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

СЛАБОЕ ГЕОДЕЗИЧЕСКОЕ ЧИСЛО И ЧИСЛО НЕСОКРАТИМОСТИ

НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА

студентки 5 курса 531 группы
направления 10.05.01 — Компьютерная безопасность
факультета КНиИТ
Ивановой Ксении Владиславовны

Проверил
д. ф.-м. н., профессор ______________ М. Б. Абросимов

Саратов 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В теории графов часто изучают числовые инварианты, чтобы лучше понять устройство графов и применять знания в решении сложных задач. Предлагаю остановиться на рассмотрении слабого геодезического числа и числа нескоратимости графа.

Обычно эти инварианты рассматривают отдельно и их связь почти не изучалась. Исследование того, как они связаны друг с другом, может дать новые интересные результаты о структуре графов. В этой работе мы проанализируем данные инварианты и их связь для всех графов до 11 вершин, попробуем выявить закономерности и составить галерею особенных графов.

1 Теоретическая часть

Расстоянием $d(u, v)$ между вершинами u и vназывается длина кратчайшего пути между вершинами u и v. Очевидно, что этот путь будет цепью. Любой путь между вершинами u и v длины $d(u, v)$ называется геодезическим.

Геодезический граф — граф, в котором для любых двух вершин существует единственная соединяющая их геодезическая цепь.

Все деревья являются геодезическими графами, так как любые две вершины дерева соединены единственной цепью. Также геодезическими графами являются полные графы и циклы нечётной длины.

Геодезическое множество графа G — множество его вершин S такое, что каждая вершина графа G принадлежит какой-либо геодезической, соединяющей пару вершин из S.

Минимальным геодезическое множество называется, если никакое его собственное подмножество не является геодезическим множеством.

Геодезическое число $g(G)$ графа G — минимальная мощность его геодезического множества, а сами такие множества называются наименьшими геодезическими множествами. В некоторых работах вместо термина «геодезическое число» используется термин «число геодоминирования».

Слабое геодезическое (геодоминирующее) множество графа G — множество его вершин S такое, что каждая вершина графа G слабо геодоминируется некоторой парой вершин из S.

Слабое геодезическое число $wg(G)$ графа G — минимальная мощность его слабого геодезического множества, а сами такие множества называются наименьшими слабыми геодезическими множествами. Очевидно, что $wg(G) \geq g(G)$, для полного графа $K_n : wg(K_n) = g(K_n) = n$ [**?**], [**?**].

Доминирующее множество — это такое подмножество вершин $D \subseteq V$, что любая вершина не из D смежна хотя бы одной вершине из D. Говорят, что вершины v и u доминируют друг друга, если в графе есть ребро $\{v, u\}$ или $v = u$[1].

Несократимое множество D для графа G — множество вершин, где для каждой вершины $v \in D$ выполняется условие $N[v] - N[S - \{v\}] \neq \varnothing$. Другими словами, для любой вершины $v \in D$ или среди вершин графа не из D существует вершина, которая доминируется вершиной v и не доминируется никакой другой вершиной из D, или в S нет вершин, смежных с v. Каждое минимальное доминирующее множество является несократимым.

Число несократимости $ir(G)$ для графа G — минимальная мощность его максимального несократимого множества [**?**].

2 Алгоритмы

2.1 Нахождение слабого геодезического числа

Для нахождения вычисления слабого геодезического числа графа $wg(G)$ нужно найти такое минимальное множество вершин S, чтобы все вершины графа лежали на кратчайших путях между некоторыми парами вершин из множества S.

Для начала пронумеруем все вершины графа, и с помощью BFS вычислим кратчайшие расстояния от текущей до всех остальных. Так мы узнаем, какие вершины лежат на кратчайших путях между другими вершинами.

Для каждой пары вершин (i, j) рассмотрим и сохраним множество всех таких вершин k, что $dist[i][k] + dist[k][j] = dist[i][j]$ (т.е. лежат на кратчайшем пути).

Далее будем перебирать все возможные подмножества вершин графа (начиная с подмножеств размером 2). Для каждого будем вычислять объединение всех вершин, лежащих на кратчайших путях между его элементами. Если объединение покрывает все вершины графа, то размер этого множества будет являться выходом алгоритма. Если ни одно из меньших подмножеств не покрывает граф полностью, то выходом алгоритма будет количество вершин.

Алгоритм вычисления слабого геодезического числа графа

Входные данные: связный граф $G = (V, E)$.

Выходные данные: слабое геодезическое число $wg(G)$.

<u>Шаг 1</u>. Пусть $n = |V|$. Если $n = 0$, вернуть $wg(G) = 0$.

<u>Шаг 2</u>. Пронумеруем вершины $v_i \in V$, где $i \in \{0, \ldots, n-1\}$.

<u>Шаг 3</u>. Вычислим матрицу кратчайших расстояний $dist[i][j]$ между всеми парами вершин графа G. Для каждой вершины $u \in V$ инициализируем $dist[u][u] = 0$, запускаем BFS из u, далее обновляем расстояния до остальных вершин.

<u>Шаг 4</u>. Для каждой пары (i, j), где $i < j$, посчитаем расстояние $d_{ij} = dist[i][j]$. Множество P_{ij} всех вершин k таких, что $dist[i][k] + dist[k][j] = d_{ij}$, закодируем битовой маской.

<u>Шаг 5</u>. Зададим битовую маску $full = (1 \ll n) - 1$, соответствующую покрытию всех вершин графа.

<u>Шаг 6</u>. Для каждого $S_i \subseteq V$, где $i \in \{2, \ldots, n\}$: вычислим объединение

всех «геодезических областей» пар $(u, v) \subseteq S_i$. Если объединение покрывает все вершины (битовая маска равна $full$), возвращаем $wg(G) = |S_i|$, иначе возвращаем $wg(G) = n$.

2.2 Нахождение числа несократимости

Чтобы найти число несократимости $ir(G)$, нужно исследовать все возможные подмножества вершин и вычислить наименьший размер такого максимального S, в котором ни одна вершина не является соседом с другой вершинами из S.

