Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительной техники»

**ОТЧЕТ**

по лабораторной работе №5

по курсу «Логика и основы алгоритмизации в инженерных задачах.»

на тему: «Определение характеристик графов.»

Выполнили**:**

студенты группы 22ВВВ2

Коробкин В. Ю.

Горбатов К. В.

Проверили:

Митрохин М. А.

Акифьев И.В.

Пенза 2023

**Цель работы**

Определение характеристик графов.

**Лабораторное задание**

**Задание 1**

1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного графа *G*. Выведите матрицу на экран.
2. Определите размер графа *G*, используя матрицу смежности графа.
3. Найдите изолированные, концевые и доминирующие вершины.

**Задание 2\***

1. Постройте для графа G матрицу инцидентности.
2. Определите размер графа *G*, используя матрицу инцидентности графа.
3. Найдите изолированные, концевые и доминирующие вершины.

**Результат выполнения программы**

**5.1**

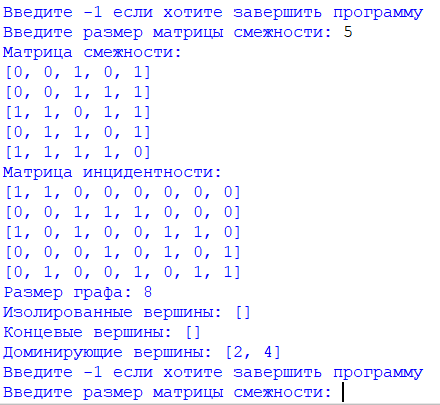
****

Рисунок 1 – вывод матриц смежности и инцидентности, размера графа, изолированных, концевых и доминирующих вершин матрицы смежности.

**5.2**

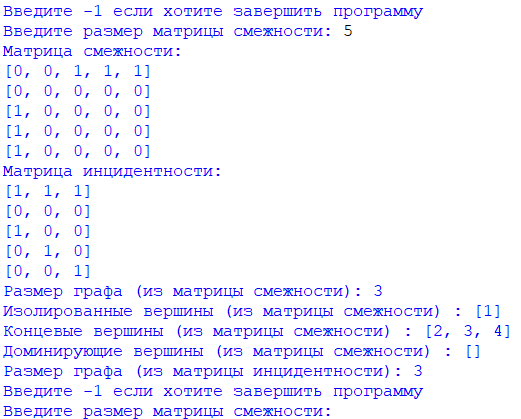
****

Рисунок 2 – добавление вывода размера графа исходя из матрицы инцидентности.

**5.3**

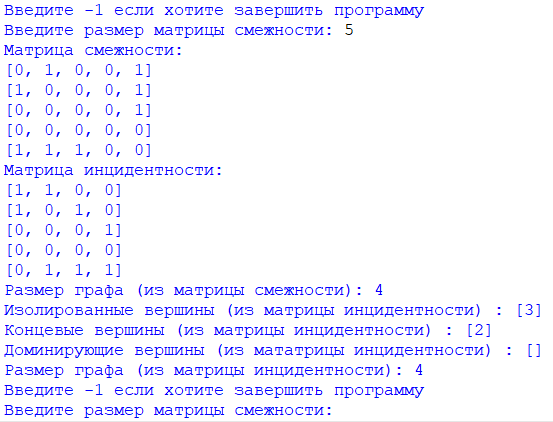
****

Рисунок 3 – добавление вывода изолированных, концевых и доминирующих вершин матрицы инцидентности.

