

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

ОТЧЕТ
по учебной практике
Тема: «Минимальное остовное дерево»

Студент гр. 1384

Бобков В.Д.

Студентка гр. 1384

Усачева Д.В.

Студентка гр. 1384

Пчелинцева К.Р.

Руководитель

Фирсов М.А.

Санкт-Петербург

2023

ЗАДАНИЕ НА УЧЕБНУЮ ПРАКТИКУ

Студент Бобков В.Д. группы 1384

Студентка Усачева Д.В. группы 1384

Студентка Пчелинцева К.Р группы 1384

Тема практики: Минимальное остовное дерево

Задание на практику:

Командная итеративная разработка визуализатора алгоритма на Kotlin с графическим интерфейсом.

Алгоритм: ближайшего соседа [Prim].

Сроки прохождения практики: 30.06.2023 – 13.07.2023

Дата сдачи отчета: 12.07.2023

Дата защиты отчета: 12.07.2023

Студент

Бобков В.Д.

Студентка

Усачева Д.В.

Студентка

Пчелинцева К.Р.

Руководитель

Фирсов М.А.

АННОТАЦИЯ

Данный проект посвящён визуализации алгоритма Прима, используемого для решения задачи минимального остовного дерева, на ЯП Kotlin. В работе представлено краткое описание алгоритма, его ключевые шаги и основные принципы работы. Основной акцент делается на визуализации процесса построения минимального остовного дерева с использованием графических элементов.

Выполнение работы состоит из таких элементов как: создание спецификации и плана тестирования; написание кода, реализующего алгоритм; визуализация алгоритма; написание отчета. Входные данные могут задаваться тремя способами: граф считывается из файла, с консоли, либо в режиме реального времени создается самим пользователем с использованием программных инструментов.

SUMMARY

This project is a visualization of Prim's algorithm used to solve the minimum spanning tree problem in Kotlin. The paper presents a brief description of the algorithm, its key steps and basic principles of operation. The main focus is on visualizing the process of building a minimum spanning tree using graphic elements.

The execution of work consists of such elements as: creation of a specification and a test plan; writing code that implements the algorithm; algorithm visualization; writing a report. The input data can be specified in three ways: the graph is read from a file, from the console, or it is created in real time by the user using software tools.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	5
1. ТРЕБОВАНИЯ К ПРОГРАММЕ	6
1.1. Исходные Требования к программе.....	6
1.1.1. Формальная постановка задачи	6
1.1.2. Описание интерфейса	6
1.1.3. Формат входных и выходных данных	8
1.2. Уточнения требований после сдачи 1-ой версии	9
2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ	10
2.1. План разработки.....	10
2.2. Распределение ролей в бригаде.....	10
3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ.....	12
3.1. Структуры данных.....	12
3.2. Основные методы	12
4. ТЕСТИРОВАНИЕ	14
4.1. Тестирование основных функций продукта	14
4.2. Тестирование граничных условий	16
4.3. Тестирование интерфейса.....	19
4.4. Тестирование структуры данных.....	23
ЗАКЛЮЧЕНИЕ	27
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	28

ВВЕДЕНИЕ

Целью данного проекта является изучение языка Kotlin, GUI, а также разработка программы, решающую задачу нахождения минимального остовного дерева, с графическим интерфейсом. Этот интерфейс позволит пользователя взаимодействовать с программой.

Задачи проекта:

- Реализация алгоритма Прима.
- Визуализация графа: создание и отображение графа, на котором будет выполняться алгоритм Прима.
- Подсветка выбранных ребер: выделение выбранных ребер в остовном дереве разным цветом или толщиной, чтобы легче отследить процесс построения дерева.
- Возможность пользовательского ввода графа: добавление возможности ввода пользователем графа с помощью интерфейса приложения. Это позволит пользователям проверять алгоритм Прима на своих собственных данных.

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Исходные Требования к программе

1.1.1. Формальная постановка задачи

Имеется следующий неориентированный взвешенный граф. Назовем остовным деревом подграф, содержащий все вершины исходного графа, который является деревом. И задача состоит в том, чтобы найти такое остовное дерево, сумма рёбер которого минимальна и визуализировать процесс его нахождения. Псевдокод рассматриваемого алгоритма представлен на рисунке 1.

```
 $T \leftarrow \{\}$   
Для каждой вершины  $i \in V$   
   $d[i] \leftarrow \infty$   
   $p[i] \leftarrow nil$   
 $d[1] \leftarrow 0$   
  
 $Q \leftarrow V$   
 $v \leftarrow Extract.Min(Q)$   
  
Пока  $Q$  не пуста  
  Для каждой вершины  $u$  смежной с  $v$   
    Если  $u \in Q$  и  $w(v, u) < d[u]$   
       $d[u] \leftarrow w(v, u)$   
       $p[u] \leftarrow v$   
     $v \leftarrow Extract.Min(Q)$   
   $T \leftarrow T + (p[v], v)$ 
```

Рисунок 1 – Псевдокод алгоритма Прима

1.1.2. Описание интерфейса

Графический интерфейс, реализуемый для решения поставленной задачи, будет представлять собой окно, на котором изображены кнопки для перехода на следующий или предыдущий шаг алгоритма, а также на первый и последний шаги, неориентированный граф, полученный на данном шаге алгоритма, с текстовым пояснением происходящего.

