МІНІСТЕРСТВО ОСВІТИ І НАУКИ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра систем штучного інтелекту

Звіт

Розрахункова робота

з дисципліни «Дискретна математика»

Варіант № 17

Виконав:

Студент групи КН-113

Бондар А.-А.

Викладач:

Мельникова Н. І.

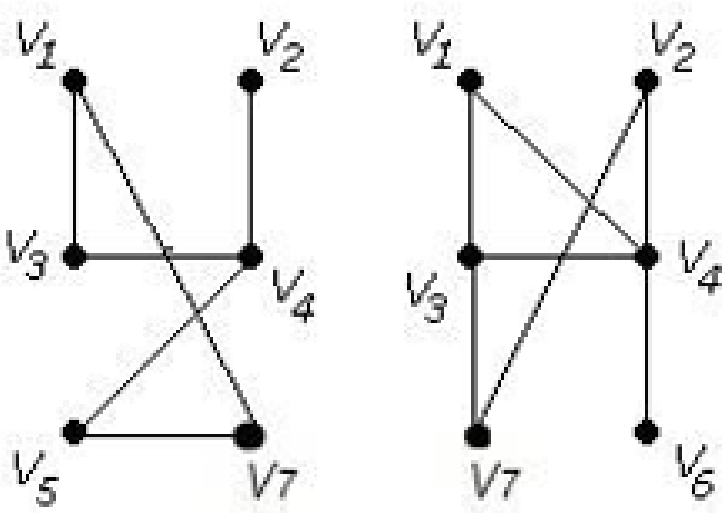
Львів-2019р.

Індивідуальні завдання

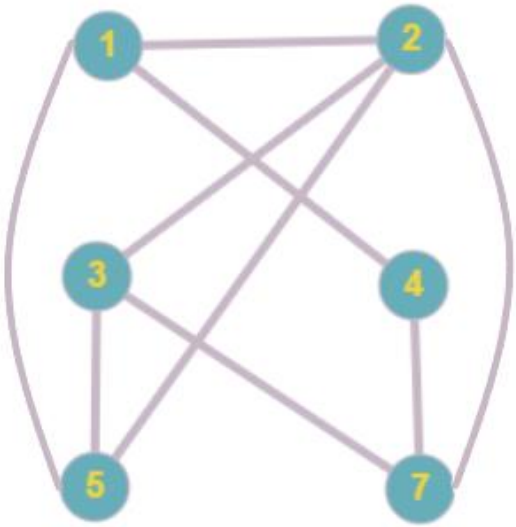
Завдання № 1.

Виконати наступні операції над графами

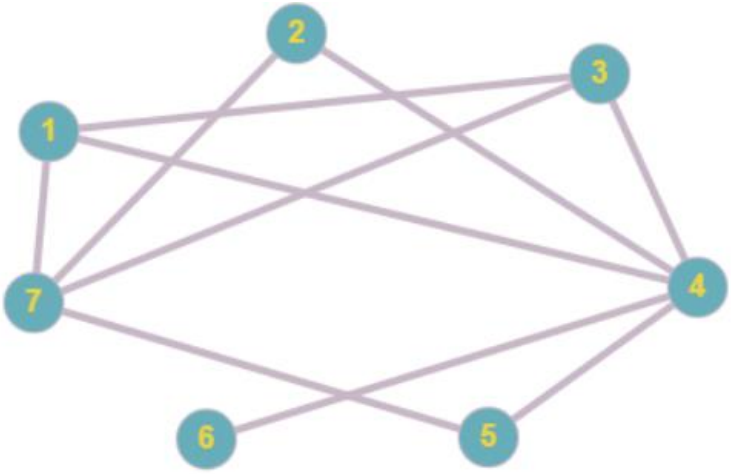
1) знайти доповнення до першого графу;
2) об’єднання графів;
3) кільцеву суму G1 та G2 (G1+G2);
4) розмножити вершину у другому графі;
5) виділити підграф А - що складається з 3-х вершин в G1;
6) добуток графів.



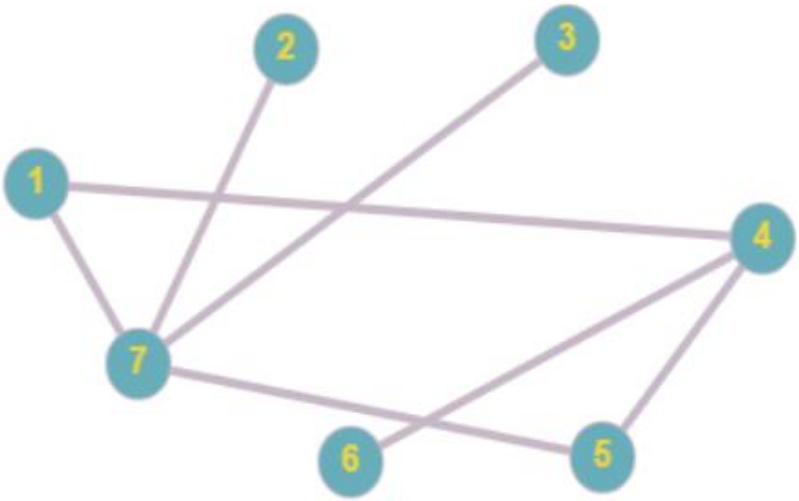
1) Доповнення до першого графу:



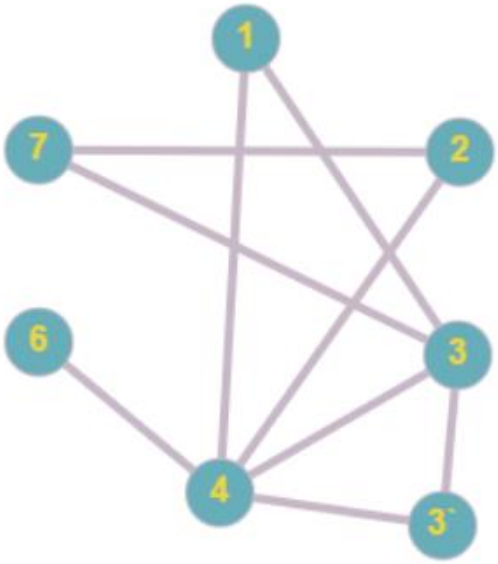
2) Об’єднання графів:



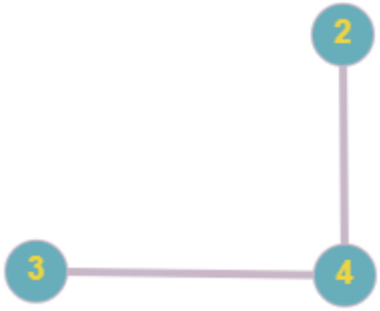
3) Кільцева суму G1 та G2 (G1+G2):



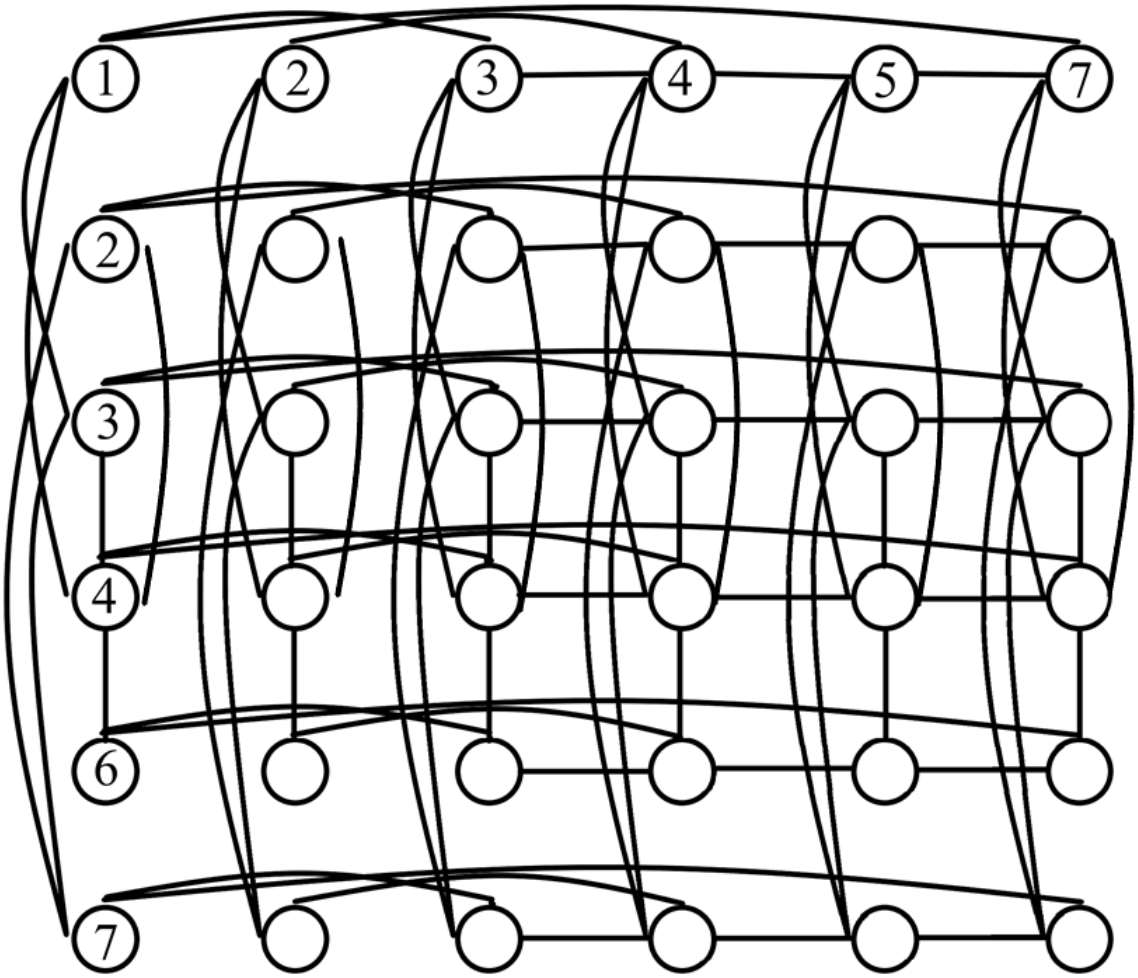
4) Розмноження вершини у другому графі:



