БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра информационных систем управления

Чернышева Ксения Юрьевна

**Отчет по заданию 8**

**(“Принятие решений в информационном обществе”)**

студента 3 курса 12 группы

Преподаватель:

*Железко Борис Александрович*

Минск, 2024

**ОГЛАВЛЕНИЕ**

[Задание 3](#_heading=h.gjdgxs)

[Постановка задания 3](#_heading=h.30j0zll)

[Решение задания 3](#_heading=h.1fob9te)

[Результат работы программы 5](#_heading=h.3znysh7)

# Задание

## Постановка задания

Реализовать лабораторную работу 2 с учетом наличия нескольких экспертов

## Решение задания

Для автоматической генерации матриц для каждого эксперта можно использовать следующий код:

import numpy as np

def generate\_matrix(size):

values = [1, 2, 3, 4, 5, 6, 7, 8, 9] # Список значений для заполнения матрицы

while True:

matrix = np.random.choice(values, (size, size)) # Генерация матрицы случайных значений

np.fill\_diagonal(matrix, 1) # Заполнение диагонали матрицы единицами

revert\_matrix(matrix) # Обращение матрицы

L = calculate\_local\_priorities(matrix) # Вычисление локальных приоритетов

if consistency\_check(matrix, L): # Проверка на согласованность матрицы

break

return matrix

def revert\_matrix(matrix):

size = len(matrix[0])

for i in range(size):

for j in range(i):

matrix[j][i] = 1 / matrix[i][j]

return matrix

def calculate\_local\_priorities(matrix):

size = len(matrix[0])

column\_sums = np.sum(matrix, axis=0)

L = np.divide(column\_sums, np.sum(column\_sums)) # Вычисление локальных приоритетов

return L

def consistency\_check(matrix, L):

N = len(matrix[0])

column\_sums = np.sum(matrix, axis=0)

lmbda = np.dot(column\_sums, L)

IS = (lmbda - N) / (N - 1)

SlS\_values = [0.58, 0.9, 1.12, 1.24, 1.32, 1.41, 1.45, 1.49]

SlS = SlS\_values[N - 3]

OS = IS / SlS

threshold = 0.2

return OS < threshold

Код, который был предоставлен, является реализацией функций для генерации и проверки матриц при применении метода анализа иерархий (МАИ). Вот объяснение каждой функции:

1. Функция generate\_matrix(size):
   * Принимает на вход размер матрицы (size).
   * Генерирует случайную матрицу заданного размера, используя функцию np.random.choice, которая выбирает случайные значения из заданного списка.
   * Заполняет диагональ матрицы единицами, используя функцию np.fill\_diagonal.
   * Обращает матрицу, вызывая функцию revert\_matrix.
   * Вычисляет локальные приоритеты матрицы, вызывая функцию calculate\_local\_priorities.
   * Проверяет согласованность матрицы, вызывая функцию consistency\_check.
   * Если матрица удовлетворяет условиям согласованности, возвращает её. В противном случае повторяет генерацию до получения согласованной матрицы.
2. Функция revert\_matrix(matrix):
   * Принимает на вход матрицу (matrix).
   * Обращает значения в матрице путем деления 1 на соответствующие элементы матрицы (элементы ниже главной диагонали).
   * Возвращает обращенную матрицу.
3. Функция calculate\_local\_priorities(matrix):
   * Принимает на вход матрицу (matrix).
   * Вычисляет сумму значений в каждом столбце матрицы с помощью функции np.sum и ось 0 (столбцы).
   * Вычисляет локальные приоритеты путем деления суммы значений каждого столбца на общую сумму всех столбцов.
   * Возвращает массив локальных приоритетов.
4. Функция consistency\_check(matrix, L):
   * Принимает на вход матрицу (matrix) и массив локальных приоритетов (L).
   * Вычисляет сумму значений в каждом столбце матрицы с помощью функции np.sum и ось 0 (столбцы).
   * Вычисляет параметр lambda (λ), умножая суммы значений каждого столбца на соответствующие локальные приоритеты и суммируя результаты.
   * Вычисляет показатель согласованности (Индекс Согласованности, IS) по формуле (λ - N) / (N - 1), где N - размер матрицы.
   * Задает значения показателя Согласованности по шкале SlS (SlS\_values) в зависимости от размера матрицы.
   * Вычисляет отношение согласованности (Отношение Согласованности, OS) путем деления IS на соответствующее значение SlS.
   * Задает пороговое значение (threshold) для согласованности.
   * Возвращает True, если OS меньше порогового значения, и False в противном случае.

Далее напишем функцию, которая будет получать от эксперта набор всех матриц:

def get\_results\_from\_expert(n, m):  
 K\_criteria = generate\_matrix(n)  
 K = []  
 for i in range(n):  
 K.append(generate\_matrix(m))  
  
 return K\_criteria, K;

Также напишем функцию, которая будет принимать список матриц одинакового размера, и возвращать усредненную матрицу:

def average\_matrices(matrix\_list):  
 if not matrix\_list:  
 return None  
  
 num\_matrices = len(matrix\_list)  
 result = revert\_matrix(np.sum(matrix\_list, axis=0) / num\_matrices)  
  
 return result

Основной код:

experts\_number = int(input("Введите число экспертов:"))  
  
# Получение матриц от экспертов и вывод  
K\_criteria = [0] \* experts\_number  
K = [0] \* experts\_number  
for i in range(experts\_number):  
 K\_criteria[i], K[i] = get\_results\_from\_expert(n, m)  
  
for i in range(experts\_number):  
 print(f"Эксперт **{**i + 1**}**:")  
 print(f"Матрица критериев:**\n{**K\_criteria[i]**}**")  
 for j in range(n):  
 print(f"Матрица K**{**j + 1**}**:**\n{**K[i][j]**}**")  
  
  
# Усреднение матриц  
K\_criteria = average\_matrices(K\_criteria)  
  
  
K\_comp = []  
for i in range(n):  
 column = []  
 for j in range(experts\_number):  
 column.append(K[j][i])  
 K\_comp.append(average\_matrices(column))  
  
print(f"**\n**Усредненная матрица критериев:**\n{**K\_criteria**}**")  
for i in range(n):  
 print(f"Усредненная Матрица K**{**i + 1**}**:**\n{**K\_comp[i]**}**")  
  
  
# Код ниже без изменений, как и был в лабораторной работе 2  
L\_criteria = calculate\_local\_priorities(K\_criteria)  
  
L\_K = [calculate\_local\_priorities(K) for K in K\_comp]  
  
print(f"Локальные приоритеты критериев: **{**L\_criteria**}**")  
for i in range(n):  
 print(f"Локальные приоритеты альтернатив относительно критерия **{**i + 1**}**: **{**L\_K[i]**}**")  
  
G = [0] \* m  
for i in range(m):  
 for j in range(n):  
 G[i] += L\_K[j][i] \* L\_criteria[j]  
  
print(f"Глобальные приоритеты: **{**G**}**")  
  
G = [(idx, val) for idx, val in enumerate(G)]  
G = sorted(G, key=lambda x: x[1], reverse=True)  
  
print("Предпочтения:")  
for i in range(m):  
 print(f"M**{**G[i][0] + 2**}**:**{**G[i][1]**}**")

## Результат работы программы

Введите число экспертов:3

Эксперт 1:

Матрица критериев:

[[1. 3. 5. ]

[0.33333333 1. 0.5 ]

[0.2 2. 1. ]]

Матрица K1:

[[1. 2. 3. 2. 0.33333333]

[0.5 1. 0.14285714 0.2 0.25 ]

[0.33333333 7. 1. 2. 0.16666667]

[0.5 5. 0.5 1. 0.125 ]

[3. 4. 6. 8. 1. ]]

Матрица K2:

[[1. 0.33333333 9. 0.33333333 0.2 ]

[3. 1. 5. 5. 2. ]

[0.11111111 0.2 1. 2. 0.11111111]

[3. 0.2 0.5 1. 0.125 ]

[5. 0.5 9. 8. 1. ]]

Матрица K3:

[[1. 0.14285714 0.33333333 1. 0.11111111]

[7. 1. 3. 9. 0.25 ]

[3. 0.33333333 1. 7. 0.11111111]

[1. 0.11111111 0.14285714 1. 0.16666667]

[9. 4. 9. 6. 1. ]]

Эксперт 2:

Матрица критериев:

[[1. 4. 9. ]

[0.25 1. 8. ]

[0.11111111 0.125 1. ]]

Матрица K1:

[[1. 0.14285714 0.11111111 3. 0.125 ]

[7. 1. 5. 8. 7. ]

[9. 0.2 1. 4. 3. ]

[0.33333333 0.125 0.25 1. 0.33333333]

[8. 0.14285714 0.33333333 3. 1. ]]

Матрица K2:

[[1. 5. 2. 6. 9. ]

[0.2 1. 0.14285714 1. 8. ]

[0.5 7. 1. 9. 3. ]

[0.16666667 1. 0.11111111 1. 3. ]

[0.11111111 0.125 0.33333333 0.33333333 1. ]]

Матрица K3:

[[1. 3. 6. 2. 1. ]

[0.33333333 1. 3. 3. 0.14285714]

[0.16666667 0.33333333 1. 3. 0.14285714]

[0.5 0.33333333 0.33333333 1. 0.125 ]

[1. 7. 7. 8. 1. ]]

Эксперт 3:

Матрица критериев:

[[1. 0.2 6. ]

[5. 1. 9. ]

[0.16666667 0.11111111 1. ]]

Матрица K1:

[[1. 3. 1. 7. 7. ]

[0.33333333 1. 0.125 1. 8. ]

[1. 8. 1. 4. 9. ]

[0.14285714 1. 0.25 1. 3. ]

[0.14285714 0.125 0.11111111 0.33333333 1. ]]

Матрица K2:

[[1. 8. 5. 8. 5. ]

[0.125 1. 0.5 7. 0.5 ]

[0.2 2. 1. 6. 1. ]

[0.125 0.14285714 0.16666667 1. 0.14285714]

[0.2 2. 1. 7. 1. ]]

Матрица K3:

[[1. 6. 3. 3. 8. ]

[0.16666667 1. 0.14285714 0.14285714 2. ]

[0.33333333 7. 1. 2. 0.5 ]

[0.33333333 7. 0.5 1. 3. ]

[0.125 0.5 2. 0.33333333 1. ]]

Усредненная матрица критериев:

[[1. 0.53731343 6.27906977]

[1.86111111 1. 1.34161491]

[0.15925926 0.74537037 1. ]]

Усредненная Матрица K1:

[[1. 0.38297872 0.29032258 3.07317073 0.26923077]

[2.61111111 1. 0.19736842 0.48979592 0.70292887]

[3.44444444 5.06666667 1. 3. 0.46551724]

[0.32539683 2.04166667 0.33333333 1. 0.26470588]

[3.71428571 1.42261905 2.14814815 3.77777778 1. ]]

Усредненная Матрица K2:

[[1. 0.90225564 3.69863014 0.91139241 0.56485356]

[1.10833333 1. 0.32608696 2.23404255 1.14285714]

[0.27037037 3.06666667 1. 3.85714286 0.29032258]

[1.09722222 0.44761905 0.25925926 1. 0.19565217]

[1.77037037 0.875 3.44444444 5.11111111 1. ]]

Усредненная Матрица K3:

[[1. 0.4 0.85714286 1.63636364 0.2962963 ]

[2.5 1. 0.39130435 0.40298507 0.26086957]

[1.16666667 2.55555556 1. 3.07317073 0.16666667]

[0.61111111 2.48148148 0.32539683 1. 0.20930233]

[3.375 3.83333333 6. 4.77777778 1. ]]

Локальные приоритеты критериев: [0.44798517376020125, 0.40521859328348936, 0.1467962329563094]

Локальные приоритеты альтернатив относительно критерия 1: [0.10499879904440804, 0.11973934427283461, 0.32048718733508064, 0.09594869242303058, 0.35882597692464624]

Локальные приоритеты альтернатив относительно критерия 2: [0.20265486912131567, 0.17896665814830895, 0.17918808500546107, 0.08690386821773317, 0.3522865195071812]

Локальные приоритеты альтернатив относительно критерия 3: [0.11049736517092824, 0.10037868108212418, 0.17217955627725184, 0.10046430639730954, 0.5164800910723862]

Глобальные приоритеты: [0.14537802318060075, 0.14089728066148183, 0.24145916229888895, 0.09294643660297253, 0.379319097256056]

Предпочтения:

M6:0.379319097256056

M4:0.24145916229888895

M2:0.14537802318060075

M3:0.14089728066148183

M5:0.09294643660297253

Process finished with exit code 0