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

ОТЧЕТ

по учебной практике

Тема: Поиск кратчайших путей. Алгоритм Дейкстры

Студент гр. 2383 _____ Сериков М.

Студент гр. 2303 _____ Мышкин Н.В.

Руководитель _____ Фирсов М.А.

Санкт-Петербург

2024

ЗАДАНИЕ

НА УЧЕБНУЮ ПРАКТИКУ

Студент Сериков М. группы 2383

Студент Мышкин Н.В. группы 2303

Тема практики: Алгоритм Дейкстры поиска кратчайших путей в графе.

Задание на практику:

Командная итеративная разработка визуализатора алгоритма на Java с графическим интерфейсом.

Алгоритм: Алгоритм Дейкстры поиска кратчайших путей в графе.

Сроки прохождения практики: 26.06.2024 – 09.07.2024

Дата сдачи отчета: 00.07.2024

Дата защиты отчета: 00.07.2024

Студент _____ Сериков М.

Студент _____ Мышкин Н.В.

Руководитель _____ Фирсов М.А.

АННОТАЦИЯ

Цель практики — создание программы с поддержкой графического интерфейса для нахождения кратчайшего пути с помощью алгоритма Дейкстры на графе с неторицательными весами ребер. Перед выполнением основного задания был составлен план разработки и спецификация программы согласно которым производилась работа.

SUMMARY

The goal of the practice is to create a program with graphical interface support to find the shortest path using Dijkstra's algorithm on a graph with non-negative edge weights. Before completing the main task, a development plan and program specification were drawn up according to which the work was carried out.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Главная цель практической работы — реализация графического представления работы алгоритма Дейкстры поиска кратчайших путей в графе. Для достижения поставленной цели необходимо реализовать рассматриваемый алгоритм, пользовательский интерфейс и визуализировать работу алгоритма, после чего произвести тестирование всех компонент проекта.

1. ТРЕБОВАНИЯ К ПРОГРАММЕ

1.1. Исходные Требования к программе

1.1.1. Требования к вводу исходных данных

На вход программе должен подаваться неотрицательно взвешенный неориентированный граф и исходная вершина. Название вершин состоят из одного символа латинского алфавита. Ввод начальных данных осуществляется двумя возможностями в зависимости от выбора пользователя: непосредственно в рабочей пространстве программы или посредством файла формата txt с данными о графе в следующем формате:

количество вершин

количество ребер

Название вершины координата X, координата Y

Вершина начала ребра, вершина конца ребра, вес ребра

Пример:

2

1

A 50 50

B 100 100

A B 10

1.1.2. Требования к визуализации

Алгоритм в ходе работы сохраняет промежуточные решения в виде списка по которому будет пошаговая визуализация с контролем шагов со стороны пользователя. На каждом шагу вывод одного из следующих этапов работы алгоритма:

Проверка соседей текущей вершины: Для каждого соседа проверяется нахождение нового более оптимального пути. (цветом выделяется проверяемая вершина и ребро до соседней вершины).

Обновление данных соседней вершины: Если найден более оптимальный путь, то обновляется метка стоимости пути. (изменение цвета вершины)

Переход к следующей вершине: Из очереди выбирается новая вершина в качестве текущей с наименьшим весом (отрисовка пути с нуля путем перекрашивания соответствующих ребер).

На каждом шаге алгоритма сохраняется текущее состояние алгоритма и очередь вершин.

1.1.3. Требования к интерфейсу

Эскиз интерфейса представлен на рисунке 1.

