Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №2.4

з дисципліни

«Алгоритми і структури даних»

Виконав:
Студент групи ІМ-34

Никифоров Артем Михайлович

Номер у списку групи:16

Перевірила:
Молчанова А. А.

Київ 2024

Постановка задачі:

1. Представити напрямлений та ненапрямлений графи із заданими параметрами так само, як у лабораторній роботі №3.

   Відмінність: коефіцієнт $k = 1.0 - n_3 * 0.01 - n_4 * 0.01 - 0.3$.

   Отже, матриця суміжності A_{dir} напрямленого графа за варіантом формується таким чином:

   1) встановлюється параметр (seed) генератора випадкових чисел, рівне номеру варіанту $n_1 n_2 n_3 n_4$;
   2) матриця розміром $n \cdot n$ заповнюється згенерованими випадковими числами в діапазоні $[0, 2.0)$;
   3) обчислюється коефіцієнт $k = 1.0 - n_3 * 0.01 - n_4 * 0.01 - 0.3$, кожен елемент матриці множиться на коефіцієнт k;
   4) елементи матриці округлюються: 0 — якщо елемент менший за 1.0, 1 — якщо елемент більший або дорівнює 1.0.

2. Обчислити:

   1) степені вершин напрямленого і ненапрямленого графів;
   2) напівстепені виходу та заходу напрямленого графа;
   3) чи є граф однорідним (регулярним), і якщо так, вказати степінь однорідності графа;

4) перелік висячих та ізольованих вершин.

Результати вивести у графічне вікно, консоль або файл.

3. ***Змінити матрицю*** A_{dir}, коефіцієнт $k = 1.0 - n_3 * 0.005 - n_4 * 0.005 - 0.27$.

4. Для нового орграфа обчислити:

   1) півстепені вершин;
   2) всі шляхи довжини 2 і 3;
   3) матрицю досяжності;
   4) матрицю сильної зв'язності;
   5) перелік компонент сильної зв'язності;
   6) граф конденсації.

Завдання для конкретного варіанту:

