МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»**

Кафедра теоретических основ компьютерной безопасности и криптографии

# Методы построения неизоморфных графов без проверки на изоморфизм

АВТОРЕФЕРАТ
дипломной работы

студентки 6 курса 631 группы
специальности 10.05.01 Компьютерная безопасность
факультета компьютерных наук и информационных технологий
Юрченко Елены Михайловны

Научный руководитель
д. ф.-м. н., доцент _____ М. Б. Абросимов
22.01.2024 г.

Заведующий кафедрой
д. ф.-м. н., доцент _____ М. Б. Абросимов
22.01.2024 г.

Саратов 2024

# ВВЕДЕНИЕ

Одним из ключевых вопросов, связанных с графами, является возможность их генерации. Обычно, переборные алгоритмы строят все возможные графы, среди которых есть и изоморфные. При этом, в большинстве случаев, необходимости рассмотрения изоморфных друг другу графов нет, поскольку такие графы обладают одинаковыми свойствами. Простейший подход к исключению изоморфных копий состоит в проверке изоморфности сгенерированного графа каждому из найденных ранее графов.

В данной работе планируется решить следующие задачи:

– теоретически исследовать алгоритмы, позволяющие генерировать неизоморфные графы без проверки на изоморфизм;

– практически реализовать алгоритмы, позволяющие генерировать неизоморфные графы без проверки на изоморфизм;

– проанализировать существующие подходы к генерации неизоморфных графов без проверки на изоморфизм.

В теории графов существует множество понятий, связанных с характеристиками графа. Одно из таких понятий – хроматическое число. Оно определяет минимальное количество цветов, необходимых для раскраски вершин графа. Исследование свойств векторов степеней и их связи с хроматическим числом является важным направлением в теории графов, так как позволяет лучше понять структуру и особенности графа.

Результаты данной работы могут быть применены для решения различных практических задач, связанных с раскраской графа.

Цель работы: разработать и реализовать алгоритмы генерации неизоморфных графов без проверки на изоморфизм; изучить свойства векторов степеней графов и их связь с хроматическим числом; практически реализовать программу на C#.

Дипломная работа состоит из введения, 3 разделов, заключения, списка использованных источников и 2 приложений. Общий объем работы – 65

страниц, из них 40 страниц – основное содержание, включая 23 рисунка и 5 таблиц, список использованных источников из 21 наименований.

## КРАТКОЕ СОДЕРЖАНИЕ

Первый раздел работы содержит теоретические понятия, необходимые для дальнейшего изучения темы. В нем приводятся основные определения и обозначения, связанные с теорией графов.

*Неориентированным графом* называется пара $G = (V, \alpha)$, где $\alpha$ – симметричное и антирефлексивное отношение на множестве вершин $V$, называемое *отношением смежности*. Если $(v, u) \in \alpha$, то вершины $u$ и $v$ *смежны* и эти вершины соединены *ребром* $(u, v)$ [1].

Два графа $G_1 = (V_1, \alpha_1)$ и $G_2 = (V_2, \alpha_2)$ называются *изоморфными*, если можно установить взаимно однозначное соответствие $f$: $V_1 \to V_2$, сохраняющее отношение смежности: $(u, v) \in \alpha_1 \Leftrightarrow (f(u), f(v)) \in \alpha_2$, для любых $u, v \in V_1$. В этом случае пишут $G_1 \cong G_2$.

*Степенью вершины* $v$ в неориентированном графе $G$ будем называть количество вершин в $G$, смежных с $v$, и обозначать через $d(v)$.

*Вектором степеней* графа $G$ называется упорядоченная по невозрастанию последовательность степеней вершин.

*Канонический код* – это представление графа в виде строки, которое не зависит от порядка нумерации вершин.

*Инвариантом графа* $G$ называется набор его характеристик, одинаковых для всех изоморфных ему графов.

*Полным инвариантом графа* является некоторая характеристика, которая представляет его структуру с точностью до изоморфизма, то есть равенство полных инвариантов для двух графов гарантирует их изоморфизм.

*Матричный код* – это элементы матрицы смежности, записанные выше главной диагонали (так как ниже главной диагонали – будет симметрично).

*Раскраской графа* называется такое приписывание цветов его вершинам, что никакие две смежные вершины не получают одинакового цвета.

