МІНІСТЕРСТВО ОСВІТИ І НАУКИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних наук та інформаційних технологій Кафедра систем штучного інтелекту

Розрахунково-графічна робота з курсу

“Дискретна математика”

Виконав: ст. гр. КН-113

Байдич Володимир

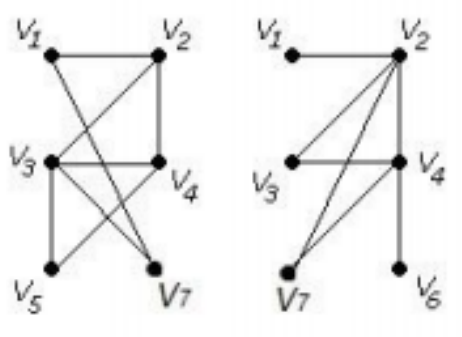
Викладач: Мельникова Н.І.

Варіант 11

Завдання № 1

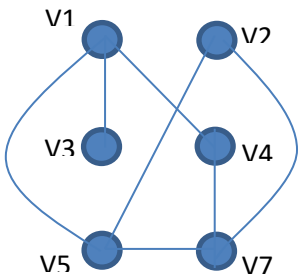
Виконати наступні операції над графами: 1) знайти доповнення до першого графу, 2) об’єднання графів, 3) кільцеву сумму G1 та G2 (G1+G2), 4) розмножити вершину у другому графі, 5) виділити підграф А - що скадається з 3-х вершин в G1 6) добуток графів.

11)

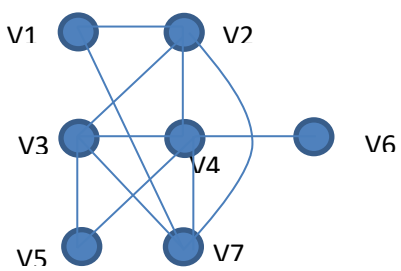


Розв’язок:

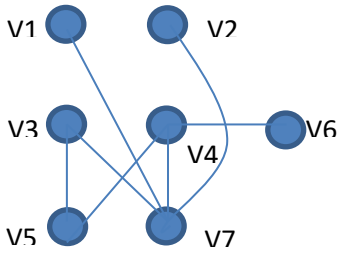
1) Доповнення:



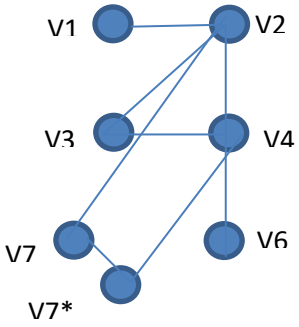
2) Об’єднання:



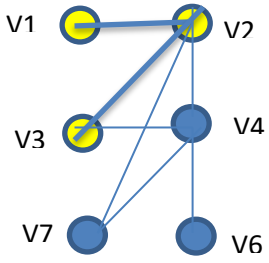
3) Симетрична різниця:

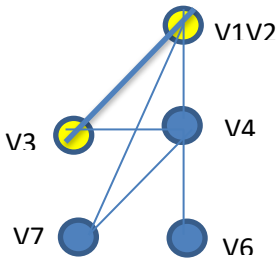


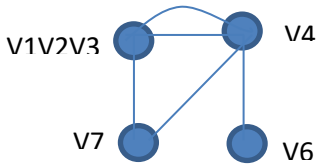
4) Розмноження вершини другого графу:



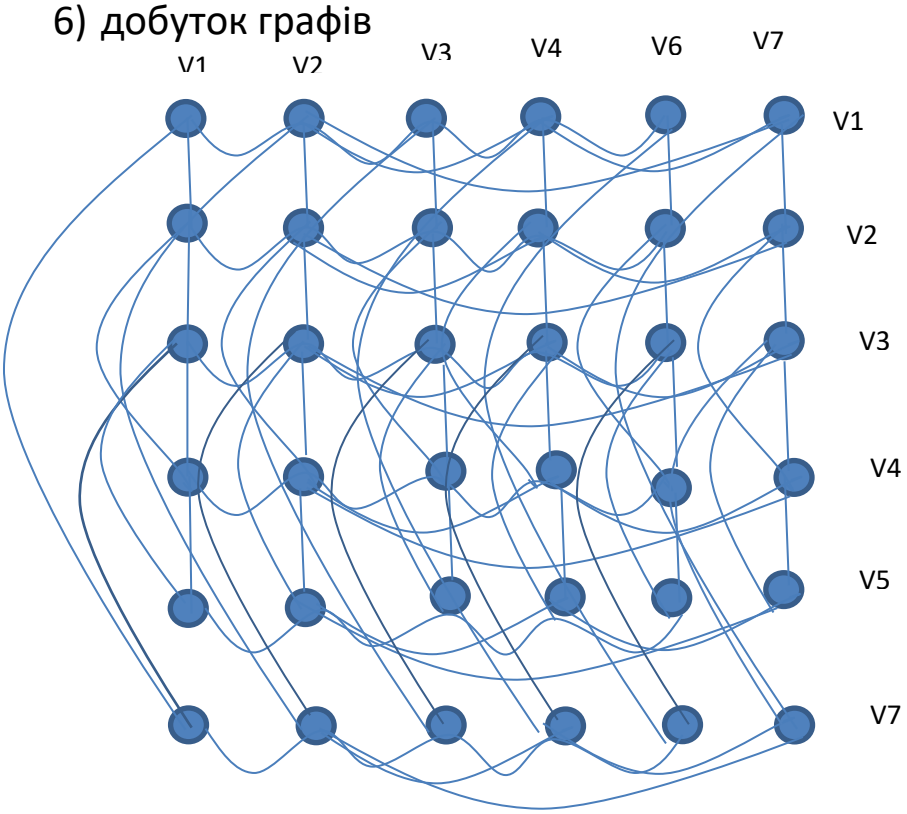
5)







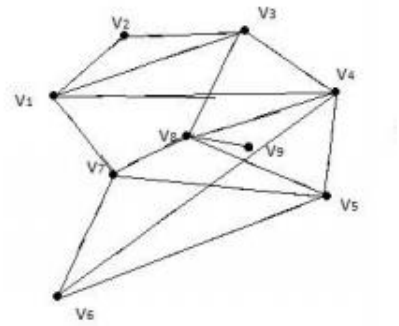
6) добуток графів



Завдання 2

Скласти таблицю суміжності для орграфа.

11)



	V1	V2	V3	V4	V5	V6	V7	V8	V9
V1	0	1	1	1	0	0	1	0	0
V2	1	0	1	0	0	0	0	0	0
V3	1	1	0	1	0	0	0	1	0
V4	1	0	1	0	1	1	0	1	0
V5	0	0	0	1	0	1	1	1	0
V6	0	0	0	1	1	0	1	0	0
V7	1	0	0	0	1	1	0	1	0
V8	0	0	1	1	1	0	1	0	1
V9	0	0	0	0	0	0	0	1	0

Завдання № 3

Для графа з другого завдання знайти діаметр.

Діаметр графа дорівнює 3.

Завдання № 4

Для графа з другого завдання виконати обхід дерева вглиб (варіант закінчується на непарне число) або вшир (закінчується на парне число).

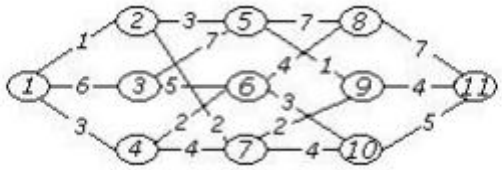
Верш	DFS	Стек
V1	1	V2V3V4V7
V2	2	V3V4V7

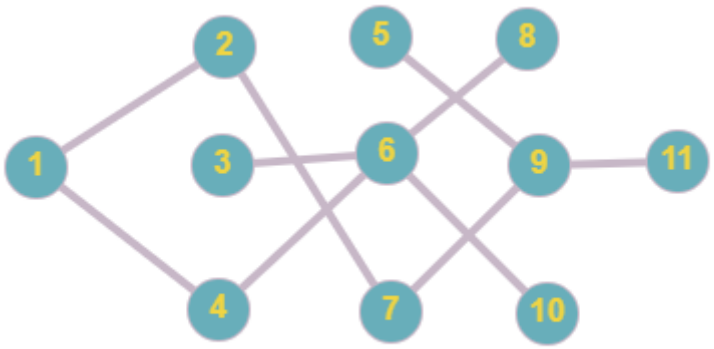
V3	3	V8V4V8
V4	4	V1V2V3V4
V5	5	V1V2V3V4V5
V6	6	V1V2V3V4V5V6
V7	7	V1V2V3V4V5V6V7
V8	8	V1V2V3V4V5V6V7V8
V9	9	V1V2V3V4V5V6V7V8V9

Завдання № 5

Знайти двома методами (Краскала і Прима) мінімальне остове дерево графа.

1)





За прима:

1-2 1

2-7 2

7-9 2

9-5 1

1-4 3

4-6 2

6-10 2

6-8 4

9-11 4

3-6 5

За краскала:

1-2 1

9-5 1

2-7 2

7-9 2

4-6 2

6-10 2

1-4 3

6-8 4

9-11 4

3-6 5

Програмна реалізація краскала:

```
#include <iostream>
#include <stdio.h>
using namespace std;
```

```
struct edge {

 int leng;
 int p1;
 int p2;
 bool in = false;
};
struct mas {

 int arr[11] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
 int c = 0;
};

int in(edge *reb, int n) {
 setlocale(LC_ALL, "Ukrainian");
 for (int i = 0; i < n; i++)
 {
  cout << "Введіть довжину " << i + 1 << " ребра: ";
  cin >> reb[i].leng;
  cout << "Введіть першу суміжну вершину з " << i + 1 << "
ребром: ";
  cin >> reb[i].p1;
  cout << "Введіть другу суміжну вершину з " << i + 1 << "
ребром: ";
```

```
 cin >> reb[i].p2;
 cout << endl;
 }
 return 0;
}
int main() {
 setlocale(LC_ALL, "Ukrainian");
 int n = 0, x = 100, y = 100;

 cout << "Введіть кількість ребер у графі: ";
 cin >> n;
 int z;
 cout << "Введіть кількість вершин у графі: ";
 cin >> z;
 cout << endl;

 edge *reb = new edge[n];
 mas inn[8];

 in(reb, n);

 for (int i = 0; i < n - 1; i++)
 {
  for (int j = 0; j < n - 1; j++)
```

```
 {
  if (reb[j].leng > reb[j + 1].leng) { swap(reb[j].leng, reb[j + 1].leng); swap(reb[j].p1, reb[j + 1].p1); swap(reb[j].p2, reb[j + 1].p2); }
 }
}

int c = -1;
for (int i = 0; i < n; i++)
{
 for (int j = 0; j < 8; j++)
 {
  for (int k = 0; k < z; k++)
  {
   if (reb[i].p1 == inn[j].arr[k]) { x = j; goto point0;; }
  }
 }
point0:;

 for (int j = 0; j < 8; j++)
 {
  for (int k = 0; k <z; k++)
  {
   if (reb[i].p2 == inn[j].arr[k]) { y = j; goto point1; }
```

```
   }
  }
 point1:;
  if (x != y && x == 100) { inn[y].arr[inn[y].c] = reb[i].p1;
inn[y].c++; }


  if (x != y && y == 100) { inn[x].arr[inn[x].c] = reb[i].p2;
inn[x].c++; }


  if (x != y && x != 100 && y != 100) {
   if (x < y) {
    for (int l = 0; l < inn[y].c; l++)
    {
     inn[x].arr[inn[x].c+l] = inn[y].arr[l];
     inn[y].arr[l] = 0;
    }
    inn[x].c += inn[y].c;
   }


   if (y < x) {
    for (int l = 0; l < inn[y].c; l++)
    {
     inn[y].arr[inn[y].c+l] = inn[x].arr[l];
     inn[x].arr[l] = 0;
```

```
    }
    inn[y].c += inn[x].c;
   }
  }
  if (x == 100 && y == 100) { c++;  inn[c].arr[inn[c].c] = reb[i].p1; inn[c].arr[inn[c].c + 1] = reb[i].p2; inn[c].c += 2; }

  reb[i].in = true;

  if (x == y && x != 100) { reb[i].in = false; }

  x = 100; y = 100;
 }

 cout << "ребра остового дерева: " << endl;

 int v = 0, s = 0;;

 for (int i = 0; i < n; i++)
 {
  if (reb[i].in == true) { cout << "Ребро" << ", що сполучає вершини " << reb[i].p1 << " " << reb[i].p2 << endl; v++; s += reb[i].leng; }
  if (v == z-1) { break; }
 }
```

```
 cout << "Остове дерево мінімальної ваги для даного графа:
" << s;
 return 0;}
```

Програмна реалізація прима:

```
#include <iostream>
#include <iomanip>

using namespace std;

void output(int size, int** adjacencyarr) {
cout << "  ";
for (int i = 0; i < size; i++) { cout << setw(3) << i + 1; }
for (int i = 0; i < size; i++) {
cout << endl << setw(3) << i + 1;
for (int j = 0; j < i; j++) {
cout << setw(3) << adjacencyarr[j][i];
}
cout << "  -";
for (int j = i + 1; j < size; j++) {
cout << setw(3) << adjacencyarr[i][j];
}
}
```

```
}


void main()
{
int size;
cout << "Enter amount of vertices ";
cin >> size;
int** adjacencyarr = new int* [size];
bool* vertex = new bool[size];
cout << "Enter adjacency matrix \n";
cout << "  ";
for (int i = 0; i < size; i++) { cout << setw(3) << i + 1;
adjacencyarr[i] = new int[size]; vertex[i] = 0; }
vertex[0] = 1;
cout << endl;
for (int i = 0; i < size; i++) {
cout << setw(2) << i + 1;
for (int j = 0; j < i; j++) { cout << setw(3) << adjacencyarr[j][i];
}
cout << "  -";
for (int j = i + 1; j < size; j++) { cin >> adjacencyarr[i][j]; }
}
cout << endl;
int min, minI, minJ;
```

```
for (int n = 0; n < size - 1; n++) {
min = 100, minI = 0, minJ = 0;
for (int i = 0; i < size; i++) {
for (int j = i + 1; j < size; j++) {
if (adjacencyarr[i][j] < min && adjacencyarr[i][j] != 0 &&
vertex[i]!=vertex[j]) {
min = adjacencyarr[i][j];
minI = i, minJ = j;
}
}
}
vertex[minI] = 1; vertex[minJ] = 1;
cout << n + 1 << ")conected " << minI + 1 << "-" << minJ + 1
<< endl;
}
}
```

