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»

Виконала:

студентка групи ІМ-21

Рабійчук Дар’я Олександрівна

номер у списку групи: 18

Перевірила:

Молчанова А.А.

Київ 2023

Постановка задачі:

1. Представити напрямлений граф з заданими параметрами так само, як у лабораторній роботі №3.

Відміна: матриця A напрямленого графа за варіантом формується за функціями:

srand(п1 п2 п3 п4);

T = randm(n,n);

A = mulmr((1.0 – n3*0.01 – n4*0.01 − 0.3)*T);

Перетворити граф у ненапрямлений.

2. Визначити степені вершин напрямленого і ненапрямленого графів. Програма на екран виводить степені усіх вершин ненапрямленого графу і напівстепені виходу та заходу напрямленого графу. Визначити, чи граф є однорідним та якщо так, то вказати степінь однорідності графу.

3. Визначити всі висячі та ізольовані вершини. Програма на екран виводить перелік усіх висячих та ізольованих вершин графу.

4. Змінити матрицю графу за функцією A = mulmr((1.0 – n3*0.005 – n4*0.005 − 0.27)*T);

Створити програму для обчислення наступних результатів:

1) матриця суміжності;

2) півстепені вузлів;

3) всі шляхи довжини 2 і 3;

4) матриця досяжності;

5) компоненти сильної зв’язності;

6) матриця зв’язності;

7) граф конденсації.

Шляхи довжиною 2 і 3 слід шукати за матрицями A2 і A3 , відповідно. Матриця досяжності та компоненти сильної зв’язності слід шукати за допомогою операції транзитивного замикання.

Варіант 18 :

