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

**Число совершенного доминирования и принудительное геодезическое число**

НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА

студентки 6 курса 631 группы

специальности 10.05.01 Компьютерная безопасность

факультета компьютерных наук и информационных технологий

Стыценковой Валерии Сергеевны

|  |  |  |  |
| --- | --- | --- | --- |
| Научный руководитель  д. ф.-м. н., доцент |  | \_\_\_\_\_\_\_\_\_\_ | М. Б. Абросимов |
|  |  | подпись, дата |  |

Саратов 2024**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ3

1 Теоретические сведения4

1.1 Ключевые определения4

1.2 Доминирование. Число совершенного доминирования5

1.3 Геодоминирование. Принудительное геодезическое число7

2 Практическая реализация9

1.4 Описание программы9

1.5 Результаты работы программы11

1.6 Выводы13

ЗАКЛЮЧЕНИЕ16

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ17

ПРИЛОЖЕНИЕ А Листинг программы main.py18

ПРИЛОЖЕНИЕ Б Галерея графов22

**ВВЕДЕНИЕ**

Графы являются мощным инструментом для моделирования различных систем и объектов реального мира, а изучение свойств графов помогает лучше анализировать эти системы. Ключевую роль в понимании свойств и структурных особенностей графов играют инварианты, то есть характеристики графа, не зависящие от его представления.

**Целью данной работы является** изучение таких инвариантов графов, как число совершенного доминирования и принудительное геодезическое число, а также изучение возможных зависимостей между ними.

**Задачами данной работы являются:** изучение уже существующих исследований по принудительному геодезическому числу и числу совершенного доминирования, реализация программы, выполняющей подсчёт значений данных инвариантов для всех графов с числом вершин от 2 до 10, представление результатов работы программы в виде таблиц, показывающих распределение графов по значениям инвариантов, анализ полученных результатов.

**1 Теоретические сведения**

**1.1 Ключевые определения**

Далее приведены ключевые определения и понятия, требующиеся для реализации программы и анализа результатов её работы, по книгам [1, 2].

**Неориентированным графом** (или просто **графом**) называется упорядоченная пара , где – симметричное и антирефлексивное отношение на множестве вершин , называемое **отношением смежности**. Если , то говорят, что вершины и **смежны**, и эти вершины соединены **ребром** . При этом и – это одно и то же ребро, которое обозначают . Говорят, что ребро **инцидентно** каждой из вершин и , и этивершины называются **концевыми вершинами** или **концами** ребра . Два ребра называются **смежными**, если они имеют общую концевую вершину.

Граф, любые две вершины которого смежны, называется **полным** и обозначается .

**Степенью вершины** в неориентированном графе будем называть количество вершин в , смежных с , и обозначать через . Минимальная из степеней вершин графа обозначается , а максимальная – . Вершина, не смежная ни с одной другой вершиной, называется **изолированной**, а вершина, смежная со всеми остальными вершинами, называется **полной** или **доминирующей**.

Два графа и называются **изоморфными**, если можно установить взаимно однозначное соответствие , сохраняющее отношение смежности: .

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

Матрица отношения смежности графа называется его **матрицей смежности**. Пусть – -вершинный граф. Тогда его матрица смежности имеет размерность , а на позиции стоит , если есть дуга , и в противном случае:

Если выписать элементы матрицы смежности по строкам, то получится некоторое двоичное число – код графа:

,…,,,…,,…,

**Путём** в графе , называется последовательность рёбер, в которой каждые два соседних ребра имеют общую вершину, и никакое ребро не встречается более одного раза. **Длиной пути** в графе называется количество входящих в состав этого пути рёбер.

**Простым путём** называется путь, каждая вершина которого принадлежит не более чем двум его рёбрам. Если начальная вершина пути совпадает с конечной, то такой путь называется **циклическим**.

**Цепью** называется простой путь, не являющийся циклическим. Две вершины пути, принадлежащие только одному ребру, называются **концами цепи (пути)**.

Вершины и называются **связанными**, если в существует проходящий через них путь. Если вершины и связаны, то наименьшая из длин цепей, соединяющих и , называется **расстоянием** между этими вершинами.

