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

ЧИСЛО НЕЗАВИСИМОГО ДОМИНИРОВАНИЯ И ЧИСЛО СОВЕРШЕННОГО ГЕОДОМИНИРОВАНИЯ

НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА

студента 5 курса 531 группы
направления 10.05.01 — Компьютерная безопасность
факультета КНиИТ
Стаина Романа Игоревича

Проверил
д. ф.-м. н., доцент _____ М. Б. Абросимов

Саратов 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3
1 Алгоритмы 4
2 Результаты исследования 5
ЗАКЛЮЧЕНИЕ 7
Приложение А Листинг `main.py` 8

ВВЕДЕНИЕ

Введение

1 Алгоритмы

Алгоритмы

2 Результаты исследования

g_p \ γ_i	1	2
1	0	0
2	1	0

Таблица 1 – Количество 2-вершинных графов, имеющих заданные g_p и γ_i

g_p \ γ_i	1	2	3
1	0	0	0
2	1	0	0
3	1	0	0

Таблица 2 – Количество 3-вершинных графов, имеющих заданные g_p и γ_i

g_p \ γ_i	1	2	3	4
1	0	0	0	0
2	1	2	0	0
3	0	0	0	0
4	3	0	0	0

Таблица 3 – Количество 4-вершинных графов, имеющих заданные g_p и γ_i

g_p \ γ_i	1	2	3	4	5
1	0	0	0	0	0
2	2	3	0	0	0
3	0	2	0	0	0
4	2	1	0	0	0
5	7	4	0	0	0

Таблица 4 – Количество 5-вершинных графов, имеющих заданные g_p и γ_i

g_p \ γ_i	1	2	3	4	5	6
1	0	0	0	0	0	0
2	4	11	0	0	0	0
3	0	8	0	0	0	0
4	8	14	0	0	0	0
5	7	11	0	0	0	0
6	15	28	4	0	0	0

Таблица 5 – Количество 6-вершинных графов, имеющих заданные g_p и γ_i

g_p \ γ_i	1	2	3	4	5	6	7
1	0	0	0	0	0	0	0
2	11	23	1	0	0	0	0
3	0	70	2	0	0	0	0
4	24	122	8	0	0	0	0
5	53	108	17	0	0	0	0
6	26	100	14	0	0	0	0
7	42	192	40	0	0	0	0

Таблица 6 – Количество 7-вершинных графов, имеющих заданные g_p и γ_i

g_p \ γ_i	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	34	137	5	0	0	0	0	0
3	0	581	4	0	0	0	0	0
4	126	1638	107	2	0	0	0	0
5	314	1489	238	1	0	0	0	0
6	283	1467	314	4	0	0	0	0
7	145	1249	350	0	0	0	0	0
8	142	1865	601	21	0	0	0	0

Таблица 7 – Количество 8-вершинных графов, имеющих заданные g_p и γ_i

g_p \ γ_i	1	2	3	4	5	6	7	8	9
1	0	0	0	0	0	0	0	0	0
2	156	888	1	0	0	0	0	0	0
3	0	8002	182	0	0	0	0	0	0
4	861	31921	1853	13	0	0	0	0	0
5	3422	38055	6310	27	0	0	0	0	0
6	3477	33948	7715	66	0	0	0	0	0
7	2461	29432	9134	144	0	0	0	0	0
8	1210	25377	10380	184	0	0	0	0	0
9	759	31055	13511	535	1	0	0	0	0

