МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”
ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра систем штучного інтелекту

Розрахунково-графічна робота

із дисципліни
«Дискретна математика»

Виконав:
студент групи КН-113
Калапунь.Н.Т.

Викладач:

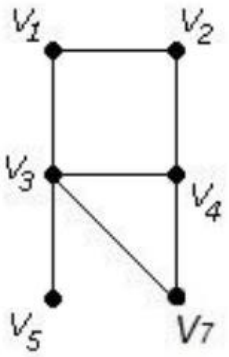
Мельникова Н.І.

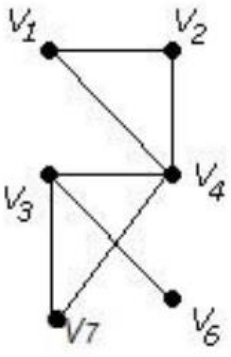
Львів – 2019 р.

Варіант – 13

Завдання 1

Розв'язати на графах наступні задачі:





1. Виконати наступні операції над графами:

1) знайти доповнення до першого графу.

2) об’єднання графів.

3) кільцеву суму G1 та G2 (G1+G2)

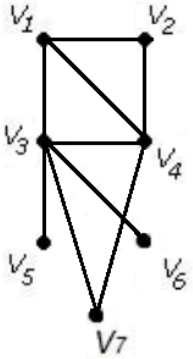
4) розщепити вершину у другому графі.

5) виділити підграф А, що складається з 3-х вершин в G1 і знайти стягнення А в G1 (G1\ A).

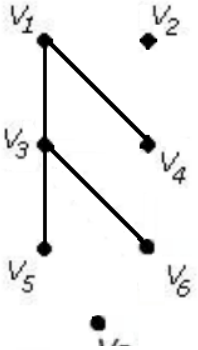
6) добуток графів.

1)

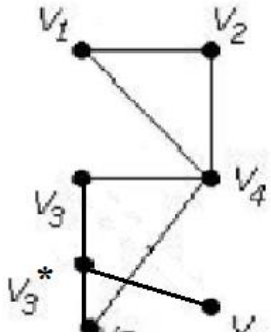
2)



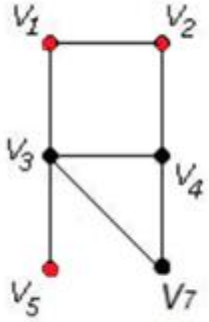
2)

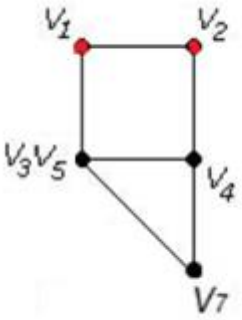


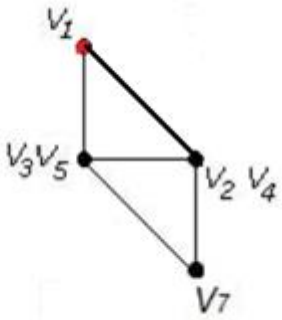
4)

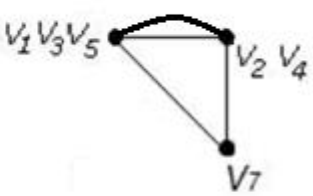


5)

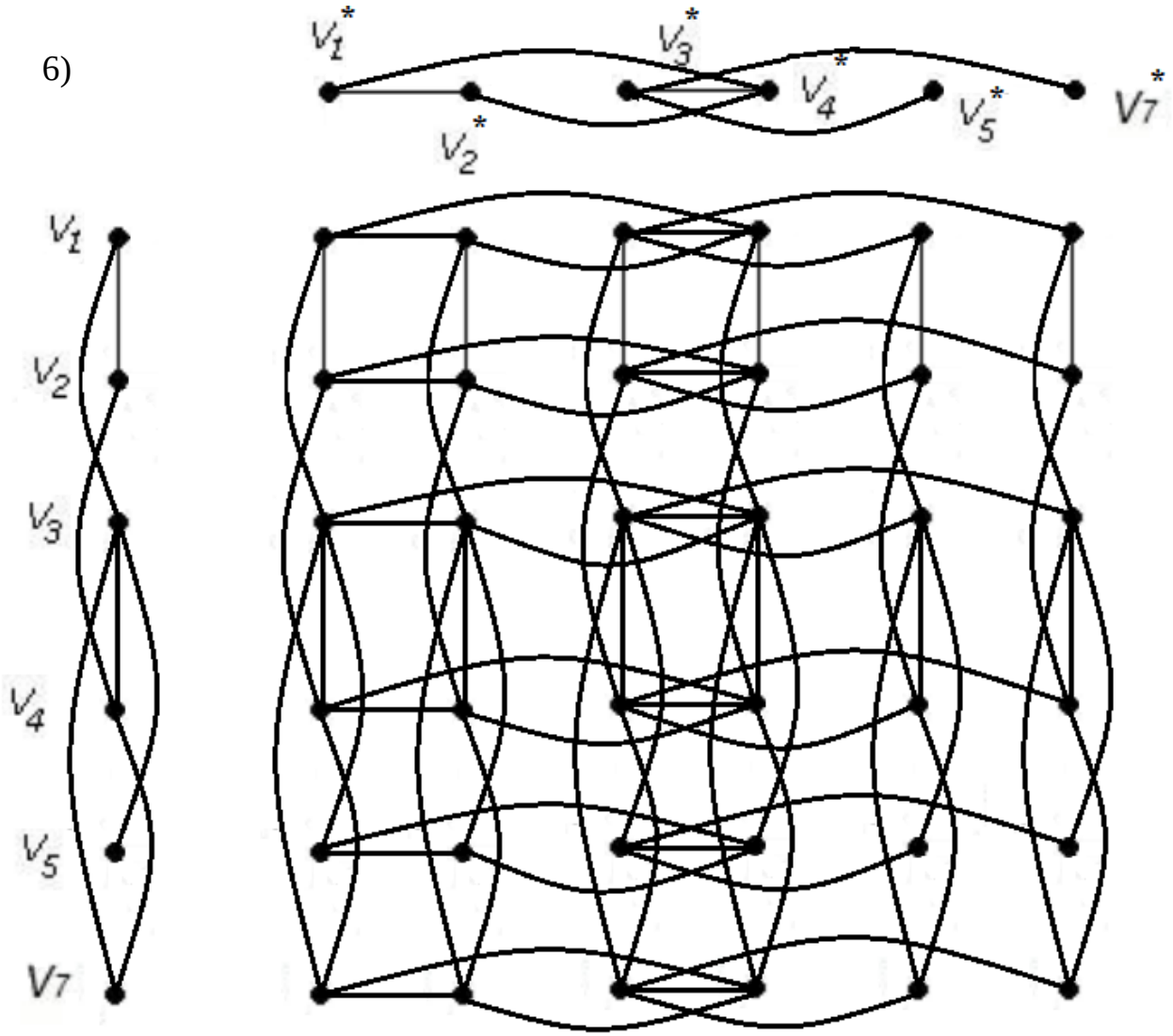






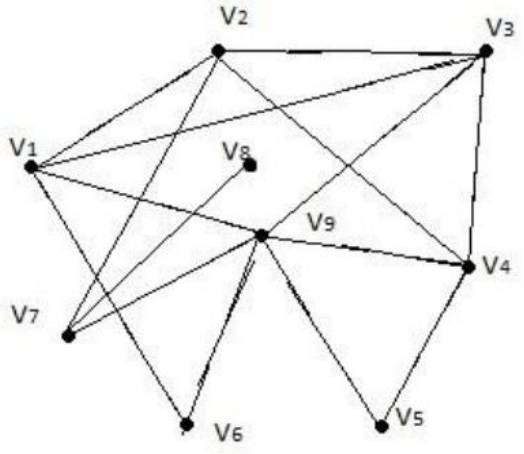


6)



Завдання № 2

Скласти таблицю суміжності для орграфа.



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

Завдання № 3

Для графа з другого завдання знайти діаметр.

	V1	V2	V3	V4	V5	V6	V7	V8	V9
V1	-	1	1	2	2	1	2	3	1
V2	1	-	1	1	2	3	1	2	2
V3	1	1	-	1	2	2	2	3	1
V4	2	1	1	-	1	2	2	3	1
V5	2	2	2	1	-	2	2	3	1
V6	1	3	2	2	2	-	2	3	1
V7	2	1	2	2	2	2	-	1	1
V8	3	2	3	3	3	3	1	-	2
V9	1	2	1	1	1	1	1	2	-

Діаметр = 3

Завдання № 4

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

Вершина	DFS	стек
V_1	1	V_1
V_2	2	V_1 V_2
V_3	3	V_1 V_2 V_3
V_4	4	V_1 V_2 V_3 V_4
V_5	5	V_1 V_2 V_3 V_4 V_5
V_9	6	V_1 V_2 V_3 V_4 V_5 V_9
V_6	7	V_1 V_2 V_3 V_4 V_5 V_9 V_6
-	-	V_1 V_2 V_3 V_4 V_5 V_9
V_7	8	V_1 V_2 V_3 V_4 V_5 V_9 V_7
V_8	9	V_1 V_2 V_3 V_4 V_5 V_9 V_7 V_8
-	-	V_1 V_2 V_3 V_4 V_5 V_9 V_7
-	-	V_1 V_2 V_3 V_4 V_5 V_9
-	-	V_1 V_2 V_3 V_4 V_5
-	-	V_1 V_2 V_3 V_4
-	-	V_1 V_2 V_3
-	-	V_1 V_2
-	-	V_1
-	-	-

Перевіримо програмно:

```
#include <iostream>
#include <string>

using namespace std;

struct vershyna
{
	bool dfs = false;
};
struct rebro
{
```

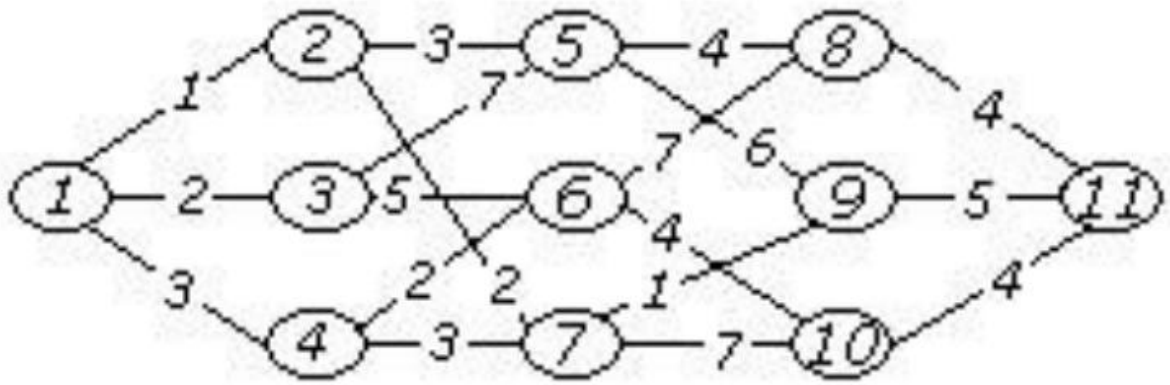
```
	int v1;
	int v2;
};

int leng(string str)
{
	int i = 0;

	while (str[i] != '\0')
	{
		i++;
	}
	return i;
}


int main()
{
	setlocale(LC_ALL, "Ukrainian");
	int n, m, p;
	int begin;
	int count = 0;
	int t = 0;
	int head = 0;

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

	int* stek = new int[m];
	rebro* reb = new rebro[n];
	vershyna* v = new vershyna[m];


	for (int i = 0; i < n; i++)
	{
		cout << "Ребро " << i + 1<<endl;
		cout << "Введіть першу вершину: ";
		reb[i].v1 = correct(1, m);
		cout << "Введіть другу вершину: ";
		reb[i].v2 = correct(1, m);
		cout << endl;
	}

	cout << "З якої вершини почати обхід? ";
	cin>>begin;

	stek[0] = begin;
	v[begin - 1].dfs = true;
	count++;

		while (count != 0)
		{
			for (int i = 0; i < n; i++)
			{

				if ((stek[count - 1] == reb[i].v1 && v[reb[i].v2 - 1].dfs ==
false) || (stek[count - 1] == reb[i].v2 && v[reb[i].v1 - 1].dfs == false))
				{
					t++;
				}
			}

			if (t == 0)
			{
				count--;
```

```
                }
                else
                {
                        for (int i = 0; i < n; i++)
                        {
                                if (stek[count - 1] == reb[i].v2 && v[reb[i].v1 - 1].dfs
== false)
                                {
                                        stek[count] = reb[i].v1;
                                        v[reb[i].v1 - 1].dfs = true;

                                        count++;
                                        goto point;
                                }

                                if (stek[count - 1] == reb[i].v1 && v[reb[i].v2 - 1].dfs
== false)
                                {
                                        stek[count] = reb[i].v2;
                                        v[reb[i].v2 - 1].dfs = true;

                                        count++;
                                        goto point;
                                }
                        }
                }
        point:
                for (int i = 0; i < count; i++)
                {
                        cout << stek[i] << " ";
                }
                if (count != 0)
                {
                        cout << endl;
                }
                t = 0;
        }
        cout << "Стек пустий" << endl;
}
```

