МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних наук та інформаційних технологій

Кафедра систем штучного інтелекту

Розрахункова робота

З дисципліни «Дискретна математика»

Виконав:

Студент групи КН-113

Пантьо Ростислав

Викладач:

Мельникова Н.І.

Львів 2019

Напишіть алгоритм:

62. Краскала знаходження найменшого остового дерева.

1) Спочатку ребра сортують за зростанням ваги.
2) Додають найменше ребро в дерево.
3) Зі списку ребер із найменшою вагою вибирають таке нове ребро, щоб одна з його вершин належала дереву, а інша — ні.
4) Це ребро додають у дерево і знову переходять до кроку 3.
5) Робота закінчується, коли всі вершини будуть у дереві.

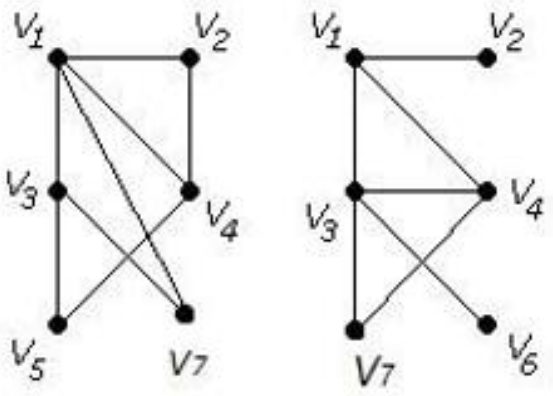
ІНДИВІДУАЛЬНІ ЗАВДАННЯ

Варіант №9

Завдання №1

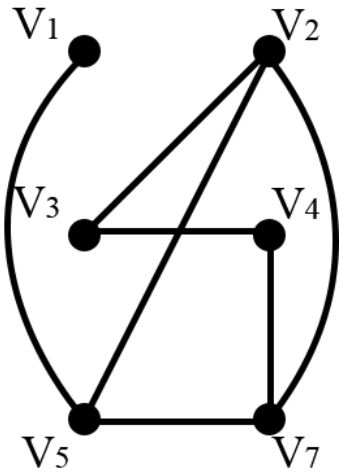
Виконати наступні операції над графами:

1) знайти доповнення до першого графу,
2) об’єднання графів,
3) кільцеву суму G1 та G2 (G1+G2),
4) розмножити вершину у другому графі,
5) виділити підграф А - що складається з 3-х вершин в G1
6) добуток графів.

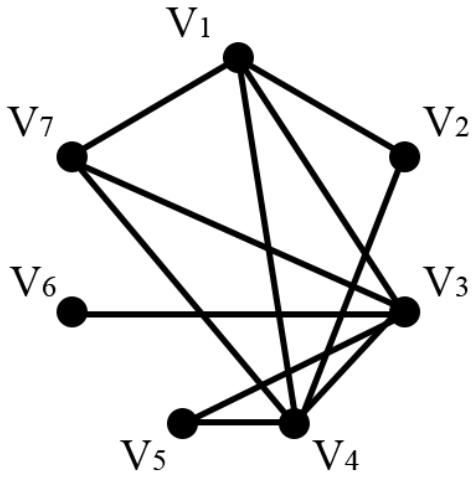


Розв’язання:

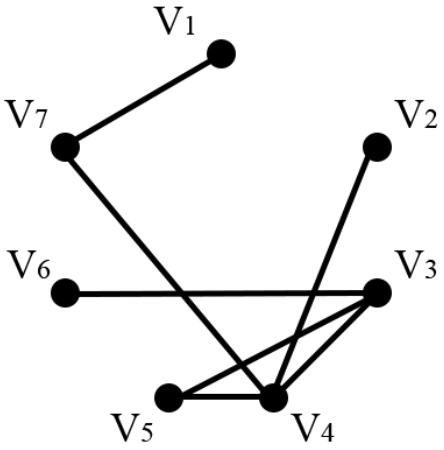
1) Доповнення до першого графу



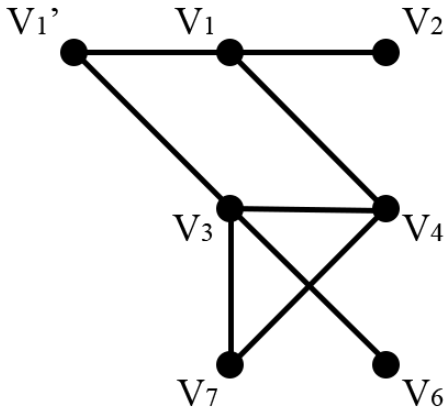
2) Об’єднання графів



3) кільцева сума G1 та G2 (G1+G2)

