МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ

“ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Розрахункова робота

з дисципліни

«Дискретна математика»

Виконав:

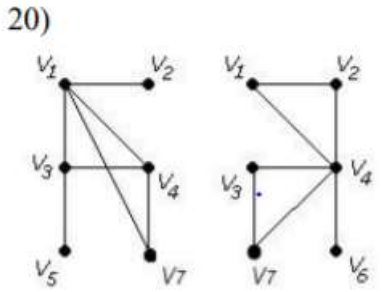
студент групи КН-114

Чорний Святослав

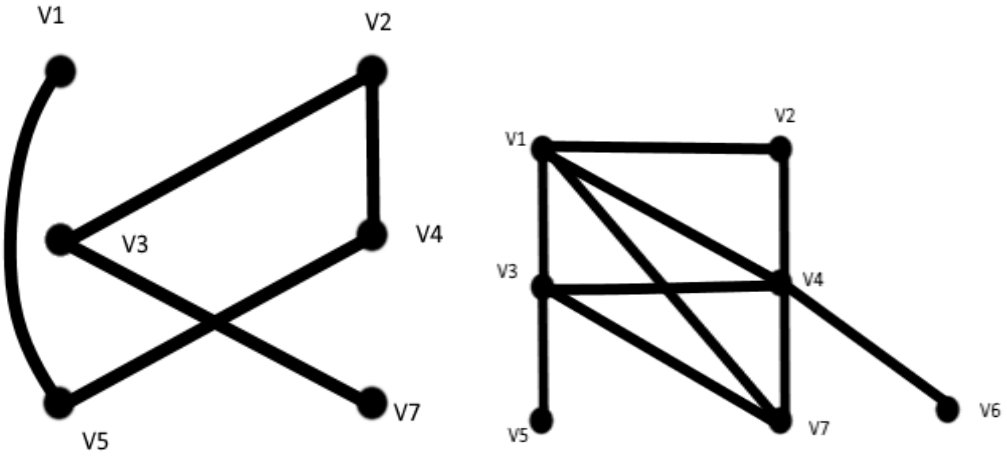
Львів – 2019р.

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.

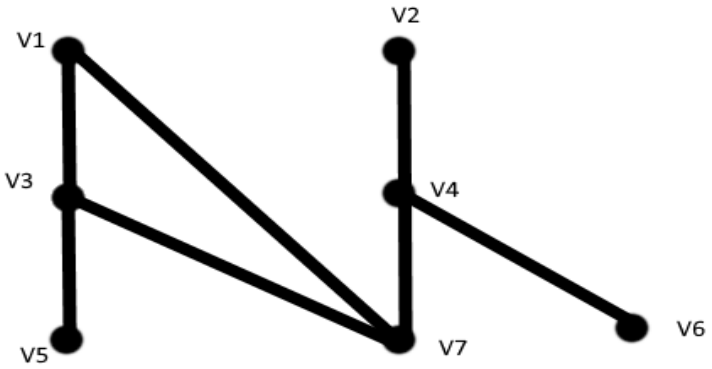
Дано:



