|  |
| --- |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **"МИРЭА – Российский технологический университет"**  **РТУ МИРЭА** |

Институт комплексной безопасности и специального приборостроения

Кафедра КБ-14 «Цифровые технологии обработки данных»

**ОТЧЕТ   
о выполнении лабораторной работы №2**

**«Алгоритмы на графах»**

**по дисциплине   
«Алгоритмы и структуры данных»**

**Вариант № 99**

Выполнил: студент 2 курса

группы \_\_\_\_\_\_\_\_\_\_\_

шифр \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(фио студента)*

Проверил: *\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Москва 2021 г.

**Вариант № 99.**

**Алгоритм:** определить минимальное число красок, которыми можно раскрасить   
граф и вывести пример такой раскраски.

**Способ представления графа:** Список смежности.

**Теория о Графах.**

Граф — это математический объект, который состоит из точек и линий, которые их соединяют. Точки называют вершинами графа, а линии — ребрами. Граф, ребра которого имеют направления, называется ориентированным, если же ребра графа не имеют направления, то такой граф называется неориентированным.



Рисунок 1. – Пример ориентированного графа.

Матрица смежности — это вид представления графа в виде матрицы, когда пересечение столбцов и строк задаёт дуги. Используя матрицу смежности, можно задать вес дуг и ориентацию. Каждая строка и столбец матрицы соответствуют вершинам, номер строки соответствует вершине, из которой выходит дуга, а номер столбца - в какую входит дуга. Пример матрицы смежности графа, изображенного на рисунке 1, представлен на рисунке 2.



Рисунок 2. – Пример матрицы смежности.

**Список смежности** — одна из форм представления графа. Номер строки в списке означает номер вершины, из которой выходят ребра, а номера в строке означают номера вершин смежных с текущей вершиной.

Например список смежности для графа из рисунка 1 может выглядеть так:

1. B G
2. C D
3. D E
4. A
5. B F
6. A B
7. A

Для поиска хроматического числа и раскраски графа отсортируем список вершин по их степени с сохранением номеров.

Далее внешним циклом будем проходить по непокрашенным вершинам и, если такая вершина существует, то создавать новый цвет.

Внутренним циклом по непокрашенным вершинам будем проверять не имеет ли эта вершина связей с вершинами, покрашенными этим цветом, если связей нет, то красим вершину и добавляем ее в список покрашенных этим цветом.

После завершения цикла, добавляем в список покрашенных вершин список покрашенных вершин этим цветом.

На выходе в переменной текущего цвета мы будем иметь хроматическое число нашего графа, а в списке покрашенных вершин будет пример его раскраски.

**Листинг программы.**

from typing import Tuple, List

COLORS = {

    1: "RED",

    2: "GREEN",

    3: "BLUE",

    4: "YELLOW",

    5: "BLACK",

    6: "PURPLE"

}

def read\_adj\_list(file\_path: str):

    with open(file\_path, "r") as f:

        dim = int(f.readline().strip())

        matrix = [[0] \* dim for \_ in range(dim)]

        for i, line in enumerate(f.readlines()):

            for adj in line.split():

                adj = int(adj)

                matrix[i][adj] = 1

    return matrix

def colorize(matrix: List[List[int]]) -> Tuple[int, List[Tuple[int, int]]]:

    v\_power = [(i, sum(row)) for i, row in enumerate(matrix)]

    v\_power.sort(key=lambda tup: -tup[1])

    v\_color = []

    colored = set()

    color = 0

    for v1, \_ in v\_power:

        if v1 in colored:

            continue

        color += 1

        colored\_on\_step = set()

        for v2, \_ in v\_power:

            if v2 in colored:

                continue

            need\_skip = False

            for v3 in colored\_on\_step:

                if matrix[v2][v3]:

                    need\_skip = True

                    break

            if need\_skip:

                continue

            colored\_on\_step.add(v2)

            v\_color.append((v2, color))

        colored.update(colored\_on\_step)

    v\_color.sort(key=lambda tup: tup[0])

    return color, v\_color

if \_\_name\_\_ == '\_\_main\_\_':

    matrix = read\_adj\_list('graph\_task99.txt')

    colors, node\_color = colorize(matrix)

    print(f"Хроматическое число графа: {colors}")

    print("Пример раскраски:")

    for node, color in node\_color:

        print(f"{node}: {COLORS[color]}")

**Скриншот работы программы:**

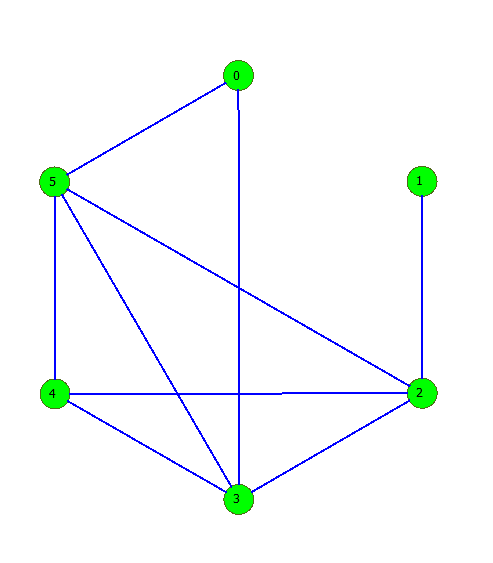


Рисунок 3. – Пример неориентированного ненагруженного графа.

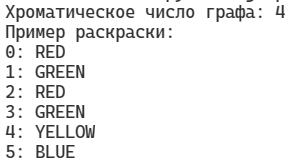
****

Рисунок 4. – Пример работы программы.

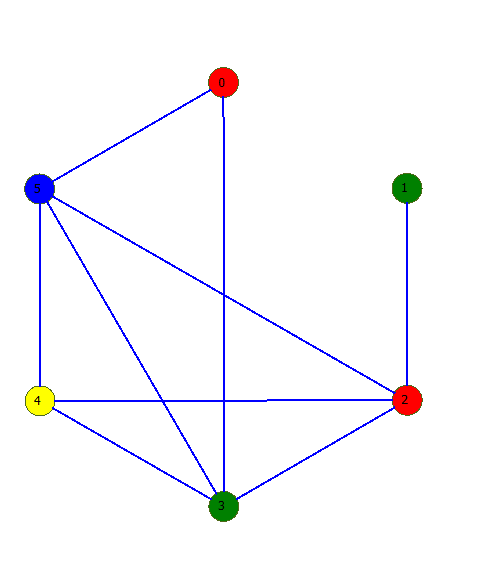


Рисунок 5. Пример графа после раскраски

**Выводы.**

В результате выполнения данной работы были изучены свойства графа, способы его задания и работа с ним. Также был реализован алгоритм для поиска хроматического числа графа и его раскраски.

**Литература:**

1. Алгоритмы: построение и анализ. Кормен Т.Х., Лейзерсон Ч.И.
2. Донец, Г. А. & Шор, Н. З. (1982), Алгебраический подход к проблеме раскраски плоских графов, Наукова думка.
3. Beigel, R. & Eppstein, D. (2005), 3-coloring in time O(1.3289n), Journal of Algorithms.