Начнем с проверки, является ли множество пустым — в этом случае число несократимости графа будет равно нулю, иначе будем перебирать все возможные подмножества множества вершин V и каждое подмножество будем проверять на несократимость. Множество считается несократимым, если для каждой его вершины v существует хотя бы одна вершина u в том же подмножестве, такая что $u \neq v$ и u не смежна с v. Если вершины такое множество пусто, подмножество не будет являться несократимым.

Все выявленные несократимые подмножества будем проверять на максимальность. Для этого к каждому множеству будем поочерёдно добавлять все вершины, которые не входят в него, а далее проверять остаётся ли оно все еще несократимым. Если остается, тогда исходное множество является максимальным.

Среди всех найденных максимальных несократимых множеств выбираем то, которое имеет наименьший размер. Это и будет числом несократимости графа. Если ни одно множество не удовлетворяет условиям несократимости и максимальности, тогда $ir(G) = 1$.

Алгоритм вычисления числа несократимости графа

Входные данные: $G = (V, E)$.

Выходные данные: число несократимости $ir(G)$.

<u>Шаг 1</u>. Сначала проверим $V \neq \emptyset$, иначе вернем $ir(G) = 0$.

<u>Шаг 2</u>. Определим $min\_size \leftarrow \infty$.

<u>Шаг 3</u>. Для $S_i \subseteq V$, где $i \in \{0, \ldots, n\}$, рассмотрим вершины $v_k \in S_i$, где $k \subseteq |S_i|$, и вычислим «множество несоседей» $pn(v_k, S_i)$ для каждой. Если для v_k множество $pn(v_k, S_i) = \emptyset$, то S_i не несократимое.

Шаг 4. Из выборки несократимых S_i, выбираем максимальное таким образом:

1. Для каждой вершины $w \in V \setminus S$ сформируем $S' = S \cup \{w\}$.
2. Проверим несократимость S' (аналогично шагу 4).
3. Если хотя бы одно такое S' несократимо, то S не максимально.

Шаг 5. Если S несократимо и максимально, обновим значение минимального множества : $min\_size = \min(min\_size,\ |S|)$.

Шаг 6. Если $min\_size = \infty$, возвращаем $ir(G) = 1$, иначе $ir(G) = min\_size$.

3 Исследовательская часть

3.1 Экспериментальная среда

Генерация всех графов в формате graph6 [?] с заданным числом вершин была произведена с помощью консольной программы geng из пакета nauti [?], результаты генерации были записаны в файл, пример использования

```
for i in$(seq 1 11)
do
  ./geng -c $i > res$i
end
```

Для вычисления самих инвариантов была написана программа на языке Python. Сначала происходит чтение графа из ранее сгенерированного файла. Обработка идёт параллельно т.к. используется модуль `multiprocessing`, что позволяет задействовать несколько процессоров сразу.

Каждый граф превращается в объект библиотеки NetworkX [?]. Затем для него последовательно вычисляются инварианты. Пока программа работает, она собирает статистику — сколько раз встретилась каждая пара значений этих двух инвариантов, полученные данные записываются в результирующую матрицу. В конце анализа для каждого файла программа выводит на экран и сохраняет в файл матрицу распределения значений, общее количество успешно обработанных графов, число ошибок и общее время работы.

Вычисления были произведены на персональном компьютере с данными характеристиками:

1. Операционная система: Linux Ubuntu;
2. Процессор: AMD Ryzen 5 2600 Six-Core Processor (12 ядер);
3. Оперативная память: 32 ГБ;
4. Тип системы: x64;

3.2 Результаты исследования

На таблице можно увидеть, как соотносятся число вершин и число графов к времени работы программы.

Число вершин	Число графов	Время работы
1	1	0.01 с
2	1	0.05 с
3	2	0.3 с
4	6	0.3 с
5	21	0.3 с
6	112	0.5 с
7	853	0.7 с
8	11117	2.23 с
9	261080	1.2 мин
10	11716571	2 ч
11	1006700565	16 дней

Таблица 1 – Время работы программы

На следующих таблицах отражены полученные в ходе вычислений результаты для графов до 11 вершин.

$wg(G)$ \ $ir(G)$	1
2	1

Таблица 2 – Количество 2-вершинных графов со значениями заданных инвариантов.

$wg(G)$ \ $ir(G)$	2	3
1	1	0
2	0	1

Таблица 3 – Количество 3-вершинных графов со значениями заданных инвариантов.

$wg(G)$ \ $ir(G)$	2	3	4
1	1	2	0
2	2	0	0
3	0	0	1

Таблица 3 – Количество 4-вершинных графов со значениями заданных инвариантов.

$wg(G)$ \ $ir(G)$	2	3	4	5
1	3	4	4	0
2	2	5	0	0
3	2	0	0	0
4	0	0	0	1

Таблица 4 – Количество 5-вершинных графов со значениями заданных инвариантов.

$wg(G)$ \ $ir(G)$	2	3	4	5	6
1	9	26	15	6	0
2	14	17	8	0	0
3	3	10	0	0	0
4	3	0	0	0	0
5	0	0	0	0	1

Таблица 5 – Количество 6-вершинных графов со значениями заданных инвариантов.

$wg(G)$ \ $ir(G)$	2	3	4	5	6	7
1	39	171	126	39	10	0
2	69	173	76	14	0	0
3	11	84	16	0	0	0
4	7	14	0	0	0	0
5	3	0	0	0	0	0
6	0	0	0	0	0	1

Таблица 6 – Количество 7-вершинных графов со значениями заданных инвариантов.

$wg(G)$ \ $ir(G)$	2	3	4	5	6	7	8
1	219	1601	1593	473	94	14	0
2	608	2363	1529	211	24	0	0
3	268	1150	560	26	0	0	0
4	39	255	51	0	0	0	0
5	13	21	0	0	0	0	0
6	4	0	0	0	0	0	0
7	0	0	0	0	0	0	1

Таблица 7 – Количество 8-вершинных графов со значениями заданных инвариантов.