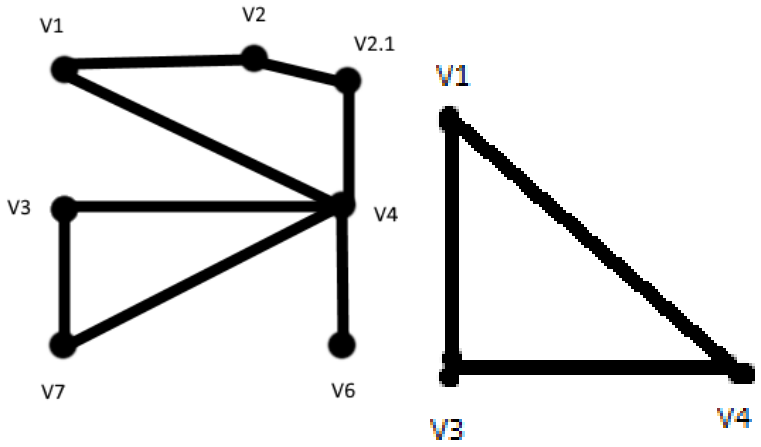
1.1,1.2



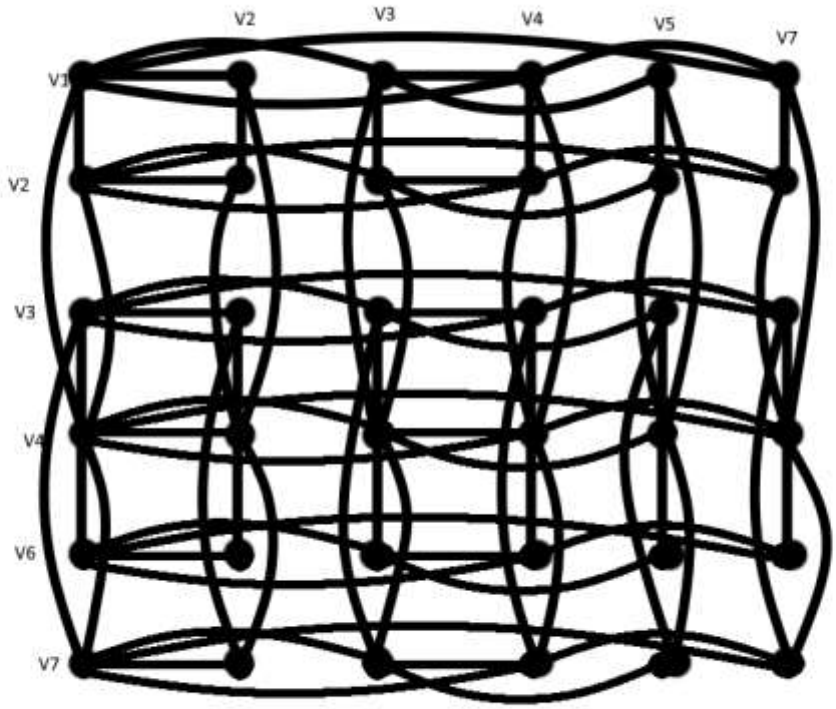
1.3



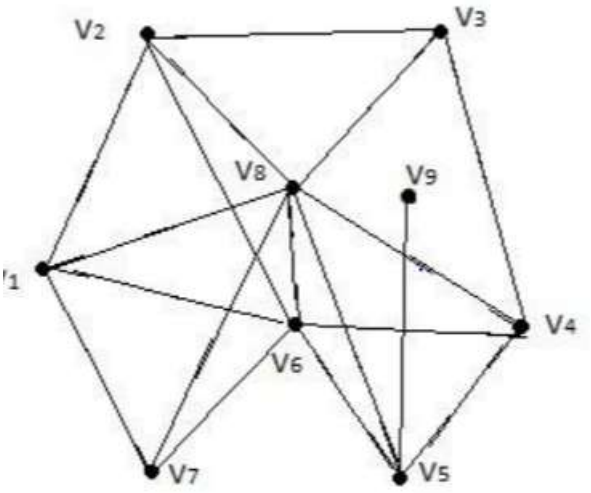
1.4,1.5



1.6



2.Скласти таблицю суміжності для орграфа.



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

Завдання №3

Діаметр графа=(V1->V9)=3

Завдання №4

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

V1	1	V1
V2	2	V1,v2
V8	3	V1,V2,V8
V6	4	V1,V2,V8,V6

V7	5	V1,V2,V8,V6,V7
-	-	V2,V8,V6,V7
V3	6	V2,V8,V6,V7,V3
-	-	V8,V6,V7,V3
V4	7	V8,V6,V7,V3,V4
V5	8	V8,V6,V7,V3,V4,V5
-	-	V6,V7,V3,V4,V5
-	-	V7,V3,V4,V5
	-	V3,V4,V5
	-	V4,V5
-	-	V5
V9	9	V5,V9
-	-	V9
-	-	~~0~~

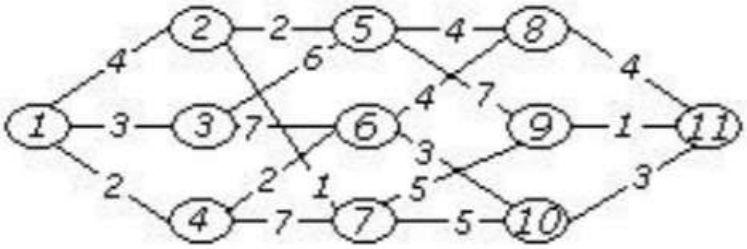
```
#include <iostream>
#include <queue>
using namespace std;
int main()
{
	queue <int> queue_bfs;
	int mas[9][9] = {

	{0,1,0,0,0,1,1,1,0},
	{1,0,1,0,0,1,0,1,0},
	{0,1,0,1,0,0,0,1,0},
	{0,0,1,0,1,1,0,1,0},
	{0,0,0,1,0,1,0,1,1},
	{1,1,0,1,1,0,1,1,0},
	{1,0,0,0,0,1,0,1,0},
	{1,1,1,1,1,1,1,0,0},
	{0,0,0,0,1,0,0,0,0},
	 };
	int vertices[9];
	for (int i = 0; i < 9; i++)
		vertices[i] = 0;
	queue_bfs.push(0);
	cout << "BFS= " ;
	while (!queue_bfs.empty())
	{
		int node = queue_bfs.front();
		queue_bfs.pop();
		vertices[node] = 2;
		for (int j = 0; j < 9; j++)
		{
			if (mas[node][j] == 1 && vertices[j] == 0)
```