Также подразумевается возможность взаимодействия с графическими элементами (переход на предыдущий или на следующий шаг алгоритма, задание начальных условий – добавление вершин и ребер при помощи мышки и клавиатуры). Эскиз интерфейса представлен на рисунке 2. Описание элементов эскиза: 1 – неориентированный граф, 2 – кнопки для перехода на следующий и предыдущий шаг алгоритма, а также на первый и последний шаги, 3 – текстовое пояснение происходящего на данном шаге, 4 – удаление элемента графа, 5 – добавление вершины, 6 – добавление ребра. Интерфейс должен быть ясным и удобным для пользователя.

Для создания графа в приложении необходимо создать некоторое количество вершин и ребер. Чтобы создать вершину пользователю необходимо нажать на кнопку добавления вершины и выбрать место для её добавления. Чтобы создать ребро пользователю необходимо нажать на кнопку добавления ребра, выбрать щелчком две вершины, между которыми будет нарисовано ребро, и ввести вес ребра». Если же пользователь хочет удалить вершину или ребро, ему необходимо нажать на кнопку удаления, а после мышкой выбрать необходимый элемент.

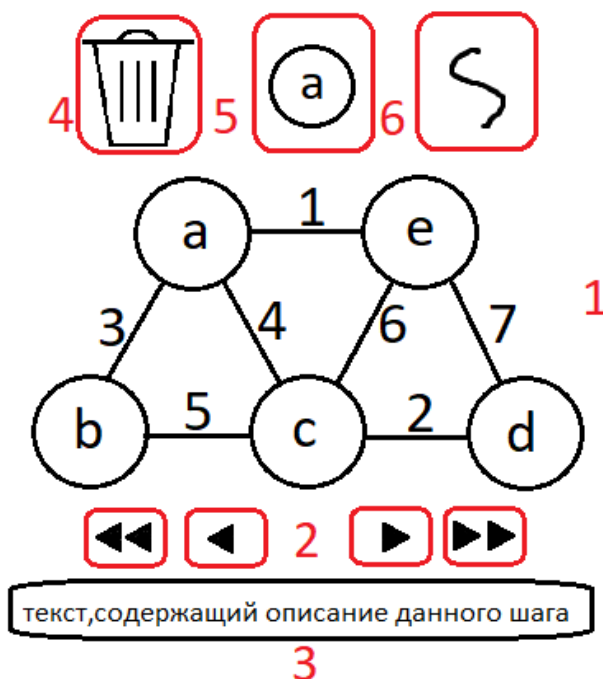


Рисунок 2 – Эскиз интерфейса

1.1.3. Формат входных и выходных данных

Для данной задачи должен быть предусмотрен ввод входных данных разными способами. Перед запуском приложения пользователю будут предложены варианты вида ввода графа: ввод из файла, ввод из консоли, создание графа в приложении. При выборе первых двух вариантов приложение запустится с заданным заранее графом, который пользователь может отредактировать перед началом работы алгоритма. Если же пользователь выберет третий вариант, то ему необходимо будет полностью создать неориентированный граф.

Задаётся неориентированный взвешенный граф $G = (V, E)$ в виде квадратной симметричной матрицы смежности, где $V(|V|=n)$ – это вершины графа; $E(|E|=m)$ – это ребра между вершинами графа.

Каждому ребру m_{ij} можно сопоставить критерий выгодности маршрута – вес ребра (натуральное число, формат числа int), $m_{ij}=-1$, если $i=j$ или ребро между вершинами i и j не существует.

Входные данные задаваемые из файла также представляют собой двумерный массив – матрицу смежности, заданную вышеописанным способом.

Выходные данные должны содержать пошаговую визуализацию работы алгоритма и текстовое описание проделанных действий. На каждом шаге уже выбранные вершины и ребра будут подсвечиваться красным цветом, а множество возможных ребер для следующего выбора – синим.

Текстовые сообщения будут иметь вид: «На шаге «номер шага» была добавлена вершина «имя вершины». Ребра, рассматриваемые на данном шаге, имеют веса: «последовательность весов ребер».» для промежуточных шагов, «Вершина «имя вершины» была выбрана в качестве начальной. Ребра, рассматриваемые на данном шаге, имеют веса «последовательность весов ребер»» для начального и «На шаге «номер шага» была добавлена вершина «имя вершины». Построение минимального остовного дерева окончено.» для конечного.

1.2. Уточнения требований после сдачи 1-ой версии

- [Описать формат матрицы смежности в файле/консоли.]
- До матрицы смежности в первой строке должна быть возможность задать имена вершин.
- Возможность задать стартовую вершину щелчком мыши.
- Корректная обработка ошибки отсутствия файла.
- Увеличить размер шрифта весов рёбер и чуть-чуть уменьшить размер вершин.

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

3 июля (понедельник) – первичное согласование спецификации и плана разработки.

5 июля (среда) – согласование скорректированных спецификации и плана разработки. Первичная сдача прототипа. Прототип должен содержать рабочий код рассматриваемого алгоритма и демонтировать интерфейс без реализации основных функций. Сдача плана тестирования.