n1n2n3n4 = 3416

Кількість вершин = 10 + n3 = 11

Розташування колом з вершиною в центрі, тому що n4 = 6

Текст програми:

```
import turtle
import math
import random
import copy
import numpy as np

random.seed(3416)

matrix_dir = [[random.uniform(0.0, 2.0) for j in range(11)] for i in range(11)]
matrix_mod = [[random.uniform(0.0, 2.0) for j in range(11)] for i in range(11)]

k = 1.0 - 1 * 0.001 - 6 * 0.001 - 0.3
k2 = 1.0 - 1 * 0.005 - 6 * 0.005 - 0.27

for i in range(len(matrix_dir)):
    for j in range(len(matrix_dir[1])):
        matrix_dir[i][j] *= k
        if matrix_dir[i][j] < 1:
            matrix_dir[i][j] = 0
        else:
            matrix_dir[i][j] = 1
print("Матриця напрямленого графу:")
for i in range(len(matrix_dir)):
    print(matrix_dir[i], sep="\n")
print("")

matrix_undir = copy.deepcopy(matrix_dir)

for i in range(len(matrix_undir)):
    for j in range(len(matrix_undir[1])):
        if matrix_dir[i][j] == 1:
            matrix_undir[i][j] = 1
            matrix_undir[j][i] = 1
print("Матриця ненапрямленого графу:")
for i in range(len(matrix_undir)):
    print(matrix_undir[i], sep="\n")

for i in range(len(matrix_mod)):
    for j in range(len(matrix_mod[1])):
        matrix_mod[i][j] *= k
        if matrix_mod[i][j] < 1:
            matrix_mod[i][j] = 0
        else:
```

```
            matrix_mod[i][j] = 1

screen = turtle.Screen()
screen.setup(width=600, height=600)
screen.bgcolor("white")
turtle.speed(0)
turtle.hideturtle()

dir_check = int(input("choose graph to output 0 = undir, 1 = dir, 2 = mod, 3 = con
\n"))

def draw_circles_in_circle():
    radius = 200
    num_circles = 10
    angle = 360 / num_circles
    for i in range(num_circles):
        x = radius * math.cos(math.radians(angle * i))
        y = radius * math.sin(math.radians(angle * i))
        turtle.penup()
        turtle.color('black')
        turtle.goto(x, y)
        turtle.pendown()
        turtle.begin_fill()
        turtle.circle(20)
        turtle.end_fill()
        turtle.penup()
        turtle.goto(x, y + 10)
        turtle.color('white')
        turtle.write(str(i+1), align="center", font=("Arial", 12, "normal"))
        turtle.penup()

def draw_circles_con():
    radius = 200
    num_circles = 7
    angle = 360 / num_circles
    for i in range(num_circles):
        x = radius * math.cos(math.radians(angle * i))
        y = radius * math.sin(math.radians(angle * i))
        turtle.penup()
        turtle.color('black')
        turtle.goto(x, y)
        turtle.pendown()
        turtle.begin_fill()
        turtle.circle(20)
        turtle.end_fill()
```

```
        turtle.penup()
        turtle.goto(x, y + 10)
        turtle.color('white')
        turtle.write(str(i+1), align="center", font=("Arial", 12, "normal"))
        turtle.penup()

def draw_11():
    turtle.color('black')
    turtle.penup()
    turtle.goto(0, 0)
    turtle.pendown()
    turtle.begin_fill()
    turtle.circle(20)
    turtle.end_fill()
    turtle.color('white')
    turtle.penup()
    turtle.goto(0, 0 + 10)
    turtle.write(str(11), align="center", font=("Arial", 12, "normal"))
    turtle.penup()

matrix_con = [[0,1,0,0,0,0,0],
              [0,0,1,0,0,0,0],
              [0,0,0,1,0,0,0],
              [0,0,0,0,1,0,0],
              [0,0,0,0,0,1,0],
              [0,0,0,0,0,0,1],
              [0,0,0,0,0,0,0]]

def draw_edges_undir(matrix_undir): #lines of undir matrix
    num_vertices = len(matrix_undir)
    for i in range(num_vertices):
        for j in range(i, num_vertices):
            if matrix_undir[i][j] == 1:
                x1, y1 = get_vertex_position(i)
                x2, y2 = get_vertex_position(j)
                if i == j:
                    draw_circle(x1, y1)
                else:
                    if (i == 2 and j == 7) or (i == 7 and j == 2):
                        cursed_line(x1, y1, x2, y2)
                    else:
                        draw_line(x1, y1, x2, y2)

def draw_edges_dir(matrix_dir):  # lines of dir matrix
    num_vertices = len(matrix_dir)
```

```
    for i in range(num_vertices):
        for j in range(num_vertices):
            if matrix_dir[i][j] == 1:
                x1, y1 = get_vertex_position(i)
                x2, y2 = get_vertex_position(j)
                if i == j:
                    pass
                    draw_circle_dir(x1, y1)
                else:
                    if matrix_dir[j][i] == 1:
                        cursed_line_dir(x1, y1, x2, y2)
                    elif (i == 2 and j == 7) or (i == 7 and j == 2):
                        cursed_line_dir(x1, y1, x2, y2)
                    else:
                        draw_dir_line(x1, y1, x2, y2)

def draw_edges_mod(matrix_mod):  # lines of mod matrix
    num_vertices = len(matrix_mod)
    for i in range(num_vertices):
        for j in range(num_vertices):
            if matrix_mod[i][j] == 1:
                x1, y1 = get_vertex_position(i)
                x2, y2 = get_vertex_position(j)
                if i == j:
                    draw_circle_dir(x1, y1)
                else:
                    if matrix_mod[j][i] == 1:
                        cursed_line_mod(x1, y1, x2, y2)
                    elif (i == 6 and j == 1) or (i == 2 and j == 7) or (i == 0 and j == 5):
                        cursed_line_mod(x1, y1, x2, y2)
                    else:
                        draw_dir_line(x1, y1, x2, y2)

def draw_edges_con(matrix_con):  # lines of con matrix
    num_vertices = 7
    for i in range(num_vertices):
        for j in range(num_vertices):
            if matrix_con[i][j] == 1:
                x1, y1 = get_vertex_position_con(i)
                x2, y2 = get_vertex_position_con(j)
                if i == j:
                    draw_circle_dir(x1, y1)
                else:
                    if matrix_con[j][i] == 1:
                        cursed_line(x1, y1, x2, y2)
```

```
                else:
                    draw_dir_line(x1, y1, x2, y2)

print("Степені вершин напрямленого і ненапрямленого графів")
print("Напрмямлений граф:")
deg_dir = []
for i in range (11):
    count = 0
    for j in range (11):
        if matrix_dir[j][i] == 1:
            count += 1
        if matrix_dir[i][j] == 1:
            count +=1
    deg_dir.append(count)
for i in range(len(deg_dir)):
    print(i+1,"-",deg_dir[i])
print("")

print("Ненапрмямлений граф:")
deg_undir = []
for i in range (11):
    count = 0
    for j in range (11):
        if matrix_undir[i][j] == 1:
            count += 1
        if i == j and matrix_undir[i][j] == 1:
            count += 1
    deg_undir.append(count)
for i in range(len(deg_undir)):
    print(i+1,"-",deg_undir[i])
print("")

print("Напівстепені виходу та заходу напрямленого графа:")
in_deg = [0] * len(matrix_dir)
out_deg = [0] * len(matrix_dir)
dir_matrix_len = len(matrix_dir)
for i in range(dir_matrix_len):
    for j in range(dir_matrix_len):
        if matrix_dir[i][j] == 1:
            out_deg[i] += 1
            in_deg[j] += 1
for i in range(dir_matrix_len):
    print(i+1," - виходу:",out_deg[i], "заходу: ",in_deg[i])
print("")
```

```
print("Чи є граф однорідним(регулярним), і якщо так, вказати степінь
однорідності графа")

def is_regular_graph(degrees):
    return all(degree == degrees[0] for degree in degrees)
if is_regular_graph(deg_dir):
    print("Граф є однорідним. Степінь однорідності:", deg_dir[0])
else:
    print("Граф не є однорідним.")
print("")

print("Перелік висячих та ізольованих вершин:")
def find_hang_iso_vert(out_deg, in_deg):
    hang_vert = []
    iso_vert = []
    for i in range(len(out_deg)):
        if out_deg[i] + in_deg[i] == 1:
            hang_vert.append(i+1)
        elif out_deg[i] + in_deg[i] == 0:
            iso_vert.append(i+1)
    return hang_vert, iso_vert