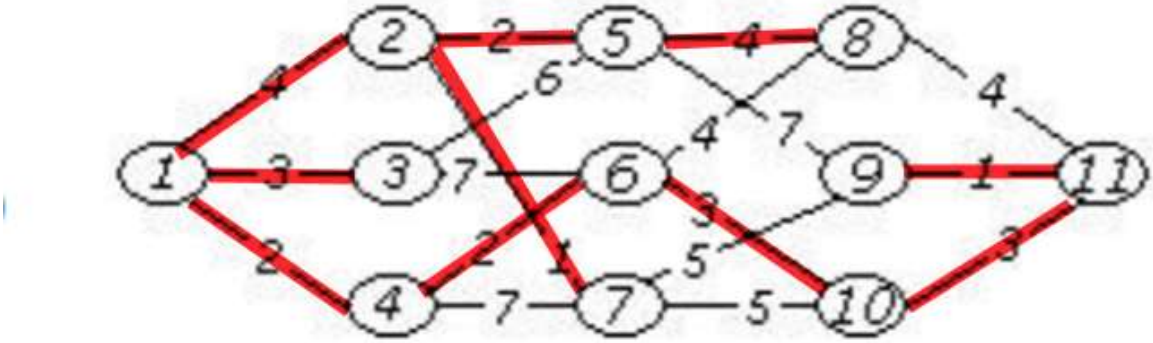
```
                {
                        queue_bfs.push(j);
                        vertices[j] = 1;
                }
        }
        cout << node + 1 << " ";
    }

    return 0;
                                        }
```

```
BFS= 1 2 6 7 8 3 4 5 9
C:\Users\sviti\source\re
```

5

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





Алгоритм Прима

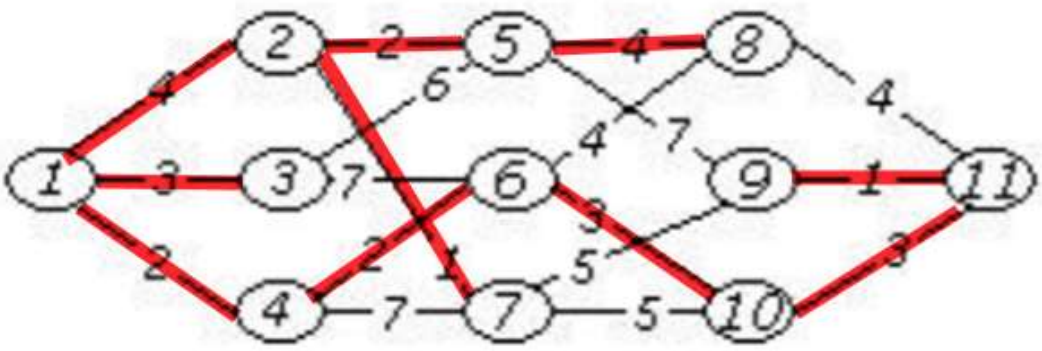
```
#include <iostream>

using namespace std;

struct rebro
{
    int t1;
    int t2;
    int waga;
};

void vvid (rebro*p, int n, int m)
{
    cout<<"Enter rebra(1 vershyna | 2vershyna | waga):\n";
    for (int i=0; i<m; i++)
    {
     cout<<"rebro"<<i+1<<": ";
```

```
     cin>>p[i].t1>>p[i].t2>>p[i].waga;
     while (p[i].t1<0 || p[i].t1>n || p[i].t2<0 || p[i].t2>n || p[i].waga<0)
     {
         cout<<"Uncorrect  "<<endl;
         cout<<"Enter again"<<endl;
         cin>>p[i].t1>>p[i].t2>>p[i].waga;
    }
}
}
void bulb(rebro*p, int n)
{
    rebro temp;
    for (int i=0; i<n; i++)
    {
         for (int j=0; j<n-i-1; j++)
         {
             if (p[j].waga > p[j+1].waga)
             {
                 temp = p[j];
                 p[j] = p[j+1];
                 p[j+1] = temp;

             }
         }
    }
}

bool vkluchene(int* a, int n, int f)
{
    for (int i=0; i<n; i++)
    {
        if (f == a[i])
        {
            return true;
        }
    }
    return false;
}
bool minn(int w, rebro* ed, int m, int* v, int n)
{
     for (int j=0; j<m; j++)
     {
         if(((!vkluchene(v, n, ed[j].t1) && vkluchene(v, n, ed[j].t2)) ||
             (vkluchene(v, n, ed[j].t1) &&  !vkluchene(v, n, ed[j].t2)))
            && ed[j].waga < w)
         {
             return false;
         }
     }
     return true;
}

void pryma (rebro* ed, int*v, rebro*tree, int n, int m, int&i, int&j)
{
    if (i==n)
    {
        return;
    }
    else if (j==n)
    {
        j=1;
        vkluchene(v, n, ed[j].t2);
    }
    if(vkluchene(v, n, ed[j].t1) && vkluchene(v, n, ed[j].t2))
    {
```

```
        j++;
        pryma(ed, v, tree, n, m, i, j);
    }
    else if(!(vkluchene(v, n, ed[j].t1)) && vkluchene(v, n, ed[j].t2)
          )
    {
     tree[i-1]=ed[j];
     v[i] = ed[j].t1;
    j++;
    i++;
    pryma(ed, v, tree, n, m, i, j);
    }
    else if (vkluchene(v, n, ed[j].t1) && !vkluchene(v, n, ed[j].t2) && minn(ed[j].waga,
ed, m, v, n))
            {
                tree[i-1]=ed[j];
                v[i] = ed[j].t2;

               j++;
               i++;
               pryma(ed, v, tree, n, m, i, j);
            }
            else
               {
                   j++;
                   pryma(ed, v, tree, n, m, i, j);
               }
 }

int main()
{
int n,m;
cout<<"Number of verticles=";
cin>>n;
cout<<"Number of rebres=";
cin>>m;
cout<<endl;
rebro *ed = new rebro[m];
int *v = new int[n];
rebro *tree = new rebro [n-1];
vvid (ed,n,m);
bulb (ed,m);
v[0]=ed[0].t1;
v[1]=ed[0].t2;
tree[0]=ed[0];
int i=2;
int j=1;
pryma (ed,v,tree,n,m,i,j);
cout<<"\nV = { ";
for (int x=0; x<n; x++)
{
   cout<<v[x]<<",";
}
 cout<<"}\nE = { ";
for (int x=0; x<n-1; x++)
{
    cout<<"{"<<tree[x].t1<<";"<<tree[x].t2<<"} ";
}
cout<<"}\n";
return 0;
```

```
V = { 2,7,5,1,4,6,3,10,11,9,8,}
E = { {2;7} {2;5} {1;2} {1;4} {4;6} {1;3} {6;10} {10;11} {9;11} {5;8} }
```

```
}
```



