**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по учебной практике**

Тема: Кратчайшие пути в графе. Алгоритм Дейкстры

|  |  |  |
| --- | --- | --- |
| Студент гр. 1304 |  | Сулименко М. А. |
| Студент гр. 1304 |  | Клепнев Д. А. |
| Студент гр. 1381 |  | Таргонский М. А. |
| Руководитель |  | Шестопалов Р. П. |

Санкт-Петербург

2023

**ЗАДАНИЕ**

**на учебную практику**

|  |  |  |
| --- | --- | --- |
| Студент Сулименко М. А. группы 1304 | | |
| Студент Клепнев Д. А. группы 1304 | | |
| Студент Таргонский М. А. группы 1381  Тема практики: Кратчайшие пути в графе. Алгоритм Дейкстры | | |
| Задание на практику:  Командная итеративная разработка визуализатора алгоритма на Java с графическим интерфейсом.  Алгоритм: Дейкстра. | | |
| Сроки прохождения практики: 30.06.2023 – 13.07.2023 | | |
| Дата сдачи отчета: 00.07.2023 | | |
| Дата защиты отчета: 00.07.2023 | | |
|  | | |
| Студент |  | Сулименко М. А. |
| Студент |  | Клепнев Д. А. |
| Студент |  | Таргонский М. А. |
| Руководитель |  | Шестопалов Р. П. |

**Аннотация**

Задача практики – реализовать графическое приложение, отображающее последовательную работу алгоритма Дейкстры. Перед выполнением были поставлены точные цели и сроки их реализации. После чего выполнялась работа по установленным срокам.

**Summary**

The task of practice is to implement a graphical application that displays the sequential operation of Dijkstra's algorithm. Before implementation, precise goals and deadlines were set. After that, the work was carried out according to the established deadlines.

**введение**

Данная практическая работа состоит в реализации графического отображения работы алгоритма Дейкстры. Результат работы – готовое приложение с графическим интерфейсом, позволяющее пользователю удобно настраивать параметры работы алгоритма и наблюдать за каждым этапом его работы.

Алгоритм Дейкстры – это алгоритм поиска кратчайшего пути между двумя вершинами на произвольном графе. Алгоритм проходит по всем ребрам графа, выбирая наиболее выгодный путь.

**1. требования к программе**

* 1. **Исходные Требования к программе**
     1. **Требования к функциональности**

Программа должна выполнять ряд задач.

В первую очередь, программа должна уметь работать с графами любого типа: ориентированными и неориентированными. Будет лишь установлено ограничение в максимальном количестве ребер между любыми двумя вершинами.

В обоих видах графа будут запрещены «петли» - ребра, начало и конец которых совпадают, т.к. их наличие в графе не повлияет на результат работы алгоритма.

Также в ориентированных графах будет разрешено лишь одно ребро между двумя вершинами. Это также не повлияет на результат алгоритма, т.к., при наличии кратных ребер, алгоритм бы выбирал ребро с минимальной стоимостью, что может сделать пользователь вручную в процессе создания графа.

В неориентированных графах можно будет иметь максимум два ребра между любыми двумя вершинами, при условии, что эти ребра будут направлены в противоположные стороны. Опять же, такое сокращение не повлияет на результат алгоритма по тем же причинам, что и в ориентированных графах.

Во-вторых, она должна уметь реализовывать алгоритм Дейкстры. Алгоритм Дейкстры — это алгоритм поиска кратчайшего пути до всех вершин графа от некой фиксированной вершины S. В самом начале алгоритма считается, что расстояние до всех вершин графа бесконечно большое. Выбирается вершина с кратчайшим путём, которая помечается как проверенная с неким расстоянием. После чего, выбирается непроверенная вершина, смежная с любой из проверенных, которая имеет наименьший вес пути от вершины S. Вес считается по формуле Vрассматриваемая=Vребра\_между\_вершинами+Vтекущая. Если в процессе поиска обнаруживается более короткий путь, чем ранее найденный, то информация о найденном пути обновляется.