7 июля (пятница) – сдача скорректированного прототипа и плана тестирования, первичная сдача 1-ой версии . 1-ая версия демонстрирует частичный функционал графического интерфейса – перемещение между шагами работы алгоритма. Ввод в данной версии алгоритма доступен только двумя способами – из консоли и из файла. Также не предусмотрено редактирование графа в приложении перед началом работы алгоритма.

10 июля (понедельник) – сдача скорректированной 1-ой версии и первичная сдача 2-ой версии. 2-ая версия содержит дополнение функционала – задание начальных условий через графический интерфейс и предварительное редактирование графа перед запуском алгоритма.

12 июля (среда) – сдача скорректированной 2-ой версии, сдача финальной версии с отчётом. Финальная версия содержит рабочий алгоритм с корректной пошаговой визуализацией и все необходимые дополнения функционала.

13 июля (четверг) – сдача финальной версии с отчётом со всеми требуемыми правками.

2.2. Распределение ролей в бригаде

Студент Бобков В.Д. группы 1384:

- Реализация первичного графического интерфейса;
- Реализация возможности задания начальных условий через графический интерфейс.

Студентка Усачева Д.В. группы 1384:

- Реализация перемещения между шагами работы алгоритма;
- Преобразование выходных данных работы алгоритма в удобный формат для последующей визуализации.

Студентка Пчелинцева К.Р группы 1384:

- Реализация алгоритма Прима;
- Тестирование программы и создание плана тестирования.

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1. Структуры данных

1. Класс *Graph* – открытый класс, содержащий поля и методы необходимые для работы с графом, представленным в виде матрицы смежности.

Данный класс имеет следующие поля:

- Поле *name_vertex* – изменяемый список имен вершин графа.
- Поле *data* – изменяемый двумерный список с весами.

3.2. Основные методы

1. Методы класса *Graph*:

- Метод *fun default_name(quotient: Int): String* – задаёт имена вершин по умолчанию в количестве *quotient*.
- Метод *fun read_from_file(filename: String)* – для чтения матрицы из файла и записи её в поле класса *data*. В зависимости от выбранной опции имена для вершин задаются по умолчанию при помощи метода *default_name*, либо происходит считывание с консоли.
- Метод *fun read_from_console()* – для чтения матрицы с консоли. В зависимости от выбранной опции имена для вершин задаются по умолчанию при помощи метода *default_name*, либо происходит считывание с консоли. После чего происходит построчное считывание данных и запись их в поле класса *data*.
- Метод *fun reflect_matrix()* – для приведения нижнетреугольной матрицы к матрице смежности.
- Метод *fun check_correct_matrix(): Boolean* – вспомогательная функция для проверки правильной записи нижнетреугольной матрицы.
- Метод *fun check_symmetry()* – для проверки матрицы на симметричность.
- Метод *fun modify_weights()* - для переопределения весов ребер (если в файле было -1, значит ребра нет).
- Метод *override fun toString(): String* - возвращающий строку для вывода

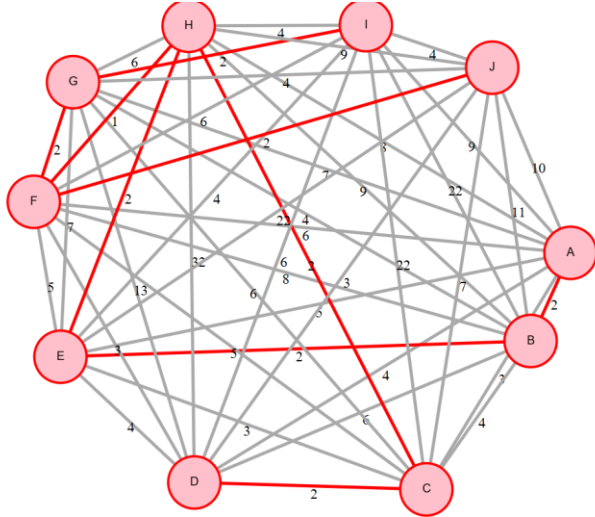
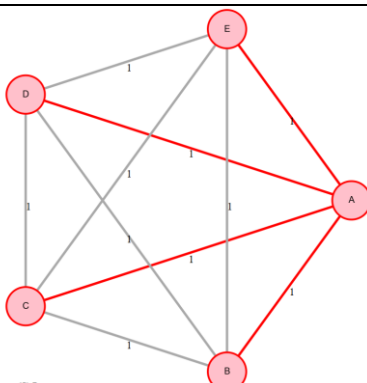
информации о графе в консоль.

- Метод *fun set_start_vertex(new_start_vertex: Int)* – для переопределения стартовой вершины.
- Метод *fun PrimAlgorithm()* – реализация алгоритма Прима для графа, возвращает массив пар, соответствующих началу и концу ребра. Внутри метода запускается цикл, который выполняется до тех пор, пока количество добавленных вершин в остовное дерево меньше количества вершин в графе.
- Метод *fun correct_graph(): Boolean* – проверка корректности графа.
- Метод *fun iterated_PrimAlgorithm* – реализации итеративного алгоритма Прима. В данной функции происходит поиск смежной вершины с минимальным весом по отношению к текущей вершине. Происходит выбор ребра, добавляемого в остовное дерево.
- Метод *fun create_new_vertex(name: String)* – создание новой вершины.
- Метод *fun delete_vertex(number: Int)* – удаление вершины.
- Метод *fun get_matrix()* – геттер матрицы смежности для графа.
- Метод *fun get_names()* – геттер имен вершин графа.

