МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

# СОЗДАНИЕ ОБУЧАЮЩЕГО ИНТЕРАКТИВНОГО ПРИЛОЖЕНИЯ ДЛЯ ИЗУЧЕНИЯ НЕКОТОРЫХ АЛГОРИТМОВ ТЕОРИИ ГРАФОВ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 5 курса 551 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Антоняна Гегама Арменовича

Научный руководитель
старший преподаватель _____ М. И. Сафрончик

Заведующий кафедрой
к. ф.-м. н. _____ С. В. Миронов

Саратов 2022

# ВВЕДЕНИЕ

Графы используются во многих областях науки и техники, теория графов является важным разделом математики и изучается на младших курсах естественно-научных направлений. В настоящее время можно найти главы чистой математики, например теория математических отношений, в которых теория графов служит естественным аппаратом; с другой стороны графы находят применении и в разнообразных практических вопросах. Самые распространенные из них – это решение транспортных задач, задач о потоках в сетях и масса других задач. Теория графов теперь применяется и в экономике, в биологии и даже в психологии.

При изучении алгоритмов теории графов крайне полезным является пошаговая демонстрация шагов их выполнения. Приложение, позволяющее отрисовывать и редактировать граф, а также добавлять различные алгоритмы с демонстрацией их пошагового выполнения является весьма полезным инструментом в изучении теории графов.

Интерактивное приложение разрабатывается для помощи студентам младших курсов естественно-научных специальностей в изучении алгоритмов теории графов.

Для создания приложения использована бесплатная среда разработки Visual Studio 2022 Community, язык программирования C# с рабочей средой .Net Framework 4.7.2. Для реализации графического интерфейса использован Windows Forms.

В последние несколько лет теория графов стала важнейшим математическим инструментом, широко используемым во многих областях науки, от исследования операций и химии до генетики и лингвистики и от информатики и географии до социологии и архитектуры. В то же самое время она выросла в самостоятельную математическую дисциплину.

**Цель бакалаврской работы** – создать приложение с удобным и интуитивно понятным графическим интерфейсом для создания графов и изучения различных алгоритмов. Приложение должно иметь следующие функциональные возможности.

— Создавать различные виды графов (орграф/граф, взвешенный/невзвешенный).

— В графическом редакторе с помощью инструментов создавать вершины,

перемещать, соединять, удалять вершины и ребра. Для более удобной работы должна быть возможность отменить и вернуть ходы.

- Сохранять созданный граф в файл и открыть его из файла.
- Выбирая нужные входные параметры выполнять различные алгоритмы с их возможной реализаций.
- В исходном коде должна быть возможность добавлять новые алгоритмы.

**Практическая значимость бакалаврской работы** – это помощь студентам в изучении различных алгоритмов теории графа, моделирование ходов выполнение алгоритма.

**Структура и объем работы.** Бакалаврская работа состоит из введения, двух разделов, заключения, списка использованных источников и одна приложения. Общий объем работы – 44 страниц, из них 41 страниц – основное содержание, включая 15 рисунков, цифровой носитель в качестве приложения, список использованных источников информации – 20 наименований.

**Краткое содержание работы**

**Первый раздел "Графы, описание задачи"** посвящен теоретическим основам теории графов и описание задачи.

Подраздел 1.1 содержит теоретическую информацию о теории графов, графическое представление графа, структура данных для представление графов.

Вершины изображаются кружочками, а ребра – отрезками линий. Внутри кружочков записаны метки вершин. Ребра ориентированного графа снабжаются стрелками, указывающими направление движения по ребру.

Информацию о графах или орграфах можно хранить двумя способами: в виде матрицы смежности или в виде списка смежности. Применяемые методы хранения информации устраняют различие в обработке графов и орграфов.

Матрица смежности $AdjMat$ (adjoining matrix) графа $G = (V, E)$ с числом вершин $|V| = N$ записывается в виде двумерного квадратного массива размером $NN$. В каждой ячейке $[i, j]$ этого массива записано значение $1$ – если из вершины $V_i$ в вершину $V_j$ ведет ребро, $0$ – в противном случаи. Математически это можно посмотреть в формуле 1.

$$AdjMat[i,j] = \begin{cases} 1, \text{если } v_i v_j \in E \\ 0, \text{если } v_i v_j \notin E \end{cases} \quad i, j \in [0, N) \text{ (нумерация с нуля)} \qquad (1)$$

Список смежности – один из способов представления графа в виде коллекции списков вершин. Каждой вершине графа соответствует список, состоящий из "соседей"этой вершины.