N1 = 2

N2 = 1

N3 = 1

N4 = 8

Число вершин n дорівнює 11

Текст програми:

```
#include <windows.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

double ** randm(int rows, int cols) {
    double ** matrix = (double ** ) malloc(rows * sizeof(double * ));

    for (int i = 0; i < rows; i++)
        matrix[i] = (double * ) malloc(cols * sizeof(double));

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            double temp = rand() % 21;
            matrix[i][j] = temp / 10;
        }
    }
    return matrix;
}

double ** mulmr(double coef, double ** matrix, int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            matrix[i][j] = matrix[i][j] * coef;

            if (matrix[i][j] < 1.0) {
                matrix[i][j] = 0;
            } else matrix[i][j] = 1;
        }
    }
    return matrix;
}

double ** mirrorMatrix(double ** matrix, int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (matrix[i][j] == 1) {
                matrix[j][i] = 1;
            }
        }
    }

    return matrix;
}

double ** cloneMatrix(double ** matrix, double ** matrixToCopy, int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            matrix[i][j] = matrixToCopy[i][j];
        }
    }
    return matrix;
}

void graphPower(double ** matrix, double ** mirror_matrix, int rows, int cols) {
    int counter = 0, counterp = 0, counterm = 0; {
        printf("\n\n");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                if (matrix[i][j] == 1) counterp++;
            }
            printf("Graph top: %d , deg+ is: %d  ", i + 1, counterp);
            counterp = 0;
        }
        printf("\n\n");
        for (int i = 0; i < cols; i++) {
            for (int j = 0; j < rows; j++) {
                if (matrix[j][i] == 1) counterm++;
            }
            printf("Graph top: %d , deg- is: %d  ", i + 1, counterm);
            counterm = 0;
        }
        printf("\n\n");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                if (mirror_matrix[i][j] == 1) counter++;
            }
            if (counter == 1) {
                printf("Graph top: %d , a hanging top, deg is: %d  ", i + 1, counter);
            } else if (counter == 0) {
                printf("Graph top: %d , an isolated top, deg is: %d  ", i + 1, counter);
```

```
            } else {
                printf("Graph top: %d , deg is: %d  ", i + 1, counter);
            }
            counter = 0;
        }
    }
}

void wayMatrix(double ** matrix, int rows, int cols) {
    printf_s("\n\n");
    printf_s("2-length ways\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (matrix[i][j] == 1) {
                for (int k = 0; k < cols; k++) {
                    if (matrix[j][k] == 1) printf("| %d->%d->%d  |", i + 1, j + 1, k + 1);
                }
            }
        }
        printf_s("\n");
    }
    printf_s("\n\n");
    printf_s("3-length ways\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (matrix[i][j] == 1) {
                for (int count1 = 0; count1 < cols; count1++) {
                    if (matrix[j][count1] == 1) {
                        for (int count2 = 0; count2 < cols; count2++) {
                            if (matrix[count1][count2] == 1) {
                                printf("| %d->%d->%d->%d |", i + 1, j + 1, count1 + 1, count2 + 1);
                            }
                        }
                    }
                }
            }
        }
        printf_s("\n");
    }
}

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

char ProgName[] = "LAB4 RABIICHUK IM-21";

int WINAPI WinMain(HINSTANCE hInstance,
                   HINSTANCE hPrevInstance,
                   LPSTR lpszCmdLine,
                   int nCmdShow) {
    HWND hWnd;
    MSG lpMsg;

    WNDCLASS w;

    w.lpszClassName = ProgName;
    w.hInstance = hInstance;
    w.lpfnWndProc = WndProc;
    w.hCursor = LoadCursor(NULL, IDC_ARROW);
    w.hIcon = 0;
    w.lpszMenuName = 0;
    w.hbrBackground = WHITE_BRUSH;
    w.style = CS_HREDRAW | CS_VREDRAW;
    w.cbClsExtra = 0;
    w.cbWndExtra = 0;

    if (!RegisterClass( & w)) return 0;

    hWnd = CreateWindow(ProgName,
                        "LAB4 RABIICHUK IM-21",
                        WS_OVERLAPPEDWINDOW,
                        0,
                        0,
                        600,
                        600,
                        (HWND) NULL,
                        (HMENU) NULL,
                        (HINSTANCE) hInstance,
                        (HINSTANCE) NULL);

    ShowWindow(hWnd, nCmdShow);

    while (GetMessage( & lpMsg, hWnd, 0, 0)) {

        TranslateMessage( & lpMsg);
```

```
        DispatchMessage( & lpMsg);
    }
    return lpMsg.wParam;
}

LRESULT CALLBACK WndProc(HWND hWnd,
                         UINT messg,
                         WPARAM wParam,
                         LPARAM lParam) {
    HDC hdc;
    PAINTSTRUCT ps;

    void arrow(float fi, int px, int py) {
        fi = 3.1416 * (180.0 - fi) / 180.0;
        int lx, ly, rx, ry;
        lx = px + 15 * cos(fi + 0.3);
        rx = px + 15 * cos(fi - 0.3);
        ly = py + 15 * sin(fi + 0.3);
        ry = py + 15 * sin(fi - 0.3);
        MoveToEx(hdc, lx, ly, NULL);
        LineTo(hdc, px, py);
        LineTo(hdc, rx, ry);
    }

    switch (messg) {
        case WM_PAINT:
            hdc = BeginPaint(hWnd, & ps);
            int n = 11;
            char * nn[11] = {
                    "1",
                    "2",
                    "3",
                    "4",
                    "5",
                    "6",
                    "7",
                    "8",
