МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”
Кафедра систем штучного інтелекту

Розрахункова робота

з дисципліни
<<Дискретна математика>>

Виконав:
студент групи КН-114
Микицький Назар
Викладач:
Мельникова Н.І

Варіант 7

1.**Виконати наступні операції над графами:**

1) знайти доповнення до першого графу,

2) об’єднання графів,

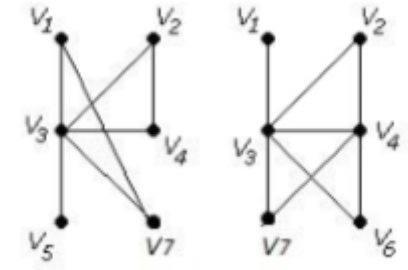
3) кільцеву сумму G1 та G2 (G1+G2),

4) розмножити вершину у другому графі,

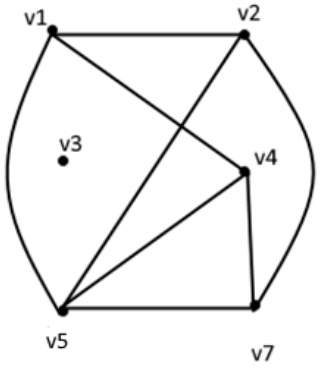
5) виділити підграф А - що скадається з 3-х вершин в G1

6) добуток графів.

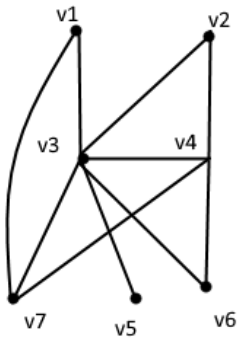
7)



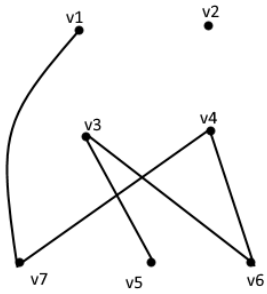
а) доповнення до першого графу,



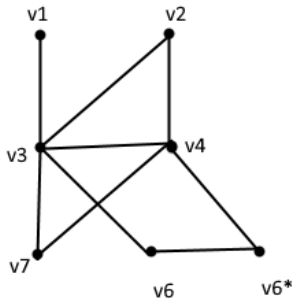
б) об’єднання графів,



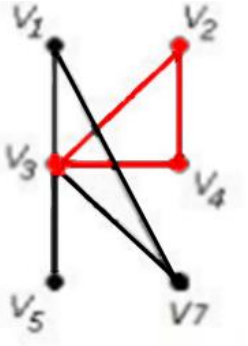
в) кільцеву сумма G1 та G2 (G1+G2),



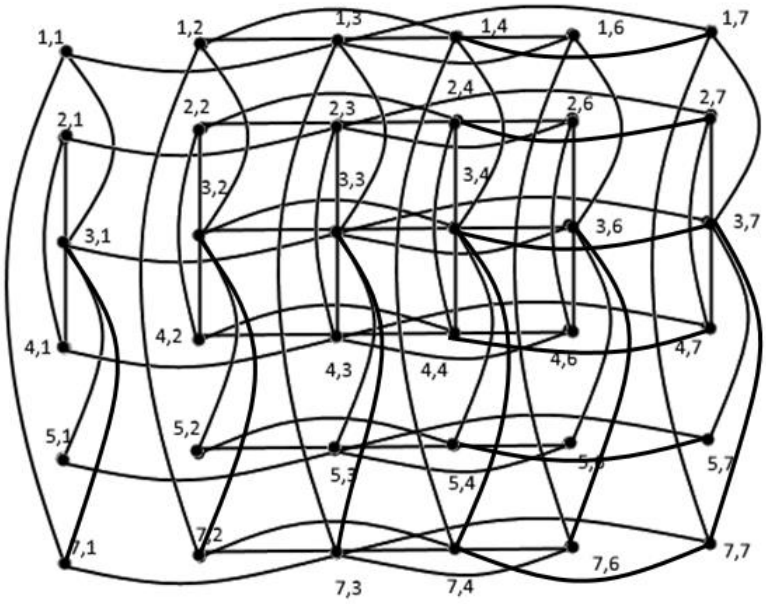
г) розмноження вершини у другому графі,



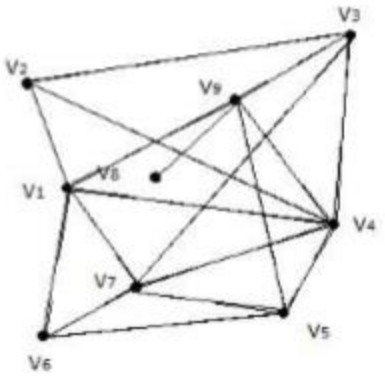
5) підграф А - що скадається з 3-х вершин в G1



6) добуток графів.



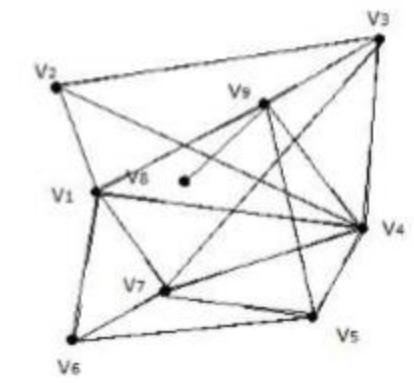
2**. Скласти таблицю суміжності для орграфа.**



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

3.**Діаметр графа**=3;

4.**Для графа з другого завдання виконати обхід дерева вглиб**



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

Вертаємось в вершину 4 і

4->9->8

```
Microsoft Visual Studio Debug Console
Матриця сум?жност? графа:
 0 1 0 1 0 1 1 0 1
 0 1 1 1 0 0 0 0 0
 0 1 0 1 0 0 1 0 1
 0 1 1 0 1 0 1 0 1
 0 0 0 1 0 1 1 0 1
 1 0 0 0 1 0 1 0 0
 1 0 1 1 1 1 0 0 0
 0 0 0 0 0 0 0 0 1
 1 0 1 1 1 0 0 1 0
Старт >> 1
Порядок обходу: 1 2 3 4 5 6 7 9 8
C:\Users\Lenovo\source\repos\Project6\Debug\Project6.exe (process 13300) exited with code 0.                    To automatically close the console when debuggin
g stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

```
#include <iostream>
using namespace std;
const int n = 9;
int i, j;
bool *visited = new bool[n];

int graph[n][n] =
{
                                //1 2 3 4 5 6 7 8 9
                                {0,1,0,1,0,1,1,0,1}, //1
                                {0,1,1,1,0,0,0,0,0}, //2
                                {0,1,0,1,0,0,1,0,1}, //3
                                {0,1,1,0,1,0,1,0,1}, //4
                                {0,0,0,1,0,1,1,0,1}, //5
                                {1,0,0,0,1,0,1,0,0}, //6
                                {1,0,1,1,1,1,0,0,0}, //7
                                {0,0,0,0,0,0,0,0,1}, //8
                                {1,0,1,1,1,0,0,1,0} //9
};

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);
}

void main()
{
        setlocale(LC_ALL, "Ukr");
        int start;
        cout << "Матриця суміжності графа: " << endl;
```

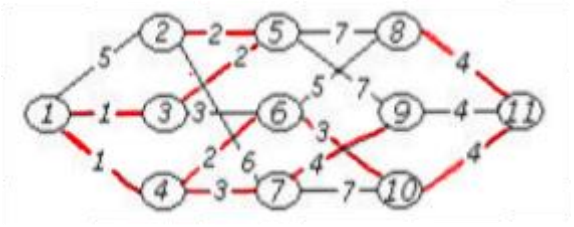
```
for (i = 0; i < n; i++)
{
		visited[i] = false;
		for (j = 0; j < n; j++)
			cout << " " << graph[i][j];
		cout << endl;
}
cout << "Старп >> "; cin >> start;

bool *vis = new bool[n];
cout << "Порядок обходу: ";
DFS(start - 1);
delete[]visited;		}
```

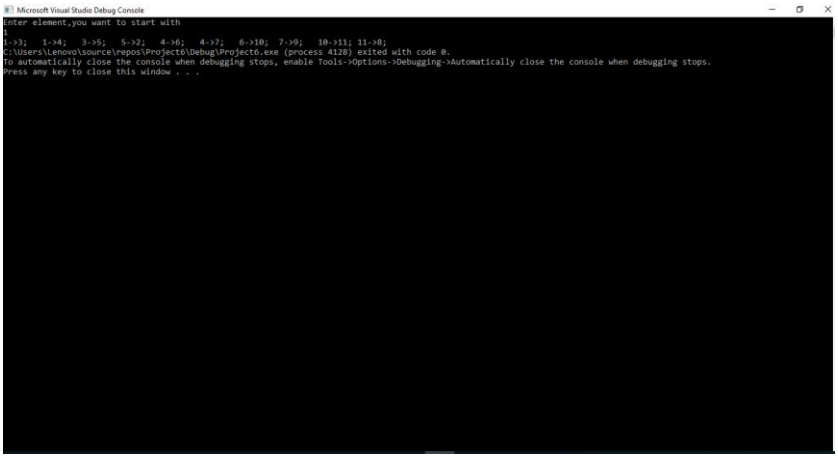
5. **Знайти двома методами (Краскала і Прима) мінімальне остове дерево графа.**

Прима

Порядок додавання ребер до остового дерева

1->3 1->4 4->6 3->5 5->2 4->7 6->10 7->9 10->11 11->8






```
#include <iostream>
#include <vector>
using namespace std;
double ValidInput();

vector<int> DeleteElement(vector<int> vector, int valueToDelete);