4. ТЕСТИРОВАНИЕ


4.1. Тестирование основных функций продукта

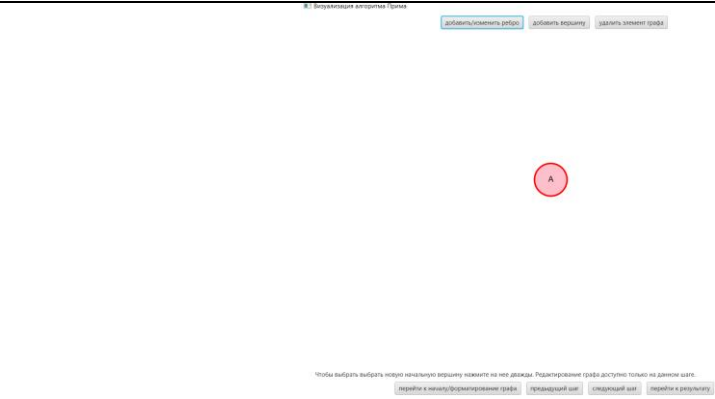
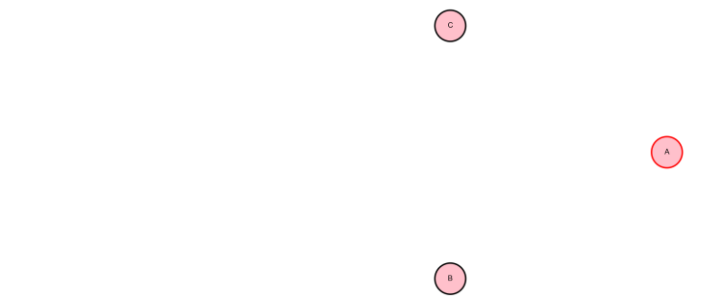
Тест на граф с отрицательными весами ребер	
Входные данные:	Граф с несколькими вершинами и ребрами, имеющими отрицательные веса.
Ожидаемый результат:	Алгоритм должен вернуть минимальное остовное дерево, учитывая отрицательные веса ребер.
Полученный результат:	<p>Корректное выполнение работы.</p>
Тест на граф с одинаковыми весами ребер	
Входные данные:	Граф с несколькими вершинами и ребрами, имеющими одинаковые веса.
Ожидаемый результат:	Алгоритм должен вернуть минимальное остовное дерево, учитывая одинаковые веса ребер.
Полученный результат:	<p>Корректное выполнение работы.</p>

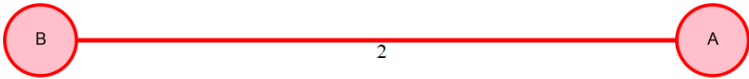
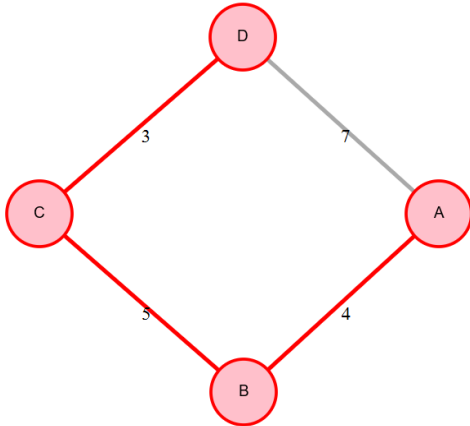
Тест на граф с большим количеством вершин и ребер	
Входные данные:	Граф с большим количеством вершин и ребер.
Ожидаемый результат:	Алгоритм должен вернуть минимальное остовное дерево, учитывая все вершины и связанные ребра в графе.
Полученный результат:	 <p>Корректное выполнение работы.</p>
Тест на граф с множеством возможных минимальных остовных деревьев	
Входные данные:	Граф с несколькими вершинами и ребрами, где существует несколько различных минимальных остовных деревьев.
Ожидаемый результат:	Алгоритм должен вернуть одно из возможных минимальных остовных деревьев, не обязательно совпадающее с другими возможными решениями.
Полученный результат:	 <p>Корректное выполнение работы.</p>

Тест на граф с большим количеством вершин и небольшим количеством ребер	
Входные данные:	Граф с большим количеством вершин и небольшим количеством ребер.
Ожидаемый результат:	Алгоритм должен вернуть минимальное остовное дерево, учитывая все вершины и связанные ребра в графе, даже при ограниченном количестве ребер.
Полученный результат:	 <p>Корректное выполнение работы.</p>