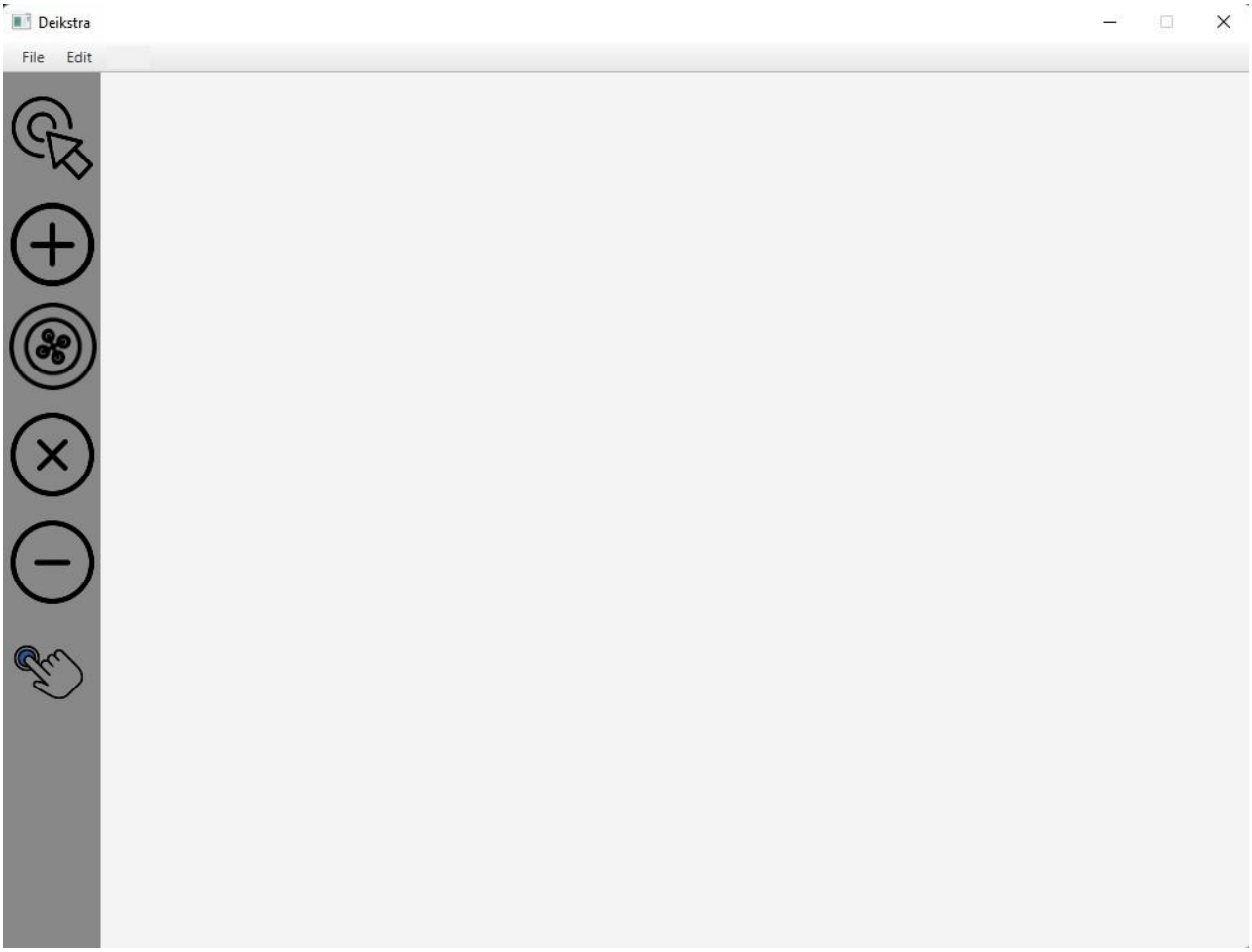

Рисунок 1 — интерфейс программы

В верхней панели программы находится меню с вкладками File, Edit.

Вкладка File: содержит в себе инструменты для загрузки графа из файла, кнопку выхода из программы

Вкладка Edit: содержит в себе инструменты для перехода к окну визуализации

Основная панель инструментов содержит в себе кнопки для построения графа, рассмотрим все кнопки.

Рисунок 2 — кнопка перемещения вершин

Кнопка на рис.2 необходима для перемещения вершин графа, пока активен режим каждую вершину можно перетащить посредством мыши.