                    "9",
                    "10",
                    "11"
            };
            int nx[11] = {};
            int ny[11] = {};
            int num = 120;
            for (int i = 0; i < n; i++) {
                if (i == 0) {
                    nx[i] = num;
                    ny[i] = num;
                } else if (i < 4) {
                    nx[i] = nx[i - 1] + num;
                    ny[i] = ny[i - 1];
                } else if (i < 6) {
                    nx[i] = nx[i - 1];
                    ny[i] = ny[i - 1] + num;
                } else if (i < 9) {
                    nx[i] = nx[i - 1] - num;
                    ny[i] = ny[i - 1];
                } else if (i < 10) {
                    nx[i] = nx[i - 1];
                    ny[i] = ny[i - 1] - num;
                } else {
                    nx[i] = nx[i - 1] + 3 * num / 2;
                    ny[i] = ny[i - 1] + 2;
                }
            }
            int dx = 20, dy = 20, dtx = 5;
            int i;
            HPEN BPen = CreatePen(PS_SOLID, 2, RGB(76, 0, 153));
            HPEN KPen = CreatePen(PS_SOLID, 1, RGB(204, 153, 255));

            srand(2118);
            double ** T = randm(11, 11);
            double coefficient2 = 1.0 - 1 * 0.005 - 8 * 0.005 - 0.27;
            double ** A = mulmr(coefficient2, T, n, n);
            double ** B_temp = randm(n, n);
            double ** B_temp2 = cloneMatrix(B_temp, A, n, n);
            double ** B = mirrorMatrix(B_temp2, n, n);

            void printGrapgMatrix(double ** graphMatrix) {

                for (int i = 0; i < 11; i++) {
                    for (int j = 0; j < 11; j++) {
                        printf("%g ", graphMatrix[i][j]);
                    }
```

```
                printf("\n");
            }
        }

        SelectObject(hdc, KPen);

        void drawGraphDir(double ** graphMatrix) {
            for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                    if (graphMatrix[i][j] == 1) {
                        MoveToEx(hdc, nx[i], ny[i], NULL);

                        if (i == j) {
                            if (i < n * 0.25) {
                                Arc(hdc, nx[j], ny[j], nx[j] - 50, ny[j] - 50, nx[j], ny[j], nx[j],
ny[j]);
                                arrow((-90 * 3.1416) / 180, nx[j], ny[j] - dy);
                            } else if (i < n * 0.5) {
                                Arc(hdc, nx[j], ny[j], nx[j] + 50, ny[j] - 50, nx[j], ny[j], nx[j],
ny[j]);
                                arrow((0 * 3.1416) / 180, nx[j] + dx, ny[j]);
                            } else if (i < n * 0.75) {
                                Arc(hdc, nx[j], ny[j], nx[j] + 50, ny[j] + 50, nx[j], ny[j], nx[j],
ny[j]);
                                arrow((90 * 3.1416) / 180, nx[j], ny[j] + dy);
                            } else {
                                Arc(hdc, nx[j], ny[j], nx[j] - 50, ny[j] + 50, nx[j], ny[j], nx[j],
ny[j]);
                                arrow((180 * 3.1416) / 180, nx[j] - dx, ny[j]);
                            }
                        }
                        if ((ny[i] == ny[j]) && (nx[j] != nx[i] + num) && (nx[j] != nx[i] - num)) {
                            if (i <= 4) {
                                if (nx[i] < nx[j]) {
                                    if (nx[i] + 3 * num == nx[j]) {
                                        Arc(hdc, nx[i], ny[i] - 70, nx[j], ny[j] + 70, nx[j], ny[j],
nx[i], ny[i]);
                                        arrow((-140 * 3.1416) / 180, nx[j] - 16 * cos(-45), ny[j] + 16 *
sin(-45) - 3);
                                    } else {
                                        Arc(hdc, nx[i], ny[i] - 50, nx[j], ny[j] + 50, nx[j], ny[j],
nx[i], ny[i]);
                                        arrow((-145 * 3.1416) / 180, nx[j] - 16 * cos(-45) - 2, ny[j] +
16 * sin(-45) - 2);
                                    }
                                } else if (nx[i] > nx[j]) {
                                    if (nx[i] == nx[j] + 3 * num) {
                                        Arc(hdc, nx[i], ny[i] - 70, nx[j], ny[j] + 70, nx[j], ny[j],
nx[i], ny[i]);
                                        arrow((40 * 3.1416) / 180, nx[j] + dx * cos(-145) - 7, ny[j] + dy
* sin(-145) + 27);
                                    } else {
                                        Arc(hdc, nx[i], ny[i] - 50, nx[j], ny[j] + 50, nx[j], ny[j],
nx[i], ny[i]);
                                        arrow((25 * 3.1416) / 180, nx[j] + dx * cos(-145) - 7, ny[j] + dy
* sin(-145) + 25);
                                    }
                                }
                            } else if (i >= n * 0.5 && i <= 10) {
                                if (nx[i] < nx[j]) {
                                    if ((nx[i] + 3 * num == nx[j]) || (nx[i] == nx[j] + 3 * num)) {
                                        Arc(hdc, nx[i], ny[i] - 70, nx[j], ny[j] + 70, nx[j], ny[j],
nx[i], ny[i]);
                                        arrow((-140 * 3.1416) / 180, nx[j] + dx * cos(-145) - 28, ny[j] +
dy * sin(-145) - 8);
                                    } else {

                                        Arc(hdc, nx[i], ny[i] - 30, nx[j], ny[j] + 30, nx[j], ny[j],
nx[i], ny[i]);
                                        arrow((-160 * 3.1416) / 180, nx[j] + dx * cos(-145) - 34, ny[j] +
dy * sin(-145) - 2);
                                    }
                                } else if (nx[i] > nx[j]) {
                                    Arc(hdc, nx[i], ny[i] - 40, nx[j], ny[j] + 40, nx[j], ny[j], nx[i],
ny[i]);
