Міністерство освіти і науки України  
Черкаський державний технологічний університет

Кафедра програмного забезпечення автоматизованих систем

**Звіт**

Про виконання лабораторної роботи №4

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 №4**

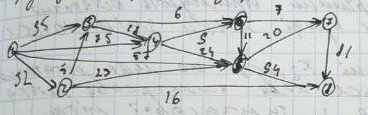
**Тема:** Знаходження найкоротшого шляху в графі. Алгоритм Дейкстри.

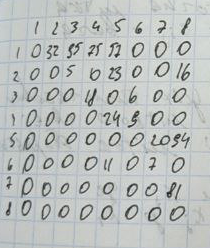
**Мета:** Опанувати алгоритмом Дейкстри для знаходження найкоротшого шляху в графі.

**Завдання:** Скласти програму, що реалізує алгоритм Дейкстри для заданого зваженого графа. Граф зв’язний, неповний, орієнтований.

**Теоретичні відомості:**

Структура графа задається матрицею ваг:

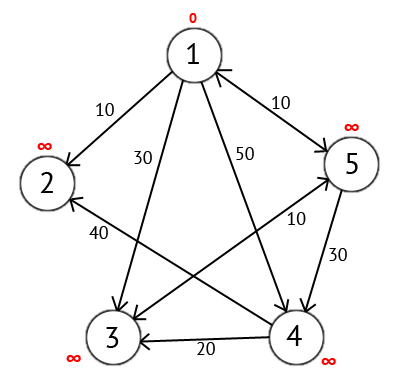




Дано: зважений орієнтований граф від вершин x, до всіх інших вершин цього графа.

Ініціалізація

Кожній вершині з множинами x ставимо у відповідність мітку – мінімальна відома відстань від цієї вершини до вершини x1. Алгоритм працює покроково, на кожному кроці відвідується одна вершина і зменшується її мітка. Робота алгоритму завершується коли відвідані вершини, мітка самої вершини x приймається рівною 0. Мітка інших вершин – нескінченність, тобто відстань невідома.



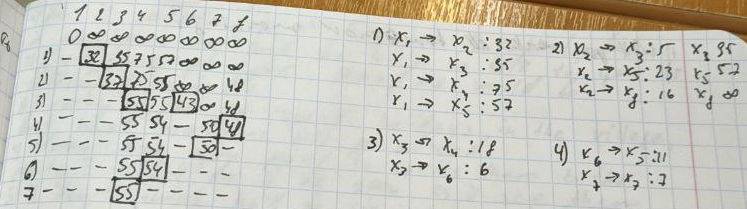
Мітки можуть бути тимчасовими і хаотичними.

Тимчасова мітка xj– це мінімальна відстань від x1 до xj, коли у визначені шляху на графі враховуються не всі маршрути від x1 до xj.

Остаточна мітка xj - це мінімальна відстань на графі від x1 до xj, тобто в кожний момент часу роботи алгоритму деякі вершини будуть мати остаточні мітки, а інша їх частина – тимчасові. Алгоритм закінчується тоді, коли вершина кінцева, або всі вершини отримують остаточні мітки , тобто відстань від x0 до z.

На кожному кроці мітки змінюються за правилом:

* Кожній вершині xj., яка не має остаточної мітки, присвоюється нова тимчасова мітка – найменша з її тимчасової мітки і числа xj та cij (wij - остаточна мітка xі ), де xі – вершина, якій присвоєна остаточна мітка на попередньому кроці, а cij – вага ребра з xi в xj.
* Визначається найменша з усіх тимчасових міток, яка і стає остаточною міткою своєї вершини. У випадку рівності міток вибирається довільна з них.
* Найкоротша відстань від xj до x1.