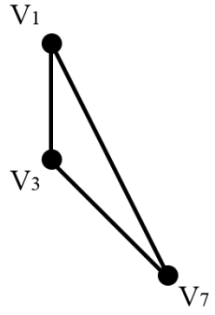


4) Розмноження вершини у другому графі

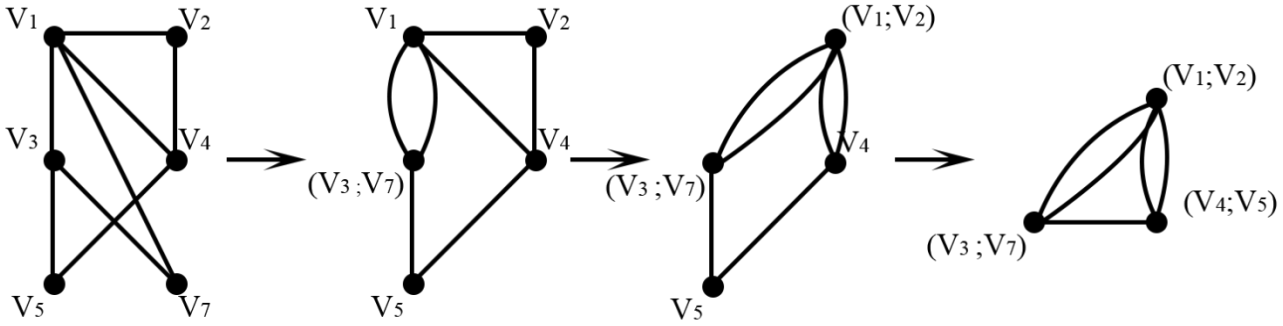


5) Виділення підграфа А - що складається з 3-х вершин в G1

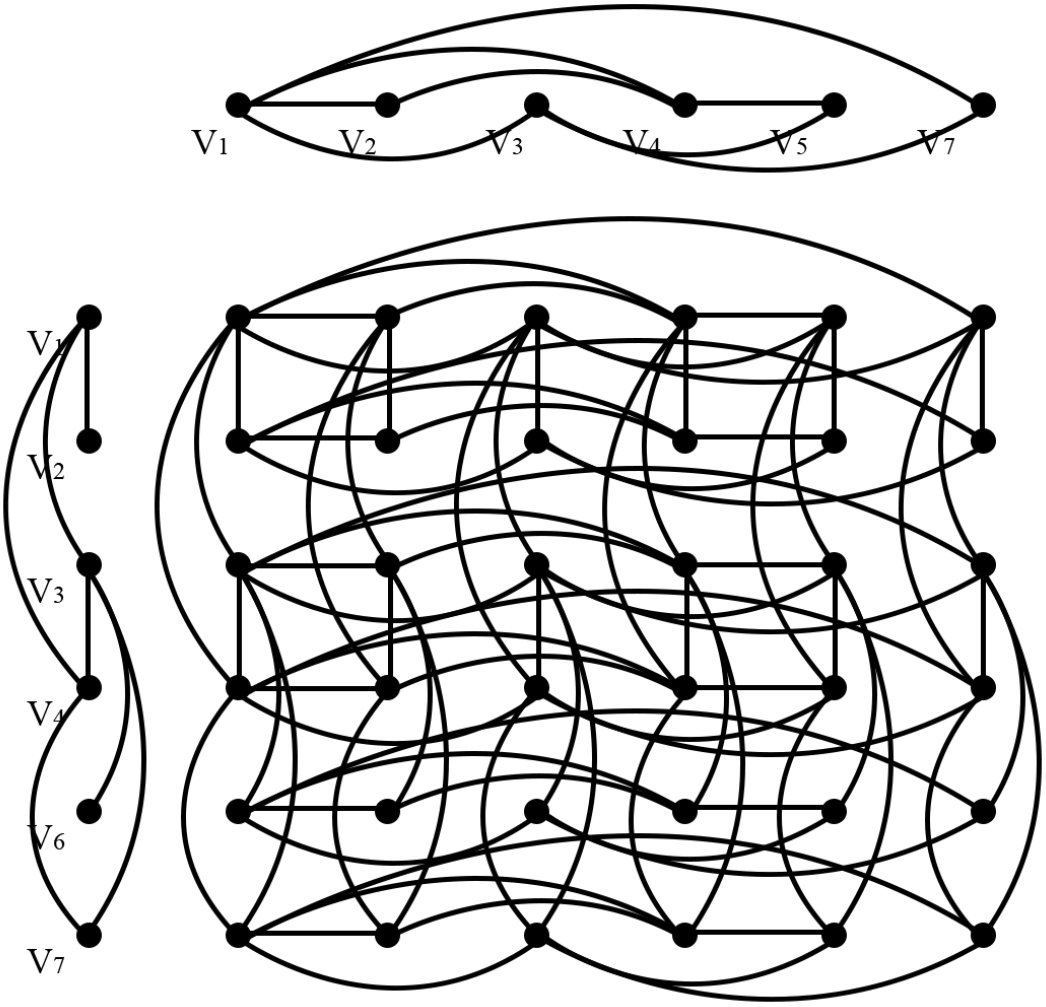
Підграф А:



Стягування:

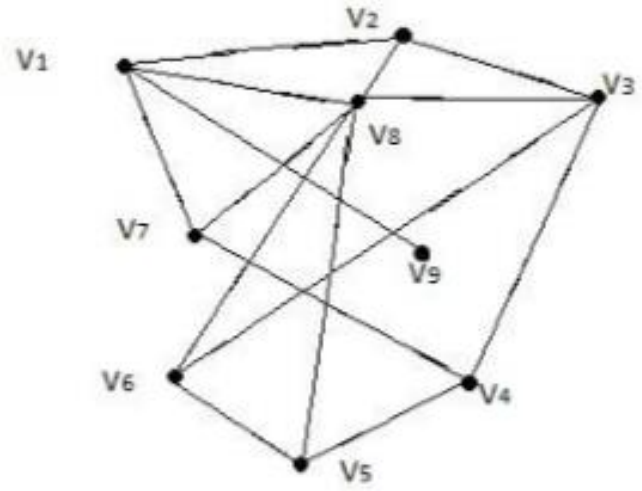


6) Добуток графів



Завдання №2

Скласти таблицю суміжності для орграфа.

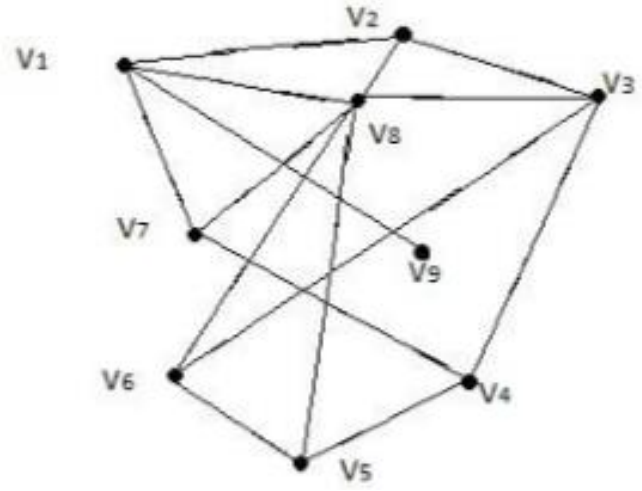


Розв’язання:

	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9
V_1	0	1	0	0	0	0	1	1	1
V_2	1	0	1	0	0	0	0	1	0
V_3	0	1	0	1	0	1	0	1	0
V_4	0	0	1	0	1	0	1	0	0
V_5	0	0	0	1	0	1	0	1	0
V_6	0	0	1	0	1	0	0	1	0
V_7	1	0	0	1	0	0	0	1	0
V_8	1	1	1	0	1	1	1	0	0
V_9	1	0	0	0	0	0	0	0	0

Завдання №3

Для графа з другого завдання знайти діаметр.



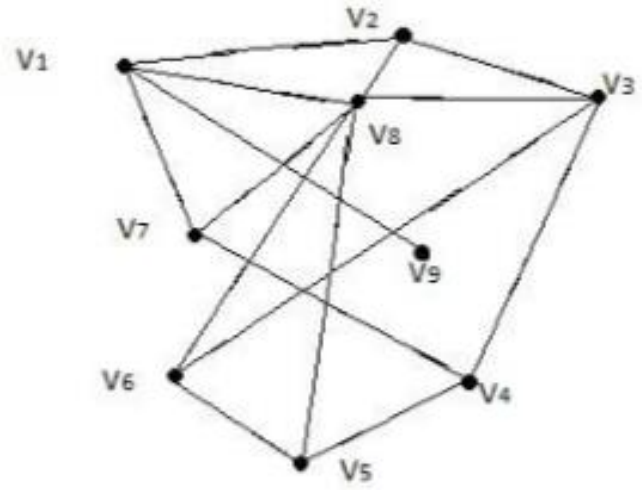
Розв’язання:

№	маршрут	точка	1	2	3	4	5	6	7	8	9
1	9	9	1	∞	∞	∞	∞	∞	∞	∞	-
2	9, 1	1	-	2	∞	∞	∞	∞	2	2	-
3	9, 1, 2	2	-	-	3	∞	∞	∞	2	2	-
4	9, 1, 2, 7	7	-	-	3	3	∞	∞	-	2	-
5	9, 1, 2, 7, 8	8	-	-	3	3	3	3	-	-	-

Отже, діаметр =3.

