|  |
| --- |
| 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 №3**

**«Алгоритмы на графах»**

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

**Вариант № 45**

Выполнил: студент 2 курса

группы \_\_\_\_\_\_\_\_\_\_\_

шифр \_\_\_\_\_\_\_\_\_\_\_\_

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

*(фио студента)*

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

Москва 2021 г.

**Вариант № 45.**

**Алгоритм:** Вывести на экран все существующие пути в ациклическом орграфе

**Способ представления графа:** Матрица 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.

Для поиска всех путей в ациклическом орграфе воспользуемся поиском в глубину. Применим его к каждой вершине на графе и после каждого рекурсивного вызова будем добавлять текущий путь в список путей. На выходе получим все возможные пути в данном оргафе.

**Листинг программы.**

def load\_matrix(filename):

    with open(filename) as f:

        matrix = []

        for line in f.readlines():

            matrix.append(list(map(int, line.split())))

        return matrix

def DFS(matrix):

    path\_stack = []

    paths = []

    def \_dfs(v1: int):

        path\_stack.append(v1)

        for v2, e in enumerate(matrix[v1]):

            if e > 0 and not v2 in path\_stack:

                \_dfs(v2)

        if len(path\_stack) > 1:

            paths.append(path\_stack.copy())

        path\_stack.remove(v1)

    for v in range(len(matrix)):

        \_dfs(v)

    return sorted(paths)

matrix = load\_matrix('graph\_task45.txt')

paths = DFS(matrix)

print("Количество путей:", len(paths))

print("Пути:")

print("\n".join(["->".join(map(str, path)) for path in paths]))

**Скриншот работы программы:**

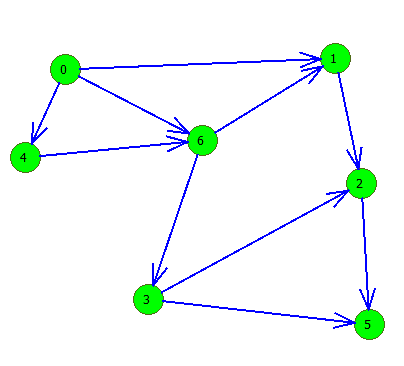


Рисунок 3. – Пример ориентированного ненагруженного графа.

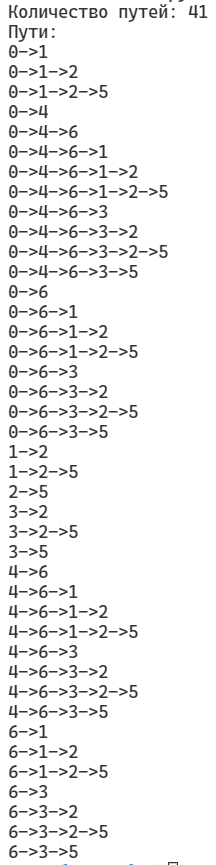
****

Рисунок 4. – Пример работы программы.

**Выводы.**

В результате выполнения данной работы была изучена структура граф, ее свойства и операции над ней. Так же был реализован алгоритм поиска в глубину на для поиска всех путей в ациклическом орграфе.

**Литература:**

1. Алгоритмы: построение и анализ. Кормен Т.Х., Лейзерсон Ч.И.

2. https://en.wikipedia.org/wiki/Depth-first\_search