|  |
| --- |
| МИНОБРНАУКИ РОССИИФедеральное государственноебюджетное образовательное учреждениевысшего образования **«МИРЭА – Российский технологический университет»****РТУ МИРЭА** |

Институт информационных технологий

Кафедра корпоративных информационных систем

|  |
| --- |
| **ОТЧЕТ****по лабораторной работе №3** |
| **по дисциплине** |
| **«Структуры и алгоритмы обработки данных»****Тема лабораторной работы: «**Графовые структуры данных**»** |
| Студент группы  | ИКБО-08-18 | Валяев Д.А. |
| Принял  | ассистент кафедры КИС | Исаева И.А |
|  |  |  |
| Выполнено  | «\_\_\_» \_\_\_\_\_\_ 201\_\_ г. | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* |
|  |  | *(подпись студента)* |
| Зачтено | «\_\_\_» \_\_\_\_\_\_ 201\_\_ г. | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* |
|  |  | *(подпись преподавателя)* |

Москва 2020

1. **Задача №1**
	1. **Постановка задачи**

Задан ориентированный граф с N (1 <N <10) вершинами, пронумерованными целыми числами от 1 до N. Напишите программу, которая подсчитывает количество различных путей между всеми парами вершин графа.

* 1. **Описание используемых структур данных**

Граф — абстрактный математический объект, представляющий собой множество вершин графа и набор рёбер, то есть соединений между парами вершин. Например, за множество вершин можно взять множество аэропортов, обслуживаемых некоторой авиакомпанией, а за множество рёбер взять регулярные рейсы этой авиакомпании между городами.

Ориентированный граф — граф, рёбрам которого присвоено направление. Направленные рёбра именуются также дугами. Граф, ни одному ребру которого не присвоено направление, называется неориентированным графом.



Рисунок 1.1 – Ориентированный граф

* 1. **Описание алгоритма**

При запуске программы пользователь выбирает способ заполнения графа (автоматически или вводом с клавиатуры), при выборе ручного заполнения, программа запрашивает количество узлов графа. Далее программа поочередно запрашивает существующие пути между узлами. После заполнения матрицы, по которой строится граф, с помощью библиотеки networkX происходит процесс графического представления графа. Получив на вход необходимые данные, программа подсчитывает количество путей между каждой парой точек.

* 1. **Тестирование**

На вход программы поступает матрица:

[0, 1, 1, 0, 0, 0]

[0, 0, 0, 1, 1, 0]

[0, 0, 0, 1, 1, 0]

[0, 0, 0, 0, 0, 1]

[0, 0, 0, 0, 0, 1]

[0, 0, 0, 0, 0, 0]

|  |  |
| --- | --- |
| Теоретический результат работы программы | Практический результат работы программы |
| Кол-во путей от 1 до 2: 1Кол-во путей от 1 до 3: 1Кол-во путей от 1 до 4: 2Кол-во путей от 1 до 5: 2Кол-во путей от 1 до 6: 4Кол-во путей от 2 до 1: 0Кол-во путей от 2 до 3: 0Кол-во путей от 2 до 4: 1Кол-во путей от 2 до 5: 1Кол-во путей от 2 до 6: 2Кол-во путей от 3 до 1: 0Кол-во путей от 3 до 2: 0Кол-во путей от 3 до 4: 1Кол-во путей от 3 до 5: 1Кол-во путей от 3 до 6: 2Кол-во путей от 4 до 1: 0Кол-во путей от 4 до 2: 0Кол-во путей от 4 до 3: 0Кол-во путей от 4 до 5: 0Кол-во путей от 4 до 6: 1Кол-во путей от 5 до 1: 0Кол-во путей от 5 до 2: 0Кол-во путей от 5 до 3: 0Кол-во путей от 5 до 4: 0Кол-во путей от 5 до 6: 1Кол-во путей от 6 до 1: 0Кол-во путей от 6 до 2: 0Кол-во путей от 6 до 3: 0Кол-во путей от 6 до 4: 0Кол-во путей от 6 до 5: 0  |  Кол-во путей от 1 до 2: 1Кол-во путей от 1 до 3: 1Кол-во путей от 1 до 4: 2Кол-во путей от 1 до 5: 2Кол-во путей от 1 до 6: 4Кол-во путей от 2 до 1: 0Кол-во путей от 2 до 3: 0Кол-во путей от 2 до 4: 1Кол-во путей от 2 до 5: 1Кол-во путей от 2 до 6: 2Кол-во путей от 3 до 1: 0Кол-во путей от 3 до 2: 0Кол-во путей от 3 до 4: 1Кол-во путей от 3 до 5: 1Кол-во путей от 3 до 6: 2Кол-во путей от 4 до 1: 0Кол-во путей от 4 до 2: 0Кол-во путей от 4 до 3: 0Кол-во путей от 4 до 5: 0Кол-во путей от 4 до 6: 1Кол-во путей от 5 до 1: 0Кол-во путей от 5 до 2: 0Кол-во путей от 5 до 3: 0Кол-во путей от 5 до 4: 0Кол-во путей от 5 до 6: 1Кол-во путей от 6 до 1: 0Кол-во путей от 6 до 2: 0Кол-во путей от 6 до 3: 0Кол-во путей от 6 до 4: 0Кол-во путей от 6 до 5: 0 |



Рисунок 1.2 – Графическое представление графа

* 1. **Листинг программы**
1. **import** networkx **as** nx
**from** networkx **import** NetworkXError

path = **''**
arr = [[0, 1, 1, 0, 0, 0], [0, 0, 0, 1, 1, 0], [0, 0, 0, 1, 1, 0], [0, 0, 0, 0, 0, 1], [0, 0, 0, 0, 0, 1],
 [0, 0, 0, 0, 0, 0]]
graph = {1: [2, 3],
 2: [4, 5],
 3: [4, 5],
 4: [6],
 5: [6],
 6: []}

**def** fill\_matrix():
 **global** arr
 arr = []
 inp = int(input(**"Введите кол-во вершин графа 1<k<10: "**))
 **if** 1 >= inp **or** inp > 10:
 print(**"Некорректный ввод, завершаю работу"**)
 quit()
 **for** z **in** range(inp):
 arr2 = []
 arr3.append([])
 **for** y **in** range(inp):
 arr2.append(0)
 arr.append(arr2)
 **for** z **in** range(inp):
 temp = []
 **for** y **in** range(inp):
 **if** arr[z][y] == 0 **and** z != y:
 \_inp = int(
 input(**"Если путь от {} до {} существует введите 1, иначе 0: "**.
 format(str(z + 1), str(y + 1))))
 arr[z][y] = \_inp
 **if** \_inp != 0:
 temp.append(y + 1)
 **elif** z == y:
 arr[z][y] = 0
 print(temp)
 graph[z + 1] = temp
 **return** inp

**def** Print(array):
 **for** row **in** array:
 print(row)

**def** draw():
 **try**:

Gr = nx.Graph()
 **for** i **in** range(len(arr)):
 **for** y **in** range(len(arr[i])):
 **if** arr[i][y] == 1:
 Gr.add\_edge(i + 1, y + 1)
 nx.draw(Gr, with\_labels=**True**)
 **except** NetworkXError:
 print(**"Где-то закралась ошибочка, завершаю работу"**)
 quit()

**def** findPaths(Graph, st, ed):
 **if** ed == 0:
 **return** [[st]]
 paths = [[st] + pas **for** neighbor **in** Graph.neighbors(st) **for** pas **in** findPaths(Graph, neighbor, ed - 1)
 **if** st **not in** pas]
 **return** paths

**def** pather(gr, start, finish):
 stack = [(start, [start])]
 **while** stack:
 (x, y) = stack.pop()
 **for** i **in** set(gr[x]) - set(y):
 **if** i == finish:
 **yield** y + [i]
 **else**:
 stack.append((i, y + [i]))

**try**:
 fill = int(input(**"заполнение\nПресет - 1\nС клавиатуры - 2"**))
 **if** fill == 2:
 path = fill\_matrix()
 **else**:
 path = len(arr)
**except** ValueError:
 print(**"Некорректный ввод, завершаю работу"**)
 quit()

draw()
Print(arr)

**for** a **in** range(len(arr)):
 **for** b **in** range(len(arr)):
 **if** a != b:
 print(**"Кол-во путей от {} до {}: "**.format(a + 1, b + 1) + str(len(list(pather(graph, a + 1, b + 1)))))