Список смежности $AdjList$ графа $G = (V, E)$ с числом вершин $|V| = N$ записывается в виде одномерного массива длины $N$, каждый элемент которого представляет собой ссылку на список. Такой список приписан каждой вершине графа, и он содержит по одному элементу на каждую вершину графа, соседнюю с данной.

Если $G$ является ориентированным графом, то сумма длин всех списков смежности равна $|E|$, поскольку ребру $(u, v)$ однозначно соответствует элемент $v$ в списке $Adj[u]$. Если же $G$ представляет собой неориентированный граф, то сумма длин всех списков смежности равна $2|E|$, поскольку ребро $(u, v)$, будучи неориентированным, появляется как в списке смежности $u$, так и в списке $v$. Как

для орграфов, так и для графов представление в виде списков требует объема памяти, равного $\Theta(V + E)$.

Элементы списка смежности для взвешенного графа содержат дополнительное поле, предназначенное для хранения веса ребра.

Подраздел 1.2 содержит информацию об описание задачи, аналогичные приложения, и преимущество разработанного приложения.

**Второй раздел "Программная реализация"** посвящен программной реализации. Описывает используемые средства для разработки, структуру приложения, используемые паттерны, структуру данных для представление графа, реализованные алгоритмы, графическая реализация и т.д. Для реализации приложения используется язык программирование C# с рабочей средой .NET Framework 4.7.2. Графическая часть реализована на Windows Forms.

Подраздел 2.1 содержит информацию о паттерне проектирования Model-View-Controller. MVC – схема разделения данных приложения и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

Подраздел 2.2 содержит информацию об паттерне Command и его реализации. Паттерн Command позволяет инкапсулировать запрос на выполнение определенного действия в виде отдельного объекта. Этот объект запроса на действие и называется командой. При этом объекты, инициирующие запросы на выполнение действия, отделяются от объектов, которые выполняют это действие. Команды могут использовать параметры, которые передают ассоциированную с командой информацию. Кроме того, команды могут ставиться в очередь и также могут быть отменены.

Для реализации паттерна Command в приложении реализованы следующие сущности.

- ICommand. Интерфейс команды.
- ACommandArgs. Класс для хранения аргументов команд.
- CommandRepository. Класс одиночка для хранилище команд.
- CommadManager. Класс для непосредственной работы с командами.
- CommandDispatcher. Класс посредник между пользовательского интерфейса и ядром приложения.

Для обеспечения универсальности работы с командами в приложении объяв-

лен интерфейс ICommand который имеет 2 метода, Execute и Clone. Первый выполняет алгоритм, а второй клонирует себя, принимая другие аргументы. Так как некоторые команды могут быть отменены пользователем, этот интерфейс наследуют два интерфейса, IStoredCommand и INonStoredCommand. INonStoredCommand просто наследует интерфейс без добавление нового функционала, для более выразительности исходного кода. IStoredCommand добавляет метод Undo, что дает классу, реализующий данный интерфейс возможность отменить сам себя.

Подраздел 2.3 содержит информацию о структуре данных графа и его визуальное представление. ля выполнения различных алгоритмов вершина графа может иметь разные параметры. Для этого создан абстрактный класс AVertex, наследники которого при необходимости могут добавлять новые параметры. AVertex реализует интерфейсы IComparable и IEquatable и абстрактно объявляет методы CompareTo и Equals соответственно. Для хранения пар вершина и вес используется параметризованный класс VertexWeightPair<TVertex>, который реализует интерфейс IEquatable. TVertex может быть только производным класса AVertex. Для хранения графа используются списки смежности. Класс параметризован, Graph<TVertex>, где TVertex может быть только производным класса AVertex. Хранилище реализовано с помощью коллекции SortedDictionary, который представляет собой коллекцию пар "ключ-значение упорядоченных по ключу, где ключ TVertex а значение VertexWeightPair.

