|  |
| --- |
| 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 № 32**

Выполнил: студент 2 курса

группы \_\_\_\_\_\_\_\_\_\_\_

шифр \_\_\_\_\_\_\_\_\_\_\_\_

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

*(фио студента)*

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

Москва 2021 г.

**Вариант № 32.**

**Алгоритм:** определить внешний радиус невзвешенного ориентированного   
графа методом обхода в ширину. (Внешним радиусом графа   
будем называть наибольшее среди кратчайших расстояние от   
центра до какого-либо узла.)

**Способ представления графа:** Список дуг

**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 дуг (ребер)** – список ребер графа, где каждая новая строка списка состоит из 3х элементов: исходящая вершина, входящая вершина и вес данного ребра.

Например, для графа изображенного на рисунке 1 список дуг будет выглядеть так:

* A B 1
* A G 1
* B C 1
* B D 1
* C D 1
* C E 1
* D A 1
* E B 1
* E F 1
* F A 1
* F B 1
* G A 1

Для поиска центра графа методом поиска в ширину составим при помощи него матрицу расстояний. Матрицей расстояний назовем матрицу, где в i-ой строке каждый j элемент – это минимальное расстояние от вершины i до вершины j (если существует путь проходящий через вершины графа).

Далее для каждой строки матрицы, где все элементы ненулевые, найдем максимальное значение расстояния. А потом из списка максимальных расстояний найдем минимальное, номер строки, которой принадлежит данное расстояние и является центром графа, а расстояние – радиус

**Листинг программы.**

def read\_edge\_list(filename: str):

    with open(filename, 'r') as f:

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

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

        for line in f.readlines():

            v1, v2, l = map(int, line.split())

            matrix[v1][v2] = l

    return matrix

def BFS(matrix):

    distance\_matrix = [[0] \* len(matrix) for i in range(len(matrix))]

    def \_bfs(v: int):

        queue = [v]

        visited = set()

        while queue:

            cur\_node = queue.pop(0)

            for i, e in enumerate(matrix[cur\_node]):

                if i in visited:

                    continue

                if e > 0:

                    distance = e + distance\_matrix[v][cur\_node]

                    distance\_matrix[v][i] = distance

                    queue.append(i)

                    visited.add(i)

    for i in range(len(matrix)):

        \_bfs(i)

    return distance\_matrix

def find\_center(matrix):

    max\_distances = []

    for i, row in enumerate(matrix):

        skip = False

        for j, l in enumerate(row):

            if l == 0 and j != i:

                skip = True

                break

        if skip:

            continue

        else:

            max\_distances.append((i, max(row)))

    max\_distances.sort(key=lambda tup: tup[1])

    return max\_distances[0]

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

    from pprint import pprint

    matrix = read\_edge\_list('graph\_task32.txt')

    distance\_matrix = BFS(matrix)

    print("Матрица расстояний:")

    pprint(distance\_matrix)

    center\_v, radius = find\_center(distance\_matrix)

    print(f"Центр графа: {center\_v}, радиус: {radius}")

**Скриншот работы программы:**

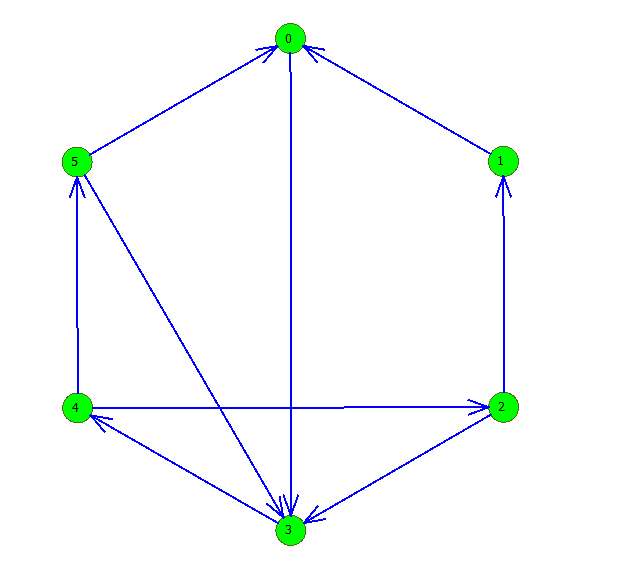


Рисунок 3. – Пример ориентированного ненагруженного графа.

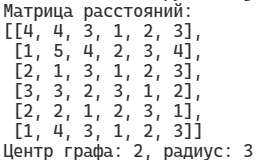
****

Рисунок 4. – Пример работы программы для графа из рисунка 3.

**Выводы.**

В результате выполнения данной работы была изучена структура граф, ее свойства и операции над ней. Так же был реализован алгоритм поиска в ширину для поиска центра графа и внешнего радиуса.

**Литература:**

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

2. https://e-maxx.ru/algo/bfs

3. https://ru.wikipedia.org/wiki/Центр\_графа