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 №4

з дисципліни

«Алгоритми і структури даних»

Виконав:	Перевірив:
Студент групи ІМ-42	Сергієнко А. М.
Лобань Михайло Юрійович	
номер у списку групи: 20	

Київ 2025

Загальна постановка завдання

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

Відмінність: коефіцієнт k = 1.0 - n3 * 0.01 - n4 * 0.01 - 0.3.

Отже, матриця суміжності Adir напрямленого графа за варіантом формується таким чином:

1) встановлюється параметр (seed) генератора випадкових чисел, рівне номеру варіанту n1n2n3n4;

2) матриця розміром n * n заповнюється згенерованими випадковими

числами в діапазоні [0, 2.0);

3) обчислюється коефіцієнт k = 1.0 - n3 * 0.01 - n4 * 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 Adir, коефіцієнт k = 1.0-n3 * 0.005-n4 * 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.

Результати вивести у графічне вікно, в консоль або файл.

Шляхи довжиною 2 і 3 слід шукати за матрицями A^2 і A^3, відповідно. Як результат вивести перелік шляхів, включно з усіма проміжними вершинами, через які проходить шлях.

Матрицю досяжності та компоненти сильної зв’язності слід шукати за допомогою операції транзитивного замикання. У переліку компонент слід вказати, які вершини належать до кожної компоненти.

Граф конденсації вивести у графічне вікно.

Завдання за варіантом

Варіант 20

n1n2n3n4 = 4220

Кількість вершин – 10 + 2 = 12

Розміщення вершин – колом, n4 = 0

Текст програм

Файл 1, graph.py:

```
import math
import tkinter as tk
import random


n3 = 2
n4 = 0
vertexes = n3 + 10


variant = 4220
random.seed(variant)


k1 = 1 - n3 * 0.01 - n4 * 0.01 - 0.3
k2 = 1 - n3 * 0.005 - n4 * 0.005 - 0.27
```

```
def calculate_element(k):
    return math.floor(random.random() * 2 * k)

matrix_dir = [[0 for _ in range(vertexes)] for _ in range(vertexes)]
matrix_undir = [[0 for _ in range(vertexes)] for _ in range(vertexes)]

for i in range(vertexes):
    for j in range(vertexes):
        matrix_dir[i][j] = calculate_element(k1)

for i in range(vertexes):
    for j in range(vertexes):
        matrix_undir[i][j] = matrix_dir[i][j] or matrix_dir[j][i]

root = tk.Tk()
root.title("Graph")

canvas = tk.Canvas(root, width=800, height=800, bg="white")
canvas.pack()

mid_x = mid_y = 400
angle = math.pi * 2 / vertexes
R = 20

def get_x(i):
    return mid_x + math.sin(i * angle) * 200
```

```
def get_y(i):
    return mid_y - math.cos(i * angle) * 200

def rotate_around_center(x, y, cx, cy, theta):
    x -= cx
    y -= cy
    new_x = x * math.cos(theta) - y * math.sin(theta) + cx
    new_y = x * math.sin(theta) + y * math.cos(theta) + cy
    return new_x, new_y

def draw_graph(matrix, vertexes, is_directed):
    for i in range(vertexes):
        x = get_x(i) - R
        y = get_y(i) - R
        canvas.create_oval(x, y, x + 2 * R, y + 2 * R, fill="white")
        canvas.create_text(x + R, y + R, text=str(i + 1), font=("Montserrat", 12))

    for i in range(vertexes):
        for j in range(vertexes):
            if matrix[i][j] == 1:
                if i == j:
                    cx, cy = get_x(i), get_y(i)
                    theta = i * angle

                    cx += R * math.sin(theta)
                    cy -= R * math.cos(theta)

                    dx = 3 * R / 4
                    dy = R * (1 - math.sqrt(7)) / 4

                    p1 = (cx - dx, cy - dy)
                    p2 = (cx - 3 * dx / 2, cy - R / 2)
                    p3 = (cx + 3 * dx / 2, cy - R / 2)
                    p4 = (cx + dx, cy - dy)

                    p1 = rotate_around_center(p1[0], p1[1], cx, cy, theta)
                    p2 = rotate_around_center(p2[0], p2[1], cx, cy, theta)
                    p3 = rotate_around_center(p3[0], p3[1], cx, cy, theta)
                    p4 = rotate_around_center(p4[0], p4[1], cx, cy, theta)

                    canvas.create_line(p1[0], p1[1], p2[0], p2[1], width=2)
                    canvas.create_line(p2[0], p2[1], p3[0], p3[1], width=2)
                    if (is_directed):
                        canvas.create_line(p3[0], p3[1], p4[0], p4[1], width=2,
arrow=tk.LAST)
                    else:
                        canvas.create_line(p3[0], p3[1], p4[0], p4[1], width=2)
                else:
                    x1, y1 = get_x(i), get_y(i)
                    x2, y2 = get_x(j), get_y(j)

                    dx, dy = x2 - x1, y2 - y1
                    length = math.sqrt(dx ** 2 + dy ** 2)

                    dx /= length
                    dy /= length
```

```
                    x1 += dx * R
                    y1 += dy * R
                    x2 -= dx * R
                    y2 -= dy * R

                    if (is_directed):
                        canvas.create_line(x1, y1, x2, y2, width=2, arrow=tk.LAST)
                    else:
                        canvas.create_line(x1, y1, x2, y2, width=2)
    root.mainloop()
```

Файл 2, utils.py:

```
def print_array(arr, text, separator):
    print(text, end = " ")
    if(len(arr) == 0):
        print("no such vertexes", end=" ")
        print()
    else:
        for i in range(len(arr)):
            print(arr[i], end=separator)
        print()

def print_matrix(matrix):
    for row in matrix:
        for element in row:
            print(element, end=" ")
        print()

def matrix_multiply(A, B):
    n = len(A)
    result = [[0] * n for _ in range(n)]
    for i in range(n):
        for j in range(n):
            for k in range(n):
                result[i][j] += A[i][k] * B[k][j]
    return result


def matrix_add(A, B):
    n = len(A)
    result = [[0] * n for _ in range(n)]
    for i in range(n):
        for j in range(n):
            result[i][j] = A[i][j] + B[i][j]
    return result
```

Файл 3, main.py:

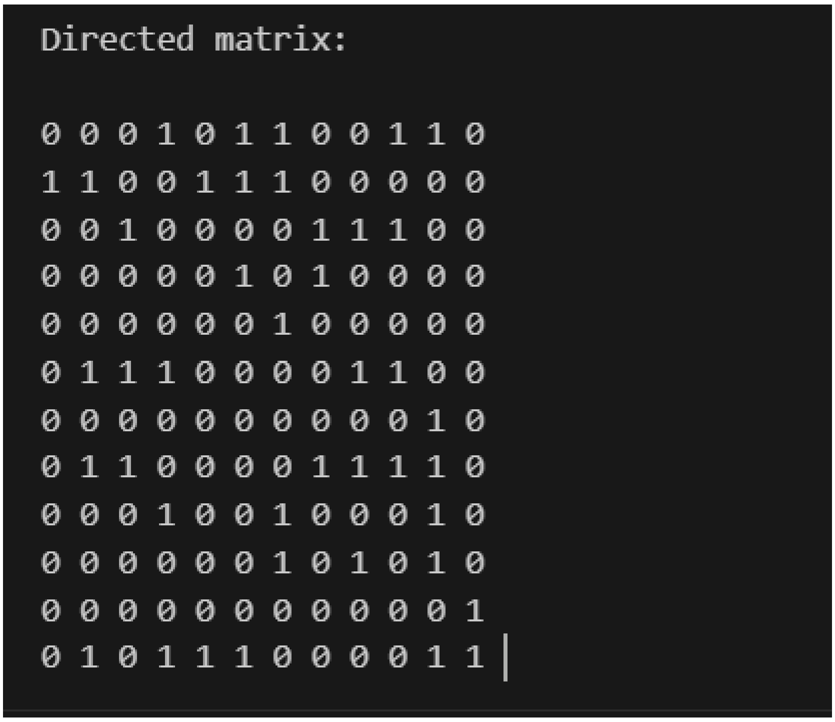
```
from graph import *
from utils import *


def get_graph_info(matrix, isDirected):
    in_degrees = []
    out_degrees = []
    vertex_degrees = []

    print()

    if (isDirected):
        for i in range(vertexes):
            in_degree = 0
            for j in range(vertexes):
```

```
            if(matrix[j][i] == 1):
                in_degree += 1
        in_degrees.append(in_degree)

    print_array(in_degrees, "Vertex degrees (IN):", " ")

    for i in range(vertexes):
        out_degree = 0
        for j in range(vertexes):
            if(matrix[i][j] == 1):
                out_degree += 1
        out_degrees.append(out_degree)

    print_array(out_degrees, "Vertex degrees (OUT):", " ")

    for i in range(len(in_degrees)):
        vertex_degrees.append(in_degrees[i] + out_degrees[i])

    print_array(vertex_degrees, "Vertex degrees:", " ")
else:
    for i in range(vertexes):
        vertex_degree = 0
        for j in range(vertexes):
            if(matrix[i][j] == 1):
                if(i == j):
                    vertex_degree += 2
                else:
                    vertex_degree += 1
            vertex_degrees.append(vertex_degree)
        print_array(vertex_degrees, "Vertex degrees:", " ")

    is_regular = True
    for i in range(len(vertex_degrees)):
        if(vertex_degrees[0] != vertex_degrees[i]): is_regular = False

    if(is_regular):
        print_array([is_regular], "Is regular:", " ")
        print_array([vertex_degrees[0]], "Regularity degree:", " ")
    else:
        print_array([is_regular], "Is regular:", " ")

    index = 1
    leap_vertexes = []
    isolated_vertexes = []
    for i in vertex_degrees:
        if (i == 0):
            isolated_vertexes.append(index)
        elif (i == 1):
            leap_vertexes.append(index)
        index += 1

    print_array(leap_vertexes, "Leap vertexes:", " ")
    print_array(isolated_vertexes, "Isolated vertexes:", " ")

def log_paths_length_2(matrix):
    vertex_count = len(matrix)
    squared = matrix_multiply(matrix, matrix)
    paths = []

    for i in range(vertex_count):
        for j in range(vertex_count):
            if squared[i][j] > 0:
                for k in range(vertex_count):
                    if matrix[i][k] == 1 and matrix[k][j] == 1:
                        paths.append(f"{i+1} -> {k+1} -> {j+1}")

    return paths

def log_paths_length_3(matrix):
    vertex_count = len(matrix)
    cubed = matrix_multiply(matrix_multiply(matrix, matrix), matrix)
    paths = []

    for i in range(vertex_count):
        for j in range(vertex_count):
            if cubed[i][j] > 0:
                for k in range(vertex_count):
                    for l in range(vertex_count):
                        if matrix[i][k] == 1 and matrix[k][l] == 1 and matrix[l][j] == 1:
                            paths.append(f"{i+1} -> {k+1} -> {l+1} -> {j+1}")

    return paths

def reachability_matrix(matrix):
    vertex_count = len(matrix)
    reach = [row[:] for row in matrix]
    matrix_to_power = [row[:] for row in matrix]

    for _ in range(vertex_count - 1):
        matrix_to_power = matrix_multiply(matrix_to_power, matrix)
        reach = matrix_add(reach, matrix_to_power)

    for i in range(vertex_count):
        for j in range(vertex_count):
            if reach[i][j] > 0:
                reach[i][j] = 1
    return reach

def connectivity_matrix(matrix):
    n = len(matrix)
    connectivity = [[0] * n for _ in range(n)]
    for i in range(n):
        for j in range(n):
            if matrix[i][j] == 1 and matrix[j][i] == 1:
                connectivity[i][j] = 1
    return connectivity

def find_strong_components(conn):
    visited = [False] * len(conn)
    components = []

    for v in range(len(conn)):
        if not visited[v]:
            component = []
            for u in range(len(conn)):
                if conn[v][u] == 1:
                    component.append(u + 1)
                    visited[u] = True

            if not component:
                component = [v + 1]

            components.append(component)

    return components

def condensed_graph_matrix(graph, sccs):
    n = len(sccs)
    condensed = [[0] * n for _ in range(n)]

    vertex_to_scc = {}
    for scc_index, scc in enumerate(sccs):
        for vertex in scc:
            vertex_to_scc[vertex] = scc_index

    for i in range(len(graph)):
        for j in range(len(graph)):
            if graph[i][j] == 1:
                scc_i = vertex_to_scc[i + 1]
                scc_j = vertex_to_scc[j + 1]
                if scc_i != scc_j:
                    condensed[scc_i][scc_j] = 1

    return condensed

def get_graph_new_info(matrix):
    in_degrees = []
    out_degrees = []

    print()
    for i in range(vertexes):
        in_degree = 0
        for j in range(vertexes):
            if(matrix[j][i] == 1):
                in_degree += 1
        in_degrees.append(in_degree)
    print_array(in_degrees, "Vertex degrees (IN):", " ")
    for i in range(vertexes):
        out_degree = 0
        for j in range(vertexes):
            if(matrix[i][j] == 1):
                out_degree += 1
        out_degrees.append(out_degree)
    print_array(out_degrees, "Vertex degrees (OUT):", " ")

    print_array(log_paths_length_2(matrix), "All paths of 2:", "; ")
    print_array(log_paths_length_3(matrix), "All paths of 3:", "; ")
    print("\nReachability matrix:\n")
    print_matrix(reachability_matrix(matrix))

    print("\nConnectivity matrix:\n")
    print_matrix(connectivity_matrix(reachability_matrix(matrix)))

    scc = find_strong_components(connectivity_matrix(reachability_matrix(matrix)))

    print("\nStrong connectivity components:\n")
    for i in range(len(scc)):
        print(f"{i + 1}) {scc[i]}", end = "\n")

    condensed = condensed_graph_matrix(new_matrix_dir, scc)

    print("\nCondensed graph matrix:\n")
    print_matrix(condensed)

    # draw_graph(condensed, len(condensed), 1)

print("\nDirected matrix:\n")
print_matrix(matrix_dir)
get_graph_info(matrix_dir, True)

print("\nUndirected matrix:\n")
print_matrix(matrix_undir)
get_graph_info(matrix_undir, False)

new_matrix_dir = [[0 for _ in range(vertexes)] for _ in range(vertexes)]
```

```
for i in range(vertexes):
    for j in range(vertexes):
        new_matrix_dir[i][j] = calculate_element(k2)


print("\nUpdated directed matrix:\n")
print_matrix(new_matrix_dir)
get_graph_new_info(new_matrix_dir)


draw_graph(matrix_dir, vertexes, 1)
# draw_graph(matrix_undir, vertexes, 0)
# draw_graph(new_matrix_dir, vertexes, 1)
```