Завдання №4

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



Розв’язання:

Вершина	DFS	Вміст стеку
9	1	9
1	2	91
2	3	912
3	4	9123
4	5	91234
5	6	912345
6	7	9123456
8	8	91234568
7	9	912345687
-	-	91234568
-	-	9123456
-	-	912345
-	-	91234
-	-	9123
-	-	912
-	-	91
-	-	9

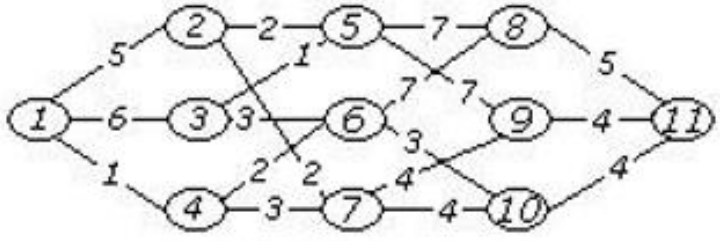
Програма:

```
#include <iostream>
using namespace std;
const int n=9;
int i, j;
int graph[n][n]=
{
    0, 1, 0, 0, 0, 0, 1, 1, 1,
    1, 0, 1, 0, 0, 0, 0, 1, 0,
    0, 1, 0, 1, 0, 1, 0, 1, 0,
    0, 0, 1, 0, 1, 0, 1, 0, 0,
    0, 0, 0, 1, 0, 1, 0, 1, 0,
    0, 0, 1, 0, 1, 0, 0, 1, 0,
    1, 0, 0, 1, 0, 0, 0, 1, 0,
    1, 1, 1, 0, 1, 1, 1, 0, 0,
    1, 0, 0, 0, 0, 0, 0, 0, 0
};
bool *visited=new bool[n];
//пошук вглиб
void DFS(int st)
{
    int r;
    cout<<st+1<<" ";
    visited[st]=true;
    for (r=0; r<n; r++)
    if ((graph[st][r]!=0) && (!visited[r]))
    DFS(r);
}

int main()
{
    setlocale(LC_ALL, "Ukr");
    int start;
    cout<<" adjacency matrix : "<<endl;
    for (i=0; i<n; i++)
    {
    visited[i]=false;
    for (j=0; j<n; j++)
    cout<<" "<<graph[i][j];
    cout<<endl;
    }
    cout<<" The first vertex >> "; cin>>start;
    bool *vis=new bool[n];
    cout<<"Answer : ";
    DFS(start-1);
    delete []visited;
}
```

```
 adjacency matrix :
 0 1 0 0 0 0 1 1 1
 1 0 1 0 0 0 0 1 0
 0 1 0 1 0 1 0 1 0
 0 0 1 0 1 0 1 0 0
 0 0 0 1 0 1 0 1 0
 0 0 1 0 1 0 0 1 0
 1 0 0 1 0 0 0 1 0
 1 1 1 0 1 1 1 0 0
 1 0 0 0 0 0 0 0 0
 The first vertex >> 9
Answer : 9 1 2 3 4 5 6 8 7
```

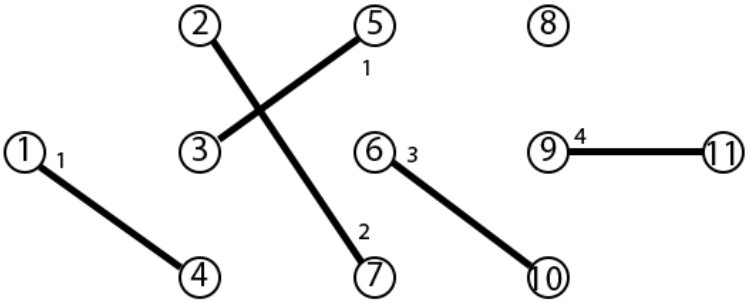
Завдання №5

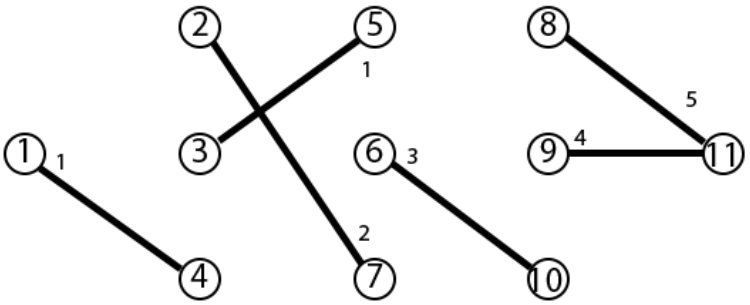
Знайти двома методами (Краскала і Прима) мінімальне остове дерево графа.

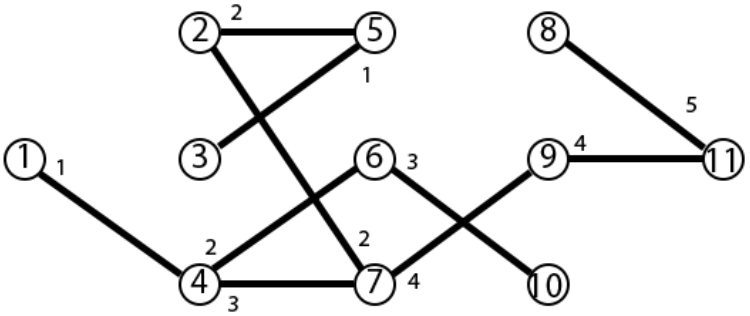


Розв’язання:

1) Методом Краскала



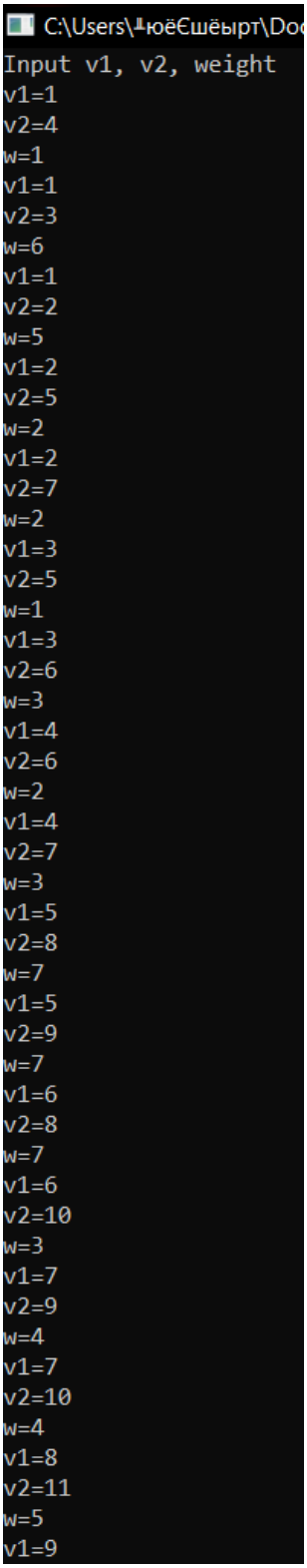


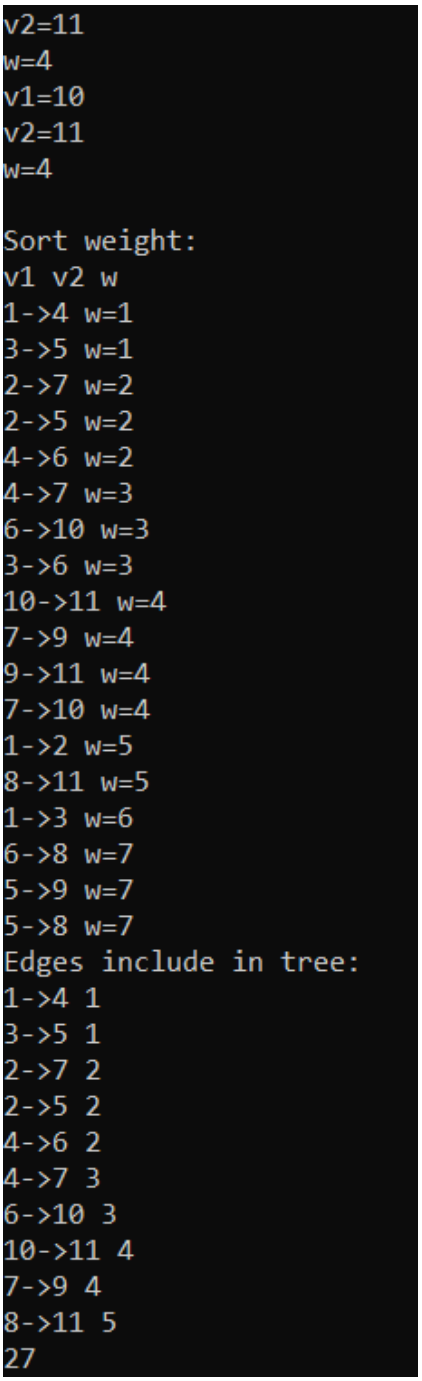


Програма:

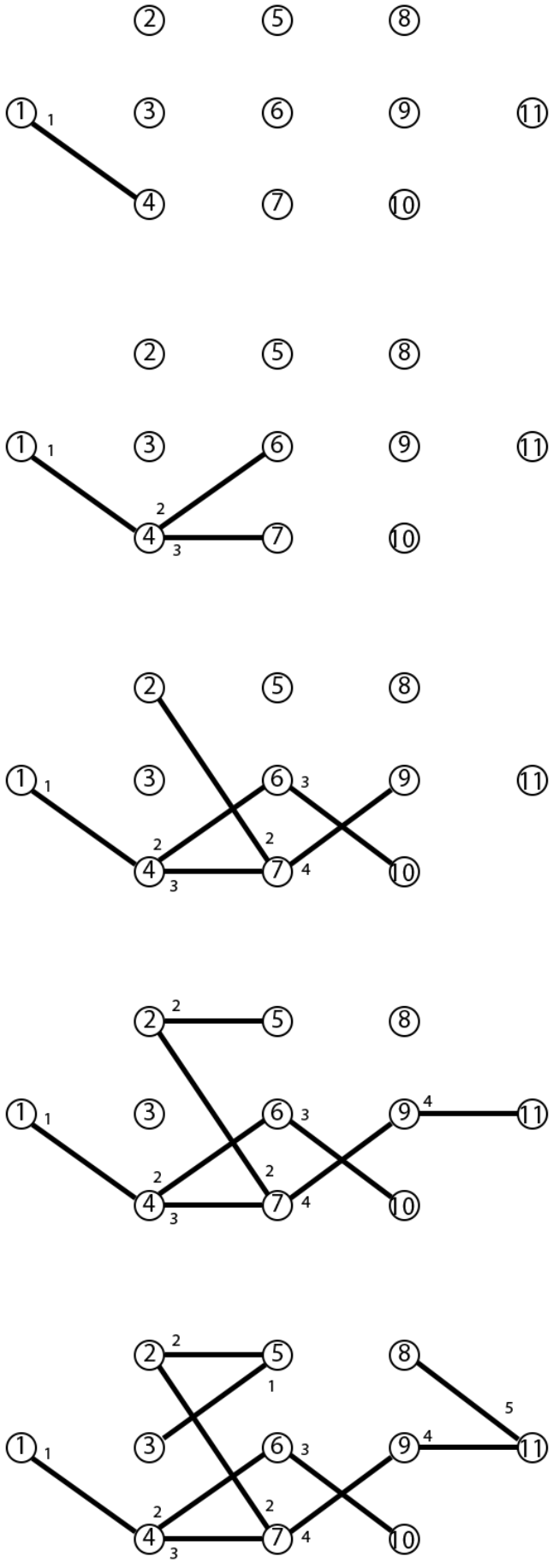
```
#include <iostream>
#include <stdlib.h>

using namespace std;

#define NV 12
#define NE 18



struct edge {
  int v1, v2;
  int w;
} edges[NE];



int comp(edge* a, edge* b)
{
  if (a->w > b->w)
    return 1;
  if (a->w < b->w)
    return -1;


}

int nodes[NV];
int last_n;


int getColor(int n) {
  int c;
  if (nodes[n] < 0)
    return nodes[last_n = n];
  c = getColor(nodes[n]);
  nodes[n] = last_n;
  return c;
}

int main() {

  for (int i = 0; i < NV; i++) {
    nodes[i] = -1 - i;
  }
  cout << "Input v1, v2, weight" << endl;
  for (int i = 0; i < NE; i++) {
```

```
48      cout << "v1=";
49      cin >> edges[i].v1;
50      cout << "v2=";
51      cin >> edges[i].v2;
52      cout << "w=";
53      cin >> edges[i].w;
54    }
55
56
57
58    qsort(edges, NE, sizeof(edge), (int (*)(const void*, const void*))comp);
59
60    cout << " " << endl;
61    cout << "Sort weight:" << endl;
62    cout << "v1 v2 w" << endl;
63    for (int i = 0; i < NE; i++) {
64      cout << edges[i].v1 << "->" << edges[i].v2 << " " << "w=" << edges[i].w << endl;
65    }
66
67    int sum = 0;
68    cout << "Edges include in tree:" << endl;
69    for (int i = 0; i < NE; i++) {
70      int c2 = getColor(edges[i].v2);
71      if (getColor(edges[i].v1) != c2) {
72        nodes[last_n] = edges[i].v2;
73        cout << edges[i].v1 << "->" << edges[i].v2 << " " << edges[i].w << endl;
74        sum += edges[i].w;
75      }
76    }
77
78    cout << sum;
79    cout << endl;
80    return 0;
81  }
```