Во втором разделе работы рассматриваются методы построения графов. Кроме того, для построенных графов была произведена раскраска вершин.

Для работы с графами часто применяется матричный код.

Матрица отношения смежности графа называется его матрицей смежности. Пусть $G = (V, \alpha)$ – $n$-вершинный граф. Тогда его матрица смежности $A = M(\alpha)$ имеет размерность $n$ x $n$ , а на позиции $(i, j)$ стоит 1, если есть дуга $(v_i,v_j)$ и 0 в противном случае:

Общий вид матрицы представлен на рисунке ниже.

$$A = \begin{bmatrix} X_{1,1} & X_{1,2} & \dots & X_{1,n} \\ X_{2,1} & X_{2,2} & \dots & X_{2,n} \\ & \dots\dots\dots & & \\ X_{n,1} & X_{n,2} & \dots & X_{n,n} \end{bmatrix}$$

Рисунок 1 – Матрица смежности графа

Если выписать все элементы, то получается некоторое двоичное число – это и будет кодом графа: $X_{1,1}$, $X_{1,2}$, ... , $X_{1,n}$, $X_{2,1}$, $X_{2,2}$, ... , $X_{2,n}$, ..... , $X_{n,1}$, $X_{n,2}$, ..., $X_{n,n}$.

Само число не может являться инвариантом, так как матрицы смежности у изоморфных графов могут различаться. Если из всех изоморфных графов выбрать матрицу с максимальным или минимальным значением, то есть её максимальный или минимальный код, тогда получится полный инвариант.

В данной работе рассматривается задача генерации неизморфных графов с помощью матричного перебора, в котором перебираются все числа от 1 до $2^k$. В этом случае $k$ – константа, которая вычисляется $k = (n \times n - n) / 2$, а $n$-количество вершин графа.

Существует насколько способов проверки на изоморфизм.

**1. Простой перебор.** Для этого применяется хеш-таблица с кодами графа, изначально она пуста. Происходит обработка текущего графа: идет перебор

всех возможных вершин. Для каждого такого графа вычисляется его код и проверяется: если этот код есть в таблице – то этот граф был изоморфный некоторому раньше, тогда происходит переход к следующему графу. Если же на всех возможных нумерациях граф дал уникальные коды, тогда код от начального графа заносится в хэш-таблицу, а граф считается неизоморфным и выводится в консоль или в файл в зависимости от параметра.

Таким образом, данное решение не требует большого количества памяти, но оно крайне не эффективно по времени. Каждый раз происходит полный перебор $n!$ перестановок для каждого найденного графа и каждый новый граф увеличивает время проверки.

**2. Перебор с помощью кодов.** Есть хэш-таблица с кодами графов. Для текущего графа происходит проверка: есть ли в хэш-таблице код этого графа? Если да – то происходит переход к следующему графу. Если нет – то перебираются все перестановки текущего графа, для каждой перестановки получается изоморфный граф, для каждого такого графа высчитывается его код и добавляется в хэш-таблицу.

**3. Метод канонических представителей.** Чтобы уменьшить количество используемой памяти, о чём говорилось в предыдущем методе, храниться будет только один код: максимальный или минимальный. Максимальный и минимальный код – это полный инвариант графа. Тогда для всех новых графов вычисляется максимальный или минимальный код и сравнивается с существующими кодами. Если такого кода не существует, нужно добавить его в список существующих. При таком подходе сам граф можно не хранить. Поскольку код графа является уникальным, его можно восстановить. При этом, вычисление минимального и максимального кода также является трудоемкой задачей.

Из всех перечисленных способов проверки на изоморфизм, самым эффективным является последний вариант, так как затраты на память сведены к минимальному значению.

Далее происходит раскраска вершин графа.

Одна из важных задач в теории графов заключается в определении минимальной раскраски для неориентированного графа.

*Раскраской графа* называется такое приписывание цветов его вершинам, что никакие две смежные вершины не получают одинакового цвета. Если граф можно раскрасить в $p$ цветов, то говорят, что он обладает $p$-раскраской и является $p$-раскрашиваемым. Наименьшее $p$, при котором граф $G$ имеет $p$-раскраску, называется *хроматическим числом графа* $G$ и обозначается $\chi(G)$.