hang_vert, iso_vert = find_hang_iso_vert(out_deg, in_deg)
print("Висячі вершини:", hang_vert)
print("Ізольовані вершини:", iso_vert)
print("")

print("Другий орграф:")
for i in range(len(matrix_mod)):
    print(matrix_mod[i], sep="\n")
print("")

print("Півстепені другого орграфу:")
in_deg_mod = [0] * len(matrix_mod)
out_deg_mod = [0] * len(matrix_mod)
mod_matrix_len = len(matrix_mod)
for i in range(len(matrix_mod)):
    for j in range(len(matrix_mod)):
        if matrix_mod[i][j] == 1:
            out_deg_mod[i] += 1
            in_deg_mod[j] += 1
for i in range(dir_matrix_len):
    print(i+1," - виходу:",out_deg_mod[i], "заходу: ",in_deg_mod[i])
print("")

print("Квадрат матриці суміжності")
```

```
print(np.linalg.matrix_power(matrix_mod,2))
print("")

def paths2(matrix_mod):
    paths_length_2 = []
    n = len(matrix_mod)
    for i in range(n):
        for k in range(n):
            if matrix_mod[i][k] == 1:
                for j in range(n):
                    if matrix_mod[k][j] == 1:
                        paths_length_2.append((i+1, k+1, j+1))
    return paths_length_2

paths_length_2_mod = paths2(matrix_mod)
print("Шляхи довжиною 2 модифікованого графу:")
for path in paths_length_2_mod:
    print(path)
print("")

print("Куб матриці суміжності")
print(np.linalg.matrix_power(matrix_mod,3))
print("")

def paths3(matrix_mod):
    paths_length_3 = []
    n = len(matrix_mod)
    for i in range(n):
        for k in range(n):
            if matrix_mod[i][k] == 1:
                for j in range(n):
                    if matrix_mod[k][j] == 1:
                        for l in range(n):
                            if matrix_mod[j][l] == 1:
                                paths_length_3.append((i+1, k+1, j+1, l+1))
    return paths_length_3

paths_length_3_mod = paths3(matrix_mod)
print("Шляхи довжиною 3 модифікованого графу:")
for path in paths_length_3_mod:
    print(path)
print("")

print("Матриця досяжності")
I = [[1,0,0,0,0,0,0,0,0,0,0],
```

```
    [0,1,0,0,0,0,0,0,0,0,0],
    [0,0,1,0,0,0,0,0,0,0,0],
    [0,0,0,1,0,0,0,0,0,0,0],
    [0,0,0,0,1,0,0,0,0,0,0],
    [0,0,0,0,0,1,0,0,0,0,0],
    [0,0,0,0,0,0,1,0,0,0,0],
    [0,0,0,0,0,0,0,1,0,0,0],
    [0,0,0,0,0,0,0,0,1,0,0],
    [0,0,0,0,0,0,0,0,0,1,0],
    [0,0,0,0,0,0,0,0,0,0,1]
]
I = np.array(I)
E = np.array(matrix_mod)
E2 = E.dot(E)
E3 = E2.dot(E)
E4 = E3.dot(E)
E5 = E4.dot(E)
E6 = E5.dot(E)
E7 = E6.dot(E)
E8 = E7.dot(E)
E9 = E8.dot(E)
E10 = E9.dot(E)
R = I + E + E2 + E3 + E4 + E5 + E6 + E7 + E8 + E9 + E10
for i in range(11):
    for j in range(11):
        if R[i][j] >= 1:
            R[i][j] = 1
        else:
            R[i][j] = 0
print(R)
S = copy.deepcopy(R)
print("")

print("Квадрат матриці досяжності")
print(np.linalg.matrix_power(R,2))
print("")

print("Транспонована матриця досяжності")
R_trans = R.transpose()
print(R_trans)
print("")

print("Матриця сильної зв'язності")
for i in range(len(R)):
    for j in range(len(R[0])):
```

```
        S[i][j] = R[i][j] * R_trans[i][j]
print(S)
print("")

print("Компоненти сильної зв'язності")
def strong_components(matrix):
    n = matrix.shape[1]
    columns = [matrix[:, i] for i in range(n)]
    unique_columns = []
    groups = []

    for i, col in enumerate(columns):
        found = False
        for j, unique_col in enumerate(unique_columns):
            if np.array_equal(col, unique_col):
                groups[j].append(i)
                found = True
                break
        if not found:
            unique_columns.append(col)
            groups.append([i])

    return groups
groups = strong_components(S)
for i, group in enumerate(groups):
    print(f"Component {i+1}: Columns {group}")

def cursed_line(x1, y1, x2, y2):
    turtle.penup()
    turtle.goto(x1, y1 + 15)
    turtle.pendown()
    turtle.color('red')
    turtle.width(1)

    if x1 == x2:
        control_offset = 15
        cx1, cy1 = (x1 + x2) / 2, (y1 + y2) / 2
        if y1 < y2:
            cy1 += control_offset
        else:
            cy1 -= control_offset
        turtle.goto(cx1, cy1 + 15)
    else:
        control_offset = 15
        cx1, cy1 = (x1 + x2) / 2, (1.2* y1 + y2) / 2
```

```
        if x1 < x2:
            cx1 += control_offset
        else:
            cx1 -= control_offset
        turtle.goto(cx1, cy1 + 15)
    turtle.goto(x2, y2 + 15)

def cursed_line_dir(x1, y1, x2, y2):
    turtle.penup()
    turtle.goto(x1, y1 + 15)
    turtle.pendown()
    turtle.color('red')
    turtle.width(1)

    if x1 == x2:
        control_offset = 15
        cx1, cy1 = (x1 + x2) / 2, (y1 + y2) / 2
        if y1 < y2:
            cy1 += control_offset
        else:
            cy1 -= control_offset
        turtle.goto(cx1, cy1 + 15)
    else:
        control_offset = 15
        cx1, cy1 = (x1 + x2) / 2, (1.2*y1 + y2) / 2
        if x1 < x2:
            cx1 += control_offset
        else:
            cx1 -= control_offset
        turtle.goto(cx1, cy1 + 15)
    turtle.goto(x2, y2 + 15)
    turtle_angle = math.degrees(math.atan2(y2 - y1, x2 - x1))
    turtle.setheading(turtle_angle)
    turtle.stamp()

def cursed_line_mod(x1, y1, x2, y2):
    turtle.penup()
    turtle.goto(x1, y1 + 15)
    turtle.pendown()
    turtle.color('red')
    turtle.width(1)

    if x1 == x2:
        control_offset = 15
        cx1, cy1 = (x1 + x2) / 2, (y1 + y2) / 2
```

```
        if y1 < y2:
            cy1 += control_offset
        else:
            cy1 -= control_offset
        turtle.goto(cx1, cy1 + 15)
    else:
        control_offset = 15
        cx1, cy1 = (x1 + x2) / 2, (1.2*y1 + y2) / 2
        if x1 < x2:
            cx1 += control_offset
        else:
            cx1 -= control_offset
        turtle.goto(cx1, cy1 + 70)
    turtle.goto(x2, y2 + 15)
    turtle_angle = math.degrees(math.atan2(y2 - y1, x2 - x1))
    turtle.setheading(turtle_angle)
    turtle.stamp()

def draw_circle(x, y):
    turtle.color('black')
    turtle.penup()
    turtle.goto(x, y+20)
    turtle.pendown()
    turtle.circle(30)
    turtle.penup()

def draw_circle_dir(x, y):
    turtle.color('blue')
    turtle.penup()
    turtle.goto(x, y+20)
    turtle.pendown()
    turtle.circle(30)
    turtle.stamp()
    turtle.penup()

def get_vertex_position(vertex_index):
    if vertex_index == 10:
        return 0, 0
    else:
        radius = 200
        num_vertices = 10
        angle = 360 / num_vertices
        x = radius * math.cos(math.radians(angle * vertex_index))
        y = radius * math.sin(math.radians(angle * vertex_index))
        return x, y
```

```
def get_vertex_position_con(vertex_index):
    radius = 200
    num_vertices = 7
    angle = 360 / num_vertices
    x = radius * math.cos(math.radians(angle * vertex_index))
    y = radius * math.sin(math.radians(angle * vertex_index))
    return x, y

def draw_line(x1, y1, x2, y2):
    turtle.penup()
    turtle.goto(x1, y1+15)
    turtle.pendown()
    turtle.color('black')
    turtle.width(1)
    turtle.goto(x2, y2+15)

def draw_dir_line(x1, y1, x2, y2):
    turtle.penup()
    turtle.goto(x1, y1+15)
    turtle.pendown()
    turtle.color('blue')
    turtle.width(1)
    end_x_line = x1 + 0.95 * (x2 - x1)
    end_y_line = y1 + 0.95 * (y2 - y1)
    turtle.goto(end_x_line, end_y_line + 15)
    turtle_angle = math.degrees(math.atan2(end_y_line - y1, end_x_line - x1))
    turtle.setheading(turtle_angle)
    turtle.stamp()

if dir_check == 1:
    draw_circles_in_circle()
    draw_11()
    draw_edges_dir(matrix_dir)
    num_vertices = 10
if dir_check == 0:
    draw_edges_undir(matrix_undir)
    draw_circles_in_circle()
    num_vertices = 10
    draw_11()
if dir_check == 2:
    draw_circles_in_circle()
    draw_11()
    draw_edges_mod(matrix_mod)
    num_vertices = 10
```

```
if dir_check == 3:
    draw_circles_con()
    draw_edges_con(matrix_con)

screen.exitonclick()
```