5) Підграф A - що складається з 3-х вершин в G1:

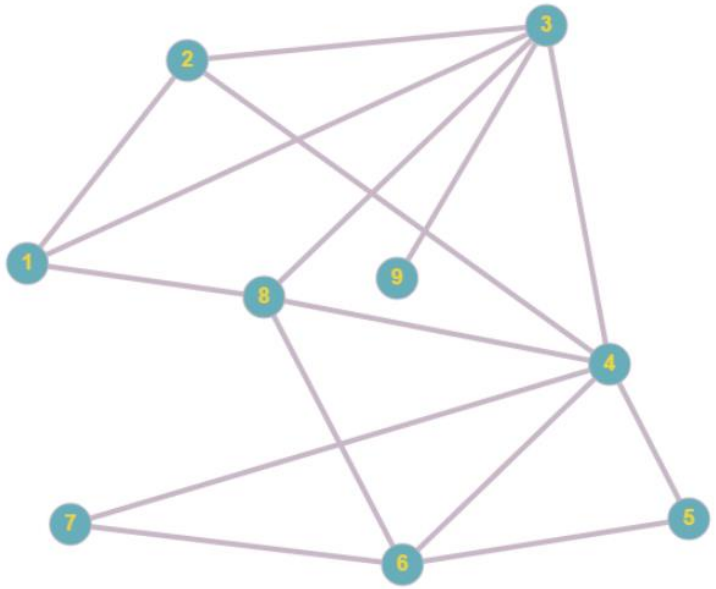


6) Добуток графів:



Завдання № 2.

Скласти таблицю суміжності для орграфа.



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

Завдання № 3.

Для графа з другого завдання знайти діаметр.

Діаметр даного графа дорівнює **3**, оскільки цьому дорівнює максимальна відстань найкоротшого шлях між двома вершинами (у цьому випадку V1 і V5).

Завдання № 4

Для графа з другого завдання виконати обхід дерева вглиб

Вершина	**DFS - номер**	**Вміст стеку**
a	1	a
b	2	ab
c	3	abc
i	4	abci
-	-	abc
h	5	abch
d	6	abchd
e	7	abchde
f	8	abchdef
g	9	abchdefg
-	-	abchdef
-	-	abchde
-	-	abchd
-	-	abch
-	-	abc
-	-	ab
-	-	a
-	-	∅

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

struct verh{
    bool dfs = false;
};
struct koctb{
    int ver1;
    int ver2;
};

int leng(string str){
    int i = 0;
    while (str[i] != '\0'){
        i++;
    }
    return i;
}
int correctly(int m, int n){
    int c = 0;
    bool count = false;
    string str;
    stringstream ss;
    while (count == false){
        cin >> str;
        for (int i = 0; i < leng(str); i++){
            if (!isdigit(str[i])){
                if (i == 0 && str[i] == '-'){
                    count = true;
                }
                else{
                    count = false;
                    break;
                }
            }
            else{
                count = true;
            }
        }
        if (count == true){
            ss << str;
            ss >> c;
            ss.clear();
            if (c < m || c > n){
                count = false;
            }
            else{
                count = true;
            }
        }
        if (count == false){
            cout << "Error! Try again!" << endl;
        }
        str = "";
    }
    return c;
}

void input(koctb* reb, int n, int m){
    for (int i = 0; i < n; i++){
        cout << "вершина 1 належить ребру " << i + 1 << ": ";
        reb[i].ver1 = correctly(1, m);
        cout << "вершина 2 належить ребру " << i + 1 << ": ";;
        reb[i].ver2 = correctly(1, m);
        cout << endl;
    }
```

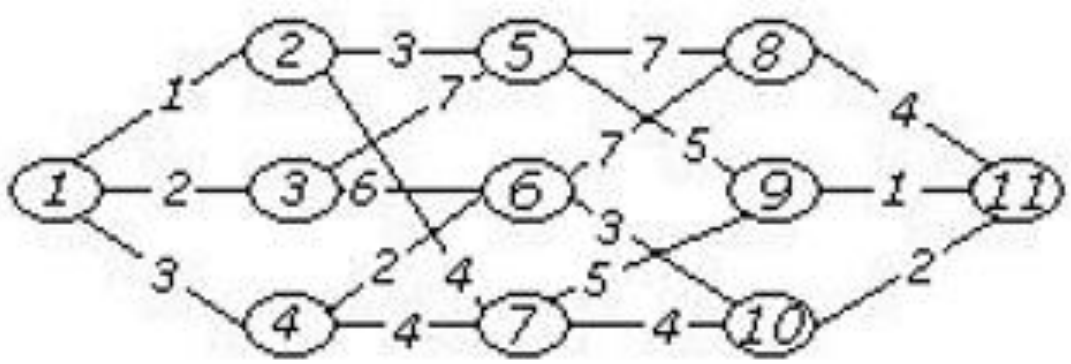
```
}

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

    cout << "Введіть к-сть ребер графу: ";
    n = correctly(1, 1000);
    cout << "Введіть к-сть вершин графу: ";
    m = correctly(1, 1000);
    cout << endl;
    int* vec = new int[m];
    koctb* reb = new koctb[n];
    verh* v = new verh[m];
    input(reb, n, m);
    cout << "Починаэмо з вершини? ";
    begin = correctly(1, m);
    vec[0] = begin;
    v[begin - 1].dfs = true;
    coun++;

    while (coun != 0){
        for (int i = 0; i < n; i++){
            if ((vec[coun - 1] == reb[i].ver1 && v[reb[i].ver2 - 1].dfs == false) ||
(vec[coun - 1] == reb[i].ver2 && v[reb[i].ver1 - 1].dfs == false)){
                t++;
            }
        }
        if (t == 0){
            coun--;
        }
        else{
            for (int i = 0; i < n; i++){
                if (vec[coun - 1] == reb[i].ver2 && v[reb[i].ver1 - 1].dfs == false){
                    vec[coun] = reb[i].ver1;
                    v[reb[i].ver1 - 1].dfs = true;
                    coun++;
                    goto point;
                }
                if (vec[coun - 1] == reb[i].ver1 && v[reb[i].ver2 - 1].dfs == false){
                    vec[coun] = reb[i].ver2;
                    v[reb[i].ver2 - 1].dfs = true;

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

```
Введіть к-сть ребер графу: 15
Введіть к-сть вершин графу: 9

вершина 1 належить ребру 1: 1
вершина 2 належить ребру 1: 2

вершина 1 належить ребру 2: 1
вершина 2 належить ребру 2: 3

вершина 1 належить ребру 3: 1
вершина 2 належить ребру 3: 8

вершина 1 належить ребру 4: 2
вершина 2 належить ребру 4: 3

вершина 1 належить ребру 5: 2
вершина 2 належить ребру 5: 4

вершина 1 належить ребру 6: 3
вершина 2 належить ребру 6: 8

вершина 1 належить ребру 7: 3
вершина 2 належить ребру 7: 9

вершина 1 належить ребру 8: 3
вершина 2 належить ребру 8: 4

вершина 1 належить ребру 9: 8
вершина 2 належить ребру 9: 4

вершина 1 належить ребру 10: 8
вершина 2 належить ребру 10: 6

вершина 1 належить ребру 11: 4
вершина 2 належить ребру 11: 5

вершина 1 належить ребру 12: 4
вершина 2 належить ребру 12: 6

вершина 1 належить ребру 13: 4
вершина 2 належить ребру 13: 7

вершина 1 належить ребру 14: 6
вершина 2 належить ребру 14: 7

вершина 1 належить ребру 15: 5
вершина 2 належить ребру 15: 6
```