**Приложение**

*Программа 5.1*

*import random*

*def generate\_random\_adjacency\_matrix(n, p):*

*adjacency\_matrix = [[0] \* n for \_ in range(n)]*

*for i in range(n):*

*for j in range(i + 1, n):*

*# Генерируем случайное число в диапазоне [0, 1]*

*random\_value = random.random()*

*# Если случайное число меньше или равно p, то устанавливаем ребро*

*if random\_value <= p:*

*adjacency\_matrix[i][j] = 1*

*adjacency\_matrix[j][i] = 1*

*return adjacency\_matrix*

*def generate\_incidence\_matrix(adjacency\_matrix):*

*num\_vertices = len(adjacency\_matrix)*

*num\_edges = sum(sum(row) for row in adjacency\_matrix) // 2*

*incidence\_matrix = [[0] \* num\_edges for \_ in range(num\_vertices)]*

*edge\_index = 0*

*for i in range(num\_vertices):*

*for j in range(i + 1, num\_vertices):*

*if adjacency\_matrix[i][j] == 1:*

*incidence\_matrix[i][edge\_index] = 1*

*incidence\_matrix[j][edge\_index] = 1*

*edge\_index += 1*

*return incidence\_matrix*

*#вероятность ребра #####################################*

*p = 0.6 #####################################*

*while True:*

*print("Введите -1 если хотите завершить программу ")*

*n = int(input("Введите размер матрицы смежности: "))*

*if n == -1:*

*break*

*# Генерируем матрицу смежности*

*adjacency\_matrix = generate\_random\_adjacency\_matrix(n, p)*

*# Определяем размер графа*

*edge\_count = sum(sum(row) for row in adjacency\_matrix) // 2*

*# Определяем размер графа (количество вершин)*

*graph\_size = len(adjacency\_matrix)*

*# Находим изолированные, концевые и доминирующие вершины*

*isolated\_vertices = [i for i in range(graph\_size) if sum(adjacency\_matrix[i]) == 0]*

*end\_vertices = [i for i in range(graph\_size) if sum(adjacency\_matrix[i]) == 1]*

*dominating\_vertices = [i for i in range(graph\_size) if sum(adjacency\_matrix[i]) > 0 and sum(adjacency\_matrix[i]) == graph\_size - 1]*

*# Строим матрицу инцидентности*

*incidence\_matrix = generate\_incidence\_matrix(adjacency\_matrix)*

*# Выводим матрицу смежности, размер графа, количество рёбер и найденные вершины*

*print("Матрица смежности:")*

*for row in adjacency\_matrix:*

*print(row)*

*# Выводим также матрицу инцидентности*

*print("Матрица инцидентности:")*

*for row in incidence\_matrix:*

*print(row)*

*print("Размер графа:", edge\_count)*

*print("Изолированные вершины:", isolated\_vertices)*

*print("Концевые вершины:", end\_vertices)*

*print("Доминирующие вершины:", dominating\_vertices)*

*Программа 5.2*

import random

def generate\_random\_adjacency\_matrix(n, p):

adjacency\_matrix = [[0] \* n for \_ in range(n)]

for i in range(n):

for j in range(i + 1, n):

# Генерируем случайное число в диапазоне [0, 1]

random\_value = random.random()

# Если случайное число меньше или равно p, то устанавливаем ребро

if random\_value <= p:

adjacency\_matrix[i][j] = 1

adjacency\_matrix[j][i] = 1

return adjacency\_matrix

def generate\_incidence\_matrix(adjacency\_matrix):

num\_vertices = len(adjacency\_matrix)

num\_edges = sum(sum(row) for row in adjacency\_matrix) // 2

incidence\_matrix = [[0] \* num\_edges for \_ in range(num\_vertices)]

edge\_index = 0

for i in range(num\_vertices):

for j in range(i + 1, num\_vertices):

if adjacency\_matrix[i][j] == 1:

incidence\_matrix[i][edge\_index] = 1

incidence\_matrix[j][edge\_index] = 1

edge\_index += 1

return incidence\_matrix

#вероятность ребра #####################################

p = 0.6 #####################################

while True:

print("Введите -1 если хотите завершить программу ")

n = int(input("Введите размер матрицы смежности: "))

if n == -1:

break

# Генерируем матрицу смежности

adjacency\_matrix = generate\_random\_adjacency\_matrix(n, p)

# Определяем размер графа

edge\_count = sum(sum(row) for row in adjacency\_matrix) // 2

# Определяем количество вершин

graph\_size = len(adjacency\_matrix)

# Находим изолированные, концевые и доминирующие вершины

isolated\_vertices = [i for i in range(graph\_size) if sum(adjacency\_matrix[i]) == 0]

end\_vertices = [i for i in range(graph\_size) if sum(adjacency\_matrix[i]) == 1]

dominating\_vertices = [i for i in range(graph\_size) if sum(adjacency\_matrix[i]) > 0 and sum(adjacency\_matrix[i]) == graph\_size - 1]

# Строим матрицу инцидентности

incidence\_matrix = generate\_incidence\_matrix(adjacency\_matrix)

