Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Звіт

з лабораторної роботи No 3 з дисципліни

«Алгоритми та структури даних 2. Структури даних»

«Прикладні задачі теорії графів »

Виконав (ла) *ІП-22, Андреєва Уляна Андріївна*

(шифр , прізвище , ім' я, по батькові)

Перевірила *Халус Олена Андріївна*

(прізвище , ім'я, по батькові)

Київ 2023

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи – вивчити основні прикладні алгоритми на графах та способи їх імплементації.

2 ЗАВДАННЯ

Варіант 1

№	Задача	Алгоритм	Тип графу	Спосіб задання графу
1	Обхід графу	DFS	Неорієнтований	Матриця суміжності

3.1 Псевдокод алгоритму

```
function dfs(G, u):
    vis[u] = 1
    print u
    for v from 0 to G.V:
        if vis[v] == 0 and G.Adj[u][v] == 1:
            dfs(G, v)


function dfsTraversal(G):
    fill vis array with 0s
    for i from 0 to G.V:
        if vis[i] == 0:
            dfs(G, i)
```

3.2 Вихідний код

```
import java.util.*;

class Graph {
    int V, E;
    public int[][] Adj;

    public Graph(int v, int e) {
        V = v;
        E = e;
        Adj = new int[V][V];
    }

    void addEdge(int u, int v) {
        Adj[u][v] = 1;
        Adj[v][u] = 1;
    }
}
class DFS {
    int[] vis = new int[100];
    void dfs(Graph G, int u) {
        vis[u] = 1;
        System.out.print(u + " ");
        for (int v = 0; v < G.V; v++) {
            if (vis[v] == 0 && G.Adj[u][v] == 1) {
                dfs(G, v);
            }
        }
    }

    void dfsTraversal(Graph G) {
        Arrays.fill(vis, 0);
        for (int i = 0; i < G.V; i++) {
            if (vis[i] == 0) {
                dfs(G, i);
            }
        }
    }
}

class Main {
    public static void main(String[] args) {
        Scanner input = new Scanner ( System.in );
        System.out.print ( "Enter the number of vertices: " );
        int v = input.nextInt ();
        Graph G = new Graph ( v, v * (v - 1) / 2 );
        Random random = new Random ();
        for (int i = 0; i < v; i++) {
            for (int j = i + 1; j < v; j++) {
                int randNum = random.nextInt ( 2 );
                if (randNum == 1) {
                    G.addEdge ( i, j );
                }
            }
```

```
        }

        System.out.println("Adjacency matrix:");
        for (int i = 0; i < G.V; i++) {
            for (int j = 0; j < G.V; j++) {
                System.out.print(G.Adj[i][j] + ", ");
            }
            System.out.println();
        }
        System.out.println("\n");

        DFS dfs = new DFS ();
        System.out.print( "DFS traversal result: ");
        dfs.dfsTraversal ( G );
    }
}
```

3.2.1 Програмна реалізація

```
Enter the number of vertices: 5
Adjacency matrix:
0, 1, 1, 1, 1,
1, 0, 0, 1, 0,
1, 0, 0, 1, 1,
1, 1, 1, 0, 1,
1, 0, 1, 1, 0,


DFS traversal result: 0 1 3 2 4
```

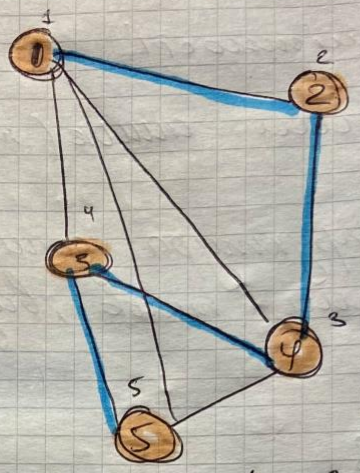
```
Enter the number of vertices: 10
Adjacency matrix:
0, 0, 0, 0, 1, 1, 1, 1, 1, 0,
0, 0, 1, 1, 1, 0, 1, 1, 0, 1,
0, 1, 0, 1, 0, 1, 0, 0, 0, 0,
0, 1, 1, 0, 0, 1, 0, 1, 1, 0,
1, 1, 0, 0, 0, 0, 0, 1, 1, 1,
1, 0, 1, 1, 0, 0, 1, 1, 0, 1,
1, 1, 0, 0, 0, 1, 0, 1, 0, 1,
1, 1, 0, 1, 1, 1, 1, 0, 1, 0,
1, 0, 0, 1, 1, 0, 0, 1, 0, 1,
0, 1, 0, 0, 1, 1, 1, 0, 1, 0,


DFS traversal result: 0 4 1 2 3 5 6 7 8 9
```

3.3 Розв’язання задачі вручну

Adjacency Matrix

1 0, 1, 1, 1, 1
2 1, 0, 0 1 0
3 0, 0, 0 1 1
4 0, 1, 1 0 1
5 0, 0 1 1 0

Number of vertices : 5



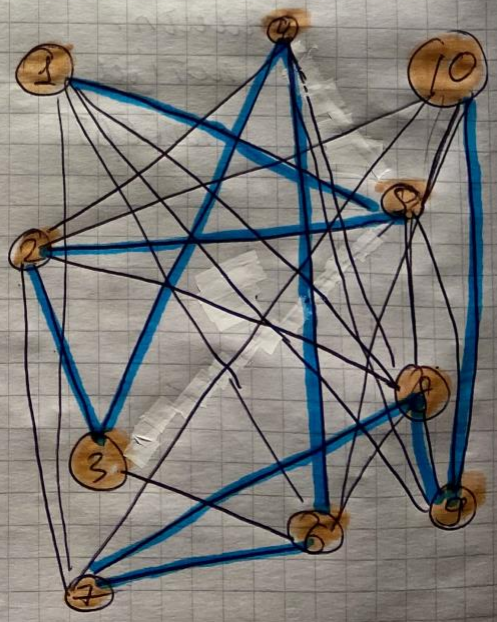
Traversal result:
: 1 → 2 → 4 → 3 → 5

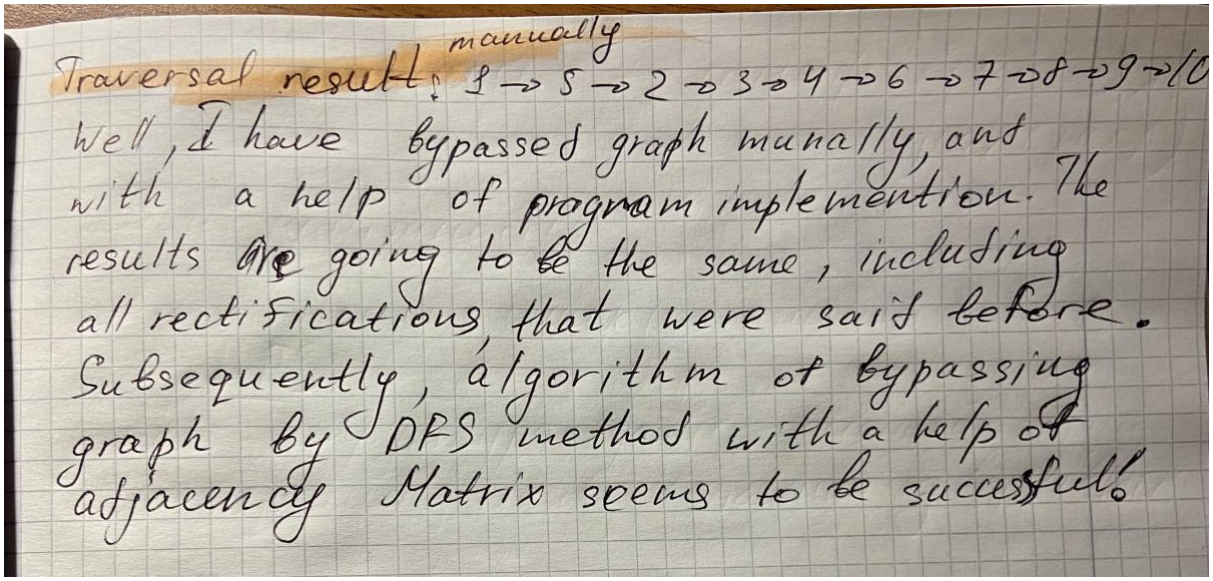
The program output seems to be → 0 → 1 → 3 → 2 → 5, that is cause my graph implemention here starts from the first point, but not from the zero one. So, I'd like to make a consequence, that my algorithm of DfS works correctly and I have equal replies to my app.

Adjacency Matrix

0 0 0 0 1 1 1 1 0
0 0 1 1 1 0 1 1 0 1
0 1 0 1 1 0 1 0 0 0 0
0 1 1 0 0 1 0 1 1 0
1 0 0 0 0 0 1 1 1
1 0 1 1 0 0 1 1 0 1
1 1 0 0 0 1 0 1 0 1
1 1 0 1 1 1 1 0 1 0
1 0 0 1 1 0 0 1 0 1
0 1 0 0 1 1 1 0 1 0

Numb of vertices : 10





Висновок

Зробивши обхід графу DFS за допомогою матриці суміжності і отримавши однакові результати вручну та програмною реалізацією, можна зробити висновок про правильність рішення. Це свідчить про те, що алгоритм був виконаний коректно та відповідно до правил обходу графу DFS. Отже, використання матриці суміжності є ефективним методом для реалізації обходу графу DFS, а виконання процесу вручну дозволяє краще зрозуміти принципи роботи алгоритму.