Отношение связанности является эквивалентностью на множестве вершин графа. Классы этого отношения называются **компонентами связности** (или просто **компонентами**). Граф с универсальным отношением связанности называется **связным**.

**1.2** **Доминирование. Число совершенного доминирования.**

Далее приведены определения и понятия, связанные с доминированием и совершенным доминированием, по источникам [1, 3, 4].

**Доминирующее множество** – это такое подмножество вершин , что любая вершина не из смежна хотя бы одной вершине из . Говорят, что вершины и доминируют друг друга, если в графе есть ребро или .

Если никакая собственная часть доминирующего множества не является доминирующим множеством, то множество называется **минимальным доминирующим множеством**.

Доминирующее множество графа минимальной мощности называется его **наименьшим доминирующим множеством**.

**Число доминирования** – это число вершин в наименьшем доминирующем множестве графа .

Доминирующее множество графа называется **совершенным**, если каждая вершина графа доминируется в точности одной вершиной из . Очевидно, что совершенное доминирующее множество будет являться и наименьшим доминирующим множеством. Однако не у каждого графа есть совершенное доминирующее множество. Например, полное n-арное дерево высоты больше 1 не имеет совершенного доминирующего множества. Цепь имеет два совершенных множества: две вершины степени 1 или две вершины степени 2.

**Число совершенного доминирования** – это число вершин в наименьшем совершенном доминирующем множестве графа .

На рисунке 1 изображены пять примеров доминирующих множеств графа. В этих примерах каждая синяя вершина смежна по меньшей мере одной красной вершине и говорят, что синие вершины доминируются красными вершинами. Доминирующее число этого графа равно 2 – примеры (b), (c), (d) и (e) показывают, что существует доминирующее множество с двумя вершинами. Эти примеры представляют собой все возможные наименьшие доминирующие множества данного графа. Можно проверить, что для него не существует доминирующего множества только с одной вершиной. На примерах (c) и (d) изображены совершенные доминирующие множества, поскольку каждая синяя вершина доминируется в точности одной красной вершиной. Таким образом, число совершенного доминирования данного графа также равно 2.

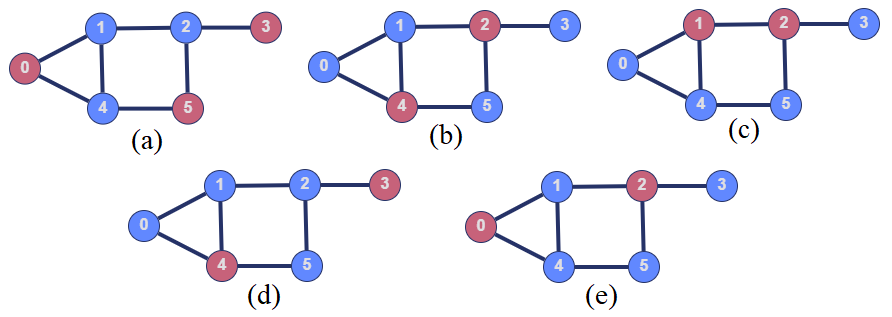


Рисунок 1 – Примеры наименьших и совершенных доминирующих множеств

**1.3** **Геодоминирование. Принудительное геодезическое число**

Далее приведены определения и понятия, связанные с геодоминированием и принудительным геодезическим числом, по источникам [1, 5].

Любой кратчайший путь (длины ) между вершинами и называется **геодезическим**. Говорят, что вершины и **геодоминируют** вершины, лежащие на соединяющем их геодезическом пути.

Граф, в котором для любых двух вершин существует единственная соединяющая их геодезическая цепь, называется **геодезическим графом***.*

**Геодезическиммножеством** графа называется множество его вершин такое, что каждая вершина графа принадлежит какой-либо геодезической цепи, соединяющей пару вершин из . Геодезическое множество называется **минимальным**, если никакое его собственное подмножество не является геодезическим множеством.

**Геодезическимчислом** графа называется минимальная мощность его геодезического множества, а сами такие множества называются **наименьшими геодезическими множествами**. В некоторых работах вместо термина «геодезическое число» используется термин **«число геодоминирования»**.

Для наименьшего геодезического множества графа его подмножество такое, что оно не является частью никакого другого наименьшего геодезического множества, называется **принудительным подмножеством**. Минимальная мощность принудительного подмножества среди всех принудительных подмножеств наименьших геодезических множеств графа называется **принудительным геодезическим числом** графа .

Граф, представленный на рисунке 2, имеет геодезическое число . Четыре его наименьших геодезических множества: , , и . Так как только содержит вершину , то мощность принудительного подмножества принудительное геодезическое число . Мощность других принудительных подмножеств для .

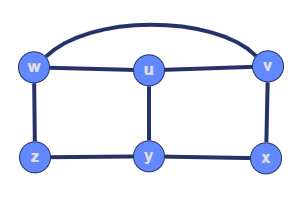


Рисунок 2 – Пример принудительного подмножества с минимальной мощностью

Очевидно, что если граф имеет только одно минимальное геодезическое множество, то в этом и только в этом случае .

тогда и только тогда, когда граф имеет более одного минимального геодезического множества, причём некоторая вершина графа принадлежит только одному из них.

тогда и только тогда, когда каждая вершина в каждом минимальном геодезическом множестве принадлежит хотя бы двум минимальным геодезическим множествам.

**2 Практическая реализация**

**2.1. Описание программы**

В практической части работы была написана программа на языке Python в IDE PyCharm, выполняющая поиск числа совершенного доминирования и принудительного геодезического числа для заданных графов. Для проведения вычислений использовался компьютер с четырьмя процессорными ядрами с тактовой частотой 1101МГц, работающий под управлением операционной системы Windows.

В качестве входных данных использовались связные неориентированные графы с числом вершин , сгенерированные с помощью консольных утилит geng (для генерации графов) и showg (для представления графов в виде матриц смежности), входящих в пакет nauty для Linux систем [6].

В работе с данными утилитами использовалась UNIX-подобная среда и оболочка командной строки Cygwin [7].

В программе были использованы следующие пакеты:

* **Начало формы**itertools – стандартная библиотека, предоставляющая инструменты для работы с итерациями;
* multiprocessing – библиотека, позволяющая запускать задачи в параллельном режиме;
* time – модуль, предоставляющий функции для работы со временем;
* tqdm – модуль, предназначенный для быстрого внедрения индикаторов выполнения во внешние интерфейсы Python;
* collections – библиотека, предоставляющая специализированные типы данных контейнеров.

Для повышения удобства тестирования программы, она разделена на функции. Каждая функция реализует отдельный шаг алгоритма и возвращает список множеств, представляющих результат этого шага.

Вычисление принудительного геодезического числа выполняется в следующих четырёх функциях:

1. find\_all\_shortest\_paths – поиск всех кратчайших путей между всеми вершинами графа с помощью обхода в ширину;
2. find\_min\_geodetic\_sets – перебор всех комбинаций вершин графа, начиная с комбинаций с наименьшим числом вершин. Поиск минимальных геодезических множеств, то есть таких множеств вершин с минимальной мощностью, что в кратчайших путях между ними встречаются все остальные вершины графа;
3. find\_minimal\_forcing\_subsets – поиск в каждом множестве, полученном на предыдущем шаге, принудительного подмножества, то есть такого подмножества, что оно не является частью никакого другого множества. Среди всех найденных принудительных подмножеств функция возвращает подмножество с минимальной мощностью;
4. find\_forcing\_geodetic\_number – подсчёт длины подмножества, найденного с помощью функции find\_minimal\_forcing\_subsets, если на шаге 2 найдено больше одного минимального геодезического множества. В противном случае функция возвращает принудительное геодезическое число, равное 0.

Вычисление числа совершенного доминирования выполняется в следующих пяти функциях:

1. is\_dominating\_set – проверка, является ли заданное множество вершин доминирующим, то есть доминируют ли его вершины все остальные вершины графа;
2. find\_minimum\_dominating\_sets­­­­ – перебор всех комбинаций вершин графа, начиная с комбинаций с наименьшим числом вершин, и поиск среди них наименьших доминирующих множеств с помощью функции is\_dominating\_set;
3. is\_perfect\_dominating\_set – проверка, имеют ли все вершины графа, не принадлежащие наименьшему доминирующему множеству, в точности одного соседа из этого множества;
4. find\_perfect\_dominating\_set – нахождение первого множества, удовлетворяющего условию, с помощью функции is\_perfect\_dominating\_set;
5. find\_perfect\_dominating\_number – подсчёт длины совершенного доминирующего множества, если оно найдено. В противном случае функция возвращает число совершенного доминирования, равное 0.

Графы считываются из входного файла, содержащего матрицы смежности, с помощью функции parse\_graph\_file. В функции process\_graph осуществляется параллельное вычисление значений инвариантов для каждого графа. Прогресс обработки графов отображается в консоли с помощью модуля tqdm. Программа измеряет общее время выполнения с помощью функции time.perf\_counter. В конце работы алгоритма в консоль выводится матрица распределения значений инвариантов, полученные значения для каждого графа сохраняются в текстовый файл, и программа предлагает пользователю найти интересующие его графы по заданным значениям инвариантов и вывести в консоль их матрицы смежности.

**2.2.** **Результаты работы программы**

Программа была запущена для всех связных неориентированных графов с числом вершин от 2 до 10. Количество таких графов для каждого числа вершин приведено в таблице 1.

Таблица 1 – Количество связных неориентированных графов

|  |  |
| --- | --- |
|  | Количество графов |
| 2 | 1 |
| 3 | 2 |
| 4 | 6 |
| 5 | 21 |
| 6 | 112 |
| 7 | 853 |
| 8 | 11117 |
| 9 | 261080 |
| 10 | 11716571 |

Результаты вычислений приведены в таблицах 2-10. По вертикали отображается число совершенного доминирования, а по горизонтали – принудительное геодезическое число.

Таблица 2 – Результат для 2-вершинных графов

|  |  |
| --- | --- |
|  | 0 |
| 0 | 0 |
| 1 | 1 |

Таблица 3 – Результат для 3-вершинных графов

|  |  |
| --- | --- |
|  | 0 |
| 0 | 0 |
| 1 | 2 |

Таблица 4 – Результат для 4-вершинных графов

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | 0 | 0 |
| 1 | 4 | 0 |
| 2 | 1 | 1 |

Таблица 5 – Результат для 5-вершинных графов

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | 1 | 0 | 1 |
| 1 | 9 | 2 | 0 |
| 2 | 7 | 1 | 0 |

Таблица 6 – Результат для 6-вершинных графов

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | 13 | 7 | 1 |
| 1 | 25 | 8 | 1 |
| 2 | 37 | 17 | 1 |
| 3 | 2 | 0 | 0 |

Таблица 7 – Результат для 7-вершинных графов

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | 159 | 122 | 11 |
| 1 | 91 | 57 | 8 |
| 2 | 242 | 123 | 4 |
| 3 | 25 | 9 | 2 |

Таблица 8 – Результат для 8-вершинных графов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 |
| 0 | 2612 | 2471 | 361 | 4 |
| 1 | 512 | 457 | 75 | 0 |
| 2 | 2429 | 1328 | 225 | 3 |
| 3 | 358 | 248 | 28 | 0 |

Таблица 9 – Результат для 9-вершинных графов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 |
| 0 | 70846 | 81740 | 14067 | 116 |
| 1 | 4965 | 6020 | 1341 | 20 |
| 2 | 38040 | 23914 | 4394 | 50 |
| 3 | 7755 | 7004 | 650 | 10 |
| 4 | 119 | 28 | 1 | 0 |

Таблица 10 – Результат для 10-вершинных графов

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | 3326777 | 4788805 | 780871 | 7705 | 10 | 1 |
| 1 | 94521 | 144748 | 34749 | 650 | 0 | 0 |
| 2 | 1088268 | 748440 | 127123 | 2406 | 29 | 0 |
| 3 | 257099 | 281846 | 26224 | 396 | 0 | 0 |
| 4 | 3717 | 2008 | 151 | 5 | 1 | 0 |
| 5 | 21 | 0 | 0 | 0 | 0 | 0 |

Таблица 11 содержит данные о времени, которое потребовалось программе для расчёта каждой из таблиц 2-10.

Таблица 11 – Время работы программы

|  |  |
| --- | --- |
|  | Время работы |
| 2 | 0.53 секунд |
| 3 | 0.53 секунд |
| 4 | 0.53 секунд |
| 5 | 0.53 секунд |
| 6 | 0.62 секунд |
| 7 | 1.9 секунд |
| 8 | 21.4 секунд |
| 9 | 13 минут |
| 10 | 17 часов |

**2.3. Выводы**

На основе таблиц были сформулированы следующие выводы:

1. Ни в одной из таблиц не нашлось графа с принудительным геодезическим числом или числом совершенного доминирования, превышающим . Можно выдвинуть гипотезу, что данная закономерность сохранится при увеличении числа вершин графа.
2. С увеличением числа вершин растёт доля графов, не имеющих совершенного доминирующего множества. В таблице 12 для каждого числа вершин представлено процентное соотношение количества таких графов к общему числу связных графов.

Таблица 12 – Доля графов с числом совершенного доминирования, равным 0

|  |  |
| --- | --- |
|  | Доля графов с |
| 2 | 0% |
| 3 | 0% |
| 4 | 0% |
| 5 | 10% |
| 6 | 18% |
| 7 | 34% |
| 8 | 49% |
| 9 | 64% |
| 10 | 75% |

1. Наоборот, наблюдается тенденция к уменьшению доли графов, имеющих принудительное геодезическое число, равное нулю. В таблице 13 для каждого числа вершин представлено процентное соотношение количества таких графов к общему числу связных графов.

Таблица 13 – Доля графов с принудительным геодезическим числом, равным 0

|  |  |
| --- | --- |
|  | Доля графов с |
| 2 | 100% |
| 3 | 100% |
| 4 | 83% |
| 5 | 80% |
| 6 | 69% |
| 7 | 60% |
| 8 | 53% |
| 9 | 47% |
| 10 | 41% |

1. Существует только один граф с , , . Это регулярный граф, степень всех его вершин равна 5.
2. Существует только один граф с , , .
3. Существует только один граф с , , .

Графические представления этих графов приведены в Приложении Б.

В ходе исследования были определены комбинации числа совершенного доминирования и принудительного геодезического числа, для которых наблюдается наименьшее количество графов. Эти графы предложены для дальнейшего анализа, и их графические представления приведены в Приложении Б.

**ЗАКЛЮЧЕНИЕ**

В этой работе были изучены базовые понятия теории графов, включая определения совершенного доминирующего множества, принудительного подмножества, числа совершенного доминирования и принудительного геодезического числа.

На языке Python в среде разработки PyCharm была реализована программа, которая методом перебора осуществляет поиск значений данных инвариантов и строит таблицу их распределения для всех связных неориентированных графов с числом вершин от двух до десяти. Программа использует алгоритм обхода в ширину для вычисления кратчайших путей в графе.

На основе табличных данных были сформулированы наблюдения и выдвинуты гипотезы, требующие дальнейшего изучения.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Абросимов, М. Б. Методы оптимизации графовых систем. Учебное пособие / М. Б. Абросимов. – М. : Издательство «Перо», 2024. – 110 с.
2. Богомолов, А. М. Алгебраические основы теории дискретных систем / А. М. Богомолов, В. Н. Салий. М. : Наука. Физматлит, 1997. – 368 с.
3. Доминирующее множество [Электронный ресурс] / URL : https://ru.wikipedia.org/wiki/Доминирующее\_множество (дата обращения 20.09.2024). – Загл. с экрана. – Яз. рус.
4. Livingston, M. Perfect dominating sets / M. Livingston, Q. F. Stout. Congressus Numerantium, 79, 187-203, 1990
5. Chartrand, G. The forcing geodetic number of a graph / G. Chartrand, P. Zhang. Discussiones Mathematicae Graph Theory. – 1999 – Vol. 19, № 1 – P. 45–58.
6. McKay, B. Nauty and Traces / B. McKay, A. Piperno [Электронный ресурс]. – URL: https://pallini.di.uniroma1.it/ (дата обращения 20.09.2024). – Загл. с экрана. – Яз. англ.
7. Cygwin [Электронный ресурс] : [сайт] – URL: https://cygwin.com/ (дата обращения 20.09.2024). – Загл. с экрана. – Яз. англ.

**ПРИЛОЖЕНИЕ А**

**Листинг программы main.py**

from itertools import combinations

from multiprocessing import Pool, cpu\_count

import time

from collections import deque

from tqdm import tqdm

def find\_all\_shortest\_paths(adj\_matrix):

num\_vertices = len(adj\_matrix)

def bfs\_all\_paths(u, v):

queue = deque([(u, [u])])

all\_paths = []

shortest\_length = None

while queue:

current\_vertex, path = queue.popleft()

if current\_vertex == v:

if shortest\_length is None:

shortest\_length = len(path)

all\_paths.append(path)

elif len(path) == shortest\_length:

all\_paths.append(path)

else:

break

continue

for neighbor in range(num\_vertices):

if adj\_matrix[current\_vertex][neighbor] == 1 and neighbor not in path:

queue.append((neighbor, path + [neighbor]))

return all\_paths

shortest\_paths = {}

for u in range(num\_vertices):

for v in range(num\_vertices):

if u != v:

shortest\_paths[(u, v)] = bfs\_all\_paths(u, v)

return shortest\_paths

def find\_min\_geodetic\_sets(graph\_paths):

vertices = set()

for key in graph\_paths.keys():

vertices.update(key)

vertices = list(vertices)

n = len(vertices)

valid\_sets = []

for r in range(2, n + 1):

found\_set = False

for combo in combinations(vertices, r):

visited\_vertices = set()

for v1 in combo:

for v2 in combo:

if v1 != v2 and (v1, v2) in graph\_paths:

for path in graph\_paths[(v1, v2)]:

visited\_vertices.update(path)

if visited\_vertices == set(vertices):

valid\_sets.append(combo)

found\_set = True

if found\_set:

break

return valid\_sets

def find\_minimal\_forcing\_subsets(sets):

forcing\_subsets = []

for i, current\_set in enumerate(sets):

found = False

for r in range(1, len(current\_set) + 1):

for subset in combinations(current\_set, r):

subset = set(subset)

is\_unique = True

for j, other\_set in enumerate(sets):

if i != j and subset.issubset(other\_set):

is\_unique = False

break

if is\_unique:

forcing\_subsets.append(subset)

found = True

break

if found:

break

return min(forcing\_subsets, key=len)

def find\_forcing\_geodetic\_number(adj\_matrix):

shortest\_paths = find\_all\_shortest\_paths(adj\_matrix)

min\_geodetic\_sets = find\_min\_geodetic\_sets(shortest\_paths)

if len(min\_geodetic\_sets) > 1:

return len(find\_minimal\_forcing\_subsets(min\_geodetic\_sets))

else:

return 0

def is\_dominating\_set(adj\_matrix, subset):

n = len(adj\_matrix)

dominated = set(subset)

for vertex in subset:

for neighbor in range(n):

if adj\_matrix[vertex][neighbor] == 1:

dominated.add(neighbor)

return len(dominated) == n

def find\_minimum\_dominating\_sets(adj\_matrix):

n = len(adj\_matrix)

min\_dominating\_sets = []

for k in range(1, n + 1):

for subset in combinations(range(n), k):

if is\_dominating\_set(adj\_matrix, subset):

if not min\_dominating\_sets or len(subset) == len(min\_dominating\_sets[0]):

min\_dominating\_sets.append(subset)

elif len(subset) < len(min\_dominating\_sets[0]):

min\_dominating\_sets = [subset]

if min\_dominating\_sets:

break

return min\_dominating\_sets

def is\_perfect\_dominating\_set(adj\_matrix, subset):

n = len(adj\_matrix)

for vertex in range(n):

if vertex not in subset:

neighbors\_count = sum(adj\_matrix[vertex][neighbor] for neighbor in subset)

if neighbors\_count != 1:

return False

return True

def find\_perfect\_dominating\_set(adj\_matrix):

min\_dominating\_sets = find\_minimum\_dominating\_sets(adj\_matrix)

for subset in min\_dominating\_sets:

if is\_perfect\_dominating\_set(adj\_matrix, subset):

return subset

return None

def find\_perfect\_dominating\_number(adj\_matrix):

perfect\_dominating\_set = find\_perfect\_dominating\_set(adj\_matrix)

if perfect\_dominating\_set is not None:

return len(perfect\_dominating\_set)

else:

return 0

def parse\_graph\_file(file\_path):

graphs = []

with open(file\_path, 'r') as f:

lines = f.readlines()

i = 0

while i < len(lines):

if lines[i].startswith("Graph"):

\_, order = lines[i].strip().split(" order ")

order = int(order[:-1])

i += 1

graph = []

for \_ in range(order):

graph.append(list(map(int, list(lines[i].strip()))))

i += 1

graphs.append(graph)

else:

i += 1

return graphs

def process\_graph(graph\_info):

idx, graph = graph\_info

min\_forc\_geo = find\_forcing\_geodetic\_number(graph)

min\_perf\_dom = find\_perfect\_dominating\_number(graph)

return idx, graph, min\_forc\_geo, min\_perf\_dom

if \_\_name\_\_ == "\_\_main\_\_":

input\_file = 'connected\_input5.txt'

output\_file = 'output5.txt'

graphs = parse\_graph\_file(input\_file)

num\_vertices = int(input\_file.split('input', 1)[1].split('\_')[0].split('.')[0])

table = [[0] \* (num\_vertices+1) for \_ in range(num\_vertices+1)]

graph\_infos = [(idx, graph) for idx, graph in enumerate(graphs)]

total\_graphs = len(graph\_infos)

start\_time = time.perf\_counter()

with Pool(cpu\_count()) as pool:

results = list(tqdm(pool.imap(process\_graph, graph\_infos), total=total\_graphs,

desc="Прогресс"))

with open(output\_file, 'w') as f:

for idx, graph, min\_forc\_geo, min\_perf\_dom in results:

f.write(f"Graph {idx + 1}: g={min\_forc\_geo} d={min\_perf\_dom}\n")

table[min\_perf\_dom][min\_forc\_geo] += 1

end\_time = time.perf\_counter()

print("Количество вершин:", num\_vertices)

print("Таблица распределения графов по значениям инвариантов:")

for row in table:

print(" ".join(map(str, row)))

elapsed\_time = end\_time - start\_time

print(f"Прошедшее время: {elapsed\_time:.4f} секунд")

while True:

forc\_geo\_num = input("Введите принудительное геодезическое число (0-10): ")

perf\_dom\_num = input("Введите число совершенного доминирования (0-10): ")

if forc\_geo\_num and perf\_dom\_num:

forc\_geo\_num = int(forc\_geo\_num)

perf\_dom\_num = int(perf\_dom\_num)

found\_graphs = [g for g in results if g[2] == forc\_geo\_num and g[3] == perf\_dom\_num]

if found\_graphs:

print(f"Графы с числом совершенного доминирования: {perf\_dom\_num}, "

f"принудительным геодезическим числом: {forc\_geo\_num}")

for idx, graph, forc\_geo\_num, perf\_dom\_num in found\_graphs:

print(graph)

print("")

else:

print("Графы с заданными параметрами не найдены.")