**Лістинг програми:**

#include <iostream>

#include <windows.h>

#include <stdlib.h>

#include <iomanip>

using namespace std;

void random (int W[20][20], int n); // прототип функии рандомизации

void input (int W[20][20], int n); // прототип функии ввода

char bufRus [256];

char \*Rus (const char\*text)

{

CharToOemA(text,bufRus);

return bufRus;

}

//объявления глобальных переменных

int i, j, n;

//главная функция main

int main()

{

int W[20][20]; // матрица весов

int u1, u2; // u1 - начальная вершина, u2 - конечная вершина

int length; // длина пути

int weight; // вес пути

int P[20];

const int GM = 842150451; // используется в качестве обозначения максимально возможного числа.

int m[20]; // метка вершин

int t; // текущая вершина

int d[20]; // всем вершинам преписывается вес.

int min; // минимальное значение

int k, c; // переменные для временного хранения данных

int Path[20]; // последовательность номеров вершин определяющая путь

int choise;

// текстовые сообщения выводимые на экран

cout<<Rus("Алгоритм Дейкстры\n");

C:cout<<Rus("Введите колличество вершин в графе от 2 до 20:\n");

cin >> n;

cout << endl;

if(n>20) // если вершин больше 20

{

cout<<Rus("Вы превысили колличество вершин в графе!!!");

goto C; // знаю что глупо использовать go to , зато работает

}

if(n<2) // если меньше 2 , то смысла нету

{

cout<<Rus("Вы занизили колличество вершин в графе!!!");

goto C;

}

A:cout<<Rus("\n Нажмите 2 чтобы заполнить ее вручную ...");

cin >> choise;

cout << endl;

// функция case для выбора варианта ввода

switch(choise){

case 1:

random(W, n);// если 1 то заполняем рандомно

break;

case 2:

input(W, n);

break;

case '\n':

case '\t':

case ' ':

break;

default:

{

cout<<Rus("Введен неверный символ\n\t\t повторите заново")<<endl<<endl;

goto A;

break;

}

}

//заменяем элементы главной диагонали нулями

cout<<setw(40)<<Rus("МАТРИЦА ДЛИН ДУГ ГРАФА\n\n");

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

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

{

if(W[i][j]<0)

W[i][j]=GM;

if(i==j)

W[i][j]=0;

W[i][j]=W[j][i];

};

// оформление и вывод матрицы весов

//вывод номеров вершин

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

cout<<setw(4)<<"V"<<i;

cout<<endl;

// вывод вертикальной верхний черты

for(i=1; i<=n\*3-2; i++)

cout<<setw(2)<<"-";

cout<<endl;

// вывод массива

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

{ for (j=1; j<=n; j++)

cout<<setw(4)<<W[i][j]<<"|";

cout<<setw(3)<<"V"<<i<<endl;

}

// вывод вертикальной нижней черты

for(i=1; i<=n\*3-2; i++)

cout<<setw(2)<<"-";

// ввод начальной и конечной вершины в графе

B:cout<<Rus("\n Номер начальной вершины пути <от 1 до ")<<n<<"> ";

cin>>u1;

cout<<endl;

cout<<Rus("Введите номер конечной вершины пути <от 1 до ")<<n<<"> ";

cin>>u2;

cout<<endl;

if(u1==u2)

{

cout<<Rus("Номер начальной вершины соответсвует конечной.\n\tПовторите попытку ввода искомых вершин еще раз!!!\n");

cout<<Rus("Путь найден\n");

cout<<Rus("Минимальная длина пути 0\n");

cout<<Rus("Вес пути 0\n" );

cout<<Rus("Последовательность номеров вершин 0\n" );

goto B;

}

// реализация алгоритма Дейкстры

length=0;

if(u1!=u2)

{

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

{

d[i]=GM;

P[i]=0;

m[i]=0;

}

d[u1]=0;

t=u1;

while(length==0)

{

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

{

if(W[t][i]<GM)

{

c=d[t]+W[t][i];

if(d[i]>c)

{

d[i]=c;

P[i]=t;

}

}

}

min=GM;

k=0;

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

{

if(m[i]==0)

if(d[i]<min)

{

min=d[i];

k=i;

}

}

if (k!=0)

{

m[k]=1;

t=k;

if(t==u2)

length=1;

}

else length=-1;

}

if(length==1)

{

Path[1]=u2;

length=2;

//j=u2;

for(j=u2; j!=u1; length++)

{

Path[length]=P[j];

j=P[j];

}

k=length/2;

for(i=1; i<=k; i++)

{

t=Path[i];

Path[i]=Path[length-i];

Path[length-i]=t;

}

length--;

}

weight=d[u2];

// Проверка и вывод полученных результатов

if (length == -1)

cout<<Rus("Путь не найден")<<endl;

else

{

cout<<Rus("Путь найден")<<endl;

cout<<Rus("Колличество соединенных путей ")<<length<<endl;

cout<<Rus("Минимальная длина пути " )<<weight<<endl;

cout<<Rus("Последовательность номеров вершин " );

for (j=1; j<=length; j++)

cout<<setw(3)<<"V"<<Path[j];

cout<<endl;

}

cout<<endl;

// выход из программы

D:cout<<Rus("Для повторного ввода данных нажмите 1, для выхода из программы нажмите 2: ");

cin>>choise;

cout<<endl<<endl;

switch(choise){

case 1: goto C;

break;

case 2: exit(1);

break;

default:

{

cout<<Rus("Введен неверный символ\n\t\t повторите заново")<<endl<<endl;

goto D;

break;

}

}

}

return 0;

}

void random (int W[20][20], int n)

{

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

for (j = 1; j<=n; j++)

W[i][j] =rand()%10;

}

void input (int W[20][20], int n)

{

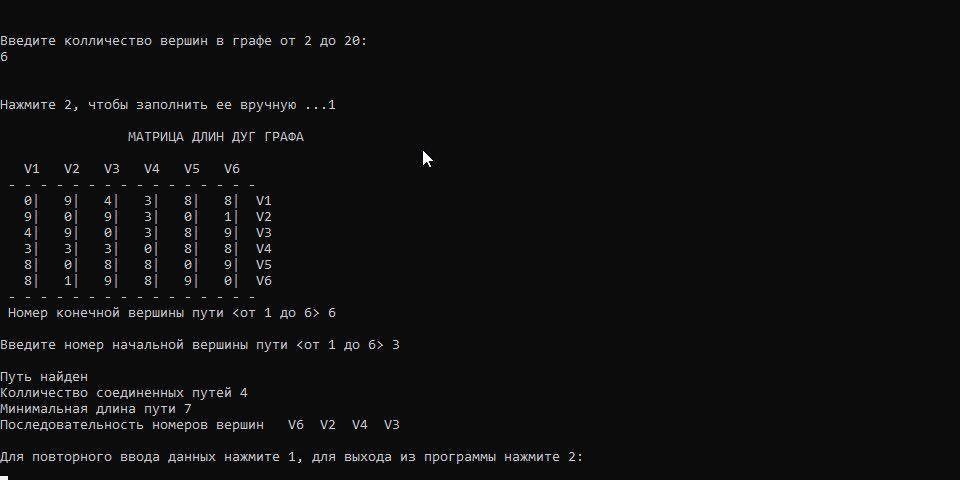
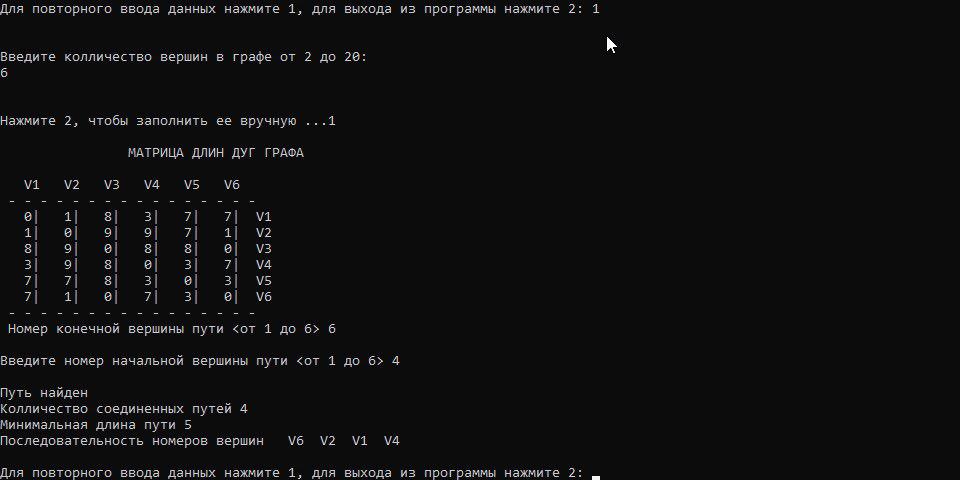
for (i=1; i<=n; i++)