                                    if ((nx[i] + 3 * num == nx[j]) || (nx[i] == nx[j] + 3 * num)) {
                                        arrow((20 * 3.1416) / 180, nx[j] + dx * cos(-145) - 2, ny[j] + dy
* sin(-145) + 22);
                                    } else {
                                        arrow((25 * 3.1416) / 180, nx[j] + dx * cos(-145) - 2, ny[j] + dy
* sin(-145) + 24);
                                    }
                                }
                            }
```

```
                            } else if (((nx[i] == nx[j]) && (ny[j] != ny[i] + num) && (ny[j] != ny[i] - num)
&& (nx[i] == num || nx[i] == num * 4)) &&
                                    ((nx[i] == nx[j]) && (ny[j] != ny[i] + num * 1.5) && (ny[j] != ny[i] -
num * 1.5))) {
                                if (i >= n * 0.25 && i <= 7) {
                                    if (ny[i] < ny[j]) {
                                        Arc(hdc, nx[i] - 40, ny[i], nx[j] + 40, ny[j], nx[j], ny[j], nx[i],
ny[i]);
                                        if (ny[i] + 2 * num == ny[j]) {
                                            arrow((-70 * 3.1416) / 180, nx[j] + dx * cos(-145) - 2, ny[j] +
dy * sin(-145) - 5);
                                        } else {
                                            arrow((-70 * 3.1416) / 180, nx[j] + dx * cos(-145) - 4, ny[j] +
dy * sin(-145) - 7);
                                        }
                                    } else if (ny[i] > ny[j]) {
                                        Arc(hdc, nx[j] - 100, ny[j], nx[i] + 100, ny[i], nx[i], ny[i], nx[j],
ny[j]);
                                        if (ny[i] == ny[j] + 2 * num) {
                                            arrow((30 * 3.1416) / 180, nx[j] + dx * cos(-145) + 1, ny[j] + dy
* sin(-145) + 13);
                                        } else {
                                            arrow((40 * 3.1416) / 180, nx[j] + dx * cos(-145) + 2, ny[j] + dy
* sin(-145) + 15);
                                        }
                                    }
                                } else if ((i >= 9) || (j >= 9)) {
                                    if (ny[i] < ny[j]) {
                                        Arc(hdc, nx[j] - 80, ny[j], nx[i] + 80, ny[i], nx[i], ny[i], nx[j],
ny[j]);
                                        arrow((-130 * 3.1416) / 180, nx[j] + dx * cos(-145) - 36, ny[j] + dy
* sin(-145) + 2);
                                    } else if (ny[i] > ny[j]) {
                                        Arc(hdc, nx[i] - 40, ny[i], nx[j] + 40, ny[j], nx[j], ny[j], nx[i],
ny[i]);
                                        arrow((110 * 3.1416) / 180, nx[j] + dx * cos(-145) - 30, ny[j] + dy *
sin(-145) + 27);
                                    }
                                }
                            } else {
                                double fi = 3.141 + acos((nx[j] - nx[i]) / (sqrt((nx[j] - nx[i]) * (nx[j] -
nx[i]) + (ny[j] - ny[i]) * (ny[j] - ny[i]))));
                                if (ny[j] < ny[i]) fi *= -1;

                                if (graphMatrix[i][j] == graphMatrix[j][i] && i < j) {
                                    if ((ny[i] + 3 * num == ny[j]) || (ny[i] == ny[j] + 3 * num)) {
                                        MoveToEx(hdc, nx[i] + 5, ny[i] + 5, NULL);
                                        LineTo(hdc, nx[j] + 5, ny[j] + 5);
                                        if (nx[i] == nx[j] + 3 * num) {
                                            arrow(fi, nx[j] + dx * cos(fi) + 5, ny[j] + dy * sin(fi) + 7);
                                        } else {
                                            if (nx[i] == nx[j]) {
                                                arrow(fi, nx[j] + dx * cos(fi) + 7, ny[j] + dy * sin(fi));
                                            } else {
                                                if (nx[i] < nx[j]) {
                                                    arrow(fi, nx[j] + dx * cos(fi) + 2, ny[j] + dy *
sin(fi));
                                                } else {
                                                    arrow(fi, nx[j] + dx * cos(fi) + 6, ny[j] + dy * sin(fi)
+ 3);
                                                }
                                            }
                                        }
                                    } else {
                                        MoveToEx(hdc, nx[i] + 10, ny[i] + 5, NULL);
                                        LineTo(hdc, nx[j] + 15, ny[j] + 5);
                                        if (ny[i] + num == ny[j]) {
                                            arrow(fi + 0.1, nx[j] + dx * cos(fi) + 5, ny[j] + 15 + dy *
sin(fi));
                                        } else if ((ny[i] == ny[j] + num * 1.5) || (ny[i] + num * 1.5 ==
ny[j])) {
                                            if (nx[i] == nx[j]) {
                                                arrow(fi, nx[j] + dx * cos(fi) + 7, ny[j] + 5 + dy * sin(fi)
- 4);
                                            } else {
                                                if (nx[i] == nx[j] + 3 * num) {
                                                    arrow(fi, nx[j] + dx * cos(fi) + 2, ny[j] + dy * sin(fi)
+ 12);
                                                } else {
                                                    arrow(fi, nx[j] + dx * cos(fi) - 2, ny[j] + dy * sin(fi)
+ 2);
                                                }
                                            }
                                        } else {
```

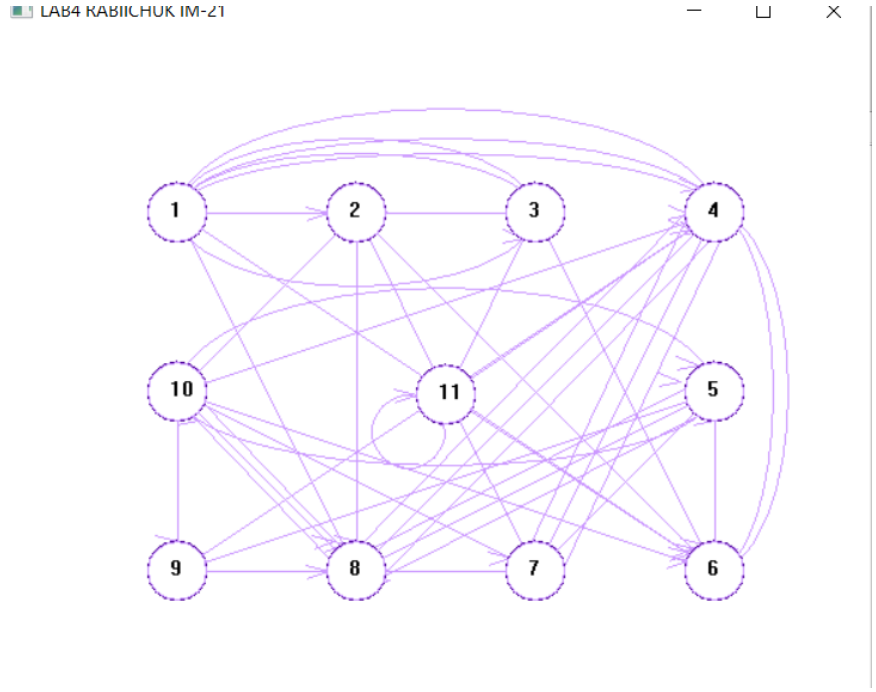
```
                                        if ((nx[i] == nx[j] + num) || (nx[i] + num == nx[j])) {
                                            arrow(fi, nx[j] + dx * cos(fi), ny[j] + 5 + dy * sin(fi));
                                        } else {
                                            if (ny[i] + 2 * num == ny[j]) {
                                                arrow(fi, nx[j] + dx * cos(fi) + 3, ny[j] + dy * sin(fi)
+ 14);
                                            } else {
                                                arrow(fi, nx[j] + dx * cos(fi), ny[j] + 3 + dy *
sin(fi));
                                            }
                                        }
                                    }
                                }
                            } else if (graphMatrix[i][j] == graphMatrix[j][i] && i > j) {
                                if ((ny[i] + 3 * num == ny[j]) || (ny[i] == ny[j] + 3 * num)) {
                                    MoveToEx(hdc, nx[i] - 10, ny[i] - 5, NULL);
                                    LineTo(hdc, nx[j] - 15, ny[j] - 5);
                                    if (nx[i] + 3 * num == nx[j]) {
                                        arrow(fi, nx[j] + dx * cos(fi) - 7, ny[j] - 5 + dy * sin(fi) -
7);
                                    } else {
                                        if (nx[i] == nx[j]) {
                                            arrow(fi, nx[j] + dx * cos(fi) - 14, ny[j] - 5 + dy *
sin(fi));
                                        } else {
                                            if (nx[i] > nx[j]) {
                                                arrow(fi, nx[j] + dx * cos(fi) - 10, ny[j] - 5 + dy *
sin(fi) + 7);
                                            } else {
                                                arrow(fi, nx[j] + dx * cos(fi) - 12, ny[j] - 5 + dy *
sin(fi) - 3);
                                            }
                                        }
                                    }
                                } else {
                                    MoveToEx(hdc, nx[i] - 10, ny[i] - 5, NULL);
                                    LineTo(hdc, nx[j] - 15, ny[j] - 5);
                                    if (ny[i] == ny[j] + num) {
                                        arrow(fi - 0.1, nx[j] + dx * cos(fi) - 5, ny[j] + dy * sin(fi) -
15);
                                    } else if ((ny[i] == ny[j] + num * 1.5) || (ny[i] + num * 1.5 ==
ny[j])) {
                                        if (nx[i] == nx[j]) {
                                            arrow(fi, nx[j] + dx * cos(fi) - 4, ny[j] - 5 + dy * sin(fi)
+ 5);
                                        } else {
                                            if (nx[i] + 3 * num == nx[j]) {
                                                arrow(fi, nx[j] + dx * cos(fi) - 4, ny[j] + dy * sin(fi)
- 10);
                                            } else {
                                                arrow(fi, nx[j] + dx * cos(fi) + 1, ny[j] - 5 + dy *
sin(fi) + 2);
                                            }
                                        }
                                    } else {
                                        if ((nx[i] == nx[j] + num) || (nx[i] + num == nx[j])) {
                                            arrow(fi, nx[j] + dx * cos(fi), ny[j] - 5 + dy * sin(fi));
                                        } else {
                                            if (ny[i] == ny[j] + 2 * num) {
                                                arrow(fi, nx[j] + dx * cos(fi) - 2, ny[j] + dy * sin(fi)
- 13);
                                            } else {
                                                arrow(fi, nx[j] + dx * cos(fi), ny[j] - 2 + dy *
sin(fi));
                                            }
                                        }
                                    }
                                }
                            } else {
                                LineTo(hdc, nx[j], ny[j]);
                                arrow(fi, nx[j] + dx * cos(fi) + 1, ny[j] + dy * sin(fi));
                            }
                        }
                    }
                }

                SelectObject(hdc, BPen);
                for (i = 0; i < n; i++) {
                    Ellipse(hdc, nx[i] - dx, ny[i] - dy, nx[i] + dx, ny[i] + dy);
                    if (i < 9) {
                        TextOut(hdc, nx[i] - dtx, ny[i] - dy / 2, nn[i], 1);
                    } else TextOut(hdc, nx[i] - dtx, ny[i] - dy / 2, nn[i], 2);
                }
```

```
            }

            void drawGraphUndir(double ** graphMatrix) {
                SelectObject(hdc, KPen);

                for (int i = 0; i < 11; i++) {
                    for (int j = 0; j < 11; j++) {
                        if (graphMatrix[i][j] == 1) {
                            MoveToEx(hdc, nx[i], ny[i], NULL);

                            if ((ny[i] == ny[j]) && (nx[j] != nx[i] + num) && (nx[j] != nx[i] - num)) {
                                if (i < 4) {
                                    if (nx[i] < nx[j]) {
                                        Arc(hdc, nx[i], ny[i] - 50, nx[j], ny[j] + 50, nx[j], ny[j], nx[i],
ny[i]);
                                    } else if (nx[i] > nx[j]) {
                                        Arc(hdc, nx[j], ny[j] - 40, nx[i], ny[i] + 40, nx[i], ny[i], nx[j],
ny[j]);
                                    }
                                } else if (i > 5 && i < 10) {
                                    if (nx[i] < nx[j]) {
                                        Arc(hdc, nx[j], ny[j] - 50, nx[i], ny[i] + 50, nx[i], ny[i], nx[j],
ny[j]);
                                    } else if (nx[i] > nx[j]) {
                                        Arc(hdc, nx[i], ny[i] - 40, nx[j], ny[j] + 40, nx[j], ny[j], nx[i],
ny[i]);
                                    }
                                }
                            } else if (((nx[i] == nx[j]) && (ny[j] != ny[i] + num) && (ny[j] != ny[i] - num)
