|  |
| --- |
| 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 № 92**

Выполнил: студент 2 курса

группы \_\_\_\_\_\_\_\_\_\_\_

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

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

*(фио студента)*

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

Москва 2021 г.

**Вариант № 92.**

**Алгоритм:** методом обхода в ширину вычислить цикломатическую сложность   
графа

**Способ представления графа:** Матрица инцидентности.

**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.

**Матрица инцидентности** — одна из форм представления графа, в которой указываются связи между инцидентными элементами графа (ребро(дуга) и вершина). Столбцы матрицы соответствуют ребрам, строки — вершинам. Ненулевое значение в ячейке матрицы указывает связь между вершиной и ребром (их инцидентность).

В случае ориентированного графа каждой дуге <x,y> ставится в соответствующем столбце: «1» в строке вершины x и «-1» в строке вершины y; если связи между вершиной и ребром нет, то в соответствующую ячейку ставится «0».

Например, для графа изображенного на рисунке 1 матрица инцидентности будет выглядеть так:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
| A | 1 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | -1 | 0 | 0 |
| B | -1 | 1 | 1 | 0 | 0 | 0 | -1 | 0 | 0 | -1 | 0 |
| C | 0 | -1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| D | 0 | 0 | -1 | -1 | 0 | 1 | 0 | 0 | 0 | 0 | -1 |
| E | 0 | 0 | 0 | 0 | -1 | 0 | 1 | 1 | 0 | 0 | 0 |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 1 | 1 | 0 |
| G | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Для поиска цикломатической сложности графа наложим некоторые ограничения на вид графа. Граф должен быть ориентированным, не взвешенным. Должен иметь вход (вершина в которую нет пути) и выход (вершина из которой нет пути). Так же граф может содержать циклы.

Методом поиска в ширину найдем количество циклов в графе. Далее по формуле , где E – количество ребер, N – количество вершин, Р – количество циклов найдем цикломатическую сложность графа.

**Листинг программы.**

def read\_inc\_matrix(filename):

    with open(filename, 'r') as f:

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

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

        inc\_matrix = []

        for line in f.readlines():

            inc\_matrix.append(list(map(int, line.split())))

        for i in range(len(inc\_matrix[0])):

            first = None

            second = None

            for j in range(dim):

                if inc\_matrix[j][i] != 0:

                    if inc\_matrix[j][i] < 0:

                        second = j

                    else:

                        first = j

                    if first and second:

                        break

            matrix[first][second] = 1

    return matrix, dim, len(inc\_matrix[0])

def bfs(matrix):

    bfs\_queue = [0]

    visited = set()

    cycles\_count = 0

    while bfs\_queue:

        for i, e in enumerate(matrix[bfs\_queue.pop(0)]):

            if not e:

                continue

            if i in visited:

                cycles\_count += 1

            else:

                bfs\_queue.append(i)

                visited.add(i)

    return cycles\_count

matrix, v, e = read\_inc\_matrix('graph\_task92.txt')

cycles\_count = bfs(matrix)

print(f"Количество вершин: {v}")

print(f"Количество граней: {e}")

print(f"Количество циклов: {cycles\_count}")

print(f"Цикломатическая сложность графа: {e - v + 2 \* cycles\_count}")

**Скриншот работы программы:**

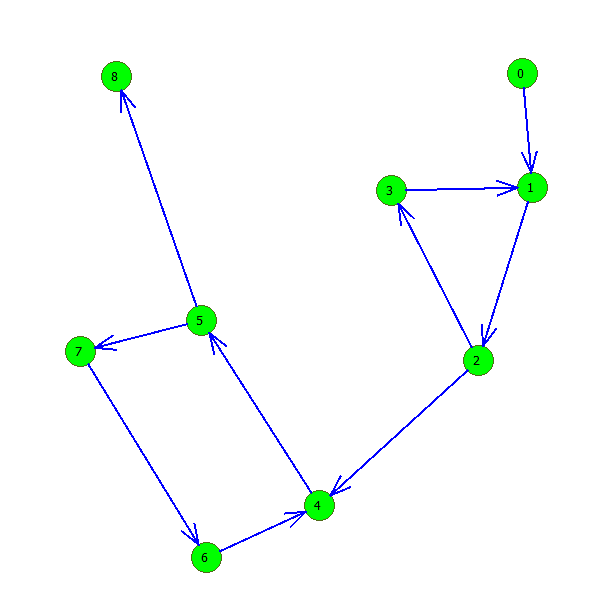
****

Рисунок 3. – Пример ориентированного ненагруженного графа.

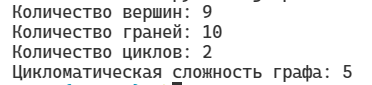
****

Рисунок 4. – Пример работы программы.

**Выводы.**

В результате выполнения данной работы были изучены свойства графа, способы его задания и работа с ним. Также был реализован алгоритм поиска в ширину для подсчета циклов в графе и подсчет цикломатической сложности графа

**Литература:**

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

2. https://ru.wikipedia.org/wiki/Цикломатическая\_сложность

3. https://ru.wikipedia.org/wiki/Поиск\_в\_ширину