Матриця суміжності напрямленого графу:

```
Матриця напрямленого графу:
[0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
[1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0]
[0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1]
[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1]
[0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0]
[0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1]
[0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1]
[1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0]
[0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0]
```

Матриця суміжності ненапрямленого графу:

```
Матриця ненапрямленого графу:
[0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0]
[1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
[1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0]
[1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1]
[0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1]
[0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1]
[0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1]
[1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1]
[1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0]
[0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1]
[0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0]
```

Степені напрямленого графу:

```
Степені вершин напрямленого і ненапрямленого графів
Напрямлений граф:
1 - 5
2 - 2
3 - 7
4 - 9
5 - 4
6 - 6
7 - 4
8 - 9
9 - 6
10 - 6
11 - 6
```

Степені ненапрямленого графу:

```
Ненапрямлений граф:
1 - 5
2 - 2
3 - 7
4 - 7
5 - 4
6 - 4
7 - 3
8 - 8
9 - 5
10 - 5
11 - 6
```

Півстепені напрямленого графу:

```
Напівстепені виходу та заходу напрямленого графа:
1  - виходу: 3 заходу:  2
2  - виходу: 1 заходу:  1
3  - виходу: 4 заходу:  3
4  - виходу: 5 заходу:  4
5  - виходу: 2 заходу:  2
6  - виходу: 2 заходу:  4
7  - виходу: 3 заходу:  1
8  - виходу: 5 заходу:  4
9  - виходу: 3 заходу:  3
10  - виходу: 2 заходу:  4
11  - виходу: 2 заходу:  4
```