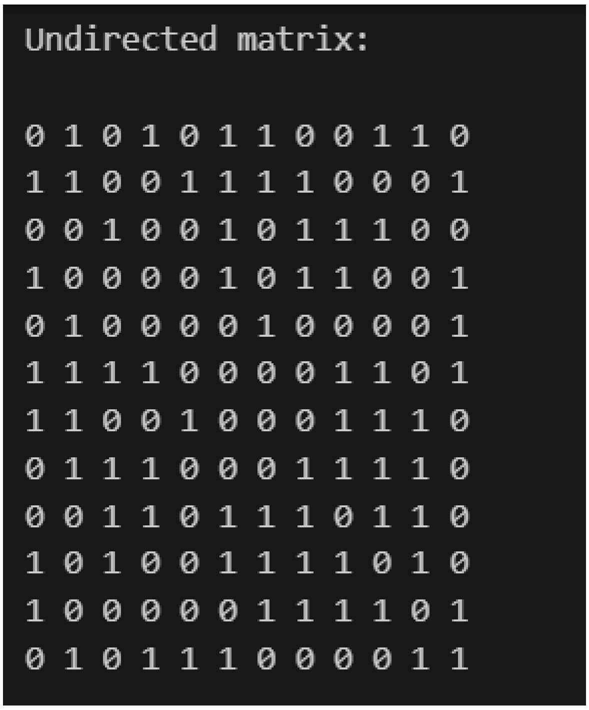
Матриці суміжності

Матриця суміжності напрямленого графа:



```
Directed matrix:

0 0 0 1 0 1 1 0 0 1 1 0
1 1 0 0 1 1 1 0 0 0 0 0
0 0 1 0 0 0 0 1 1 1 0 0
0 0 0 0 0 1 0 1 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0
0 1 1 1 0 0 0 0 1 1 0 0
0 0 0 0 0 0 0 0 0 0 1 0
0 1 1 0 0 0 0 1 1 1 1 0
0 0 0 1 0 0 1 0 0 0 1 0
0 0 0 0 0 0 1 0 1 0 1 0
0 0 0 0 0 0 0 0 0 0 0 1
0 1 0 1 1 1 0 0 0 0 1 1
```

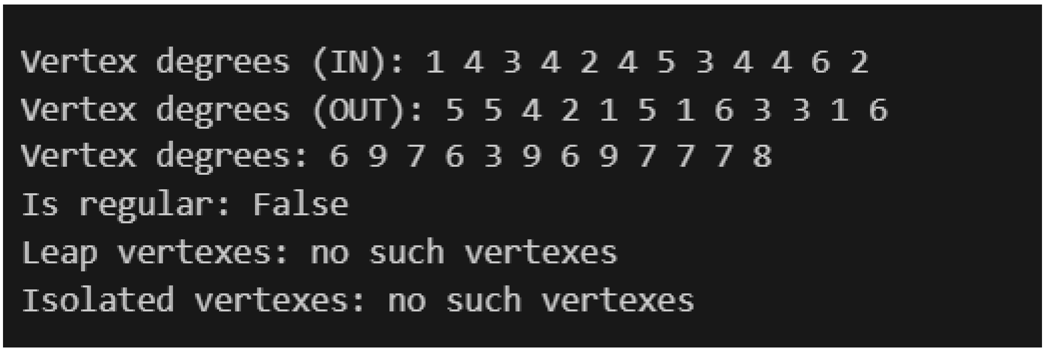
Матриця суміжності ненапрямленого графа:



```
Undirected matrix:

0 1 0 1 0 1 1 0 0 1 1 0
1 1 0 0 1 1 1 1 0 0 0 1
0 0 1 0 0 1 0 1 1 1 0 0
1 0 0 0 0 1 0 1 1 0 0 1
0 1 0 0 0 0 1 0 0 0 0 1
1 1 1 1 0 0 0 0 1 1 0 1
1 1 0 0 1 0 0 0 1 1 1 0
0 1 1 1 0 0 0 1 1 1 1 0
0 0 1 1 0 1 1 1 0 1 1 0
1 0 1 0 0 1 1 1 1 0 1 0
1 0 0 0 0 0 1 1 1 1 0 1
0 1 0 1 1 1 0 0 0 0 1 1
```

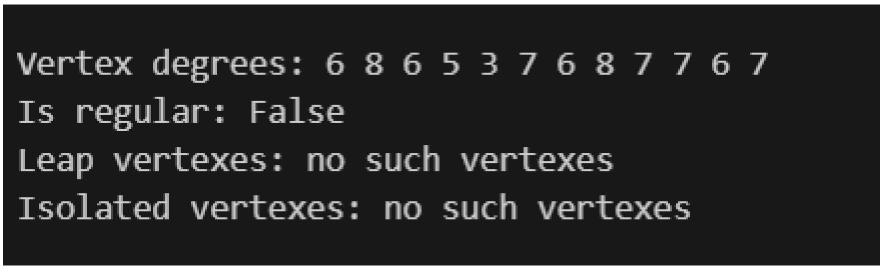
Характеристика вершин

Для напрямленого графа:



```
Vertex degrees (IN): 1 4 3 4 2 4 5 3 4 4 6 2
Vertex degrees (OUT): 5 5 4 2 1 5 1 6 3 3 1 6
Vertex degrees: 6 9 7 6 3 9 6 9 7 7 7 8
Is regular: False
Leap vertexes: no such vertexes
Isolated vertexes: no such vertexes
```