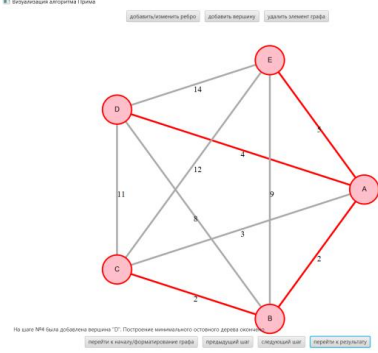
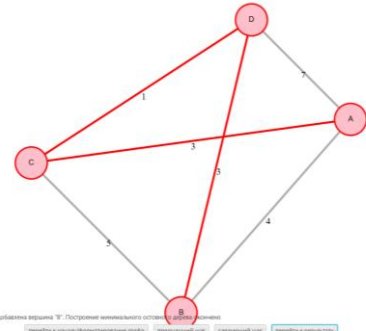
4.2. Тестирование граничных условий

Тест на пустой граф	
Входные данные:	<p>Пустой граф (без вершин и ребер).</p> <pre> Как вы хотите ввести граф : "1" - из файла(матрица смежности), "2" - из консоли(матрица смежности), "3" - в режиме реального времени самому нарисовать граф. 2 Матрица смежности должна быть представлена в формате : симметричная матрица с "-1" на главной диагонали, где "-1" означает отсутствие ребра между двумя вершинами. Сколько вершин должно быть в графе? 0 </pre>
Ожидаемый результат:	Алгоритм должен вернуть пустое минимальное остовное дерево.
Полученный результат:	 <p>Корректное выполнение работы.</p>
Тест на граф с одной вершиной	

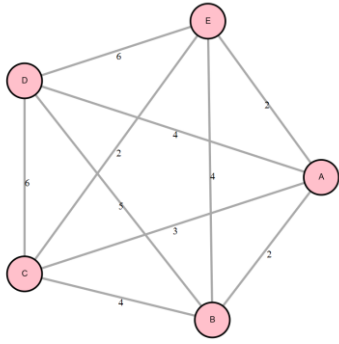

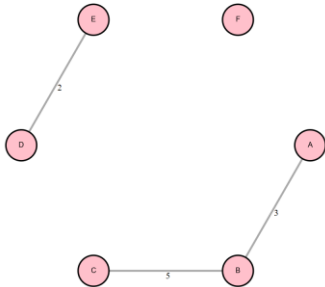
Входные данные:	<p>Граф с одной вершиной.</p> <pre> Как вы хотите ввести граф : "1" - из файла(матрица смежности), "2" - из консоли(матрица смежности), "3" - в режиме реального времени самому нарисовать граф. 2 Матрица смежности должна быть представлена в формате : симметричная матрица с "-1" на главной диагонали, где "-1" означает отсутствие ребра между двумя вершинами. Сколько вершин должно быть в графе? 1 </pre>
Ожидаемый результат:	Алгоритм должен вернуть пустое минимальное остовное дерево.
Полученный результат:	 <p>Созданный граф является несвязным или задано не более двух вершин.</p> <p>Корректное выполнение работы.</p>
Тест на граф без ребер	
Входные данные:	<p>Граф с несколькими вершинами, но без ребер.</p> <pre> Введите матрицу смежности: -1 -1 -1 -1 -1 -1 -1 -1 -1 </pre>
Ожидаемый результат:	Алгоритм должен вернуть пустое минимальное остовное дерево.
Полученный результат:	 <p>Созданный граф является несвязным или задано не более двух вершин.</p>

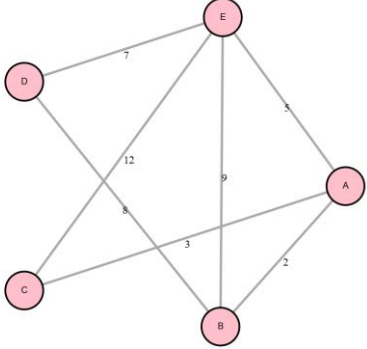
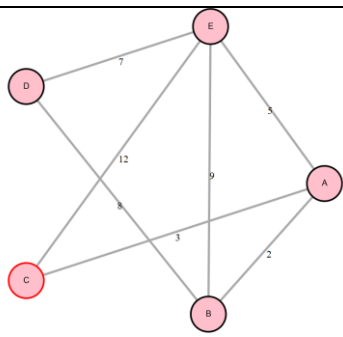
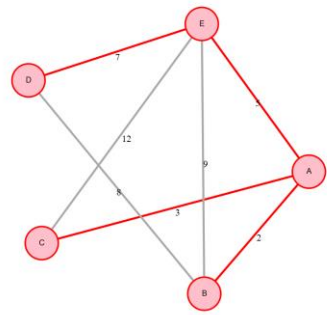
	Корректное выполнение работы.
Тест на граф с одним ребром	
Входные данные:	Граф с двумя вершинами и одним ребром.
Ожидаемый результат:	Алгоритм должен вернуть минимальное остовное дерево, состоящее из этого единственного ребра.
Полученный результат:	 <p>Корректное выполнение работы.</p>
Тест на граф с циклом	
Входные данные:	Граф с несколькими вершинами и циклическим путем.
Ожидаемый результат:	Алгоритм должен вернуть минимальное остовное дерево, исключив ребра, образующие цикл.
Полученный результат:	 <p>Корректное выполнение работы.</p>