Для визуализации элементов графа использованы модели элементов графа. Базовый для всех моделей это абстрактный класс GraphModel. Так как все элементы должны иметь цвет и толщину, строковое представление, то эти свойства уже определены и их можно менять. Также каждый элемент имеет ключ, для однозначного определения. Единственный абстрактный метод – это извлечение ключа элемента по координатам полученным от пользователя. Этот метод обязан определить каждый производный модель. AVertexModel – это тоже абстрактный класс, производный от GraphModel, специализированный именно для вершин графа. В классе добавлены новые свойства VertexStr, как строковая визуализация метки вершины и Pos – координаты вершины в двумерном пространстве. Также добавлен один виртуальный метод UpdatePos, который принимает как аргумент новые координаты вершины. VertexDrawModel – это уже конкретная модель для приложения, производная от AVertexModel. В классе вершина

определена как окружность со строковой меткой в середине с определенным радиусом. Определены абстрактные и виртуальные методы базовых классов PosKey и UpdatePos. PosKey использует уравнение окружности для получения ключа вершины, используя как аргументы координаты и радиус. UpdatePos создает матрицу перемещения и перемещает координаты точек окружности. AEdgeModel – это абстрактный класс для ребер, производный от GraphModel. Класс определяет основные свойства модели ребра, такие как модель истока и стока, вес, позиция веса и т.д. Объявлен абстрактный метод RefreshPos, для изменения позиции ребра. Для отображения конкретных моделей ребер используются классы OrientEdgeModel и NonOrientEdgeModel для визуализации ориентированных и неориентированных ребер соответственно. Неориентированное ребро – это линия соединяющая две точки, а ориентированный – фигура из трех отрезков, для отображения используются четыре точки. В этих классах определены абстрактные методы базовых классов PosKey и RefreshPos.

Весь математический необходимый функционал для работы с двумерной графикой реализован в файле Geometry.cs. Так как программа реализована с использованием паттерна MVC, сам класс Graph не носит никакой информации для графического представления. Для этого предназначен класс ModelField реализующий интерфейс IModelField. В классе хранятся экземпляры абстрактного класса GraphModel, наследники которого определяют как будут изображены вершины и ребра. В экземпляре класса можно добавлять, удалять вершины и ребра как и в графе. Также можно перемещать вершины, указать цвет и метки. При выполнении алгоритма именно с помощью этого интерфейса передаются параметры визуализации. Этот экземпляр класса принимает позицию мышки при нажатии, и с помощью метода GetPosKey определяет, на какой элемент нажал пользователь. Каждая модель графа принимая координаты, может определить нажали на него или нет.

Все команды для изменения состояния графа работают напрямую с интерфейсом IModelField. Часть интерфейса, который работает с классом Graph приведен в листинге **??**. Интерфейс объявляет методы для добавления и удаления моделей вершины и ребер, а также массив этих элементов. Имеется боле универсальный метод для добавления всех видов моделей. С помощью интерфейса также можно узнать о параметрах графа. Каждый метод интерфейса, который может изменить состояние IModelField, имеет параметр raise (по умол-

чанию true). Если параметр имеет значение истина, то метод поднимает события FieldUpdate. Основным наблюдателем является графический интерфейс, который соответственно с изменениями в IModelField меняет свое состояние. Конкретный класс, реализующий IModelField, ModelField определяет эти методы. При каждом изменении при необходимости вызывается событие
FieldUpdate, за которой следит UserIntarface в соответствии с канонам паттерна MVC. Для передачи изменения создан перечисление FieldEvents. Для отображение графа используется класс Pointer, который принимает экземпляр класса GraphModel. Класс сильно связана с Windows Forms. Основной метод класса – это Draw. В зависимости от модели, в методе Draw вызываются следующие приватные методы.

- DrawVertex – рисует вершину графа.
- DrawNonOrientEdge – рисует неориентированное ребро.
- DrawOrientedEdge – рисует ориентированное ребро.
- DrawWeight – рисует вес ребра.

Подраздел 2.4 содержит информацию о командах. В приложении реализованы конкретные команды. Команды, которые мож- но отменить и вернуть, реализуют интерфейс IStoredCommand. Команды, ко- торые работают с IModelField должны быть производными от абстрактного класса AFieldCommand. Данный класс просто хранит ссылку на экземпляр класса, реализующий интерфейс IModelField. При добавлении модели графа (GraphModel) используется команда AddModelCommand и аргумент для команды AddModelCommandArgs. Класс AddModelCommand является производным от AFieldCommand и реализует интерфейс IStoredCommand. Команада для выполнении использует аналогичные методы интерфейса IFieldModel. Используя аргумент команды для выполнения добавляет модель, для отмены удаляет. Другие команды, которые должны иметь возможность быть отмененными, и работают с IModelField аналогичны.

Подраздел 2.5 содержит информацию об реализованных алгоритмов. В разделе боле подробно описаны алгоритмы обхода в глубину, Эдмондса-Карпа и Крускала.

Алгоритмы реализуют интерфейс INonStoredCommand (из паттерна Command). В приложения реализованы следующие алгоритмы с визуализацией.

