**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 iз заданими параметрами так само, як у лабораторнiй роботi №3.

Вiдмiннiсть: коефiцiєнт k = 1.0 - n3 \* 0.01 - n4 \* 0.01 - 0.3.

Отже, матриця сумiжностi Adir напрямленого графа за варiантом формується таким чином:

1) встановлюється параметр (seed) генератора випадкових чисел, рiвне номеру варiанту n1n2n3n4;

2) матриця розмiром n \* n заповнюється згенерованими випадковими

числами в дiапазонi [0, 2.0);

3) обчислюється коефiцiєнт k = 1.0 - n3 \* 0.01 - n4 \* 0.01 - 0.3, кожен

елемент матрицi множиться на коефiцiєнт k;

4) елементи матрицi округлюються: 0 — якщо елемент менший за 1.0,

1 — якщо елемент бiльший або дорiвнює 1.0.

2. Обчислити:

1) степенi вершин напрямленого i ненапрямленого графiв;

2) напiвстепенi виходу та заходу напрямленого графа;

3) чи є граф однорiдним (регулярним), i якщо так, вказати степiнь

однорiдностi графа;

4) перелiк висячих та iзольованих вершин.

Результати вивести у графiчне вiкно, консоль або файл.

3. Змiнити матрицю Adir, коефiцiєнт k = 1.0-n3 \* 0.005-n4 \* 0.005-

0.27.

4. Для нового орграфа обчислити:

1) пiвстепенi вершин;

2) всi шляхи довжини 2 i 3;

3) матрицю досяжностi;

4) матрицю сильної зв’язностi;

5) перелiк компонент сильної зв’язностi;

6) граф конденсацiї.

Результати вивести у графiчне вiкно, в консоль або файл.

Шляхи довжиною 2 i 3 слiд шукати за матрицями A^2 i A^3, вiдповiдно. Як результат вивести перелiк шляхiв, включно з усiма промiжними

вершинами, через якi проходить шлях.

Матрицю досяжностi та компоненти сильної зв’язностi слiд шукати за

допомогою операцiї транзитивного замикання. У перелiку компонент

слiд вказати, якi вершини належать до кожної компоненти.

Граф конденсацiї вивести у графiчне вiкно.

**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:

Зображення, що містить текст, знімок екрана, чорно-білий, дизайн

Вміст, створений ШІ, може бути неправильним.

Матриця суміжності ненапрямленого графа:

Зображення, що містить текст, знімок екрана, Шрифт, чорно-білий

Вміст, створений ШІ, може бути неправильним.

**Характеристика вершин**

Для напрямленого графа:

Зображення, що містить текст, Шрифт, знімок екрана

Вміст, створений ШІ, може бути неправильним.

Для ненапрямленого графа:

Зображення, що містить текст, Шрифт, знімок екрана, чорний

Вміст, створений ШІ, може бути неправильним.

**Модифікований граф**

Матриця суміжності:

Зображення, що містить текст, знімок екрана, чорно-білий, Шрифт

Вміст, створений ШІ, може бути неправильним.

**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:**

Зображення, що містить текст, знімок екрана, дизайн, чорно-білий

Вміст, створений ШІ, може бути неправильним.

**Матриця сильної зв’язності:**

Зображення, що містить знімок екрана, текст, дизайн, чорно-білий

Вміст, створений ШІ, може бути неправильним.

**Компоненти сильної зв’язності:**

**Зображення, що містить текст, Шрифт, знімок екрана, чорний

Вміст, створений ШІ, може бути неправильним.**

**Конденсована матриця суміжності:**

**Зображення, що містить текст, Шрифт, знімок екрана, чорний

Вміст, створений ШІ, може бути неправильним.**

**Зображення графів**

1. Напрямлений граф

Зображення, що містить ескіз, малюнок, коло, схема

Вміст, створений ШІ, може бути неправильним.

1. Ненапрямлений граф

Зображення, що містить ескіз, коло, схема, оригамі

Вміст, створений ШІ, може бути неправильним.

1. Модифікований граф

Зображення, що містить ескіз, коло, малюнок, дизайн

Вміст, створений ШІ, може бути неправильним.

1. Граф конденсації

Зображення, що містить ескіз, коло, чорно-білий

Вміст, створений ШІ, може бути неправильним.

**Висновок**

Модифікував програму лабораторної №3, щоб вона рахувала півстепені вершин, шляхи довжиною 2 та 3, матрицю досяжностi, матрицю сильної зв’язностi, перелiк компонент сильної зв’язностi та граф конденсацiї; перевіряла регулярність та перелічувала висячі та ізольовані вершини графа.