$wg(G)$ \ $ir(G)$	2	3	4	5	6	7	8	9
1	2055	21540	32384	9259	1554	201	21	0
2	7400	49476	48283	7406	574	37	0	0
3	7347	31324	26424	1268	47	0	0	0
4	332	9792	3257	48	0	0	0	0
5	111	799	82	0	0	0	0	0
6	21	33	0	0	0	0	0	0
7	4	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	1

Таблица 8 – Количество 9-вершинных графов со значениями заданных инвариантов.

$wg(G)$ \ $ir(G)$	2	3	4	5	6	7	8	9	10
1	33819	475821	1084140	330706	43520	4584	410	29	0
2	158786	1623761	2362193	453457	29823	1385	58	0	0
3	267603	1477177	2012307	141949	3234	74	0	0	0
4	55005	529624	519695	4706	75	0	0	0	0
5	2004	79799	17547	69	0	0	0	0	0
6	345	2586	193	0	0	0	0	0	0
7	34	47	0	0	0	0	0	0	0
8	5	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	1

Таблица 9 – Количество 10-вершинных графов со значениями заданных инвариантов.

$wg(G)$ \ $ir(G)$	2	3	4	5	6
1	2 905 766	40 883 063	93 150 496	28 414 623	3 739 286
2	13 643 068	139 515 329	202 962 217	38 961 524	2 562 425
3	22 992 744	126 920 671	172 899 628	12 196 414	277 869
4	4 726 090	45 505 876	44 652 765	404 345	6 444
5	172 186	6 856 418	1 507 658	5 929	56
6	29 643	222 192	16 583	27	0
7	2 921	4 038	43	0	0
8	98	34	0	0	0
9	5	0	0	0	0
10	0	0	0	0	0

Таблица 10.1 – Количество 11-вершинных графов со значениями заданных инвариантов.

$wg(G)$ \ $ir(G)$	7	8	9	10	11
1	393 862	35 228	976	46	0
2	119 001	4 983	82	0	0
3	6 358	75	0	0	0
4	32	0	0	0	0
5	0	0	0	0	0
6	0	0	0	0	0
7	0	0	0	0	0
8	0	0	0	0	0
9	0	0	0	0	0
10	0	0	0	0	1

Таблица 10.2 – Количество 11-вершинных графов со значениями заданных инвариантов.

3.3 Построение гипотез

Общие наблюдения

1. С ростом числа вершин максимальные значения для слабого геодезичсеского числа сосредотачиваются в диапазоне от 1 до 3, числа несократимости — от 3 до 4;
2. Значения вне центральных областей таблицы стремятся к нулю, что видно по резкому уменьшению числа графов с увеличением этих параметров;
3. Матрица на побочной диагонали и под ней заполнена нулями, за исключением одного элемнта. Графы с большими значениями инфвариант уникальны, их структура близка к полным или сильно связанным графам.

Гипотеза 1.

Для множества графов с n вершинами, максимальными возможными значениями слабого геодезического числа и числа несократимости будут $wg(G) = n - 1$ и $ir(G) = n$, причем такой граф будет только один. Визуально закономерность можно увидеть на рисунках **??**, **??** , **??**.

$wg(G)$ \ $ir(G)$	2	3	4	5
1	3	4	4	0
2	2	5	0	0
3	2	0	0	0
4	0	0	0	1

Рисунок 1 – Матрица отношений инвариантов для $n = 5$

$wg(G)$ \ $ir(G)$	2	3	4	5	6	7	8
1	219	1601	1593	473	94	14	0
2	608	2363	1529	211	24	0	0
3	268	1150	560	26	0	0	0
4	39	255	51	0	0	0	0
5	13	21	0	0	0	0	0
6	4	0	0	0	0	0	0
7	0	0	0	0	0	0	1

Рисунок 2 – Матрица отношений инвариантов для $n = 8$

$ir(G)$ / $wg(G)$	2	3	4	5	6	7	8	9	10
1	33819	475821	1084140	330706	43520	4584	410	29	0
2	158786	1623761	2362193	453457	29823	1385	58	0	0
3	267603	1477177	2012307	141949	3234	74	0	0	0
4	55005	529624	519695	4706	75	0	0	0	0
5	2004	79799	17547	69	0	0	0	0	0
6	345	2586	193	0	0	0	0	0	0
7	34	47	0	0	0	0	0	0	0
8	5	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	1

Рисунок 3 – Матрица отношений инвариантов для $n = 10$

Гипотеза 2.

Для множества графов с n вершинами, не существует графа G который будет обладать равными инвариантами: $ir(G) = wg(G) = K$, со значением $K > \left\lfloor \frac{n}{2} \right\rfloor$. Визуально закономерность видна на рисунках **??**, **??** , **??**.

$ir(G)$ / $wg(G)$	2	3	4	5
1	3	4	4	0
2	2	5	0	0
3	2	0	0	0
4	0	0	0	1

Рисунок 4 – Матрица отношений инвариантов для $n = 5$

$wg(G)$ \ $ir(G)$	2	3	4	5	6	7	8
1	219	1601	1593	473	94	14	0
2	608	2363	1529	211	24	0	0
3	268	1150	560	26	0	0	0
4	39	255	51	0	0	0	0
5	13	21	0	0	0	0	0
6	4	0	0	0	0	0	0
7	0	0	0	0	0	0	1

Рисунок 5 – Матрица отношений инвариантов для $n = 8$

$wg(G)$ \ $ir(G)$	2	3	4	5	6	7	8	9	10
1	33819	475821	1084140	330706	43520	4584	410	29	0
2	158786	1623761	2362193	453457	29823	1385	58	0	0
3	267603	1477177	2012307	141949	3234	74	0	0	0
4	55005	529624	519695	4706	75	0	0	0	0
5	2004	79799	17547	69	0	0	0	0	0
6	345	2586	193	0	0	0	0	0	0
7	34	47	0	0	0	0	0	0	0
8	5	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	1

Рисунок 6 – Матрица отношений инвариантов для $n = 10$

Гипотеза 3.