- Обход в глубину – рекурсивная функция, которая обходит все вершины графа только один раз.

— Обход в ширину – как и обход в глубину, обходит все вершины графа только один раз. Использует очередь.
— Связные компоненты – находит связные компоненты графа и сильно связанные компоненты орграфа.
— Алгоритм Крускала – определяет минимальное остовное дерево (каркас).
— Алгоритм Дейкстры – находит кратчайшие пути от всех вершин до выбранной вершины
— N-периферия – определяет N-периферию выбранной вершины. Используется алгоритм Беллмана-Форда.
— Алгоритм Эдмондса-Карпа – находит максимальный поток в сети.

Для добавления новых алгоритмов надо создавать команду реализующею интерфейс INonStoredCommand с определенным именем и класс для аргументов, производной от ACommandArgs. Далее команду надо добавить в CommandRepository. В пользовательском интерфейсе достаточно назначить кнопку, которой передает имя команды и аргументы в определенном string формате.

Все алгоритмы выполняются в отдельном потоке.

Подраздел 2.6 описывает графическую реализацию приложения. Графическая часть приложения реализована на Windows Forms. Окно состоит из трех частей:

— панель управления;
— раздел для отображение графа (слева);
— раздел для отображения списка смежности.

В панели управлении слева направо расположены следующие кнопки.

1. Открыть – позволяет открыть и отображать граф из файла, где данные хранятся в определенном формате. При нажатии открывается диалоговое окно.
2. Сохранить – сохраняет созданный граф в файл. При нажатии открывается диалоговое окно.
3. Отменить – отменяет последний ход пользователя.
4. Вернуть – возвращает отмененный ход.
5. Обновить – возвращает отображение графа в исходное состояние.
6. Создать граф – создает граф используя комбинации граф/орграф и взвешенный/невзвешенный.
7. Удалить граф – удаляет граф.

8. Добавить вершины – после нажатия этой кнопки, при нажатии в разделе отображение графа, создается вершина с автонумерацией.
9. Удалить выбранные элементы – при нажатии удаляются все выбранные элементы.
10. Перемещение – после нажатия можно выбрать вершину графа и перемещать, при этом ребра будут подстраиваться автоматически.
11. Соединить вершины – при выборе двух вершин и нажатии этой кнопки создается ребро.
12. Обходы – алгоритмы обхода в глубину и в ширину с визуализацией.
13. Алгоритмы – список алгоритмов.

Используя весь этот функционал можно легко создать графическое представление графа, а программа на его основе создает списки смежности для выполнения различных алгоритмов.

# ЗАКЛЮЧЕНИЕ

В ходе дипломной работы было создано приложение, позволяющее с помощью визуализации обучать студентов теории графов. Функционал приложения дает возможность создавать, редактировать различные виды графов, выполнять ряд алгоритмов, в том числе позволяет легко добавлять новые алгоритмы. Для реализации приложения использован паттерн проектирования MVC, паттерн Command и т.д. При разработке приложения графическая и логическая часть были максимально отделены, что дает возможность легко изменять графическую часть, например из Windows Forms перейти на WPF. В программе широко принимается линейная алгебра, с помощью которой и рисуются, перемещаются и определяются элементы графа. Хотя программа является обучающей, но его логическая часть построена на оптимальной структуре данных для представления графа, и в случае добавления нужной команды с алгоритмом, можно выполнять нужные вычисления.

**Основные источник информации:**


1. Ойстин, О. Х. Графы и их применение. Перевод с английского Л. И. Головиной / О. Х. Ойстин.— Москва: Мир, 1965.
2. Р. Дж. Уилсон,. Введение в ТЕОРИЮ ГРАФОВ / Р. Дж. Уилсон.— Москва: Диалектика, 2019.
3. Макконнелл, Д. Основы современных алгоритмов. 2-е доп издание. Перевод с англ. С. К. Ландо. Дополнение М. В.Ульянова / Д. Макконнелл.— Москва: Техносфера, 2004.
4. Райгородский, А. М. Модели случайных графов / А. М. Райгородский.— ЛитРес, 2017.
5. Т. Х. Кормен,. Алгоритмы: построение и анализ. 3-е издание. доп издание. Перевод с англ. С. К. Ландо. Дополнение М. В. Ульянова / Т. Х. Кормен.— Москва: Вильямс, 2013.
6. Левитин, А. В. Алгоритмы. Введение в разработку и анализ / А. В. Левитин.— Москва: Вильямс, 2006.