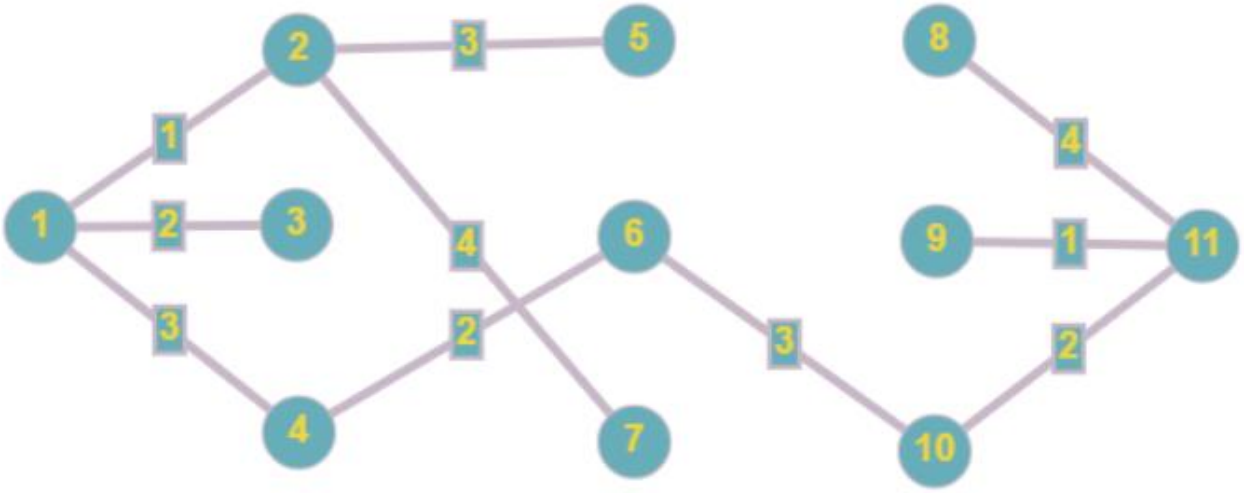
```
Починаэмо з вершини? 1
1 2
1 2 3
1 2 3 8
1 2 3 8 4
1 2 3 8 4 5
1 2 3 8 4 5 6
1 2 3 8 4 5 6 7
1 2 3 8 4 5 6
1 2 3 8 4 5
1 2 3 8 4
1 2 3 8
1 2 3
1 2 3 9
1 2 3
1 2
1
```

Завдання № 5.

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



Краскала



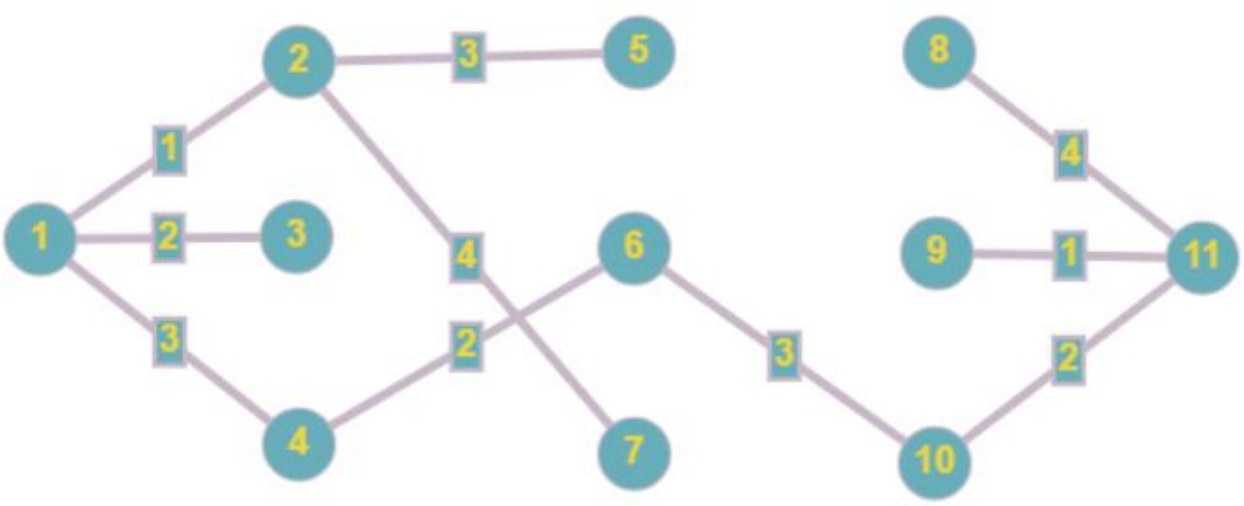

```
#include <stdio.h>
#include <iostream>
#include <stdlib.h>
using namespace std;

const int q = 11;
int BuildTrees(int n, int A[q][q]);
void DeleteDuplicates(int n, int A[q][q]);
int InDifferTrees(int n, int A[q][q], int first, int second);
void AddToTheTree(int n, int A[q][q], int first, int second);

int main()
{
	setlocale(LC_ALL, "Ukrainian");
	int A[11][11] =
	{ 0, 1, 2, 3, 0, 0, 0, 0, 0, 0, 0,
	  1, 0, 0, 0, 3, 0, 4, 0, 0, 0, 0,
	  2, 0, 0, 0, 7, 6, 0, 0, 0, 0, 0,
	  3, 0, 0, 0, 0, 2, 4, 0, 0, 0, 0,
	  0, 3, 7, 0, 0, 0, 0, 7, 5, 0, 0,
	  0, 0, 6, 2, 0, 0, 0, 7, 0, 3, 0,
	  0, 4, 0, 4, 0, 0, 0, 0, 5, 4, 0,
	  0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 4,
	  0, 0, 0, 0, 5, 0, 5, 0, 0, 0, 1,
	  0, 0, 0, 0, 0, 3, 4, 0, 0, 0, 2,
```

```
          0, 0, 0, 0, 0, 0, 0, 4, 1, 2, 0 };

        DeleteDuplicates(11, A);
        for (int i = 1; i <= 7; i++){
                cout << "\nВузли з вагою: " << i << ": ";
                for (int j = 1; j <= 11; j++){
                        for (int k = 1; k <= 11; k++){
                                if (A[j - 1][k - 1] == i){
                                        cout << " " << j << "-" << k;;
                                }
                        }
                }
        }
        cout << "\n";

        int B[11][11];
        BuildTrees(11, B);
        cout << "\n\nНове дерево: ";//вага 7 - максимальна вага
        for (int i = 1; i <= 7; i++){
                //перший вузол
                for (int j = 1; j <= 11; j++){
                        //другий вузол
                        for (int k = 1; k <= 11; k++){
                                if (A[j - 1][k - 1] == i && InDifferTrees(11, B, j, k)){
                                        AddToTheTree(11, B, j, k);
                                        cout << " " << j << "-" << k;
                                }
                        }
                }
        }
        return 0;
}

void DeleteDuplicates(int n, int A[q][q]) {
        for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                        if (j < i) {
                                A[i][j] = 0;
                        }
                }
        }
}

int BuildTrees(int n, int A[q][q]) {
        for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                        A[i][j] = 0;
                }
        }
        for (int i = 0; i < n; i++) {
                A[i][i] = i + 1;
        }
        return A[n][n];
}
//Перевірте відсортовані вузли та додайте до дерева

void AddToTheTree(int n, int A[q][q], int first, int second) {
        int scndLine;
        for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                        if (A[i][j] == second) {
                                scndLine = i;
                        }
                }
        }
        for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                        if (A[i][j] == first) {
```

```
                        for (int k = 0; k < n; k++) {
                                if (A[scndLine][k]) {
                                        A[i][k] = A[scndLine][k];
                                        A[scndLine][k] = 0;
                                }
                        }
                }
        }
    }
}

int InDifferTrees(int n, int A[q][q], int first, int second){
    int temp1, temp2;
    //Лінія
    for (int i = 0; i < n; i++){
        temp1 = 0;
        temp2 = 0;
        //перший елемент
        for (int j = 0; j < n; j++){
            if (A[i][j] == first){
                temp1 = 1;
            }
        }
        //другий елемент
        for (int k = 0; k < n; k++){
            if (A[i][k] == second){
                temp2 = 1;
            }
        }
        if (temp1 && temp2){
            return 0;
        }
    }
    return 1;
}
```

```
Вузли з вагою: 1:  1-2 9-11
Вузли з вагою: 2:  1-3 4-6 10-11
Вузли з вагою: 3:  1-4 2-5 6-10
Вузли з вагою: 4:  2-7 4-7 7-10 8-11
Вузли з вагою: 5:  5-9 7-9
Вузли з вагою: 6:  3-6
Вузли з вагою: 7:  3-5 5-8 6-8


Нове дерево:  1-2 9-11 1-3 4-6 10-11 1-4 2-5 6-10 2-7 8-11
```

