**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 №2.3

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 у програмі напрямлений і ненапрямлений графи з заданими параметрами:
   - кількість вершин $n$;
   - розміщення вершин;
   - матриця суміжності A.
2. Створити програму для формування зображення напрямленого і ненапрямленого графів у графічному вікні.

Згадані вище параметри графа задаються на основі чотиризначного номера варіанту $n_1n_2n_3n_4$, де $n_1n_2$ це десяткові цифри номера групи, а $n_3n_4$ — десяткові цифри номера варіанту, який був у студента для двох попередніх робіт (див. таблицю з поточними оцінками з АСД, надану викладачем на початку поточного семестру).

***Кількість вершин*** $n$ дорівнює $10 + n_3$.

***Розміщення вершин***:

- колом при $n_4 = 0, 1$;
- квадратом (прямокутником) при $n_4 = 2, 3$;
- трикутником при $n_4 = 4, 5$;
- колом з вершиною в центрі при $n_4 = 6, 7$;
- квадратом (прямокутником) з вершиною в центрі при $n_4 = 8, 9$.

***Матриця суміжності*** $A_{dir}$ напрямленого графа за варіантом формується таким чином:

1) встановлюється параметр (seed) генератора випадкових чисел, рівний номеру варіанту $n_1n_2n_3n_4$ — детальніше див. с. 12;
2) матриця розміром $n \cdot n$ заповнюється згенерованими випадковими числами в діапазоні $[0, 2.0)$;
3) обчислюється коефіцієнт $k = 1.0 - n_3 * 0.02 - n_4 * 0.005 - 0.25$;
4) кожен елемент матриці множиться на коефіцієнт $k$;
5) елементи матриці округлюються: 0 — якщо елемент менший за 1.0, 1 — якщо елемент більший або дорівнює 1.0.

Матриця суміжності $A_{undir}$ ненапрямленого графа одержується з матриці $A_{dir}$:

$$a_{dir_{i,j}} = 1 \Rightarrow a_{undir_{i,j}} = 1, a_{undir_{j,i}} = 1.$$

**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

random.seed(3416)

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

k = 1.0 - 1 * 0.002 - 6 * 0.005 - 0.25

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

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

matrix2 = copy.deepcopy(matrix_dir)

for i in range(len(matrix2)):
```

```
    for j in range(len(matrix2[1])):
        if matrix_dir[i][j] == 1:
            matrix2[i][j] = 1
            matrix2[j][i] = 1

for i in range(len(matrix2)):
    print(matrix2[i], sep="\n")

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 \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_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()

def draw_edges_undir(matrix2): #lines of undir matrix
    num_vertices = len(matrix2)
    for i in range(num_vertices):
        for j in range(i, num_vertices):
```

```
            if matrix2[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:
                    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 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 = 17
        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 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 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
```

```
else:
    draw_edges_undir(matrix2)
    draw_circles_in_circle()
    num_vertices = 10
    draw_11()