4.3. Тестирование интерфейса

Тестирование интерфейса	
Входные данные:	<p>Граф, представленный пользователю через интерфейс и введенный через файл.</p> <p>Как вы хотите ввести граф : "1" - из файла(матрица смежности), "2" - из консоли(матрица смежности), "3" - в режиме реального времени самому нарисовать граф.</p> <p>1</p> <p>Матрица смежности должна быть представлена в формате : симметричная матрица с "-1" на главной диагонали, где "-1" означает отсутствие ребра между двумя вершинами.</p> <p>Если вы желаете задать имена вершин в файле, введите "1". Введите "2", чтобы имена вершин были заданы автоматически.</p> <p>2</p>
Ожидаемый результат:	Алгоритм должен корректно обработать введенные пользователем данные и вернуть минимальное остовное дерево для данного графа.
Полученный результат:	 <p>Корректное выполнение работы.</p>
Входные данные:	<p>Граф, представленный пользователю через интерфейс и введенный через консоль.</p> <p>Как вы хотите ввести граф : "1" - из файла(матрица смежности), "2" - из консоли(матрица смежности), "3" - в режиме реального времени самому нарисовать граф.</p> <p>2</p> <p>Матрица смежности должна быть представлена в формате : симметричная матрица с "-1" на главной диагонали, где "-1" означает отсутствие ребра между двумя вершинами.</p> <p>Сколько вершин должно быть в графе?</p> <p>4</p> <p>Если вы желаете задать имена вершин, введите "1". Введите "2", чтобы имена вершин были заданы автоматически.</p> <p>2</p> <p>Введите матрицу смежности:</p> <pre>-1 4 3 7 4 -1 5 3 3 5 -1 1 7 3 1 -1</pre>
Ожидаемый результат:	Алгоритм должен корректно обработать введенные пользователем данные и вернуть минимальное остовное дерево для данного графа.
Полученный результат:	 <p>Корректное выполнение работы.</p>

<p>Входные данные:</p>	<p>Граф, представленный пользователю через интерфейс и введенный через файл (нижнестреугольная матрица).</p> <div data-bbox="432 264 1350 584"> <p>Как вы хотите ввести граф : "1" - из файла(матрица смежности), "2" - из консоли(матрица смежности), "3" - в режиме реального времени самому нарисовать граф.</p> <p>1</p> <p>Матрица смежности должна быть представлена в формате : симметричная матрица с "-1" на главной диагонали, где "-1" означает отсутствие ребра между двумя вершинами.</p> <p>Если вы желаете задать имена вершин в файле, введите "1". Введите "2", чтобы имена вершин были заданы автоматически.</p> <p>2</p> </div>
<p>Ожидаемый результат:</p>	<p>Алгоритм должен корректно обработать введенные пользователем данные и вернуть минимальное остовное дерево для данного графа.</p>
<p>Полученный результат:</p>	<div data-bbox="692 712 1190 1178"> <p>На шаге КМ была добавлена вершина "D". Построение минимального остовного дерева окончено.</p> <p>перейти к началу/форматирование графа предыдущий шаг следующий шаг перейти к результату</p> </div> <p>Корректное выполнение работы.</p>
<p>Входные данные:</p>	<p>Граф, представленный пользователю через интерфейс и введенный через консоль (нижнестреугольная матрица)..</p> <div data-bbox="432 1368 1034 1809"> <p>Как вы хотите ввести граф :</p> <p>"1" - из файла(матрица смежности),</p> <p>"2" - из консоли(матрица смежности),</p> <p>"3" - из файла(нижнестреугольная матрица),</p> <p>"4" - из консоли(нижнестреугольная матрица),</p> <p>"5" - в режиме реального времени самому нарисовать граф.</p> <p>4</p> <p>Матрица должна быть представлена в формате : нижнестреугольная матрица с "-1" на главной диагонали, где "-1" означает отсутствие ребра между двумя вершинами.</p> <p>Сколько вершин должно быть в графе?</p> <p>5</p> <p>Если вы желаете задать имена вершин, введите "1". Введите "2", чтобы имена вершин были заданы автоматически.</p> <p>2</p> <p>Введите матрицу:</p> <pre>-1 2 -1 3 4 -1 4 5 6 -1 2 4 2 6 -1</pre> </div>
<p>Ожидаемый результат:</p>	<p>Алгоритм должен корректно обработать введенные пользователем данные и вернуть минимальное остовное дерево для данного графа.</p>