C:\Users\Ююёєшёырт\Doc
Input v1, v2, weight
v1=1
v2=4
w=1
v1=1
v2=3
w=6
v1=1
v2=2
w=5
v1=2
v2=5
w=2
v1=2
v2=7
w=2
v1=3
v2=5
w=1
v1=3
v2=6
w=3
v1=4
v2=6
w=2
v1=4
v2=7
w=3
v1=5
v2=8
w=7
v1=5
v2=9
w=7
v1=6
v2=8
w=7
v1=6
v2=10
w=3
v1=7
v2=9
w=4
v1=7
v2=10
w=4
v1=8
v2=11
w=5
v1=9


v2=11
w=4
v1=10
v2=11
w=4
Sort weight:
v1 v2 w
1->4 w=1
3->5 w=1
2->7 w=2
2->5 w=2
4->6 w=2
4->7 w=3
6->10 w=3
3->6 w=3
10->11 w=4
7->9 w=4
9->11 w=4
7->10 w=4
1->2 w=5
8->11 w=5
1->3 w=6
6->8 w=7
5->9 w=7
5->8 w=7
Edges include in tree:
1->4 1
3->5 1
2->7 2
2->5 2
4->6 2
4->7 3
6->10 3
10->11 4
7->9 4
8->11 5
27

2) Методом Прима



Програма:

```cpp
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n,i,j,k;
    cin>>n;
    int a[n][n];
    for (i=0;i<n;i++)
    {
        for (j=0;j<n;j++)
        {
            cin>>a[i][j];
        }
    }
    cout<<endl;
    int numb[n]{0};
    int numbsiz=1;
    int min_n,i_min,j_min;
    while (numbsiz<n)
    {
        min_n=1000;
        for (k=0;k<numbsiz;k++)
        {
            for (i=0;i<n;i++)
            {
                if (a[numb[k]][i]<min_n&&a[numb[k]][i]!=0)
                {
                    min_n=a[numb[k]][i];
                    i_min=i;
                    j_min=numb[k];
                }
            }
        }
        a[j_min][i_min]=0;
        a[i_min][j_min]=0;
        bool itrue=0;
        for (i=0;i<numbsiz;i++) if (i_min==numb[i]) itrue=1;
        if (itrue==0)
        {
            numbsiz++;
            numb[numbsiz-1]=i_min;
            cout<<"( "<<j_min+1<<", "<<i_min+1<<" )"<<';';
        }
    }
```

```
11
0 5 6 1 0 0 0 0 0 0 0
5 0 0 0 2 0 2 0 0 0 0
6 0 0 0 1 3 0 0 0 0 0
1 0 0 0 0 2 3 0 0 0 0
0 2 1 0 0 0 0 7 7 0 0
0 0 3 2 0 0 0 7 0 3 0
0 2 0 3 0 0 0 0 4 4 0
0 0 0 0 7 7 0 0 0 0 5
0 0 0 0 7 0 4 0 0 0 4
0 0 0 0 0 3 4 0 0 0 4
0 0 0 0 0 0 0 5 4 4 0

( 1, 4 );( 4, 6 );( 4, 7 );( 7, 2 );( 2, 5 );( 5, 3 );( 6, 10 );( 7, 9 );( 10, 11 );( 11, 8 );
```

Завдання №6

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

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

Розв’язання:

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

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

	2	**3**	**4**	**5**	**6**
2	∞	4	3	2	1
3	4	∞	4	5	6
4	3	4	∞	1	5
5	2	5	1	∞	5
6	1	6	5	5	∞

	2	**3**	**5**	**6**
2	∞	4	2	1
3	4	∞	5	6
5	2	5	∞	5
6	1	6	5	∞

	2	**3**	**6**
2	∞	4	1
3	4	∞	6
6	1	6	∞

	3	**6**
3	∞	6
6	6	∞

Отже маршрут:

(1) → (8) → (7) → (4) → (5) → (2) → (6) → (3).

Програма:

```
#include <iostream>
#include <vector>

using namespace std;

int coun = 0, min_way = INT_MAX;

bool check(vector<int> V, int Nod)
{
	for (auto i = V.begin(); i != V.end(); i++)
	{
		if (*i == Nod) return false;
	}
	return true;
}

int minim(vector<int>* V, int** arr, int n, int i)
{
	int nmin = INT_MAX;
	int j;
	for (j = 0; j < n; j++)
	{
		if (arr[i][j] < nmin && arr[i][j] != 0 && check((*V), j)) nmin = arr[i][j];
	}
	return nmin;
}

void depth(vector<int>* V, int** arr, int n, int pos, vector<int>* VCON)
{
	int nmin, i,j, k;
	for (i = pos, k = 0; k < 1; i++, k++)
	{
		nmin = minim(V, arr, n, i);
		for (j = 0; j < n; j++)
		{
			if (arr[i][j] == nmin && check((*V), j))
			{
				(*V).push_back(j);
				depth(V, arr, n, j, VCON);
			}
		}

		if (V->size() == n)
		{
			(*V).push_back((*V)[0]);
			coun = 0;
			int l;
			for (l = 1; l <= n; l++)
			{
				coun += arr[(*V)[l - 1]][(*V)[l]];
			}
			for (auto it = (*V).begin(); it != (*V).end(); it++)
				cout << *it + 1 << ' ';
			cout << " = " << " " << endl;
			if (min_way == coun)
			{
				for (int op = 0; op <= n; op++)
				{
					(*VCON).push_back((*V)[op]);
					(*VCON).push_back(coun);
				}
			}
			else if (min_way > coun)
			{
				(*VCON).clear();
```

