Министерство науки и высшего образования РФ
Пензенский государственный университет
Кафедра «Вычислительная техника»

ОТЧЕТ

по лабораторной работе №7
по дисциплине «логика и основы алгоритмизации в инженерных задачах»
на тему «Обход графа в глубину»

Выполнили:
студенты группы 24ВВВ4
Королёв Д.В
Алешин К.А

Приняли:
Юрова О.В
Деев М.В

Пенза 2024

Цель работы – научиться на практике реализации алгоритма обхода графа в ширину

Общие сведения – Обход графа – одна из наиболее распространенных операций с графами. Задачей обхода является прохождение всех вершин в графе. Обходы применяются для поиска информации, хранящейся в узлах графа, нахождения связей между вершинами или группами вершин и т.д.

Листинг

Задание 1

1)

```
#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <time.h>


void generate_adjacency_matrix(int n, int matrix[n][n]) {

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

            matrix[i][j] = 0;

        }

    }



    for (int i = 0; i < n; i++) {

        for (int j = i + 1; j < n; j++) {

            int edge = rand() % 2;

            matrix[i][j] = edge;

            matrix[j][i] = edge;
```

```
        }
    }
}

void dfs(int matrix[][10], int n, int vertex, bool
visited[]) {

    visited[vertex] = true;
    printf("%d ", vertex);

    for (int i = 0; i < n; i++) {
        if (matrix[vertex][i] == 1 && !visited[i]) {
            dfs(matrix, n, i, visited);
        }
    }
}

int main() {
    int n;
    printf("Введите размер матрицы: ");
    scanf("%d", &n);

    int matrix[n][n];
    bool visited[n]; // массив посещенных вершин

    for (int i = 0; i < n; i++) {
```

```
        visited[i] = false;
    }

    srand(time(NULL));

    generate_adjacency_matrix(n, matrix);

    printf("Adjacency Matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }

    // обход в глубину с вершины 0
    printf("Depth-First Search starting from vertex 0:\n");
    dfs(matrix, n, 0, visited);
    printf("\n");

    return 0;
}
```

2)

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <stdbool.h>

struct Node {
    int vertex;
    struct Node* next;
};

struct Graph {
    int numVertices;
    struct Node** adjLists;
};

struct Node* createNode(int vertex) {
    struct Node* newNode = malloc(sizeof(struct Node));
    newNode->vertex = vertex;
    newNode->next = NULL;
    return newNode;
}

// создание графа
struct Graph* createGraph(int vertices) {
    struct Graph* graph = malloc(sizeof(struct Graph));
    graph->numVertices = vertices;

    // массив списков смежности
```

```
    graph->adjLists = malloc(vertices * sizeof(struct
Node*));

    for (int i = 0; i < vertices; i++) {
        graph->adjLists[i] = NULL;
    }

    return graph;
}

// добавление ребра в граф
void addEdge(struct Graph* graph, int src, int dest) {
    struct Node* newNode = createNode(dest);
    newNode->next = graph->adjLists[src];
    graph->adjLists[src] = newNode;

    newNode = createNode(src);
    newNode->next = graph->adjLists[dest];
    graph->adjLists[dest] = newNode;
}

// обход в глубину
void dfs(struct Graph* graph, int vertex, bool visited[]) {
    visited[vertex] = true;
    printf("%d ", vertex);
```

```
    struct Node* adjList = graph->adjLists[vertex];
    while (adjList != NULL) {
        int connectedVertex = adjList->vertex;
        if (!visited[connectedVertex]) {
            dfs(graph, connectedVertex, visited);
        }
        adjList = adjList->next;
    }
}

int main() {
    int vertices = 5; // Количество вершин
    struct Graph* graph = createGraph(vertices);

    // Добавляем ребра
    addEdge(graph, 0, 1);
    addEdge(graph, 0, 4);
    addEdge(graph, 1, 2); // ребро между 1 и 2
    addEdge(graph, 1, 3);
    addEdge(graph, 1, 4);
    addEdge(graph, 2, 3);
    addEdge(graph, 3, 4);

    bool visited[vertices];
    for (int i = 0; i < vertices; i++) {
```

```
        visited[i] = false;
    }

    printf("Depth-First Search starting from vertex 0:\n");
    dfs(graph, 0, visited);
    printf("\n");

    return 0;
}
```

Задание 2

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>

#define MAX_VERTICES 100 // Максимальное количество вершин

void generate_adjacency_matrix(int n, int matrix[n][n]) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            matrix[i][j] = 0;
        }
    }

    for (int i = 0; i < n; i++) {
```

```
        for (int j = i + 1; j < n; j++) {
            int edge = rand() % 2;
            matrix[i][j] = edge;
            matrix[j][i] = edge;
        }
    }
}

void dfs_iterative(int matrix[][MAX_VERTICES], int n, int start_vertex) {
    bool visited[MAX_VERTICES] = {false}; // Массив посещенных вершин
    int stack[MAX_VERTICES]; // Стек для хранения вершин
    int top = -1; // Индекс верхушки стека

    // начинаем с начальной вершины
    stack[++top] = start_vertex;

    while (top != -1) {
        // Извлекаем вершину из стека
        int vertex = stack[top--];

        if (!visited[vertex]) {
            visited[vertex] = true;
            printf("%d ", vertex);
```

```
            // добавляем соседние вершины в стек
            for (int i = n - 1; i >= 0; i--) { // обход в
обратном порядке
                if (matrix[vertex][i] == 1 && !visited[i]) {
                    stack[++top] = i;
                }
            }
        }
    }
}

int main() {
    int n;
    printf("Введите размер матрицы: ");
    scanf("%d", &n);

    int matrix[n][n];

    srand(time(NULL));

    generate_adjacency_matrix(n, matrix);

    printf("Adjacency Matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", matrix[i][j]);
```

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

    printf("Depth-First Search starting from vertex 0:\n");
    dfs_iterative(matrix, n, 0);
    printf("\n");

    return 0;
}
```

Рисунок 1– Результат выполнения 1-го задания 1 часть

```
Введите размер матрицы: 10
Adjacency Matrix:
0 0 0 1 0 1 0 1 1 0
0 0 1 0 1 0 0 1 0 1
0 1 0 1 1 0 0 0 1 1
1 0 1 0 1 1 1 0 1 0
0 1 1 1 0 1 0 0 0 0
1 0 0 1 1 0 1 1 1 1
0 0 0 1 0 1 0 1 0 1
1 1 0 0 0 1 1 0 0 0
1 0 1 1 0 1 0 0 0 1
0 1 1 0 0 1 1 0 1 0
Depth-First Search starting from vertex 0:
0 3 2 1 4 5 6 7 9 8
```

Рисунок 2– Результат выполнения 1-го задания 2 часть

```
Depth-First Search starting from vertex 0:
0 4 3 2 1
```

Рисунок 3– Результат выполнения 2-го задания

```
Введите размер матрицы: 10
Adjacency Matrix:
0 1 1 0 0 1 0 0 0 1
1 0 0 0 0 1 0 0 1 1
1 0 0 0 1 0 1 1 0 1
0 0 0 0 0 1 0 0 1 1
0 0 1 0 0 0 0 0 0 0
1 1 0 1 0 0 1 0 1 0
0 0 1 0 0 1 0 1 1 0
0 0 1 0 0 0 1 0 1 1
0 1 0 1 0 1 1 1 0 0
1 1 1 1 0 0 0 1 0 0
Depth-First Search starting from vertex 0:
0 1 2 5 9
```

Вывод: В ходе выполнения лабораторной работы были изучены на практике навыки реализации алгоритма обхода графа в ширину