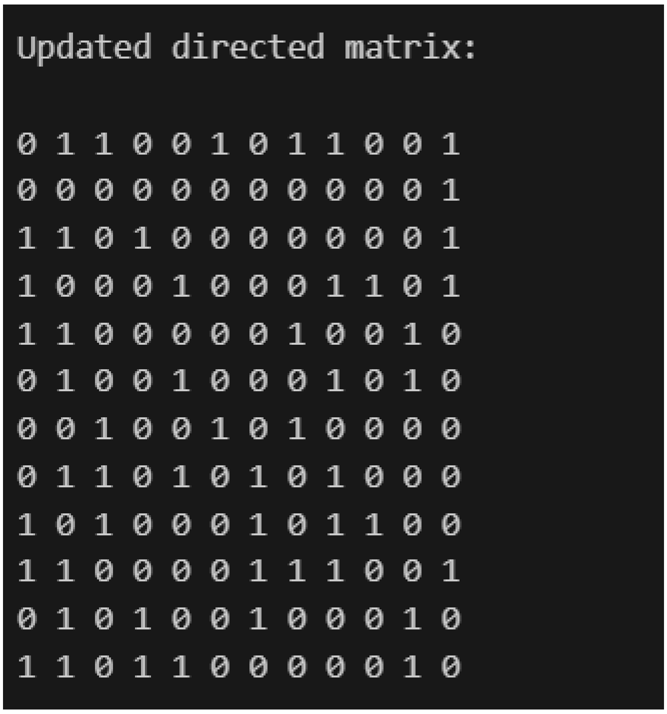
Для ненапрямленого графа:



```
Vertex degrees: 6 8 6 5 3 7 6 8 7 7 6 7
Is regular: False
Leap vertexes: no such vertexes
Isolated vertexes: no such vertexes
```

Модифікований граф

Матриця суміжності:



```
Updated directed matrix:

0 1 1 0 0 1 0 1 1 0 0 1
0 0 0 0 0 0 0 0 0 0 0 1
1 1 0 1 0 0 0 0 0 0 0 1
1 0 0 0 1 0 0 0 1 1 0 1
1 1 0 0 0 0 0 1 0 0 1 0
0 1 0 0 1 0 0 0 1 0 1 0
0 0 1 0 0 1 0 1 0 0 0 0
0 1 1 0 1 0 1 0 1 0 0 0
1 0 1 0 0 0 1 0 1 1 0 0
1 1 0 0 0 0 1 1 1 0 0 1
0 1 0 1 0 0 1 0 0 0 1 0
1 1 0 1 1 0 0 0 0 0 1 0
```

Усі шляхи довжиною 2:

All paths of 2: 1 -> 3 -> 1; 1 -> 9 -> 1; 1 -> 12 -> 1; 1 -> 3 -> 2; 1 -> 6 -> 2; 1 -> 8 -> 2; 1 -> 12 -> 2; 1 -> 8 -> 3; 1 -> 9 -> 3; 1 -> 3 -> 4; 1 -> 12 -> 4; 1 -> 6 -> 5; 1 -> 8 -> 5; 1 -> 12 -> 5; 1 -> 8 -> 7; 1 -> 9 -> 7; 1 -> 6 -> 9; 1 -> 8 -> 9; 1 -> 9 -> 9; 1 -> 9 -> 10; 1 -> 6 -> 11; 1 -> 12 -> 11; 1 -> 2 -> 12; 1 -> 3 -> 12; 2 -> 12 -> 1; 2 -> 12 -> 2; 2 -> 12 -> 4; 2 -> 12 -> 5; 2 -> 12 -> 11; 3 -> 4 -> 1; 3 -> 12 -> 1; 3 -> 1 -> 2; 3 -> 12 -> 2; 3 -> 1 -> 3; 3 -> 12 -> 4; 3 -> 4 -> 5; 3 -> 12 -> 5; 3 -> 1 -> 6; 3 -> 1 -> 8; 3 -> 1 -> 9; 3 -> 4 -> 9; 3 -> 4 -> 10; 3 -> 12 -> 11; 3 -> 1 -> 12; 3 -> 2 -> 12; 3 -> 4 -> 12; 4 -> 5 -> 1; 4 -> 9 -> 1; 4 -> 10 -> 1; 4 -> 12 -> 1; 4 -> 1 -> 2; 4 -> 5 -> 2; 4 -> 10 -> 2; 4 -> 12 -> 2; 4 -> 1 -> 3; 4 -> 9 -> 3; 4 -> 12 -> 4; 4 -> 12 -> 5; 4 -> 1 -> 6; 4 -> 9 -> 7; 4 -> 10 -> 7; 4 -> 1 -> 8; 4 -> 5 -> 8; 4 -> 10 -> 8; 4 -> 1 -> 9; 4 -> 9 -> 9; 4 -> 10 -> 9; 4 -> 9 -> 10; 4 -> 5 -> 11; 4 -> 12 -> 11; 4 -> 1 -> 12; 4 -> 10 -> 12; 5 -> 1 -> 2; 5 -> 8 -> 2; 5 -> 11 -> 2; 5 -> 1 -> 3; 5 -> 8 -> 3; 5 -> 11 -> 4; 5 -> 8 -> 5; 5 -> 1 -> 6; 5 -> 8 -> 7; 5 -> 11 -> 7; 5 -> 1 -> 8; 5 -> 1 -> 9; 5 -> 8 -> 9; 5 -> 11 -> 11; 5 -> 1 -> 12; 5 -> 2 -> 12; 6 -> 5 -> 1; 6 -> 9 -> 1; 6 -> 5 -> 2; 6 -> 11 -> 2; 6 -> 9 -> 3; 6 -> 11 -> 4; 6 -> 9 -> 7; 6 -> 11 -> 7; 6 -> 5 -> 8; 6 -> 9 -> 9; 6 -> 9 -> 10; 6 -> 5 -> 11; 6 -> 11 -> 11; 6 -> 2 -> 12; 7 -> 3 -> 1; 7 -> 3 -> 2; 7 -> 6 -> 2; 7 -> 8 -> 2; 7 -> 8 -> 3; 7 -> 3 -> 4; 7 -> 6 -> 5; 7 -> 8 -> 5; 7 -> 8 -> 7; 7 -> 6 -> 9; 7 -> 8 -> 9; 7 -> 6 -> 11; 7 -> 3 -> 12; 8 -> 3 -> 1; 8 -> 5 -> 1; 8 -> 9 -> 1; 8 -> 3 -> 2; 8 -> 5 -> 2; 8 -> 7 -> 3; 8 -> 9 -> 3; 8 -> 3 -> 4; 8 -> 7 -> 6; 8 -> 9 -> 7; 8 -> 5 -> 8; 8 -> 7 -> 8; 8 -> 9 -> 9; 8 -> 9 -> 10; 8 -> 5 -> 11; 8 -> 2 -> 12; 8 -> 3 -> 12; 9 -> 3 -> 1; 9 -> 9 -> 1; 9 -> 10 -> 1; 9 -> 1 -> 2; 9 -> 3 -> 2; 9 -> 10 -> 2; 9 -> 1 -> 3; 9 -> 7 -> 3; 9 -> 9 -> 3; 9 -> 3 -> 4; 9 -> 1 -> 6; 9 -> 7 -> 6; 9 -> 9 -> 7; 9 -> 10 -> 7; 9 -> 1 -> 8; 9 -> 7 -> 8; 9 -> 10 -> 8; 9 -> 1 -> 9; 9 -> 9 -> 9; 9 -> 10 -> 9; 9 -> 9 ->

10; 9 -> 1 -> 12; 9 -> 3 -> 12; 9 -> 10 -> 12; 10 -> 9 -> 1; 10 -> 12 -> 1; 10 -> 1 -> 2; 10 -> 8 -> 2; 10 -> 12 -> 2; 10 -> 1 -> 3; 10 -> 7 -> 3; 10 -> 8 -> 3; 10 -> 9 -> 3; 10 -> 12 -> 4; 10 -> 8 -> 5; 10 -> 12 -> 5; 10 -> 1 -> 6; 10 -> 7 -> 6; 10 -> 8 -> 7; 10 -> 9 -> 7; 10 -> 1 -> 8; 10 -> 7 -> 8; 10 -> 1 -> 9; 10 -> 8 -> 9; 10 -> 9 -> 9; 10 -> 9 -> 10; 10 -> 12 -> 11; 10 -> 1 -> 12; 10 -> 2 -> 12; 11 -> 4 -> 1; 11 -> 11 -> 2; 11 -> 7 -> 3; 11 -> 11 -> 4; 11 -> 4 -> 5; 11 -> 7 -> 6; 11 -> 11 -> 7; 11 -> 7 -> 8; 11 -> 4 -> 9; 11 -> 4 -> 10; 11 -> 11 -> 11; 11 -> 2 -> 12; 11 -> 4 -> 12; 12 -> 4 -> 1; 12 -> 5 -> 1; 12 -> 1 -> 2; 12 -> 5 -> 2; 12 -> 11 -> 2; 12 -> 1 -> 3; 12 -> 11 -> 4; 12 -> 4 -> 5; 12 -> 1 -> 6; 12 -> 11 -> 7; 12 -> 1 -> 8; 12 -> 5 -> 8; 12 -> 1 -> 9; 12 -> 4 -> 9; 12 -> 4 -> 10; 12 -> 5 -> 11; 12 -> 11 -> 11; 12 -> 1 -> 12; 12 -> 2 -> 12; 12 -> 4 -> 12;

Усі шляхи довжиною 3:

All paths of 3: 1 -> 2 -> 12 -> 1; 1 -> 3 -> 4 -> 1; 1 -> 3 -> 12 -> 1; 1 -> 6 -> 5 -> 1; 1 -> 6 -> 9 -> 1; 1 -> 8 -> 3 -> 1; 1 -> 8 -> 5 -> 1; 1 -> 8 -> 9 -> 1; 1 -> 9 -> 3 -> 1; 1 -> 9 -> 9 -> 1; 1 -> 9 -> 10 -> 1; 1 -> 12 -> 4 -> 1; 1 -> 12 -> 5 -> 1; 1 -> 2 -> 12 -> 2; 1 -> 3 -> 1 -> 2; 1 -> 3 -> 12 -> 2; 1 -> 6 -> 5 -> 2; 1 -> 6 -> 11 -> 2; 1 -> 8 -> 3 -> 2; 1 -> 8 -> 5 -> 2; 1 -> 9 -> 1 -> 2; 1 -> 9 -> 3 -> 2; 1 -> 9 -> 10 -> 2; 1 -> 12 -> 1 -> 2; 1 -> 12 -> 5 -> 2; 1 -> 12 -> 11 -> 2; 1 -> 3 -> 1 -> 3; 1 -> 6 -> 9 -> 3; 1 -> 8 -> 7 -> 3; 1 -> 8 -> 9 -> 3; 1 -> 9 -> 1 -> 3; 1 -> 9 -> 7 -> 3; 1 -> 9 -> 9 -> 3; 1 -> 12 -> 1 -> 3; 1 -> 2 -> 12 -> 4; 1 -> 3 -> 12 -> 4; 1 -> 6 -> 11 -> 4; 1 -> 8 -> 3 -> 4; 1 -> 9 -> 3 -> 4; 1 -> 12 -> 11 -> 4; 1 -> 2 -> 12 -> 5; 1 -> 3 -> 4 -> 5; 1 -> 3 -> 12 -> 5; 1 -> 12 -> 4 -> 5; 1 -> 3 -> 1 -> 6; 1 -> 8 -> 7 -> 6; 1 -> 9 -> 1 -> 6; 1 -> 9 -> 7 -> 6; 1 -> 12 -> 1 -> 6; 1 -> 6 -> 9 -> 7; 1 -> 6 -> 11 -> 7; 1 -> 8 -> 9 -> 7; 1 -> 9 -> 9 -> 7; 1 -> 9 -> 10 -> 7; 1 -> 12 -> 11 -> 7; 1 -> 3 -> 1 -> 8; 1 -> 6 -> 5 -> 8; 1 -> 8 -> 5 -> 8; 1 -> 8 -> 7 -> 8; 1 -> 9 -> 1 -> 8; 1 -> 9 -> 7 -> 8; 1 -> 9 -> 10 -> 8; 1 -> 12 -> 1 -> 8; 1 -> 12 -> 5 -> 8; 1 -> 3 -> 1 -> 9; 1 -> 3 -> 4 -> 9; 1 -> 6 -> 9 -> 9; 1 -> 8 -> 9 -> 9; 1 -> 9 -> 1 -> 9; 1 -> 9 -> 9 -> 9; 1 -> 9 -> 10 -> 9; 1 -> 12 -> 1 -> 9; 1 -> 12 -> 4 -> 9; 1 -> 3 -> 4 -> 10; 1 -> 6 -> 9 -> 10; 1 -> 8 -> 9 -> 10; 1 -> 9 -> 9 -> 10; 1 -> 12 -> 4 -> 10; 1 -> 2 -> 12 -> 11; 1 -> 3 -> 12 -> 11; 1 -> 6 -> 5 -> 11; 1 -> 6 -> 11 -> 11; 1 -> 8 -> 5 -> 11; 1 -> 12 -> 5 -> 11; 1 -> 12 -> 11 -> 11; 1 -> 3 -> 1 -> 12; 1 -> 3 -> 2 -> 12; 1 -> 3 -> 4 -> 12; 1 -> 6 -> 2 -> 12; 1 -> 8 -> 2 -> 12; 1 -> 8 -> 3 -> 12; 1 -> 9 -> 1 -> 12; 1 -> 9 -> 3 -> 12; 1 -> 9 -> 10 -> 12; 1 -> 12 -> 1 -> 12; 1 -> 12 -> 2 -> 12; 1 -> 12 -> 4 -> 12; 2 -> 12 -> 4 -> 1; 2 -> 12 -> 5 -> 1; 2 -> 12 -> 1 -> 2; 2 -> 12 -> 5 -> 2; 2 -> 12 -> 11 -> 2; 2 -> 12 -> 1 -> 3; 2 -> 12 -> 11 -> 4; 2 -> 12 -> 4 -> 5; 2 -> 12 -> 1 -> 6; 2 -> 12 -> 11 -> 7; 2 -> 12 -> 1 -> 8; 2 -> 12 -> 5 -> 8; 2 -> 12 -> 1 -> 9; 2 -> 12 -> 4 -> 9; 2 -> 12 -> 4 -> 10; 2 -> 12 -> 5 -> 11; 2 -> 12 -> 11 -> 11; 2 -> 12 -> 1 -> 12; 2 -> 12 -> 2 -> 12; 2 -> 12 -> 4 -> 12; 3 -> 1 -> 3 -> 1; 3 -> 1 -> 9 -> 1; 3 -> 1 -> 12 -> 1; 3 -> 2 -> 12 -> 1; 3 -> 4 -> 5 -> 1; 3 -> 4

-> 9 -> 1; 3 -> 4 -> 10 -> 1; 3 -> 4 -> 12 -> 1; 3 -> 12 -> 4 -> 1; 3 -> 12 -> 5 -> 1; 3 -> 1 -> 3 -> 2; 3 -> 1 -> 6 -> 2; 3 -> 1 -> 8 -> 2; 3 -> 1 -> 12 -> 2; 3 -> 2 -> 12 -> 2; 3 -> 4 -> 1 -> 2; 3 -> 4 -> 5 -> 2; 3 -> 4 -> 10 -> 2; 3 -> 4 -> 12 -> 2; 3 -> 12 -> 1 -> 2; 3 -> 12 -> 5 -> 2; 3 -> 12 -> 11 -> 2; 3 -> 1 -> 8 -> 3; 3 -> 1 -> 9 -> 3; 3 -> 4 -> 1 -> 3; 3 -> 4 -> 9 -> 3; 3 -> 12 -> 1 -> 3; 3 -> 1 -> 3 -> 4; 3 -> 1 -> 12 -> 4; 3 -> 2 -> 12 -> 4; 3 -> 4 -> 12 -> 4; 3 -> 12 -> 11 -> 4; 3 -> 1 -> 6 -> 5; 3 -> 1 -> 8 -> 5; 3 -> 1 -> 12 -> 5; 3 -> 2 -> 12 -> 5; 3 -> 4 -> 12 -> 5; 3 -> 12 -> 4 -> 5; 3 -> 4 -> 1 -> 6; 3 -> 12 -> 1 -> 6; 3 -> 1 -> 8 -> 7; 3 -> 1 -> 9 -> 7; 3 -> 4 -> 9 -> 7; 3 -> 4 -> 10 -> 7; 3 -> 12 -> 11 -> 7; 3 -> 4 -> 1 -> 8; 3 -> 4 -> 5 -> 8; 3 -> 4 -> 10 -> 8; 3 -> 12 -> 1 -> 8; 3 -> 12 -> 5 -> 8; 3 -> 1 -> 6 -> 9; 3 -> 1 -> 8 -> 9; 3 -> 1 -> 9 -> 9; 3 -> 4 -> 1 -> 9; 3 -> 4 -> 9 -> 9; 3 -> 4 -> 10 -> 9; 3 -> 12 -> 1 -> 9; 3 -> 12 -> 4 -> 9; 3 -> 1 -> 9 -> 10; 3 -> 4 -> 9 -> 10; 3 -> 12 -> 4 -> 10; 3 -> 1 -> 6 -> 11; 3 -> 1 -> 12 -> 11; 3 -> 2 -> 12 -> 11; 3 -> 4 -> 5 -> 11; 3 -> 4 -> 12 -> 11; 3 -> 12 -> 5 -> 11; 3 -> 12 -> 11 -> 11; 3 -> 1 -> 2 -> 12; 3 -> 1 -> 3 -> 12; 3 -> 4 -> 1 -> 12; 3 -> 4 -> 10 -> 12; 3 -> 12 -> 1 -> 12; 3 -> 12 -> 2 -> 12; 3 -> 12 -> 4 -> 12; 4 -> 1 -> 3 -> 1; 4 -> 1 -> 9 -> 1; 4 -> 1 -> 12 -> 1; 4 -> 9 -> 3 -> 1; 4 -> 9 -> 9 -> 1; 4 -> 9 -> 10 -> 1; 4 -> 10 -> 9 -> 1; 4 -> 10 -> 12 -> 1; 4 -> 12 -> 4 -> 1; 4 -> 12 -> 5 -> 1; 4 -> 1 -> 3 -> 2; 4 -> 1 -> 6 -> 2; 4 -> 1 -> 8 -> 2; 4 -> 1 -> 12 -> 2; 4 -> 5 -> 1 -> 2; 4 -> 5 -> 8 -> 2; 4 -> 5 -> 11 -> 2; 4 -> 9 -> 1 -> 2; 4 -> 9 -> 3 -> 2; 4 -> 9 -> 10 -> 2; 4 -> 10 -> 1 -> 2; 4 -> 10 -> 8 -> 2; 4 -> 10 -> 12 -> 2; 4 -> 12 -> 1 -> 2; 4 -> 12 -> 5 -> 2; 4 -> 12 -> 11 -> 2; 4 -> 1 -> 8 -> 3; 4 -> 1 -> 9 -> 3; 4 -> 5 -> 1 -> 3; 4 -> 5 -> 8 -> 3; 4 -> 9 -> 1 -> 3; 4 -> 9 -> 7 -> 3; 4 -> 9 -> 9 -> 3; 4 -> 10 -> 1 -> 3; 4 -> 10 -> 7 -> 3; 4 -> 10 -> 8 -> 3; 4 -> 10 -> 9 -> 3; 4 -> 12 -> 1 -> 3; 4 -> 1 -> 3 -> 4; 4 -> 1 -> 12 -> 4; 4 -> 5 -> 11 -> 4; 4 -> 9 -> 3 -> 4; 4 -> 10 -> 12 -> 4; 4 -> 12 -> 11 -> 4; 4 -> 1 -> 6 -> 5; 4 -> 1 -> 8 -> 5; 4 -> 1 -> 12 -> 5; 4 -> 5 -> 8 -> 5; 4 -> 10 -> 8 -> 5; 4 -> 10 -> 12 -> 5; 4 -> 12 -> 4 -> 5; 4 -> 5 -> 1 -> 6; 4 -> 9 -> 1 -> 6; 4 -> 9 -> 7 -> 6; 4 -> 10 -> 1 -> 6; 4 -> 10 -> 7 -> 6; 4 -> 12 -> 1 -> 6; 4 -> 1 -> 8 -> 7; 4 -> 1 -> 9 -> 7; 4 -> 5 -> 8 -> 7; 4 -> 5 -> 11 -> 7; 4 -> 9 -> 9 -> 7; 4 -> 9 -> 10 -> 7; 4 -> 10 -> 8 -> 7; 4 -> 10 -> 9 -> 7; 4 -> 12 -> 11 -> 7; 4 -> 5 -> 1 -> 8; 4 -> 9 -> 1 -> 8; 4 -> 9 -> 7 -> 8; 4 -> 9 -> 10 -> 8; 4 -> 10 -> 1 -> 8; 4 -> 10 -> 7 -> 8; 4 -> 12 -> 1 -> 8; 4 -> 12 -> 5 -> 8; 4 -> 1 -> 6 -> 9; 4 -> 1 -> 8 -> 9; 4 -> 1 -> 9 -> 9; 4 -> 5 -> 1 -> 9; 4 -> 5 -> 8 -> 9; 4 -> 9 -> 1 -> 9; 4 -> 9 -> 9 -> 9; 4 -> 9 -> 10 -> 9; 4 -> 10 -> 1 -> 9; 4 -> 10 -> 8 -> 9; 4 -> 10 -> 9 -> 9; 4 -> 12 -> 1 -> 9; 4 -> 12 -> 4 -> 9; 4 -> 1 -> 9 -> 10; 4 -> 9 -> 9 -> 10; 4 -> 10 -> 9 -> 10; 4 -> 12 -> 4 -> 10; 4 -> 1 -> 6 -> 11; 4 -> 1 -> 12 -> 11; 4 -> 5 -> 11 -> 11; 4 -> 10 -> 12 -> 11; 4 -> 12 -> 5 -> 11; 4 -> 12 -> 11 -> 11; 4 -> 1 -> 2 -> 12; 4 -> 1 -> 3 -> 12; 4 -> 5 -> 1 -> 12; 4 -> 5 -> 2 -> 12; 4 -> 9 -> 1 -> 12; 4 -> 9 -> 3 -> 12; 4 -> 9 -> 10 -> 12; 4 -> 10 -> 1 -> 12; 4 -> 10 -> 2 -> 12; 4 -> 12 -> 1 -> 12; 4 -> 12 -> 2 -> 12; 4 -> 12 -> 4 -> 12; 5 -> 1 -> 3 -> 1; 5 -> 1 -> 9 -> 1; 5 -> 1 -> 12 -> 1; 5 ->