int main()
{
		int matrix[11][11];
		matrix[0][0] = INT_MAX;
		matrix[0][1] = 5;
		matrix[0][2] = 1;
		matrix[0][3] = 1;
		matrix[0][5] = INT_MAX;
		matrix[0][6] = INT_MAX;
		matrix[0][7] = INT_MAX;
		matrix[0][8] = INT_MAX;
		matrix[0][9] = INT_MAX;
		matrix[0][10] = INT_MAX;
		matrix[1][0] = 5;
		matrix[1][1] = INT_MAX;
		matrix[1][2] = INT_MAX;
		matrix[1][3] = INT_MAX;
```

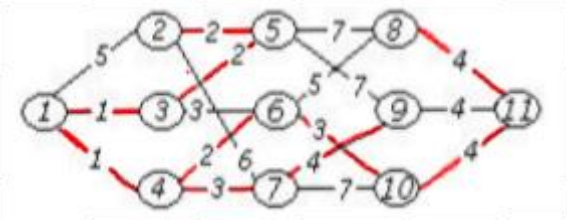
```
matrix[1][4] = 2;
matrix[1][5] = INT_MAX;
matrix[1][6] = 6;
matrix[1][7] = INT_MAX;
matrix[1][8] = INT_MAX;
matrix[1][9] = INT_MAX;
matrix[1][10] = INT_MAX;
matrix[2][0] = 1;
matrix[2][1] = INT_MAX;
matrix[2][2] = INT_MAX;
matrix[2][3] = INT_MAX;
matrix[2][4] = 2;
matrix[2][5] = 3;
matrix[2][6] = INT_MAX;
matrix[2][7] = INT_MAX;
matrix[2][8] = INT_MAX;
matrix[2][9] = INT_MAX;
matrix[2][10] = INT_MAX;
matrix[3][0] = 1;
matrix[3][1] = INT_MAX;
matrix[3][2] = INT_MAX;
matrix[3][3] = INT_MAX;
matrix[3][4] = INT_MAX;
matrix[3][5] = 2;
matrix[3][6] = 3;
matrix[3][7] = INT_MAX;
matrix[3][8] = INT_MAX;
matrix[3][9] = INT_MAX;
 matrix[3][10] = INT_MAX;
matrix[4][0] = INT_MAX;
matrix[4][1] = 2;
matrix[4][2] = 2;
matrix[4][3] = INT_MAX;
matrix[4][4] = INT_MAX;
matrix[4][5] = INT_MAX;
matrix[4][6] = INT_MAX;
matrix[4][7] = 7;
matrix[4][8] = 7;
matrix[4][9] = INT_MAX;
matrix[4][10] = INT_MAX;
matrix[5][0] = INT_MAX;
matrix[5][1] = INT_MAX;
matrix[5][2] = 3;
matrix[5][3] = 2;
matrix[5][4] = INT_MAX;
matrix[5][5] = INT_MAX;
matrix[5][6] = INT_MAX;
matrix[5][7] = 5;
matrix[5][8] = INT_MAX;
matrix[5][9] = 3;
matrix[5][10] = INT_MAX;
matrix[6][0] = INT_MAX;
matrix[6][1] = 6;
matrix[6][2] = INT_MAX;
matrix[6][3] = 3;
matrix[6][4] = INT_MAX;
matrix[6][5] = INT_MAX;
matrix[6][6] = INT_MAX;
matrix[6][7] = INT_MAX;
matrix[6][8] = 4;
matrix[6][9] = 7;
matrix[6][10] = INT_MAX;
matrix[7][0] = INT_MAX;
matrix[7][1] = INT_MAX;
matrix[7][2] = INT_MAX;
matrix[7][3] = INT_MAX;
matrix[7][4] = 7;
matrix[7][5] = 5;
matrix[7][6] = INT_MAX;
matrix[7][7] = INT_MAX;
matrix[7][8] = INT_MAX;
matrix[7][9] = INT_MAX;
matrix[7][10] = 4;
matrix[8][0] = INT_MAX;
matrix[8][1] = INT_MAX;
matrix[8][2] = INT_MAX;
matrix[8][3] = INT_MAX;
matrix[8][4] = 7;
matrix[8][5] = INT_MAX;
matrix[8][6] = 4;
matrix[8][7] = INT_MAX;
matrix[8][8] = INT_MAX;
matrix[8][9] = INT_MAX;
matrix[8][10] = 4;
matrix[9][0] = INT_MAX;
matrix[9][1] = INT_MAX;
matrix[9][2] = INT_MAX;
matrix[9][3] = INT_MAX;
matrix[9][4] = INT_MAX;
matrix[9][5] = 3;
matrix[9][6] = 7;
matrix[9][7] = INT_MAX;
matrix[9][8] = INT_MAX;
matrix[9][9] = INT_MAX;
matrix[9][10] = 4;
matrix[10][0] = INT_MAX;
matrix[10][1] = INT_MAX;
matrix[10][2] = INT_MAX;
matrix[10][3] = INT_MAX;
matrix[10][4] = INT_MAX;
matrix[10][5] = INT_MAX;
matrix[10][6] = INT_MAX;
matrix[10][7] = 4;
matrix[10][8] = 4;
matrix[10][9] = 4;
matrix[10][10] = INT_MAX;
cout << "Enter element,you want to start with\n";
```

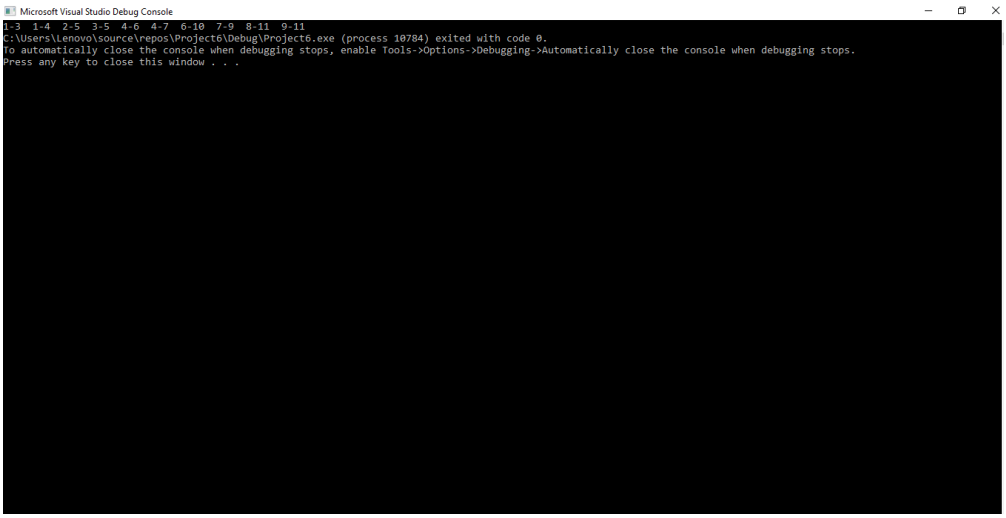
```
	double element = ValidInput();
	vector<int> usedPoints;
	vector<int> unusedPoints{ 1,2,3,4,5,6,7,8,9,10,11 };
	double i = 0;
	do
	{
		if (i == 10 && unusedPoints.at(i) != element)
		{
			cout << "Bad entry\n";
			element = ValidInput();
			i = 0;
		}
		else if (unusedPoints.at(i) == element)

		{
			break;
		}
		i++;
	} while (i != 11);

	usedPoints.push_back(element);
	unusedPoints = DeleteElement(unusedPoints, element);
	while (unusedPoints.size() != 0)
	{
		int min = INT_MAX;
		int usedPoint;
		int valueToPush;
		for (int i = 0; i < unusedPoints.size(); i++)
		{
			for (int j = 0; j < usedPoints.size(); j++)
			{
				if (min > matrix[unusedPoints[i] - 1][usedPoints[j] - 1])
				{
					min = matrix[unusedPoints[i] - 1][usedPoints[j] - 1];
					valueToPush = unusedPoints[i];
					usedPoint = usedPoints[j];
				}
			}
		}

cout << usedPoint << "->" << valueToPush << ";\t";
usedPoints.push_back(valueToPush);
unusedPoints = DeleteElement(unusedPoints, valueToPush);