Алгоритм Краскала

```
#include <iostream>
#include<iomanip>
using namespace std;
#define INF 10000000

int* parent;

int find(int i)
{
	while (parent[i] != i)
		i = parent[i];
	return i;
}
void createSets(int i, int j)
{
	int a = find(i);
	int b = find(j);
	parent[a] = b;
}

void kruskal(int** cost, int ver)
{
	cout<<"мінімальне остове дерево: "<<endl;


	for (int i = 0; i < ver; i++)
		parent[i] = i;

	int edge_count = 0;
	while (edge_count < ver - 1) {
		int min = INF, a = -1, b = -1;
		for (int i = 0; i < ver; i++) {
			for (int j = 0; j < ver; j++) {
				if (find(i) != find(j) && cost[i][j] < min) {
					min = cost[i][j];
					a = i;
					b = j;
				}
			}
		}

		createSets(a, b);
		cout << "Ребро" << edge_count++ << ": (" << a + 1 << ", " << b + 1 << ") =
" << min << endl;


	}
```

```
}

int main()
{
	setlocale(LC_ALL, "Ukrainian");


	int ver;
	int reb;
	cout << "Кількість вершин: ";
	cin >> ver;
	cout << "Кількість ребер: ";
	cin >> reb;

	parent = new int[ver];
	int** cost = new int* [ver];
	for (int i = 0; i < ver; i++) {
		cost[i] = new int[ver];
	}
	for (int i = 0; i < ver; i++) {
		for (int j = 0; j < ver; j++) {
			cost[i][j] = INF;
		}
	}
	int a, b, weight;
	for (int i = 0; i < reb; i++) {
		cout << "Ребро" << i << ": " << endl;
		cout << "Вершина1 = ";
		cin >> a;
		cout << "Вершина2 = ";
		cin >> b;
		cout << "Вага = ";
		cin >> weight;
		cost[--a][--b] = weight;
		cost[b][a] = weight;
	}


	cout <<endl<< "Матриця: " << endl;
	for (int i = 0; i < ver; i++) {
		for (int j = 0; j < ver; j++) {
			cout << setw(12) << cost[i][j];
		}
		cout << endl;
	}
	cout << endl;


	kruskal(cost, ver);

	return 0;
```

}

Завдання № 6

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

20)

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

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

	2	3	4	15	6	7	8
2	-	5	1	5	1	5	1
3	5	-	5	6	1	5	7
4	1	5	-	6	4	5	5
15	5	6	6	-	3	2	2
6	1	1	4	3	-	2	2
7	5	5	5	2	2	-	2
8	1	7	5	2	2	2	-

	2	3	4	6	157	8
2	-	5	1	1	5	1
3	5	-	5	1	5	7
4	1	5	-	4	5	5
6	1	1	4	-	2	2
157	5	5	5	2	-	2
8	1	7	5	2	2	-

	2	3	4	1576	8
2	-	5	1	1	1
3	5	-	5	1	7
4	1	5	-	4	5
1576	1	1	4	-	2
8	1	7	5	2	-

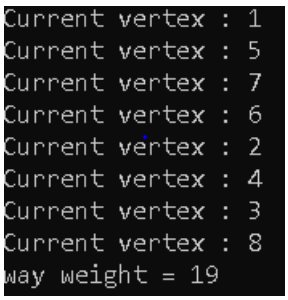
	15762	3	4	8
15762	-	5	1	1
3	5	-	5	7
4	1	5	-	5
8	1	7	5	-