# Определяем количество рёбер на основе матрицы инцидентности

num\_edges\_from\_incidence = len(incidence\_matrix[0])

# Выводим информацию о графе, включая количество рёбер из матрицы инцидентности

print("Матрица смежности:")

for row in adjacency\_matrix:

print(row)

# Выводим также матрицу инцидентности

print("Матрица инцидентности:")

for row in incidence\_matrix:

print(row)

print("Размер графа (из матрицы смежности):", edge\_count)

print("Изолированные вершины (из матрицы смежности) :", isolated\_vertices)

print("Концевые вершины (из матрицы смежности) :", end\_vertices)

print("Доминирующие вершины (из матрицы смежности) :", dominating\_vertices)

print("Размер графа (из матрицы инцидентности):", num\_edges\_from\_incidence)

*Программа 5.3*

*import random*

*def generate\_random\_adjacency\_matrix(n, p):*

*adjacency\_matrix = [[0] \* n for \_ in range(n)]*

*for i in range(n):*

*for j in range(i + 1, n):*

*# Генерируем случайное число в диапазоне [0, 1]*

*random\_value = random.random()*

*# Если случайное число меньше или равно p, то устанавливаем ребро*

*if random\_value <= p:*

*adjacency\_matrix[i][j] = 1*

*adjacency\_matrix[j][i] = 1*

*return adjacency\_matrix*

*def generate\_incidence\_matrix(adjacency\_matrix):*

*num\_vertices = len(adjacency\_matrix)*

*num\_edges = sum(sum(row) for row in adjacency\_matrix) // 2*

*incidence\_matrix = [[0] \* num\_edges for \_ in range(num\_vertices)]*

*edge\_index = 0*

*for i in range(num\_vertices):*

*for j in range(i + 1, num\_vertices):*

*if adjacency\_matrix[i][j] == 1:*

*incidence\_matrix[i][edge\_index] = 1*

*incidence\_matrix[j][edge\_index] = 1*

*edge\_index += 1*

*return incidence\_matrix*

*#вероятность ребра*

*p = 0.6*

*while True:*

*print("Введите -1 если хотите завершить программу ")*

*n = int(input("Введите размер матрицы смежности: "))*

*if n == -1:*

*break*

*# Генерируем матрицу смежности*

*adjacency\_matrix = generate\_random\_adjacency\_matrix(n, p)*

*# Определяем размер графа*

*edge\_count = sum(sum(row) for row in adjacency\_matrix) // 2*

*# Определяем количество вершин*

*graph\_size = len(adjacency\_matrix)*

*# Строим матрицу инцидентности*

*incidence\_matrix = generate\_incidence\_matrix(adjacency\_matrix)*

*# Определяем количество рёбер на основе матрицы инцидентности*

*num\_edges\_from\_incidence = len(incidence\_matrix[0])*

*# Находим изолированные, концевые и доминирующие вершины на основе матрицы инцидентности*

*isolated\_vertices\_incidence = [i for i in range(graph\_size) if all(incidence\_matrix[i][j] == 0 for j in range(num\_edges\_from\_incidence))]*

*end\_vertices\_incidence = [i for i in range(graph\_size) if sum(incidence\_matrix[i][j] == 1 for j in range(num\_edges\_from\_incidence)) == 1]*

*dominating\_vertices\_incidence = [i for i in range(graph\_size) if sum(incidence\_matrix[i][j] == 1 for j in range(num\_edges\_from\_incidence)) == edge\_count]*

*# Выводим информацию о графе, включая вершины из матрицы инцидентности*

*print("Матрица смежности:")*

*for row in adjacency\_matrix:*

*print(row)*

*# Выводим также матрицу инцидентности*

*print("Матрица инцидентности:")*

*for row in incidence\_matrix:*

*print(row)*

*print("Размер графа (из матрицы смежности):", edge\_count)*

*print("Изолированные вершины (из матрицы инцидентности) :", isolated\_vertices\_incidence)*

*print("Концевые вершины (из матрицы инцидентности) :", end\_vertices\_incidence)*

*print("Доминирующие вершины (из мататрицы инцидентности) :", dominating\_vertices\_incidence)*

*print("Размер графа (из матрицы инцидентности):", num\_edges\_from\_incidence)*

Вывод: в ходе выполнения лабораторной работы мы научились определять характеристики графов.