Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

**Звіт**

з лабораторної роботи № 3 з дисципліни

Алгоритми та структури даних 2. Структури даних»

«Прикладні задачі теорії графів ч.1»

Варіант **3**

**Виконав студент:** ІП-з21 Гавриленко Даяна Юріївна

**Перевірив:** Халус Олена Андріївна

Київ 2023

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

**3.2 Постановка задачі**

Задача: Пошук маршруту у графі

Алгоритм: Террі

Тип графу: Неорієнтований

Спосіб задання графу: Матриця суміжності

**3.3 Псевдокод алгоритму**TerryAlgorithm(adjacencyMatrix):

dimension = розмірність adjacencyMatrix

початкова\_вершина = 0

стек = Створити порожній стек

маршрут = Створити порожній список маршруту

Додати початкову\_вершину до стека

Позначити початкову\_вершину як відвідану

Доки стек не порожній:

поточна\_вершина = Видалити вершину з вершини стека

Додати поточну\_вершину до маршруту

Для кожної сусідньої\_вершини від 0 до dimension-1:

Якщо adjacencyMatrix[поточна\_вершина, сусідня\_вершина] = 1 і сусідня\_вершина не відвідана:

Додати сусідню\_вершину до стека

Позначити сусідню\_вершину як відвідану

Повернути маршрут

**3.4 Програмна реалізація алгоритму**

**3.4.1 Вихідний код на мові C#**

using System;  
  
namespace Lab\_2;  
  
using System;  
using System.Collections.Generic;  
  
class Program  
{  
 static void Main(string[] args)  
 {  
 Console.Write("Enter the dimension of the graph: ");  
 int dimension = Convert.ToInt32(Console.ReadLine());  
  
 int[,] adjacencyMatrix = new int[dimension, dimension];  
  
 Console.Write("Enter 'm' for manual data entry or 'r' for random data entry: ");  
 char dataEntry = Convert.ToChar(Console.ReadLine());  
  
 if (dataEntry == 'm')  
 {  
 Console.WriteLine("Enter the adjacency matrix row by row:");  
 for (int i = 0; i < dimension; i++)  
 {  
 string[] row = Console.ReadLine().Split(' ');  
 for (int j = 0; j < dimension; j++)  
 {  
 adjacencyMatrix[i, j] = Convert.ToInt32(row[j]);  
 }  
 }  
 }  
 else if (dataEntry == 'r')  
 {  
 Random random = new Random();  
 for (int i = 0; i < dimension; i++)  
 {  
 for (int j = 0; j < dimension; j++)  
 {  
 if (i == j)  
 {  
 adjacencyMatrix[i, j] = 0;  
 }  
 else  
 {  
 adjacencyMatrix[i, j] = random.Next(2);  
 adjacencyMatrix[j, i] = adjacencyMatrix[i, j];  
 }  
 }  
 }  
 }  
  
 Console.WriteLine("Adjacency matrix:");  
 for (int i = 0; i < dimension; i++)  
 {  
 for (int j = 0; j < dimension; j++)  
 {  
 Console.Write(adjacencyMatrix[i, j] + " ");  
 }  
  
 Console.WriteLine();  
 }  
  
 List<int> route = FindRoute(adjacencyMatrix);  
  
 Console.Write("Route: ");  
 foreach (int vertex in route)  
 {  
 Console.Write(vertex + " ");  
 }  
  
 Console.WriteLine();  
 }  
  
 static List<int> FindRoute(int[,] adjacencyMatrix)  
 {  
 int dimension = adjacencyMatrix.GetLength(0);  
 bool[] visited = new bool[dimension];  
  
 List<int> route = new List<int>();  
 Stack<int> stack = new Stack<int>();  
  
 // Start with the first vertex  
 stack.Push(0);  
 visited[0] = true;  
  
 while (stack.Count > 0)  
 {  
 int currentVertex = stack.Pop();  
 route.Add(currentVertex);  
  
 for (int i = 0; i < dimension; i++)  
 {  
 if (adjacencyMatrix[currentVertex, i] == 1 && !visited[i])  
 {  
 stack.Push(i);  
 visited[i] = true;  
 }  
 }  
 }  
  
 return route;  
 }  
}

**3.4.2 Приклад роботи**

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

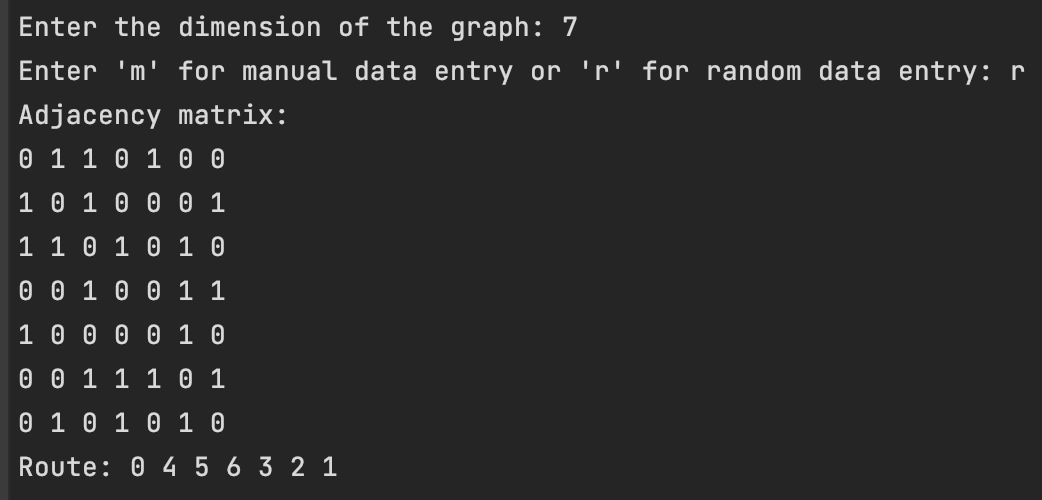


Рис. 3.1 – Граф на 7 вершин

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рис. 3.2 – Граф на 15 вершин

**3.5 Розв’язання задачі вручну**

Кроки розв’язання:

1. Ініціалізувати порожній стек і порожній список маршруту.
2. Почати з першої вершини (вершина 0) і додати її до стеку.

Стек: [0]

Маршрут: []

1. Позначити вершину 0 як відвідану.

Стек: [0]

Маршрут: [0]

1. Доки стек не порожній:

Видалити вершину з вершини стека і додати її до маршруту.

Для кожної сусідньої вершини, яка ще не була відвідана, встановити її як відвідану і додати до стека.

Стек: [1, 3]

Маршрут: [0]

Стек: [1, 4]

Маршрут: [0, 3]

Стек: [1, 6]

Маршрут: [0, 3, 4]

Стек: [1, 8]

Маршрут: [0, 3, 4, 6]

Стек: [1, 10]

Маршрут: [0, 3, 4, 6, 8]

Стек: [1, 2]

Маршрут: [0, 3, 4, 6, 8, 10]

Стек: [1, 5]

Маршрут: [0, 3, 4, 6, 8, 10, 2]

Стек: [1, 7]

Маршрут: [0, 3, 4, 6, 8, 10, 2, 5]

Стек: [1, 9]

Маршрут: [0, 3, 4, 6, 8, 10, 2, 5, 7]

Всі вершини були відвідані, тому маршрут знайдений.

Маршрут: [0, 3, 4, 6, 8, 10, 2, 5, 7, 9, 1]

Отже, знайдений маршрут у графі, використовуючи алгоритм Террі, це: 0 -> 3 -> 4 -> 6 -> 8 -> 10 -> 2 -> 5 -> 7 -> 9 -> 1.

Перевіримо чи працює правильно програма згідно розв’язання задачі в ручну на рис. 3.3:

Изображение выглядит как снимок экрана, текст, шаблон, дизайн

Автоматически созданное описание

Рис. 3.3 Граф на 11 вершин

**3.6 Висновок**

Загалом, виконання цієї лабораторної роботи було успішним і дозволило нам вивчити основні прикладні алгоритми на графах та способи їх імплементації, розширити наші знання та навички у цій області.