	}

}
vector<int> DeleteElement(vector<int> vector, int valueToDelete)
{
	for (int i = 0; i < vector.size(); i++)
	{
		if (vector.at(i) == valueToDelete)
		{
			vector.erase(vector.begin() + i);
		}
	}
	return vector;
}
double ValidInput()
{
	double x;
	cin >> x;
	while (std::cin.fail())
	{
		cin.clear();
		cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
		cout << "Bad entry. \n";
		cin >> x;
	}
	return x;
}
```

Краскала

Порядок додавання ребер до остового дерева(повершинно):

1-3, 1-4 ,4-6 ,3-5 ,5-2 ,4-7, 6-10, 7-9 ,8-11 ,10-11






```
#include<iostream>
#include<vector>
#include<algorithm
using namespace std;
int main()
{
	int m = 18;
	int n = 12;
	vector < pair < int, pair<int, int> > > graf(m);
	graf = {
	{5,{1,2}},
	{1,{1,3}},
	{1,{1,4}},
	{2,{2,5}},
	{2,{3,5}},
	{3,{3,6}},
	{2,{4,6}},
	{6,{2,7}},
	{3,{4,7}},
	{3,{6,10}},
	{5,{6,8}},
	{7,{5,8}},
	{7,{5,9}},
	{4,{7,9}},
	{7,{7,10}},
	{4,{9,11}},
	{4,{8,11}},
	{4,{10,11}}
	};
	int cost = 0;
	vector < pair<int, int> > res;
	sort(graf.begin(), graf.end());
	vector<int> tree_id(n);
	for (int i = 1; i < n; ++i)

		tree_id[i] = i;

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

	{
		int a = graf[i].second.first, b = graf[i].second.second, l = graf[i].first;
		if (tree_id[a] != tree_id[b])

		{
			cost += l;
			res.push_back(make_pair(a, b));
			cout << a << "-" << b << "  ";
			int old_id = tree_id[b], new_id = tree_id[a];
			for (int j = 0; j < n; ++j)
				if (tree_id[j] == old_id)


tree_id[j] = new_id;

}}}
```

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

7)

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

Розписавши всі Розгалуження матиму:

1->5->4->8 ->6 ->7->2->3 Вага шляху 22

1->5->4->8->6->7->3->2 Вага шляху 22

1->5->7->6->4->8->2->3 Вага шляху 22

1->5->7->6->4->8->3->2 Вага шляху 22

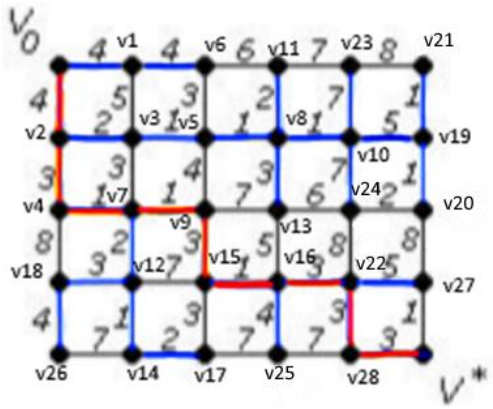
1->5->8->4->6->7->2->3 Вага шляху 21

1->5->8->4->6->7->3->2 Вага шляху 21

Отже,найкоротший шлях 1->5->8->4->6->7->2->3 або 1->5->8->4->6->7->3->2 Вага шляху 21

```
#include<stdio.h>
#include<conio.h>
#include<iostream>

using namespace std;

int c = 0, cost = 999;
int graph[8][8] = {
                        {999,5,5,5,4,6,5,5 },
                        {5,999,7,3,3,5,4,5 },
                        {5,7,999,4,5,6,4,5 },
                        {5,3,4,999,1,2,5,1 },
                        {4,3,5,1,999,5,1,1 },
                        {6,5,6,2,5,999,2,3 },
                        {5,4,2,5,1,2,999,5 },
                        {5,5,5,1,1,3,5,999 }
};

void swap(int *x, int *y)
{
           int temp;
           temp = *x;
           *x = *y;
           *y = temp;
}
void copy_array(int *a, int n)
{
           int i, sum = 0;
           for (i = 0; i <= n; i++)
           {
                      sum += graph[a[i % 8]][a[(i + 1) % 8]];
           }
           if (cost > sum)
           {
                      cost = sum;
           }
}
void permute(int *a, int i, int n)
{
           int j, k;
           if (i == n)
           {
                      copy_array(a, n);
           }
           else
           {
                      for (j = i; j <= n; j++)
                      {
                                 swap((a + i), (a + j));
                                 permute(a, i + 1, n);
                                 swap((a + i), (a + j));
                      }
           }
}
int main()
{
           int i, j;
           int a[] = {0,1,2,3,4,5,6,7};
           permute(a, 0, 7);
           cout << "minimum cost:" << cost << endl;
}
```

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



Будем позначати найближчі вершини(v1,v2,v3…) у порядку їх появи

l(v1)=4,l(v2)=4,l(v3)=6,l(v4)=7,l(v5)=7,l(v6)=8,l(v7)=8,l(v9)=9,l(v10)=9,l(v11)=10,l(v12)=10,l(v13)=11,

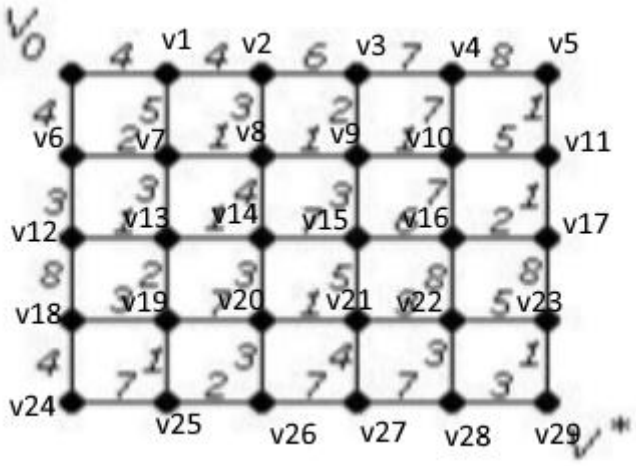
l(v14)=11,l(v15)=12,l(v16)=13,l(v17)=13,l(v18)=13,l(v19)=14,l(v20)=15,l(v21)=15,l(v22)=16,l(v23)=16,

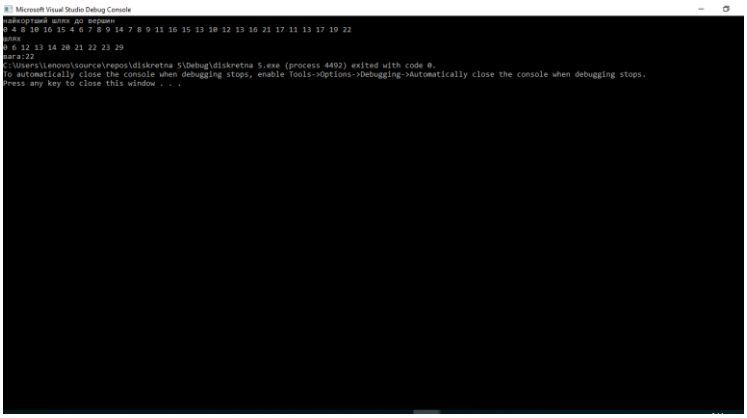
l(v24)=16,l(v25)=17,l(v26)=19,l(v27)=21,l(v28)=19,l(v*)=22

Дерево найближчих вершин виділено синіми лініями і є кістяковим деревом, тому що містить усі вершини графа.

Шуканий найкоротший ланцюг(v0,v2,v4,v7,v9,v15,v16,v22,v28,v*) вага 22.

Позначимо вершини так для зручності в програмному застосуванні






```
#include<locale>
#include<iostream>
#include <stdio.h>
#include <stdlib.h>
#define SIZE 30
using namespace std;
int main()
{
	setlocale(LC_ALL, "Ukrainian");
	int arr[SIZE][SIZE];
	int dov[SIZE]; // min vid
	int v[SIZE]; // ver
	int temp, index, min;
	int begin_index = 0;


	for (int i = 0; i < SIZE; i++)
	{
		for (int j = 0; j < SIZE; j++) {
			arr[i][j] = 0;
		}
	}
	arr[0][1] =4;
	arr[0][6] =4 ;
	arr[1][0] =4 ;
	arr[1][2] =4 ;
	arr[1][7] =5 ;
	arr[2][1] =4;
	arr[2][8] =3 ;
	arr[2][3] =6 ;
```

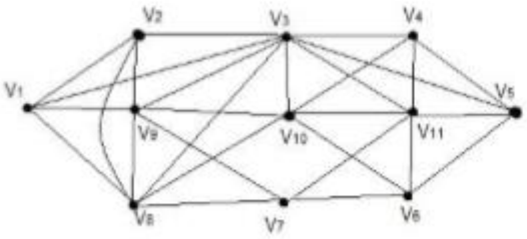
```
arr[3][2] =6 ;
arr[3][4] =7;
arr[3][9] =2 ;
arr[4][3] =7 ;
arr[4][5] =8 ;
arr[4][10] =7 ;
arr[5][4] =8;
arr[5][11] =1 ;
arr[6][0] =4;
arr[6][7] =2;
arr[6][12] =3;
arr[7][1] =5 ;
arr[7][6] =2;
arr[7][8] =1;
arr[7][13] =3 ;
arr[8][7] =1 ;
arr[8][2] =3;
arr[8][14] =4;
arr[8][9] =1;
arr[9][8] =1;
arr[9][3] =2 ;
arr[9][15] =3;
arr[9][10] =1 ;
arr[10][9] =1;
arr[10][4] =7;
arr[10][16] =7;
arr[10][11] =5 ;
arr[11][10] =5 ;
arr[11][5] =1 ;
arr[11][17] =1 ;
arr[12][6] =3;
arr[12][13] =1 ;
arr[12][18] =8 ;
arr[13][12] =1;
arr[13][7] =3;
arr[13][19]=2;
arr[13][14] =1;
arr[14][13] =1;
arr[14][8] =4;
arr[14][20] =3;
arr[14][15] =7;
arr[15][14] =7;
arr[15][9] = 3;
arr[15][21] =5;
arr[15][16] =6;
arr[16][15] = 6;
arr[16][10] =7 ;
arr[16][22] = 8;
arr[16][17] =2;
arr[17][16] =2;
arr[17][11] =1;
arr[17][23] =8 ;
arr[18][12] =8 ;
arr[18][19] =3 ;
arr[18][24] =4 ;
arr[19][18] =3;
arr[19][13] =2;
arr[19][25] =1;
arr[19][20] =7 ;
arr[20][19] =7 ;
arr[20][14] =3 ;
arr[20][26] =3;
arr[20][21] =1;
arr[21][20] =1;
arr[21][15] =5;
arr[21][27] =4;
arr[21][22] =3 ;
arr[22][21] =3;
arr[22][16] =8;
arr[22][28] =3;
arr[22][23] =5;
arr[23][22] =5;
arr[23][17] =8;
arr[23][29] =1;
arr[24][18] =4;
arr[24][25] =7;
arr[25][24] =7;
arr[25][19] =1;
arr[25][26] = 2;
arr[26][25] = 2;
arr[26][20] =3;
arr[26][27] =7;
arr[27][26] =7;
arr[27][21] =4;
arr[27][28] =7;
arr[28][27] =7;
arr[28][22] =3;
arr[28][29] =3;
arr[29][28] =3;
arr[29][23] =1;
```

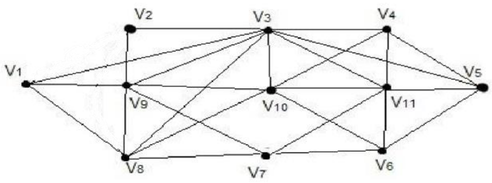
```
		for (int i = 0; i < SIZE; i++)
	{
		dov[i] = 10000;
		v[i] = 1;
	}
	dov[begin_index] = 0;

	do {
		index = 10000;
		min = 10000;
		for (int i = 0; i < SIZE; i++)
		{
			if ((v[i] == 1) && (dov[i] < min))
			{
				min = dov[i];
				index = i;
			}
		}

		if (index != 10000)
		{
			for (int i = 0; i < SIZE; i++)
			{
				if (arr[index][i] > 0)
				{
					temp = min + arr[index][i];
					if (temp < dov[i])
					{
						dov[i] = temp;
					}
				}
			}
			v[index] = 0;
		}
	} while (index < 10000);

	cout << "найкортший шлях до вершин" << endl;
	for (int i = 0; i < SIZE; i++)
		cout<<dov[i]<<" ";

		int ver[SIZE]; // від
	int end = 29;
	ver[0] = end ;
	int k = 1;
	int weight = dov[end]; //

	while (end != begin_index)
	{
		for (int i = 0; i < SIZE; i++)
			if (arr[end][i] != 0)                          {
				int temp = weight - arr[end][i];
				if (temp == dov[i])                                    {
					weight = temp;
					end = i;
					ver[k] = i  ;
					k++;
				}
			}
	}
	cout << endl;
	cout << "шлях" << endl;
	for (int i = k - 1; i >= 0; i--) {
		cout << ver[i] << " ";
	}
	cout << endl;
	cout <<"вага:"<< dov[29];
	return 0;
}
```

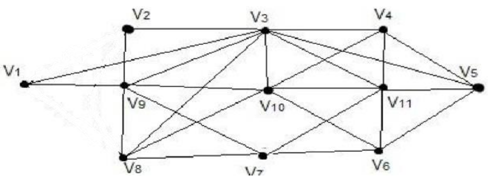
8.Знайти ейлеровий цикл в ейлеровому графі двома методами: а) Флері; б) елементарних циклів.

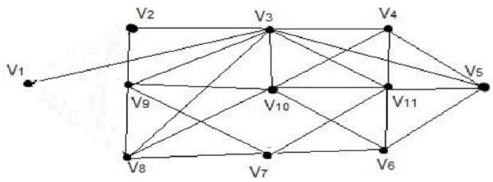
: а) Флері

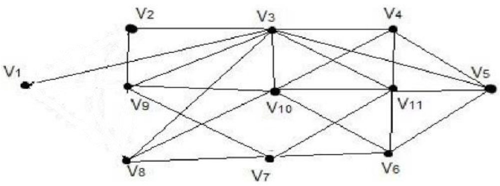
7)

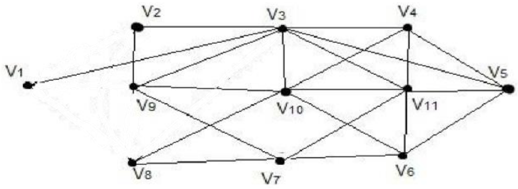

V_2
V_3
V_4
V_1
V_5
V_9
V_{10}
V_{11}
V_8
V_7
V_6

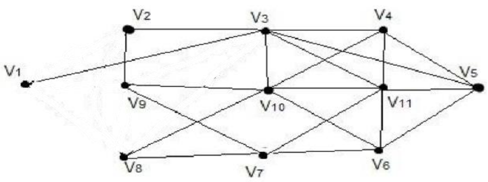

V_2
V_3
V_4
V_1
V_5
V_9
V_{10}
V_{11}
V_8
V_7
V_6

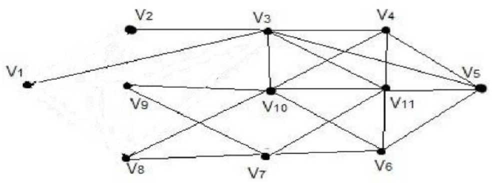

V_2
V_3
V_4
V_1
V_5
V_9
V_{10}
V_{11}
V_8
V_6

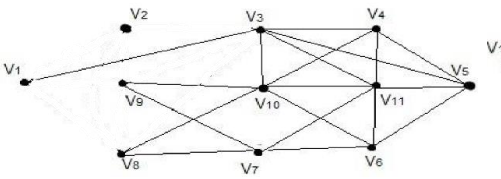

V_2
V_3
V_4
V_1
V_5
V_9
V_{10}
V_{11}
V_8
V_7
V_6

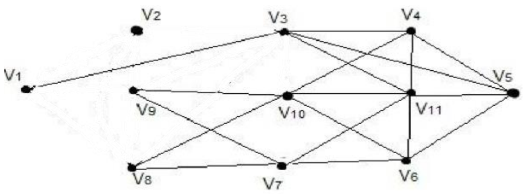

V_2
V_3
V_4
V_1
V_5
V_9
V_{10}
V_{11}
V_8
V_7
V_6

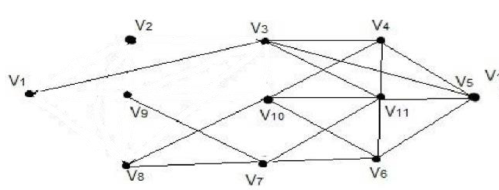

V_2
V_3
V_4
V_1
V_5
V_9
V_{10}
V_{11}
V_8
V_7
V_6

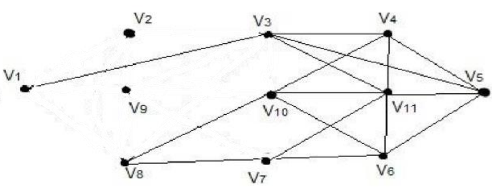

V_2
V_3
V_4
V_1
V_5
V_9
V_{10}
V_{11}
V_8
V_7
V_6

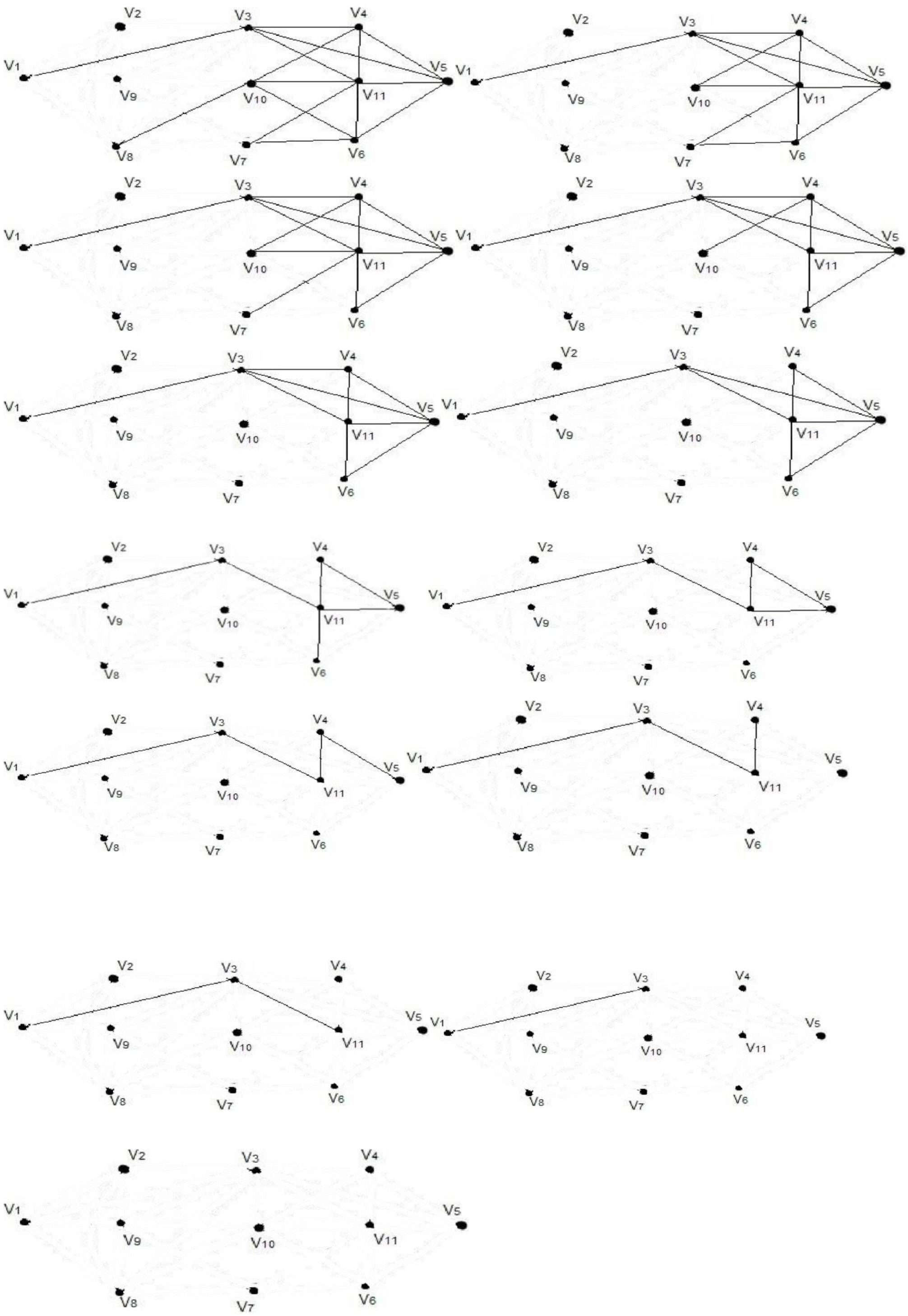

V_2
V_3
V_4
V_1
V_5
V_9
V_{10}
V_{11}
V_8
V_7
V_6

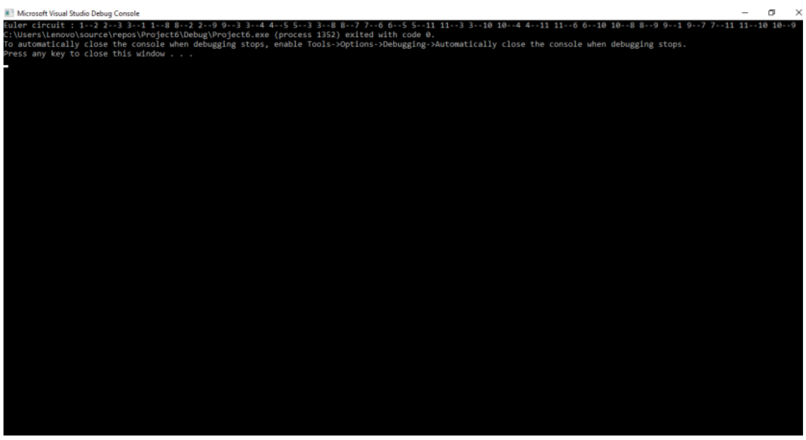

V_2
V_3
V_4
V_1
V_5
V_9
V_{10}
V_{11}
V_8
V_7
V_6


V_2
V_3
V_4
V_1
V_5
V_9
V_{10}
V_{11}
V_8
V_7
V_6


V_2
V_3
V_4
V_1
V_5
V_9
V_{10}
V_{11}
V_8
V_7
V_6


V_2
V_3
V_4
V_1
V_5
V_9
V_{10}
V_{11}
V_8
V_7
V_6

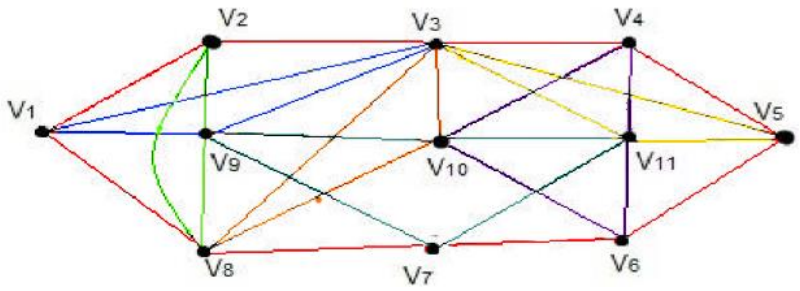





```
#include<iostream>
#include<vector>
#include<cmath>
```

```
#define N 11

using namespace std;
int graph[N][N] = {
   {0,1,1,0,0,0,0,1,1,0,0},
   {1,0,1,0,0,0,0,1,1,0,0},
   {1,1,0,1,1,0,0,1,1,1,1},
   {0,0,1,0,1,0,0,0,0,1,1},
   {0,0,1,1,0,1,0,0,0,0,1},
   {0,0,0,0,1,0,1,0,0,1,1},
   {0,0,0,0,0,1,0,1,1,0,1},
   {1,1,1,0,0,0,1,0,1,1,0},
   {1,1,1,0,0,0,1,1,0,1,0},
   {0,0,1,1,0,1,0,1,1,0,1},
   {0,0,1,1,1,1,1,0,0,1,0},


};
int helpGraph[N][N];
int findStart() {
           for (int i = 0; i < N; i++) {
                      int deg = 0;
                      for (int j = 0; j < N; j++) {
                                 if (helpGraph[i][j])
                                            deg++;
                      }
                      if (deg % 2 != 0)
                                 return i;
           }
           return 0;
}
int deep(int prev, int start, bool visit[]) {
           int count = 1;
           visit[start] = true;
           for (int u = 0; u < N; u++) {
                      if (prev != u) {
                                 if (!visit[u]) {
                                            if (helpGraph[start][u]) {
                                                       count += deep(start, u, visit);
                                            }
                                 }
                      }
           }
           return count;
}
bool isBridge(int u, int v) {
           int deg = 0;
           for (int i = 0; i < N; i++)
                      if (helpGraph[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 (helpGraph[i][j])
                                            count++;
           return count;
}
void fleury(int start) {
           static int edge = edgeCount();
           static int v_count = N;
           for (int v = 0; v < N; v++) {
                      if (helpGraph[start][v]) {
                                 bool visited[N] = { false };
                                 if (isBridge(start, v)) {
                                            v_count--;
                                 }
                                 int cnt = deep(start, v, visited);
                                 if (abs(v_count - cnt) <= 2) {
                                            cout << start+1 << "--" << v+1 << " ";
                                            if (isBridge(v, start)) {
                                                       v_count--;
                                            }
                                            helpGraph[start][v] = helpGraph[v][start] = 0;
                                            edge--;
                                            fleury(v);
                                 }
                      }
           }
}
int main() {
           for (int i = 0; i < N; i++)
                      for (int j = 0; j < N; j++)
                                 helpGraph[i][j] = graph[i][j];
           cout << "Euler circuit: ";
           fleury(findStart());
}
```

б) елементарних циклів.



Елементарні цикли

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

1-3-9

2-9-8

8-3-10

9-7-11-10

5-11-3

6-4-10

```
Microsoft Visual Studio Debug Console
1->9->10->11->7->9->8->10->6->11->4->10->3->11->5->6->7->8->3->5->4->3->9->2->8->1->3->2->1->
C:\Users\Lenovo\source\repos\Project6\Debug\Project6.exe (process 3752) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

```
#include<iostream>
#include<vector>
#include<cmath>
#include<stack>

using namespace std;
int main() {

          int n=11;

          vector < vector<int> > g(n, vector<int>(n));
          g = {
   {0,1,1,0,0,0,0,1,1,0,0},
   {1,0,1,0,0,0,0,1,1,0,0},
   {1,1,0,1,1,0,0,1,1,1,1},
   {0,0,1,0,1,0,0,0,0,1,1},
   {0,0,1,1,0,1,0,0,0,0,1},
   {0,0,0,0,1,0,1,0,0,1,1},
   {0,0,0,0,0,1,0,1,1,0,1},
   {1,1,1,0,0,0,1,0,1,1,0},
   {1,1,1,0,0,0,1,1,0,1,0},
   {0,0,1,1,0,1,0,1,1,0,1},
   {0,0,1,1,1,1,1,0,0,1,0},


          };
```

```
		vector<int> deg(n);
	for (int i = 0; i < n; ++i)
		for (int j = 0; j < n; ++j)
			deg[i] += g[i][j];

	int first = 0;
	while (!deg[first])  ++first;

	int v1 = -1, v2 = -1;
	bool bad = false;
	for (int i = 0; i < n; ++i)
		if (deg[i] & 1)
			if (v1 == -1)
				v1 = i;
			else if (v2 == -1)
				v2 = i;
			else
				bad = true;

	if (v1 != -1)
		++g[v1][v2], ++g[v2][v1];

	stack<int> st;
	st.push(first);
	vector<int> res;
	while (!st.empty())
	{
		int v = st.top();
		int i;
		for (i = 0; i < n; ++i)
			if (g[v][i])
				break;
		if (i == n)
		{
			res.push_back(v);
			st.pop();
		}
		else
		{
			--g[v][i];
			--g[i][v];
			st.push(i);
		}
	}

	if (v1 != -1)
		for (size_t i = 0; i + 1 < res.size(); ++i)
			if (res[i] == v1 && res[i + 1] == v2 || res[i] == v2 && res[i + 1] == v1)
			{
				vector<int> res2;
				for (size_t j = i + 1; j < res.size(); ++j)
					res2.push_back(res[j]);
				for (size_t j = 1; j <= i; ++j)
					res2.push_back(res[j]);
				res = res2;
				break;
			}

	for (int i = 0; i < n; ++i)
		for (int j = 0; j < n; ++j)
			if (g[i][j])
				bad = true;

	if (bad)
		puts("-1");
	else
		for (size_t i = 0; i < res.size(); ++i)
			cout<<res[i] + 1<<"->";

}
```

9.Спростити формули (привести їх до скороченої ДНФ). $(x\overline{x\bar{x}} \to y\overline{\bar{y}} \to z)$

Спочатку складемо таблицю істинності

x	y	z	$\bar{x}$	$\bar{y}$	$y\bar{y}$	$x\bar{x}$	$x\bar{x} \to y\bar{y}$	$\overline{x\bar{x} \to y\bar{y}}$	$(x\overline{x\bar{x}} \to y\overline{\bar{y}} \to z)$
1	0	0	0	1	0	0	1	0	1
1	0	1	0	1	0	0	1	0	1
1	1	0	0	0	0	0	1	0	1
1	1	1	0	0	0	0	1	0	1
0	0	1	1	1	0	0	1	0	1
0	1	0	1	0	0	0	1	0	1
0	1	1	1	0	0	0	1	0	1
0	0	0	1	1	0	0	1	0	1

Оскільки в нас завжди формула приймає значення True то скорочена ДНФ формула=1