствуют сопутствующие кнопки.

В ходе выполнения работы, были получены практические навыки в работе с высокоуровневым языком программирования Java, изучены библиотеки, позволяющие работать с графическим интерфейсом.