Національний технічний університет України

«Київський політехнічний інститут»

Кафедра автоматизованих систем обробки інформації та управління

Звіт про виконання лабораторної роботи №10 з дисципліни

«Алгоритми та структури даних»

Перевірено: Ковалюк Т.В. Виконав ст. 1 курсу ФІОТ

Проскура С.Л. гр. ІС-52

Дорошенко А.В.

Київ 2016

## Лабораторна робота 10

# **Алгоритми на графах. Мінімальні кістяки**

Варіант № 9

*Мета:*

1. Розглянути алгоритми знаходження мінімільних кістякових дерев

**2.** Навчитися застосовувати алгоритми в практичних задачах.

Завдання

|  |  |
| --- | --- |
| У місті знаходяться мережу магазинів, які необхідно з'єднати деякою системою каналів зв'язку так, щоб будь-які два магазини були пов'язані або безпосередньо з'єднуючим їх каналом, або через інші магазини і канали, і щоб загальна довжина і вартість каналів зв'язку була мінімальною.  **Вхідні дані.** Перший рядок файлу містить два числа - кількості магазинів n і каналів зв'язку m, де 2 < n < 100, n - 1 < m < 200. Наступні т рядків містять дані про відстань між магазинами - трійки чисел виду a, b, d, де a, b - номери магазинів від 0 до n - 1, d - відстань між ними.  **Вихідні дані.** Вихідний файл повинен містити список каналів зв'язку (пар номерів магазинів), загальну довжину каналів і мінімальну вартість каналів. |  |

### Блок-схема алгоритму





### Код програми

//Laboratory work

//written by student of the first curse

//of the group IC-52

//Anton Doroshenko

//2016.05.04

//==========================================================================

#include <stdio.h>

#include <iostream>

#include <math.h>

#include <iostream>

#include <iomanip>

using namespace std;

int numberOfVertexs, //кількість магазинів

numberOfRibs, //кількість ребер

listOfRibs[6000][3], //масив ребер з їх довжинами

numbersOfRibsForMinDistWay[300], //номери ребер, що входять до мінімального шляху

arrForConnectednessOfGraph[100], //масив для превірки зв'язності

minDist, //мінімальна довжина прокладки лінії зв'язку

amountOfRibsInMinDistWay; //кількість ребер в мінмальному шляху прокладки

bool resOfCheck[6000];//масив для запам. зв'язний граф чи ні в можливих других по мінімальності шляхів

//============ перевірка графа на зв'зність =============

int Check()

{

bool f=true;

for (int i = 1; i < numberOfVertexs; i++)

{

if (arrForConnectednessOfGraph[0] != arrForConnectednessOfGraph[i])

{

f = false;

break;

}

}

return f;

}

//============== сортування рядків масиву по неспаданню елементів в третьому стовпці ====================

void Sort(int a[][3], int left, int right)

{

int l = left;

int r = right;

int tmp = 0;

int mid = a[r][2];

while (l <= r)

{

while ((a[l][2] < mid) && (l <= right))

{

l++;

}

while ((a[r][2] > mid) && (r >= left))

{

r--;

}

if (l <= r)

{

for (int i = 0; i < 3; i++)

{

tmp = a[l][i];

a[l][i] = a[r][i];

a[r][i] = tmp;

}

l++;

r--;

}

}

if (r > left)

{

Sort(a, left, r);

}

if (l < right)

{

Sort(a, l, right);

}

}

//======== задання початкових значень в масиві для перевірки зв'язності ========

void ResetConection()

{

for (int i = 0; i <= numberOfVertexs; i++)

{

arrForConnectednessOfGraph[i] = i;

}

}

//=============== знаходження мінімальної відстані ==================

void MinimalDistance()

{

int u = 0;

for (int i = 0; i < numberOfRibs; i++)

{

if (arrForConnectednessOfGraph[listOfRibs[i][0]] != arrForConnectednessOfGraph[listOfRibs[i][1]])

{

minDist += listOfRibs[i][2];

numbersOfRibsForMinDistWay[u] = i;

u++;

int col = arrForConnectednessOfGraph[listOfRibs[i][0]];

for (int j = 0; j < numberOfVertexs; j++)

{

if (arrForConnectednessOfGraph[j] == col)

{

arrForConnectednessOfGraph[j] = arrForConnectednessOfGraph[listOfRibs[i][1]];;

}

}

}

}

amountOfRibsInMinDistWay = u;

}

//================================ введення данних =====================================

void Input()

{

cin >> numberOfVertexs >> numberOfRibs;

for (int i = 0; i < numberOfRibs; i++)

{

for (int j = 0; j < 3; j++)

{

cin >> listOfRibs[i][j];

}

}

}

void Output()

{

if (Check())