Завдання № 6

Розв’язати задачу комівояжера для повного 8-ми вершинного графа методом «іди у найближчий», матриця вагів якого має вигляд:

11)

	1	2	3	4	5	6	7	8
1	∞	7	6	5	4	3	2	1
2	7	∞	1	5	6	4	2	3
3	6	1	∞	1	5	6	2	3
4	5	5	1	∞	3	2	2	2
5	4	6	5	3	∞	2	2	2
6	3	4	6	2	2	∞	5	5
7	2	2	2	2	2	5	∞	2
8	1	3	3	2	2	5	2	∞

S – (1,8),(8,4),(4,3),(3,2),(2,7),(7,5),(5,6),(6,1)

Довжина маршруту 1+2+1+1+2+2+2+3 = 14

Програмна реалізація:

```
#include <iostream>
#include <windows.h>
#include <bits/stdc++.h>
using namespace std;
int consisting(string x,set <string> s){
int c = 0;
for(auto i : s){
   if (x == i)
      c = 1;
}
if (c ==1)
   return 1;
else
   return 0;
}
class edge {
   public :
      string vertices_1;
      string vertices_2;
      int weight;
};
```

```
int main()
{
    SetConsoleOutputCP(CP_UTF8);
    set < string > vertices;

    char *p;
    char * s = new char[255];

    map <string , int > vertice_weight;
    set <string> ban;
    gets(s);
    p = strtok(s," ");
    vertices.insert(p);
    string current = p;
    while(p){
        vertices.insert(p);
        vertice_weight.insert(pair <string,int>(p,-1));
        p = strtok(NULL," ");

    }
    vector <edge> edges;
    while(1){
        edge x;
        cin>>x.vertices_1;
        if (x.vertices_1=="end")
            break;
        cin>>x.vertices_2>>x.weight;
        edges.push_back(x);
```

```
    }
    int min;
    string next,current_city,start;
    current_city = start;
    int sum,sum_min = 99999999;
    vector <string> way;
    for(auto start:vertices){
        vector <string> buf;
        buf.push_back(start);
        current_city = start;
        sum = 0;
        int n =0;
        ban.clear();
        while(1){
            min = 99999999;
            for (auto eedge : edges){
                if (eedge.vertices_1 == current_city){
                    if ((eedge.weight< min)&&(!consisting(eedge.vertices_2,ban)||((n
==vertices.size()-1)&&(eedge.vertices_2==start)))){
                        min = eedge.weight;
                        next = eedge.vertices_2;
                    }
                }
            }
            buf.push_back(next);
            if (min == 99999999){
                cout<<"шлях неможливий\n\n";
                break;
```

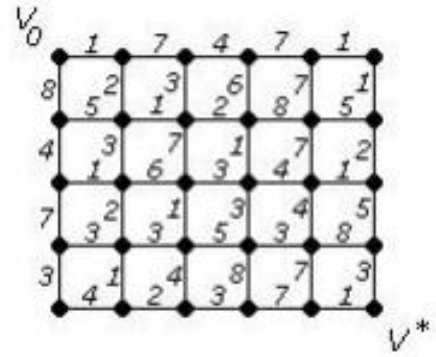
```
        }
        sum += min;
        n++;
        cout<<sum<<current_city<<next<<endl;
        ban.insert(current_city);
        current_city = next;
        if (current_city ==start){
            if (sum < sum_min){
                sum_min = sum;
                way.clear();
                for (auto vert: buf ){
                    way.push_back(vert);

                }
            }
            break;
        }

    }
    cout<<endl;
}
cout << sum_min<<endl<<"[";
for(auto i : way){
    cout<<i<<",";
}
cout<<"] - Мінімальний шлях та його вага";

return 0;
```

}

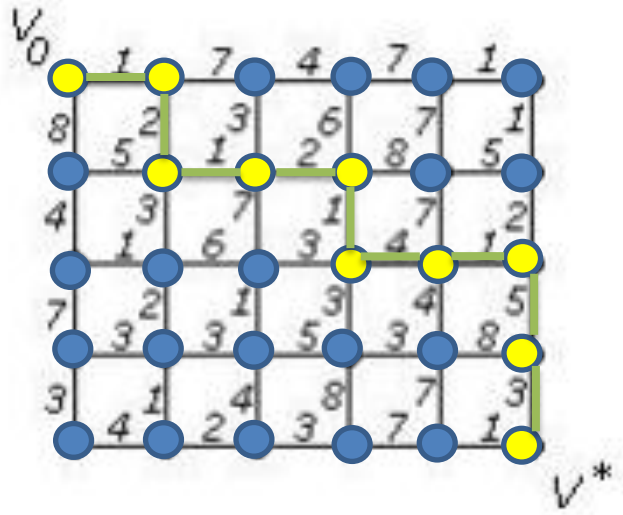
Завдання № 7

За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі між парою вершин V0 і V * .

11)



11)



Програмна реалізація алгоритму дейкстри:

```
#include <bits/stdc++.h>
```

```
using namespace std;

int main()
{
    int n,i,j;
    n=30;
    int mygraph[n][n];
    for (i=0;i<n;i++)
    {
        for (j=0;j<n;j++) mygraph[i][j]=0;
    }

    mygraph[0][1]=mygraph[1][0]=1;
    mygraph[0][6]=mygraph[6][0]=8;
    mygraph[1][2]=mygraph[2][1]=7;
    mygraph[1][7]=mygraph[7][1]=2;
    mygraph[2][3]=mygraph[3][2]=4;
    mygraph[2][8]=mygraph[8][2]=3;
    mygraph[3][4]=mygraph[4][3]=7;
    mygraph[3][9]=mygraph[9][3]=6;
    mygraph[4][5]=mygraph[5][4]=1;
    mygraph[4][10]=mygraph[10][4]=7;
    mygraph[5][11]=mygraph[11][5]=1;
    mygraph[6][7]=mygraph[7][6]=5;
```

```
mygraph[6][12]=mygraph[12][6]=4;
mygraph[7][8]=mygraph[8][7]=1;
mygraph[7][13]=mygraph[13][7]=3;
mygraph[8][9]=mygraph[9][8]=2;
mygraph[8][14]=mygraph[14][8]=7;
mygraph[9][10]=mygraph[10][9]=8;
mygraph[9][15]=mygraph[15][9]=1;
mygraph[10][11]=mygraph[11][10]=5;
mygraph[10][16]=mygraph[16][10]=7;
mygraph[11][17]=mygraph[17][11]=2;
mygraph[12][13]=mygraph[13][12]=1;
mygraph[12][18]=mygraph[18][12]=7;
mygraph[13][14]=mygraph[14][13]=6;
mygraph[13][19]=mygraph[19][13]=2;
mygraph[14][15]=mygraph[15][14]=3;
mygraph[14][20]=mygraph[20][14]=1;
mygraph[15][16]=mygraph[16][15]=4;
mygraph[15][21]=mygraph[21][15]=3;
mygraph[16][17]=mygraph[17][16]=1;
mygraph[16][22]=mygraph[22][16]=4;
mygraph[17][23]=mygraph[23][17]=5;
mygraph[18][19]=mygraph[19][18]=3;
mygraph[18][24]=mygraph[24][18]=3;
mygraph[19][20]=mygraph[20][19]=3;
```