Результат перевірки на однорідність:

```
Чи є граф однорідним(регулярним), і якщо так, вказати степінь однорідності графа
Граф не є однорідним.
```

Перелік висячих та ізольованих вершин:

```
Перелік висячих та ізольованих вершин:
Висячі вершини: []
Ізольовані вершини: []
```

Матриця другого орграфа:

```
Другий орграф:
[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1]
[1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1]
[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
[1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0]
[0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0]
[0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0]
[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
[0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1]
[0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0]
```

Півстепені другого орграфу:

```
1  - виходу: 1 заходу:  2
2  - виходу: 2 заходу:  5
3  - виходу: 6 заходу:  2
4  - виходу: 1 заходу:  0
5  - виходу: 4 заходу:  1
6  - виходу: 2 заходу:  7
7  - виходу: 4 заходу:  1
8  - виходу: 3 заходу:  5
9  - виходу: 1 заходу:  3
10  - виходу: 4 заходу:  2
11  - виходу: 3 заходу:  3
```

Шляхи довжиною 2:

(1, 6, 8)

(1, 6, 9)

(2, 6, 8)

(2, 6, 9)

(2, 11, 2)

(2, 11, 8)

(2, 11, 9)

(3, 1, 6)

(3, 2, 6)

(3, 2, 11)

(3, 6, 8)

(3, 6, 9)

(3, 8, 6)

(3, 8, 8)

(3, 8, 9)

(3, 10, 2)

(3, 10, 6)

(3, 10, 8)

(3, 10, 11)

(3, 11, 2)

(3, 11, 8)

(3, 11, 9)

(4, 6, 8)

(4, 6, 9)

(5, 1, 6)

(5, 2, 6)

(5, 2, 11)

(5, 3, 1)

(5, 3, 2)

(5, 3, 6)

(5, 3, 8)

(5, 3, 10)

(5, 3, 11)

(5, 7, 2)

(5, 7, 3)

(5, 7, 5)

(5, 7, 10)

(6, 8, 6)

(6, 8, 8)

(6, 8, 9)

(6, 9, 6)

(7, 2, 6)

(7, 2, 11)

(7, 3, 1)

(7, 3, 2)

(7, 3, 6)

(7, 3, 8)

(7, 3, 10)

(7, 3, 11)

(7, 5, 1)

(7, 5, 2)

(7, 5, 3)

(7, 5, 7)

(7, 10, 2)

(7, 10, 6)

(7, 10, 8)

(7, 10, 11)

(8, 6, 8)

(8, 6, 9)

(8, 8, 6)

(8, 8, 8)

(8, 8, 9)

(8, 9, 6)

(9, 6, 8)

(9, 6, 9)

(10, 2, 6)

(10, 2, 11)

(10, 6, 8)

(10, 6, 9)

(10, 8, 6)

(10, 8, 8)

(10, 8, 9)

(10, 11, 2)

(10, 11, 8)

(10, 11, 9)

(11, 2, 6)

(11, 2, 11)

(11, 8, 6)

(11, 8, 8)

(11, 8, 9)

(11, 9, 6)

Шляхи довжиною 3:

(1, 6, 8, 6)

(1, 6, 8, 8)

(1, 6, 8, 9)

(1, 6, 9, 6)

(2, 6, 8, 6)

(2, 6, 8, 8)

(2, 6, 8, 9)

(2, 6, 9, 6)

(2, 11, 2, 6)

(2, 11, 2, 11)

(2, 11, 8, 6)

(2, 11, 8, 8)

(2, 11, 8, 9)

(2, 11, 9, 6)

(3, 1, 6, 8)

(3, 1, 6, 9)

(3, 2, 6, 8)

(3, 2, 6, 9)

(3, 2, 11, 2)

(3, 2, 11, 8)

(3, 2, 11, 9)

(3, 6, 8, 6)

(3, 6, 8, 8)

(3, 6, 8, 9)

(3, 6, 9, 6)

(3, 8, 6, 8)

(3, 8, 6, 9)

(3, 8, 8, 6)

(3, 8, 8, 8)

(3, 8, 8, 9)

(3, 8, 9, 6)

(3, 10, 2, 6)

(3, 10, 2, 11)

(3, 10, 6, 8)

(3, 10, 6, 9)

(3, 10, 8, 6)

(3, 10, 8, 8)

(3, 10, 8, 9)

(3, 10, 11, 2)

(3, 10, 11, 8)

(3, 10, 11, 9)

(3, 11, 2, 6)

(3, 11, 2, 11)

(3, 11, 8, 6)

(3, 11, 8, 8)

(3, 11, 8, 9)

(3, 11, 9, 6)

(4, 6, 8, 6)

(4, 6, 8, 8)

(4, 6, 8, 9)

(4, 6, 9, 6)

(5, 1, 6, 8)

(5, 1, 6, 9)

(5, 2, 6, 8)

(5, 2, 6, 9)

(5, 2, 11, 2)

(5, 2, 11, 8)

(5, 2, 11, 9)

(5, 3, 1, 6)

(5, 3, 2, 6)

(5, 3, 2, 11)

(5, 3, 6, 8)

(5, 3, 6, 9)

(5, 3, 8, 6)