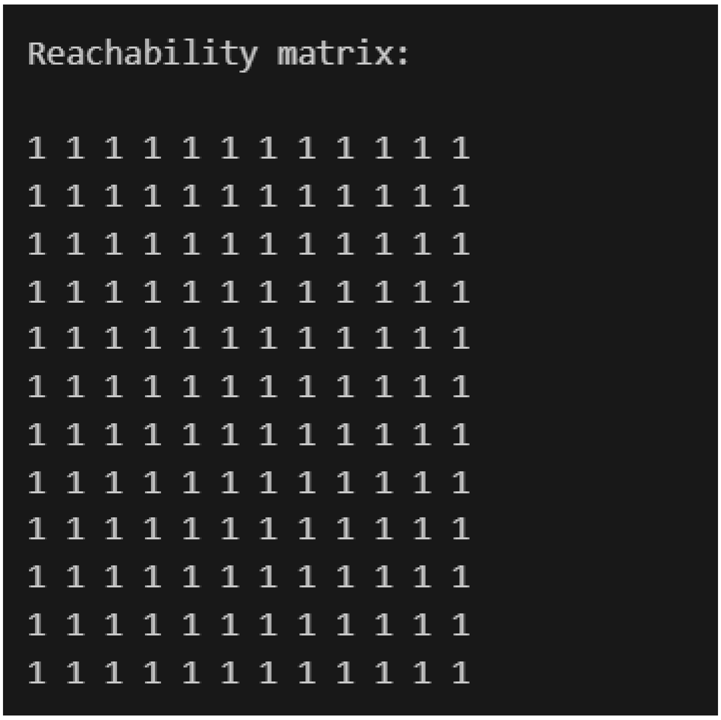
2 -> 12 -> 1; 5 -> 8 -> 3 -> 1; 5 -> 8 -> 5 -> 1; 5 -> 8 -> 9 -> 1; 5 -> 11 -> 4 -> 1; 5 -> 1 -> 3 -> 2; 5 -> 1 -> 6 -> 2; 5 -> 1 -> 8 -> 2; 5 -> 1 -> 12 -> 2; 5 -> 2 -> 12 -> 2; 5 -> 8 -> 3 -> 2; 5 -> 8 -> 5 -> 2; 5 -> 11 -> 11 -> 2; 5 -> 1 -> 8 -> 3; 5 -> 1 -> 9 -> 3; 5 -> 8 -> 7 -> 3; 5 -> 8 -> 9 -> 3; 5 -> 11 -> 7 -> 3; 5 -> 1 -> 3 -> 4; 5 -> 1 -> 12 -> 4; 5 -> 2 -> 12 -> 4; 5 -> 8 -> 3 -> 4; 5 -> 11 -> 11 -> 4; 5 -> 1 -> 6 -> 5; 5 -> 1 -> 8 -> 5; 5 -> 1 -> 12 -> 5; 5 -> 2 -> 12 -> 5; 5 -> 11 -> 4 -> 5; 5 -> 8 -> 7 -> 6; 5 -> 11 -> 7 -> 6; 5 -> 1 -> 8 -> 7; 5 -> 1 -> 9 -> 7; 5 -> 8 -> 9 -> 7; 5 -> 11 -> 11 -> 7; 5 -> 8 -> 5 -> 8; 5 -> 8 -> 7 -> 8; 5 -> 11 -> 7 -> 8; 5 -> 1 -> 6 -> 9; 5 -> 1 -> 8 -> 9; 5 -> 1 -> 9 -> 9; 5 -> 8 -> 9 -> 9; 5 -> 11 -> 4 -> 9; 5 -> 1 -> 9 -> 10; 5 -> 8 -> 9 -> 10; 5 -> 11 -> 4 -> 10; 5 -> 1 -> 6 -> 11; 5 -> 1 -> 12 -> 11; 5 -> 2 -> 12 -> 11; 5 -> 8 -> 5 -> 11; 5 -> 11 -> 11 -> 11; 5 -> 1 -> 2 -> 12; 5 -> 1 -> 3 -> 12; 5 -> 8 -> 2 -> 12; 5 -> 8 -> 3 -> 12; 5 -> 11 -> 2 -> 12; 5 -> 11 -> 4 -> 12; 6 -> 2 -> 12 -> 1; 6 -> 9 -> 3 -> 1; 6 -> 9 -> 9 -> 1; 6 -> 9 -> 10 -> 1; 6 -> 11 -> 4 -> 1; 6 -> 2 -> 12 -> 2; 6 -> 5 -> 1 -> 2; 6 -> 5 -> 8 -> 2; 6 -> 5 -> 11 -> 2; 6 -> 9 -> 1 -> 2; 6 -> 9 -> 3 -> 2; 6 -> 9 -> 10 -> 2; 6 -> 11 -> 11 -> 2; 6 -> 5 -> 1 -> 3; 6 -> 5 -> 8 -> 3; 6 -> 9 -> 1 -> 3; 6 -> 9 -> 7 -> 3; 6 -> 9 -> 9 -> 3; 6 -> 11 -> 7 -> 3; 6 -> 2 -> 12 -> 4; 6 -> 5 -> 11 -> 4; 6 -> 9 -> 3 -> 4; 6 -> 11 -> 11 -> 4; 6 -> 2 -> 12 -> 5; 6 -> 5 -> 8 -> 5; 6 -> 11 -> 4 -> 5; 6 -> 5 -> 1 -> 6; 6 -> 9 -> 1 -> 6; 6 -> 9 -> 7 -> 6; 6 -> 11 -> 7 -> 6; 6 -> 5 -> 8 -> 7; 6 -> 5 -> 11 -> 7; 6 -> 9 -> 9 -> 7; 6 -> 9 -> 10 -> 7; 6 -> 11 -> 11 -> 7; 6 -> 5 -> 1 -> 8; 6 -> 9 -> 1 -> 8; 6 -> 9 -> 7 -> 8; 6 -> 9 -> 10 -> 8; 6 -> 11 -> 7 -> 8; 6 -> 5 -> 1 -> 9; 6 -> 5 -> 8 -> 9; 6 -> 9 -> 1 -> 9; 6 -> 9 -> 9 -> 9; 6 -> 9 -> 10 -> 9; 6 -> 11 -> 4 -> 9; 6 -> 9 -> 9 -> 10; 6 -> 11 -> 4 -> 10; 6 -> 2 -> 12 -> 11; 6 -> 5 -> 11 -> 11; 6 -> 11 -> 11 -> 11; 6 -> 5 -> 1 -> 12; 6 -> 5 -> 2 -> 12; 6 -> 9 -> 1 -> 12; 6 -> 9 -> 3 -> 12; 6 -> 9 -> 10 -> 12; 6 -> 11 -> 2 -> 12; 6 -> 11 -> 4 -> 12; 7 -> 3 -> 4 -> 1; 7 -> 3 -> 12 -> 1; 7 -> 6 -> 5 -> 1; 7 -> 6 -> 9 -> 1; 7 -> 8 -> 3 -> 1; 7 -> 8 -> 5 -> 1; 7 -> 8 -> 9 -> 1; 7 -> 3 -> 1 -> 2; 7 -> 3 -> 12 -> 2; 7 -> 6 -> 5 -> 2; 7 -> 6 -> 11 -> 2; 7 -> 8 -> 3 -> 2; 7 -> 8 -> 5 -> 2; 7 -> 3 -> 1 -> 3; 7 -> 6 -> 9 -> 3; 7 -> 8 -> 7 -> 3; 7 -> 8 -> 9 -> 3; 7 -> 3 -> 12 -> 4; 7 -> 6 -> 11 -> 4; 7 -> 8 -> 3 -> 4; 7 -> 3 -> 4 -> 5; 7 -> 3 -> 12 -> 5; 7 -> 3 -> 1 -> 6; 7 -> 8 -> 7 -> 6; 7 -> 6 -> 9 -> 7; 7 -> 6 -> 11 -> 7; 7 -> 8 -> 9 -> 7; 7 -> 3 -> 1 -> 8; 7 -> 6 -> 5 -> 8; 7 -> 8 -> 5 -> 8; 7 -> 8 -> 7 -> 8; 7 -> 3 -> 1 -> 9; 7 -> 3 -> 4 -> 9; 7 -> 6 -> 9 -> 9; 7 -> 8 -> 9 -> 9; 7 -> 3 -> 4 -> 10; 7 -> 6 -> 9 -> 10; 7 -> 8 -> 9 -> 10; 7 -> 3 -> 12 -> 11; 7 -> 6 -> 5 -> 11; 7 -> 6 -> 11 -> 11; 7 -> 8 -> 5 -> 11; 7 -> 3 -> 1 -> 12; 7 -> 3 -> 2 -> 12; 7 -> 3 -> 4 -> 12; 7 -> 6 -> 2 -> 12; 7 -> 8 -> 2 -> 12; 7 -> 8 -> 3 -> 12; 8 -> 2 -> 12 -> 1; 8 -> 3 -> 4 -> 1; 8 -> 3 -> 12 -> 1; 8 -> 7 -> 3 -> 1; 8 -> 9 -> 3 -> 1; 8 -> 9 -> 9 -> 1; 8 -> 9 -> 10 -> 1; 8 -> 2 -> 12 -> 2; 8 -> 3 -> 1 -> 2; 8 -> 3 -> 12 -> 2; 8 -> 5 -> 1 -> 2; 8 -> 5 -> 8 -> 2; 8 -> 5 -> 11 -> 2; 8 -> 7 -> 3 -> 2; 8 -> 7 -> 6 -> 2; 8 -> 7 -> 8 -> 2; 8 -> 9 -> 1 -> 2; 8 -> 9 -> 3 -> 2; 8 -> 9 -> 10 -> 2; 8 -> 3 -> 1 -> 3; 8 -> 5 -> 1 -> 3; 8 -> 5 -> 8 -> 3; 8 -> 7 -> 8 -> 3; 8 ->