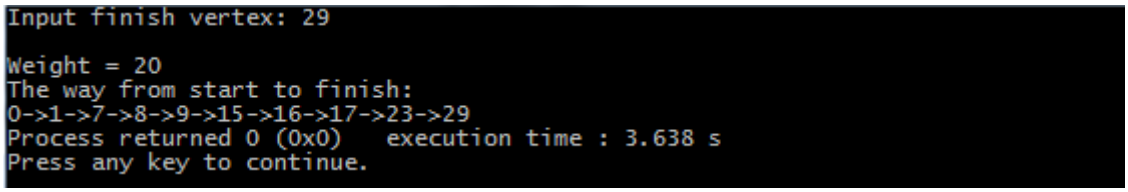
```
mygraph[19][25]=mygraph[25][19]=1;
mygraph[20][21]=mygraph[21][20]=5;
mygraph[20][26]=mygraph[26][20]=4;
mygraph[21][22]=mygraph[22][21]=3;
mygraph[21][27]=mygraph[27][21]=8;
mygraph[22][28]=mygraph[28][22]=7;
mygraph[22][23]=mygraph[23][22]=8;
mygraph[23][29]=mygraph[29][23]=3;
mygraph[24][25]=mygraph[25][24]=4;
mygraph[25][26]=mygraph[26][25]=2;
mygraph[26][27]=mygraph[27][26]=3;
mygraph[27][28]=mygraph[28][27]=7;
mygraph[28][29]=mygraph[29][28]=1;

int vertindex[n]{-1};
int numbvectors=0;
for (i=0;i<n;i++)
{
   for (j=0;j<n;j++)
   {
      if (mygraph[i][j]!=0) numbvectors++;
   }
}
int vert[n];
```

```
bool vertdot[n];
for (i=0;i<n;i++)
{
   vert[i]=INT_MAX;
   vertdot[i]=0;
}
vert[0]=0;
int siz=1;
int nmin,imin,jmin;
vertdot[0]=1;
while (numbvectors!=0)
{
   nmin=INT_MAX;
   for (i=0;i<n;i++)
   {
      if (vertdot[i]==1)
      {
         for (j=0;j<n;j++)
         {
            if
(vert[i]+mygraph[i][j]<nmin&&mygraph[i][j]!=0)
               {
                  nmin=vert[i]+mygraph[i][j];
                  imin=i;
```

```
                jmin=j;
            }
        }
    }
    if (vert[jmin]>nmin)
    {
        vert[jmin]=nmin;
        vertindex[jmin]=imin;
    }
    vertdot[jmin]=1;
    mygraph[imin][jmin]=mygraph[jmin][imin]=0;
    numbvectors-=2;//віднімаю ребра
}

int thelast;
int way[n];
i=0;
cout<<"Input finish vertex: ";
cin>>thelast;
cout<<"\nWeight = "<<vert[thelast]<<endl;
cout<<"The way from start to finish:"<<endl;
while (thelast!=0)
{
```

```
        way[i]=thelast;
        thelast=vertindex[thelast];
        i++;
    }
    way[i]=0;

    for (i;i>=0;i--)
        {
            cout<<way[i];
            if (i!=0) cout<<"->";
        }
}
```

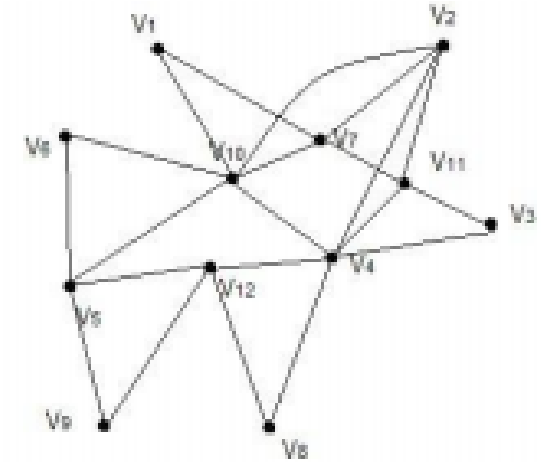
Результати роботи програми:



Завдання № 8

Знайти ейлеровий цикл в ейлеровому графі двома методами: а) Флері; б) елементарних циклів.

11)



За методом Флері

(1,7)

(7,2)

(2,4)

(4,11)

(11,2)

(2,10)

(10,6)

(6,5)

(5,12)

(12,9)

(9,5)

(5,10)

(7,11)

(11,3)

(3,4)

(4,12)

(12,8)

(8,4)

(4,10)

(10,1)

За методом елементарних циклів:

(1->7->2->10->1),(1->7->11->2->10->1),(1->7->11->3->4->2->10->1),(1->7->11->4->2->10->1),(1->7->2->4->10->1),(1->7->11->2->4->10->1),(1->7->2->11->3->4->10->1),(1->7->11->3->4->10->1),(1->7->2->11->4->10->1),(1->7->11->4->10->1),(1->7->2->4->12->9->5->10->1),(1->7->11->2->4->12->9->5->10->1),(1->7->2->11->3->4->12->9->5->10->1),(1->7->11->3->4->12->9->5->10->1),(1->7->2->11->4->12->9->5->10->1),(1->7->11->4->12->9->5->10->1),(1->7->2->4->8->12->9->5->10->1),(1->7->11->2->4->8->12->9->5->10->1),(1->7->2->11->3->4->8->12->9->5->10->1),(1->7->11->3->4->8->12->9->5->10->1),(1->7->2->11->4->8->12->9->5->10->1),(1->7->11->4->8->12->9->5->10->1),(1->7->2->4->12->5->10->1),(1->7->11->2->4->12->5->10->1),(1->7->2->11->3->4->12->5->10->1),(1->7->11->3->4->12->5->10->1),(1->7->2->11->4->12->5->10->1),(1->7->11->4->12->5->10->1),(1->7->2->4->8->12->5->10->1),(1->7->11->2->4->8->12->5->10->1),(1->7->2->11->3->4->8->12->5->10->1),(1->7->11->3->4->8->12->5->10->1),(1->7->2->11->4->8->12->5->10->1),(1->7->11->4->8->12->5->10->1),(1->7->2->4->12->9->5->6->10->1),(1->7->11->2->4->12->9->5->6->10->1),(1->7->2->11->3->4->12->9->5->6->10->1),(1->7->11->3->4->12->9->5->6->10->1),(1->7->2->11->4->12->9->5->6->10->1),(1->7->11->4->12->9->5->6->10->1),(1->7->2->4->8->12->9->5->6->10->1),(1->7->11->2->4->8->12->9->5->6->10->1),(1->7->2->11->3->4->8->12->9->5->6->10->1),(1->7->11->3->4->8->12->9->5->6->10->1),(1->7->2->11->4->8->12->9->5->6->10->1),(1->7->11->4->8->12->9->5->6->10->1),(1->7->2->4->12->5->6->10->1),(1->7->11->2->4->12->5->6->10->1),(1->7->2->11->3->4->12->5->6->10->1),(1->7->11->3->4->12->5->6->10->1),(1->7->2->11->4->12->5->6->10->1),(1->7->11->4->12->5->6->10->1),(1->7->2->4->8->12->5->6->10->1),(1->7->11->2->4->8->12->5->6->10->1),(1->7->2->11->3->4->8->12->5->6->10->1),(1->7->11->3->4->8->12->5->6->10->1),(1->7->2-

>11->4->8->12->5->6->10->1),(1->7->11->4->8->12->5->6->10->1),(1->7->10->1),(2->10->7->1),(2->10->4->3->11->7->1),(2->10->6->5->12->8->4->3->11->7->1),(2->10->5->12->8->4->3->11->7->1),(2->10->6->5->9->12->8->4->3->11->7->1),(2->10->5->9->12->8->4->3->11->7->1),(2->10->6->5->12->4->3->11->7->1),(2->10->5->12->4->3->11->7->1),(2->10->6->5->9->12->4->3->11->7->1),(2->10->5->9->12->4->3->11->7->1),(2->10->4->11->7->1),(2->10->6->5->12->8->4->11->7->1),(2->10->5->12->8->4->11->7->1),(2->10->6->5->9->12->8->4->11->7->1),(2->10->5->9->12->8->4->11->7->1),(2->10->6->5->12->4->11->7->1),(2->10->5->12->4->11->7->1),(2->10->6->5->9->12->4->11->7->1),(2->10->5->9->12->4->11->7->1),(2->10->4->3->11->1),(2->10->6->5->12->8->4->3->11->1),(2->10->5->12->8->4->3->11->1),(2->10->6->5->9->12->8->4->3->11->1),(2->10->5->9->12->8->4->3->11->1),(2->10->6->5->12->4->3->11->1),(2->10->5->12->4->3->11->1),(2->10->6->5->9->12->4->3->11->1),(2->10->5->9->12->4->3->11->1),(2->10->4->11->1),(2->10->6->5->12->8->4->11->1),(2->10->5->12->8->4->11->1),(2->10->6->5->9->12->8->4->11->1),(2->10->5->9->12->8->4->11->1),(2->10->6->5->12->4->11->1),(2->10->5->12->4->11->1),(2->10->6->5->9->12->4->11->1),(2->10->5->9->12->4->11->1),(2->10->7->11->1),(2->10->7->11->3->4->1),(2->10->7->11->4->1),(2->10->4->1),(2->10->6->5->12->8->4->1),(2->10->5->12->8->4->1),(2->10->6->5->9->12->8->4->1),(2->10->5->9->12->8->4->1),(2->10->6->5->12->4->1),(2->10->5->12->4->1),(2->10->6->5->9->12->4->1),(2->10->5->9->12->4->1),(2->4->10->7->1),(2->4->10->7->11->1),(2->11->3->4->10->7->1),(3->4->10->7->11->1),(2->11->4->10->7->1),(4->10->7->11->1),(4->10->6->5->12->8->1),(4->10->5->12->8->1),(4->10->6->5->9->12->8->1),(4->10->5->9->12->8->1),(4->10->6->5->12->1),(4->10->5->12->1),(4->10->6->5->9->12->1),(4->10->5->9->12->1),(2->4->12->9->5->10->7->1),(2->4->12->9->5->10->7->11->1),(2->11->3->4->12->9->5->10->7->1),(3->4->12->9->5->10->7->11->1),(2->11->4->12->9->5->10->7->1),(4->12->9->5->10->7->11->1),(2->4->8->12->9->5->10->7->1),(2->4->8->12->9->5->10->7->11->1),(2->11->3->4->8->12->9->5->10->7->1),(3->4->8->12->9->5->10->7->11->1),(2->11->4->8->12->9->5->10->7->1),(4->8->12->9->5->10->7->11->1),(2->4->12->5->10->7->1),(2->4->12->5->10->7->11->1),(2->11->3->4->12->5->10->7->1),(3->4->12->5->10->7->11->1),(2->11->4->12->5->10->7->1),(4->12->5->10->7->11->1),(2->4->8->12->5->10->7->1),(2->4->8->12->5->10->7->11->1),(2->11->3->4->8->12->5->10->7->1),(3->4->8->12->5->10->7->11->1),(2->11->4->8->12->5->10->7->1),(4->8->12->5->10->7->11->1),(5->10->6->1),(2->4->12->9->5->6->10->7->1),(2->4->12->9->5->6->10->7->11->1),(2->11->3->4->12->9->5->6->10->7->1),(3->4->12->9->5->6->10->7->11->1),(2->11->4->12->9->5->6->10->7->1),(4->12->9->5->6->10->7->11->1),(2->4->8->12->9->5->6->10->7->1),(2->4->8->12->9->5->6->10->7->11-

>1),(2->11->3->4->8->12->9->5->6->10->7->1),(3->4->8->12->9->5->6->10->7->11->1),(2->11->4->8->12->9->5->6->10->7->1),(4->8->12->9->5->6->10->7->11->1),(2->4->12->5->6->10->7->1),(2->4->12->5->6->10->7->11->1),(2->11->3->4->12->5->6->10->7->1),(3->4->12->5->6->10->7->11->1),(2->11->4->12->5->6->10->7->1),(4->12->5->6->10->7->11->1),(2->4->8->12->5->6->10->7->1),(2->4->8->12->5->6->10->7->11->1),(2->11->3->4->8->12->5->6->10->7->1),(3->4->8->12->5->6->10->7->11->1),(2->11->4->8->12->5->6->10->7->1),(4->8->12->5->6->10->7->11->1),(2->7->11->1),(2->7->11->3->4->1),(2->7->11->4->1),(2->4->3->11->1),(2->4->11->1),(3->11->4->1),(4->12->8->1),(5->12->9->1),(1->7->2->10->1)

Програмна реалізація алгоритму Флері:

```
#include <iostream>

#include <string.h>

#include <algorithm>

#include <list>

using namespace std;

class Graph

{

int V;

list<int>* adj;

public:

Graph(int V) { this->V = V; adj = new list<int>[V+1]; }
```

```
~Graph() { delete[] adj; }

void addEdge(int u, int v) { adj[u].push_back(v);
adj[v].push_back(u); }

void rmvEdge(int u, int v);

void printEulerTour();

void printEulerUtil(int s);

int DFSCount(int v, bool visited[]);

bool isValidNextEdge(int u, int v);

};

void Graph::printEulerTour()

{

int u = 1;

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

if (adj[i].size() & 1)

{

u = i; break;

}

printEulerUtil(u);
```

```
cout << endl;
}
void Graph::printEulerUtil(int u)
{
list<int>::iterator i;
for (i = adj[u].begin(); i != adj[u].end(); ++i)
{
int v = *i;
if (v != -1 && isValidNextEdge(u, v))
{
cout << u << "-" << v << " ";
rmvEdge(u, v);
printEulerUtil(v);
}
}
}
bool Graph::isValidNextEdge(int u, int v)
```

```
{
int count = 0;
list<int>::iterator i;
for (i = adj[u].begin(); i != adj[u].end(); ++i)
if (*i != -1)
count++;
if (count == 1)
return true;
bool visited[20];
memset(visited, false, V);
int count1 = DFSCount(u, visited);
rmvEdge(u, v);
memset(visited, false, V);
int count2 = DFSCount(u, visited);
addEdge(u, v);
return (count1 > count2) ? false : true;
}
void Graph::rmvEdge(int u, int v)
```

```
{
list<int>::iterator iv = find(adj[u].begin(), adj[u].end(), v);
*iv = -1;
list<int>::iterator iu = find(adj[v].begin(), adj[v].end(), u);
*iu = -1;
}
int Graph::DFSCount(int v, bool visited[])
{
visited[v] = true;
int count = 1;
list<int>::iterator i;
for (i = adj[v].begin(); i != adj[v].end(); ++i)
if (*i != -1 && !visited[*i])
count += DFSCount(*i, visited);
return count;
}
int main()
```

```
{
Graph g1(20);
g1.addEdge(1, 10);
g1.addEdge(1, 7);
g1.addEdge(2, 11);
g1.addEdge(2, 10);
g1.addEdge(2, 7);
g1.addEdge(2.4);
g1.addEdge(3, 11);
g1.addEdge(3, 4);
g1.addEdge(3, 6);
g1.addEdge(3, 4);
g1.addEdge(3, 8);
g1.addEdge(4, 7);
g1.addEdge(5, 12);
g1.addEdge(5, 6);
g1.addEdge(7, 12);
g1.addEdge(7, 8);
```

```
g1.addEdge(8, 11);

g1.addEdge(9, 10);

g1.addEdge(10, 11);

g1.addEdge(11, 12);

g1.printEulerTour();

return 0;

}
```

Завдання №9