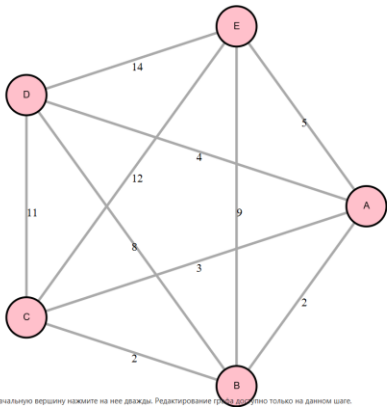
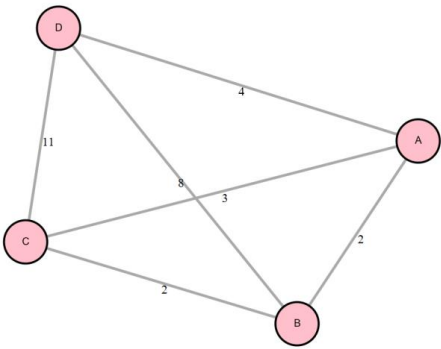
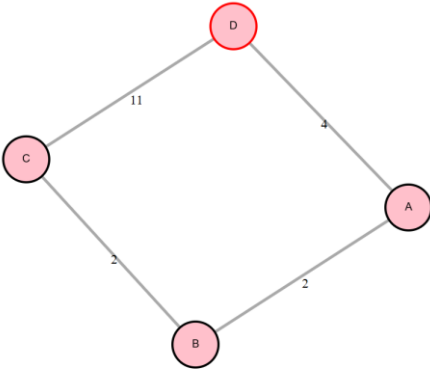
Полученный результат:	 <p>Корректное выполнение работы.</p>
Тест на случай отсутствия графа	
Входные данные:	Отсутствие графа
Ожидаемый результат:	Алгоритм должен обработать данную ситуацию и вернуть пустой результат.
Полученный результат:	 <p>Корректное выполнение работы.</p>
Тест на случай некорректных входных данных	
Входные данные:	<p>Граф, содержащий некорректные или неполные данные (несвязный граф)</p> 
Ожидаемый результат:	Алгоритм должен обработать данную ситуацию и вернуть сообщение об ошибке или пустой результат
Полученный результат:	<p>Созданный граф является несвязным или задано не более двух вершин.</p> <p>Корректное выполнение работы.</p>

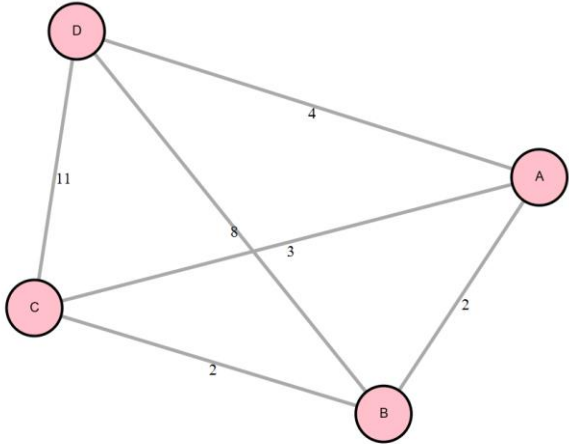
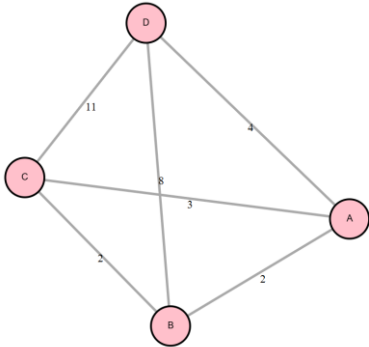
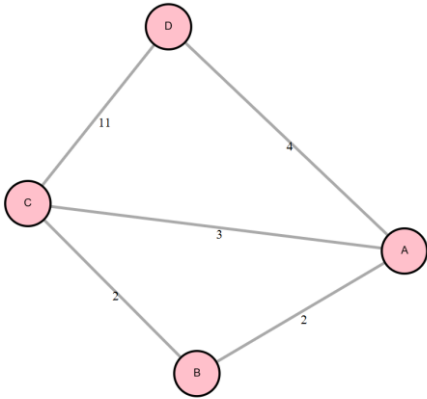
Тест на задание новой стартовой вершины двойным щелчком	
Входные данные:	<p>Граф с несколькими вершинами и ребрами.</p> 
Ожидаемый результат:	После двойного щелчка должна быть выделена новая стартовая вершина.
Полученный результат:	 <p>Корректное выполнение работы.</p>
Тест на блокировку кнопок	
Входные данные:	Граф с несколькими вершинами и ребрами.
Ожидаемый результат:	После получения результата должна будет произойти блокировка кнопок: «добавить/изменить ребро», «добавить вершину», «удалить элемент графа».
Полученный результат:	 <p>Корректное выполнение работы.</p>

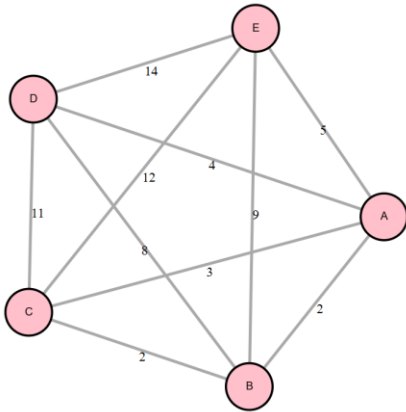
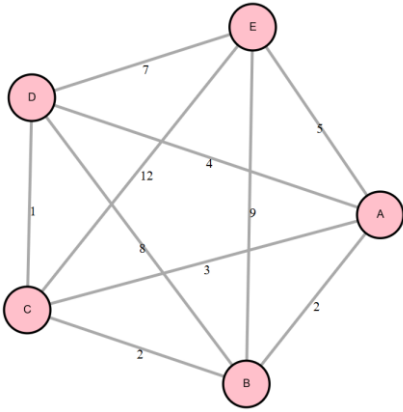
Тест на ввод имён вершин с консоли	
Входные данные:	<p>Граф с несколькими вершинами и ребрами. Имена вершин задаются с консоли.</p> <pre> Если вы желаете задать имена вершин, введите "1". Введите "2", чтобы имена вершин были заданы автоматически. 1 Введите имена вершин через пробел: k f g r </pre>
Ожидаемый результат:	Вершины будут иметь имена через консоль.
Полученный результат:	<p>Корректное выполнение работы.</p>