(5, 3, 8, 8)

(5, 3, 8, 9)

(5, 3, 10, 2)

(5, 3, 10, 6)

(5, 3, 10, 8)

(5, 3, 10, 11)

(5, 3, 11, 2)

(5, 3, 11, 8)

(5, 3, 11, 9)

(5, 7, 2, 6)

(5, 7, 2, 11)

(5, 7, 3, 1)

(5, 7, 3, 2)

(5, 7, 3, 6)

(5, 7, 3, 8)

(5, 7, 3, 10)

(5, 7, 3, 11)

(5, 7, 5, 1)

(5, 7, 5, 2)

(5, 7, 5, 3)

(5, 7, 5, 7)

(5, 7, 10, 2)

(5, 7, 10, 6)

(5, 7, 10, 8)

(5, 7, 10, 11)

(6, 8, 6, 8)

(6, 8, 6, 9)

(6, 8, 8, 6)

(6, 8, 8, 8)

(6, 8, 8, 9)

(6, 8, 9, 6)

(6, 9, 6, 8)

(6, 9, 6, 9)

(7, 2, 6, 8)

(7, 2, 6, 9)

(7, 2, 11, 2)

(7, 2, 11, 8)

(7, 2, 11, 9)

(7, 3, 1, 6)

(7, 3, 2, 6)

(7, 3, 2, 11)

(7, 3, 6, 8)

(7, 3, 6, 9)

(7, 3, 8, 6)

(7, 3, 8, 8)

(7, 3, 8, 9)

(7, 3, 10, 2)

(7, 3, 10, 6)

(7, 3, 10, 8)

(7, 3, 10, 11)

(7, 3, 11, 2)

(7, 3, 11, 8)

(7, 3, 11, 9)

(7, 5, 1, 6)

(7, 5, 2, 6)

(7, 5, 2, 11)

(7, 5, 3, 1)

(7, 5, 3, 2)

(7, 5, 3, 6)

(7, 5, 3, 8)

(7, 5, 3, 10)

(7, 5, 3, 11)

(7, 5, 7, 2)

(7, 5, 7, 3)

(7, 5, 7, 5)

(7, 5, 7, 10)

(7, 10, 2, 6)

(7, 10, 2, 11)

(7, 10, 6, 8)

(7, 10, 6, 9)

(7, 10, 8, 6)

(7, 10, 8, 8)

(7, 10, 8, 9)

(7, 10, 11, 2)

(7, 10, 11, 8)

(7, 10, 11, 9)

(8, 6, 8, 6)

(8, 6, 8, 8)

(8, 6, 8, 9)

(8, 6, 9, 6)

(8, 8, 6, 8)

(8, 8, 6, 9)

(8, 8, 8, 6)

(8, 8, 8, 8)

(8, 8, 8, 9)

(8, 8, 9, 6)

(8, 9, 6, 8)

(8, 9, 6, 9)

(9, 6, 8, 6)

(9, 6, 8, 8)

(9, 6, 8, 9)

(9, 6, 9, 6)

(10, 2, 6, 8)

(10, 2, 6, 9)

(10, 2, 11, 2)

(10, 2, 11, 8)

(10, 2, 11, 9)

(10, 6, 8, 6)

(10, 6, 8, 8)

(10, 6, 8, 9)

(10, 6, 9, 6)

(10, 8, 6, 8)

(10, 8, 6, 9)

(10, 8, 8, 6)

(10, 8, 8, 8)

(10, 8, 8, 9)

(10, 8, 9, 6)

(10, 11, 2, 6)

(10, 11, 2, 11)

(10, 11, 8, 6)

(10, 11, 8, 8)

(10, 11, 8, 9)

(10, 11, 9, 6)

(11, 2, 6, 8)

(11, 2, 6, 9)

(11, 2, 11, 2)

(11, 2, 11, 8)

(11, 2, 11, 9)

(11, 8, 6, 8)

(11, 8, 6, 9)

(11, 8, 8, 6)

(11, 8, 8, 8)

(11, 8, 8, 9)

(11, 8, 9, 6)

(11, 9, 6, 8)

(11, 9, 6, 9)

Матриця досяжності:

```
Матриця досяжності
[[1 0 0 0 0 1 0 1 1 0 0]
 [0 1 0 0 0 1 0 1 1 0 1]
 [1 1 1 0 0 1 0 1 1 1 1]
 [0 0 0 1 0 1 0 1 1 0 0]
 [1 1 1 0 1 1 1 1 1 1 1]
 [0 0 0 0 0 1 0 1 1 0 0]
 [1 1 1 0 1 1 1 1 1 1 1]
 [0 0 0 0 0 1 0 1 1 0 0]
 [0 0 0 0 0 1 0 1 1 0 0]
 [0 1 0 0 0 1 0 1 1 1 1]
 [0 1 0 0 0 1 0 1 1 0 1]]
```