	3	157624	8
3	-	5	7
157624	5	-	5
8	7	5	-

	1574263	8
1576243	-	7
8	7	-

1+2+2+1+1+5+7=19

```
#include<iostream>
#include<vector>
using namespace std;
const int INF = INT_MAX;
const int n = 8;
vector<int>visited;
int way = 0;
int graph[n][n] =
{
{INF,4,6,5,1,2,3,5},
{4,INF,5,1,5,1,5,1},
{6,5,INF,5,6,1,5,7},
{5,1,5,INF,6,4,5,5},
{1,5,6,6,INF,3,2,2},
{2,1,1,4,3,INF,2,2},
{3,5,5,5,2,2,INF,2},
{5,1,7,5,2,2,2,INF}
};
bool checkVis(int value) {
	bool flag = false;
	for (auto v : visited) if (v == value)flag = true;
	return flag;
}
void findMin() {
	int ver = visited.at(visited.size() - 1);
	cout << "Current vertex : " << ver + 1 << endl;
	int index;
	int min = INF;
	for (int i = 0; i < n; i++)
		if (graph[i][ver] < min && !checkVis(i)) {
			min = graph[i][ver];
			index = i;
		}
	if (visited.size() != n) {//остання вершина
		way += min;
	}
	visited.push_back(index);
}
int main() {
	visited.push_back(0);
	while (visited.size() <= n) {
		findMin();
	}
	cout << "way weight = " << way;
	return 0;
}
```



Комивояжера починаючи з 2 вершини

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

	1	3	24	5	6	7	8
1	-	6	5	1	2	3	5
3	6	-	5	6	1	5	7
24	5	5	-	6	4	5	5
5	1	6	6	-	3	2	2
6	2	1	4	3	-	2	2
7	3	5	5	2	2	-	2
8	5	7	5	2	2	2	-

	1	3	5	246	7	8
1	-	6	1	2	3	5
3	6	-	6	1	5	7
5	1	6	-	3	2	2
246	2	1	3	-	2	2
7	3	5	2	2	-	2
8	5	7	5	2	2	-

	1	2463	5	7	8
1	-	6	1	3	5
2463	6	-	6	5	7
5	1	6	-	2	2
7	3	5	2	-	2
8	5	7	5	2	-

	1	5	24637	8
1	-	1	3	5
5	1	-	2	2
24637	3	2	-	2
8	5	5	2	-

	1	246375	8
1	-	1	5
246375	1	-	2
8	5	5	-

	2463751	8
2463751	-	5
8	5	-

Мин шлях=1+4+1+5+2+1+5=19

Комивояжера починаючи з 3 вершини

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

	1	2	4	5	36	7	8
1	-	4	5	1	2	3	5
2	4	-	1	5	1	5	1
4	5	1	-	6	4	5	5
5	1	5	6	-	3	2	2
36	2	1	4	3	-	2	2
7	3	5	5	2	2	-	2
8	5	1	5	2	2	2	-

	1	362	4	5	7	8
1	-	4	5	1	3	5
362	4	-	1	5	5	1
4	5	1	-	6	5	5
5	1	5	6	-	2	2
7	3	5	5	2	-	2
8	5	1	5	2	2	-