4.4. Тестирование структуры данных

Тест на добавление вершины в граф	
Входные данные:	Пустой граф, команда добавления вершины.
Ожидаемый результат:	Вершина успешно добавлена в граф.
Полученный результат:	<p>Корректное выполнение работы.</p>

Тест на удаление вершины из графа	
Входные данные:	<p>Граф с несколькими вершинами, команда удаления одной из вершин.</p>  <p><small>На новую начальную вершину нажмите на нее дважды. Редактирование графа возможно только на данном шаге.</small></p>
Ожидаемый результат:	Вершина успешно удалена из графа и связанные с ней ребра также удалены.
Полученный результат:	 <p>Корректное выполнение работы.</p>
Тест на добавление ребра в граф	
Входные данные:	<p>Граф с несколькими вершинами, команда добавления ребра между двумя вершинами.</p> 

Ожидаемый результат:	Ребро успешно добавлено в граф.
Полученный результат:	 <p>Корректное выполнение работы.</p>
Тест на удаление ребра из графа	
Входные данные:	<p>Граф с несколькими вершинами и ребрами, команда удаления одного из ребер.</p> 
Ожидаемый результат:	Ребро успешно удалено из графа.
Полученный результат:	 <p>Корректное выполнение работы.</p>

Тест на изменение веса ребра	
Входные данные:	<p>Граф с некоторым количеством вершин и количеством ребер.</p> 
Ожидаемый результат:	Успешная замена веса ребра.
Полученный результат:	<p>Вес на ребрах DE, CD успешно изменен .</p>  <p>Корректное выполнение работы.</p>

ЗАКЛЮЧЕНИЕ

Кратко подвести итоги, проанализировать соответствие поставленной цели и полученного результата.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Ниже представлены примеры библиографического описания, В КАЧЕСТВЕ НАЗВАНИЯ ИСТОЧНИКА в примерах приводится вариант, в котором применяется то или иное библиографическое описание.

1. Иванов И. И. Книга одного-трех авторов. М.: Издательство, 2010. 000 с.
2. Книга четырех авторов / И. И. Иванов, П. П. Петров, С. С. Сидоров, В. В. Васильев. СПб.: Издательство, 2010. 000 с.
3. Книга пяти и более авторов / И. И. Иванов, П. П. Петров, С. С. Сидоров и др.. СПб.: Издательство, 2010. 000 с.
4. Описание книги под редакцией / под ред. И.И. Иванова СПб., Издательство, 2010. 000 с.
5. Иванов И.И. Описание учебного пособия и текста лекций: учеб. пособие. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2010. 000 с.
6. Описание методических указаний / сост.: И.И. Иванов, П.П. Петров. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2010. 000 с.
7. Иванов И.И. Описание статьи с одним-тремя авторами из журнала // Название журнала. 2010, вып. (№) 00. С. 000–000.
8. Описание статьи с четырьмя и более авторами из журнала / И. И. Иванов, П. П. Петров, С. С. Сидоров и др. // Название журнала. 2010, вып. (№) 00. С. 000–000.
9. Иванов И.И. Описание тезисов доклада с одним-тремя авторами / Название конференции: тез. докл. III международной науч.-техн. конф., СПб, 00–00 янв. 2000 г. / СПбГЭТУ «ЛЭТИ», СПб, 2010, С. 000–000.
10. Описание тезисов доклада с четырьмя и более авторами / И. И. Иванов, П. П. Петров, С. С. Сидоров и др. // Название конференции: тез. докл. III международной науч.-техн. конф., СПб, 00–00 янв. 2000 г. / СПбГЭТУ «ЛЭТИ», СПб, 2010, С. 000–000.
11. Описание электронного ресурса // Наименование сайта. URL: <http://east-front.narod.ru/memo/latchford.htm> (дата обращения: 00.00.2010).

12. ГОСТ 0.0–00. Описание стандартов. М.: Изд-во стандартов, 2010.
13. Пат. RU 000000000. Описание патентных документов / И. И. Иванов, П. П. Петров, С. С. Сидоров. Оpubл. 00.00.2010. Бюл. № 00.
14. Иванов И.И. Описание авторефератов диссертаций: автореф. дисс. канд. техн. наук / СПбГЭТУ «ЛЭТИ», СПб, 2010.
15. Описание федерального закона: Федер. закон [принят Гос. Думой 00.00.2010] // Собрание законодательств РФ. 2010. № 00. Ст. 00. С. 000–000.
16. Описание федерального постановления: постановление Правительства Рос. Федерации от 00.00.2010 № 00000 // Опубликовавшее издание. 2010. № 0. С. 000–000.
17. Описание указа: указ Президента РФ от 00.00.2010 № 00 // Опубликовавшее издание. 2010. № 0. С. 000–000.

ПРИЛОЖЕНИЕ А

НАЗВАНИЕ ПРИЛОЖЕНИЯ

полный код программы должен быть в приложении, печатать его не надо