МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Севастопольский государственный университет»

Кафедра «Информационные системы»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

по дисциплине

«Методы системного анализа и проектирования информационных систем»

Вариант 9

Выполнил:
Донец Н.О.
Проверил:
Кудрявченко И.В.

Севастополь
2024 г.

Цель работы:

Углубление теоретических знаний в области системного анализа, исследование способов оценки сложных систем.

Задание:

На основе исходного варианта задания, который представлен на таблицах 1-4, построить взаимосвязь всех уровней системы. Вычислить влияние вариантов самого нижнего уровня на проектирование всей системы. Написать программу на языке программирования python (или др.), которая решает задачу методом решающих матриц любой размерности. Сверить решение по варианту при помощи написанной программы. Дополнительное задание: написать программу на языке программирования python (или др), которая рисует в консоли граф на основе введенных матриц.

Таблица 1 – Веса первого уровня

a1	a2	a3	a4
0,4	0,2	0,3	0,1

Таблица 2 – Матрица связей первого и второго уровней

	a1	a2	a3	a4
b1	0,6		0,4	
b2				1
b3	0,2	0,7	0,1	
b4	0,9			0,1
b5	0,2	0,4	0,2	0,2

Таблица 3 – Матрица связей второго и третьего уровней

	b1	b2	b3	b4	b5
c1	0,5		0,1		0,4
c2		0,5	0,3	0,2	
c3	0,6			0,4	
c4		0,4	0,3		0,3

Таблица 4 – Матрица связей третьего и четвёртого уровней

	c1	c2	c3	c4
d1	0,6	0,2		0,2
d2	0,3		0,2	0,5
d3		0,5	0,3	0,2

Ход работы:

Согласно с полученным заданием был построен граф, который показан на рисунке 1.

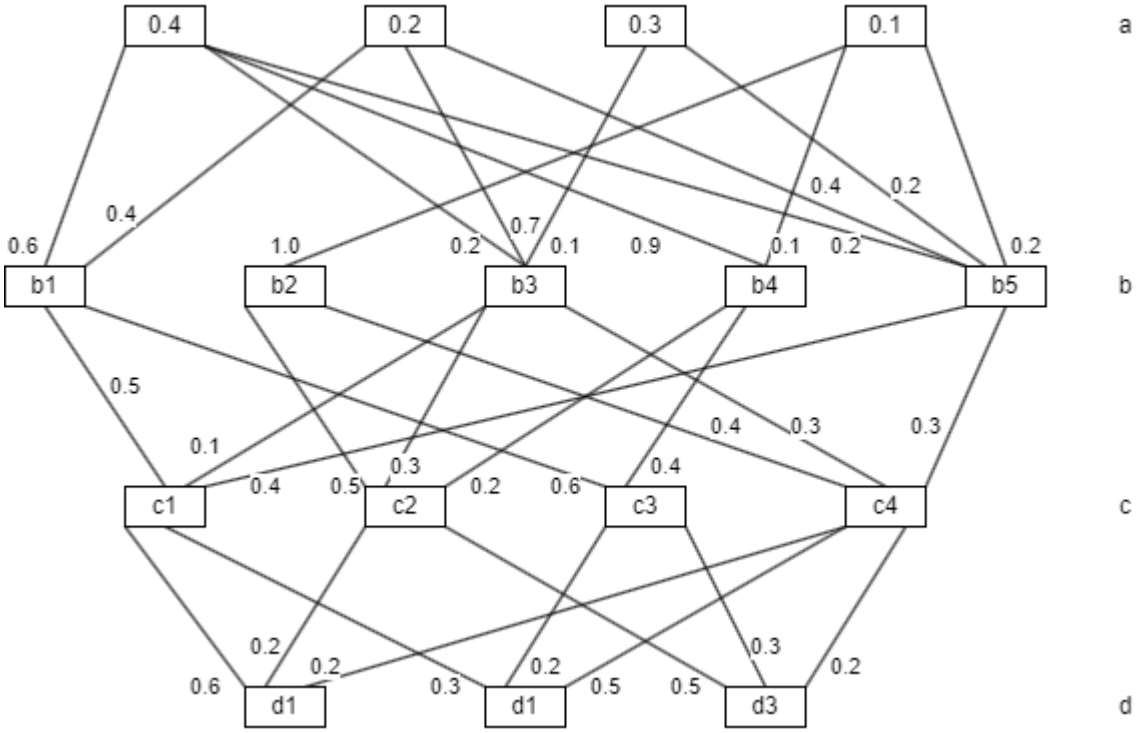

Рисунок 1 – Граф

По формуле на рисунке 2 были посчитаны веса для каждого уровня. Результаты расчётов можно увидеть в таблице 5

$$b_i = \sum_{j=1}^{na} p_{ij} a_j$$

Рисунок 2 – Формула расчёта весов

Таблица 5 – Рассчитанные веса всех уровней

0,36	b1	0,22803	c1	0,24529	d1
0,1	b2	0,150758	c2	0,244148	d2
0,25	b3	0,275758	c3	0,234158	d3
0,37	b4	0,141667	c3		
0,24	b5				
1,32	b_{sum}	0,8	c_{sum}	0,72	d_{sum}

Так как сумма значений весов должна быть равна единице, при необходимости веса были нормированы. Нормированные веса представлены в таблице 6.

Таблица 6 – Нормированные веса

b1	0,272727	c1	0,286394	d1	0,338988
b2	0,075758	c2	0,189343	d2	0,33741
b3	0,189394	c3	0,346337	d3	0,323603
b4	0,280303	c3	0,177926		
b5	0,181818				
Сумма	1		1		1

Исходя из вышепредставленных расчётов можно сделать вывод, что все вариант имеют примерно равное влияние на систему, однако d_1 немного больше.

Далее на языке программирования python была написана программа, которая решает задачу методом решающих матриц любой размерности, а также строит соответствующий граф. Ввод матриц осуществлялся через файлы. Для проверки был использован тот же вариант задания. Граф и ответ программы представлены на рисунках 3 и 4 соответственно. Программа показана на листинге 1.

Листинг 1 – Программа, решающая задачу методом решающих матриц

```
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
from copy import copy, deepcopy

def read_matrix(file):
    matrix = []
    for line in file:
        line = line.strip()
        if not line:
            break
        matrix.append(list(map(float, line.split())))
    return matrix

def read_matrix_from_file(filename):
    with open(filename, 'r') as file:
        return read_matrix(file)

def read_matrices_from_file(filename):
    matrices = []
    with open(filename, 'r') as file:
        while True:
            matrix = read_matrix(file)
            if not matrix:
                break
            matrices.append(matrix)
    return matrices


def draw_combined_graph(initial_nodes, weight_matrices):
    G = nx.Graph()
    node_positions = {}
    x_offset = 0
    y_offset = 0
    spacing = 6

    num_initial_nodes = len(initial_nodes[0])
    for i in range(num_initial_nodes):
        node_name = f'I_{i}'
        G.add_node(node_name, label=f'a{i + 1}')
        node_positions[node_name] = (x_offset, y_offset - i * spacing)

    previous_layer_nodes = [f'I_{i}' for i in range(num_initial_nodes)]
    x_offset += 2

    for idx, matrix in enumerate(weight_matrices):
        num_rows = len(matrix)
        num_cols = max(len(row) for row in matrix)

        if idx == len(weight_matrices) - 1:
            current_layer_nodes = [f'd{j + 1}' for j in range(num_rows)]
        else:
            current_layer_nodes = [f'N{idx}_{j}' for j in range(num_rows)]

        for i in range(num_rows):
            node_label = f'd{i + 1}' if idx == len(weight_matrices) - 1 else
f'b{idx + 1}_{i + 1}'
            G.add_node(current_layer_nodes[i], label=node_label)
            node_positions[current_layer_nodes[i]] = (x_offset, y_offset - i *
spacing)
```

```
        for i in range(num_rows):
            for j in range(num_cols):
                if matrix[i][j] != 0:
                    G.add_edge(previous_layer_nodes[j],
current_layer_nodes[i], weight=matrix[i][j])

        previous_layer_nodes = current_layer_nodes
        x_offset += 3

    pos = {node: (x, y) for node, (x, y) in node_positions.items()}
    edge_labels = nx.get_edge_attributes(G, 'weight')
    labels = nx.get_node_attributes(G, 'label')

    plt.figure(figsize=(16, 8))
    nx.draw(G,    pos,    with_labels=True,    labels=labels,    node_size=700,
edge_color='gray', linewidths=1, font_size=10)
    nx.draw_networkx_edge_labels(G,        pos,        edge_labels=edge_labels,
label_pos=0.85)
    plt.show()

filename = 'firstLevel.txt'
first_level = read_matrix_from_file(filename)
print(f"First level :")
for row in first_level:
    print(row)
print()

filename = 'input.txt'
matrices = read_matrices_from_file(filename)
for i, matrix in enumerate(matrices):
    print(f"Matrix {i + 1}:")
    for row in matrix:
        print(row)
    print()

draw_combined_graph(first_level, matrices)

weights = [first_level]

for i in range(0, len(matrices)):
    new_weights = []
    for row in matrices[i]:
        line = np.multiply(row, weights[i])
        new_weight = np.sum(line)/np.sum(weights[i])
        new_weights.append(new_weight)
    weights.append(new_weights)

answer = weights[len(weights) - 1] / sum(weights[len(weights) - 1])
for i in range(0, len(weights)):
    print(f"Weights {i + 1}:")
    print(weights[i])
print()

print("Answer: ")
print(answer)
```

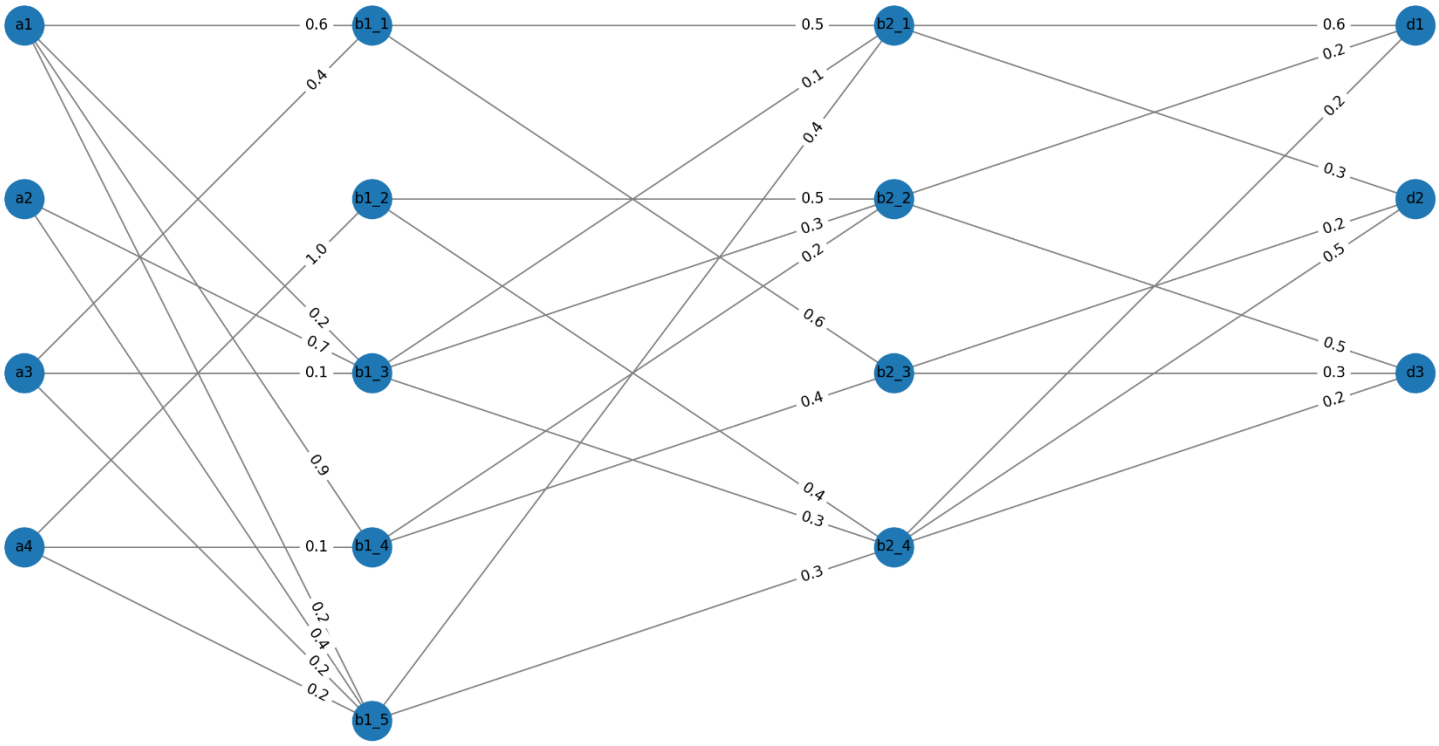

Рисунок 3 – Граф программы

```
Answer:
[0.33898751 0.3374096  0.32360289]
```

Рисунок 4 – Ответ программы

Вывод программы совпадает со сделанными ранее расчётами, что только подтверждает их правильность.

Выводы

В ходе лабораторной работы были углублены теоретические знания в области системного анализа, а также были исследованы способы оценки сложных систем. С помощью метода решающих матриц были получены оценки на всех этапах, и был выявлен вариант, который сильнее всего влияет на проектирование системы. Также был построен граф, соответствующий

задаче. Была написана программа, которая решала поставленную задачу методом решающих матриц, а также строила граф в соответствии с введёнными матрицами.