else:

break

**ПРИЛОЖЕНИЕ Б**

**Галерея графов**

|  |  |  |  |
| --- | --- | --- | --- |
| Рисунок графа | Число вершин | Принудительное геодезическое число | Число совершенного доминирования |
| 4_0_2.png | 4 | 0 | 2 |
| 4_1_2.png | 4 | 1 | 2 |
| 5_0_0.png | 5 | 0 | 0 |
| 5_2_0.png | 5 | 1 | 1 |
| 5_2_1.png | 5 | 1 | 1 |
| 5_1_2.png | 5 | 1 | 2 |
| 5_2_0.png | 5 | 2 | 0 |
| 6_0_3 (1).png | 6 | 0 | 3 |
| 6_0_3 (2).png | 6 | 0 | 3 |
| 6_2_0.png | 6 | 2 | 0 |
| 6_2_1.png | 6 | 2 | 1 |
| 6_2_2.png | 6 | 2 | 2 |
| 7_1_3 (1).png | 7 | 1 | 3 |
| 7_1_3 (2).png | 7 | 1 | 3 |
| 7_1_3 (3).png | 7 | 1 | 3 |
| 7_1_3 (4).png | 7 | 1 | 3 |
| 7_1_3 (5).png | 7 | 1 | 3 |
| 7_1_3 (6).png | 7 | 1 | 3 |
| 7_1_3 (7).png | 7 | 1 | 3 |
| 7_1_3 (8).png | 7 | 1 | 3 |
| 7_1_3 (9).png | 7 | 1 | 3 |
| 7_2_1 (1).png | 7 | 2 | 1 |
| 7_2_1 (2).png | 7 | 2 | 1 |
| 7_2_1 (3).png | 7 | 2 | 1 |
| 7_2_1 (4).png | 7 | 2 | 1 |
| 7_2_1 (5).png | 7 | 2 | 1 |
| 7_2_1 (6).png | 7 | 2 | 1 |
| 7_2_1 (7).png | 7 | 2 | 1 |
| 7_2_1 (8).png | 7 | 2 | 1 |
| 7_2_2 (1).png | 7 | 2 | 2 |
| 7_2_2 (2).png | 7 | 2 | 2 |
| 7_2_2 (3).png | 7 | 2 | 2 |
| 7_2_2 (4).png | 7 | 2 | 2 |
| 7_2_3 (1).png | 7 | 2 | 3 |
| 7_2_3 (2).png | 7 | 2 | 3 |
| 8_3_0_1.png | 8 | 3 | 0 |
| 8_3_0_2.png | 8 | 3 | 0 |
| 8_3_0_3.png | 8 | 3 | 0 |
| 8_3_0_4.png | 8 | 3 | 0 |
| 8_3_2_1.png | 8 | 3 | 2 |
| 8_3_2_2.png | 8 | 3 | 2 |
| 8_3_2_3.png | 8 | 3 | 2 |
| 9_2_4.png | 9 | 2 | 4 |
| 9_3_3_1.png | 9 | 3 | 3 |
| 9_3_3_2.png | 9 | 3 | 3 |
| 9_3_3_3.png | 9 | 3 | 3 |
| 9_3_3_4.png | 9 | 3 | 3 |
| 9_3_3_5.png | 9 | 3 | 3 |
| 9_3_3_6.png | 9 | 3 | 3 |
| 9_3_3_7.png | 9 | 3 | 3 |
| 9_3_3_8.png | 9 | 3 | 3 |
| 9_3_3_9.png | 9 | 3 | 3 |
| 9_3_3_10.png | 9 | 3 | 3 |
| 10_3_4_1.png | 10 | 3 | 4 |
| 10_3_4_2.png | 10 | 3 | 4 |
| 10_3_4_3.png | 10 | 3 | 4 |
| 10_3_4_4.png | 10 | 3 | 4 |
| 10_3_4_5.png | 10 | 3 | 4 |
| 10_4_0_1.png | 10 | 4 | 0 |
| 10_4_0_2.png | 10 | 4 | 0 |
| 10_4_0_3.png | 10 | 4 | 0 |
| 10_4_0_4.png | 10 | 4 | 0 |
| 10_4_0_5.png | 10 | 4 | 0 |
| 10_4_0_6.png | 10 | 4 | 0 |
| 10_4_0_7.png | 10 | 4 | 0 |
| 10_4_0_8.png | 10 | 4 | 0 |
| 10_4_0_9.png | 10 | 4 | 0 |
| 10_4_0_10.png | 10 | 4 | 0 |
| 10_4_4.png | 10 | 4 | 4 |
| 10_5_0.png | 10 | 5 | 0 |