Приложение позволит наглядно продемонстрировать работу алгоритма Дейкстры. Для этого в процессе работы, алгоритм будет разделен на ряд шагов, которые будут обозначать какие-либо действия в алгоритме, поддающиеся несложной анимации. Переключаться между шагами можно будет с помощью специальных кнопок управления на интерфейсе пользователя. Или же можно будет включить автоматическое воспроизведение шагов, когда шаги будут показываться друг за другом с определенной пользователем в настройках паузой. Также можно приостановить алгоритм с помощью кнопки паузы или отключить показ совсем на кнопку стоп. Пошаговое отображение позволит пользователю остановится в любой момент или же рассматривать алгоритм в обратном порядке.

В-третьих, приложение должно иметь интуитивно понятный интерфейс, позволяющий пользователю удобно пользоваться всеми возможность программы, позволяющий графически отображать граф и все его изменения, позволяющий реализовать некоторое количество инструментов по работе с графом, а именно: кнопки по постройке графа (режим перетаскивания вершин, режим добавления вершины, режим добавления ребра, режим очистки графа), кнопка настройки начальной и конечной точки, для работы алгоритма Дейкстры, кнопки запуска и остановки вычислений алгоритма, кнопки пошагового вывода следующего и предыдущего шага алгоритма Дейкстры, а также позволяющий наглядно видеть все шаги программы в соответствующей текстовой области. В отдельной области экрана должна выводится информация по происходящему на экране. Если пользователь взял инструмент, туда выведется информация о том, для чего предназначен этот инструмент. Во время работы алгоритма, каждый шаг будет в текстовом виде выведен в эту область.

Важным моментом является возможность сохранять граф. Для этого в отдельном месте интерфейса будут располагаться кнопки для обычного сохранения и так называемого «сохранения как», позволяющего настроить путь и название сохраняемого файла. При первом сохранении обе кнопки будут работать, как «сохранить как».

Также граф можно будет загружать из файла. Для этого также будет создана кнопка, при нажатии на которую, откроется интерфейс, сходный с проводником, для выбора загружаемого файла.

Помимо этого, должна быть возможность выводить граф на экран в виде ряда вершин (кругов), соединенных ребрами (отрезками). Нужно уметь выводить как ориентированный граф, так и неориентированный с соответствующими изменениями в виде ребер (отрезки со стрелкой или без). У каждого элемента графа будет своя приписка, обозначающая какую-либо информацию об этом элементе. В случае вершины – это название вершины, в случае ребра – его вес. Кроме того, нужно иметь холст, на котором и будет рисоваться граф. При этом мы должны уметь увеличивать или уменьшать масштаб отображения холста.

В случае загрузки из файла, граф должен добавится на экран с использованием какого-либо правила расстановки вершин.

Также пользователь должен иметь возможность взаимодействовать с графом. Это включает в себя перемещение вершин графа. Нажимая и удерживая ЛКМ на вершине, можно будет переместить мышь, и вершина будет перемещаться с ней; создание ребер и вершин; стирание всего графа; выделение вершин для поиска кратчайшего пути.

Создание вершин подразумевает под собой нажатие по свободному участку экрана, где создастся вершина, под которой можно будет ввести ее название.

Создание ребер будет выполнятся, как выбор пары вершин: начала и конца (в случае неориентированного графа это будет задавать направление). Выбор каждой вершины отображается в виде окантовки круга в определенный цвет. После чего будет нарисовано ребро, посередине которого нужно будет ввести его вес.

Стирание графа – удаление из экрана всех построенных элементов графа.

Под выделением вершин для алгоритма поиска имеется в виду выбор двух вершин (с графическим подтверждением их выбора, как это было при создании ребра, но уже с другим цветом). Можно будет выбрать максимум 2 вершины. Нажатие на уже выбранную вершину будет снимать выбор.

Все элементы графа должны поддаваться стилистике вне классов самих элементов. Все стили должны указываться в специальном css файле.

* + 1. **Требования к интерфейсу**

Интерфейс должен быть интуитивно понятен пользователю, т.е. назначение элементов интерфейса должно быть понятно без дополнительных надписей или с малым их количеством, и удобен в использовании.

Он будет состоять из двух режимов: режима настройки и режима воспроизведения. Режим настройки должен включать в себя набор инструментов, позволяющий настраивать граф перед запуском алгоритма (это включает в себя инструменты создания, перемещения и очистки графа, а также инструмент выбора начальной и конечной вершины последующего поиска кратчайшего пути). В то же время, режим воспроизведения будет содержать следующие инструменты: переход на шаг вперед/назад, запуск автоматического проигрывания шагов, пауза и стоп.

В самом верху программы будет меню с различными вкладками: File, Edit и Help. Вкладка File содержит в себе инструменты для создания нового графа с нуля, для загрузки графа из файла и для сохранения графа в файл, а также кнопку выхода из программы. Вкладка Edit состоит из продублированных инструментов из обоих режимов работы, из функции переключения этих самым режимов и из еще двух кнопок, одна из которых будет открывать окно с логами (куда будет выводится информация о любом взятом пользователем инструменте и информация по каждому проигранному шагу алгоритма), а другая – настройки, в которых можно настроить некоторые параметры отображения графа и стандартный путь сохранения всех графов (он будет автоматически открываться при первом сохранении). Остается вкладка Help, в которой будет кнопка, открывающая репозиторий с исходным кодом, и кнопка, открывающая окно с информацией о самом приложении.

Посмотрим на сам интерфейс:

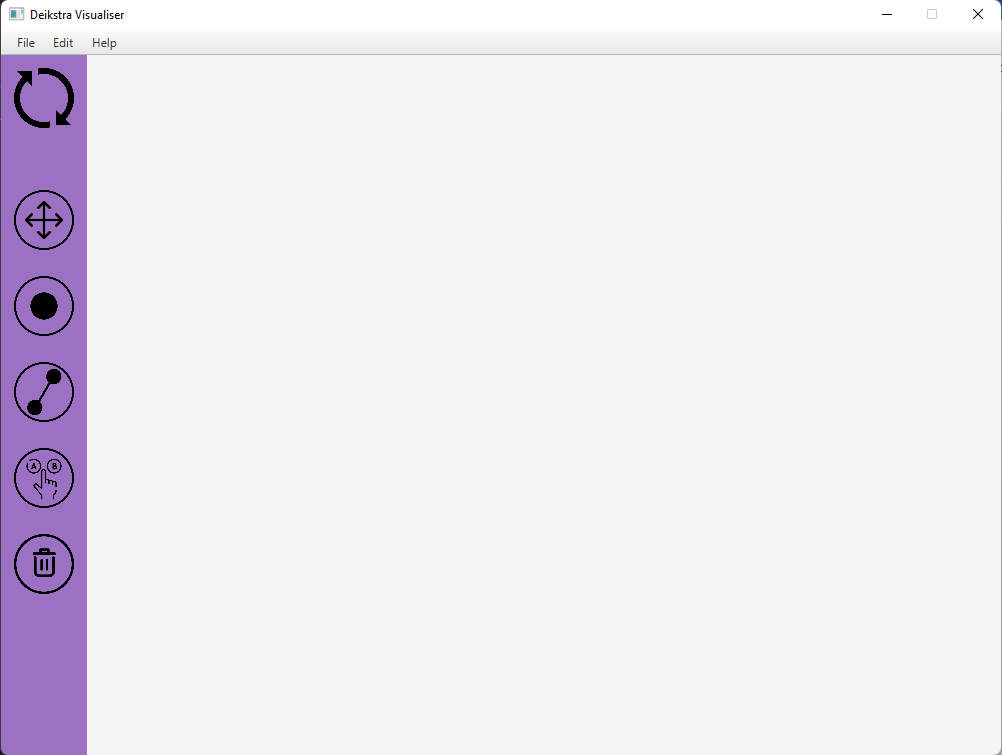


Рисунок 1 – Начальный экран программы.