{

cout << "Dovguna minimal'noyi zv'aznoyi meregi: ";

cout << minDist << endl;

cout << "Kil'kist' reber y minimal'niy meregi: ";

cout << amountOfRibsInMinDistWay << endl;

cout << "Paru magazuniv mig yakumu potribno proklastu meregy" << endl;

for (int i = 0; i < amountOfRibsInMinDistWay; i++)

{

for (int j = 0; j < 3; j++)

{

cout << setw(3) << listOfRibs[numbersOfRibsForMinDistWay[i]][j];

}

cout << endl;

}

}

else

cout << "Nemogluvo proklastu meregy, schob ysi magazunu bylu zvyazani" << endl;

}

//==================== головна функція =====================

int main()

{

Input();

Sort(listOfRibs, 0, numberOfRibs - 1);

ResetConection();

MinimalDistance();

Output();

system("pause");

}

1. **Screen Shot результатів**

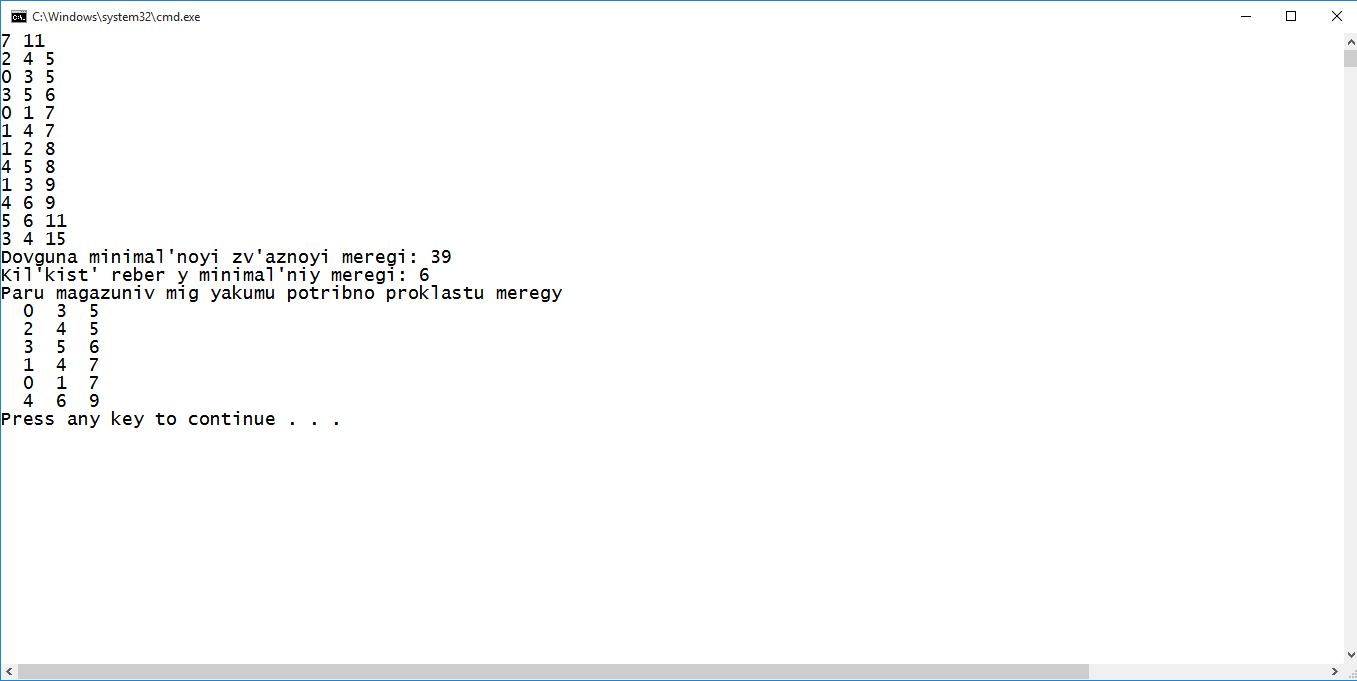
****

Рис.9 (Screen Shot результатів 1)

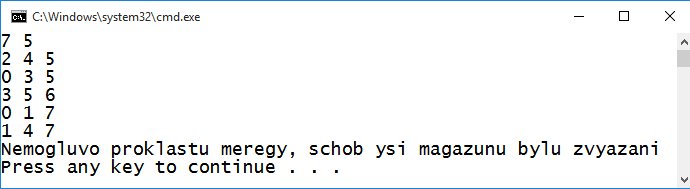


Рис.10 (Screen Shot результатів 2)

1. **Аналіз результатів**

Як ми бачимо зі ScreenShot’ів

Програма в першому випадку вивела мінімальне кістякове дерево, в другому повідомлення, що він не зв'язний.

1. **Висновок**

Ціллю цієї лабораторної роботи було знайти мінімальне кістякове, тобто було достатньо реалізувати алгоритм Пріма чи Крускала. Я використав алгоритм Пріма, так як не потрібно робити додаткових дій до сортування ребер.