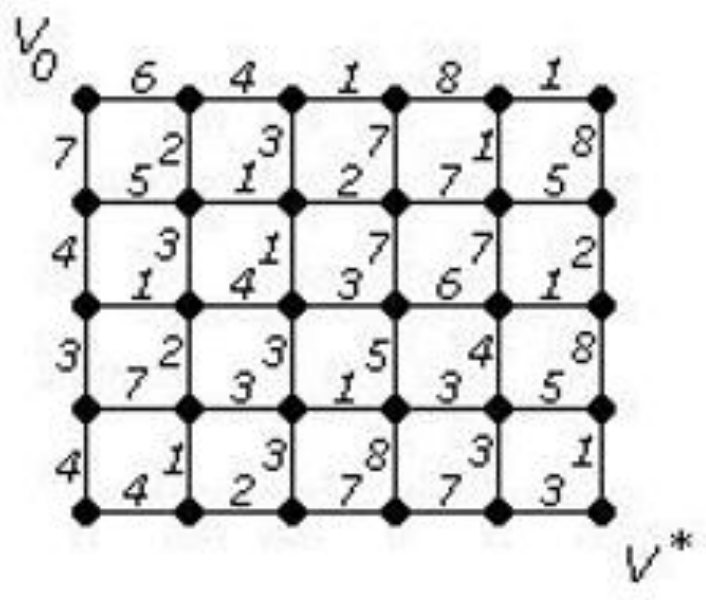
```
                        for (int op = 0; op <= n; op++)
                        {
                                (*VCON).push_back((*V)[op]);
                                (*VCON).push_back(coun);
                                min_way = coun;
                        }
                }
                V->pop_back();
        }
    }
}
int main()
{
    int n, i, j;
    cout << "Enter number of nodes: ";
    cin >> n;
    int** arr = new int* [n];
    for (i = 0; i < n; i++)
    {
        arr[i] = new int[n];
    }
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            cin >> arr[i][j];
        }
    }
    vector<int> V;
    vector<int> VCON;
    cout << endl;
    for (i = 0; i < n; i++)
    {
        V.clear();
        V.push_back(i);
        depth(&V, arr, n, i, &VCON);
    }
    cout << "\n\n\n" << "rez" << "\n\n";

    for (i = 1; i <= VCON.size(); i++)
    {
        if (i != 0 && i % (n + 2) == 0)
            cout << " (" << VCON[i - 1] << ") " << endl;
        else
            cout << VCON[i - 1] + 1 << " -> ";
    }
}
```

```
Enter number of nodes: 8
9 5 5 3 3 4 4 1
5 9 4 3 2 1 4 6
5 4 9 4 5 6 5 5
3 3 4 9 1 5 1 7
3 2 5 1 9 5 5 2
4 1 6 5 5 9 7 3
4 4 5 1 5 7 9 2
1 6 5 7 2 3 2 9
```

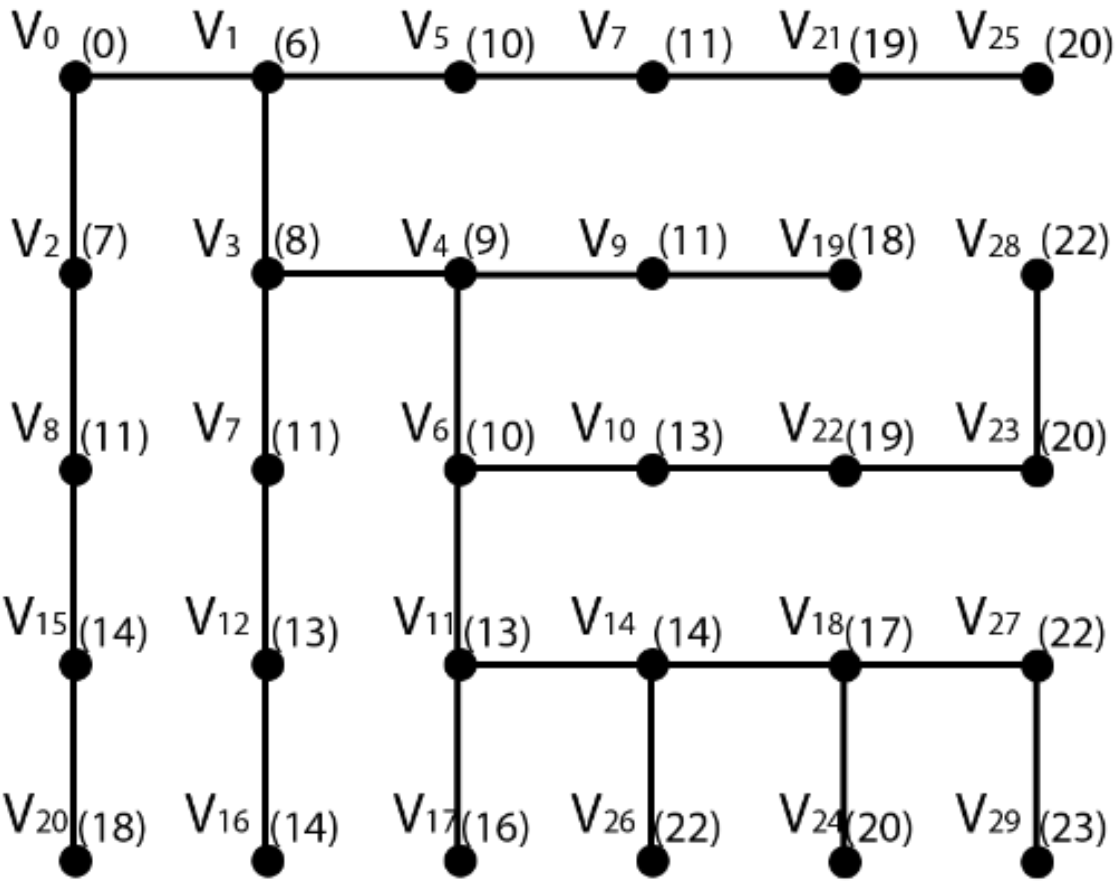
```
rez

5 -> 20 -> 4 -> 20 -> 7 -> 20 -> 8 -> 20 -> 1 ->  (19)
6 -> 20 -> 2 -> 20 -> 3 -> 20 -> 5 -> 20 -> 5 ->  (19)
4 -> 20 -> 7 -> 20 -> 8 -> 20 -> 1 -> 20 -> 6 ->  (19)
2 -> 20 -> 3 -> 20 -> 5 -> 20 -> 5 -> 20 -> 4 ->  (19)
7 -> 20 -> 8 -> 20 -> 1 -> 20 -> 6 -> 20 -> 2 ->  (19)
```

Завдання №7

За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі між парою вершинV_0 і V^*.



Розв’язання:



Програма:

```
#include <bits/stdc++.h>
using namespace std;
ifstream fin;
int main()
{
    int n,i,j;
    cout<<"Input the number of vertexes:"<<endl;
    cin>>n;
    int a[n][n];
    fin.open("тест.txt");
    for (i=0;i<n;i++)
    {
        for (j=0;j<n;j++)
        {
            fin>>a[i][j];
        }
    }
    int vertindex[n]{-1};
    int numbvectors=0;
    for (i=0;i<n;i++)
    {
        for (j=0;j<n;j++)
        {
            if (a[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]+a[i][j]<nmin&&a[i][j]!=0)
                    {
                        nmin=vert[i]+a[i][j];
                        imin=i;
                        jmin=j;
                    }
                }
            }
        }
        if (vert[jmin]>nmin)
        {
            vert[jmin]=nmin;
            vertindex[jmin]=imin;
        }
        vertdot[jmin]=1;
        a[imin][jmin]=a[jmin][imin]=0;
        numbvectors-=2;
    }

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

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

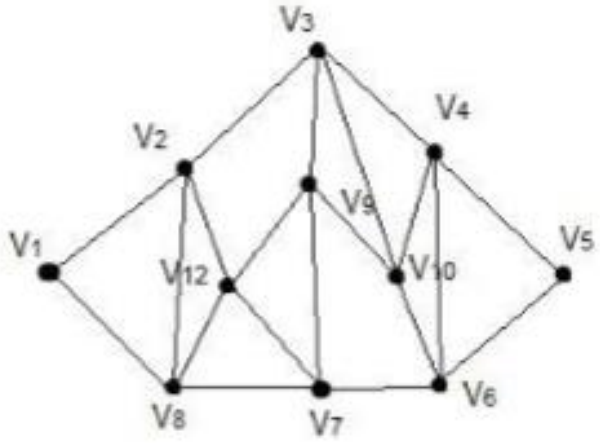
Номерація точок проводиться по рядках

