|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **НУЛП, ІКНІ, САПР** | | Тема | оцінка | підпис |
| КН-414 | 2 | Алгоритм рішення задачі листоноші |  |  |
| Коцюба В.С. | |
| № залікової: 16081031 | |
| **Дискретні моделі в САПР** | | Викладач: | |
| к.т.н., асистент  Кривий Р.З. | |

**Мета:**

Метою даної лабораторної роботи є вивчення і дослідження алгоритмів рішення задачі листоноші.

**Завдання:**

Написати програму для демонстрації роботи алгоритму задачі листоноші.

**Теоретичні відомості:**

Задача листоноші. Основні поняття. Властивості.

Будь-який листоноша перед тим, як відправитись в дорогу повинен підібрати на пошті листи, що відносяться до його дільниці, потім він повинен рознести їх адресатам, що розмістились вздовж маршрута його проходження, і повернутись на пошту. Кожен листоноша, бажаючи втратити якомога менше сил, хотів би подолати свій маршрут найкоротшим шляхом. Загалом, задача листоноші полягає в тому, щоб пройти всі вулиці маршрута і повернутися в його початкову точку, мінімізуючи при цьому довжину пройденого шляху.

Перша публікація, присвячена рішенню подібної задачі, появилась в одному з китайських журналів, де вона й була названа задачею листоноші. Очевидно, що така задача стоїть не тільки перед листоношею. Наприклад, міліціонер хотів би знати найбільш ефективний шлях патрулювання вулиць свого району, ремонтна бригада зацікавлена у виборі найкоротшого шляху переміщення по всіх дорогах.

Задача листоноші може бути сформульована в термінах теорії графів. Для цього побудуємо граф G = (X , E), в якому кожна дуга відповідає вулиці в маршруті руху листоноші, а кожна вершина - стик двох вулиць. Ця задача являє собою задачу пошуку найкоротшого маршруту, який включає кожне ребро хоча б один раз і закінчується у початковій вершині руху.

Нехай S-початкова вершина маршруту і a(i,j)>0 - довжина ребра (і, j) . В графі на рис. 1 існує декілька шляхів, по яким листоноша може обійти всі ребра і повернутись у вершину S.

Наприклад :

Шлях 1: (S,a), (a,b), (b,c), (c,d), (d,b), (b,S)

Шлях 2: (S,a), (a,b), (b,d), (d,c), (c,b), (b,S)

Шлях 3: (S,b), (b,c), (c,d), (d,b), (d,a), (a,S)

Шлях 4: (S,b), (b,d), (d,c), (c,b), (b,a), (a,S)

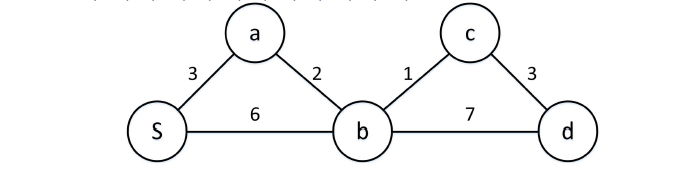


Рис. 1.

В будь-який з чотирьох шляхів кожне ребро входить тільки один раз.

Таким чином, загальна довжина кожного маршруту дорівнює 3+2+1+3+7+6=22.

Кращих маршрутів у листоноші не існує.

Ейлеровий цикл

Ейлеревим циклом в графі називається шлях, який починається і закінчується в тій самій вершині, при чому всі ребра графа проходяться тільки один раз.

Ейлеревим шляхом називається шлях, який починається в вершині А, а закінчується в вершині Б, і всі ребра проходяться лише по одному разу.

Граф, який включає в себе ейлерів цикл називається ейлеревим.

**Індивідуальне завдання**

*Варіант 1. Реалізувати програму для вирішення задачі листоноші.*

**Робота з програмою:**

Після запуску програми в лівому краю вікна, у верхньому текст боксі задана початкова матриця суміжності графу, за потреби її там можна міняти.

Для запуску алгоритму натискаємо кнопку ‘Старт’, у нижньому текст боксі бачимо результати роботи алгоритму у центрі в полі канвас- відображення графа.

