

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет Информатика и вычислительная техника

Кафедра Кибербезопасность информационных систем

**ПРАКТИЧЕСКАЯ РАБОТА № 3**

на тему «Компоненты связности графа»

Выполнила студентка гр. ВКБ32

Дедиченко Дарья Валентиновна

Проверил

Доцент, Савельев Василий Александрович

Ростов-на-Дону

2022

Цель работы:изучить компоненты связности и прочие топологические характеристики графа.

**Задание 1.**

def read\_graph(file\_name: str):

    graph\_list = []

    try:

        with open(file\_name, 'r') as file:

            for line in file:

                graph\_list.append(list(map(int, line.split())))

        return graph\_list

    except:

        return None

def dfs(graph\_list: list, vertex: int):

    visited = [False] \* len(graph\_list)

    def step(vertex):

        visited[vertex] = True

        for sub\_vertex in graph\_list[vertex]:

            if not visited[sub\_vertex]:

                step(sub\_vertex)

    step(vertex)

    return visited

def count\_component(graph\_list: list):

    vertex\_component = [-1] \* len(graph\_list)

    visited = [False] \* len(graph\_list)

    number = 0

    while False in visited:

       visited\_dfs = dfs(graph\_list, visited.index(False))

       for ind in range(len(visited\_dfs)):

           if visited\_dfs[ind]:

               visited[ind] = True

               vertex\_component[ind] = number

       number += 1

    return number, vertex\_component

def is\_connected(graph\_list: list, first\_vertex: int, second\_vertex: int):

    component = count\_component(graph\_list)[1]

    if component[first\_vertex] == component[second\_vertex]:

        return True

    else:

        return False

g = read\_graph('graph1.txt')

print(dfs(g, 0))

print(count\_component(g)[0])

print(count\_component(g)[1])

print(is\_connected(g, 1, 3))

print(is\_connected(g, 0, 6))

class Graph():

    def \_\_init\_\_(self, V):

        self.V = V

        self.graph = [[0 for column in range(V)] \

                                for row in range(V)]

    def isBipartite(self, src):

         colorArr = [-1] \* self.V

        colorArr[src] = 1

        queue = []

        queue.append(src)

        while queue:

            u = queue.pop()

            if self.graph[u][u] == 1:

                return False;

            for v in range(self.V):

                if self.graph[u][v] == 1 and colorArr[v] == -1:

                    colorArr[v] = 1 - colorArr[u]

                    queue.append(v)

                elif self.graph[u][v] == 1 and colorArr[v] == colorArr[u]:

                    return False

        return True

g = Graph(4)

g.graph = [[0, 1, 0, 1],

            [1, 0, 1, 0],

            [0, 1, 0, 1],

            [1, 0, 1, 0]]

print "Yes" if g.isBipartite(0) else "No"