Запустившись, приложение выводит то, что показано на рис. 1. Это окно можно менять в размерах. Пока что, ни один из инструментов не доступен для пользователя, т.к. для начала ему нужен граф. Он должен перейти во вкладку File в верхнем меню и выбрать подходящий для него метод создания или загрузки графа (см. рис. 2).

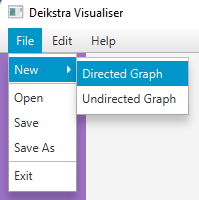


Рисунок 2 – Вкладка File из меню и ее содержимое.

Как видно из рисунка, существует два способа создать граф (создать ориентированный или неориентированный граф) и два способа его сохранить. При первом сохранении кнопка «Save» будет вести себя, как «Save As», т.е. будет открывать новое окно с проводником, спрашивая пользователя, куда он хочет сохранить свой граф. В той же вкладке меня можно выйти из приложения (стандартный способ для всех приложений также работает).

Создав или загрузив граф, пользователю открываются для использования инструменты его настройки. Рассмотрим каждый в отдельности:

Кнопка на рис. 3 отвечает за перемещение вершин графа. Пока она активна каждую вершину можно перетащить вышкой по холсту.



Рисунок 3 – Кнопка перемещения вершин графа

Следующая кнопка на рис. 4 отвечает за создание новых вершин. Активировав ее, можно будет создавать вершины, просто ткнув в любое свободное от элементов графа место на холсте.



Рисунок 4 – Кнопка создания новых вершин

Далее идет кнопка создания новых ребер. Она расположена на рис. 5. С ее помощью, можно будет выделить 2 вершины, между которыми образуется новое ребро.



Рисунок 5 – Кнопка создания новых ребер

Важной кнопкой является кнопка выбора двух вершин для последующей работы алгоритма. Это кнопка на рис. 6.



Рисунок 6 – Кнопка выбора вершин для алгоритма

И остается кнопка, которая полностью стирает весь граф с экрана. Она на рис. 7.



Рисунок 7 – Кнопка стирания всего графа

На самом деле осталась еще одна кнопка в самом верхнем левом углу (см. рис. 8). Но она занимается не настройкой графа, а переключением режимов. Такая кнопка есть в обоих режимах.



Рисунок 8 – Кнопка переключения режимов

В этом режиме воспроизведения доступны инструменты управления воспроизведением, такие как: переход на шаг назад или вперед, старт воспроизведения, пауза и стоп.

Если в предыдущем режиме не были выбраны вершины для алгоритма, то все инструменты этого режима будут заблокированы. В ином же случае, после переключения будет доступна для нажатия кнопка запуска, как бы показывая, что можно начать показ шагов. При нажатии на эту кнопку, запустится автоматическое воспроизведение шагов с небольшой, но достаточной для принятия решения приостановки показа, паузой и разблокируются все остальные кнопки. Приостановку показа можно сделать, нажав на кнопку паузы. Если автоматическое воспроизведение приостановлено, то шагами можно управлять кнопками налево и направо, перемещаясь по одному шагу назад и вперед соответственно. Или же можно нажать на эти кнопки самостоятельно, и тогда программа сама отключит автоматический показ шагов. Если дальнейший просмотр алгоритма не нужен или хочется быстро вернутся в его начало, то можно полностью остановить показ на кнопку стоп.