	1	3624	5	7	8
1	-	5	1	3	5
3624	5	-	6	5	5
5	1	6	-	2	2
7	3	5	2	-	2
8	5	5	2	2	-

	1	5	7	36248
1	-	1	3	5
5	1	-	2	2
7	3	2	-	2
36248	5	2	2	-

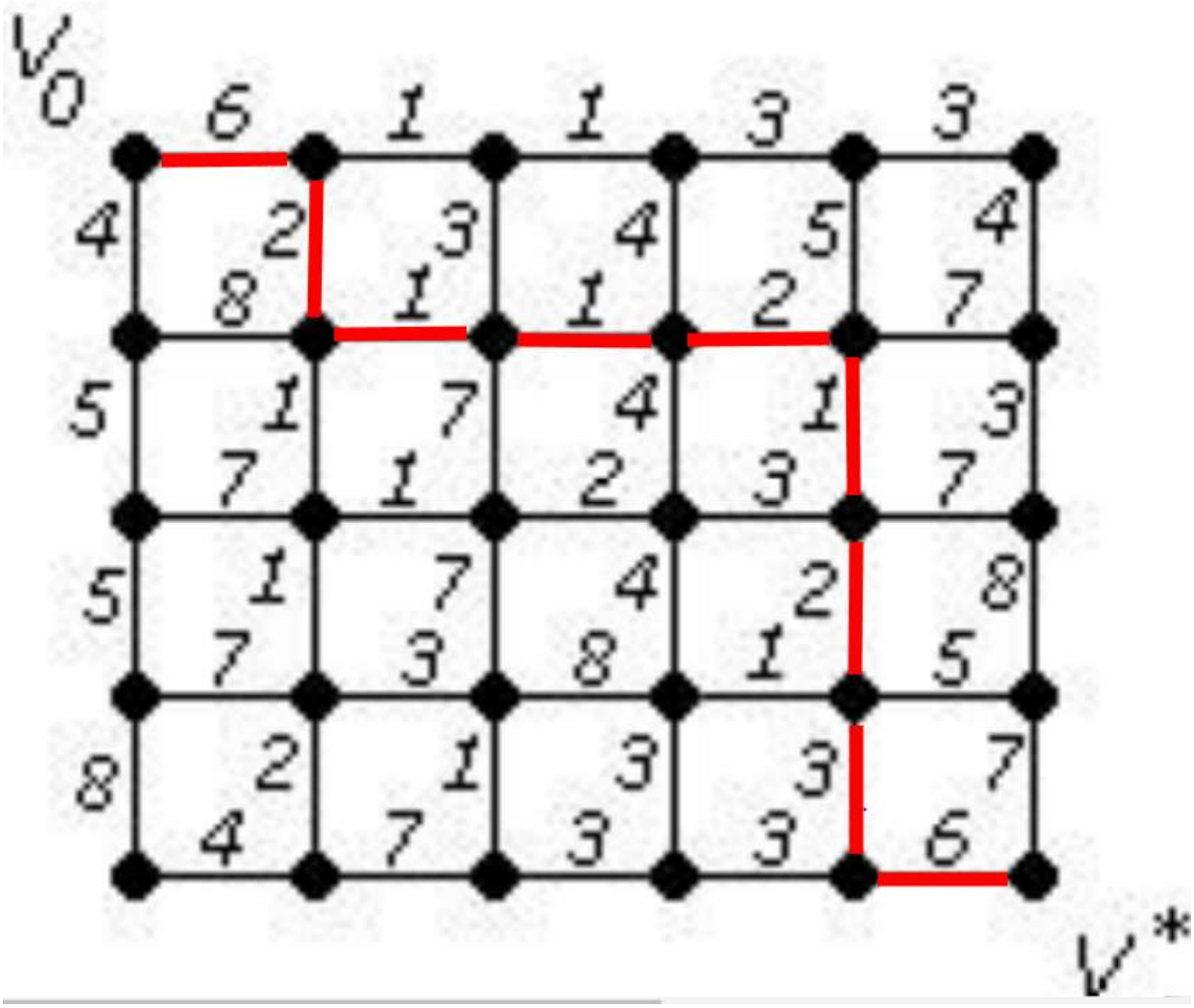
	1	362485	7
1	-	1	3
362485	1	-	2
7	3	2	-

	3624851	7
3624851	-	3
7	3	-

Мин шлях=1+1+1+5+2+1+3=14

Завдання № 7

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



Мин шлях=24

```
#define _CRT_SECURE_NO_WARNINGS
#include<iostream>

using namespace std;


int main()
{

	int maximalNumber = 10000;

	int vertex, edges;//к-сть вершин, ребер
```

```
cout << " vertexes: ";
cin >> vertex;
cout << " edges: ";
cin >> edges;
int** a = new int* [vertex];
int* d = new int[vertex];
int* visited = new int[vertex];
int temp, minindex, min;

	for (int i = 0; i < vertex; i++)
{
	a[i] = new int[vertex];
}


for (int i = 0; i < vertex; i++)
{
	for (int j = 0; j < vertex; j++) {
		a[i][j] = 0;
	}
}



	a[0][1] = 6;
	a[0][6] = 4;
	a[1][0] = 6;
	a[1][2] = 1;
	a[1][7] = 2;
	a[2][1] = 1;
	a[2][8] = 3;
	a[2][3] = 1;
	a[3][2] = 1;
	a[3][4] = 3;
	a[3][9] = 4;
	a[4][3] = 3;
	a[4][5] = 3;
	a[4][10] = 5;
	a[5][4] = 3;
	a[5][11] = 4;
	a[6][0] = 4;
	a[6][7] = 8;
	a[6][12] =5;
	a[7][1] = 2;
	a[7][6] = 8;
	a[7][8] = 1;
	a[7][13] = 1;
	a[8][7] = 1;
	a[8][2] = 3;
	a[8][14] = 7;
	a[8][9] = 1;
	a[9][8] = 1;
	a[9][3] = 4;
	a[9][15] = 4;
	a[9][10] = 2;
	a[10][9] = 2;
	a[10][4] = 5;
	a[10][16] = 1;
	a[10][11] = 7;
	a[11][10] = 7;
	a[11][5] = 4;
	a[11][17] = 3;
	a[12][6] = 5;
	a[12][13] = 7;
```

```
        a[12][18] = 5;
        a[13][12] = 7;
        a[13][7] = 1;
        a[13][19] = 1;
        a[13][14] = 2;
        a[14][13] = 2;
        a[14][8] = 7;
        a[14][20] = 7;
        a[14][15] = 2;
        a[15][14] = 2;
        a[15][9] = 4;
        a[15][21] = 4;
        a[15][16] = 3;
        a[16][15] = 3;
        a[16][10] = 1;
        a[16][22] = 2;
        a[16][17] = 7;
        a[17][16] = 7;
        a[17][11] = 3;
        a[17][23] = 8;
        a[18][12] = 5;
        a[18][19] = 7;
        a[18][24] = 8;
        a[19][18] = 7;
        a[19][13] = 1;
        a[19][25] = 2;
        a[19][20] = 3;
        a[20][19] = 3;
        a[20][14] = 7;
        a[20][26] = 1;
        a[20][21] = 8;
        a[21][20] = 8;
        a[21][15] = 4;
        a[21][27] = 3;
        a[21][22] = 1;
        a[22][21] = 1;
        a[22][16] = 2;
        a[22][28] = 3;
        a[22][23] = 5;
        a[23][22] = 5;
        a[23][17] = 8;
        a[23][29] = 7;
        a[24][18] = 8;
        a[24][25] = 4;
        a[25][24] = 4;
        a[25][19] = 2;
        a[25][26] = 7;
        a[26][25] = 7;
        a[26][20] = 1;
        a[26][27] = 3;
        a[27][26] = 3;
        a[27][21] = 3;
        a[27][28] = 3;
        a[28][27] = 3;
        a[28][22] = 3;
        a[28][29] = 6;
        a[29][28] = 6;
        a[29][23] = 7;



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

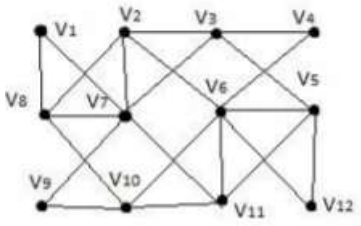
```
                cout << a[i][j] << " ";
        cout << endl;
    }