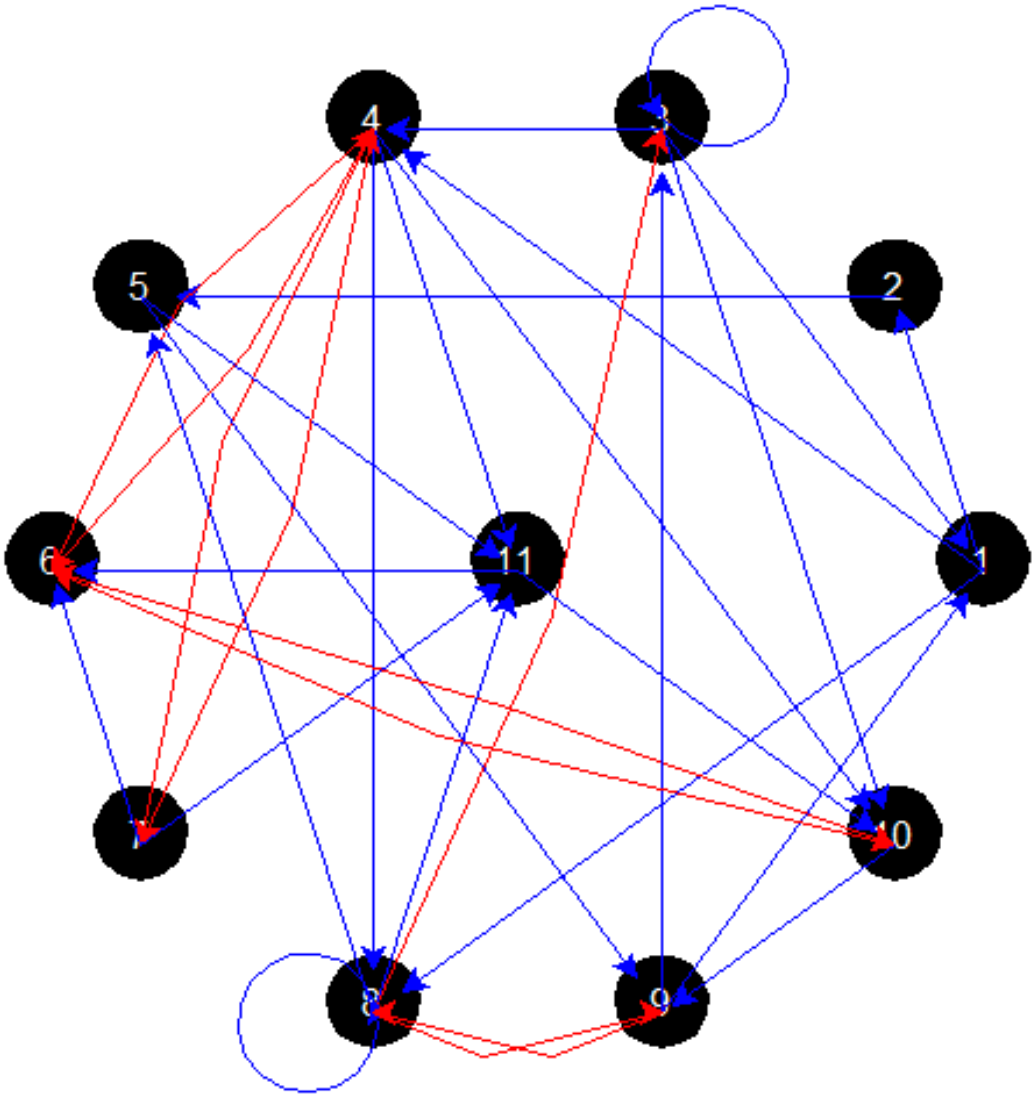
Матриця сильної зв’язності:

```
Матриця сильної зв'язності
[[1 0 0 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0 0 1]
 [0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 1 0 0 0 0]
 [0 0 0 0 0 1 0 1 1 0 0]
 [0 0 0 0 1 0 1 0 0 0 0]
 [0 0 0 0 0 1 0 1 1 0 0]
 [0 0 0 0 0 1 0 1 1 0 0]
 [0 0 0 0 0 0 0 0 0 1 0]
 [0 1 0 0 0 0 0 0 0 0 1]]
```

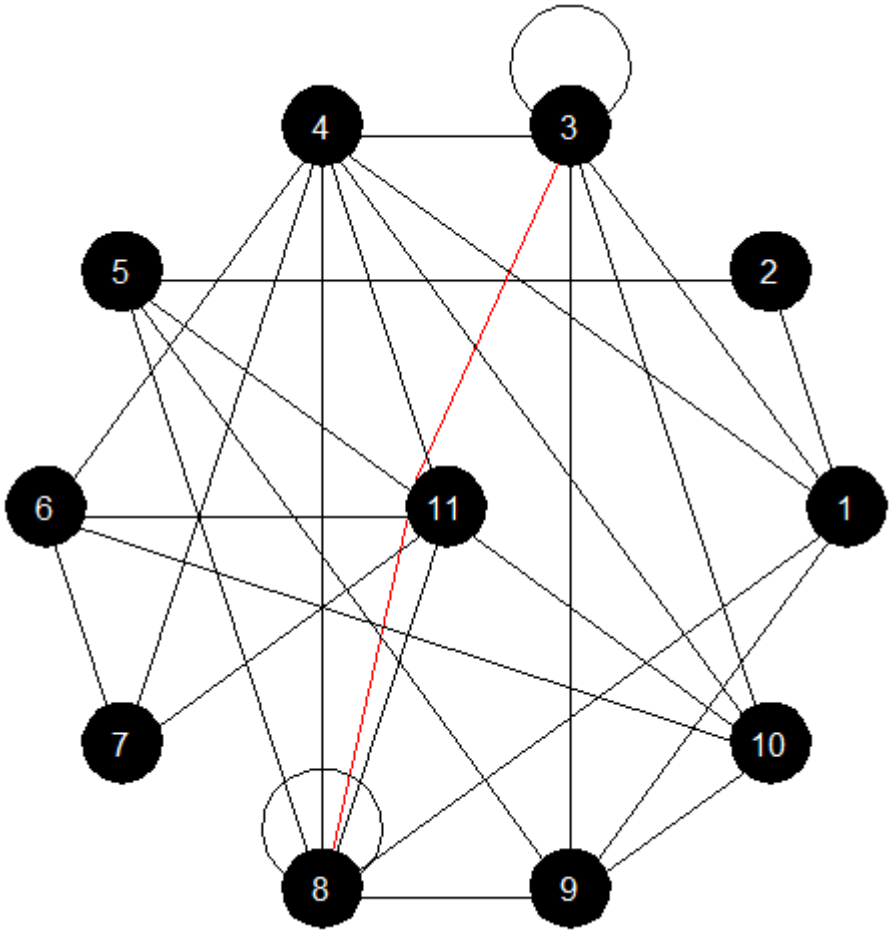
Перелік компонент сильної зв’язності:

```
Компоненти сильної зв'язності
Компонент 1: Колонка [0]
Компонент 2: Колонка [1, 10]
Компонент 3: Колонка [2]
Компонент 4: Колонка [3]
Компонент 5: Колонка [4, 6]
Компонент 6: Колонка [5, 7, 8]
Компонент 7: Колонка [9]
```