screen.exitonclick()
```

**Матриця суміжності напрямленого графа:**

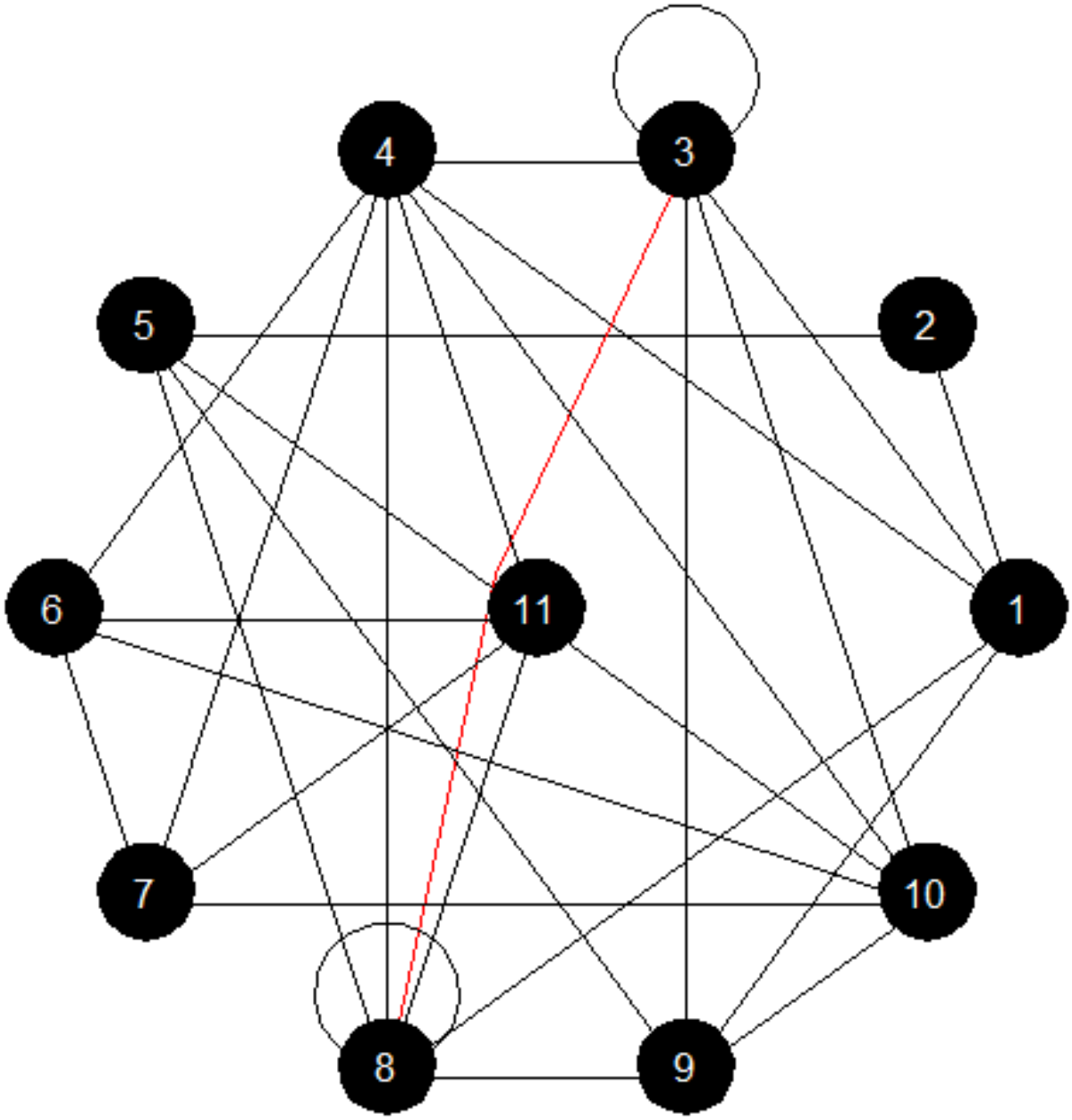
```
[0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
[1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0]
[0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1]
[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1]
[0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0]
[0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1]
[0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1]
[1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0]
[0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0]
```

**Матриця суміжності ненапрямленого графа:**

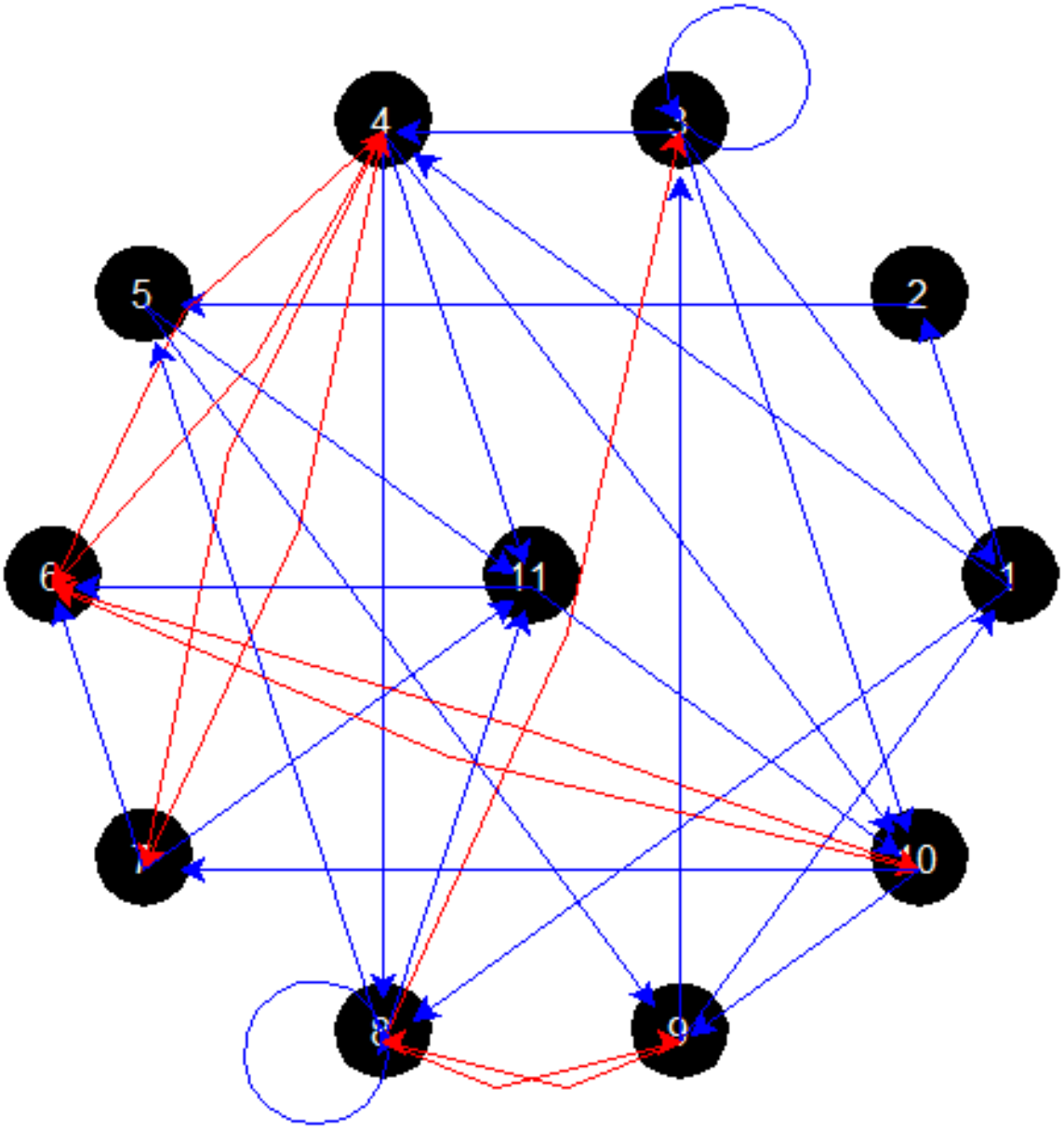
```
[0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0]
[1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
[1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0]
[1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1]
[0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1]
[0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1]
[0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1]
[1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1]
[1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0]
[0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1]
[0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0]
```

**Скриншоти напрямленого і ненапрямленого графів, які побудовані за варіантом:**

**Ненапрямлений граф:**



**Напрямлений граф:**




**Висновок:**

Під час виконання цієї лабораторної роботи, я набув практичних навичок представлення графів у комп’ютері та ознайомився з принципами роботи ОС.