Для множества графов с n вершинами, будет существовать ровно $C = \left\lfloor \frac{n}{2} \right\rfloor$ графов со слабым геодезическим числом $wg(G) = n - 2$ и числом несократимости $ir(G) = 2$. Визуально закономерность видна на рисунках **??**, **??** , **??**.

$wg(G)$ \ $ir(G)$	2	3	4	5
1	3	4	4	0
2	2	5	0	0
3	2	0	0	0
4	0	0	0	1

Рисунок 7 – Матрица отношений инвариантов для $n = 5$

$wg(G)$ \ $ir(G)$	2	3	4	5	6	7	8
1	219	1601	1593	473	94	14	0
2	608	2363	1529	211	24	0	0
3	268	1150	560	26	0	0	0
4	39	255	51	0	0	0	0
5	13	21	0	0	0	0	0
6	4	0	0	0	0	0	0
7	0	0	0	0	0	0	1

Рисунок 8 – Матрица отношений инвариантов для $n = 8$

$wg(G)$ \ $ir(G)$	2	3	4	5	6	7	8	9	10
1	33819	475821	1084140	330706	43520	4584	410	29	0
2	158786	1623761	2362193	453457	29823	1385	58	0	0
3	267603	1477177	2012307	141949	3234	74	0	0	0
4	55005	529624	519695	4706	75	0	0	0	0
5	2004	79799	17547	69	0	0	0	0	0
6	345	2586	193	0	0	0	0	0	0
7	34	47	0	0	0	0	0	0	0
8	5	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	1

Рисунок 9 – Матрица отношений инвариантов для $n = 10$

4 Каталог графов

На рисунках 10-19 представлены графы соответствующие гипотезе 1.

Рисунок 10 – Граф $wg(G) = 2, ir(G) = 1$

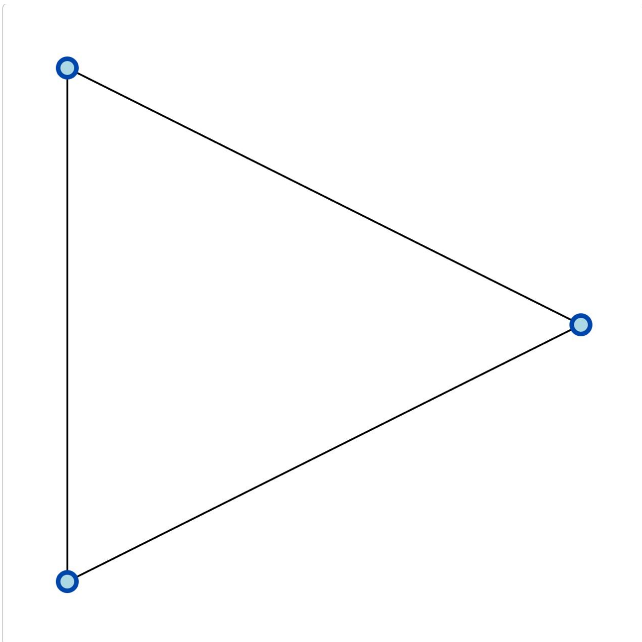

Рисунок 11 – Граф $wg(G) = 2, ir(G) = 3$

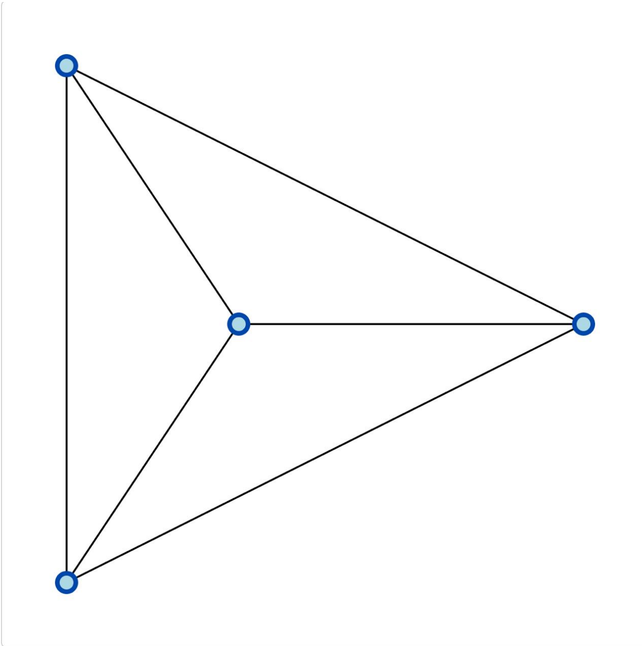

Рисунок 12 – Граф $wg(G) = 3, ir(G) = 4$

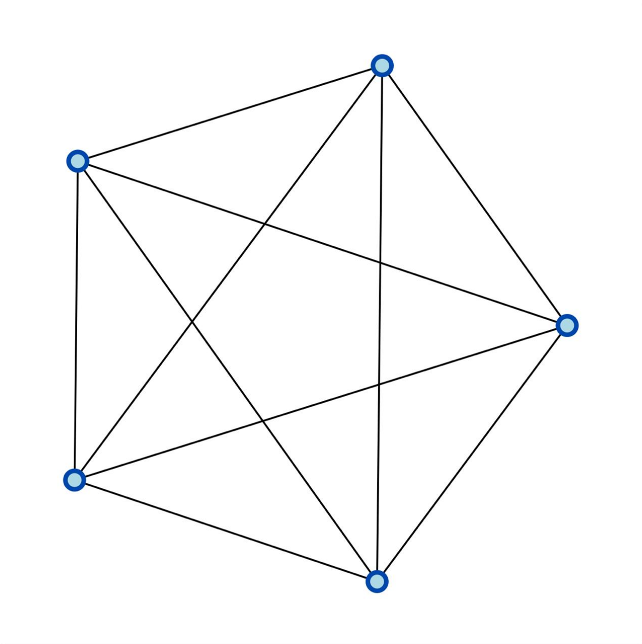

Рисунок 13 – Граф $wg(G) = 4, ir(G) = 5$

На рисунках 20-26 представлены графы соответствующие гипотезе 2.

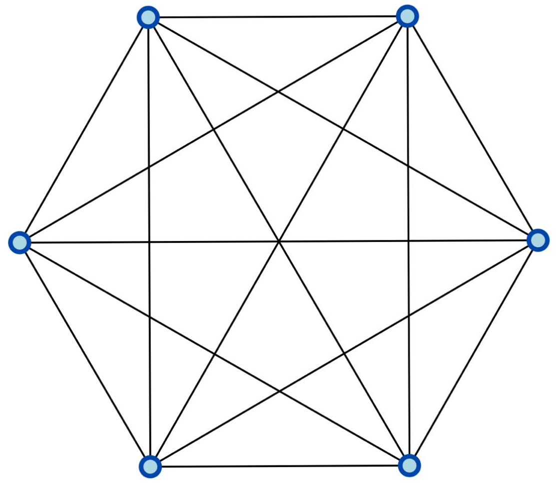

Рисунок 14 – Граф $wg(G) = 5, ir(G) = 6$

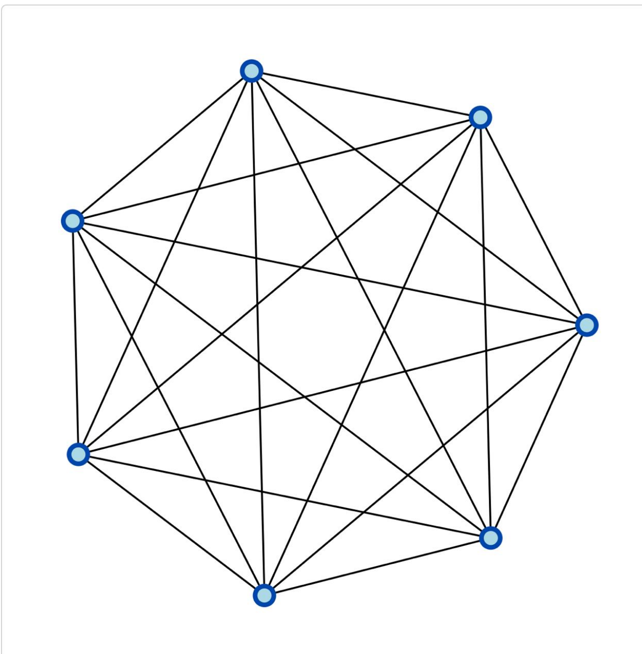

Рисунок 15 – Граф $wg(G) = 6, ir(G) = 7$

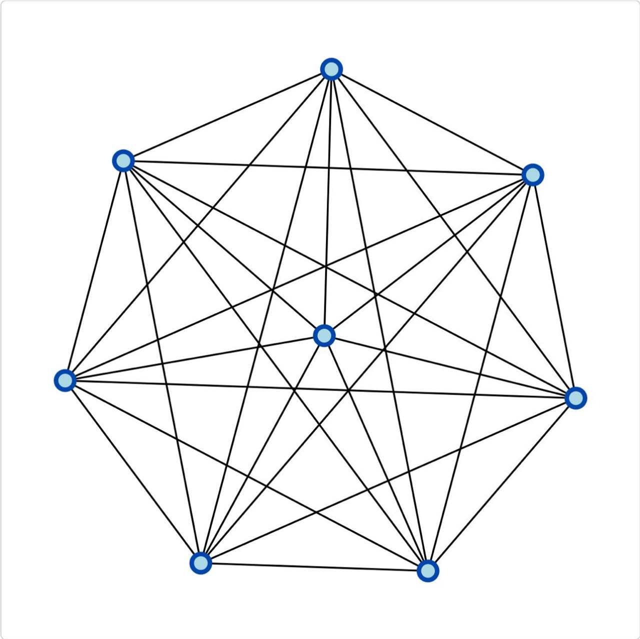

Рисунок 16 – Граф $wg(G) = 7, ir(G) = 8$

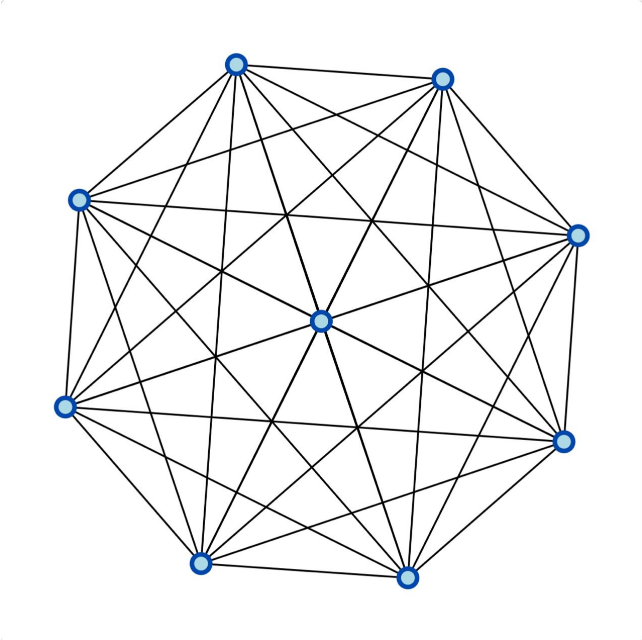

Рисунок 17 – Граф $wg(G) = 8, ir(G) = 9$

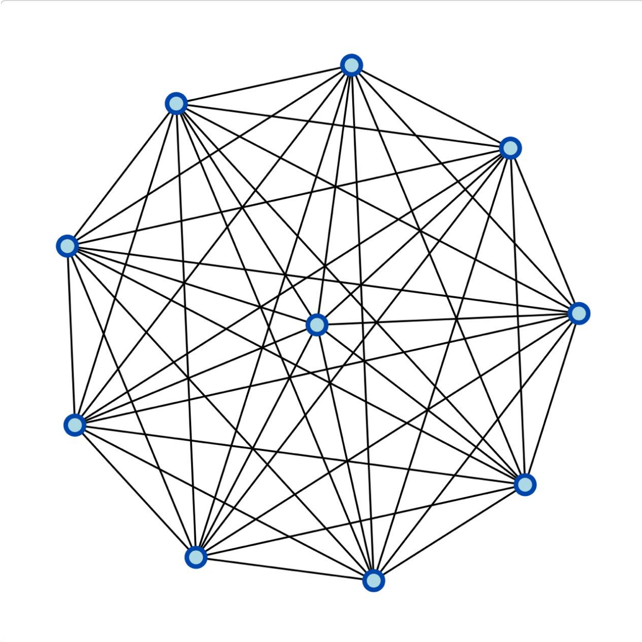

Рисунок 18 – Граф $wg(G) = 9, ir(G) = 10$

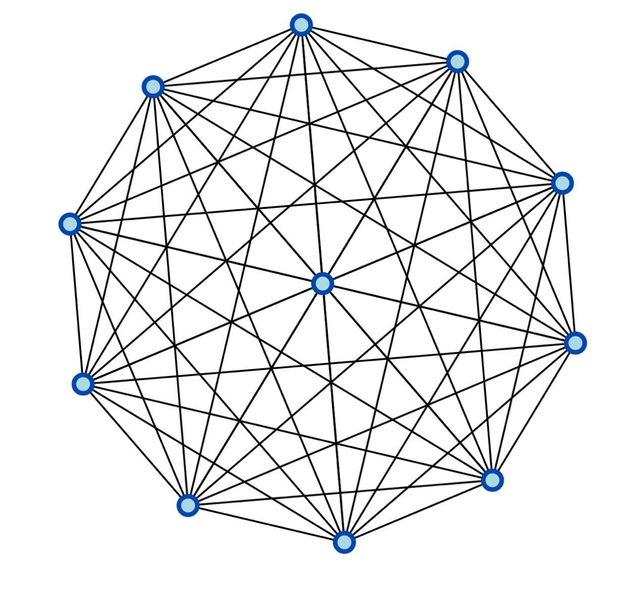

Рисунок 19 – Граф $wg(G) = 10, ir(G) = 11$

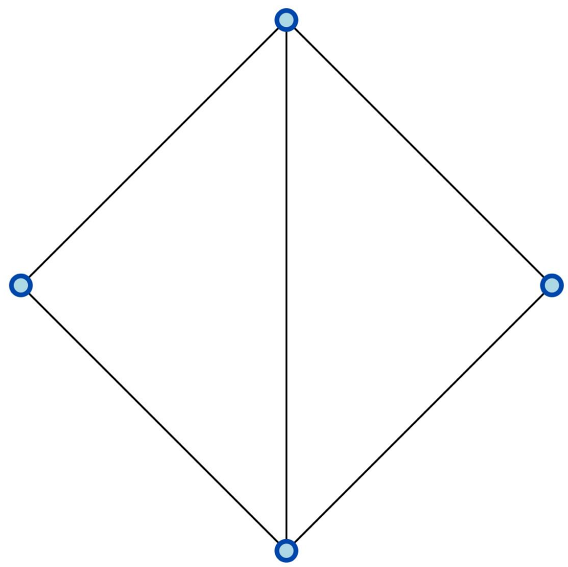

Рисунок 20 – Граф $wg(G) = ir(G) = 2$

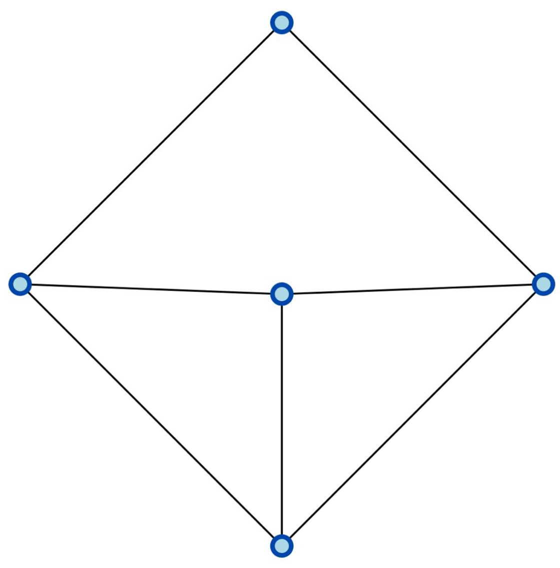

Рисунок 21 – Граф $wg(G) = ir(G) = 2$

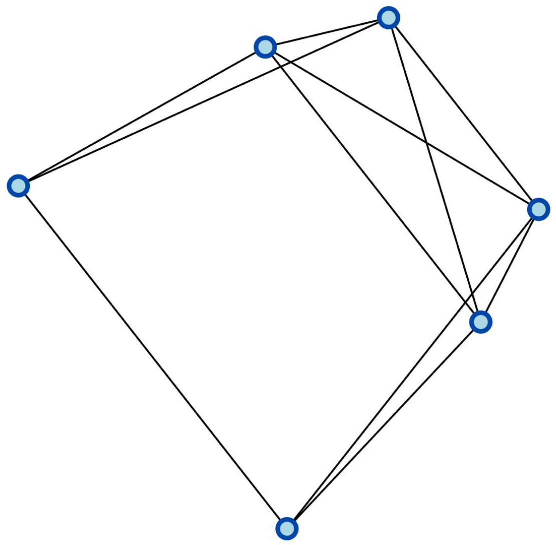

Рисунок 22 – Граф $wg(G) = ir(G) = 3$

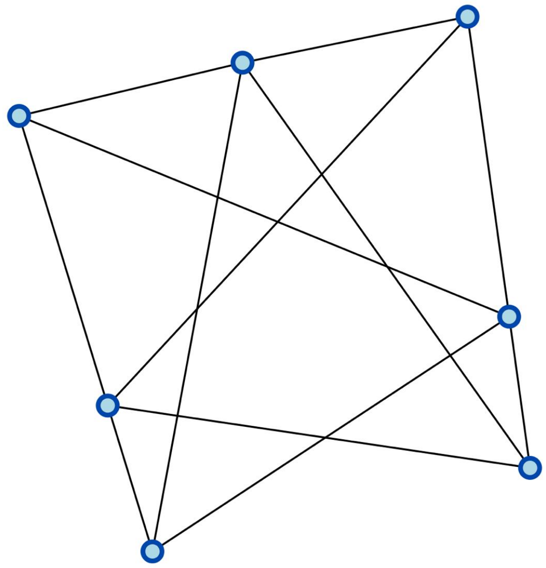

Рисунок 23 – Граф $wg(G) = ir(G) = 3$

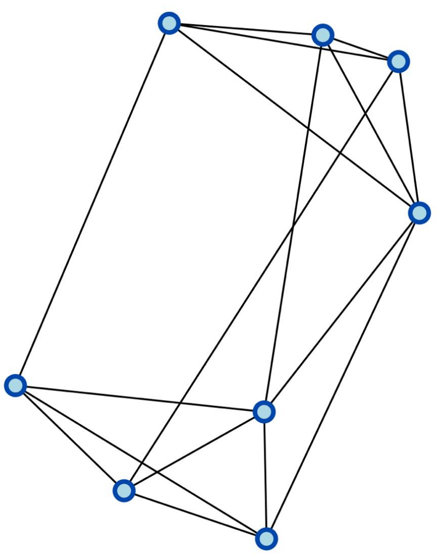

Рисунок 24 – Граф $wg(G) = ir(G) = 4$

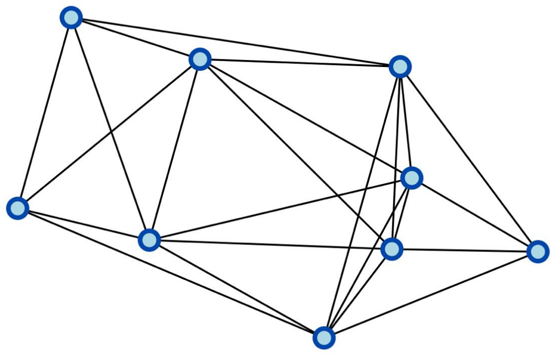