Поиск хроматического числа относится к классу $NP$ и не позволяет отыскание оптимального решения, поэтому на практике для ее решения применяются различные алгоритмы.

В работе были реализованы два алгоритма для нахождения хроматического числа – жадный алгоритм и полный перебор.

Жадный алгоритм не всегда гарантирует нахождение оптимального хроматического числа. Некоторые графы могут требовать большего числа цветов, чем предложенный алгоритм выдаст. Но в большинстве случаев, особенно при работе с графами небольшого или среднего размера, алгоритм дает удовлетворительные результаты.

Используя полный перебор, алгоритм последовательно просматривает все возможные комбинации раскрасок графа, чтобы определить, какое число цветов потребуется для его правильной раскраски.

Одним из главных преимуществ полного перебора является его гарантированная точность результатов.

Таким образом, полный перебор дает точное значение $\chi(G)$, но имеет экспоненциальную сложность. Жадный алгоритм работает быстрее, но не гарантирует оптимальности.

В третьем раздела описывается разработанная и реализованная программа «GraphGenerator», с помощью которой происходит генерация неориентированных графов с заданным числом вершин. Программа была написана на языке программирования C# в среде разработки Microsoft Visual Studio 2022.

Для проведения сравнительного анализа генерации результаты их работы были систематизированы, и представлены в таблице 1.

Таблица 1 – Сравнение работ методов генерации

| | | Время работы (в секундах) | | | |
|---|---|---|---|---|---|
| \|V\| | N | Канонический код | Простой перебор | Перебор с помощью кодов | Оптимизация |
| 1 | 1 | 0,0492315 | 0,0064928 | 0,009566 | 0,0085695 |
| 2 | 1 | 0,0166033 | 0,0068952 | 0,0110153 | 0,0107141 |
| 3 | 2 | 0,0179942 | 0,0066162 | 0,0091125 | 0,0085514 |
| 4 | 6 | 0,0222291 | 0,0108774 | 0,0146281 | 0,0123753 |
| 5 | 21 | 0,0749271 | 0,105699 | 0,0158861 | 0,016974 |
| 6 | 112 | 1,3823236 | 27,1205932 | 0,1968414 | 0,1292081 |
| 7 | 853 | 3,6254359 | 65035,757207 | 7,8926612 | 3,4917614 |
| 8 | 1117 | 369,15485 | 216900,52368 | 1225,1523 | 293,51953 |
| 9 | 261080 | 96258,563 | 625896389,78 | 79625,952 | 12593,782 |
| 10 | 11716571 | 108526,365 | 998369852,62 | 97816,843 | 31684,561 |
| 11 | 1006700565 | 152579,269 | 11089638525,35 | 116178,269 | 548616,792 |

Для проверки работоспособности программы, была проведена сравнительная характеристика с программой geng. Результат работы программы с заданным числом вершин продемонстрирована в таблице 2.

Таблица 2 – Результат работы программы geng

| Количество вершин в графе | Количество графов | Время |
|---|---|---|
| 0 | 0 | 0,00 |
| 1 | 1 | 0,00 |
| 2 | 1 | 0,00 |
| 3 | 2 | 0,00 |
| 4 | 6 | 0,00 |
| 5 | 21 | 0,00 |

| | | |
|---|---|---|
| 6 | 112 | 0,00 |
| 7 | 853 | 0,00 |
| 8 | 11117 | 0,03 |
| 9 | 261080 | 0,66 |
| 10 | 11716571 | 23,50 |
| 11 | 1006700565 | 16235,50 |

Для построенных графов был проведен вычислительный эксперимент, который позволит посмотреть связь хроматического числа и вектора степеней.

Для исследования связи хроматического числа и вектора степеней графа была выполнена генерация с формированием групп графа по размеру вектора степеней и хроматического числа.

Для удобного просмотра, графы были упорядочены в зависимости от их вектора степеней.

Таким образом, можно увидеть, что при увеличении степени вершин требуется больше цветов для правильной раскраски графа.

