**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

По лабораторному практикуму

Тема: Поиск минимального остовного дерева, с помощью алгоритма Борувки.

|  |  |  |
| --- | --- | --- |
| Студенты гр. 6383 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Никитин К.В. |
| гр. 6383 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Спас А.А. |
| гр.6383  Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Карамышев А.О.  Ефремов М.А. |

Санкт-Петербург

2018

**ЗАДАНИЕ**

**на учебную практику**

|  |  |  |
| --- | --- | --- |
| Студент Никитин К.В. группы 6383 | | |
| Студент Спас А.А. группы 6383 | | |
| Студент Карамышев А.О. группы 6383  Тема практики: Нахождение минимального остовного дерева с помощью алгоритма Борувки | | |
| Задание на практику:  Командная итеративная разработка визуализатора алгоритма Борувки на Java с графическим интерфейсом. | | |
| Сроки прохождения практики: 27.06.2018 – 10.07.2018 | | |
| Дата сдачи отчета: | | |
| Дата защиты отчета: | | |
|  | | |
| Студент гр.6383 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Никитин К.В. |
| Студент гр.6383 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Спас А.А. |
| Студент гр.6383 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Карамышев А.О. |
| Руководитель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Ефремов М.А. |

**Аннотация**

В данной работе рассмотрена программа, которая ищет минимальное остовное дерево с помощью алгоритма Борувки. Программа разработана в среде IntelliJ IDEA. Язык разработанной программы Java. Программа подробно показывает процесс нахождения минимального остовного дерева.

**Summary**

In this paper, we consider a program that searches for a minimal spanning tree using the Borovka algorithm. The program is developed in the IntelliJ IDEA environment. The language of the developed Java program. The program details the process of finding the minimal spanning tree.

**Введение.**

**Формулировка задания.** Требуется разработать программу, поиск минимального остовного дерева с помощью алгоритма Борувки. При этом должен присутствовать графический интерфейс.

**Алгоритм Борувки** (англ. *Borůvka's algorithm*) — алгоритм поиска [минимального остовного дерева](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9E%D1%81%D1%82%D0%BE%D0%B2%D0%BD%D1%8B%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D1%8C%D1%8F:_%D0%BE%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F,_%D0%BB%D0%B5%D0%BC%D0%BC%D0%B0_%D0%BE_%D0%B1%D0%B5%D0%B7%D0%BE%D0%BF%D0%B0%D1%81%D0%BD%D0%BE%D0%BC_%D1%80%D0%B5%D0%B1%D1%80%D0%B5) во взвешенном неориентированном связном графе. Впервые был опубликован в 1926 году Отакаром Борувкой.

Алгоритм состоит из нескольких шагов:

1. Изначально каждая вершина графа G— тривиальное дерево, а ребра не принадлежат никакому дереву.
2. Для каждого дерева T найдем минимальное инцидентное ему ребро. Добавим все такие ребра.
3. Повторяем шаг 2 пока в графе не останется только одно дерево T.

Данный алгоритм может работать неправильно, если в графе есть ребра равные по весу. Например, полный граф из трех вершин, вес каждого ребра равен один. В T могут быть добавлены все три ребра. Избежать эту проблему можно, например, выбирая в первом пункте среди ребер, равных по весу, ребро с наименьшим номером.

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 4 |
| 1. | Требования к программе | 6 |
| 2. | План разработки и распределение ролей в бригаде | 6 |
| 2.1. | План разработки | 6 |
| 2.2. | Распределение ролей в бригаде | 6 |
| 3. | Особенности реализации | 6 |
| 3.1. | Описание классов | 6 |
| 3.2. | Алгоритм работы | 9 |
| 3.3  3.4 | Входные/выходные данные  Описание интерфейса | 9  9 |
| 4. | Тестирование | 10 |
| 4.1 | Тестирование графического интерфейса | 10 |
| 4.2 | Тестирование кода алгоритма | 10 |
|  | Заключение | 10 |
|  | Список использованных источников | 10 |
|  | Приложение А. Код программы. | 11 |
|  | Приложение Б. UML - диаграмма | 11 |

1. **Требование к программе**

Данная программа должна корректно работать на всевозможных входных данных и выводить минимальное остовное дерево.

Программа должна подробно демонстрировать результат работы алгоритма и пошагово показывать его выполнение.

1. **План разработки и распределение ролей в бригаде**
   1. **План разработки:**
   2. **Распределение ролей в бригаде:**

Спас Артём: Визуализация алгоритма.

Карамышев Альберт: Архитектура программы.

Никитин Кирилл: Тестирование, написание отчёта.

1. **Особенности реализации**

**3.1. Описание классов:**

**Canvas.java**

**(DESCRIPTOIN)**

|  |  |
| --- | --- |
| public void setContent(Graph data) |  |
| public void setMST(Graph data) |  |
| public void checkBox() |  |

**Coord.java**

**(DESCRIPTION)**

**Edge.java**

**(DESCRIPTION)**

|  |  |
| --- | --- |
| public double weight() |  |
| public int either() |  |
| public int other() |  |
| public int compareTo(Edge that) |  |
| public boolean equals(Edge that) |  |
| public String toString() |  |

**Graph.java**

**(DESCRIPTION)**

|  |  |
| --- | --- |
| public int V() |  |
| public int E() |  |
| public void addEdge(Edge e) |  |
| public Collection<Edge> adj(int v) |  |
| public Collection<Edge> edges() |  |
| public String toString() |  |

**InputOutput.java**

**(DESCRIPTION)**

|  |  |
| --- | --- |
| public Graph getData(Graph g, BufferedReader cin, BufferedReader cinV) throws IOException |  |

**Launcher.java**

**(DESCRIPTION)**

**????**

**MST.java**

**(DESCRIPTION)**

|  |  |
| --- | --- |
| public void build() |  |
| public void buildStep() |  |
| public Graph MSTtoGraph() |  |
| public boolean equal(MST x) |  |
| private boolean less(Edge e, Edge f) |  |
| public Collection<Edge> edges() |  |
| public double weight() |  |
| public String toString() |  |

**UnionField.java**

**(DESCRIPTION)**

|  |  |
| --- | --- |
| public int count() |  |
| public boolean connected(int p, int q) |  |
| public int find(int p) |  |
| public void unoin(int p, int q) |  |

**VGraph.java**

**(DESCRIPTION)**

|  |  |
| --- | --- |
| public void paint(Graphics g) |  |
| boolean attend(HashMap<String, Object> e) |  |
| private void addVertex(String name) |  |
| private void addEdge(String fromName, String toName, double weight) |  |
| private void addMstEdge(String fromName, String toName, double weight) |  |
| private HashMap<String,Object> vertexLookup(String name) |  |
| private void reposition() |  |
| public void recolor(int id, Color c) |  |
| public void SetMST(MST original) |  |
| public VGraph(Graph original) |  |
| public HashMap<String, Object> find(Point2D p) |  |
| private class MyMouse extends MouseAdapter |  |
| private class MyMove implements MouseMotionListener |  |

**3.2.Алгоритм работы:**

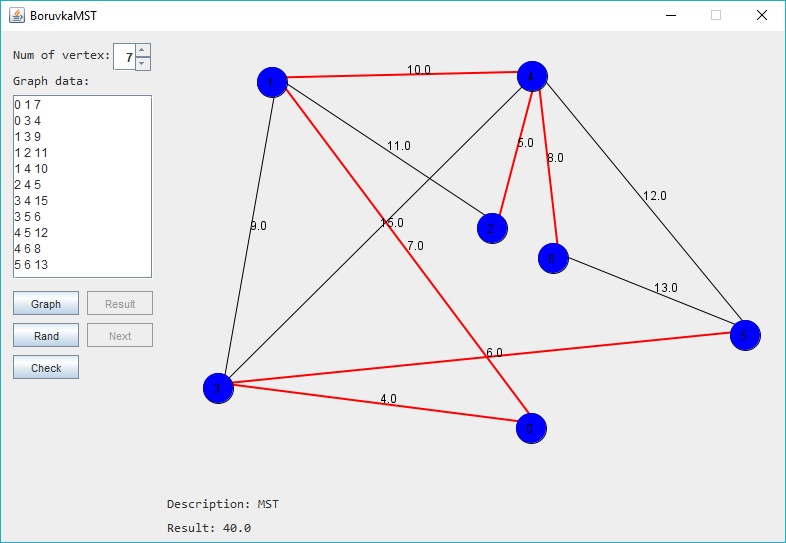
**NAPISHITE PLEASE**

**3.3. Входные/выходные данные:**

На *вход* пользователь вводит количество вершин и все пути в графе.

На *выходе* пользователь получит минимальное остовное дерево.

**3.4. Описание интерфейса:**



Пользовательский интерфейс состоит из :

* Поле ввода количества вершин
* Поле ввода путей в графе
* Кнопка Graph, которая строит граф
* Кнопка Result, которая выводит на экран минимальное остовное дерево
* Кнопка Rand, которая генерирует случайно дерево
* Кнопка Next, которая пошагово ищет минимальное остовное дерево
* Кнопка Check, которая включает/отключает отображение весов

**4. Тестирование**

**4.1. Тестирование графического интерфейса**

Все кнопки нашего графического интерфейса работают корректно. В ходе тестирования никаких ошибок выявлено не было.

**4.2. Тестирование кода алгоритма**

**IN PROGRESS**

**Заключение:**

В ходе данной учебной практики были изучены основы программирования на языке Java, пройден интерактивный курс «Java.Базовый курс». После чего была разработана программа, которая с помощью алгоритма Борувки находит минимальное остовное дерево, наглядно демонстрируя его пользователю.

Программа соответствует требованиям спецификации и прошла проверку на всевозможных тестах, никаких ошибок выявлено не было.

**Список использованных источников:**

1. Java Базовый курс.URL: stepik.org/course/Java-Базовый-курс-187/syllabus

**Приложение А.**

**Код программы.**

IN PROGRESS

**Приложение Б.**

**UML – диаграмма**