Рисунок 25 – Граф $wg(G) = ir(G) = 4$

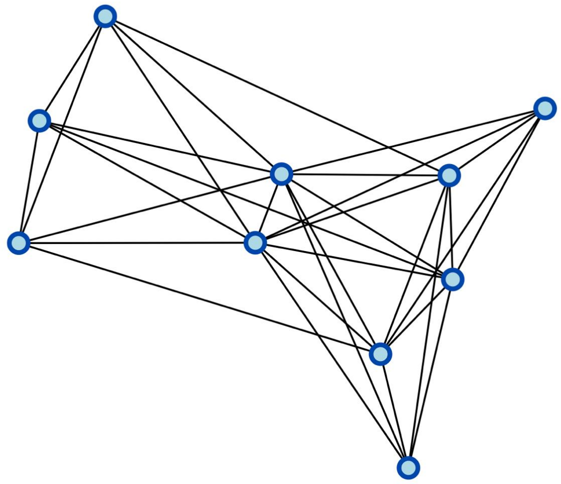

Рисунок 26 – Граф $wg(G) = ir(G) = 5$

ЗАКЛЮЧЕНИЕ

В данной работе было проведено исследование инвариантов графов, в ходе которого был разработан алгоритм для нахождения слабого геодезического числа и числа несократимости графа. В рамках вычислительных экспериментов изучены связные графы с количеством вершин до 11, что позволило получить важные статистические данные и выявить интересные закономерности в распределении рассматриваемых инвариантов.

Анализ полученных экспериментальных данных привел к формулированию следующих гипотез. В частности гипотеза о несуществовании графов с равными значениями инвариантов, начиная с определенного, об особенностях графов с максимальными значениями инвариантов, а также о количестве графов для минимального значения числа нескоратимости и максимального значения слабого геодезического числа.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Rad, N.J. Strong geodomination in graphs [Текст] / Rad, N.J. и Mojdeh, D.A. // Opuscula Mathematica. — 2008. — Т. 28, № 3. — С. 279–285.

2 Rad, N.J. Weak geodomination in graphs [Текст] // Analele Stiintifice ale Universitatii Ovidius Constanta. — 2008. — Т. 16, № 1. — С. 127–134.

3 Cockayne, E.J. Properties of hereditary hypergraphs and middle graphs [Текст] / Cockayne, E.J., Hedetniemi, S.T. и Miller, D.J. // Canadian Mathematical Bulletin. — 1978. — Т. 21, № 4. — С. 461–468.

4 Description of graph6, sparse6 and digraph6 encodings [Электронный ресурс онлайн]. — [Б. м. : б. и.]. — Режим доступа: https://users.cecs.anu.edu.au/~bdm/data/formats.txt (дата обращения: 20.06.2025).

5 Nauty and Traces [Электронный ресурс онлайн]. — [Б. м. : б. и.]. — Режим доступа: https://pallini.di.uniroma1.it/ (дата обращения: 20.06.2025).

6 NetworkX [Электронный ресурс онлайн]. — [Б. м. : б. и.]. — Режим доступа: https://networkx.org/ (дата обращения: 20.06.2025).

