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 №5

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 за варіантом формується за функцією: A = mulmr((1.0 – n3*0.01 – n4*0.005 – 0.15)*T);

2. Створити програми для обходу в глибину та в ширину. Обхід починати з вершини, яка має вихідні дуги. При цьому у програмі: — встановити зупинку у точці призначення номеру черговій вершині за допомогою повідомлення про натискання кнопки, — виводити зображення графа у графічному вікні перед кожною зупинкою.

3. Під час обходу графа побудувати дерево обходу. Вивести побудоване дерево у графічному вікні.

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 **createArray(int n) {
    double **matrix = (double *) malloc(n * sizeof(double *));

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

    return matrix;
}

double **createMatrix(int n) {
    double **matrix = (double **) malloc(n * sizeof(double *));

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

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++)
            matrix[i][j] = 2.0 / RAND_MAX * rand();
    }

    return matrix;
}

double **rewriteMatrix(double num, double **matrix, int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            matrix[i][j] = matrix[i][j] * num;

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

```
    return matrix;
}

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

char ProgName[] = "LAB5 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,
                        "LAB5 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;
}

void printMatrix(double **A, int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%g ", A[i][j]);
        }
        printf("\n");
    }
}

double added_this_tick[11];

void enqueue(int n, int queue[n + 1], int value) {
    for (int i = 0; i < n; i++) {
        if (queue[i] == -1) {
            queue[i] = value;
            added_this_tick[value] = 1.0f;
            break;
        }
    }
}

int dequeue(int n, int queue[n + 1]) {
    int ret = queue[0];
    for (int i = 1; i < n; i++) {
        queue[i - 1] = queue[i];
    }
    return ret;
}

void push(int n, int stack[n], int val) {
```

```
    for (int i = 0; i < n; i++) {
        if (stack[i] == -1) {
            stack[i] = val;
            added_this_tick[val] = 1.0f;
            break;
        }
    }
}

int pop(int n, int stack[n]) {
    int ret = -1;

    for (int i = 0; i < n; i++) {
        if (i + 1 == n || stack[i + 1] == -1) {
            ret = stack[i];
            stack[i] = -1;
            break;
        }
    }
    return ret;
}

int queue[11];
int stack[11];
bool visited[11];
bool sheduled[11];

int dfs(double **matrix, int n) {
    if (stack[0] != -1) {
        int v = pop(n, stack);

        if (!visited[v]) {
            visited[v] = 1;

            for (int i = 0; i < n; i++) {
                if (matrix[v][i] >= 0.95) {
                    if (!sheduled[i]) {
                        push(n, stack, i);
                        sheduled[i] = 1;
                    }
                }
            }
            return v;
        } else
            return dfs(matrix, n);
    }
}

int bfs(double **matrix, int n) {
    if (queue[0] != -1) {
        int v = dequeue(n, queue);

        for (int i = 0; i < n; i++) {
            if (matrix[v][i] >= 0.95) {
                if (!visited[i]) {
                    visited[i] = 1;
                    enqueue(n, queue, i);
                }
            }
        }
        return v;
    }
}

double **A;
double **parseTree;
int selected[11];
int tick;
bool drawparsetree;

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);
    }

    void line(double *nx, double *ny, int i, int j, double fi) {
        MoveToEx(hdc, nx[i], ny[i], NULL);
        LineTo(hdc, nx[j], ny[j]);
        arrow(fi, nx[j], ny[j]);
    }

    void shiftedLine(double *nx, double *ny, int i, int j, double fi) {
        int rad = 16;
        double px1 = nx[i] - rad * cos(fi - 3.1416 / 10);
        double py1 = ny[i] - rad * sin(fi - 3.1416 / 10);
        double px2 = nx[j] + rad * cos(fi + 3.1416 / 10);
        double py2 = ny[j] + rad * sin(fi + 3.1416 / 10);

        MoveToEx(hdc, px1, py1, NULL);
        LineTo(hdc, px2, py2);
        arrow(fi, px2 - 15 * cos(fi), py2 - 15 * sin(fi));
    }

    void bentLine(double *nx, double *ny, int i, int j, double fi, int centreX, int centreY) {

        int radius2 = 50;
        double pointX = centreX + radius2 * cos(fi + 3.1416 / 2);
        double pointY = centreY + radius2 * sin(fi + 3.1416 / 2);

        MoveToEx(hdc, nx[i], ny[i], NULL);
        LineTo(hdc, pointX, pointY);
        LineTo(hdc, nx[j], ny[j]);

        fi = 3.1416 + acos((nx[j] - pointX) / (sqrt(pow(nx[j] - pointX, 2) + pow(ny[j] - pointY,
2))));
        if (ny[j] < pointY) fi *= -1;

        arrow(fi, nx[j], ny[j]);
    }

    void loop(double *nx, double *ny, int i, int j, int centreX, int centreY) {
        int znakX = 1;
        int znakY = 1;
        if (nx[i] - centreX <= 0) znakX = -1;
        if (ny[i] - centreY <= 0) znakY = -1;

        Arc(hdc, nx[i], ny[i], nx[j] + znakX * 40, ny[j] + znakY * 40, nx[i], ny[i], nx[j],
ny[j]);

        if (znakX == 1 && znakY == -1) arrow(0, nx[j], ny[j] - 2);
        if (znakX == -1 && znakY == -1) arrow(3.1416 * 1.5, nx[j] - 2, ny[j]);