&& (nx[i] == num || nx[i] == num * 4)) &&
                                    ((nx[i] == nx[j]) && (ny[j] != ny[i] + num * 1.5) && (ny[j] != ny[i] -
num * 1.5))) {

                                if (i > 2 && i < 7) {
                                    if (ny[i] < ny[j]) {
                                        Arc(hdc, nx[i] - 40, ny[i], nx[j] + 40, ny[j], nx[j], ny[j], nx[i],
ny[i]);
                                    } else if (ny[i] > ny[j]) {
                                        Arc(hdc, nx[j] - 50, ny[j], nx[i] + 50, ny[i], nx[i], ny[i], nx[j],
ny[j]);
                                    }
                                } else if (i > 8) {
                                    if (ny[i] < ny[j]) {
                                        Arc(hdc, nx[j] - 40, ny[j], nx[i] + 40, ny[i], nx[i], ny[i], nx[j],
ny[j]);
                                    } else if (ny[i] > ny[j]) {
                                        Arc(hdc, nx[i] - 40, ny[i], nx[j] + 40, ny[j], nx[j], ny[j], nx[i],
ny[i]);
                                    }
                                }
                            } else {
                                LineTo(hdc, nx[j], ny[j]);
                            }
                        }
                    }
                }

                for (i = 0; i < 11; i++) {
                    Ellipse(hdc, nx[i] - dx, ny[i] - dy, nx[i] + dx, ny[i] + dy);
                    if (i < 10) {
                        TextOut(hdc, nx[i] - dtx, ny[i] - dy / 2, nn[i], 2);
                    } else TextOut(hdc, nx[i] - dtx, ny[i] - dy / 2, nn[i], 2);
                }
            }

            double ** reach_connectivity_Matrix(double ** matrix, int rows, int cols) {
                printf("\n\nThe reach matrix:\n");
                int reach[rows][cols];
                int transitiveMatrix[rows][cols];
                int connectivityMatrix[rows][cols];

                for (int i = 0; i < rows; i++)
                    for (int j = 0; j < cols; j++)
                        reach[i][j] = matrix[i][j];

                for (int k = 0; k < rows; k++) {
                    for (int i = 0; i < cols; i++) {
                        for (int j = 0; j < cols; j++) {
                            reach[i][j] = reach[i][j] ||
                                          (reach[i][k] && reach[k][j]);
                        }
                    }
                }
```

```
            for (int i = 0; i < rows; i++) {
                for (int j = 0; j < cols; j++) {
                    printf("%d ", reach[i][j]);
                }
                printf("\n");
            }
            for (int i = 0; i < 11; i++) {
                for (int j = 0; j < 11; j++) {
                    transitiveMatrix[i][j] = reach[j][i];
                }
            }
             {
                int counter = 1;
                int countery = 1;
                double ** used_verticles = (double * ) malloc(n * sizeof(double * ));
                for (int i = 0; i < n; i++)
                    used_verticles[i] = (double * ) malloc(n * sizeof(double));
                double ** components = (double * ) malloc(n * sizeof(double * ));
                for (int i = 0; i < n; i++)
                    components[i] = (double * ) malloc(n * sizeof(double));
                double powered[n][n];
                for (int i = 0; i < n; i++) {
                    for (int j = 0; j < n; j++) {
                        powered[i][j] = 0;
                        for (int k = 0; k < n; k++)
                            powered[i][j] += reach[i][k] * reach[k][i];
                    }
                }
                for (int count = 0; count < rows; count++) {
                    counter = 1;
                    for (int i = 0; i < cols; i++) {
                        if (used_verticles[i] == 1) continue;
                        for (int j = 0; j < rows; j++) {
                            if (powered[count][j] != powered[i][j]) break;
                            if (j == rows - 1) {
                                used_verticles[i] = 1;
                                components[countery][i] = 1;
                                printf("Komponent %d\n", countery);
                                counter++;
                            }
                        }
                    }
                    if (components[countery][0] || components[countery][1] || components[countery][2] ||
                        components[countery][3] || components[countery][4] || components[countery][5] ||
                        components[countery][6] || components[countery][7] || components[countery][8] ||
                        components[countery][9] || components[countery][10]) countery++;
                }

                for (int i = 0; i < 11; i++) {
                    for (int j = 0; j < 11; j++) {
                        if (reach[i][j] == 1 && transitiveMatrix[i][j] == 1) {
                            connectivityMatrix[i][j] = 1;
                        } else {
                            connectivityMatrix[i][j] = 0;
                        }
                    }
                }
                printf("\n");
                printf("Connectivity matrix: \n");
                for (int i = 0; i < 11; i++)
                {
                    for (int j = 0; j < 11; j++) {
                        printf("%d ", connectivityMatrix[i][j]);
                    }
                    printf("\n");
                }
            }
        }

        printf("Directed graph:\n");
        printGrapgMatrix(A);
        printf("Undirected graph:\n");
        printGrapgMatrix(B);

        drawGraphDir(A);
        drawGraphUndir(B);
        graphPower(A, B, n, n);
        wayMatrix(A, n, n);
        reach_connectivity_Matrix(A, n, n);

        EndPaint(hWnd, & ps);
        break;

    case WM_DESTROY:
```

```
            PostQuitMessage(0);
            break;

        default:
            return (DefWindowProc(hWnd, messg, wParam, lParam));
    }
    return 0;
}
```