ПРИЛОЖЕНИЕ А

Листинг main.py

```
import networkx as nx
import time
import itertools
from multiprocessing import Pool
import os
from collections import defaultdict
import itertools

def weak_geodetic_number(graph):
    nodes = list(graph.nodes())
    n = len(nodes)
    covered = set()
    S = set()

    shortest_paths_dict = defaultdict(dict)
    for u in nodes:
        for v in nodes:
            if u != v:
                try:
                    paths = list(nx.all_shortest_paths(graph, u, v))
                    shortest_paths_dict[u][v] = paths[0] if len(paths) == 1 else None
                except nx.NetworkXNoPath:
                    shortest_paths_dict[u][v] = None
            else:
                shortest_paths_dict[u][v] = [u]

    all_pairs = list(itertools.combinations_with_replacement(nodes, 2))

    while len(covered) < n:
        best_gain = 0
        best_pair = None
        best_new_covered = set()

        for u, v in all_pairs:
            if u == v:
                new_covered = {u}
            else:
                path = shortest_paths_dict[u][v] or shortest_paths_dict[v][u]
                if path:
                    new_covered = set(path)
                    new_covered.add(u)
                    new_covered.add(v)
                else:
                    new_covered = {u, v}

            gain = len(new_covered - covered)
```

```
            if gain > best_gain:
                best_gain = gain
                best_pair = (u, v)
                best_new_covered = new_covered

                if len(covered | new_covered) == n:
                    break

        if best_pair is None:
            break

        S.update(best_pair)
        covered.update(best_new_covered)

    return len(S)

def is_irreducible(S, graph):
    if not S:
        return True
    for v in S:
        pn = [u for u in S if u != v and not graph.has_edge(u, v)]
        if not pn:
            return False
    return True

def is_maximal_irreducible(S, all_nodes, graph):
    for w in all_nodes - S:
        extended = S | {w}
        if is_irreducible(extended, graph):
            return False
    return True


def connectivity_number(graph):
    V = set(graph.nodes)

    if not V:
        return 0

    min_size = float('inf')

    for r in range(len(V)+1):
        for subset in itertools.combinations(V, r):
            S = set(subset)
            if is_irreducible(S, graph) and is_maximal_irreducible(S, V, graph):
                min_size = min(min_size, len(S))

    return 1 if min_size == float('inf') else min_size

```

```
def process_single_graph(index, graph_string):
    """Обработка одного графа для параллелизации."""
    try:
        if not isinstance(graph_string, str):
            return {"index": index, "error": f"Неверный тип graph_string:
            ↪ {type(graph_string)}"}

        graph_string = graph_string.strip()
        if not graph_string:
            return {"index": index, "error": "Пустая строка Graph6"}

        graph = nx.from_graph6_bytes(graph_string.encode())

        start_time = time.time()
        wg_number = weak_geodetic_number(graph)
        wg_time = time.time() - start_time
        connectivity = connectivity_number(graph)
        conn_time = time.time() - start_time

        return {
            "index": index,
            "graph": graph_string,
            "weak_geodetic_number": wg_number,
            "weak_geodetic_time": wg_time,
            "connectivity_number": connectivity,
            "connectivity_time": conn_time
        }
    except Exception as e:
        return {"index": index, "error": f"Ошибка обработки графа: {str(e)}"}


def process_single_graph_star(args):
    return process_single_graph(*args)

def process_graph6_file(file_path):
    """Обработка файла Graph6 с оптимизированной параллелизацией"""
    results = []
    start_time_total = time.time()

    if not os.path.exists(file_path):
        print(f"Ошибка: Файл {file_path} не существует")
        return results, 0

    try:
        with open(file_path, 'r') as file:
            graph_strings = [line.strip() for line in file if line.strip()]
    except Exception as e:
```

```
            print(f"Ошибка чтения файла {file_path}: {str(e)}")
            return results, 0

    if not graph_strings:
        print(f"Ошибка: Файл {file_path} пуст или содержит только пустые строки")
        return results, 0


    num_processes = min(os.cpu_count() or 8, 10, len(graph_strings))
    print(f"Используется {num_processes} процессов")


    chunk_size = max(1, len(graph_strings) // (num_processes * 4))

    try:
        with Pool(processes=num_processes) as pool:

            results = list(pool.imap_unordered(
                process_single_graph_star,
                enumerate(graph_strings),
                chunksize=chunk_size
            ))
    except Exception as e:
        print(f"Ошибка при параллельной обработке: {str(e)}")
        return results, time.time() - start_time_total

    total_time = time.time() - start_time_total
    return results, total_time

def main():
    start_time_global = time.time()
    total_graphs = 0
    total_errors = 0
    result = [[0 for _ in range(11)] for _ in range(11)]
    with open(input_file, 'r') as f:
        for i in range(0, 10):
            for j in range(0, 10):
                input_file = f"./graph/outt10_00{i}{j}"
                input_file = f"./graph/res7.txt"
                print(f"\nОбработка файла: {input_file}")
                results, total_time = process_graph6_file(input_file)
                time_total1 = 0
                time_total2 = 0
                real_w_g = {}
                real_c_n = {}
                count = 0
                errors = 0

                for result in results:
```

```python
                    if "error" in result:
                        errors += 1
                        total_errors += 1
                        continue

                    count += 1
                    time_total1 += result['weak_geodetic_time']
                    time_total2 += result['connectivity_time']

                    w_g = result['weak_geodetic_number']
                    c_n = result['connectivity_number']

                    real_w_g[w_g] = real_w_g.get(w_g, 0) + 1
                    real_c_n[c_n] = real_c_n.get(c_n, 0) + 1
                    if w_g < 11 and c_n < 11:
                        result[w_g][c_n] += 1
                total_graphs += count

                print("\nМатрица [wg][connectivity]:")
                print("     " + " ".join([f"{i:>4}" for i in range(11)]))
                for i in range(11):
                    row = " ".join([f"{result[i][j]:>4}" for j in range(11)])
                    print(f"{i:>3}: {row}")

                f.write(f"Номер итерации {i}{j}")
                f.write(f"Всего графов: {count}")
                f.write(f"Ошибок: {errors}")
                f.write(f"Суммарное время слабого геодезического числа: {time_total1:.4f} сек")
                f.write(f"Распределение по значению инварианта:")
                f.write(f"1. Слабое геодезическое число: {real_w_g}")
                f.write(f"Общее время обработки файла: {total_time:.4f} сек")

                total_time_global = time.time() - start_time_global
                print("\nИтоговая статистика:")
                print(f"Всего обработано графов: {total_graphs}")
                print(f"Общее количество ошибок: {total_errors}")
                print(f"Общее время выполнения: {total_time_global:.4f} сек")

if __name__ == "__main__":
    main()
```