        if (znakX == -1 && znakY == 1) arrow(3.1416, nx[j], ny[j] + 1);
        if (znakX == 1 && znakY == 1) arrow(3.1416 * 0.5, nx[j] + 1, ny[j]);
    }

    double getFi(double *nx, double *ny, int i, int j) {
        double fi = 3.1416 + acos((nx[j] - nx[i]) / (sqrt(pow(nx[j] - nx[i], 2) + pow(ny[j] -
ny[i], 2))));
        if (ny[j] < ny[i]) fi *= -1;

        return fi;
    }

    switch (messg) {
        case WM_CREATE:

            for (int i = 0; i < 11; i++) {
                stack[i] = -1;
                queue[i] = -1;
                visited[i] = 0;
                sheduled[i] = 0;
                selected[i] = 0;
            }
            queue[11 + 1] = -1;

            stack[0] = 5;
```

```
        queue[0] = 5;

        tick = 1;
        drawparsetree = 0;

        parseTree = (double *) malloc(11 * sizeof(double *));

        for (int i = 0; i < 11; i++)
            parseTree[i] = (double *) malloc(11 * sizeof(double));

        CreateWindow(TEXT("button"), TEXT("Go"),
                     WS_VISIBLE | WS_CHILD,
                     250, 450, 100, 50,
                     hWnd, (HMENU) 1, NULL, NULL);
        break;

    case WM_PAINT:

        hdc = BeginPaint(hWnd, &ps);
        HPEN GPen = CreatePen(PS_SOLID, 3, RGB(76, 0, 53));
        HPEN BPen = CreatePen(PS_SOLID, 2, RGB(204, 153, 255));
        HPEN KPen = CreatePen(PS_SOLID, 1, RGB(20, 20, 5));

        srand(2118);
        double **T = createMatrix(11);
        double coefficient = 1.0 - 1 * 0.01 - 8 * 0.005 - 0.15;
        A = rewriteMatrix(coefficient, T, 11);

        if (drawparsetree) A = parseTree;

        printMatrix(A, 11);
        for (int i = 0; i < 11; i++) {
            if (selected[i] == 0)
                continue;
            printf("[%d %d];\n", i + 1, selected[i]);
        }
        printf("\n");

        int radius = 300;
        int dx = 16, dy = 16, dtx = 5;
        double k = 2 * 3.1416 / (11 - 1);
        char *numbers[20] = {"1", "2", "3", "4", "5", "6", "7", "8", "9", "10",
                             "11", "12", "13", "14", "15", "16", "17", "18", "19", "20"};
        double *nx = createArray(11);
        double *ny = createArray(11);

        int size = (11 + 2) / 4;
        nx[0] = 150;
        ny[0] = 100;
        int centreX = (nx[0] + size * 120) / 2 + 50;
        int centreY = (ny[0] + size * 120) / 2;
        for (int i = 0; i < 11; i++) {
            if (i == 0) {
                nx[i] = 120;
                ny[i] = 120;
            } else if (i < 4) {
                nx[i] = nx[i - 1] + 120;
                ny[i] = ny[i - 1];
            } else if (i < 6) {
                nx[i] = nx[i - 1];
                ny[i] = ny[i - 1] + 120;
            } else if (i < 9) {
                nx[i] = nx[i - 1] - 120;
                ny[i] = ny[i - 1];
            } else if (i < 10) {
                nx[i] = nx[i - 1];
                ny[i] = ny[i - 1] - 120;
            } else {
                nx[i] = nx[i - 1] + 3 * 120 / 2;
                ny[i] = ny[i - 1] + 2;
            }
        }
        nx[11 - 1] = centreX;
        ny[11 - 1] = centreY;

        for (int i = 0; i < 11; i++) {
            for (int j = 0; j < 11; j++) {

                if (selected[i] > 0)
                    SelectObject(hdc, GPen);
```

```
                    else
                        SelectObject(hdc, KPen);

                    if (A[i][j] == 1) {

                        if (A[i][j] == 1 && i == j) {
                            loop(nx, ny, i, j, centreX, centreY);
                        }
                        double fi = getFi(nx, ny, i, j);
                        if (abs(i - j) == (size) * 2) {
                            bentLine(nx, ny, i, j, fi, centreX, centreY);
                        } else if (nx[i] == nx[j] || ny[i] == ny[j]) {
                            bentLine(nx, ny, i, j, fi, (nx[i] + nx[j]) / 2, (ny[i] + ny[j]) / 2);
                        } else if (A[i][j] == A[j][i]) {
                            shiftedLine(nx, ny, i, j, fi);
                        } else {
                            line(nx, ny, i, j, fi);
                        }
                    }
                }
            }

            SelectObject(hdc, BPen);
            for (int i = 0; i < 11; 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, numbers[i], 1);
                } else
                    TextOut(hdc, nx[i] - dtx, ny[i] - dy / 2, numbers[i], 2);
            }

            EndPaint(hWnd, &ps);
            break;

        case WM_COMMAND:
            if (LOWORD(wParam) == 1) {

                int next_select = dfs(A, 11);

                if (next_select != -1) {
                    printf("%d, %d\n", next_select, tick);
                    selected[next_select] = tick;

                    for (int i = 0; i < 11; i++) {
                        parseTree[next_select][i] = added_this_tick[i];
                        added_this_tick[i] = 0;
                    }
                } else {
                    drawparsetree = 1;
                }

                RedrawWindow(hWnd, NULL, NULL, RDW_ERASE | RDW_INVALIDATE);
                tick++;
            }
            break;

        case WM_DESTROY:
            PostQuitMessage(0);
            break;

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

Результат виконання програми: