Міністерство освіти і науки України  
Черкаський державний технологічний університет

Кафедра програмного забезпечення автоматизованих систем

**Звіт**

Про виконання лабораторної роботи №3

з дисципліни «Комп’ютерна дискретна математика»

|  |  |
| --- | --- |
| Перевірив:  ст. викладач  кафедри ПМ  Мірошкіна І. В. | Виконав:  Студент 2-го курсу  Групи ПЗС-1944  Гогулов Я.В. |

Черкаси 2019

**Лабораторна робота №3**

**Тема роботи:** Побудова мінімального остовного дерева (МОД) Графа.

**Мета роботи:** Побудувати мінімальне остовне дерево використовуючи мінімальний алгоритм Крускала.

**Теоретичні відомості:**

*Алгоритм пошуку мінімального остовного дерева(МОД)*

Мінімальним остовним деревом (МОД) зв'язкового зваженого графа називається зв'язний підграф, що складається з усіх вершин вихідного дерева і деяких його ребер, причому сума ваг ребер мінімально можлива. Якщо вихідний граф несвязен, то описувану нижче процедуру можна застосовувати по черзі до кожної його компоненті зв'язності, отримуючи тим самим мінімальні головні дерева для цих компонент.

МОД – у зв’язному, алгоритм побудови мінімального кістякового дерева зваженого неорієнтовного графа. – це ост.дерево.

МОД для зв'язкового графа можна знайти за допомогою грубої сили. Оскільки безліч ребер МОД є підмножина безлічі ребер вихідного графа, можна просто перебрати всі можливі підмножини і знайти серед них МОД. Неважко бачити, що це надзвичайно трудомісткий процес. У безлічі з n ребер є 2N підмножин. Для кожного з цих підмножин ми повинні будемо перевірити, що воно задає зв'язний підграф, що охоплює всі вершини вихідного, і не містить циклів, а потім порахувати його вагу. Процес можна прискорити після того, як знайдено перше остовне дерево. Ніяке підмножина ребер, повна вага якого більше, ніж у вже знайденого найкращого остовного дерева, не може нам підійти, тому немає необхідності перевіряти зв'язність, ациклічність і охоплення всіх вершин. Однак навіть і при цьому поліпшенні складність методу грубої сили буде порядку O (2N).

*Алгоритм Крускала*

Алгоритм Дейкстри-Прима починає побудову МОД з конкретної вершини графа і поступово розширює побудовану частину дерева; на відміну від нього алгоритм Крускала робить упор на ребрах графа.

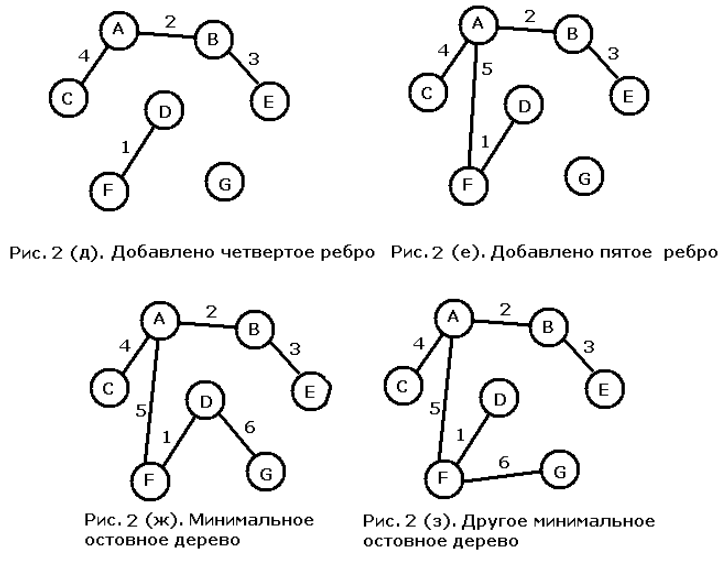
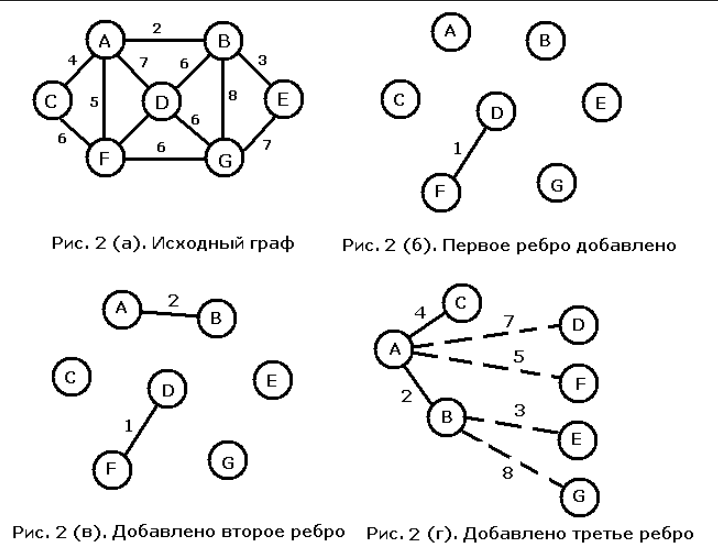
У цьому алгоритмі ми починаємо з порожнього дерева і додаємо до нього ребра в порядку зростання їх ваги поки не отримаємо набір ребер, що об'єднує всі вершини графа. Якщо ребра закінчаться до того, як всі вершини будуть з'єднані між собою, то це означає, що вихідний граф був несвязным, і отриманий нами результат являє собою об'єднання МОД всіх його компонент зв'язності.

Ми працюємо з тим самим графом (див. рис. 2 (а)), що і при описі алгоритму Дейкстри -Прима. Почнемо з ребра найменшої ваги, тобто з ребра DF. Початкова ситуація зображена на рис. 2 (б).

Наступним додається ребро ваги 2, що з'єднує вершини A і B (рис. 2 (в)), а потім - ребро ваги три, і ми отримуємо ситуацію з рис. 2 (г).

До результуючого графу послідовно додаються ребра ваг 4 і 5 (Малюнки 2 (д) і (е)). Несоединенной залишилася лише вершина G. Наступними ми повинні розглянути ребра ваги 6.

Два з чотирьох ребер ваги 6 слід відкинути, оскільки їх додавання призведе до появи циклу вже побудованої частини МОД. Приєднання ребра CF створило б цикл, що містить вершину A, а приєднання ребра BD - цикл, що містить вершини A і F. Обидві залишилися можливості підходять в рівній мірі, і, вибираючи одну з них, ми отримуємо або МОД з рис. 2 (ж), або МОД з рис. 2 (з).



**Завдання:** Wдерева = ∑ wy = 1+1+2+5 = 9

Реалізувати – це все:

* Граф зв’язний
* Числа від 1 до 10, Матриця 10×10, число вершин = 10
* Можуть повторюватись

Wy(Random)

W\*; = Wy – присвоюємо.

Wi,i = 0 – заповнюємо 0.

**Зміст звіту:**

1) Текст програми на С++.

2) Результати роботи програми.

3) Висновок до роботи.

**Лістинг програми**

#include <cstdio>

#include <iostream>

#include <cstdlib>

using namespace std;

typedef int\* tint; // указатель на int

int main ()

{ // int max=100; // максимальный вес ребра

int n; // количество вершин

tint\* G; // исходный граф

tint\* H; // матрица списка ребер с весом

tint\* K; /\*матрица, отмечающая принадлежность

вершины компоненте\*/

tint\* T; // матрица остовного дерева

tint\* L; // список ребер с ценами минимального дерева

int max;

cout<<" Maximalno dopustimoe zna4enie vesa rebra = ";

cin>> max;

cout<<"\n Vvedite 4ilo vershin: \n ";

cin>> n;

G=new tint [n];

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

G[i]=new int [n];

cout<<"Vvedite nignij treugolnik matrici stoimosti: \n ";

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

for(int j=0;j<i;j++) {

cin>> G [i] [j];

}

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

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

G[j][i]=G[i][j];

int kolreb=0;

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

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

if (G[i][j]<max&&G[i][j]!=0)

kolreb++;

H=new tint [kolreb];

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

H[i]=new int [3];

// -------------------------------------------

int a=0;

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

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

if(G[i][j]<max&&G[i][j]!=0) {

H [a] [0] =i+1;

H [a] [1] =j+1;

H [a] [2] =G [i] [j];

a++;

}

cout<<endl;

int f,d,s;

for (int i=0; i<kolreb-1; i++)

for (int j=0; j<kolreb-1; j++)

if (H [j] [2] <H [j+1] [2]) {

f=H [j] [2];

H [j] [2] =H [j+1] [2];

H [j+1] [2] =f;

d=H [j] [0];

H[j][0] =H [j+1] [0];

H[j+1][0] =d;

s=H[j][1];

H[j][1]=H[j+1][1];

H[j+1][1] =s;

}

// вывод ребер отсортированных по возрастанию веса

cout<<"Matrica dostigimosni otsortirovannoe po ubivaniuy: \n ";

for (int i=0; i<kolreb; i++) {

cout<<H[i][0] <<"-->";

cout<<H[i][1] <<" = ";

cout<<H[i][2] <<endl;

cout<<" ";

}

for (int i=0; i<kolreb; i++) {

H[i][0]--;

H[i][1]--;

}

// матрица дл¤ определени¤ компоненты

K=new tint [n];

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

K [i] =new int [2];

for (int i=0; i<n; i++) {

K [i] [0] =i;

K [i] [1] =0;

}

T=new tint [n];

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

T[i]=new int [n];

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

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

T[i][j]=0;

// -присоединение первого ребра

T[H[0][0]][H[0][1]]=H [0][2];

T[H[0][1]][H[0][0]]=H[0][2];

K[H[0][0]][1]=1;

K[H[0][1]][1]=1;

int m=2; // номер компоненты

for(int i=1;i<kolreb;i++) // пройти по всем ребрам

if(K[H[i][0]][1]!=K[H[i][1]][1])

// если 2 вершины не из одной компоненты

{

if (K [H [i] [0]] [1] >0 && K [H [i] [1]] [1] >0)

// если обе вершины принадлежат разной компоненте

{

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

if (K [H [i] [1]] [1] ==K [j] [1])

K[j][1]=K[H[i][0]][1];

K[H[i][1]][1]=K[H[i][0]][1];

T[H[i][0]][H[i][1]]=H[i][2];

T[H[i][1]][H[i][0]]=H[i][2];

}

if ((K[H[i][0]][1]>0 && K[H[i][1]][1]==0)

|| (K[H[i][0]][1]==0 && K[H[i][1]][1] >0))

// если одна вершина имеет компоненту а др. нет

{

if(K[H[i][0]][1]!=0)

K[H[i][1]][1]=K[H[i][0]][1];

if(K[H[i][1]][1]!=0)

K [H[i][0]][1]=K[H[i][1]][1];

T [H[i][0]][H[i][1]]=H[i][2];

T [H[i][1]][H[i][0]]=H[i][2];

}

if(K[H[i][0]][1]==0&&K[H[i][1]][1]==0)

// если обе вершины не имели компоненты

{

K[H[i][0]][1]=m;

K[H[i][1]][1]=m;

T[H[i][0]][H[i][1]]=H[i][2];

T[H[i][1]][H[i][0]]=H[i][2];

m++;

}

} // конец проверки всех ребер

kolreb=0;

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

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

if(T[i][j]<max&&T[i][j]!=0)

kolreb++;

L=new tint [kolreb];

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

L[i]=new int [3];

cout<<endl<<" Vivod reber maximalnogo vesa: \n ";

a=0;

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

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

if (T[i][j]<max&&T[i][j]!=0) {

L [a] [0] =i+1;

L [a] [1] =j+1;

L [a] [2] =T [i] [j];

a++;

}

for (int i=0; i<kolreb; i++) {

cout<<L [i][0] <<"-->";

cout<<L [i][1] <<" = ";

cout<<L [i][2] <<"\n ";

}

int b=0;

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

b+=L [i] [2];

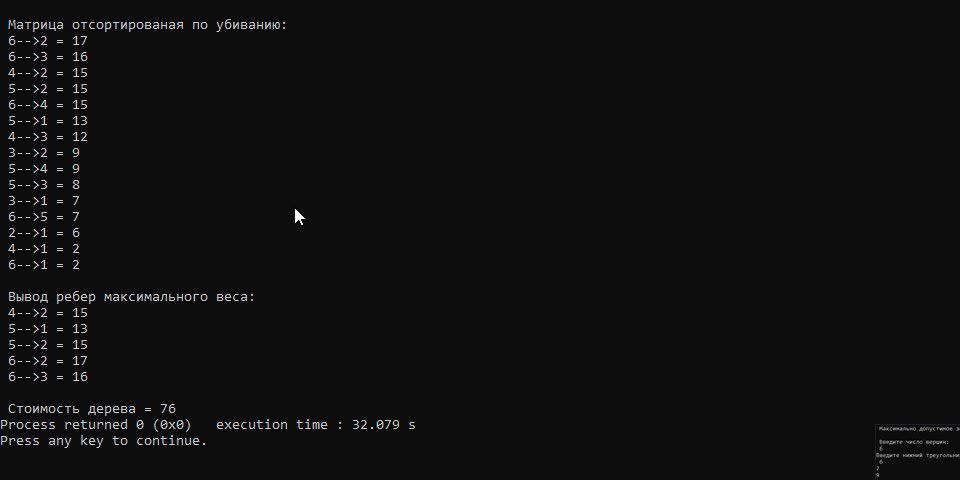
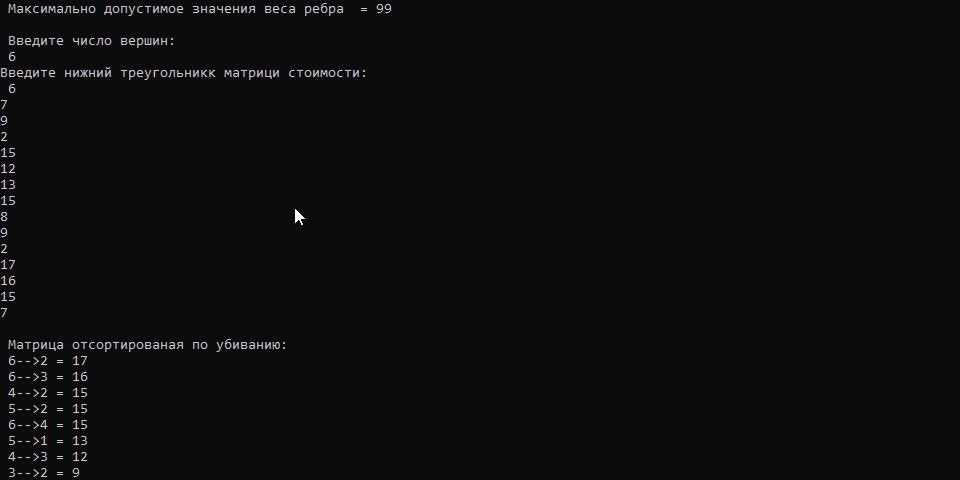
cout<<endl <<" Stoimost dereva = "<<b; // вывод стоимости

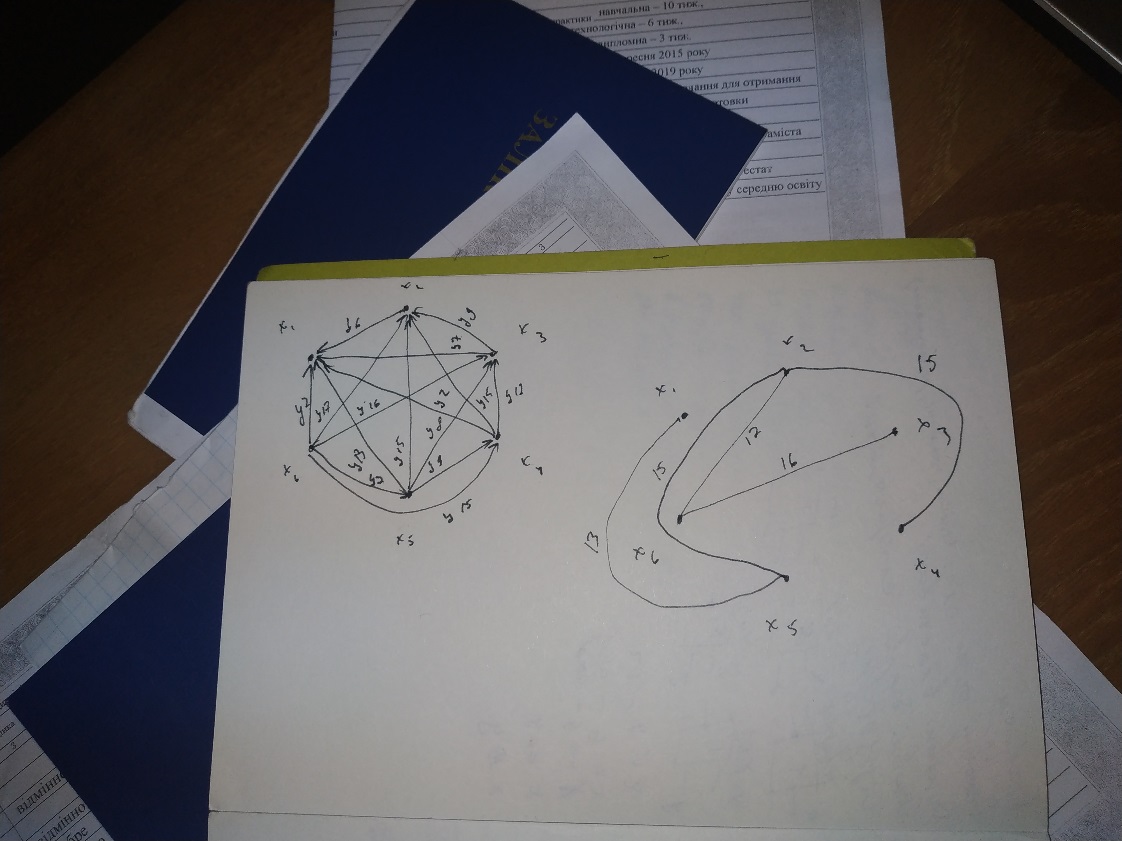
return 0;

system("pause");

}

**Результат програми:**





**Висновок:** Я сьогодні на парі навчився будувати мінімальне остовне дерево використовуючи мінімальний алгоритм Крускала.