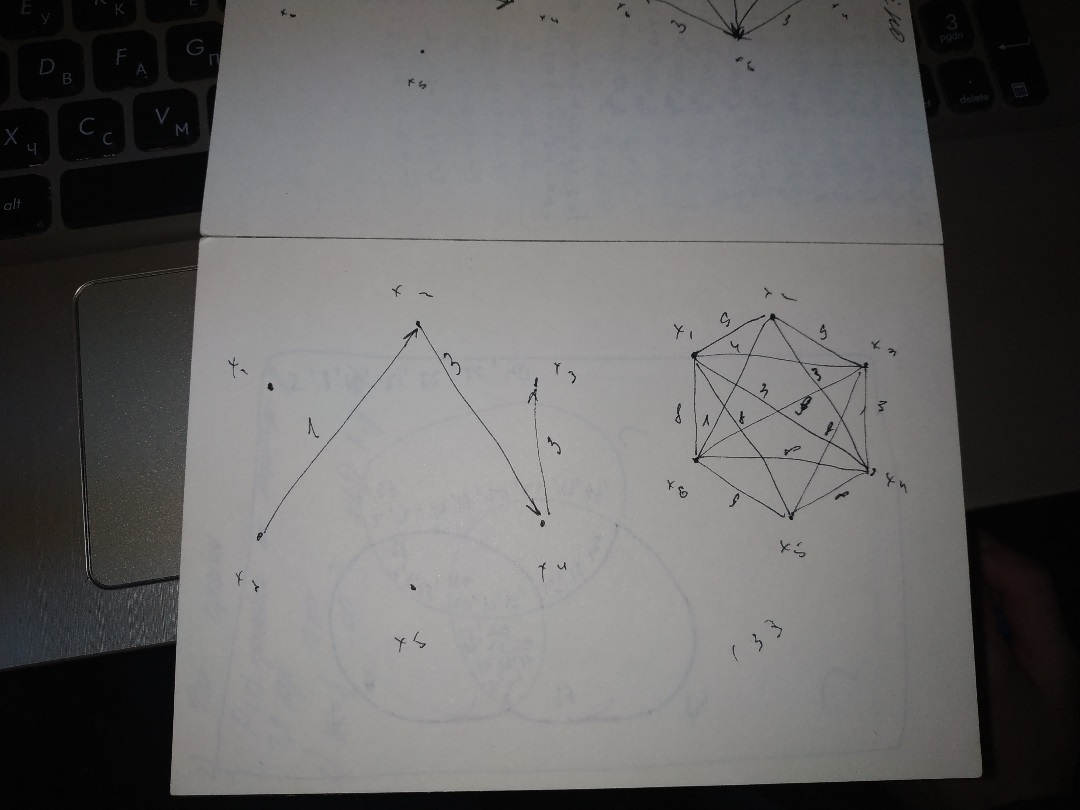
for (j = 1; j<=n; j++)

cin>>W[i][j];

}

**Результати виконання роботи:**





**Висновок:** Я сьогодні на парі опанував алгоритм Дейкстри для знаходження найкоротшого шляху в графі.