Таблица 3 – Связь вектора степеней и хроматического числа

| | | Размер вектора степеней | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 |
| Хроматическое число | 2 | 3656 | 673 | 171 | 42 | 17 | 5 | 3 | 1 | 1 |
| | 3 | 1583090 | 70719 | 4767 | 469 | 64 | 12 | 2 | 1 | 0 |
| | 4 | 7528878 | 155367 | 5245 | 290 | 26 | 3 | 1 | 0 | 0 |
| | 5 | 2461199 | 32163 | 853 | 46 | 4 | 1 | 0 | 0 | 0 |
| | 6 | 135278 | 2040 | 74 | 5 | 1 | 0 | 0 | 0 | 0 |
| | 7 | 4305 | 110 | 6 | 1 | 0 | 0 | 0 | 0 | 0 |
| | 8 | 156 | 7 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 9 | 8 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 10 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Для поиска хроматического числа были реализованы два алгоритма: жадный алгоритм и полный перебор.

В работе был проведем сравнительный анализ работы двух алгоритмов.

На рисунке ниже был сформирован график результатов работы двух алгоритмов для поиска хроматического числа.

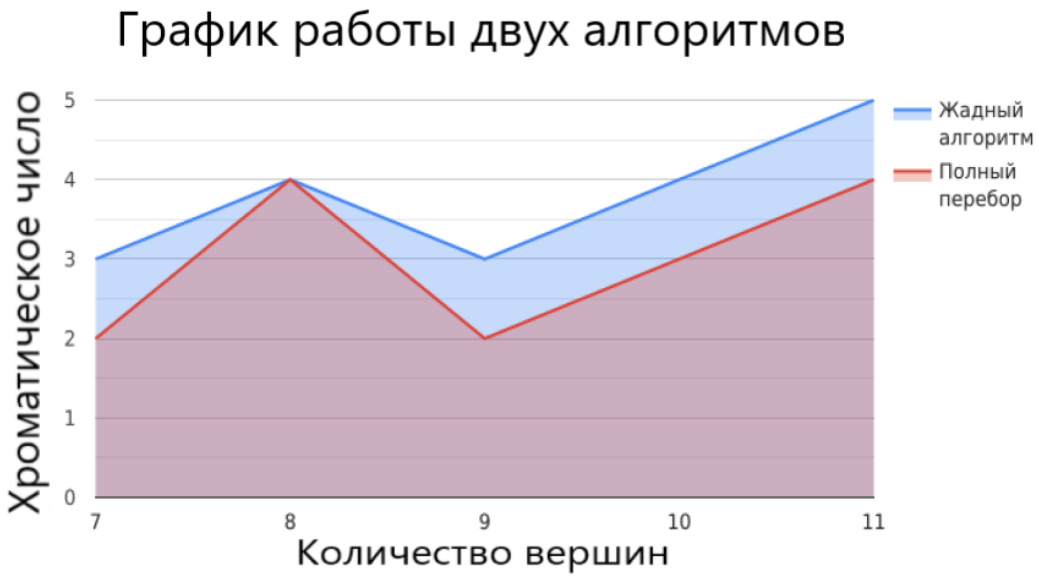


Рисунок 2 – График работы жадного алгоритма и полного перебора

Таким образом, можно сделать вывод, что для графов с большим количеством вершин использование полного перебора является наиболее оптимальным и результативным.

# ЗАКЛЮЧЕНИЕ

В данной работе было проведено исследование в области генерации неизоморфных неориентированных графов. Программа далека от результата времени работы «geng» и не может быть использована для генерации графов с большим количеством вершин, но свою задачу выполняет. Таким образом, задача, поставленная в работе, решена.

В результате проделанной работы, разработана и реализована на языке программирования C# программа «GraphGenerator», которая генерирует неориентированные графы с заданным числом вершин. За время работы были сгенерированы графы с числом вершин до 11 включительно.

На основе сгенерированных графов были рассмотрены алгоритмы поиска оптимальной раскраски графа. По полученным результатам была осуществлена сравнительная характеристика алгоритмов, на основе которой был сделан вывод о целесообразности их применения на практике.

Дополнительно был проведен вычислительный эксперимент для каждого из построенных графов, с целью определения зависимости между хроматическим числом и вектором степеней. Результаты эксперимента позволили выяснить, как эти два параметра связаны между собой.

Таким образом, можно сделать вывод, что свойство векторов степеней и его связь с хроматическим числом являются важными для понимания и анализа графов. Они могут быть использованы для решения различных задач, таких как определение минимального количества цветов, необходимых для раскраски графа или для анализа свойств сетей и компьютерных систем.