Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)
Кафедра компьютерных систем в управлении и проектировании (КСУП)

# «ГЕНЕРАЦИЯ ГРАФА»

Отчет по лабораторной работе №2

По дисциплине «Дискретная математика»

Студенты гр. 573-2
Г.А Катренко.
С.С Масликов.
Т.Р Рафиков.
«4» мая 2024 г.

Профессор каф. КСУП,
доктор техн. наук, доцент каф. КСУП
Д.В. Кручинин
«4» мая 2024 г.

Томск 2024

## Введение

Целью данной лабораторной работы является реализация функций для вычисления матрицы метрик, радиуса, диаметра, центральных и периферийных вершин для заданного графа. Для эффективного возведения матрицы смежности в степень для получения матрицы метрик реализуется функция умножения матриц. Это позволит нам находить кратчайшие пути между парами вершин без необходимости полного перебора всех возможных путей.

# 1 ОСНОВНАЯ ЧАСТЬ

## Листинг:

```
import random
import networkx as nx
import matplotlib.pyplot as plt



#Функция генерации матрицы смежности
def generate_matrix(n, graf_type):
    matrix = [[0]*n for _ in range(n)]
    if graf_type == 'полный':
        for i in range(n):
            for j in range(n):
                if i !=j:
                    matrix[i][j] = 1
    elif graf_type == 'кратные ребра и петли':
        for i in range(n):
            for j in range(n):
                    matrix[i][j] = random.randint(0, 3)


    else:
        for i in range(n):
            for j in range(i, n):
                if i != j:
                    matrix[i][j]=random.randint(0, 3)
                else:
                    matrix[i][j]=random.randint(0, 1)


    for i in range(n):
        for j in range(i):
            matrix[i][j] = matrix[j][i]


    return matrix
```

```
#Функция отрисовки графа
def draw_graf(adj_matrix):
    Graf = nx.Graph()
    n = len(adj_matrix)

    for i in range(n):
        Graf.add_node(i)

    for i in range(n):
        for j in range(n):
            if adj_matrix[i][j] > 0:
                Graf.add_edge(i, j, weight=adj_matrix[i][j])

    pos = nx.spring_layout(Graf)
    labels = nx.get_edge_attributes(Graf, 'weight')

    nx.draw(Graf, pos, with_labels=True, node_size=700, node_color='red', font_size=10,
font_weight='bold')
    nx.draw_networkx_edge_labels(Graf, pos, edge_labels=labels)
    plt.show()

#Функция расчет матрицы инцедентности
def adjacency_to_incidence(adj_matrix):
    num_edges = 0
    num_vertex = len(adj_matrix)
    for i in range(num_vertex):
        for j in range(i+1, num_vertex):
            num_edges += adj_matrix[i][j]
    for i in range(num_vertex):
        num_edges += adj_matrix[i][i]
    incidence_matrix = [[0] * num_edges for _ in range(num_vertex)]
    edge_index=0
```

```
    for i in range(num_vertex):
        for j in range(i, num_vertex):
            if adj_matrix[i][j] != 0:
                for _ in range(adj_matrix[i][j]):
                    incidence_matrix[i][edge_index] += 1
                    incidence_matrix[j][edge_index] += 1

                    edge_index += 1

    return incidence_matrix

#Вводим размерность и тип графа
n=int(input("Введите размерность матрицы: "))
graf_type = input("Выберите тип графа (случайный/полный/кратные ребра и петли): ").lower()

#Нэйминг точек вершин(в нашем случе это будут цифровые обозначения вершин)
nodes=[]
for i in range(n):
    nodes.append(i)

#Вывод матрицы смежности
adj_matrix = generate_matrix(n, graf_type)
print("Матрица смежности:")
name_column_adjacent = ' '
for i in range(n):
    name_column_adjacent += (" "+str(i)+" ")
print(name_column_adjacent)
name_row_adjacent = 0
for row in adj_matrix:
        print(str(name_row_adjacent) + str(row))
        name_row_adjacent +=1
```

```
print('')