Прима




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

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

int main()
{
	int size;
	cout << "Enter amount of vertices ";
	cin >> size;
	int** adjacencyarr = new int* [size];
	bool* vertex = new bool[size];
	cout << "Enter adjacency matrix \n";
	cout << "  ";
	for (int i = 0; i < size; i++) { cout << setw(3) << i + 1; adjacencyarr[i] = new
int[size]; vertex[i] = 0; }
	vertex[0] = 1;
	cout << endl;
	for (int i = 0; i < size; i++) {
		cout << setw(2) << i + 1;
		for (int j = 0; j < i; j++) { cout << setw(3) << adjacencyarr[j][i]; }
		cout << "  -";
		for (int j = i + 1; j < size; j++) { cin >> adjacencyarr[i][j]; }
	}
	cout << endl;
	int min, minI, minJ;
	for (int n = 0; n < size - 1; n++) {
		min = 100, minI = 0, minJ = 0;
		for (int i = 0; i < size; i++) {
			for (int j = i + 1; j < size; j++) {
				if (adjacencyarr[i][j] < min && adjacencyarr[i][j] != 0 &&
vertex[i] != vertex[j]) {
```

```
                                min = adjacencyarr[i][j];
                                minI = i, minJ = j;
                            }
                        }
                    }
                    vertex[minI] = 1; vertex[minJ] = 1;
                    cout << n + 1 << ")conected " << minI + 1 << "-" << minJ + 1 << endl;
            }
}
```

```
Enter amount of vertices 11
Enter adjacency matrix
    1  2  3  4  5  6  7  8  9 10 11
 1  -  1  2  3  0  0  0  0  0  0  0
 2  1  -  0  0  3  0  4  0  0  0  0
 3  2  0  -  0  7  6  0  0  0  0  0
 4  3  0  0  -  0  2  4  0  0  0  0
 5  0  3  7  0  -  0  0  7  5  0  0
 6  0  0  6  2  0  -  0  7  0  3  0
 7  0  4  0  4  0  0  -  0  5  4  0
 8  0  0  0  0  7  7  0  -  0  0  4
 9  0  0  0  0  5  0  5  0  -  0  1
10  0  0  0  0  0  3  4  0  0  -  2
11  0  0  0  0  0  0  0  4  1  2  -
1)conected 1-2
2)conected 1-3
3)conected 1-4
4)conected 4-6
5)conected 2-5
6)conected 6-10
7)conected 10-11
8)conected 9-11
9)conected 2-7
10)conected 8-11
```

Завдання № 6.

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

17)

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

№1

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

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

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

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

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

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

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

Шлях: 7→2→5→1→8→6→4→3→7

Вага: 1+1+1+3+2+5+7=20

№2

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

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

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

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

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

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

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

Шлях: 5→1→6→7→4→8→2→3→5

Вага: 1+3+1+1+2+5+5=18

Цей шлях є найменшим , так як циклів є велика кількість перевіримо усі можливі програмно :

```
8-> 2-> 1-> 6-> 7-> 3-> 4-> 5-> 8 (41)
8-> 2-> 1-> 6-> 7-> 3-> 5-> 4-> 8 (37)
8-> 2-> 1-> 6-> 7-> 4-> 3-> 5-> 8 (36)
8-> 2-> 1-> 6-> 7-> 4-> 5-> 3-> 8 (34)
8-> 2-> 1-> 6-> 7-> 5-> 3-> 4-> 8 (37)
8-> 2-> 1-> 6-> 7-> 5-> 4-> 3-> 8 (39)
8-> 2-> 1-> 7-> 3-> 4-> 5-> 6-> 8 (40)
8-> 2-> 1-> 7-> 3-> 4-> 6-> 5-> 8 (43)
8-> 2-> 1-> 7-> 3-> 5-> 4-> 6-> 8 (39)
8-> 2-> 1-> 7-> 3-> 5-> 6-> 4-> 8 (39)
8-> 2-> 1-> 7-> 3-> 6-> 4-> 5-> 8 (43)
8-> 2-> 1-> 7-> 3-> 6-> 5-> 4-> 8 (40)
8-> 2-> 1-> 7-> 4-> 3-> 5-> 6-> 8 (35)
8-> 2-> 1-> 7-> 4-> 3-> 6-> 5-> 8 (39)
8-> 2-> 1-> 7-> 4-> 5-> 3-> 6-> 8 (35)
8-> 2-> 1-> 7-> 4-> 5-> 6-> 3-> 8 (37)
8-> 2-> 1-> 7-> 4-> 6-> 3-> 5-> 8 (38)
8-> 2-> 1-> 7-> 4-> 6-> 5-> 3-> 8 (36)
8-> 2-> 1-> 7-> 5-> 3-> 4-> 6-> 8 (39)
8-> 2-> 1-> 7-> 5-> 3-> 6-> 4-> 8 (39)
8-> 2-> 1-> 7-> 5-> 4-> 3-> 6-> 8 (40)
8-> 2-> 1-> 7-> 5-> 4-> 6-> 3-> 8 (41)
8-> 2-> 1-> 7-> 5-> 6-> 3-> 4-> 8 (40)
8-> 2-> 1-> 7-> 5-> 6-> 4-> 3-> 8 (41)
8-> 2-> 1-> 7-> 6-> 3-> 4-> 5-> 8 (39)
```

```
8-> 6-> 4-> 5-> 7-> 1-> 3-> 2-> 8 (36)
8-> 6-> 4-> 5-> 7-> 2-> 1-> 3-> 8 (37)
8-> 6-> 4-> 5-> 7-> 2-> 3-> 1-> 8 (37)
8-> 6-> 4-> 5-> 7-> 3-> 1-> 2-> 8 (43)
8-> 6-> 4-> 5-> 7-> 3-> 2-> 1-> 8 (43)
8-> 6-> 4-> 7-> 1-> 2-> 3-> 5-> 8 (33)
8-> 6-> 4-> 7-> 1-> 2-> 5-> 3-> 8 (28)
8-> 6-> 4-> 7-> 1-> 3-> 2-> 5-> 8 (27)
8-> 6-> 4-> 7-> 1-> 3-> 5-> 2-> 8 (28)
8-> 6-> 4-> 7-> 1-> 5-> 2-> 3-> 8 (21)
8-> 6-> 4-> 7-> 1-> 5-> 3-> 2-> 8 (27)
8-> 6-> 4-> 7-> 2-> 1-> 3-> 5-> 8 (34)
8-> 6-> 4-> 7-> 2-> 1-> 5-> 3-> 8 (28)
8-> 6-> 4-> 7-> 2-> 3-> 1-> 5-> 8 (27)
8-> 6-> 4-> 7-> 2-> 3-> 5-> 1-> 8 (27)
8-> 6-> 4-> 7-> 2-> 5-> 1-> 3-> 8 (22)
8-> 6-> 4-> 7-> 2-> 5-> 3-> 1-> 8 (22)
8-> 6-> 4-> 7-> 3-> 1-> 2-> 5-> 8 (34)
8-> 6-> 4-> 7-> 3-> 1-> 5-> 2-> 8 (28)
8-> 6-> 4-> 7-> 3-> 2-> 1-> 5-> 8 (33)
8-> 6-> 4-> 7-> 3-> 2-> 5-> 1-> 8 (33)
8-> 6-> 4-> 7-> 3-> 5-> 1-> 2-> 8 (34)
8-> 6-> 4-> 7-> 3-> 5-> 2-> 1-> 8 (34)
8-> 6-> 4-> 7-> 5-> 1-> 2-> 3-> 8 (31)
8-> 6-> 4-> 7-> 5-> 1-> 3-> 2-> 8 (31)
8-> 6-> 4-> 7-> 5-> 2-> 1-> 3-> 8 (32)
8-> 6-> 4-> 7-> 5-> 2-> 3-> 1-> 8 (32)
```

```
7-> 2-> 5-> 1-> 8-> 4-> 6-> 3-> 7 (27)
7-> 2-> 5-> 1-> 8-> 6-> 3-> 4-> 7 (23)
7-> 2-> 5-> 1-> 8-> 6-> 4-> 3-> 7 (27)
7-> 2-> 5-> 3-> 1-> 4-> 6-> 8-> 7 (30)
7-> 2-> 5-> 3-> 1-> 4-> 8-> 6-> 7 (26)
7-> 2-> 5-> 3-> 1-> 6-> 4-> 8-> 7 (27)
7-> 2-> 5-> 3-> 1-> 6-> 8-> 4-> 7 (23)
7-> 2-> 5-> 3-> 1-> 8-> 4-> 6-> 7 (26)
7-> 2-> 5-> 3-> 1-> 8-> 6-> 4-> 7 (26)
7-> 2-> 5-> 3-> 4-> 1-> 6-> 8-> 7 (29)
7-> 2-> 5-> 3-> 4-> 1-> 8-> 6-> 7 (28)
7-> 2-> 5-> 3-> 4-> 6-> 1-> 8-> 7 (29)
7-> 2-> 5-> 3-> 4-> 6-> 8-> 1-> 7 (29)
7-> 2-> 5-> 3-> 4-> 8-> 1-> 6-> 7 (25)
7-> 2-> 5-> 3-> 4-> 8-> 6-> 1-> 7 (25)
7-> 2-> 5-> 3-> 6-> 1-> 4-> 8-> 7 (29)
7-> 2-> 5-> 3-> 6-> 1-> 8-> 4-> 7 (25)
7-> 2-> 5-> 3-> 6-> 4-> 1-> 8-> 7 (32)
7-> 2-> 5-> 3-> 6-> 4-> 8-> 1-> 7 (32)
7-> 2-> 5-> 3-> 6-> 8-> 1-> 4-> 7 (28)
7-> 2-> 5-> 3-> 6-> 8-> 4-> 1-> 7 (28)
7-> 2-> 5-> 3-> 8-> 1-> 4-> 6-> 7 (29)
7-> 2-> 5-> 3-> 8-> 1-> 6-> 4-> 7 (26)
7-> 2-> 5-> 3-> 8-> 4-> 1-> 6-> 7 (26)
7-> 2-> 5-> 3-> 8-> 4-> 6-> 1-> 7 (26)
7-> 2-> 5-> 3-> 8-> 6-> 1-> 4-> 7 (26)
7-> 2-> 5-> 3-> 8-> 6-> 4-> 1-> 7 (26)
7-> 2-> 5-> 4-> 1-> 3-> 6-> 8-> 7 (31)
7-> 2-> 5-> 4-> 1-> 3-> 8-> 6-> 7 (28)
7-> 2-> 5-> 4-> 1-> 6-> 3-> 8-> 7 (31)
7-> 2-> 5-> 4-> 1-> 6-> 8-> 3-> 7 (31)
7-> 2-> 5-> 4-> 1-> 8-> 3-> 6-> 7 (30)
7-> 2-> 5-> 4-> 1-> 8-> 6-> 3-> 7 (33)
7-> 2-> 5-> 4-> 3-> 1-> 6-> 8-> 7 (28)
7-> 2-> 5-> 4-> 3-> 1-> 8-> 6-> 7 (27)
7-> 2-> 5-> 4-> 3-> 6-> 1-> 8-> 7 (30)
7-> 2-> 5-> 4-> 3-> 6-> 8-> 1-> 7 (30)
7-> 2-> 5-> 4-> 3-> 8-> 1-> 6-> 7 (27)
7-> 2-> 5-> 4-> 3-> 8-> 6-> 1-> 7 (27)
7-> 2-> 5-> 4-> 6-> 1-> 3-> 8-> 7 (29)
7-> 2-> 5-> 4-> 6-> 1-> 8-> 3-> 7 (31)
7-> 2-> 5-> 4-> 6-> 3-> 1-> 8-> 7 (31)
7-> 2-> 5-> 4-> 6-> 3-> 8-> 1-> 7 (31)
7-> 2-> 5-> 4-> 6-> 8-> 1-> 3-> 7 (31)
7-> 2-> 5-> 4-> 6-> 8-> 3-> 1-> 7 (31)
7-> 2-> 5-> 4-> 8-> 1-> 3-> 6-> 7 (27)
7-> 2-> 5-> 4-> 8-> 1-> 6-> 3-> 7 (30)
7-> 2-> 5-> 4-> 8-> 3-> 1-> 6-> 7 (25)
7-> 2-> 5-> 4-> 8-> 3-> 6-> 1-> 7 (25)
7-> 2-> 5-> 4-> 8-> 6-> 1-> 3-> 7 (28)
```

```
7-> 2-> 5-> 6-> 1-> 3-> 4-> 8-> 7 (28)
7-> 2-> 5-> 6-> 1-> 3-> 8-> 4-> 7 (25)
7-> 2-> 5-> 6-> 1-> 4-> 3-> 8-> 7 (31)
7-> 2-> 5-> 6-> 1-> 4-> 8-> 3-> 7 (31)
7-> 2-> 5-> 6-> 1-> 8-> 3-> 4-> 7 (27)
7-> 2-> 5-> 6-> 1-> 8-> 4-> 3-> 7 (30)
7-> 2-> 5-> 6-> 3-> 1-> 4-> 8-> 7 (31)
7-> 2-> 5-> 6-> 3-> 1-> 8-> 4-> 7 (27)
7-> 2-> 5-> 6-> 3-> 4-> 1-> 8-> 7 (33)
7-> 2-> 5-> 6-> 3-> 4-> 8-> 1-> 7 (33)
7-> 2-> 5-> 6-> 3-> 8-> 1-> 4-> 7 (30)
7-> 2-> 5-> 6-> 3-> 8-> 4-> 1-> 7 (30)
7-> 2-> 5-> 6-> 4-> 1-> 3-> 8-> 7 (32)
7-> 2-> 5-> 6-> 4-> 1-> 8-> 3-> 7 (34)
7-> 2-> 5-> 6-> 4-> 3-> 1-> 8-> 7 (31)
7-> 2-> 5-> 6-> 4-> 3-> 8-> 1-> 7 (31)
7-> 2-> 5-> 6-> 4-> 8-> 1-> 3-> 7 (31)
7-> 2-> 5-> 6-> 4-> 8-> 3-> 1-> 7 (31)
7-> 2-> 5-> 6-> 8-> 1-> 3-> 4-> 7 (27)
7-> 2-> 5-> 6-> 8-> 1-> 4-> 3-> 7 (33)
7-> 2-> 5-> 6-> 8-> 3-> 1-> 4-> 7 (28)
7-> 2-> 5-> 6-> 8-> 3-> 4-> 1-> 7 (28)
7-> 2-> 5-> 6-> 8-> 4-> 1-> 3-> 7 (31)
7-> 2-> 5-> 6-> 8-> 4-> 3-> 1-> 7 (31)
7-> 2-> 5-> 8-> 1-> 3-> 4-> 6-> 7 (30)
7-> 2-> 5-> 8-> 1-> 3-> 6-> 4-> 7 (30)
7-> 2-> 5-> 8-> 1-> 4-> 3-> 6-> 7 (32)
7-> 2-> 5-> 8-> 1-> 4-> 6-> 3-> 7 (36)
7-> 2-> 5-> 8-> 1-> 6-> 3-> 4-> 7 (29)
7-> 2-> 5-> 8-> 1-> 6-> 4-> 3-> 7 (33)
7-> 2-> 5-> 8-> 3-> 1-> 4-> 6-> 7 (31)
7-> 2-> 5-> 8-> 3-> 1-> 6-> 4-> 7 (28)
7-> 2-> 5-> 8-> 3-> 4-> 1-> 6-> 7 (30)
7-> 2-> 5-> 8-> 3-> 4-> 6-> 1-> 7 (30)
7-> 2-> 5-> 8-> 3-> 6-> 1-> 4-> 7 (30)
7-> 2-> 5-> 8-> 3-> 6-> 4-> 1-> 7 (30)
7-> 2-> 5-> 8-> 4-> 1-> 3-> 6-> 7 (30)
7-> 2-> 5-> 8-> 4-> 1-> 6-> 3-> 7 (33)
7-> 2-> 5-> 8-> 4-> 3-> 1-> 6-> 7 (27)
7-> 2-> 5-> 8-> 4-> 3-> 6-> 1-> 7 (27)
7-> 2-> 5-> 8-> 4-> 6-> 1-> 3-> 7 (31)
7-> 2-> 5-> 8-> 4-> 6-> 3-> 1-> 7 (31)
7-> 2-> 5-> 8-> 6-> 1-> 3-> 4-> 7 (27)
7-> 2-> 5-> 8-> 6-> 1-> 4-> 3-> 7 (33)
7-> 2-> 5-> 8-> 6-> 3-> 1-> 4-> 7 (30)
7-> 2-> 5-> 8-> 6-> 3-> 4-> 1-> 7 (30)
7-> 2-> 5-> 8-> 6-> 4-> 1-> 3-> 7 (34)
7-> 2-> 5-> 8-> 6-> 4-> 3-> 1-> 7 (34)
7-> 2-> 6-> 1-> 3-> 4-> 5-> 8-> 7 (37)
7-> 2-> 6-> 1-> 3-> 4-> 8-> 5-> 7 (37)
```

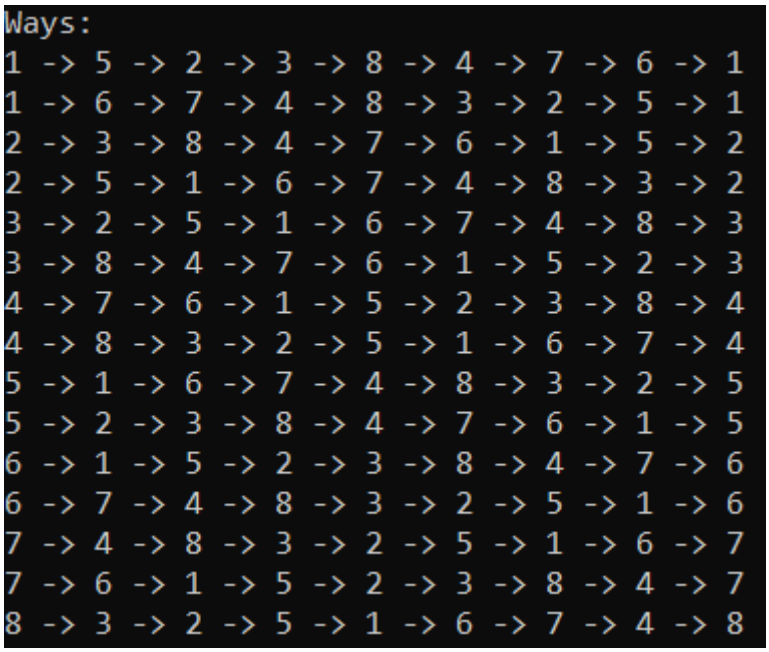
```
7-> 2-> 6-> 5-> 1-> 3-> 4-> 8-> 7 (31)
7-> 2-> 6-> 5-> 1-> 3-> 8-> 4-> 7 (28)
7-> 2-> 6-> 5-> 1-> 4-> 3-> 8-> 7 (34)
7-> 2-> 6-> 5-> 1-> 4-> 8-> 3-> 7 (34)
7-> 2-> 6-> 5-> 1-> 8-> 3-> 4-> 7 (30)
7-> 2-> 6-> 5-> 1-> 8-> 4-> 3-> 7 (33)
7-> 2-> 6-> 5-> 3-> 1-> 4-> 8-> 7 (36)
7-> 2-> 6-> 5-> 3-> 1-> 8-> 4-> 7 (32)
7-> 2-> 6-> 5-> 3-> 4-> 1-> 8-> 7 (38)
7-> 2-> 6-> 5-> 3-> 4-> 8-> 1-> 7 (38)
7-> 2-> 6-> 5-> 3-> 8-> 1-> 4-> 7 (35)
7-> 2-> 6-> 5-> 3-> 8-> 4-> 1-> 7 (35)
7-> 2-> 6-> 5-> 4-> 1-> 3-> 8-> 7 (38)
7-> 2-> 6-> 5-> 4-> 1-> 8-> 3-> 7 (40)
7-> 2-> 6-> 5-> 4-> 3-> 1-> 8-> 7 (37)
7-> 2-> 6-> 5-> 4-> 3-> 8-> 1-> 7 (37)
7-> 2-> 6-> 5-> 4-> 8-> 1-> 3-> 7 (37)
7-> 2-> 6-> 5-> 4-> 8-> 3-> 1-> 7 (37)
7-> 2-> 6-> 5-> 8-> 1-> 3-> 4-> 7 (36)
7-> 2-> 6-> 5-> 8-> 1-> 4-> 3-> 7 (42)
7-> 2-> 6-> 5-> 8-> 3-> 1-> 4-> 7 (37)
7-> 2-> 6-> 5-> 8-> 3-> 4-> 1-> 7 (37)
7-> 2-> 6-> 5-> 8-> 4-> 1-> 3-> 7 (40)
7-> 2-> 6-> 5-> 8-> 4-> 3-> 1-> 7 (40)
7-> 2-> 6-> 8-> 1-> 3-> 4-> 5-> 7 (37)
7-> 2-> 6-> 8-> 1-> 3-> 5-> 4-> 7 (32)
7-> 2-> 6-> 8-> 1-> 4-> 3-> 5-> 7 (38)
7-> 2-> 6-> 8-> 1-> 4-> 5-> 3-> 7 (38)
7-> 2-> 6-> 8-> 1-> 5-> 3-> 4-> 7 (28)
7-> 2-> 6-> 8-> 1-> 5-> 4-> 3-> 7 (33)
7-> 2-> 6-> 8-> 3-> 1-> 4-> 5-> 7 (38)
7-> 2-> 6-> 8-> 3-> 1-> 5-> 4-> 7 (28)
7-> 2-> 6-> 8-> 3-> 4-> 1-> 5-> 7 (34)
7-> 2-> 6-> 8-> 3-> 4-> 5-> 1-> 7 (34)
7-> 2-> 6-> 8-> 3-> 5-> 1-> 4-> 7 (29)
7-> 2-> 6-> 8-> 3-> 5-> 4-> 1-> 7 (29)
7-> 2-> 6-> 8-> 4-> 1-> 3-> 5-> 7 (36)
7-> 2-> 6-> 8-> 4-> 1-> 5-> 3-> 7 (32)
7-> 2-> 6-> 8-> 4-> 3-> 1-> 5-> 7 (31)
7-> 2-> 6-> 8-> 4-> 3-> 5-> 1-> 7 (31)
7-> 2-> 6-> 8-> 4-> 5-> 1-> 3-> 7 (31)
7-> 2-> 6-> 8-> 4-> 5-> 3-> 1-> 7 (31)
7-> 2-> 6-> 8-> 5-> 1-> 3-> 4-> 7 (30)
7-> 2-> 6-> 8-> 5-> 1-> 4-> 3-> 7 (36)
7-> 2-> 6-> 8-> 5-> 3-> 1-> 4-> 7 (35)
7-> 2-> 6-> 8-> 5-> 3-> 4-> 1-> 7 (35)
7-> 2-> 6-> 8-> 5-> 4-> 1-> 3-> 7 (40)
7-> 2-> 6-> 8-> 5-> 4-> 3-> 1-> 7 (40)
7-> 2-> 8-> 1-> 3-> 4-> 5-> 6-> 7 (35)
7-> 2-> 8-> 1-> 3-> 4-> 6-> 5-> 7 (39)
```

```
7-> 3-> 1-> 6-> 4-> 2-> 5-> 8-> 7 (35)
7-> 3-> 1-> 6-> 4-> 2-> 8-> 5-> 7 (43)
7-> 3-> 1-> 6-> 4-> 5-> 2-> 8-> 7 (35)
7-> 3-> 1-> 6-> 4-> 5-> 8-> 2-> 7 (39)
7-> 3-> 1-> 6-> 4-> 8-> 2-> 5-> 7 (35)
7-> 3-> 1-> 6-> 4-> 8-> 5-> 2-> 7 (31)
7-> 3-> 1-> 6-> 5-> 2-> 4-> 8-> 7 (32)
7-> 3-> 1-> 6-> 5-> 2-> 8-> 4-> 7 (31)
7-> 3-> 1-> 6-> 5-> 4-> 2-> 8-> 7 (40)
7-> 3-> 1-> 6-> 5-> 4-> 8-> 2-> 7 (36)
7-> 3-> 1-> 6-> 5-> 8-> 2-> 4-> 7 (39)
7-> 3-> 1-> 6-> 5-> 8-> 4-> 2-> 7 (36)
7-> 3-> 1-> 6-> 8-> 2-> 4-> 5-> 7 (40)
7-> 3-> 1-> 6-> 8-> 2-> 5-> 4-> 7 (31)
7-> 3-> 1-> 6-> 8-> 4-> 2-> 5-> 7 (32)
7-> 3-> 1-> 6-> 8-> 4-> 5-> 2-> 7 (28)
7-> 3-> 1-> 6-> 8-> 5-> 2-> 4-> 7 (31)
7-> 3-> 1-> 6-> 8-> 5-> 4-> 2-> 7 (36)
7-> 3-> 1-> 8-> 2-> 4-> 5-> 6-> 7 (39)
7-> 3-> 1-> 8-> 2-> 4-> 6-> 5-> 7 (43)
7-> 3-> 1-> 8-> 2-> 5-> 4-> 6-> 7 (34)
7-> 3-> 1-> 8-> 2-> 5-> 6-> 4-> 7 (34)
7-> 3-> 1-> 8-> 2-> 6-> 4-> 5-> 7 (44)
7-> 3-> 1-> 8-> 2-> 6-> 5-> 4-> 7 (40)
7-> 3-> 1-> 8-> 4-> 2-> 5-> 6-> 7 (31)
7-> 3-> 1-> 8-> 4-> 2-> 6-> 5-> 7 (41)
7-> 3-> 1-> 8-> 4-> 5-> 2-> 6-> 7 (32)
7-> 3-> 1-> 8-> 4-> 5-> 6-> 2-> 7 (37)
7-> 3-> 1-> 8-> 4-> 6-> 2-> 5-> 7 (36)
7-> 3-> 1-> 8-> 4-> 6-> 5-> 2-> 7 (31)
7-> 3-> 1-> 8-> 5-> 2-> 4-> 6-> 7 (34)
7-> 3-> 1-> 8-> 5-> 2-> 6-> 4-> 7 (35)
7-> 3-> 1-> 8-> 5-> 4-> 2-> 6-> 7 (40)
7-> 3-> 1-> 8-> 5-> 4-> 6-> 2-> 7 (40)
7-> 3-> 1-> 8-> 5-> 6-> 2-> 4-> 7 (40)
7-> 3-> 1-> 8-> 5-> 6-> 4-> 2-> 7 (39)
7-> 3-> 1-> 8-> 6-> 2-> 4-> 5-> 7 (41)
7-> 3-> 1-> 8-> 6-> 2-> 5-> 4-> 7 (32)
7-> 3-> 1-> 8-> 6-> 4-> 2-> 5-> 7 (35)
7-> 3-> 1-> 8-> 6-> 4-> 5-> 2-> 7 (31)
7-> 3-> 1-> 8-> 6-> 5-> 2-> 4-> 7 (31)
7-> 3-> 1-> 8-> 6-> 5-> 4-> 2-> 7 (36)
7-> 3-> 2-> 1-> 4-> 5-> 6-> 8-> 7 (40)
7-> 3-> 2-> 1-> 4-> 5-> 8-> 6-> 7 (39)
7-> 3-> 2-> 1-> 4-> 6-> 5-> 8-> 7 (43)
7-> 3-> 2-> 1-> 4-> 6-> 8-> 5-> 7 (43)
7-> 3-> 2-> 1-> 4-> 8-> 5-> 6-> 7 (39)
7-> 3-> 2-> 1-> 4-> 8-> 6-> 5-> 7 (40)
7-> 3-> 2-> 1-> 5-> 4-> 6-> 8-> 7 (34)
7-> 3-> 2-> 1-> 5-> 4-> 8-> 6-> 7 (30)
```

```
-> 3-> 4-> 2-> 8-> 5-> 6-> 1-> 7 (35)
-> 3-> 4-> 2-> 8-> 6-> 1-> 5-> 7 (36)
-> 3-> 4-> 2-> 8-> 6-> 5-> 1-> 7 (36)
-> 3-> 4-> 5-> 1-> 2-> 6-> 8-> 7 (37)
-> 3-> 4-> 5-> 1-> 2-> 8-> 6-> 7 (35)
-> 3-> 4-> 5-> 1-> 6-> 2-> 8-> 7 (37)
-> 3-> 4-> 5-> 1-> 6-> 8-> 2-> 7 (32)
-> 3-> 4-> 5-> 1-> 8-> 2-> 6-> 7 (36)
-> 3-> 4-> 5-> 1-> 8-> 6-> 2-> 7 (33)
-> 3-> 4-> 5-> 2-> 1-> 6-> 8-> 7 (34)
-> 3-> 4-> 5-> 2-> 1-> 8-> 6-> 7 (33)
-> 3-> 4-> 5-> 2-> 6-> 1-> 8-> 7 (35)
-> 3-> 4-> 5-> 2-> 6-> 8-> 1-> 7 (35)
-> 3-> 4-> 5-> 2-> 8-> 1-> 6-> 7 (33)
-> 3-> 4-> 5-> 2-> 8-> 6-> 1-> 7 (33)
-> 3-> 4-> 5-> 6-> 1-> 2-> 8-> 7 (42)
-> 3-> 4-> 5-> 6-> 1-> 8-> 2-> 7 (38)
-> 3-> 4-> 5-> 6-> 2-> 1-> 8-> 7 (43)
-> 3-> 4-> 5-> 6-> 2-> 8-> 1-> 7 (43)
-> 3-> 4-> 5-> 6-> 8-> 1-> 2-> 7 (38)
-> 3-> 4-> 5-> 6-> 8-> 2-> 1-> 7 (38)
-> 3-> 4-> 5-> 8-> 1-> 2-> 6-> 7 (42)
-> 3-> 4-> 5-> 8-> 1-> 6-> 2-> 7 (39)
-> 3-> 4-> 5-> 8-> 2-> 1-> 6-> 7 (41)
-> 3-> 4-> 5-> 8-> 2-> 6-> 1-> 7 (41)
-> 3-> 4-> 5-> 8-> 6-> 1-> 2-> 7 (38)
-> 3-> 4-> 5-> 8-> 6-> 2-> 1-> 7 (38)
-> 3-> 4-> 6-> 1-> 2-> 5-> 8-> 7 (37)
-> 3-> 4-> 6-> 1-> 2-> 8-> 5-> 7 (45)
-> 3-> 4-> 6-> 1-> 5-> 2-> 8-> 7 (31)
-> 3-> 4-> 6-> 1-> 5-> 8-> 2-> 7 (35)
-> 3-> 4-> 6-> 1-> 8-> 2-> 5-> 7 (37)
-> 3-> 4-> 6-> 1-> 8-> 5-> 2-> 7 (33)
-> 3-> 4-> 6-> 2-> 1-> 5-> 8-> 7 (40)
-> 3-> 4-> 6-> 2-> 1-> 8-> 5-> 7 (46)
-> 3-> 4-> 6-> 2-> 5-> 1-> 8-> 7 (32)
-> 3-> 4-> 6-> 2-> 5-> 8-> 1-> 7 (32)
-> 3-> 4-> 6-> 2-> 8-> 1-> 5-> 7 (40)
-> 3-> 4-> 6-> 2-> 8-> 5-> 1-> 7 (40)
-> 3-> 4-> 6-> 5-> 1-> 2-> 8-> 7 (39)
-> 3-> 4-> 6-> 5-> 1-> 8-> 2-> 7 (35)
-> 3-> 4-> 6-> 5-> 2-> 1-> 8-> 7 (37)
-> 3-> 4-> 6-> 5-> 2-> 8-> 1-> 7 (37)
-> 3-> 4-> 6-> 5-> 8-> 1-> 2-> 7 (41)
-> 3-> 4-> 6-> 5-> 8-> 2-> 1-> 7 (41)
-> 3-> 4-> 6-> 8-> 1-> 2-> 5-> 7 (37)
-> 3-> 4-> 6-> 8-> 1-> 5-> 2-> 7 (27)
-> 3-> 4-> 6-> 8-> 2-> 1-> 5-> 7 (39)
-> 3-> 4-> 6-> 8-> 2-> 5-> 1-> 7 (39)
-> 3-> 4-> 6-> 8-> 5-> 1-> 2-> 7 (35)
```