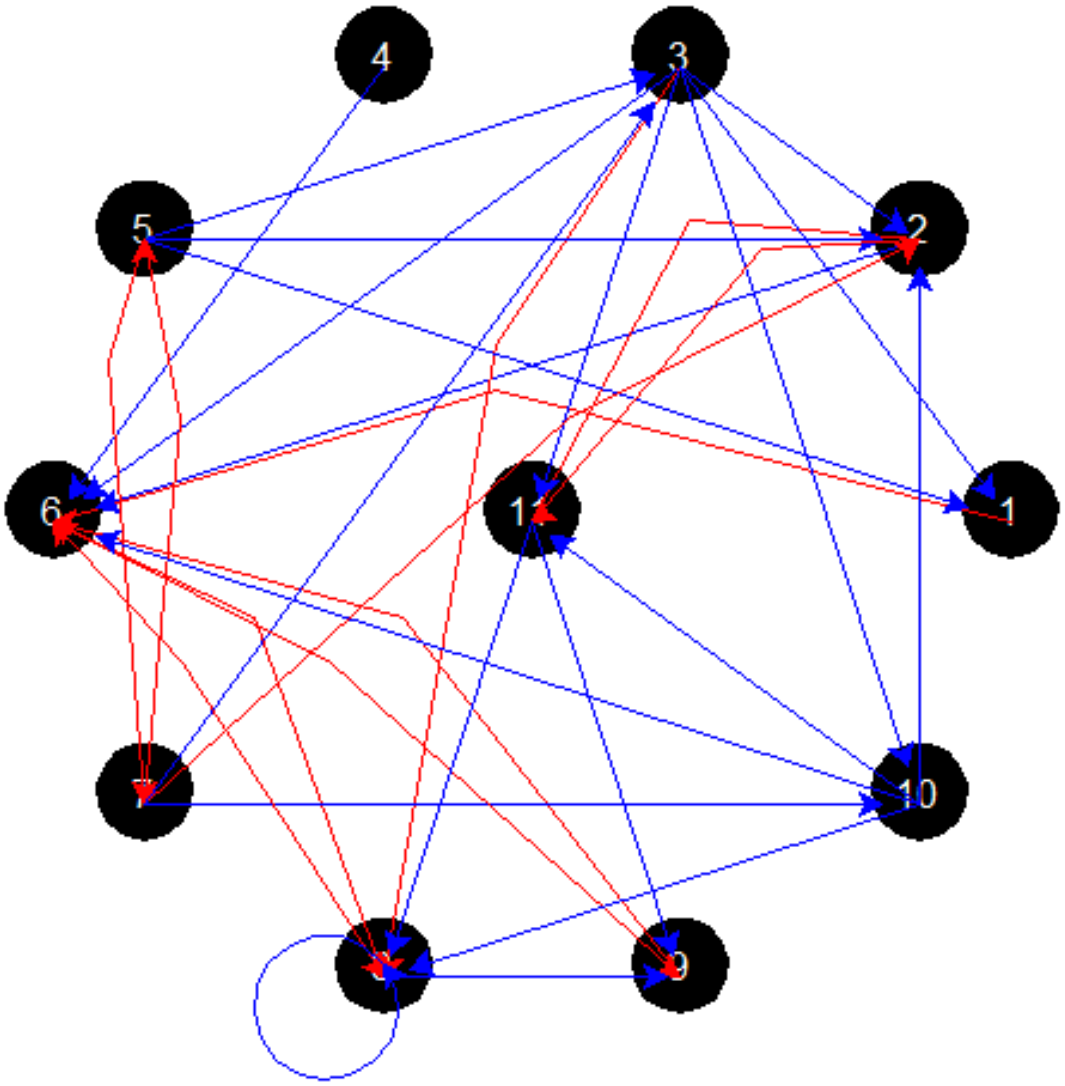
Скріншот заданого орієнтованого графу:



Скріншот заданого неорієнтованого графу:



Скріншот заданого модифікованого графу:



Скріншот заданого графу конденсації: