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

СПЕЦИФИКАЦИЯ

по учебной практике

Тема: Минимальное остовное дерево: алгоритм Борувки

Студент гр. 0303 _____ Середенков А.А.

Студент гр. 0303 _____ Сологуб Н.А.

Студент гр. 0303 _____ Пичугин М.В.

Руководитель _____ Фирсов М.А.

Санкт-Петербург

2022

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Исходные Требования к программе

1.1.1. Требования к вводу исходных данных

Исходные данные могут вводится 2 способами:

1) С помощью взаимодействие пользователя с интерфейсом по нажатию мыши на кнопки и графическое полотно. Пользователь нажимает на клавишу для создание вершин графа, чтобы указать программе, что на графическом полотне по нажатию мыши создается вершина графа. Затем пользователь нажимает на клавишу для создания рёбер, и нажимает на две произвольные вершины на полотне для отрисовки и создания ребра между ними.
2) С помощью чтения данных из файла. Приложения сначала проверяет исходный файл на правильность заполнения. Если все верно, то рисуется введенный граф, иначе выведется окно с ошибкой(“WRONG FILE!!!!”).

1.1.2. Требования к визуализации

В левой части программы будет большая область на которой будет отображаться граф, в правом верхнем углу будут находится все необходимые кнопки для работы с приложением, в правом нижнем углу будет находится консоль для вывода логов. В логах будет содержаться следующая информация: текущая рассматриваемая вершина и список инцидентных ей рёбер, минимальное ребро для рассматриваемой вершины, текущее окрашенное дерево и список инцидентных ему рёбер, минимальное ребро для рассматриваемого дерева, сообщение об объединении двух деревьев в одно в ходе добавления ребра к искомому дереву.

Описание кнопок приложения:

Кнопка добавить вершину - при нажатии, в левую часть приложения добавится новая вершина в виде круга с индексом внутри. Индекс задается порядком добавления новой вершины.

Кнопка удалить вершину - при нажатии выведется окно, в котором будет поле, в которое нужно вписать индекс вершины, которую хотим удалить. Вместе с вершиной удалятся и ребра, связанные с ней.

Кнопка добавить ребро - при нажатии выведется окно, в котором будет 3 поля, в которые нужно будет вписать индексы вершин, которые хотим соединить, и вес самого ребра.

Кнопка удалить ребро - при нажатии выведется окно, в котором будет 2 поля, в которые нужно будет вписать индексы вершин, чье ребро хотим удалить.

Кнопка открыть файл - при нажатии откроется окно, в котором можно будет выбрать файл формата *.txt из которого хотим получить исходные данные.

Кнопка сохранить граф - при нажатии откроется окно, в котором можно будет выбрать или создать файл формата *.txt в котором будут записаны параметры графа представленного в левой части приложения.

Кнопка очистить холст - при нажатии очистит левую часть приложения от нарисованного на нем графа.

Кнопка результат - при нажатии выведет конечный результат работы алгоритма в левую часть приложения, а также в правый нижний угол программы, где будет расположен терминал, выведет процесс работы алгоритма в виде логов.

Кнопка шаг вперед - при нажатии запустит алгоритм, который будет выполняться пошагово, то есть, чтобы завершить работу алгоритма нужно будет продолжать нажимать на эту кнопку, до тех пор пока полностью не закончится работа алгоритма и не выведется конечный результат. Шагом в данной реализации алгоритма будет считаться добавление нового ребра к искомому дереву.

В исходном окне приложения пользователю будет предоставляться выбор для ввода данных: через нажатие кнопки мыши или через файл. Если пользователь выберет ввод данных через файл, то ему необходимо будет нажать соответствующую кнопку после чего откроется окно, в котором можно будет выбрать файл формата *.txt из которого хотим получить исходные данные. В

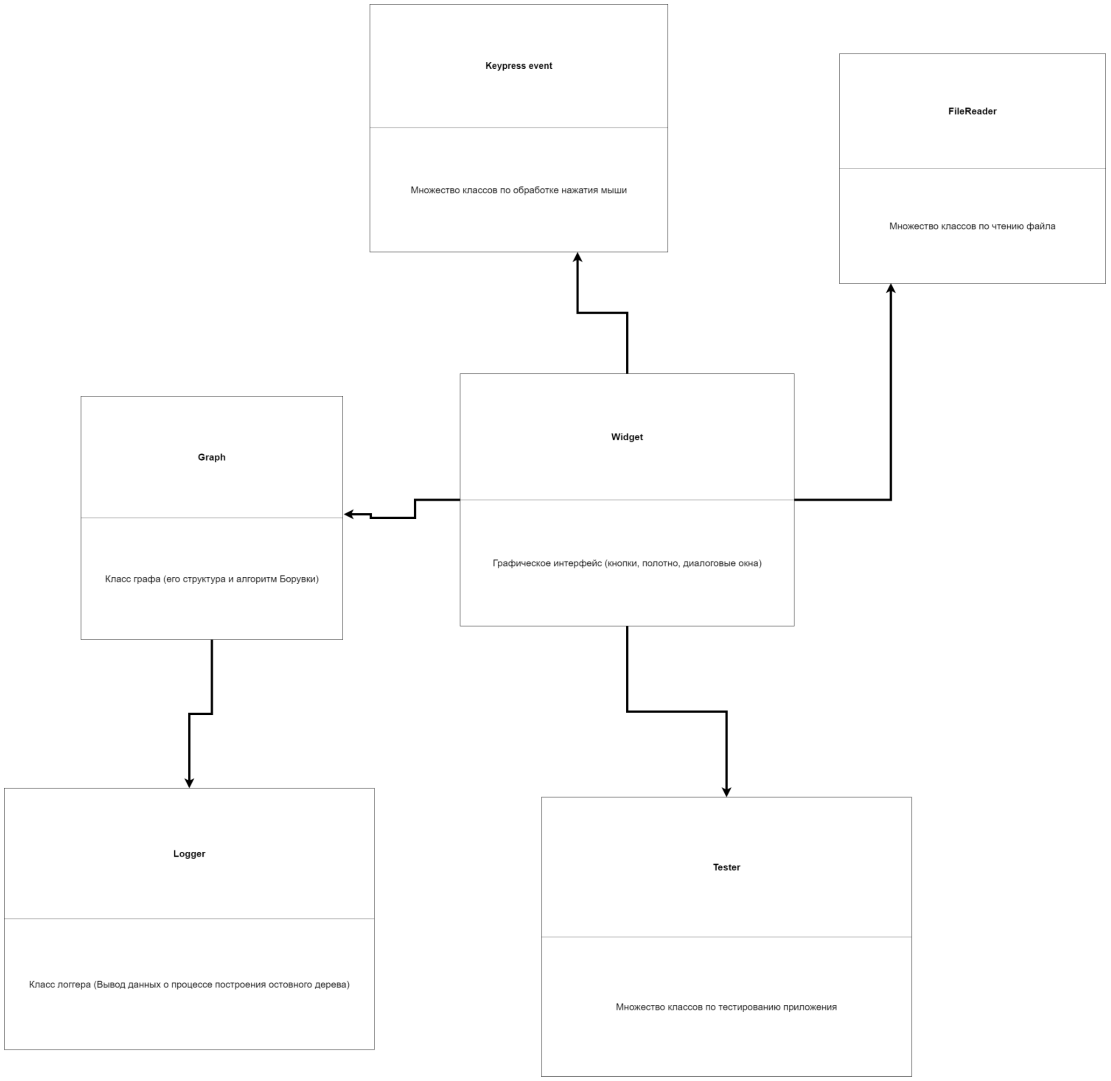
самом файле информация о графе будет написана следующим образом: на первой строчке будет написано количество (*n*) ребер в графе, после чего последует *n* строк содержащих информацию о каждом ребре графа(первые две цифры будут указывать на индексы соединенных вершин, а последняя цифра указывает на вес самого ребра). Также после этого у пользователя будет возможность добавить новые вершины и ребра, с помощью других кнопок. Если же пользователь выберет ввод данных через кнопку мыши, то с помощью других кнопок пользователю будет необходимо добавить все необходимые вершины и ребра с весами.

Элементы графа такие как вершины и ребра будут выглядеть следующим образом:

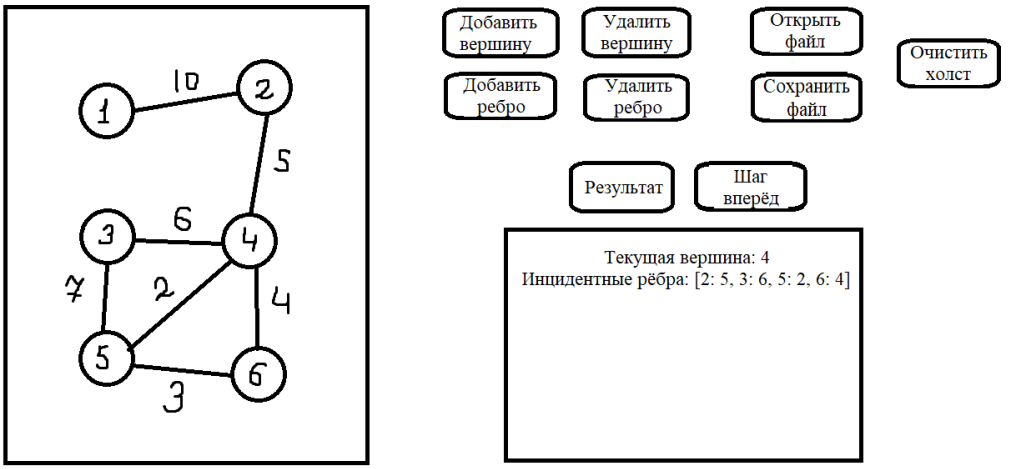
- Вершины выглядят как окружности, внутри которых будут написаны их номера.
- Рёбра представляют собой линии соединяющие вершины, над которыми будет написан их вес.

Процесс алгоритма Борувки будет отображаться через постепенное перекрашивание графа в разные цвета рёбер и вершин, а затем последующее перекрашивание деревьев, полученных на промежуточных результатах. Сначала для каждой вершины будет находится минимальное по весу инцидентное ребро и перекрашивать ребро и вершину в какой-то цвет. То же самое будет происходить для всех остальных вершин. Затем алгоритм будет приращивать и красить минимальное по весу ребро к дереву, полученному в результате предыдущих шагов. Если приращённое ребро соединяет два множества деревьев, то они перекрашиваются в один цвет и становятся единым деревом. Алгоритм работает до тех пор пока не останется одно единственное дерево с одним цветом.

1.1.3. UML диаграмма



1.1.4. Эскиз приложения



1.1.5. Псевдокод алгоритма

```
// G — исходный граф
// w — весовая функция
function boruvkaMST():
  while T.size<n−1
    for k∈ Components        // Components — множество компонент связности в T.
Для
      w(minEdge[k])=∞ //каждой компоненты связности вес минимального ребра = ∞
    findComp(T)        // Разбиваем граф T на компоненты связности обычным dfs-ом.
    for (u,v)∈E
      if u.comp≠v.comp
        if w(minEdge[u.comp])>w(u,v)
          minEdge[u.comp]=(u,v)
        if w(minEdge[v.comp])>w(u,v)
          minEdge[v.comp]=(u,v)
    for k∈ Components
      T.addEdge(minEdge[k])        // Добавляем ребро, если его не было в T
  return T
```

2. План разработки и распределение ролей в бригаде

2.1. План разработки

1) Реализация графического интерфейса без функционала
2) Реализация графа
3) Реализация алгоритма Борувки
4) Добавление графического отображение графа и его удаление (функционал кнопок добавления и удаления)
5) Добавления графического отображения алгоритма Борувки (функционал кнопок результата и шагов)

6) Добавления чтения из файла и возможности сохранить граф (функционал кнопок открытие файла и сохранения)

2.2. Распределение ролей в бригаде

Сологуб Н.А.:

- Проектирование и реализация графического интерфейса пользователя (за исключением вывода графа на экран).
- Реализация вывода пояснений хода алгоритма в отдельную консоль.
- Ручное тестирование реализованных частей интерфейса.

Середенков А.А.

- Разбиение алгоритма Борувки
- Разбиение реализованного алгоритма Борувки на логические части для возможности его последующего пошагового отображения в графическом интерфейсе.
- Реализация возможности сохранения и загрузки созданного графа.
- Тестирование работы алгоритма

Пичугин М.В.:

- Создание базовых классов графа.
- Создание классов для графического отображения графа
- Реализация графической части отображения алгоритма Борувки.
- Тестирование графического отображения алгоритма.