Результат тестування програми:




```
Directed graph:
0 1 0 1 0 1 0 1 1 0 0
0 0 0 0 0 1 0 1 0 1 0
1 1 0 0 0 1 0 0 0 0 0
0 0 0 0 0 1 1 1 1 1 0
0 0 0 0 0 0 1 1 1 0 0
0 0 0 1 0 0 0 0 0 0 1
0 0 0 1 0 0 0 1 0 0 1
0 0 0 1 1 0 0 0 0 1 0
0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 1 1 1 1 1 0 0
0 1 1 1 0 0 0 0 0 0 1
Undirected graph:
0 1 1 1 0 1 0 1 1 0 0
1 0 1 0 0 1 0 1 0 1 1
1 1 0 0 0 1 0 0 0 0 1
1 0 0 0 0 1 1 1 1 1 1
0 0 0 0 0 1 1 1 1 1 0
1 1 1 1 1 0 0 0 0 1 1
0 0 0 1 1 0 0 1 0 1 1
1 1 0 1 1 0 1 0 1 1 0
1 0 0 1 1 0 0 1 0 1 0
0 1 0 1 1 1 1 1 1 0 0
0 1 1 1 0 1 1 0 0 0 1


Graph top: 1 , deg+ is: 5  Graph top: 2 , deg+ is: 3  Graph top: 3 , deg+ is: 3  Graph top: 4 , deg+ is: 5  Graph top: 5 , deg+ is: 3  Graph top: 6 , deg+ is: 2  Graph top: 7 , deg+ is: 3  Graph top: 8 , deg+ is
: 3  Graph top: 9 , deg+ is: 1  Graph top: 10 , deg+ is: 5  Graph top: 11 , deg+ is: 4

Graph top: 1 , deg- is: 1  Graph top: 2 , deg- is: 3  Graph top: 3 , deg- is: 1  Graph top: 4 , deg- is: 4  Graph top: 5 , deg- is: 3  Graph top: 6 , deg- is: 5  Graph top: 7 , deg- is: 3  Graph top: 8 , deg- is
: 7  Graph top: 9 , deg- is: 4  Graph top: 10 , deg- is: 3  Graph top: 11 , deg- is: 3

Graph top: 1 , deg is: 6  Graph top: 2 , deg is: 6  Graph top: 3 , deg is: 4  Graph top: 4 , deg is: 7  Graph top: 5 , deg is: 5  Graph top: 6 , deg is: 7  Graph top: 7 , deg is: 5  Graph top: 8 , deg is: 7  Gra
ph top: 9 , deg is: 5  Graph top: 10 , deg is: 7  Graph top: 11 , deg is: 6

2-length ways
| 1->2->6  || 1->2->8  || 1->2->10  || 1->4->6  || 1->4->7  || 1->4->8  || 1->4->9  || 1->4->10  || 1->6->5  || 1->6->11  || 1->8->4  || 1->8->5  || 1->8->10  || 1->9->8  |
| 2->6->5  || 2->6->11  || 2->8->4  || 2->8->5  || 2->8->10  || 2->10->5  || 2->10->6  || 2->10->7  || 2->10->8  || 2->10->9  |
| 3->1->2  || 3->1->4  || 3->1->6  || 3->1->8  || 3->1->9  || 3->2->6  || 3->2->8  || 3->2->10  || 3->6->5  || 3->6->11  |
| 4->6->5  || 4->6->11  || 4->7->4  || 4->7->8  || 4->7->11  || 4->8->4  || 4->8->5  || 4->8->10  || 4->9->8  || 4->10->5  || 4->10->6  || 4->10->7  || 4->10->8  || 4->10->9  |
| 5->7->4  || 5->7->8  || 5->7->11  || 5->8->4  || 5->8->5  || 5->8->10  || 5->9->8  |
| 6->5->7  || 6->5->8  || 6->5->9  || 6->11->2  || 6->11->3  || 6->11->4  || 6->11->11  |
| 7->4->6  || 7->4->7  || 7->4->8  || 7->4->9  || 7->4->10  || 7->8->4  || 7->8->5  || 7->8->10  || 7->11->2  || 7->11->3  || 7->11->4  || 7->11->11  |
| 8->4->6  || 8->4->7  || 8->4->8  || 8->4->9  || 8->4->10  || 8->5->7  || 8->5->8  || 8->5->9  || 8->10->5  || 8->10->6  || 8->10->7  || 8->10->8  || 8->10->9  |
| 9->8->4  || 9->8->5  || 9->8->10  |
| 10->5->7  || 10->5->8  || 10->5->9  || 10->6->5  || 10->6->11  || 10->7->4  || 10->7->8  || 10->7->11  || 10->8->4  || 10->8->5  || 10->8->10  || 10->9->8  |
| 11->2->6  || 11->2->8  || 11->2->10  || 11->3->1  || 11->3->2  || 11->3->6  || 11->4->6  || 11->4->7  || 11->4->8  || 11->4->9  || 11->4->10  || 11->11->2  || 11->11->3  || 11->11->4  || 11->11->11  |
```

```
3-length ways
| 1->2->6->5 || 1->2->6->11 || 1->2->8->4 || 1->2->8->5 || 1->2->8->10 || 1->2->10->5 || 1->2->10->6 || 1->2->10->7 || 1->2->10->8 || 1->2->10->9 || 1->4->6->5 || 1->4->6->11 || 1->4->7->4 || 1->4->7->8 || 1->4-
>7->11 || 1->4->8->4 || 1->4->8->5 || 1->4->8->10 || 1->4->9->8 || 1->4->10->5 || 1->4->10->6 || 1->4->10->7 || 1->4->10->8 || 1->4->10->9 || 1->6->5->7 || 1->6->5->8 || 1->6->5->9 || 1->6->11->2 || 1->6->11->3
|| 1->6->11->4 || 1->6->11->11 || 1->8->4->6 || 1->8->4->7 || 1->8->4->8 || 1->8->4->9 || 1->8->4->10 || 1->8->5->7 || 1->8->5->8 || 1->8->5->9 || 1->8->10->5 || 1->8->10->6 || 1->8->10->7 || 1->8->10->8 || 1->8
->10->9 || 1->9->8->4 || 1->9->8->5 || 1->9->8->10 |
| 2->6->5->7 || 2->6->5->8 || 2->6->5->9 || 2->6->11->2 || 2->6->11->3 || 2->6->11->4 || 2->6->11->11 || 2->8->4->6 || 2->8->4->7 || 2->8->4->8 || 2->8->4->9 || 2->8->4->10 || 2->8->5->7 || 2->8->5->8 || 2->8->5
->9 || 2->8->10->5 || 2->8->10->6 || 2->8->10->7 || 2->8->10->8 || 2->8->10->9 || 2->10->5->7 || 2->10->5->8 || 2->10->5->9 || 2->10->6->5 || 2->10->6->11 || 2->10->7->4 || 2->10->7->8 || 2->10->7->11 || 2->10->
8->4 || 2->10->8->5 || 2->10->8->10 || 2->10->9->8 |
| 3->1->2->6 || 3->1->2->8 || 3->1->2->10 || 3->1->4->6 || 3->1->4->7 || 3->1->4->8 || 3->1->4->9 || 3->1->4->10 || 3->1->6->5 || 3->1->6->11 || 3->1->8->4 || 3->1->8->5 || 3->1->8->10 || 3->1->9->8 || 3->2->6->
5 || 3->2->6->11 || 3->2->8->4 || 3->2->8->5 || 3->2->8->10 || 3->2->10->5 || 3->2->10->6 || 3->2->10->7 || 3->2->10->8 || 3->2->10->9 || 3->6->5->7 || 3->6->5->8 || 3->6->5->9 || 3->6->11->2 || 3->6->11->3 || 3