```
Input the number of vertexes:
30
Input finish vertex: 29

Weight = 23
The way from start to finish:
0->1->7->8->14->20->21->22->23->29
Process returned 0 (0x0)   execution time : 5.001 s
Press any key to continue.
```

Завдання №8

Знайти ейлеровий цикл в ейлеровому графі двома методами:

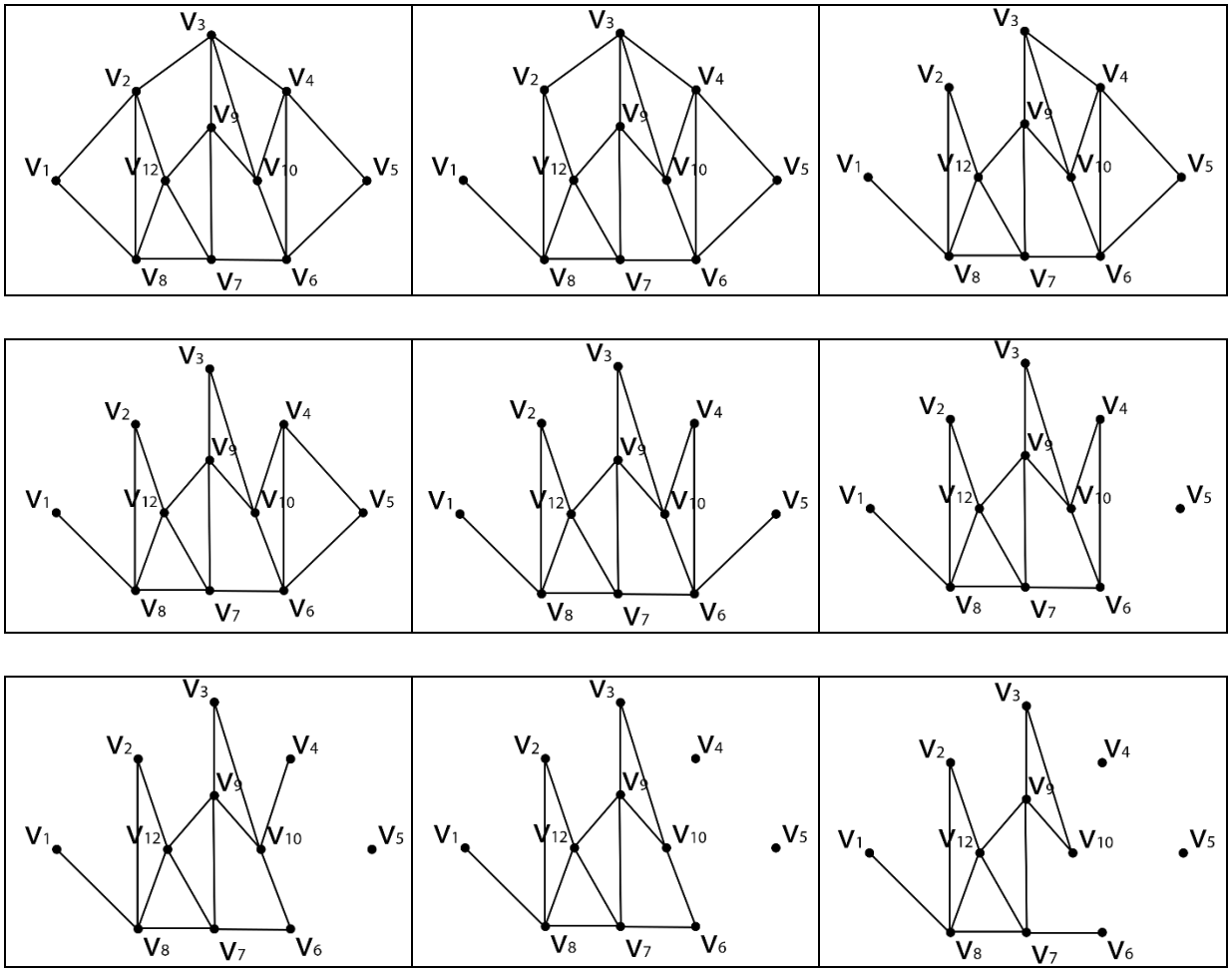
a. Флері.
b. Елементарних циклів.

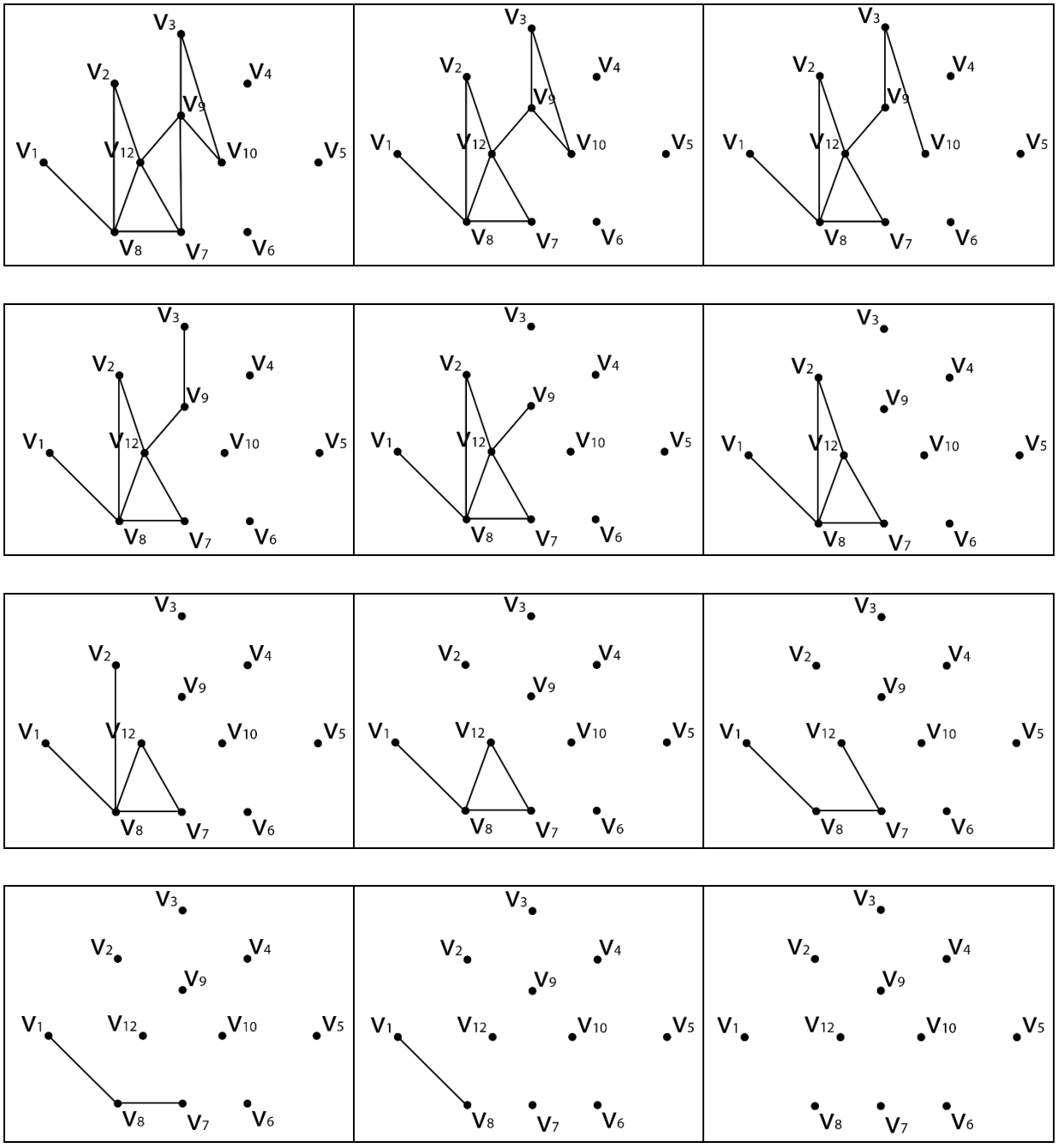


Розв’язання:

Оскільки степінь кожної верншини цього графа є парним числом то даний граф є Ейлеровим графом.

a. Метод Флері:



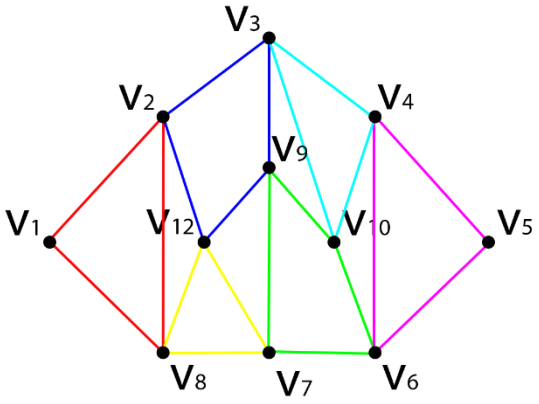


Програма:

```cpp
#include <bits/stdc++.h>
using namespace std;

void Search(int v, vector <vector<int> >* G,int N)
{
    int i;
    for (i=0;i < N;i++)
        if ((*G)[v][i])
    {
        (*G)[v][i]=(*G)[i][v]=0;
        Search(i,G,N);
    }
    cout<<v+1<<' ';
}

int main()
{
    int N=0,i,j;
    cout<<"Enter N:";
    cin>>N;
    vector <vector<int> > G(N, vector<int>(N));
    for (i=0;i<N;i++)
    {
        for (j=0;j<N;j++)
        {
            cin>>G[i][j];
        }
    }
    int Nodpow, p,q,sum;
    Nodpow=1;
    for(p=0;p<N;p++)
    {
        sum=0;
        for (q=0;q<N;q++)
        {
            sum+=G[p][q];
        }
        if (sum%2) Nodpow=0;
    }
    cout<<endl;
    if (Nodpow)
    {
        Search(0,&G,N);
    }
    else cout<<"Not Eulerian graph\n";
    cout<<endl;
}
```

```
Enter N:11
0 1 0 0 0 0 0 1 0 0 0
1 0 1 0 0 0 0 1 0 0 1
0 1 0 1 0 0 0 0 1 1 0
0 0 1 0 1 1 0 0 0 1 0
0 0 0 1 0 1 0 0 0 0 0
0 0 0 1 1 0 1 0 0 1 0
0 0 0 0 0 1 0 1 1 0 1
1 1 0 0 0 0 1 0 0 0 1
0 0 1 0 0 0 1 0 0 1 1
0 0 1 1 0 1 0 0 1 0 0
0 1 0 0 0 0 1 1 1 0 0

1 8 11 7 8 2 11 9 10 6 7 9 3 10 4 6 5 4 3 2 1
```

b. Метод елементарних циклів:



Програма:

```
#include <vector>
#include <stack>
#include <algorithm>
#include <list>
using namespace std;

    vector < list<int> > graph;
    vector <int> deg;
    stack<int> head,tail ;


int main()
{
    int n, a, x,y ;
    cin >> n >> a;
    graph.resize(n+1);
    deg.resize(n+1);
    for(; a--;)
    {
        cin >> x >> y;
        graph[x].push_back(y);
        graph[y].push_back(x);
        ++deg[x];
        ++deg[y];
    }
   head.push(1);
    while(!head.empty())
    {
        while(deg[head.top()])
        {
            int v = graph[head.top()].back();
            graph[head.top()].pop_back();
            graph[v].remove(head.top());
            --deg[head.top()] ;
            head.push(v);
            --deg[v];
        }
        while(!head.empty()&&!deg[head.top()])
        {
            tail.push( head.top());
            head.pop();
        }
    }
    while(!tail.empty())
    {
        cout << tail.top() <<' ';
        tail.pop();
```

```
11
20
1 2
1 8
2 8
2 3
3 9
9 11
8 11
7 11
7 8
2 11
3 4
4 10
3 10
9 10
7 9
6 7
6 10
4 5
5 6
4 6
1 8 7 6 4 5 6 10 9 7 11 2 3 10 4 3 9 11 8 2 1
```

Завдання №9

Спростити формули (привести їх до скороченого ДНФ).

$$(x \to y) \cdot (y \to z) \to (x \to z)$$

Розв’язання:

$$\big((\boldsymbol{x \to y}) \times (\boldsymbol{y \to z})\big) \to (\boldsymbol{x \to z}) = \overline{\big((\boldsymbol{x \to y}) \times (\boldsymbol{y \to z})\big)} + (\boldsymbol{x \to z}) =$$

$$= (\overline{(\boldsymbol{x \to y})} + \overline{(\boldsymbol{y \to z})}) + (\boldsymbol{x \to z}) = (\boldsymbol{x} + \overline{\boldsymbol{y}}) + (\boldsymbol{y} + \overline{\boldsymbol{z}}) + (\overline{\boldsymbol{x}} + \boldsymbol{z}) = \boldsymbol{U}$$