        for (int i = 0; i < vertex; i++)
    {
        d[i] = maximalNumber;
        visited[i] = 1;
    }

    int start, finish;
    cout << "From vertex :";

    cin >> start;
    start--;
    cout << "To: ";
    cin >> finish;
    finish--;
    int begin_index = start;
    d[begin_index] = 0;


    do {
        minindex = maximalNumber;
        min = maximalNumber;
        for (int i = 0; i < vertex; i++)
        {
            if ((visited[i] == 1) && (d[i] < min))
            {
                min = d[i];
                minindex = i;
            }
        }
        if (minindex != maximalNumber)
        {
            for (int i = 0; i < vertex; i++)
            {
                if (a[minindex][i] > 0)
                {
                    temp = min + a[minindex][i];
                    if (temp < d[i])
                    {
                        d[i] = temp;
                    }
                }
            }
            visited[minindex] = 0;
        }
    } while (minindex < maximalNumber);


    cout << "Minimal ways to vertex: " << endl;
    for (int i = 0; i < vertex; i++)cout << d[i] << " ";


    bool flag = false;
    for (int i = 0; i < vertex; i++)if (d[i] != 0 && d[i] != maximalNumber)flag =
true;

    if (flag) {

        int* ver = new int[vertex];
        int end = finish;
        ver[0] = end + 1;
        int k = 1;
```

```
		int weight = d[end];

		while (end != begin_index)
		{
			for (int i = 0; i < vertex; i++)
				if (a[end][i] != 0)
				{
					int temp = weight - a[end][i];
					if (temp == d[i])
					{
						weight = temp;
						end = i;
						ver[k] = i + 1;
						k++;
					}
				}
		}


		cout << endl << "Print minimal way" << endl;
		for (int i = k - 1; i >= 0; i--)cout << ver[i] << " ";



	}
	else {
		cout << "There isnt such way";
	}
	return 0;
}
```

```
From vertex :1
To: 30
Minimal ways to vertex:
0 6 7 8 11 14 4 8 9 10 12 18 9 9 11 13 13 20 14 10 13 16 15 20 16 12 14 17 18 24
Print minimal way
1 2 8 9 10 11 17 23 29 30
```

Завдання № 8

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



Флері:V2->V8->V1->V7->V2->V3->V5->V12->V6->V11->V10->V8->V7->V9->V10->V6->V5->V11->V7->V3->V4->V6->V2

1)V2-V8
2)V8-V1
3)V1-V7
4)V7-V2
5)V2-V3
6)V3-V5
8)V5-V12
9)V12-V6
10) V6-V11
11) V11-V10
12) V10-V8
13) V8-V7
14) V7-V9
15) V10-V6
16) V6-V5
17) V11-V5
18)V11-V7
19) V7-V3
20)V3-V4
21)V6-V4
22)V6-V2

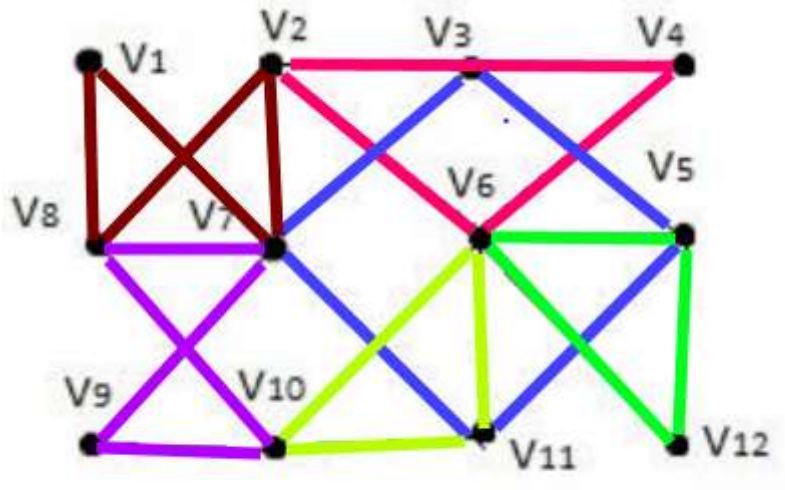
Алгоритм Флері