#Вывод матрицы инцедентности
incidence_matrix = adjacency_to_incidence(adj_matrix)
print("Матрица инцидентности:")
name_row_incidence=0
for row in incidence_matrix:
  print(str(name_row_incidence)+str(row))
  name_row_incidence+=1

def multiply_matrices(matrix1, matrix2):
  result = []
  for i in range(len(matrix1)):
    row = []
    for j in range(len(matrix2[0])):
      sum = 0
      for k in range(len(matrix2)):
        sum += matrix1[i][k] * matrix2[k][j]
      row.append(sum)
    result.append(row)
  return result

def power_matrix(matrix, power):
  if power == 0:
    return [[1, 0], [0, 1]]  # Единичная матрица
  result = matrix
  for _ in range(1, power):
    result = multiply_matrices(result, matrix)
  return result

#Функция перевода матрицы смежности в матрицу метрики
def convert_to_metric_matrix(adjacency_matrix):
  n = len(adjacency_matrix)
  S=adjacency_matrix.copy()
```

```
    ##Создаем матрицу S
    for i in range(n):
        S[i][i]=1
    k = 1
    M=[[None]*n for _ in range(n)]
    while True:
        M_updated = False
        S_power=power_matrix(S,k)
        for i in range(n):
            for j in range(n):
                if (M[i][j] is None) and S_power[i][j] != 0:
                    M[i][j] = k
                    M_updated = True

        if not M_updated:
            break

        k += 1

    for i in range(n):
        for j in range(n):
            if (M[i][j]) is None:
                M[i][j] = -1
    for i in range(n):
        M[i][i]=0

    return M



# Функция для определения диаметра, радиуса и центральных вершин графа
def nodes_from_metric_matrix(metric_matrix):
    diameter = 0  # Инициализация диаметра
    radius = 10**10  # Инициализация радиуса
```

```
    central_nodes = []  # Центральные вершины
    peripheral_nodes = []  # Периферийные вершины
    eccentricity_list = []  # Список эксцентриситетов вершин
    # Вычисление эксцентриситета для каждой вершины
    for n in range(len(metric_matrix)):
        eccentricity = 0
        for m in range(len(metric_matrix)):
            if metric_matrix[n][m] > eccentricity:
                eccentricity = metric_matrix[n][m]
        if eccentricity < radius:
            radius = eccentricity  # Обновление радиуса
        if eccentricity > diameter:
            diameter = eccentricity  # Обновление диаметра
        eccentricity_list.append(eccentricity)
    # Определение центральных и периферийных вершин
    for n in range(len(metric_matrix)):
        if eccentricity_list[n] == diameter:
            peripheral_nodes.append(nodes[n])
        if eccentricity_list[n] == radius:
            central_nodes.append(nodes[n])
    return diameter, radius, central_nodes, peripheral_nodes



print("")
print("Матрица метрики")
M_matrix=convert_to_metric_matrix(adj_matrix,)
for row in M_matrix:
    print(row)



graph_diameter, graph_radius, graph_center, graph_peripheral_nodes =
nodes_from_metric_matrix(M_matrix)
print(f'Диаметр: {graph_diameter}, радиус: {graph_radius}')
```

```
print(f'Центральные вершины: {graph_center}, периферийные вершины:
{graph_peripheral_nodes}')


#Отрисовка графа

draw_graf(adj_matrix)
```

## Заключение

В ходе лабораторной работы мы разработали программу в языке программирования Python, способную генерировать матрицы метрик для заданного графа. Также были успешно реализованы функции для вычисления радиуса, диаметра, центральных и периферийных вершин графа. Функция умножения матриц позволила эффективно возводить матрицу смежности в степень для получения матрицы метрик.

## Обратная связь

В ходе лабораторной работы мы столкнулись с некоторыми трудностями, такими как сложность в реализации функции для создания матрицы метрик.