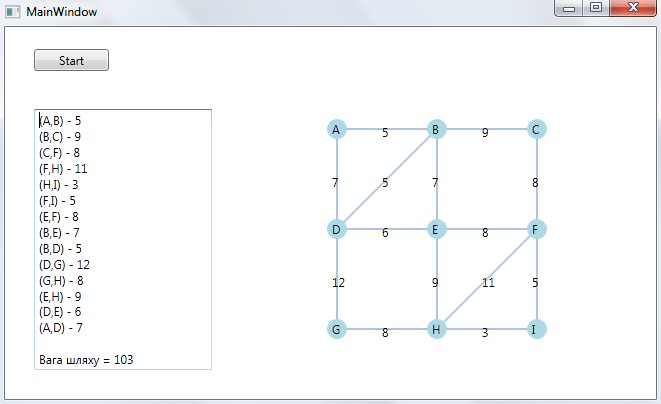


Рис.2 Вікно роботи програми

Фрагмент програми:

//зчитати ребра з файлу

private void ReadEdges(string path, List<Edge> input, List<int> input2)

{

String[] str = File.ReadAllLines(path);

int[,] matrix = new int[str.Length, 3];

for (int i = 0; i < str.Length; i++)

{

int[] arr = str[i].Split(new char[] { ';' }, StringSplitOptions.RemoveEmptyEntries).Select(s => int.Parse(s)).ToArray();

for (int j = 0; j < arr.Length; j++)

{

matrix[i, j] = arr[j];

}

}

for (int i = 0; i < matrix.GetLength(0); i++)

{

input.Add(new Edge(matrix[i, 0], matrix[i, 1], matrix[i, 2]));

input2.Add(matrix[i, 0]);

input2.Add(matrix[i, 1]);

}

}

//Список вершин

private List<Vertex> GetUniquesVer(List<int> input)

{

Dictionary<int, bool> found = new Dictionary<int, bool>();

List<Vertex> uniques = new List<Vertex>();

foreach (int value in input)

{

if (!found.ContainsKey(value))

{

found[value] = true;

char ch = Convert.ToChar(value + 64);

uniques.Add(new Vertex(ch.ToString(), value));

}

}

return uniques;

}

//Належність до підграфа

private void MarkEdge(Edge ed, List<Edge> Elist, int num)

{

for (int i = 0; i < Elist.Count; i++)

{

if (Elist[i].VERTEX1 == ed.VERTEX1 && Elist[i].VERTEX2 == ed.VERTEX2)

{

ed.COMP = num;

break;

}

}

}

//Дістаю перелік унік. ел.

private List<int> GetUnique(List<int> input)

{

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

Dictionary<int, bool> dict = new Dictionary<int, bool>();

for (int i = 0; i < input.Count; i++)

{

if (!dict.ContainsKey(input[i]))

{

dict[input[i]] = true;

output.Add(input[i]);

}

}

return output;

}

private Point[] ReadCoords(string path)

{

String[] str = File.ReadAllLines(path);

Point[] points = new Point[str.Length];

for (int i = 0; i < str.Length; i++)

{

int[] arr = str[i].Split(new char[] { ';' }, StringSplitOptions.RemoveEmptyEntries).Select(s => int.Parse(s)).ToArray();

points[i].X = arr[0];

points[i].Y = arr[1];

}

return points;

}

private List<int> CountUniques<T>(List<T> list)

{

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

Dictionary<T, int> counts = new Dictionary<T, int>();

List<T> uniques = new List<T>();

foreach (T val in list)

{

if (counts.ContainsKey(val))

counts[val]++;

else

{

counts[val] = 1;

uniques.Add(val);

}

}

foreach (T val in uniques)

{

result.Add(counts[val]);

}

return result;

}

//Словник унікальних

private Dictionary<T, int> UniquesDict<T>(List<T> list)

{

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

Dictionary<T, int> counts = new Dictionary<T, int>();

List<T> uniques = new List<T>();

foreach (T val in list)

{

if (counts.ContainsKey(val))

counts[val]++;

else

{

counts[val] = 1;

uniques.Add(val);

}

}

return counts;

}

//Визначення наявності Ейлерового циклу в графі

private bool IsEilerCycle(List<Vertex> iVer, List<Edge> iEdg)

{

bool cycle = true;

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

for (int i = 0; i < iEdg.Count; i++)

{

list.Add(iEdg[i].VERTEX1);

list.Add(iEdg[i].VERTEX2);

}

list = CountUniques<int>(list);

foreach (int item in list)

{

if (item % 2 != 0)

{

cycle = false;

break;

}

}

return cycle;

}

Висновок: На цій лабораторній роботі було здійснено ознайомлення з алгоритмом рішення задачі листоноші.