9 -> 1 -> 3; 8 -> 9 -> 7 -> 3; 8 -> 9 -> 9 -> 3; 8 -> 2 -> 12 -> 4; 8 -> 3 -> 12 -> 4; 8 -> 5 -> 11 -> 4; 8 -> 7 -> 3 -> 4; 8 -> 9 -> 3 -> 4; 8 -> 2 -> 12 -> 5; 8 -> 3 -> 4 -> 5; 8 -> 3 -> 12 -> 5; 8 -> 5 -> 8 -> 5; 8 -> 7 -> 6 -> 5; 8 -> 7 -> 8 -> 5; 8 -> 3 -> 1 -> 6; 8 -> 5 -> 1 -> 6; 8 -> 9 -> 1 -> 6; 8 -> 9 -> 7 -> 6; 8 -> 5 -> 8 -> 7; 8 -> 5 -> 11 -> 7; 8 -> 7 -> 8 -> 7; 8 -> 9 -> 9 -> 7; 8 -> 9 -> 10 -> 7; 8 -> 3 -> 1 -> 8; 8 -> 5 -> 1 -> 8; 8 -> 9 -> 1 -> 8; 8 -> 9 -> 7 -> 8; 8 -> 9 -> 10 -> 8; 8 -> 3 -> 1 -> 9; 8 -> 3 -> 4 -> 9; 8 -> 5 -> 1 -> 9; 8 -> 5 -> 8 -> 9; 8 -> 7 -> 6 -> 9; 8 -> 7 -> 8 -> 9; 8 -> 9 -> 1 -> 9; 8 -> 9 -> 9 -> 9; 8 -> 9 -> 10 -> 9; 8 -> 3 -> 4 -> 10; 8 -> 9 -> 9 -> 10; 8 -> 2 -> 12 -> 11; 8 -> 3 -> 12 -> 11; 8 -> 5 -> 11 -> 11; 8 -> 7 -> 6 -> 11; 8 -> 3 -> 1 -> 12; 8 -> 3 -> 2 -> 12; 8 -> 3 -> 4 -> 12; 8 -> 5 -> 1 -> 12; 8 -> 5 -> 2 -> 12; 8 -> 7 -> 3 -> 12; 8 -> 9 -> 1 -> 12; 8 -> 9 -> 3 -> 12; 8 -> 9 -> 10 -> 12; 9 -> 1 -> 3 -> 1; 9 -> 1 -> 9 -> 1; 9 -> 1 -> 12 -> 1; 9 -> 3 -> 4 -> 1; 9 -> 3 -> 12 -> 1; 9 -> 7 -> 3 -> 1; 9 -> 9 -> 3 -> 1; 9 -> 9 -> 9 -> 1; 9 -> 9 -> 10 -> 1; 9 -> 10 -> 9 -> 1; 9 -> 10 -> 12 -> 1; 9 -> 1 -> 3 -> 2; 9 -> 1 -> 6 -> 2; 9 -> 1 -> 8 -> 2; 9 -> 1 -> 12 -> 2; 9 -> 3 -> 1 -> 2; 9 -> 3 -> 12 -> 2; 9 -> 7 -> 3 -> 2; 9 -> 7 -> 6 -> 2; 9 -> 7 -> 8 -> 2; 9 -> 9 -> 1 -> 2; 9 -> 9 -> 3 -> 2; 9 -> 9 -> 10 -> 2; 9 -> 10 -> 1 -> 2; 9 -> 10 -> 8 -> 2; 9 -> 10 -> 12 -> 2; 9 -> 1 -> 8 -> 3; 9 -> 1 -> 9 -> 3; 9 -> 3 -> 1 -> 3; 9 -> 7 -> 8 -> 3; 9 -> 9 -> 1 -> 3; 9 -> 9 -> 7 -> 3; 9 -> 9 -> 9 -> 3; 9 -> 10 -> 1 -> 3; 9 -> 10 -> 7 -> 3; 9 -> 10 -> 8 -> 3; 9 -> 10 -> 9 -> 3; 9 -> 1 -> 3 -> 4; 9 -> 1 -> 12 -> 4; 9 -> 3 -> 12 -> 4; 9 -> 7 -> 3 -> 4; 9 -> 9 -> 3 -> 4; 9 -> 10 -> 12 -> 4; 9 -> 1 -> 6 -> 5; 9 -> 1 -> 8 -> 5; 9 -> 1 -> 12 -> 5; 9 -> 3 -> 4 -> 5; 9 -> 3 -> 12 -> 5; 9 -> 7 -> 6 -> 5; 9 -> 7 -> 8 -> 5; 9 -> 10 -> 8 -> 5; 9 -> 10 -> 12 -> 5; 9 -> 3 -> 1 -> 6; 9 -> 9 -> 1 -> 6; 9 -> 9 -> 7 -> 6; 9 -> 10 -> 1 -> 6; 9 -> 10 -> 7 -> 6; 9 -> 1 -> 8 -> 7; 9 -> 1 -> 9 -> 7; 9 -> 7 -> 8 -> 7; 9 -> 9 -> 9 -> 7; 9 -> 9 -> 10 -> 7; 9 -> 10 -> 8 -> 7; 9 -> 10 -> 9 -> 7; 9 -> 3 -> 1 -> 8; 9 -> 9 -> 1 -> 8; 9 -> 9 -> 7 -> 8; 9 -> 9 -> 10 -> 8; 9 -> 10 -> 1 -> 8; 9 -> 10 -> 7 -> 8; 9 -> 1 -> 6 -> 9; 9 -> 1 -> 8 -> 9; 9 -> 1 -> 9 -> 9; 9 -> 3 -> 1 -> 9; 9 -> 3 -> 4 -> 9; 9 -> 7 -> 6 -> 9; 9 -> 7 -> 8 -> 9; 9 -> 9 -> 1 -> 9; 9 -> 9 -> 9 -> 9; 9 -> 9 -> 10 -> 9; 9 -> 10 -> 1 -> 9; 9 -> 10 -> 8 -> 9; 9 -> 10 -> 9 -> 9; 9 -> 1 -> 9 -> 10; 9 -> 3 -> 4 -> 10; 9 -> 9 -> 9 -> 10; 9 -> 10 -> 9 -> 10; 9 -> 1 -> 6 -> 11; 9 -> 1 -> 12 -> 11; 9 -> 3 -> 12 -> 11; 9 -> 7 -> 6 -> 11; 9 -> 10 -> 12 -> 11; 9 -> 1 -> 2 -> 12; 9 -> 1 -> 3 -> 12; 9 -> 3 -> 1 -> 12; 9 -> 3 -> 2 -> 12; 9 -> 3 -> 4 -> 12; 9 -> 7 -> 3 -> 12; 9 -> 9 -> 1 -> 12; 9 -> 9 -> 3 -> 12; 9 -> 9 -> 10 -> 12; 9 -> 10 -> 1 -> 12; 9 -> 10 -> 2 -> 12; 10 -> 1 -> 3 -> 1; 10 -> 1 -> 9 -> 1; 10 -> 1 -> 12 -> 1; 10 -> 2 -> 12 -> 1; 10 -> 7 -> 3 -> 1; 10 -> 8 -> 3 -> 1; 10 -> 8 -> 5 -> 1; 10 -> 8 -> 9 -> 1; 10 -> 9 -> 3 -> 1; 10 -> 9 -> 9 -> 1; 10 -> 9 -> 10 -> 1; 10 -> 12 -> 4 -> 1; 10 -> 12 -> 5 -> 1; 10 -> 1 -> 3 -> 2; 10 -> 1 -> 6 -> 2; 10 -> 1 -> 8 -> 2; 10 -> 1 -> 12 -> 2; 10 -> 2 -> 12 -> 2; 10 -> 7 -> 3 -> 2; 10 -> 7 -> 6 -> 2; 10 -> 7 -> 8 -> 2; 10 -> 8 -> 3 -> 2; 10 -> 8 -> 5 -> 2; 10 -> 9 -> 1 -> 2; 10 -> 9 -> 3 -> 2; 10 -> 9 -> 10 -> 2; 10 -> 12 -> 1 -> 2; 10 -> 12 -> 5 -> 2; 10 -> 12 -> 11

