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

Кафедра ІСТ

Звіт

з лабораторної роботи № 4 з дисципліни
«Теорія алгоритмів»

«Прикладні задачі теорії графів ч.1»

Виконали ІА-31 Клим'юк В.Л, Самелюк А.С, Дук М.Д, Сакун Д.С

Перевірив Степанов А.С.

Київ 2024

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ 3

2 ЗАВДАННЯ 4

3 ВИКОНАННЯ 8

3.1 Псевдокод алгоритму 8

3.2 Програмна реалізація алгоритму 8

3.2.1 Вихідний код 8

ВИСНОВОК 10

КРИТЕРІЇ ОЦІНЮВАННЯ 11

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

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

2 ЗАВДАННЯ

Згідно варіанту (таблиця 2.1), розробити та записати алгоритм задачі на графах за допомогою псевдокоду (чи іншого способу за вибором).

Виконати програмну реалізацію алгоритму на будь-якій мові програмування для довільного графа, передбачити введення розмірності графа та введення даних графа вручну чи випадковим чином.

Для самостійно обраного графа (розмірності не менше 9 вершин) розв’язати задану за варіантом задачу вручну.

Зробити узагальнений висновок з лабораторної роботи, у якому порівняти програмне та ручне розв’язання задачі.

Таблиця 2.1 – Варіанти алгоритмів

№	**Задача**	**Алгоритм**	**Тип графу**	**Спосіб задання графу**
1	Обхід графу	DFS	Неорієнтований	Матриця суміжності
2	Обхід графу	BFS	Неорієнтований	Матриця суміжності
3	Пошук маршруту у графі	Террі	Неорієнтований	Матриця суміжності
4	Пошук відстані між вершинами графа	Хвильовий	Неорієнтований	Матриця суміжності
5	Пошук найкоротшого шляху між парою вершин	Дейкстри	Орієнтований	Матриця вагів
6	Пошук найкоротшого	Беллмана-Форда	Орієнтований	Матриця вагів

	шляху між парою вершин			
7	Побудова мінімальних покриваючих дерев	Прима	Неорієнтований	Матриця вагів
8	Побудова мінімальних покриваючих дерев	Крускала	Неорієнтований	Матриця вагів
9	Побудова мінімальних покриваючих дерев	Борувки	Неорієнтований	Матриця вагів
10	Побудова Ейлерового циклу	За означенням	Неорієнтований	Матриця суміжності
11	Побудова Ейлерового циклу	Флері	Неорієнтований	Матриця суміжності
12	Побудова Гамільтонового циклу	Пошук із поверненнями	Неорієнтований	Матриця суміжності
13	Обхід графу	DFS	Неорієнтований	Матриця інцидентності
14	Обхід графу	BFS	Неорієнтований	Матриця інцидентності
15	Пошук маршруту у графі	Террі	Неорієнтований	Матриця інцидентності
16	Пошук відстані між вершинами графа	Хвильовий	Неорієнтований	Матриця інцидентності
17	Пошук найкоротшого	Дейкстри	Орієнтований	Матриця вагів

	шляху між парою вершин			
18	Пошук найкоротшого шляху між парою вершин	Беллмана-Форда	Орієнтований	Матриця вагів
19	Побудова мінімальних покриваючих дерев	Прима	Неорієнтований	Матриця вагів
20	Побудова мінімальних покриваючих дерев	Крускала	Неорієнтований	Матриця вагів
21	Побудова мінімальних покриваючих дерев	Борувки	Неорієнтований	Матриця вагів
22	Побудова Ейлерового циклу	За означенням	Неорієнтований	Матриця інцидентності
23	Побудова Ейлерового циклу	Флері	Неорієнтований	Матриця інцидентності
24	Побудова Гамільтонового циклу	Пошук із поверненнями	Неорієнтований	Матриця інцидентності
25	Обхід графу	DFS	Неорієнтований	Матриця суміжності
26	Обхід графу	BFS	Неорієнтований	Матриця суміжності
27	Пошук маршруту у графі	Террі	Неорієнтований	Матриця суміжності

28	Пошук відстані між вершинами графа	Хвильовий	Неорієнтований	Матриця суміжності
29	Пошук найкоротшого шляху між парою вершин	Дейкстри	Орієнтований	Матриця вагів
30	Пошук найкоротшого шляху між парою вершин	Беллмана-Форда	Орієнтований	Матриця вагів
31	Побудова мінімальних покриваючих дерев	Прима	Неорієнтований	Матриця вагів
32	Побудова мінімальних покриваючих дерев	Крускала	Неорієнтований	Матриця вагів
33	Побудова мінімальних покриваючих дерев	Борувки	Неорієнтований	Матриця вагів
34	Побудова Ейлерового циклу	За означенням	Неорієнтований	Матриця суміжності
35	Побудова Ейлерового циклу	Флері	Неорієнтований	Матриця суміжності
36	Побудова Гамільтонового циклу	Пошук із поверненнями	Неорієнтований	Матриця суміжності

3 ВИКОНАННЯ

3.1 Псевдокод алгоритму

```
DFS(G, u)
    u.visited = true
    for each v ∈ G.Adj[u]
        if v.visited == false
            DFS(G,v)

init() {
    For each u ∈ G
        u.visited = false
     For each u ∈ G
       DFS(G, u)
}
```

3.2 Програмна реалізація алгоритму

```
namespace Classes
{
    public class Algorithm
    {
        public List<int> DeepFirstSearch(int[][] matrix)
        {
            if (!CheckMatrix(matrix))
                throw new ArgumentException("Matrix must be symmetrical");

            List<int> result = new List<int>();

            HashSet<int> visited = new HashSet<int>();

            Search(visited, result, matrix, 0);

            return result;
        }

        private void Search(HashSet<int> memo, List<int> res, int[][] matrix, int
current)
        {
            res.Add(current);
            memo.Add(current);
            for (int i = 0; i < matrix[current].Length; i++)
            {
                if (matrix[current][i] == 1 && !memo.Contains(i))
                {
                    Search(memo, res, matrix, i);
                }
            }
        }

        private bool CheckMatrix(int[][] matrix)
```

```
        {
            for(int i = 1; i < matrix.Length; i++)
            {
                for(int j = 0; j < i; j++)
                {
                    if (matrix[i][j] != matrix[j][i])
                        return false;
                }
            }
            return true;
        }
    }
}
```

3.2.1 Вихідний код

```
using Classes;

namespace Lab4
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Algorithm alg = new Algorithm();

            //int[][] matrix = new int[][]
            //{
            //    new int[]{0,0,0,1,0,0,0,0,1},
            //    new int[]{0,0,1,1,0,1,0,0,1},
            //    new int[]{0,1,0,0,1,0,1,0,0},
            //    new int[]{1,1,0,0,1,0,0,0,1},
            //    new int[]{0,0,1,1,0,0,0,1,0},
            //    new int[]{0,1,0,0,0,0,1,0,0},
            //    new int[]{0,0,1,0,0,1,0,0,0},
            //    new int[]{0,0,0,0,1,0,0,0,0},
            //    new int[]{1,1,0,1,0,0,0,0,0}
            //};

            //int[][] matrix = new int[][]
            //{
            //    new int[]{0,1,1,0,0,0,0},
            //    new int[]{1,0,0,1,1,0,0},
            //    new int[]{1,0,0,0,0,1,0},
            //    new int[]{0,1,0,0,0,0,1},
            //    new int[]{0,1,0,0,0,0,0},
            //    new int[]{0,0,1,0,0,0,0},
            //    new int[]{0,0,0,1,0,0,0},
            //};
```

```
            int[][] matrix = new int[][]
            {
                new int[]{0,0,0,0,0,0,0,0,1,0,0,0,0,0,0},
                new int[]{0,0,0,1,0,0,0,0,0,0,1,0,1,0,0},
                new int[]{0,0,0,0,0,1,0,0,0,0,0,0,0,0,0},
                new int[]{0,1,0,0,0,1,0,0,0,0,0,0,0,0,0},
                new int[]{0,0,0,0,0,0,1,0,0,0,0,0,0,0,0},
                new int[]{0,0,1,1,0,0,0,1,0,1,0,0,0,0,0},
                new int[]{0,0,0,0,1,0,0,0,1,0,0,0,0,0,0},
                new int[]{0,0,0,0,0,1,0,0,0,0,0,0,0,0,0},
                new int[]{1,0,0,0,0,0,1,0,0,0,0,0,1,0,0},
                new int[]{0,0,0,0,0,1,0,0,0,0,0,0,0,0,0},
                new int[]{0,1,0,0,0,0,0,0,0,0,0,0,0,1,0},
                new int[]{0,0,0,0,0,0,0,0,0,0,0,0,0,1,0},
                new int[]{0,1,0,0,0,0,0,0,1,0,0,0,0,0,0},
                new int[]{0,0,0,0,0,0,0,0,0,0,1,1,0,0,1},
                new int[]{0,0,0,0,0,0,0,0,0,0,0,0,0,1,0}
            };

            List<int> res = alg.DeepFirstSearch(matrix);

            foreach(int r in res)
            {
                Console.WriteLine(r);
            }
        }
    }
}
```

3.2.2 Приклад роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми для графів на 7 і 15 вершин відповідно.

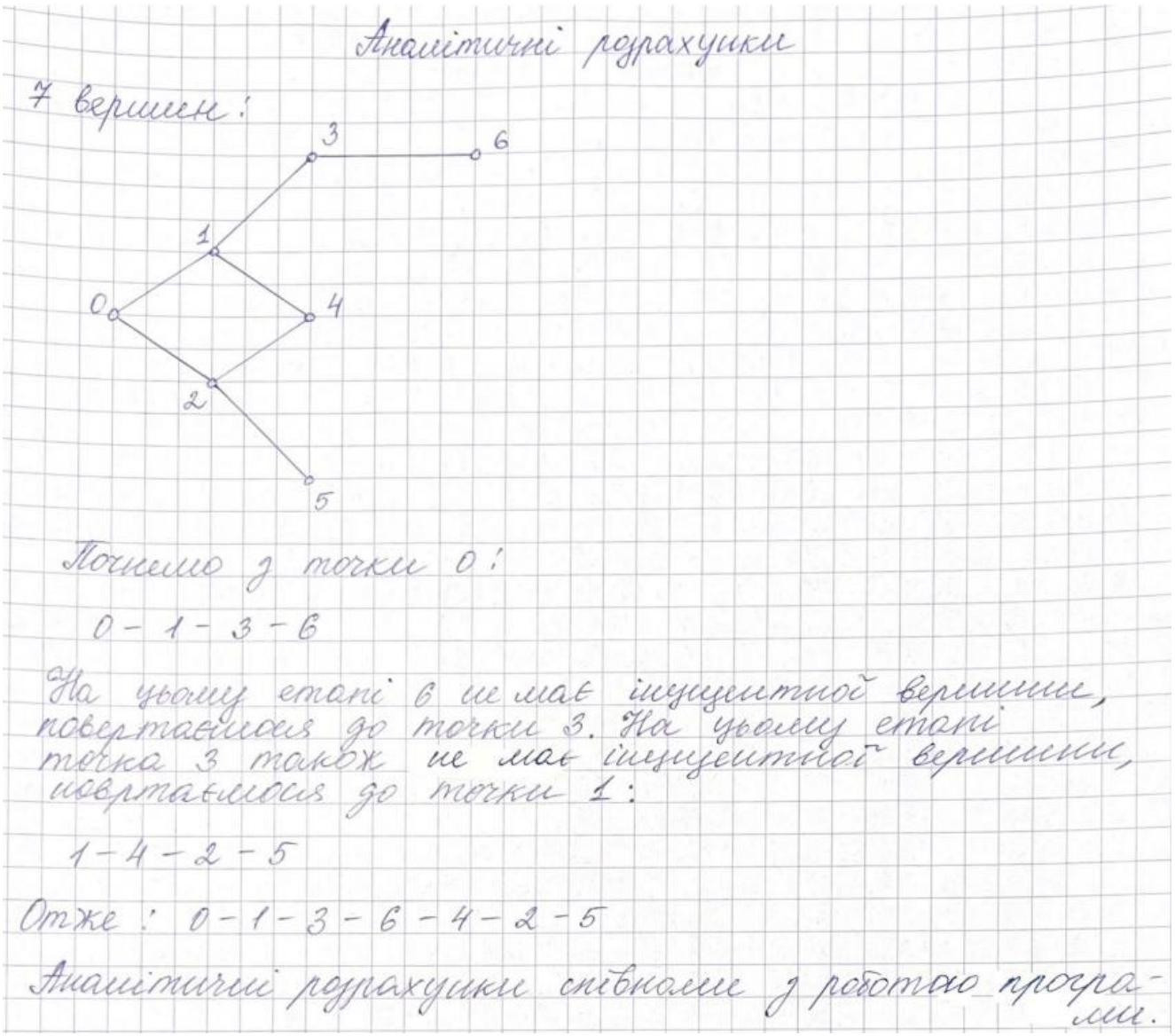
```
0 1 3 6 4 2 5
```

Рисунок 3.1 – Задача на 7 вершин

```
0 8 6 4 12 1 3 5 2 7 9 10 13 11 14
```

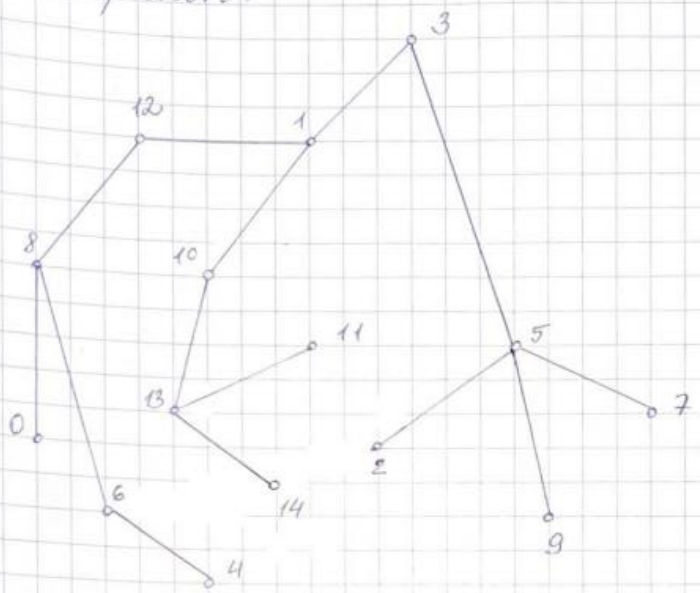
Рисунок 3.2 – Задача на 15 вершин

3.3 Розв’язання задачі вручну



На рисунку 3.3 наведено розв’язання задачі на 7 вершин вручну.

15 вершин:



Почнемо з точки 0:

0 - 8 - 6 - 4

На цьому етапі 4 не має інцидентної вершини, повертаємося в 6; на цьому етапі 6 не має інцидентної вершини, повертаємося в 8:

8 - 12 - 1 - 3 - 5 - 2

На цьому етапі 2 не має інцидентної вершини, повертаємося в 5:

5 - 7

На цьому етапі 7 не має інцидентної вершини і повертаємося в 5:

5 - 9

На цьому етапі 9 не має інцидентної вершини, повертаємося в 5; на цьому етапі 5 не має інцидентної вершини, повертаємося в 3; на цьому етапі 3 не має інцидентної вершини, повертаємося в 1:

1 - 10 - 13 - 11

На цьому етапі 11 не має інцидентної верш., повертаємося в 13:

13 - 14

Отже: 0 - 8 - 6 - 4 - 12 - 1 - 3 - 5 - 2 - 7 - 9 - 10 - 13 - 11 - 14

Рисунок 3.4 – Розв’язання задачі на 15 вершин вручну

ВИСНОВОК

У цій роботі ми вивчали прикладні задачі теорії графів та алгоритм пошуку вглиб (DFS). DFS виявився корисним для розв'язання різноманітних задач, таких як пошук шляхів, виявлення циклів, знаходження компонентів зв'язності тощо. Цей алгоритм дозволяє систематично досліджувати граф та знаходити оптимальні рішення для низки завдань. В результаті досліджень ми зрозуміли, що DFS є важливим інструментом в області теорії графів та має широкий спектр застосувань у різних областях, включаючи комп'ютерні науки, транспортні системи, біоінформатику та інші.