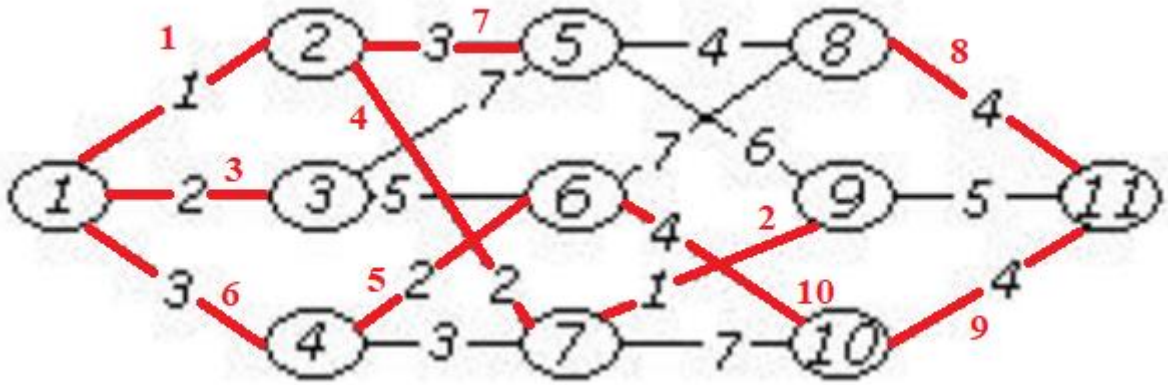
```
Ребро 11
Введіть першу вершину: 7
Введіть другу вершину: 9

Ребро 12
Введіть першу вершину: 9
Введіть другу вершину: 4

Ребро 13
Введіть першу вершину: 7
Введіть другу вершину: 8

Ребро 14
Введіть першу вершину: 1
Введіть другу вершину: 9

Ребро 15
Введіть першу вершину: 3
Введіть другу вершину: 9

З якої вершини почати обхід? 1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 9
1 2 3 4 5 9 6
1 2 3 4 5 9
1 2 3 4 5 9 7
1 2 3 4 5 9 7 8
1 2 3 4 5 9 7
1 2 3 4 5 9
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
Стек пустий

C:\CODE\Training\32\Debug\32.exe (n
```

Завдання № 5

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



Аглоритм Караскала:



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

E = {(1,2) (7,9) (1,3) (2,7) (4,6) (1,4) (2,5) (8,11) (10,11) (6,10)**}**

Довжина: 26

Перевіримо програмно:

```
#include <iostream>
#include <string>
using namespace std;

int vershyny;
int rebra;
struct Graf {
	int v1=0;
	int v2=0;
	int weight=0;
};

Graf graf[30];
```

```
int *versh =new int[vershyny];
void CreateGraf() {
	cout << "Введіть кількість вершин: ";
	    cin>> vershyny;
	    cout << "Введіть кількість ребер: ";
	    cin >> rebra;
	       int v = 0;
for (int i = 0;i < rebra;i++)
	{
		cout << "Введіть вершину 1: ";
		cin >> graf[i].v1;

		cout << "Введіть вершину 2: ";
		cin >> graf[i].v2;

		cout << "Введіть вагу: ";
		cin >> graf[i].weight;
	}
}
void Sort() {
	for (int i = 0; i < rebra;i++) {
		for (int j = 0; j < rebra;j++)
		{
			Graf test;
			if (i == j)
			{
				continue;
			}
			else if (graf[i].weight < graf[j].weight)
			{
				test = graf[i];
				graf[i] = graf[j];
				graf[j] = test;
			}
		}
	}
	}

void FillArray(int** arr, int ROWS, int COLLS) //функція заповнення масиву
{
	for (int i = 0;i < ROWS;i++)
	{
		for (int j = 0; j < COLLS;j++)
			arr[i][j] = 0;
	}
}
void PrintArray(int** arr, int ROWS, int COLLS) // функція виведення масиву на екран
{
	for (int i = 0; i < ROWS; i++)
	{
		for (int j = 0; j < COLLS; j++)
		{
			cout <<arr[i][j]<<" ";
		}
		cout << endl;
	}

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

	CreateGraf();
	Sort();


	int ROWS = 5;
	int COLLS = 11;
```

```
	int** points = new int* [ROWS];   //створюємо динамічний двомірний масив
	for (int i = 0;i < ROWS; i++)
	{
		points[i] = new int[COLLS];
	}
	FillArray(points, 5, 11);
		PrintArray(points, 5, 11);
int weight = graf[0].weight;
points[0][0] = graf[0].v1;
points[0][1] = graf[0].v2;
cout << endl;
PrintArray(points, 5, 11);
cout << endl << endl;
string v1 = std::to_string(graf[0].v1);
string v2 = std::to_string(graf[0].v2);
string result= v1 + "-" + v2 + "; ";


	int size1 = 2;
	int size2 = 0;
	int size3 = 0;
	int size4 = 0;
	int size5 = 0;
	int i = 0;
		for (int gr=1;gr < rebra;gr++)
		{

			bool v1exist = false;
			bool v2exist = false;
			int j = 0;


			while (points[i][j] != 0)
			{
				if (points[i][j] == graf[gr].v1)
				{
					v1exist = true;
					break;
				}
				else if (points[i][j + 1] == 0 || j + 1 == vershyny)
				{
					if (points[i][j] != graf[gr].v1)
					{
						v1exist = false;
						break;
					}
				}
				else
				{
					j++;
				}

			}
			int k = 0;
			while (points[i][k] != 0)
			{
				if (points[i][k] == graf[gr].v2)
				{
					v2exist = true;
					k++;
					break;
				}
				else if (points[i][k + 1] == 0 || k + 1 == vershyny)
				{
					if (points[i][k] != graf[gr].v2)
					{
						v2exist = false;
						k++;
```

```
                                        break;
                                }
                        }
                        else
                        {
                                k++;
                        }

                }
                if (v1exist == true && v2exist == true)
                {
                        continue;
                        i = 0;
                }
                else if (v1exist == true && v2exist == false)
                {
                        for (int l = 1;l <= 5;l++)
                        {
                                int g = 0;
                                if (points[l][g] == 0)
                                {
                                   switch (i)
                                   {
                                    case 0: points[i][size1] = graf[gr].v2;size1++;break;
                                    case 1: points[i][size2] = graf[gr].v2;size2++;break;
                                    case 2: points[i][size3] = graf[gr].v2;size3++;break;
                                    case 3: points[i][size4] = graf[gr].v2;size4++;break;
                                    case 4: points[i][size5] = graf[gr].v2;size5++;break;
                                      }
                                      i = 0;
                                      break;
                                }
                                else
                                {
                                        while (points[l][g] != 0)
                                        {

                                                if (points[l][g] == graf[gr].v2)
                                                {
                                                        g = 0;

                                                        while (points[l][g] != 0)
                                                        {

                                                                switch (i)
                                                                {
                                                                case 0: points[i][size1] =
points[l][g];size1++;break;

                                                                case 1: points[i][size2] =
points[l][g];size2++;break;

                                                                case 2: points[i][size3] =
points[l][g];size3++;break;

                                                                case 3: points[i][size4] =
points[l][g];size4++;break;

                                                                case 4: points[i][size5] =
points[l][g];size5++;break;

                                                                }
                                                                points[l][g] = 0;

                                                                g++;

                                                        }
                                                        switch (l)
                                                                {
                                                                case 0: size1=0;break;
                                                                case 1: size2=0;break;
                                                                case 2: size3=0;break;
                                                                case 3: size4=0;break;
```

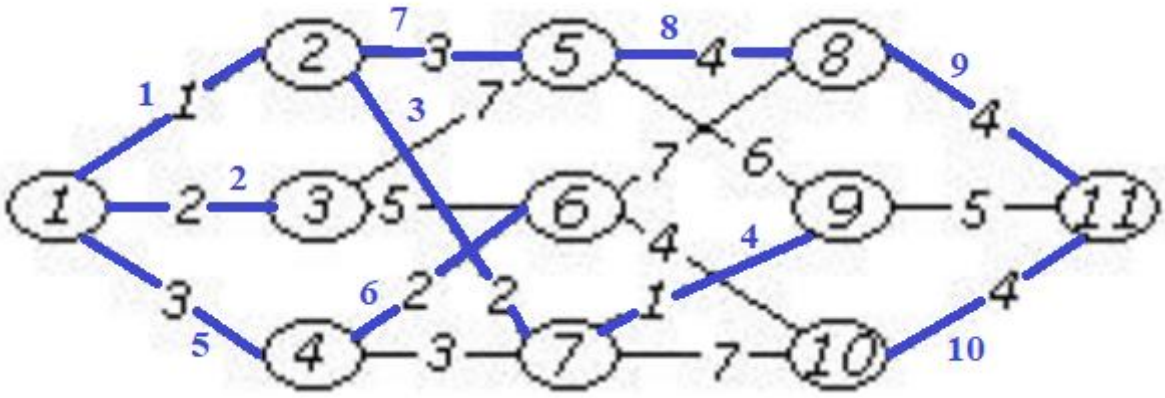
```
                                                case 4: size5=0;break;

                                                }
                                        i = 0;
                                        l = 6;
                                        break;
                                }
                                else
                                {

                                        g++;
                                }
                        }
                }
        }

}
else if (v2exist == true && v1exist == false)
{
        for (int l = 1;l <= 5;l++)
        {
                int g = 0;
                if (points[l][g] == 0)
                {
                 switch (i)
                  {
                   case 0: points[i][size1] = graf[gr].v1;size1++;break;
                   case 1: points[i][size2] = graf[gr].v1;size2++;break;
                   case 2: points[i][size3] = graf[gr].v1;size3++;break;
                   case 3: points[i][size4] = graf[gr].v1;size4++;break;
                   case 4: points[i][size5] = graf[gr].v1;size5++;break;
                       }
                       i = 0;
                       break;
                }
                while (points[l][g] != 0)
                {
                        if (points[l][g] == graf[gr].v1)
                        {
                                g = 0;
                                while (points[l][g] != 0)
                                {
                                  switch (i)
                                  {
                                        case 0: points[i][size1] =
points[l][g];size1++;break;

                                        case 1: points[i][size2] =
points[l][g];size2++;break;

                                        case 2: points[i][size3] =
points[l][g];size3++;break;

                                        case 3: points[i][size4] =
points[l][g];size4++;break;

                                        case 4: points[i][size5] =
points[l][g];size5++;break;

                                        }
                                        points[l][g] = 0;

                                        g++;
                                }
                                switch (l)
                                        {
                                        case 0: size1=0;break;
                                        case 1: size2=0;break;
                                        case 2: size3=0;break;
                                        case 3: size4=0;break;
                                        case 4: size5=0;break;

                                        }
```

```
                                        i = 0;
                                        l = 6;
                                        break;
                                }
                                else
                                {

                                        g++;
                                }
                        }
                }

        }

        else if (v2exist == false && v1exist == false)
        {
                if (points[i + 1][0] == 0)
                {
                        points[i + 1][0] = graf[gr].v1;
                        points[i + 1][1] = graf[gr].v2;
                        switch (i+1)
                        {
                        case 0: size1+=2;break;
                        case 1: size2+=2;break;
                        case 2: size3+=2;break;
                        case 3: size4+=2;break;
                        case 4: size5+=2;break;
                        }
                        i = 0;
                }
                else {
                        gr--;
                        i++;
                        continue;
                }
        }
        cout << graf[gr].v1 << "-" << graf[gr].v2 << endl;
        PrintArray(points, 5, 11);

        cout << endl << endl;
         v1 = to_string(graf[gr].v1);
         v2 = to_string(graf[gr].v2);
         result =result+ v1 + "-" + v2 + "; ";
         weight += graf[gr].weight;
    }

cout<<"++++++++++++++++++++++++++++++++++\n";


cout <<"\n\n\nМінімальне остове дерево графа складається з ребер: "<< result;
cout << "\n\nВага мінімального остового дерева графа = " << weight<<endl;
for (int i = 0;i < ROWS; i++)
{
        delete[] points[i] ;
}
delete[] versh;

}
```

Результат:

```
Мінімальне остове дерево графа закладається з ребер: 1-2; 9-7; 1-3; 7-2; 4-6; 7-4; 2-5; 8-11; 11-10; 5-8;

Вага мінімального остового дерева графа = 26
```

Аглоритм Прима:



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

E = {(1,2) (1,3) (2,7) (7,9) (1,4) (4,6) (2,5) (5,8) (8,11) (11,10)**}**

Довжина: 26

Перевіримо програмно:

```
#include <iostream>
#include <fstream>

using namespace std;

const int SIZE = 11;

int matrix[SIZE][ SIZE];
int visitedDots[SIZE];

void init() {
	ifstream fin("MyFile1.txt");
	for (int i = 0;i < SIZE;i++) {
		visitedDots[i] = 0;
		for (int j = 0;j < SIZE;j++)
			fin >> matrix[i][j];
	}
	fin.close();
}


int findNextDot(int& enterDot) {
	int min = 99999;
```

```
	int minDot = -1;
	for (int i = 0;i < SIZE;i++)
		if (visitedDots[i])
			for (int j = 0;j < SIZE;j++)
				if (matrix[i][j] < min && !visitedDots[j] && matrix[i][j]) {
					min = matrix[i][j];
					minDot = j;
					enterDot = i;
				}
	return minDot;
}
int weight = 0;
void OstTree(int dot) {
	while (dot != -1) {
		visitedDots[dot] = 1;
		int prevDot;

		dot = findNextDot(prevDot);
		if (dot != -1) {
			weight += matrix[prevDot][dot];
			cout << "(" << prevDot + 1 << ";" << dot + 1 << ") ";
		}
	}

}


int main()
{
	setlocale(LC_ALL, "Ukrainian");
	init();
	cout << "\n\nМінімальне остове дерево графа складається з ребер: ";
	OstTree(0);
	cout << "\n\nВага мінімального остового дерева графа = " << weight << endl;

}
```

Результат:

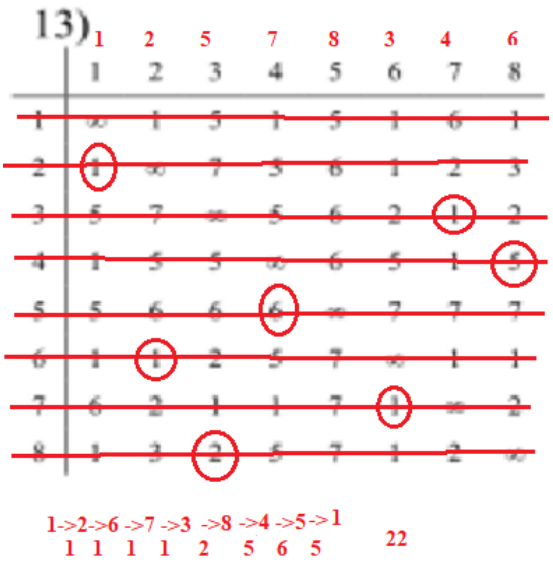
```
Мінімальне остове дерево графа складається з ребер:  1-2;  1-3;  2-7;  7-9;  1-4;  4-6;  2-5;  5-8;  6-10;  8-11;

Вага мінімального остового дерева графа = 26
```

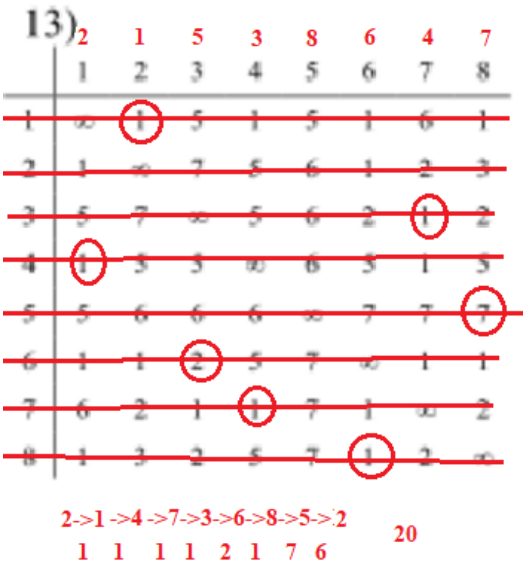
Завдання № 6

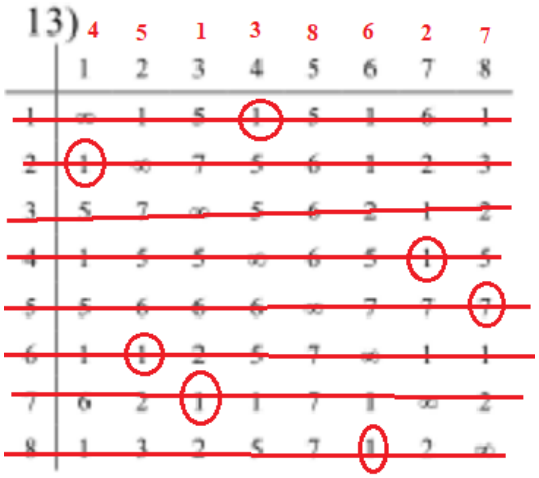
Розв’язати задачу комівояжера для повного 8-ми вершинного графа методом «іди у найближчий», матриця вагів якого має вигляд:

13)

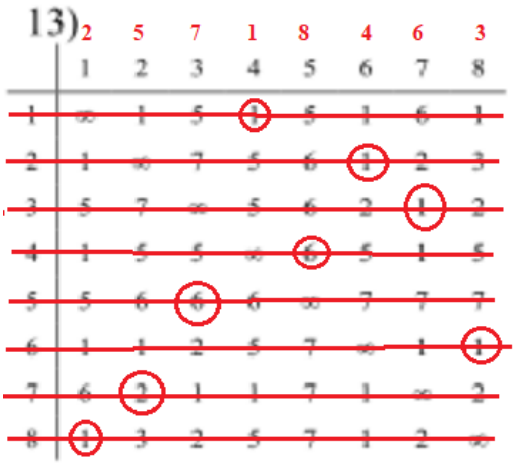
	1	2	3	4	5	6	7	8
1	∞	1	5	1	5	1	6	1
2	1	∞	7	5	6	1	2	3
3	5	7	∞	5	6	2	1	2
4	1	5	5	∞	6	5	1	5
5	5	6	6	6	∞	7	7	7
6	1	1	2	5	7	∞	1	1
7	6	2	1	1	7	1	∞	2
8	1	3	2	5	7	1	2	∞



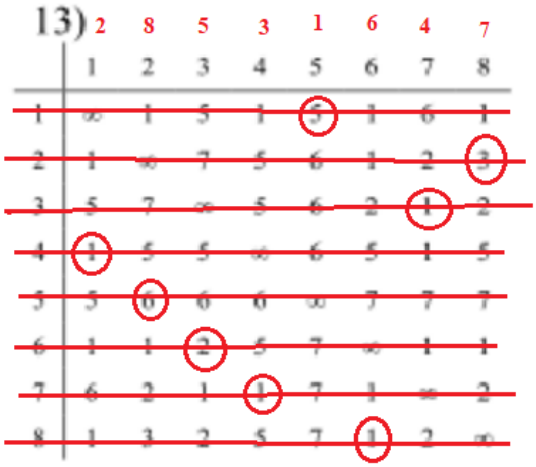




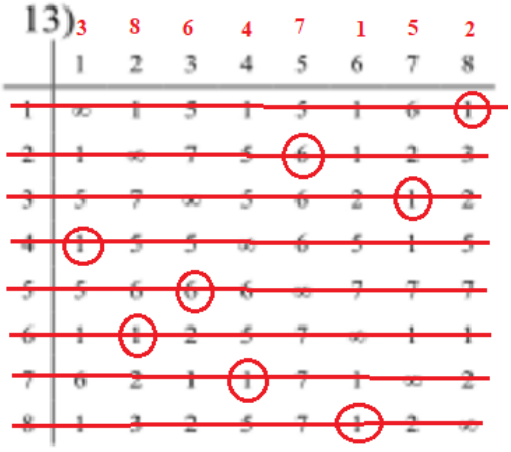
3->7->4->1->2->6->8->5->3
1 1 1 1 1 1 7 6 19



4->1->8->6->2->7->3->5->4
1 1 1 1 2 1 6 6 19



5->1->4->7->3->6->8->2->5
5 1 1 1 3 1 3 6 21



6->8->1->4->7->3->5->2->1
1 1 1 1 1 6 6 1 18

Перевіримо програмно:

```
#include <iostream>
#include <string>
#include <fstream>

using namespace std;

ifstream fin;
int** Data() {
	int count = 8;
	string str;
	fin.open("MyFile2.txt");
	int** arr;
	arr = new int* [count];
	str = "";
	for (int i = 0; i < count; i++)
		arr[i] = new int[count];

	for (int i = 0; i < count; i++)
	{
		for (int j = 0; j < count; j++)
			arr[i][j] = 0;
	}
```

```
	for (int i = 0; i < count; i++)
	{
		for (int j = i + 1; j < count; j++)
		{
			getline(fin, str);
			arr[i][j] = atoi(str.c_str());
			arr[j][i] = atoi(str.c_str());
		}
	}
	fin.close();

	return arr;
}
struct mass
{
	int mas[9];
};
bool comp(int* arr, int count)
{
	int* mas = new int[count];

	for (int i = 0; i < count; i++)
	{
		mas[i] = count - i;
	}

	for (int i = 0; i < count; i++)
	{
		if (mas[i] != arr[i])
		{
			return true;
		}
		else
		{
			continue;
		}
	}
	return false;
}
bool repeat(int* mas, int size)
{
	bool k = true;

	for (int i = 0; i < size; i++)
	{
		for (int j = 0; j < size; j++)
		{
			if (mas[i] == mas[j] && i != j)
			{
				return false;
			}
		}
	}

	return true;
}
int way(int** matrix, int* arr)
{
	int count = 0;

	for (int i = 0; i < 7; i++)
	{
		count += matrix[arr[i] - 1][arr[i + 1] - 1];
	}

	count += matrix[arr[7] - 1][arr[0] - 1];
	return count;
}
```

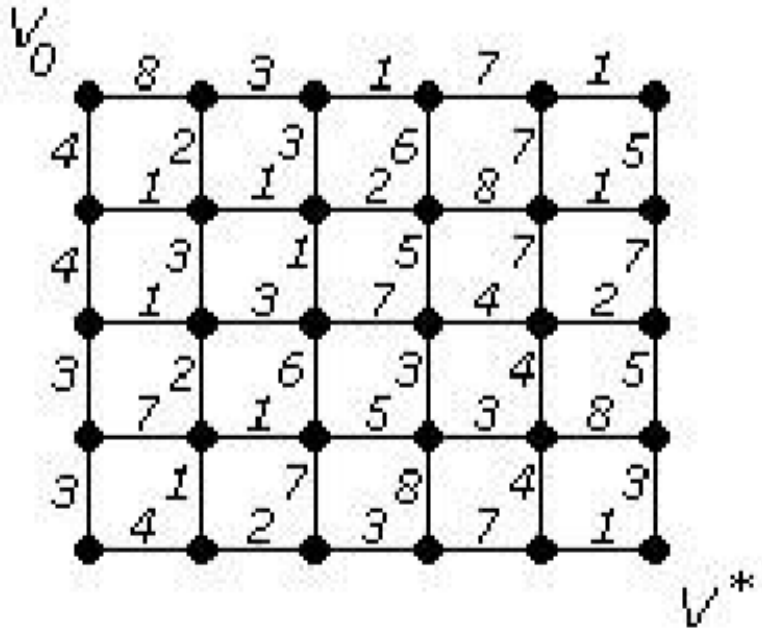
```
int main() {
	setlocale(LC_ALL, "Ukrainian");
	int const count = 8;
	int** arr;
	arr = Data();
	int* mas = new int[count];
	int* minmas = new int[9];
	int variant = count - 1;
	bool k = true;
	int min = 99999;
	int lenght = 0;
	int hm = 0;


	for (int i = 0; i < count; i++)
	{
		mas[i] = 1;
		minmas[i] = 1;
	}

	while (comp(mas, count))
	{
		while (mas[variant] != count)
		{
			mas[variant]++;

			if (repeat(mas, count))
			{
				lenght = way(arr, mas);
				if (lenght < min)
				{
					min = lenght;
					hm = 1;
				}
				if (lenght == min)
				{
					hm++;
				}
			}
		}

		while (mas[variant] == count)
		{
			mas[variant] = 1;
			variant--;
		}
		mas[variant]++;

		if (repeat(mas, count))
		{
			lenght = way(arr, mas);
			if (lenght < min)
			{
				min = lenght;
				hm = 1;
			}
			if (lenght == min)
			{
				hm++;
			}

		}
		variant = count - 1;
	}

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

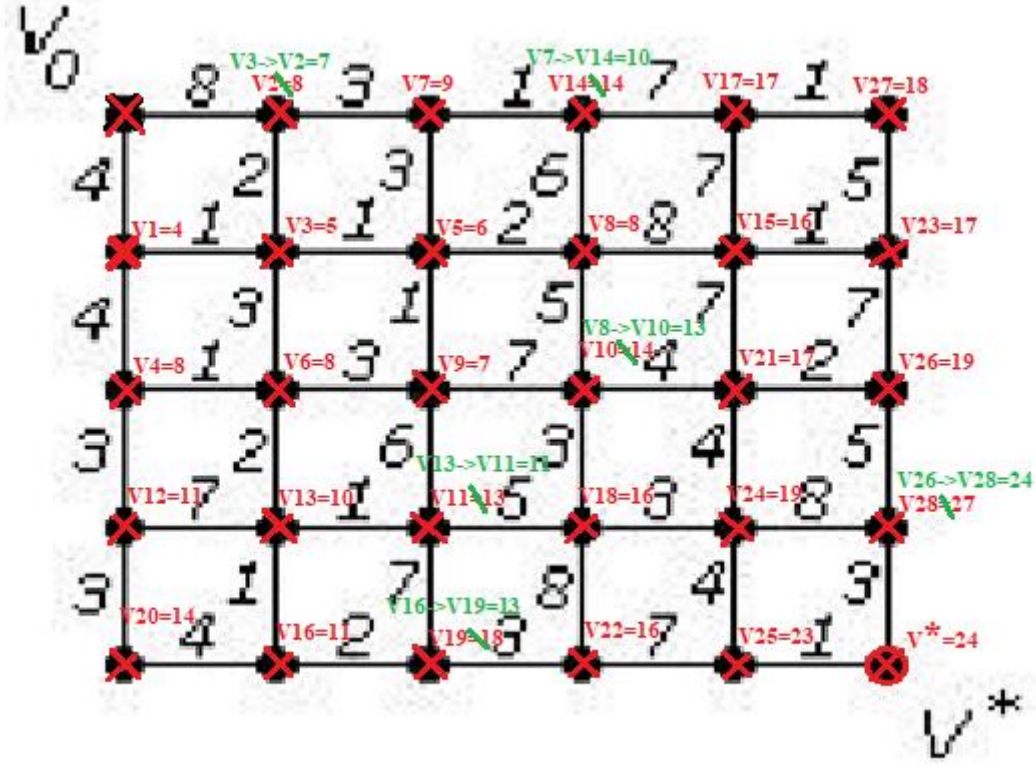
```
		mas[i] = 1;
		minmas[i] = 1;
	}
	mass* rez = new mass[hm];
	int iter = 0;

	while (comp(mas, count))
	{
		while (mas[variant] != count)
		{
			mas[variant]++;

			if (repeat(mas, count))
			{
				lenght = way(arr, mas);

				if (lenght == min)
				{
					for (int i = 0; i < count; i++)
					{
						rez[iter].mas[i] = mas[i];
					}
					rez[iter].mas[count] = mas[0];
					iter++;
				}
			}
		}
		while (mas[variant] == count)
		{
			mas[variant] = 1;
			variant--;
		}
		mas[variant]++;

		if (repeat(mas, count))
		{
			lenght = way(arr, mas);

			if (lenght == min)
			{
				for (int i = 0; i < count; i++)
				{
					rez[iter].mas[i] = mas[i];
				}
				rez[iter].mas[count] = mas[0];
				iter++;
			}

		}
		variant = count - 1;
	}

	cout << "Шляхи: " << endl;

	for (int i = 0; i < iter - 1; i++)
	{
		for (int j = 0; j <= count; j++)
		{
			if (j != 0)
			{
				cout << "--> ";
			}
			cout << rez[i].mas[j] << " ";
		}
		cout << endl;
	}
	cout << "Мінімальна довжина = " << min;
}
```

```
Шляхи:
1 --> 2 --> 6 --> 8 --> 3 --> 7 --> 4 --> 5 --> 1
1 --> 4 --> 7 --> 3 --> 5 --> 2 --> 6 --> 8 --> 1
1 --> 4 --> 7 --> 3 --> 8 --> 6 --> 2 --> 5 --> 1
1 --> 5 --> 2 --> 6 --> 8 --> 3 --> 7 --> 4 --> 1
1 --> 5 --> 4 --> 7 --> 3 --> 8 --> 6 --> 2 --> 1
1 --> 8 --> 6 --> 2 --> 5 --> 3 --> 7 --> 4 --> 1
2 --> 1 --> 5 --> 4 --> 7 --> 3 --> 8 --> 6 --> 2
2 --> 5 --> 1 --> 4 --> 7 --> 3 --> 8 --> 6 --> 2
2 --> 5 --> 3 --> 7 --> 4 --> 1 --> 8 --> 6 --> 2
2 --> 6 --> 8 --> 1 --> 4 --> 7 --> 3 --> 5 --> 2
2 --> 6 --> 8 --> 3 --> 7 --> 4 --> 1 --> 5 --> 2
2 --> 6 --> 8 --> 3 --> 7 --> 4 --> 5 --> 1 --> 2
3 --> 5 --> 2 --> 6 --> 8 --> 1 --> 4 --> 7 --> 3
3 --> 7 --> 4 --> 1 --> 5 --> 2 --> 6 --> 8 --> 3
3 --> 7 --> 4 --> 1 --> 8 --> 6 --> 2 --> 5 --> 3
3 --> 7 --> 4 --> 5 --> 1 --> 2 --> 6 --> 8 --> 3
3 --> 8 --> 6 --> 2 --> 1 --> 5 --> 4 --> 7 --> 3
3 --> 8 --> 6 --> 2 --> 5 --> 1 --> 4 --> 7 --> 3
4 --> 1 --> 5 --> 2 --> 6 --> 8 --> 3 --> 7 --> 4
4 --> 1 --> 8 --> 6 --> 2 --> 5 --> 3 --> 7 --> 4
4 --> 5 --> 1 --> 2 --> 6 --> 8 --> 3 --> 7 --> 4
4 --> 7 --> 3 --> 5 --> 2 --> 6 --> 8 --> 1 --> 4
4 --> 7 --> 3 --> 8 --> 6 --> 2 --> 1 --> 5 --> 4
4 --> 7 --> 3 --> 8 --> 6 --> 2 --> 5 --> 1 --> 4
5 --> 1 --> 2 --> 6 --> 8 --> 3 --> 7 --> 4 --> 5
5 --> 1 --> 4 --> 7 --> 3 --> 8 --> 6 --> 2 --> 5
5 --> 2 --> 6 --> 8 --> 1 --> 4 --> 7 --> 3 --> 5
5 --> 2 --> 6 --> 8 --> 3 --> 7 --> 4 --> 1 --> 5
5 --> 3 --> 7 --> 4 --> 1 --> 8 --> 6 --> 2 --> 5
5 --> 4 --> 7 --> 3 --> 8 --> 6 --> 2 --> 1 --> 5
6 --> 2 --> 1 --> 5 --> 4 --> 7 --> 3 --> 8 --> 6
6 --> 2 --> 5 --> 1 --> 4 --> 7 --> 3 --> 8 --> 6
6 --> 2 --> 5 --> 3 --> 7 --> 4 --> 1 --> 8 --> 6
6 --> 8 --> 1 --> 4 --> 7 --> 3 --> 5 --> 2 --> 6
6 --> 8 --> 3 --> 7 --> 4 --> 1 --> 5 --> 2 --> 6
6 --> 8 --> 3 --> 7 --> 4 --> 5 --> 1 --> 2 --> 6
7 --> 3 --> 5 --> 2 --> 6 --> 8 --> 1 --> 4 --> 7
7 --> 3 --> 8 --> 6 --> 2 --> 1 --> 5 --> 4 --> 7
7 --> 3 --> 8 --> 6 --> 2 --> 5 --> 1 --> 4 --> 7
7 --> 4 --> 1 --> 5 --> 2 --> 6 --> 8 --> 3 --> 7
7 --> 4 --> 1 --> 8 --> 6 --> 2 --> 5 --> 3 --> 7
7 --> 4 --> 5 --> 1 --> 2 --> 6 --> 8 --> 3 --> 7
8 --> 1 --> 4 --> 7 --> 3 --> 5 --> 2 --> 6 --> 8
8 --> 3 --> 7 --> 4 --> 1 --> 5 --> 2 --> 6 --> 8
8 --> 3 --> 7 --> 4 --> 5 --> 1 --> 2 --> 6 --> 8
8 --> 6 --> 2 --> 1 --> 5 --> 4 --> 7 --> 3 --> 8
8 --> 6 --> 2 --> 5 --> 1 --> 4 --> 7 --> 3 --> 8
Мінімальна довжина = 18
```

Завдання № 7

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





Відстань з V0 до V* = 24;

Шлях 1: V0 → V1 → V3 → V5 → V8 → V10 → V18 → V24 → V25 → V*

Шлях 2: V0 → V1 → V3 → V6 → V13 → V11 → V18 → V24 → V25 → V*

Шлях 3: V0 → V1 → V3 → V6 → V13 → V16 → V19 → V22 → V25 → V*

Перевіримо програмно:

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <fstream>
#include <locale>
using namespace std;
int main()
{
	ifstream fin("MyFile.txt");
	setlocale(LC_ALL, "Ukrainian");
	int versh, rebra;
	fin >> versh >> rebra;
	const int SIZE = 30;
	int matrix[SIZE][SIZE]; // матриця зв'язків
	int distance[SIZE]; // мінімальна відстань
	int visited[SIZE];  // чи відвідані вершни
	int dis, top, min;
	int begin_index = 0;


	for (int i = 0; i < versh; i++)     // Ініціалізація матриці зв'язків
	{
		distance[i] = 99999;
		visited[i] = 0;
		for (int j = 0; j < versh; j++)
		{
			matrix[i][j] = 0;
			matrix[j][i] = 0;
		}
	}
	for (int i = 0;i < rebra;i++) {
		int v1, v2, dis;
		fin >> v1 >> v2 >> dis;
		matrix[v1 - 1][v2 - 1] = dis;
		matrix[v2 - 1][v1 - 1] = dis;
	}

	/*for (int i = 0; i < versh; i++)     // Вивід матриці
	{
		for (int j = 0; j < versh; j++)
		{
			cout << " " << matrix[i][j];
		}
		cout << endl;
	} */
	cout <<
"\n\n++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n\n"
;

	distance[0] = 0;
	do {
		top = 99999;
		min = 99999;
		for (int i = 0;i < versh;i++)
		{
			if (visited[i] == 0 && distance[i] < min)
			{
```

```
                        min = distance[i];
                        top = i;
                }

        }
        if (top != 99999)
        {
                for (int i = 0; i < versh; i++)
                {
                        if (matrix[top][i] > 0)
                        {
                                dis = min + matrix[top][i];
                                if (dis < distance[i])
                                {
                                        distance[i] = dis;
                                }
                        }
                }
                visited[top] = 1;
        }
} while (top < 99999);

int end = versh - 1;
int waight = distance[end];
int way[30];
way[0] = versh - 1;
int k = 1;
while (end != 0)
{
        for (int i = 0;i < versh;i++)
        {
                if (matrix[end][i] > 0)
                {
                        if (distance[i] == waight - matrix[end][i])
                        {
                                waight = distance[i];
                                way[k] = i;
                                end = i;
                                k++;
                        }
                }
        }
}
for (int i = 0;i < versh;i++)
{
        cout <<"Відстань до V"<<i<<" = "<< distance[i] << endl;

}
cout << "\nНайкоротший шлях з V0 до V29: ";
for (int i = k;i > 0;i--)
{
        if (i - 1 > 0)
                cout << way[i - 1] << " -> ";
        else
                cout << way[i - 1];
}
cout << "\n\nДовжина шляху: " << distance[versh - 1] << endl;
cout <<
"\n\n++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n\n"
;
}
```

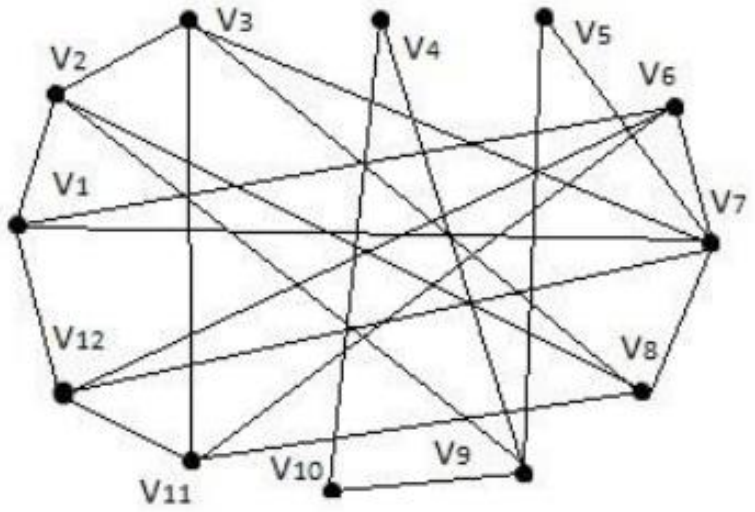
Результат:

```
Найкоротший шлях з V0 до V29: 0 -> 6 -> 7 -> 8 -> 9 -> 15 -> 21 -> 22 -> 28 -> 29

Довжина шляху: 24
```

Завдання № 8

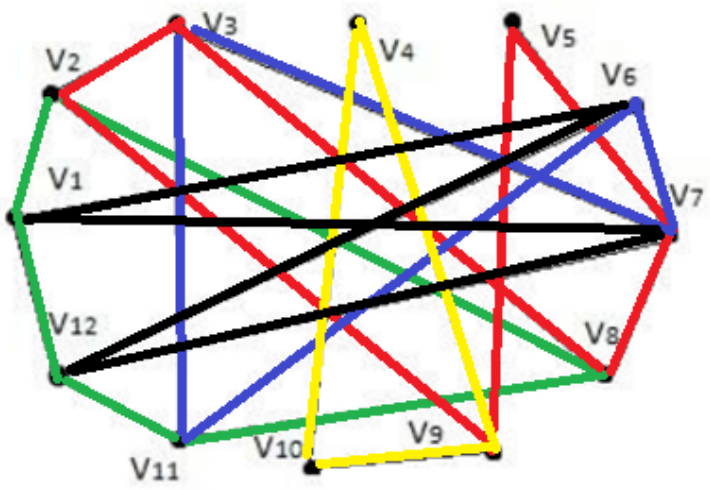
Знайти ейлеровий цикл в ейлеровому графі двома методами: а) Флері; б) елементарних циклів.

а) Флері:



V1V2 → V2V3 → V3V7 → V7V8 → V8V11 → V11V12 → V12V1 → V1V6 → V6V7 → V7V12→ V12V6 → V6V11 → V11V3 → V3V8 → V8V2 → V2V9 → V9V10 → V10V4→ V4V9→ V9V5→ V5V7→ V7V1

б) елементарних циклів:



R	**1, 6, 12, 7, 1, 2 , 3, 8, 7, 6, 11, 3, 7, 5, 9, 4, 10, 9, 2 , 8, 11, 12, 1**
R1	**1, 2, 8, 11, 12, 1**
R2	**2 , 3, 8, 7, 5, 9, 2**
R3	**7, 6, 11, 3, 7,**
R4	**9, 4, 10, 9,**
R5	**1, 6, 12, 7, 1**

Завдання №9

Спростити формули (привести їх до скороченої ДНФ).

13. $x\bar{y} \vee x\bar{z} \vee z$

Зведемо до скороченої ДНФ за допомогою карт Карно.

Z	0	0	1	1
y\ **X**	0	1	1	0
0	0	1	1	1
1	0	1	1	1

У нас виходить 2 групи. В результаті чого, скороченою диз'юнктивною нормальною формою буде: X∪Z