Рисунок 3 — кнопка добавления вершин

Кнопка на рис.3 необходима для создания вершин графа, пока активен режим нажатием ЛКМ можно создавать вершины.

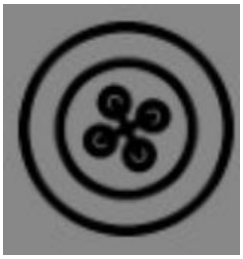

Рисунок 4 — кнопка добавления ребра

Кнопка на рис.4 необходима для добавления ребер графа, пока активен режим выделение двух вершин создает ребро.

Рисунок 5 — кнопка удаления вершин

Кнопка на рис.5 необходима для удаления вершин и ребер графа, пока активен режим выделение вершины или ребра удаляет его.

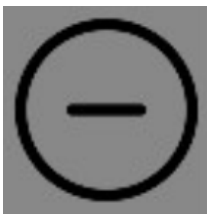

Рисунок 6 — кнопка очистки графа

Кнопка на рис.6 необходима для очистки графа, она очищает рабочую область.

Рисунок 7 — кнопка выбора начальной вершины

Кнопка на рис.7 необходима для выбора начальной вершины для работы алгоритма, она выделяет одну вершину меняя её цвет.

Эскиз окна визуализации работы алгоритма представлен на рисунке 8

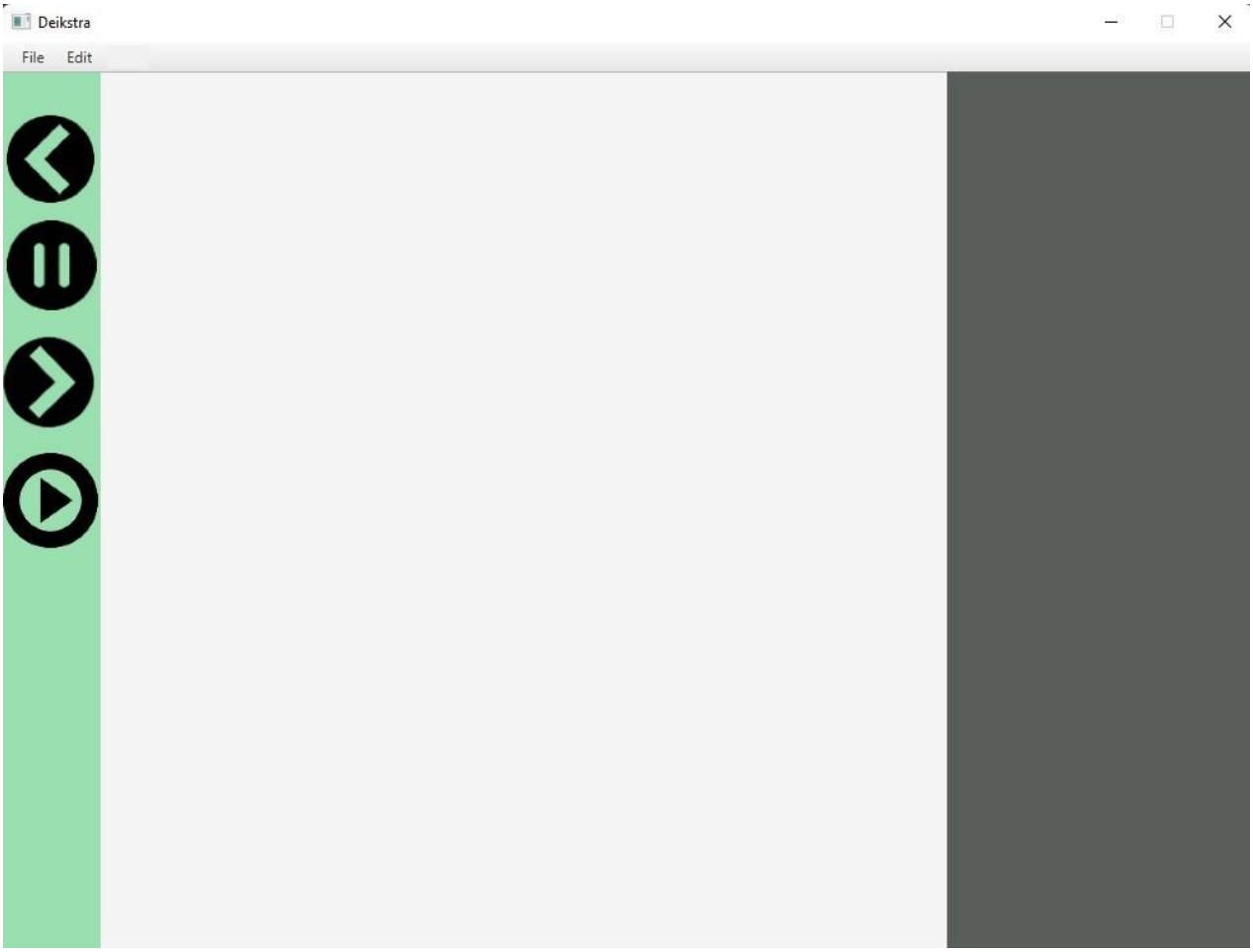

Рисунок 8 — окно визуализации программы

В данном окне представлены инструменты для контроля визуализации работы алгоритма, в случае если выбрана вершина старта алгоритма, то кнопки становятся активными, для начала запуска визуализации алгоритма используется нижняя кнопка, также имеются кнопки паузы и переключения между следующим и предыдущим шагами визуализации. Кнопка начала визуализации запускает автоматическое воспроизведение шагов с небольшой задержкой, при нажатии паузы можно перемещаться между шагами работы алгоритма. Справа от рабочей области находится поле лога работы алгоритма.

1.1.4. Требования к построению графа

Для построения будут использоваться кнопки:

Режим перетаскивания вершин — нажимая ЛКМ и удерживая курсор на вершине возможно её перемещение внутри рабочей области программы (перемещение вершины вслед за курсором)

Режим добавления вершины — нажатие ЛКМ по свободному пространству создась вершину, которой можно будет присвоить название.

Режим добавления ребра — Необходимо выбрать пару вершин, выделение будет демонстрироваться как изменение цвета вершины, после выделения второй вершины будет построено ребро с возможностью ввода его веса, являющимся неотрицательным числом.

Режим удаления объекта — выделения ЛКМ вершины или ребра удаляет его из графа.

Кнопка очистки рабочей области — удаление всего графа.

1.1.5. Требования к работе алгоритма

За основу рассматриваемого алгоритма взят следующий псевдокод:

```
function Dijkstra(Graph, source):
      dist[source] ← 0
      for each vertex v in Graph:
          if v ≠ source:
              dist[v] ← ∞
              prev[v] ← undefined
      Q ← the set of all nodes in Graph
      while Q is not empty:
          u ← vertex in Q with min dist[u]
          remove u from Q
          for each neighbor v of u:
              alt ← dist[u] + length(u, v)
              if alt < dist[v]:
                  dist[v] ← alt
                  prev[v] ← u
      return dist[], prev[]
```

Алгоритм должен уметь работать с неориентированными графами, с ограничением на не более чем 1 ребро между двумя вершинами и без петель, он должен реализовывать алгоритм Дейкстры, поиск кратчайших путей с заданной вершины. Алгоритм должен сохранять промежуточные результаты работы для пошаговой визуализации.