```
#include<iostream>
#include<vector>
using namespace std;
const int n = 12;
int graph[n][n] = {
{0,0,0,0,0,0,1,1,0,0,0,0},
{0, 0,1,0,0,1,1,1,0,0,0,0},
{0,1,0,1,1,0,1,0,0,0,0,0},
{0,0,1,0,0,1,0,0,0,0,0,0},
{0,0,1,0,0,1,0,0,0,0,1,1},
{0,1,0,1,1,0,0,0,0,1,1,1},
{0,1,1,0,0,0,0,1,1,0,1,0},
{1,1,0,0,0,0,1,0,0,1,0,0},
{0,0,0,0,0,0,1,0,0,1,0,0},
{0,0,0,0,0,1,0,1,1,0,1,0},
{0,0,0,0,1,1,1,0,0,0,0,0},
{0,0,0,0,1,1,0,0,0,0,0,0}
};
int oddGraph[n][n];
int startVertex() {
	for (int i = 0; i < n; i++) {
		int deg = 0;
		for (int j = 0; j < n; j++) {
			if (oddGraph[i][j])
				deg++;
		}
		if (deg % 2 != 0)
			return i;
	}
```

```
	return 0;
}
bool check(int u, int v) {
	int deg = 0;
	for (int i = 0; i < n; i++)
		if (oddGraph[v][i])
			deg++;
	if (deg > 1) {
		return false;
	}
	return true;
}
int edgeCount() {
	int count = 0;
	for (int i = 0; i < n; i++)
		for (int j = i; j < n; j++)
			if (oddGraph[i][j])
				count++;
	return count;
}
void func(int start) {
	static int edge = edgeCount();
	for (int v = 0; v < n; v++) {
		if (oddGraph[start][v]) {
			if (edge <= 1 || !check(start, v)) {
				cout << start << "-" << v << endl;
				oddGraph[start][v] = oddGraph[v][start] = 0;
				edge--;
				func(v);
			}
		}
	}
}
int main() {
	for (int i = 0; i < n; i++)
		for (int j = 0; j < n; j++)
			oddGraph[i][j] = graph[i][j];
	cout << "Result: " << endl;
	func(startVertex());
	return 0;
                                                }
```

```
Result:
6-1
1-2
2-3
3-5
5-1
1-7
7-0
0-6
6-2
2-4
4-5
5-9
9-7
7-6
6-8
8-9
9-10
10-4
4-11
11-5
5-10
10-6
```

Елементарних циклів




```
#include <iostream>
#include<vector>
#include<stack>
#include<algorithm>
#include<list>

using namespace std;

vector < list<int>>graf;
vector<int>step;
stack<int>head, tail;


int main()
```

```
{
	int n, a, x, y;
	cout << "Number of verticles=" << " ";
	cin >> n;
	cout << "Number of edges=" << " ";
	cin >> a;
	graf.resize(n + 1);
	step.resize(n + 1);
	for (; a--;)
	{	cin >> x >> y;
	graf[x].push_back(y);
	graf[y].push_back(x);
	++step[x];
	++step[y];
    }
	if (any_of(step.begin() + 1, step.end(), [](int i) {return i & 1; }))
		cout << "-1";
	else
	{
		head.push(1);
		while (!head.empty())
		{
			while (step[head.top()])
			{
				int v = graf[head.top()].back();
				graf[head.top()].pop_back();
				graf[v].remove(head.top());
				--step[head.top()];
				head.push(v);
				--step[v];
			}
			while (!head.empty() && !step[head.top()])
			{
				tail.push(head.top());
				head.pop();

			}
		}
		cout << "cycle-";
		while (!tail.empty())
		{
			cout << tail.top()<<' ';
			tail.pop();

		}
	}
}
```

Завдання №9

Спростити формули (привести їх до скороченої ДНФ).

$$(x \vee \bar{z})(\bar{y} \vee z)$$

$X\bar{Y} \vee X\bar{Z} \vee (\bar{Y}\bar{Y}) \vee (\bar{Y}\bar{Z}) = X\bar{Y} \vee X\bar{Z} \vee \bar{Y} \vee (\bar{Y}\bar{Z})$

$X\bar{Y} \vee \bar{Y} = \bar{Y}$ (закон поглинання)

$X\bar{Z} \vee \bar{Y} \vee (\bar{Y}\bar{Z})$

$\bar{Y} \vee (Y\bar{Z}) = \bar{Y}$ (закон поглинання)

$X\bar{Z} \vee \bar{Y}$ -ДНФ