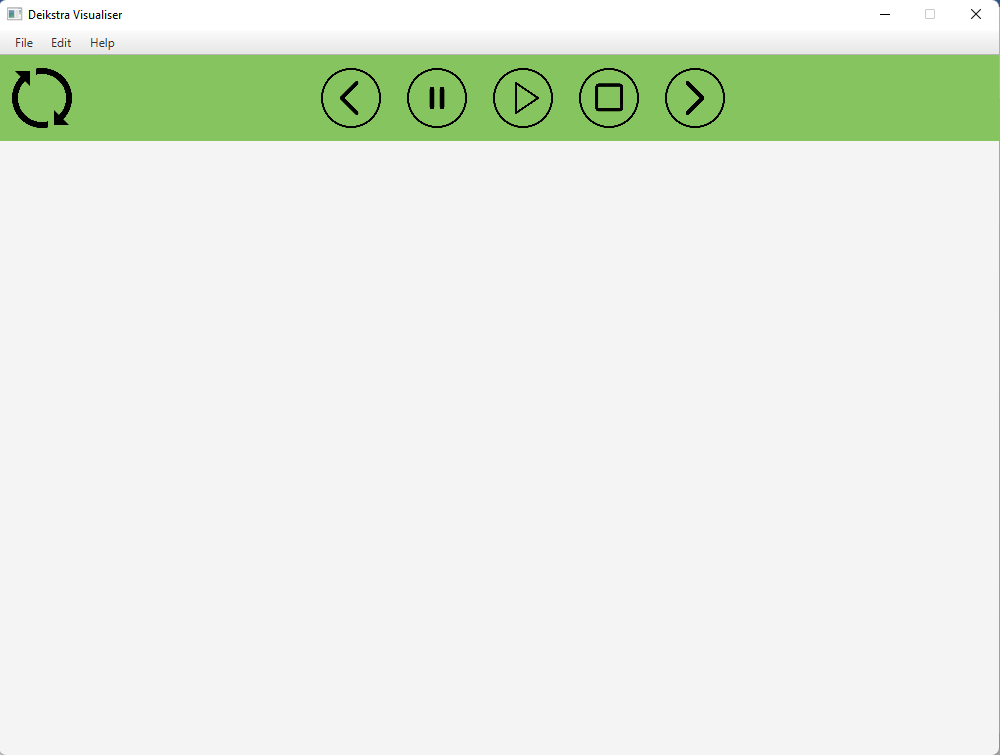


Рисунок 9 – Режим воспроизведения

Теперь вернемся ко вкладкам панели меню. Во вкладке Edit (см. рис. 10) располагаются кнопка переключения режимов, два списка с инструментами режимов (доступность этих инструментов соответствует описанному выше), кнопка, открывающая логи (см. рис. 11), и кнопка, открывающая настройки приложения (см. рис. 12), в которых можно изменить некоторые параметры отображения графа и стандартный путь сохранения.

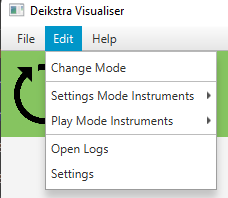


Рисунок 10 – Вкладка Edit

И остается последняя вкладка Help (см. рис. 13), в которой располагается кнопка, открывающая окно с информацией о программе и кнопка, отсылающая на репозиторий проекта.

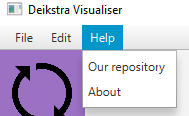


Рисунок 12 – Вкладка Help

* + 1. **Требования к визуализации алгоритма**

Работа алгоритма должна разделятся на логические части, в каждой из которых будет сохранятся состояние алгоритма в отдельный список. Далее, используя этот список, визуализатор будет пошагово (управление переключением шагов остается за пользователем) выводить работу алгоритма.

Будут выделены следующие выводимые этапы алгоритма:

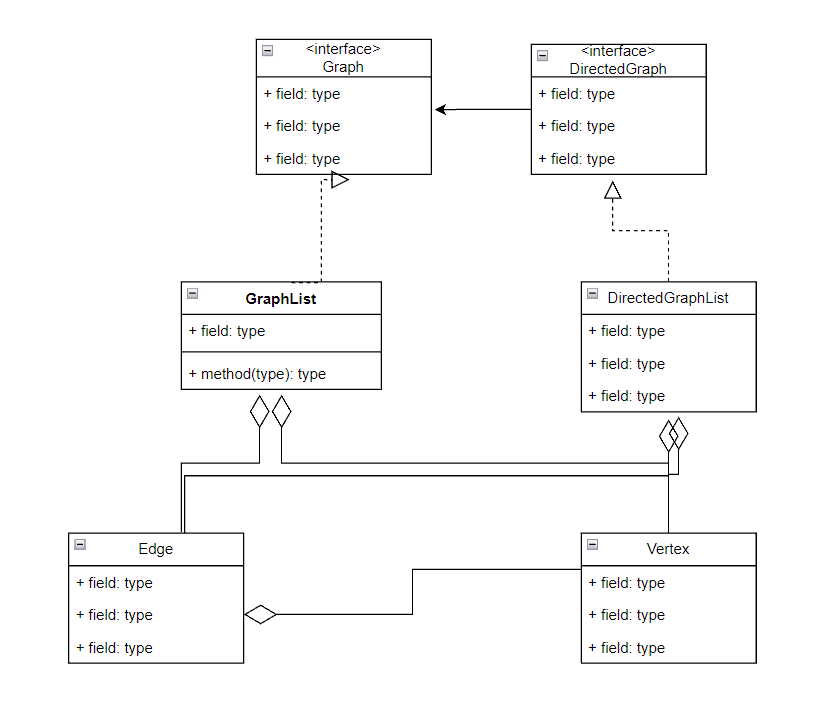
* Проверка ребёнка у узла:
  + Для каждого узла, исходящего из текущего, проверяется нахождение нового более дешёвого пути.
  + Визуализация: цветом выделяется окантовка проверяемой вершины и ребро до него.
* Обновление информации у ребёнка:
  + В случае нахождения более дешёвого пути до ребра в ней обновляется информация, такая как наименьший вес пути до неё и узел, из которого можно до неё добраться с наименьшим весом.
  + Визуализация: полная перекраска вершины определённым цветом.
* Переход к узлу:
  + Из вершин из очереди на обработку, выбирается новая текущая вершина с наименьшим весом
  + Визуализация: полная отрисовка пути с нуля.

Шагом в работе алгоритма будет считаться выполнение любого из перечисленного этапа.

При запуске алгоритма на каждом шаге будет сохраняться информация о текущем состоянии графа и очереди вершин. После завершения работы алгоритма класс пошагового отображения сможет получить информацию о пройденных шагах, чтобы отобразить их в графическом интерфейсе с возможностью переключения между шагами, причём как вперёд, так и назад. Все пути будут анимированы (ребра будут перекрашиваться постепенно друг за другом с определенной задержкой).

* + 1. **Требования к архитектуре приложения**

UML-схема базовых классов для хранения и работы с графами различного типа:



* + 1. **Требования к тестированию**

Тестирование приложения будет разбито на 2 вида: автоматическое тестирование и ручное. Автоматическое тестирование (далее Unit-тесты) будет реализовано при помощи библиотеки JUnit и будет применятся к методам и классам, в которых подобное тестирование будет возможно. Это все классы, не связанные с реализацией графического интерфейса. В остальных случаях будет применено ручное тестирование.

Unit-тесты планируется написать до непосредственной реализации методов для последующего ускорения разработки. Ручное тестирование будет производится после создания элементов интерфейса.

**2. План разработки и распределение ролей в бригаде**

**2.1. План разработки**

Предполагается следующий план разработки:

* Сдача вводного задания: до 05.07.2023
* Разработка плана и спец-ии: до 05.07.2023
* Разработка прототипа: до 05.07.2023
* Разработка 1-ой версии алгоритма: до 07.07.2023
* Реализация вывода графа на экран: до 07.07.2023
* Реализация системы вывода сообщений: до 10.07.2023
* Реализация вывода шагов алгоритма на экран: до 10.07.2023
* Реализация первой версии интерфейса: до 10.07.2023
* Реализация сохранения и загрузки графа: до 10.07.2023
* Реализация логики работы окна настроек: до 12.07.2023
* Исправление возникших проблем: до 13.07.2023

**2.2. Распределение ролей в бригаде**

Сулименко М. А.:

* Проектирование и реализация графического интерфейса пользователя

Таргонский М. А.

* Реализация алгоритма Дейкстры

Клепнев Д. А.:

* Реализация хранения ориентированного и неориентированного типов графа

**3. Особенности реализации**

**4. тестирование**

**заключение**

**список использованных источников**