Таблица 8 – Количество 9-вершинных графов, имеющих заданные g_p и γ_i

ЗАКЛЮЧЕНИЕ

Заключение

ПРИЛОЖЕНИЕ А

Листинг main.py

```
# число независимого доминирования и число совершенного геодоминирования
from sys import stdin
import networkx as nx
import matplotlib.pyplot as plt
import time
from itertools import combinations
import multiprocessing as mp


def find_maximal_independent_sets(g):
    # 1 Начальная установка
    k, s, q_minus, q_plus = 0, [], [[]], [list(g.nodes)]
    result = []

    def step2(k):
        xk = q_plus[k][0]
        s.append(xk)
        if k >= len(q_minus) - 1:
            q_minus.append([])
        else:
            q_minus[k + 1] = list(set(q_minus[k]) - set(g.adj[xk]))
        t_plus_k = q_plus[k].copy()
        t_plus_k.remove(xk)
        if k >= len(q_plus) - 1:
            q_plus.append(list(set(t_plus_k) - set(g.adj[xk])))
        else:
            q_plus[k + 1] = list(set(t_plus_k) - set(g.adj[xk]))
        k += 1
        # print(f'Шаг 2 \nk = {k}, xk = {xk} \nS = {s} \nQ+ = {q_plus} \nQ- =
        ↪  {q_minus}')
        return xk, k

    def step5(s, k):
        k -= 1
        xk = s[k]
        s.pop()
        q_plus[k].remove(xk)
        q_minus[k].append(xk)
        # print(f'Шаг 5 \nk = {k}, xk = {xk} \nS = {s} \nQ+ = {q_plus} \nQ- =
        ↪  {q_minus}')
        return k

    while len(q_plus[0]) != 0:
        if k != 0:
            # 3 Проверка
```

```
                if xk in q_minus and set(g.adj[xk]) & set(q_plus[k]) == set():
                    k = step5(s, k)
                else:
                    # 4
                    if len(q_plus[k]) == 0:
                        if len(q_minus[k]) == 0:
                            # print('=' * 40)
                            # print(f'{s} к результату')
                            result.append(s.copy())
                            # print('=' * 40)
                            k = step5(s, k)
                        else:
                            k = step5(s, k)
                    else:
                        xk, k = step2(k)
            else:
                xk, k = step2(k)
        return result


def independent_domination_number(g):
    g = nx.from_graph6_bytes(g[0:-1].encode())
    maximal_independent_sets = find_maximal_independent_sets(g)
    maximal_independent_sets.sort(key=len)
    for s in maximal_independent_sets:
        if nx.is_dominating_set(g, s):
            return len(s)


def find_perfect_geodominating_sets(g):
    g = nx.from_graph6_bytes(g[0:-1].encode())
    nodes = set(g.nodes)
    nodes_len = len(nodes)
    all_paths = [[None] * nodes_len for _ in range(nodes_len)]

    for k in range(2, len(nodes)):
        # Всевозможные варианты множества вершин S
        for si in combinations(nodes, k):
            s = set(si)
            v = nodes - s
            # print('S =', si)
            # print('V\\S =', v)

            paths = []
            # Всевозможные кратчайшие пути между вершинами из S
            for a, b in combinations(s, 2):
                if all_paths[a][b] is None:
                    all_paths[a][b] = list(nx.all_shortest_paths(g, a, b))
```

```
92                      for p in all_paths[a][b]:
93                          paths += p[1:-1]
94
95                  # print('Пути', paths)
96                  if len(paths) == 0:
97                      continue
98
99                  flag = True
100                 for node in v:
101                     count = paths.count(node)
102                     # Если какая-то вершина из V\S встречается больше одного раза,
                        ↪ значит
103                     # она геодоминируется несколькими парами вершин из S. Или вершина не
                        ↪ геод-ся вовсе
104                     if count != 1:
105                         # print('No', si)
106                         flag = False
107                         break
108
109                 if flag:
110                     # print('S', si)
111                     # print('V\\S', v)
112                     return len(si)
113             # Если не получилось найти, значит в S должны быть все вершины
114             # print('S', nodes)
115             return nodes_len
116
117
118     if __name__ == '__main__':
119         graphs6 = stdin.readlines()
120         t0 = time.time()
121         g = nx.from_graph6_bytes(graphs6[0][0:-1].encode())
122         num_nodes = len(g.nodes)
123
124         agents = 4
125         chunksize = 4
126         with mp.Pool(processes=agents) as pool:
127             result = pool.map(find_perfect_geodominating_sets, graphs6, chunksize)
128             result2 = pool.map(independent_domination_number, graphs6, chunksize)
129
130         table = [[0] * num_nodes for _ in range(num_nodes)]
131         for a, b in zip(result, result2):
132             table[a - 1][b - 1] += 1
133
134         for row in table:
135             print(row)
136
137
```

```
138             print(f'Время работы: {time.time() - t0} сек.')
139
```