->6->11->4 || 3->6->11->11 |
| 4->6->5->7 || 4->6->5->8 || 4->6->5->9 || 4->6->11->2 || 4->6->11->3 || 4->6->11->4 || 4->6->11->11 || 4->7->4->6 || 4->7->4->7 || 4->7->4->8 || 4->7->4->9 || 4->7->4->10 || 4->7->8->4 || 4->7->8->5 || 4->7->8
->10 || 4->7->11->2 || 4->7->11->3 || 4->7->11->4 || 4->7->11->11 || 4->8->4->6 || 4->8->4->7 || 4->8->4->8 || 4->8->4->9 || 4->8->4->10 || 4->8->5->7 || 4->8->5->8 || 4->8->5->9 || 4->8->10->5 || 4->8->10->6 ||
 4->8->10->7 || 4->8->10->8 || 4->8->10->9 || 4->9->8->4 || 4->9->8->5 || 4->9->8->10 || 4->10->5->7 || 4->10->5->8 || 4->10->5->9 || 4->10->6->5 || 4->10->6->11 || 4->10->7->4 || 4->10->7->8 || 4->10->7->11 ||
4->10->8->4 || 4->10->8->5 || 4->10->8->10 || 4->10->9->8 |
| 5->7->4->6 || 5->7->4->7 || 5->7->4->8 || 5->7->4->9 || 5->7->4->10 || 5->7->8->4 || 5->7->8->5 || 5->7->8->10 || 5->7->11->2 || 5->7->11->3 || 5->7->11->4 || 5->7->11->11 || 5->8->4->6 || 5->8->4->7 || 5->8->
4->8 || 5->8->4->9 || 5->8->4->10 || 5->8->5->7 || 5->8->5->8 || 5->8->5->9 || 5->8->10->5 || 5->8->10->6 || 5->8->10->7 || 5->8->10->8 || 5->8->10->9 || 5->9->8->4 || 5->9->8->5 || 5->9->8->10 |
| 6->5->7->4 || 6->5->7->8 || 6->5->7->11 || 6->5->8->4 || 6->5->8->5 || 6->5->8->10 || 6->5->9->8 || 6->11->2->6 || 6->11->2->8 || 6->11->2->10 || 6->11->3->1 || 6->11->3->2 || 6->11->3->6 || 6->11->4->6 || 6->
11->4->7 || 6->11->4->8 || 6->11->4->9 || 6->11->4->10 || 6->11->11->2 || 6->11->11->3 || 6->11->11->4 || 6->11->11->11 |
| 7->4->6->5 || 7->4->6->11 || 7->4->7->4 || 7->4->7->8 || 7->4->7->11 || 7->4->8->4 || 7->4->8->5 || 7->4->8->10 || 7->4->9->8 || 7->4->10->5 || 7->4->10->6 || 7->4->10->7 || 7->4->10->8 || 7->4->10->9 || 7->8-
>4->6 || 7->8->4->7 || 7->8->4->8 || 7->8->4->9 || 7->8->4->10 || 7->8->5->7 || 7->8->5->8 || 7->8->5->9 || 7->8->10->5 || 7->8->10->6 || 7->8->10->7 || 7->8->10->8 || 7->8->10->9 || 7->11->2->6 || 7->11->2->8 |
| 7->11->2->10 || 7->11->3->1 || 7->11->3->2 || 7->11->3->6 || 7->11->4->6 || 7->11->4->7 || 7->11->4->8 || 7->11->4->9 || 7->11->4->10 || 7->11->11->2 || 7->11->11->3 || 7->11->11->4 || 7->11->11->11 |
| 8->4->6->5 || 8->4->6->11 || 8->4->7->4 || 8->4->7->8 || 8->4->7->11 || 8->4->8->4 || 8->4->8->5 || 8->4->8->10 || 8->4->9->8 || 8->4->10->5 || 8->4->10->6 || 8->4->10->7 || 8->4->10->8 || 8->4->10->9 || 8->5-
>7->4 || 8->5->7->8 || 8->5->7->11 || 8->5->8->4 || 8->5->8->5 || 8->5->8->10 || 8->5->9->8 || 8->10->5->7 || 8->10->5->8 || 8->10->5->9 || 8->10->6->5 || 8->10->6->11 || 8->10->7->4 || 8->10->7->8 || 8->10->7->
11 || 8->10->8->4 || 8->10->8->5 || 8->10->8->10 || 8->10->9->8 |
| 9->8->4->6 || 9->8->4->7 || 9->8->4->8 || 9->8->4->9 || 9->8->4->10 || 9->8->5->7 || 9->8->5->8 || 9->8->5->9 || 9->8->10->5 || 9->8->10->6 || 9->8->10->7 || 9->8->10->8 || 9->8->10->9 |
| 10->5->7->4 || 10->5->7->8 || 10->5->7->11 || 10->5->8->4 || 10->5->8->5 || 10->5->8->10 || 10->5->9->8 || 10->6->5->7 || 10->6->5->8 || 10->6->5->9 || 10->6->11->2 || 10->6->11->3 || 10->6->11->4 || 10->6->11
->11 || 10->7->4->6 || 10->7->4->7 || 10->7->4->8 || 10->7->4->9 || 10->7->4->10 || 10->7->8->4 || 10->7->8->5 || 10->7->8->10 || 10->7->11->2 || 10->7->11->3 || 10->7->11->4 || 10->7->11->11 || 10->8->4->6 || 1
0->8->4->7 || 10->8->4->8 || 10->8->4->9 || 10->8->4->10 || 10->8->5->7 || 10->8->5->8 || 10->8->5->9 || 10->8->10->5 || 10->8->10->6 || 10->8->10->7 || 10->8->10->8 || 10->8->10->9 || 10->9->8->4 || 10->9->8->5
|| 10->9->8->10 |
| 11->2->6->5 || 11->2->6->11 || 11->2->8->4 || 11->2->8->5 || 11->2->8->10 || 11->2->10->5 || 11->2->10->6 || 11->2->10->7 || 11->2->10->8 || 11->2->10->9 || 11->3->1->2 || 11->3->1->4 || 11->3->1->6 || 11->3->
1->8 || 11->3->1->9 || 11->3->2->6 || 11->3->2->8 || 11->3->2->10 || 11->3->6->5 || 11->3->6->11 || 11->4->6->5 || 11->4->6->11 || 11->4->7->4 || 11->4->7->8 || 11->4->7->11 || 11->4->8->4 || 11->4->8->5 || 11->
4->8->10 || 11->4->9->8 || 11->4->10->5 || 11->4->10->6 || 11->4->10->7 || 11->4->10->8 || 11->4->10->9 || 11->11->2->6 || 11->11->2->8 || 11->11->2->10 || 11->11->3->1 || 11->11->3->2 || 11->11->3->6 || 11->11-
>4->6 || 11->11->4->7 || 11->11->4->8 || 11->11->4->9 || 11->11->4->10 || 11->11->11->2 || 11->11->11->3 || 11->11->11->4 || 11->11->11->11 |


Reach 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
```

```
Komponent 1
Komponent 1
Komponent 1
Komponent 1
Komponent 1
Komponent 1
Komponent 1
Komponent 1
Komponent 1
Komponent 1
Komponent 1

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
```