1.1.6. Требования к выводимым пояснениям

В ходе работы алгоритма пояснения должны включать следующие моменты:

1. **Начало алгоритма**:
   - Сообщение о старте алгоритма.
   - Указание начальной вершины, с которой начинается выполнение алгоритма.
2. **Проверка соседей текущей вершины**:
   - Сообщение о проверке соседей текущей вершины.
   - Перечисление всех соседних вершин и текущее расстояние до них.
3. **Обновление данных соседней вершины**:
   - Сообщение о нахождении более оптимального пути к соседней вершине.
   - Обновление метки стоимости пути до соседней вершины.
   - Сообщение о новом значении метки для соседней вершины.
4. **Переход к следующей вершине**:
   - Сообщение о переходе к следующей вершине с наименьшим весом.
   - Указание новой текущей вершины.
5. **Межшаговые действия**:
   - Сохранение текущего состояния алгоритма.
   - Вывод текущей очереди вершин.
   - Информация о промежуточных результатах.
6. **Завершение алгоритма**:
   - Сообщение о завершении работы алгоритма.
   - Вывод итоговых меток для всех вершин (кратчайших расстояний от начальной вершины до всех остальных вершин).
7. **Общие сообщения**:
   - Сообщение об ошибке в случае некорректных входных данных (например, отрицательные веса, несоответствие формата данных).
   - Информационные сообщения о текущем статусе и действиях программы.

Эти пояснения должны отображаться в логовом поле, находящемся справа от рабочей области, и обновляться в реальном времени по мере выполнения шагов алгоритма.

2. ПЛАН РАЗРАБОТКИ И РАСПРЕДЕЛЕНИЕ РОЛЕЙ В БРИГАДЕ

2.1. План разработки

Дата	Этап проекта	Реализованные возможности	Выполнено
27.06.24	Согласование спецификации		
02.06.24	Сдача прототипа	Графический интерфейс программы	
04.07.24	Сдача версии 1	Работа алгоритма без контроля шагов, вывод лога работы алгоритма, возможность построения графа, обработка нажатий кнопок рабочей среды.	
06.07.24	Сдача версии 2	Пошаговый контроль работы алгоритма, вывод пояснений, обработка исключений, улучшенный лог работы алгоритма, полное функционирование всех частей программы.	
08.09.24	Сдача отчёта		
08.09.24	Защита отчёта		

2.2. Распределение ролей в бригаде

Сериков М. - пользовательский интерфейс программы, визуализация алгоритма, связь графической состовляющей программы с основной логикой.

Мышкин Н.В. - реализация алгоритма, структур данных, тестирование

3. ОСОБЕННОСТИ РЕАЛИЗАЦИИ

3.1. Структуры данных

3.2. Основные методы

4. ТЕСТИРОВАНИЕ

4.1. Тестирование графического интерфейса

Тестирование интерфейса на различных операционных системах Windows, Linux, всех следующих элементов:

1. Режим изменения графа
2. Окно визуализации алгоритма
3. Панель инструментов
4. Загрузка графа из файла

При тестировании режима изменения графа проверяется корректная работа всех инструментов: создание вершины, перемещение вершин графа, создание ребра, выбор начальной вершины алгоритма, удаление вершины или ребра, очистка графа. Проверка, что при выборе инструмента кнопка меняет цвет, проверка невозможности одноверменного выбора более чем одного инструмента.

Тестирование окна визуализации алгоритма включает также проверку всех составляющих инструментов: предыдущий, следующий шаг, пауза, старт. Если не выбрана начальная вершина проверяется, что панель инструментов заблокирована. Проверка правильной работоспособности программы при проигрывании анимации посредством взаимодействия со всеми доступными во время проигрывания инструментами, при переходе в режим редактирования визуализация останавливается.

4.2. Тестирование кода алгоритма

При тестировании кода алгоритма будут использоваться Юнит-тесты реализованные при помощи библиотеки JUnit. При тестировании будет проверка работоспособности алгоритма на неориентированных графах, обработаны ситуации когда пути не существует, либо имеются несколько кратчайших путей, тесты охватывают все описанные случаи.

4.3. Тестирование кода графа

При тестировании также используется библиотека JUnit, для проверки правильной работоспособности кода неориентированного графа будут произведены тесты на получение ошибки при передаче значения null, на правильную реакцию при передаче нормальных данных и на получение ошибки при других сценариях.

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

ПРИЛОЖЕНИЕ А
НАЗВАНИЕ ПРИЛОЖЕНИЯ