-> 2; 10 -> 1 -> 8 -> 3; 10 -> 1 -> 9 -> 3; 10 -> 7 -> 8 -> 3; 10 -> 8 -> 7 -> 3; 10 -> 8 -> 9 -> 3; 10 -> 9 -> 1 -> 3; 10 -> 9 -> 7 -> 3; 10 -> 9 -> 9 -> 3; 10 -> 12 -> 1 -> 3; 10 -> 1 -> 3 -> 4; 10 -> 1 -> 12 -> 4; 10 -> 2 -> 12 -> 4; 10 -> 7 -> 3 -> 4; 10 -> 8 -> 3 -> 4; 10 -> 9 -> 3 -> 4; 10 -> 12 -> 11 -> 4; 10 -> 1 -> 6 -> 5; 10 -> 1 -> 8 -> 5; 10 -> 1 -> 12 -> 5; 10 -> 2 -> 12 -> 5; 10 -> 7 -> 6 -> 5; 10 -> 7 -> 8 -> 5; 10 -> 12 -> 4 -> 5; 10 -> 8 -> 7 -> 6; 10 -> 9 -> 1 -> 6; 10 -> 9 -> 7 -> 6; 10 -> 12 -> 1 -> 6; 10 -> 1 -> 8 -> 7; 10 -> 1 -> 9 -> 7; 10 -> 7 -> 8 -> 7; 10 -> 8 -> 9 -> 7; 10 -> 9 -> 9 -> 7; 10 -> 9 -> 10 -> 7; 10 -> 12 -> 11 -> 7; 10 -> 8 -> 5 -> 8; 10 -> 8 -> 7 -> 8; 10 -> 9 -> 1 -> 8; 10 -> 9 -> 7 -> 8; 10 -> 9 -> 10 -> 8; 10 -> 12 -> 1 -> 8; 10 -> 12 -> 5 -> 8; 10 -> 1 -> 6 -> 9; 10 -> 1 -> 8 -> 9; 10 -> 1 -> 9 -> 9; 10 -> 7 -> 6 -> 9; 10 -> 7 -> 8 -> 9; 10 -> 8 -> 9 -> 9; 10 -> 9 -> 1 -> 9; 10 -> 9 -> 9 -> 9; 10 -> 9 -> 10 -> 9; 10 -> 12 -> 1 -> 9; 10 -> 12 -> 4 -> 9; 10 -> 1 -> 9 -> 10; 10 -> 8 -> 9 -> 10; 10 -> 9 -> 9 -> 10; 10 -> 12 -> 4 -> 10; 10 -> 1 -> 6 -> 11; 10 -> 1 -> 12 -> 11; 10 -> 2 -> 12 -> 11; 10 -> 7 -> 6 -> 11; 10 -> 8 -> 5 -> 11; 10 -> 12 -> 5 -> 11; 10 -> 12 -> 11 -> 11; 10 -> 1 -> 2 -> 12; 10 -> 1 -> 3 -> 12; 10 -> 7 -> 3 -> 12; 10 -> 8 -> 2 -> 12; 10 -> 8 -> 3 -> 12; 10 -> 9 -> 1 -> 12; 10 -> 9 -> 3 -> 12; 10 -> 9 -> 10 -> 12; 10 -> 12 -> 1 -> 12; 10 -> 12 -> 2 -> 12; 10 -> 12 -> 4 -> 12; 11 -> 2 -> 12 -> 1; 11 -> 4 -> 5 -> 1; 11 -> 4 -> 9 -> 1; 11 -> 4 -> 10 -> 1; 11 -> 4 -> 12 -> 1; 11 -> 7 -> 3 -> 1; 11 -> 11 -> 4 -> 1; 11 -> 2 -> 12 -> 2; 11 -> 4 -> 1 -> 2; 11 -> 4 -> 5 -> 2; 11 -> 4 -> 10 -> 2; 11 -> 4 -> 12 -> 2; 11 -> 7 -> 3 -> 2; 11 -> 7 -> 6 -> 2; 11 -> 7 -> 8 -> 2; 11 -> 11 -> 11 -> 2; 11 -> 4 -> 1 -> 3; 11 -> 4 -> 9 -> 3; 11 -> 7 -> 8 -> 3; 11 -> 11 -> 7 -> 3; 11 -> 2 -> 12 -> 4; 11 -> 4 -> 12 -> 4; 11 -> 7 -> 3 -> 4; 11 -> 11 -> 11 -> 4; 11 -> 2 -> 12 -> 5; 11 -> 4 -> 12 -> 5; 11 -> 7 -> 6 -> 5; 11 -> 7 -> 8 -> 5; 11 -> 11 -> 4 -> 5; 11 -> 4 -> 1 -> 6; 11 -> 11 -> 7 -> 6; 11 -> 4 -> 9 -> 7; 11 -> 4 -> 10 -> 7; 11 -> 7 -> 8 -> 7; 11 -> 11 -> 11 -> 7; 11 -> 4 -> 1 -> 8; 11 -> 4 -> 5 -> 8; 11 -> 4 -> 10 -> 8; 11 -> 11 -> 7 -> 8; 11 -> 4 -> 1 -> 9; 11 -> 4 -> 9 -> 9; 11 -> 4 -> 10 -> 9; 11 -> 7 -> 6 -> 9; 11 -> 7 -> 8 -> 9; 11 -> 11 -> 4 -> 9; 11 -> 4 -> 9 -> 10; 11 -> 11 -> 4 -> 10; 11 -> 2 -> 12 -> 11; 11 -> 4 -> 5 -> 11; 11 -> 4 -> 12 -> 11; 11 -> 7 -> 6 -> 11; 11 -> 11 -> 11 -> 11; 11 -> 4 -> 1 -> 12; 11 -> 4 -> 10 -> 12; 11 -> 7 -> 3 -> 12; 11 -> 11 -> 2 -> 12; 11 -> 11 -> 4 -> 12; 12 -> 1 -> 3 -> 1; 12 -> 1 -> 9 -> 1; 12 -> 1 -> 12 -> 1; 12 -> 2 -> 12 -> 1; 12 -> 4 -> 5 -> 1; 12 -> 4 -> 9 -> 1; 12 -> 4 -> 10 -> 1; 12 -> 4 -> 12 -> 1; 12 -> 11 -> 4 -> 1; 12 -> 1 -> 3 -> 2; 12 -> 1 -> 6 -> 2; 12 -> 1 -> 8 -> 2; 12 -> 1 -> 12 -> 2; 12 -> 2 -> 12 -> 2; 12 -> 4 -> 1 -> 2; 12 -> 4 -> 5 -> 2; 12 -> 4 -> 10 -> 2; 12 -> 4 -> 12 -> 2; 12 -> 5 -> 1 -> 2; 12 -> 5 -> 8 -> 2; 12 -> 5 -> 11 -> 2; 12 -> 11 -> 11 -> 2; 12 -> 1 -> 8 -> 3; 12 -> 1 -> 9 -> 3; 12 -> 4 -> 1 -> 3; 12 -> 4 -> 9 -> 3; 12 -> 5 -> 1 -> 3; 12 -> 5 -> 8 -> 3; 12 -> 11 -> 7 -> 3; 12 -> 1 -> 3 -> 4; 12 -> 1 -> 12 -> 4; 12 -> 2 -> 12 -> 4; 12 -> 4 -> 12 -> 4; 12 -> 5 -> 11 -> 4; 12 -> 11 -> 11 -> 4; 12 -> 1 -> 6 -> 5; 12 -> 1 -> 8 -> 5; 12 -> 1 -> 12 -> 5; 12 -> 2 -> 12 -> 5; 12 -> 4 -> 12 -> 5; 12 -> 5 -> 8 -> 5; 12 -> 11 ->

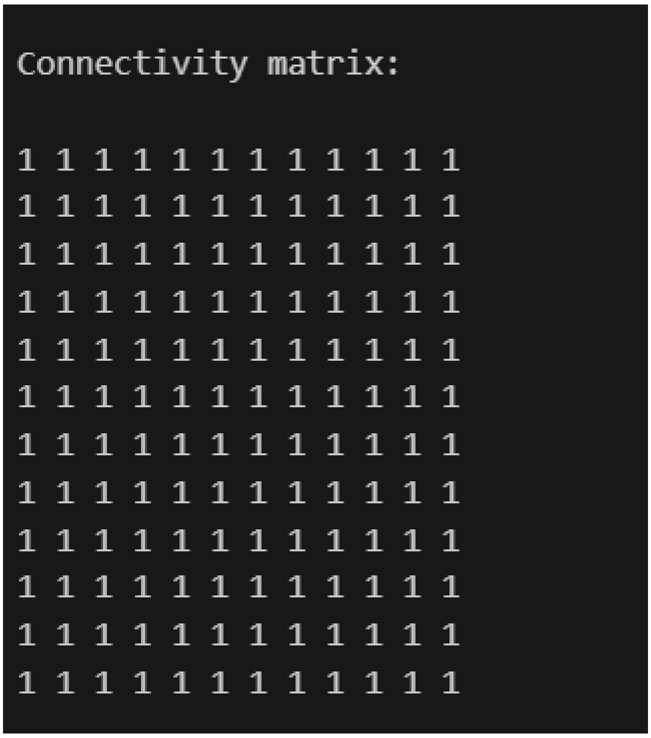
4 -> 5; 12 -> 4 -> 1 -> 6; 12 -> 5 -> 1 -> 6; 12 -> 11 -> 7 -> 6; 12 -> 1 -> 8 -> 7; 12 -> 1 -> 9 -> 7; 12 -> 4 -> 9 -> 7; 12 -> 4 -> 10 -> 7; 12 -> 5 -> 8 -> 7; 12 -> 5 -> 11 -> 7; 12 -> 11 -> 11 -> 7; 12 -> 4 -> 1 -> 8; 12 -> 4 -> 5 -> 8; 12 -> 4 -> 10 -> 8; 12 -> 5 -> 1 -> 8; 12 -> 11 -> 7 -> 8; 12 -> 1 -> 6 -> 9; 12 -> 1 -> 8 -> 9; 12 -> 1 -> 9 -> 9; 12 -> 4 -> 1 -> 9; 12 -> 4 -> 9 -> 9; 12 -> 4 -> 10 -> 9; 12 -> 5 -> 1 -> 9; 12 -> 5 -> 8 -> 9; 12 -> 11 -> 4 -> 9; 12 -> 1 -> 9 -> 10; 12 -> 4 -> 9 -> 10; 12 -> 11 -> 4 -> 10; 12 -> 1 -> 6 -> 11; 12 -> 1 -> 12 -> 11; 12 -> 2 -> 12 -> 11; 12 -> 4 -> 5 -> 11; 12 -> 4 -> 12 -> 11; 12 -> 5 -> 11 -> 11; 12 -> 11 -> 11 -> 11; 12 -> 1 -> 2 -> 12; 12 -> 1 -> 3 -> 12; 12 -> 4 -> 1 -> 12; 12 -> 4 -> 10 -> 12; 12 -> 5 -> 1 -> 12; 12 -> 5 -> 2 -> 12; 12 -> 11 -> 2 -> 12; 12 -> 11 -> 4 -> 12;

Матриця досяжності:



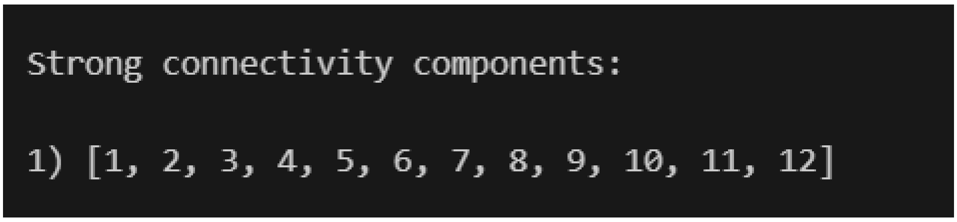
```
Reachability matrix:

1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
```

Матриця сильної зв’язності:



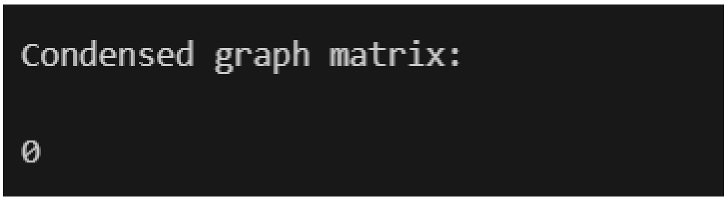
```
Connectivity matrix:

1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
```

Компоненти сильної зв’язності:



```
Strong connectivity components:

1) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
```

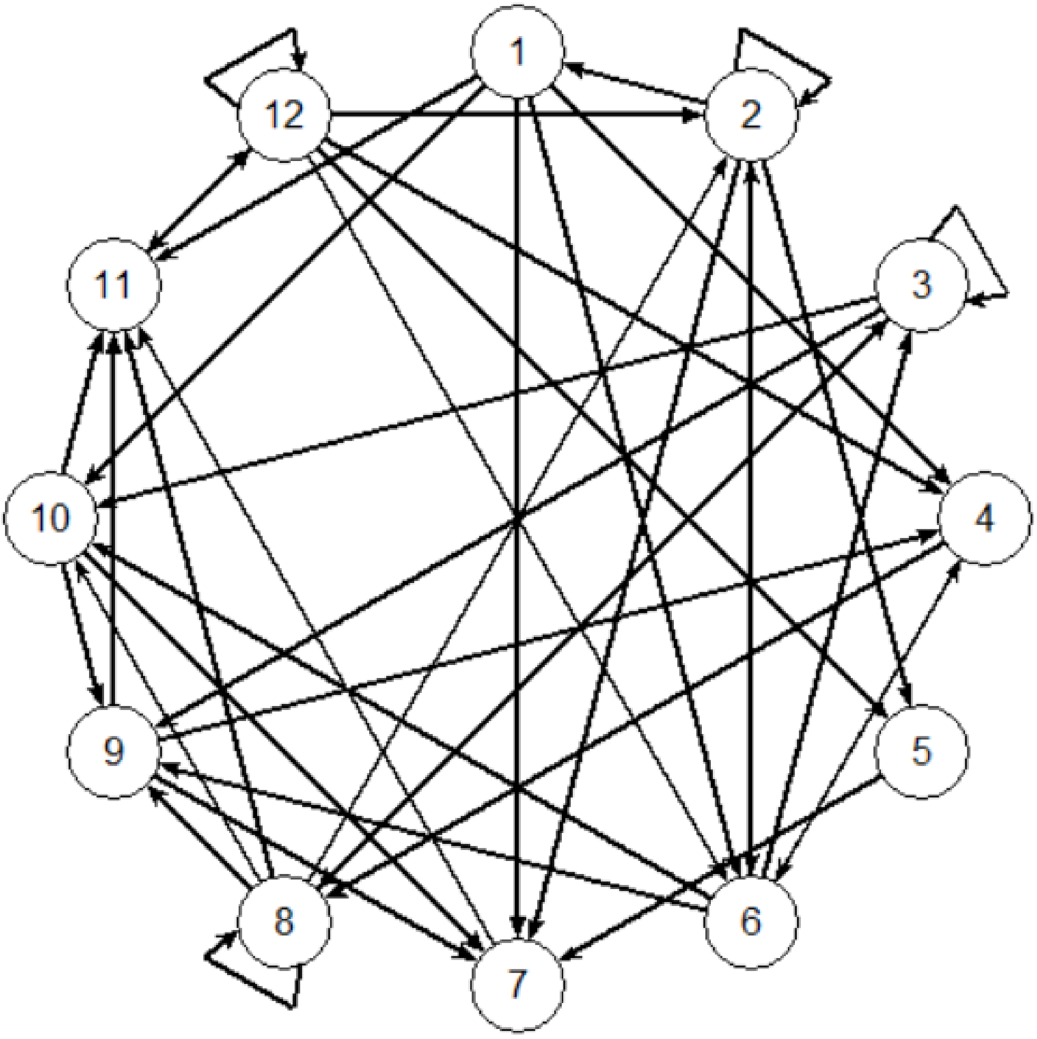
Конденсована матриця суміжності:



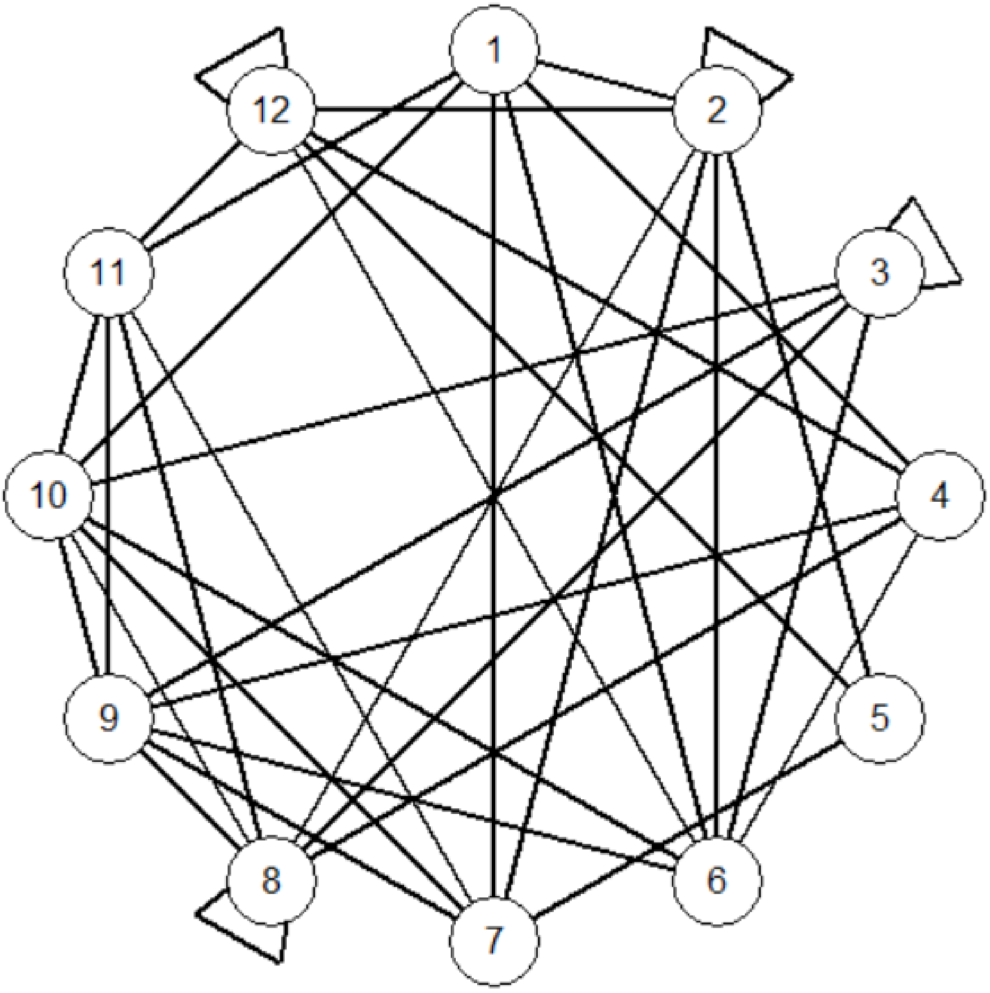
```
Condensed graph matrix:

0
```

Зображення графів

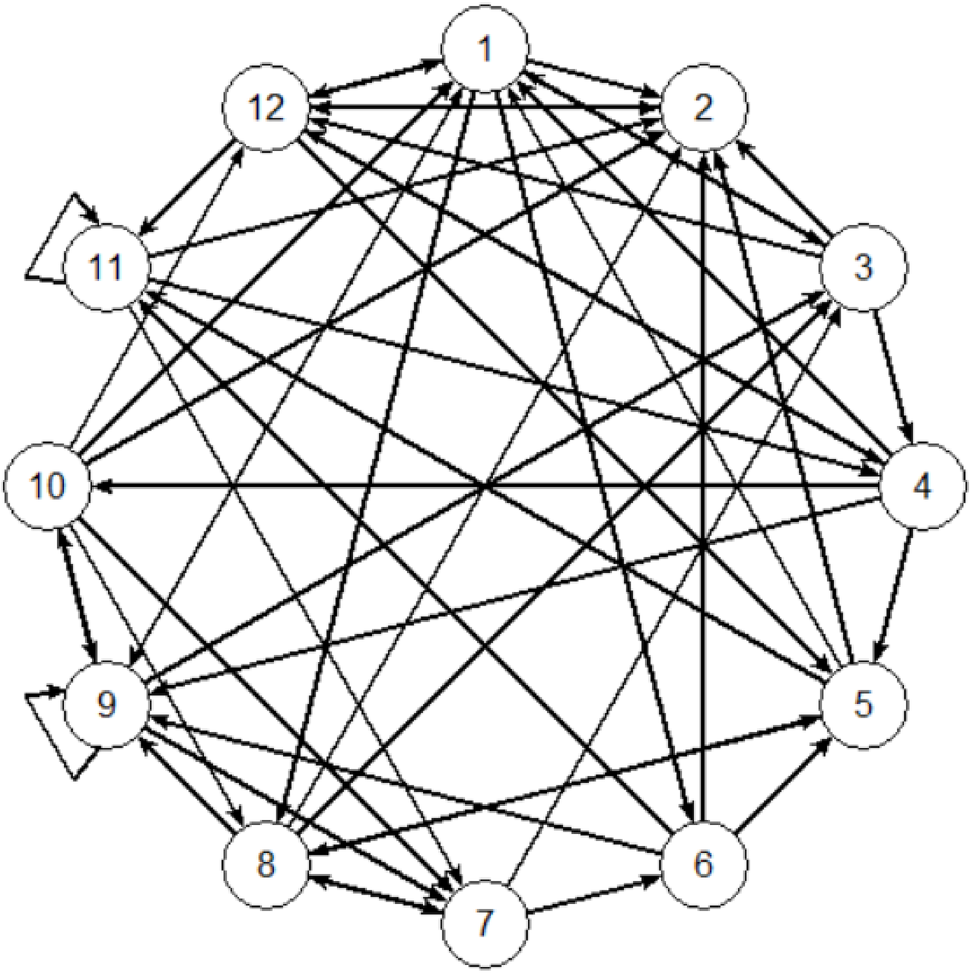
1) Напрямлений граф



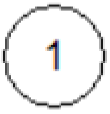
2) Ненапрямлений граф



3) Модифікований граф



4) Граф конденсації



Висновок

Модифікував програму лабораторної №3, щоб вона рахувала півстепені вершин, шляхи довжиною 2 та 3, матрицю досяжності, матрицю сильної зв’язності, перелік компонент сильної зв’язності та граф конденсації; перевіряла регулярність та перелічувала висячі та ізольовані вершини графа.