```
7-> 5-> 3-> 1-> 4-> 2-> 8-> 6-> 7 (38)
7-> 5-> 3-> 1-> 4-> 6-> 2-> 8-> 7 (43)
7-> 5-> 3-> 1-> 4-> 6-> 8-> 2-> 7 (38)
7-> 5-> 3-> 1-> 4-> 8-> 2-> 6-> 7 (39)
7-> 5-> 3-> 1-> 4-> 8-> 6-> 2-> 7 (36)
7-> 5-> 3-> 1-> 6-> 2-> 4-> 8-> 7 (37)
7-> 5-> 3-> 1-> 6-> 2-> 8-> 4-> 7 (36)
7-> 5-> 3-> 1-> 6-> 4-> 2-> 8-> 7 (39)
7-> 5-> 3-> 1-> 6-> 4-> 8-> 2-> 7 (35)
7-> 5-> 3-> 1-> 6-> 8-> 2-> 4-> 7 (35)
7-> 5-> 3-> 1-> 6-> 8-> 4-> 2-> 7 (32)
7-> 5-> 3-> 1-> 8-> 2-> 4-> 6-> 7 (38)
7-> 5-> 3-> 1-> 8-> 2-> 6-> 4-> 7 (39)
7-> 5-> 3-> 1-> 8-> 4-> 2-> 6-> 7 (36)
7-> 5-> 3-> 1-> 8-> 4-> 6-> 2-> 7 (36)
7-> 5-> 3-> 1-> 8-> 6-> 2-> 4-> 7 (36)
7-> 5-> 3-> 1-> 8-> 6-> 4-> 2-> 7 (35)
7-> 5-> 3-> 2-> 1-> 4-> 6-> 8-> 7 (39)
7-> 5-> 3-> 2-> 1-> 4-> 8-> 6-> 7 (35)
7-> 5-> 3-> 2-> 1-> 6-> 4-> 8-> 7 (36)
7-> 5-> 3-> 2-> 1-> 6-> 8-> 4-> 7 (32)
7-> 5-> 3-> 2-> 1-> 8-> 4-> 6-> 7 (35)
7-> 5-> 3-> 2-> 1-> 8-> 6-> 4-> 7 (35)
7-> 5-> 3-> 2-> 4-> 1-> 6-> 8-> 7 (36)
7-> 5-> 3-> 2-> 4-> 1-> 8-> 6-> 7 (35)
7-> 5-> 3-> 2-> 4-> 6-> 1-> 8-> 7 (36)
7-> 5-> 3-> 2-> 4-> 6-> 8-> 1-> 7 (36)
7-> 5-> 3-> 2-> 4-> 8-> 1-> 6-> 7 (32)
7-> 5-> 3-> 2-> 4-> 8-> 6-> 1-> 7 (32)
7-> 5-> 3-> 2-> 6-> 1-> 4-> 8-> 7 (37)
7-> 5-> 3-> 2-> 6-> 1-> 8-> 4-> 7 (33)
7-> 5-> 3-> 2-> 6-> 4-> 1-> 8-> 7 (40)
7-> 5-> 3-> 2-> 6-> 4-> 8-> 1-> 7 (40)
7-> 5-> 3-> 2-> 6-> 8-> 1-> 4-> 7 (36)
7-> 5-> 3-> 2-> 6-> 8-> 4-> 1-> 7 (36)
7-> 5-> 3-> 2-> 8-> 1-> 4-> 6-> 7 (38)
7-> 5-> 3-> 2-> 8-> 1-> 6-> 4-> 7 (35)
7-> 5-> 3-> 2-> 8-> 4-> 1-> 6-> 7 (35)
7-> 5-> 3-> 2-> 8-> 4-> 6-> 1-> 7 (35)
7-> 5-> 3-> 2-> 8-> 6-> 1-> 4-> 7 (35)
7-> 5-> 3-> 2-> 8-> 6-> 4-> 1-> 7 (35)
7-> 5-> 3-> 4-> 1-> 2-> 6-> 8-> 7 (42)
7-> 5-> 3-> 4-> 1-> 2-> 8-> 6-> 7 (40)
7-> 5-> 3-> 4-> 1-> 6-> 2-> 8-> 7 (42)
7-> 5-> 3-> 4-> 1-> 6-> 8-> 2-> 7 (37)
7-> 5-> 3-> 4-> 1-> 8-> 2-> 6-> 7 (41)
7-> 5-> 3-> 4-> 1-> 8-> 6-> 2-> 7 (38)
7-> 5-> 3-> 4-> 2-> 1-> 6-> 8-> 7 (38)
7-> 5-> 3-> 4-> 2-> 1-> 8-> 6-> 7 (37)
7-> 5-> 3-> 4-> 2-> 6-> 1-> 8-> 7 (39)
```

Мінімальні шляхи з кожної вершини:




```cpp
#include <iostream>
#include <stdio.h>
#include <string>
#include <fstream>
using namespace std;
ifstream fin;
string path = "MyFile1.txt";

struct mass{
    int mas[9];
};

int** input() {
    int coun = 8;
    string str;
    str = "";

    fin.open(path);
    int** arr;
    arr = new int* [coun];
    for (int i = 0; i < coun; i++)
        arr[i] = new int[coun];
    for (int i = 0; i < coun; i++){
        for (int j = 0; j < coun; j++)
            arr[i][j] = 0;
    }
    for (int i = 0; i < coun; i++){
        for (int j = i + 1; j < coun; j++){
            getline(fin, str);
            arr[i][j] = atoi(str.c_str());
            arr[j][i] = atoi(str.c_str());
        }
    }
    fin.close();
    return arr;
}
bool compon(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++){
```

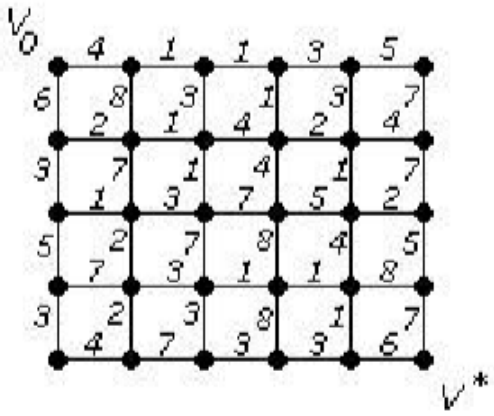
```
        if (mas[i] != arr[i]){
            return true;
        }
        else{
            continue;
        }
    }
    return false;
}
bool povt(int* mas, int size){
    bool k = true;
    for (int i = 0; i < size; i++){
        for (int j = 0; j < size; j++){
            if (mas[i] == mas[j] && i != j){
                return false;
            }
        }
    }
    return true;
}
int way(int** mat, int* arr){
    int count = 0;
    for (int i = 0; i < 7; i++){
        count += mat[arr[i] - 1][arr[i + 1] - 1];
    }
    count += mat[arr[7] - 1][arr[0] - 1];
    return count;
}
int main() {
    int const count = 8;
    int** arr;
    arr = input();
    int var = count - 1;
    bool k = true;
    int* mas = new int[count];
    int* minmas = new int[9];
    int min = 1000;
    int leng = 0;
    int m = 0;
    for (int i = 0; i < count; i++){
        mas[i] = 1;
        minmas[i] = 1;
    }
    while (compon(mas, count)){
        while (mas[var] != count){
            mas[var]++;
            if (povt(mas, count)){
                leng = way(arr, mas);
                for (int i = 0; i < count; i++){
                    cout << mas[i] << "-> ";
                }
                cout << mas[0] << " (" << leng << ") ";
                cout << endl;
                if (leng < min){
                    min = leng;
                    m = 1;
                }
                if (leng == min){
                    m++;
                }
            }
        }
        while (mas[var] == count){
            mas[var] = 1;
            var--;
        }
        mas[var]++;
        if (povt(mas, count)){
```

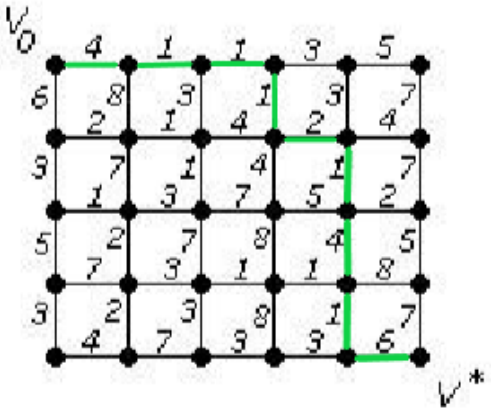
```
            for (int i = 0; i < count; i++){
                cout << mas[i] << "-> ";
            }
            cout << mas[0] << " (" << leng << ") ";
            cout << endl;
            leng = way(arr, mas);
            if (leng < min){
                min = leng;
                m = 1;
            }
            if (leng == min){
                m++;
            }
        }
        var = count - 1;
    }
    for (int i = 0; i < count; i++){
        mas[i] = 1;
        minmas[i] = 1;
    }
    mass* rez = new mass[m];
    int iter = 0;
    while (compon(mas, count)){
        while (mas[var] != count){
            mas[var]++;
            if (povt(mas, count)){
                leng = way(arr, mas);
                if (leng == min){
                    for (int i = 0; i < count; i++){
                        rez[iter].mas[i] = mas[i];
                    }
                    rez[iter].mas[count] = mas[0];
                    iter++;
                }
            }
        }
        while (mas[var] == count){
            mas[var] = 1;
            var--;
        }
        mas[var]++;
        if (povt(mas, count)){
            leng = way(arr, mas);
            if (leng == min){
                for (int i = 0; i < count; i++){
                    rez[iter].mas[i] = mas[i];
                }
                rez[iter].mas[count] = mas[0];
                iter++;
            }
        }
        var = count - 1;
    }
    cout << "Ways: " << 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 << "Minimal leng = " << min;
    return 0;
}
```

Завдання №7.

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

17)






```
#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, rebro;
	fin >> versh >> rebro;
	const int SIZE = 30;
	int matr[SIZE][SIZE]; // матриця зв'язків
	int dist[SIZE]; // мінімальна відстань
	int visit[SIZE];  // чи відвідані вершни
	int dis, top, min;
	int start_index = 0;


	for (int i = 0; i < versh; i++){     // Ініціалізація матриці зв'язків
		dist[i] = 99999;
		visit[i] = 0;
		for (int j = 0; j < versh; j++){
			matr[i][j] = 0;
			matr[j][i] = 0;
		}
	}
	for (int i = 0; i < rebro; i++) {
		int v1, v2, dis;
		fin >> v1 >> v2 >> dis;
		matr[v1 - 1][v2 - 1] = dis;
		matr[v2 - 1][v1 - 1] = dis;
	}
	cout << "Алгоритм Дейкстра\n";

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

```
                }
        }
        if (top != 99999){
                for (int i = 0; i < versh; i++){
                        if (matr[top][i] > 0){
                                dis = min + matr[top][i];
                                if (dis < dist[i]){
                                        dist[i] = dis;
                                }
                        }
                }
                visit[top] = 1;
        }
    } while (top < 99999);

    int end = versh - 1;
    int waga = dist[end];
    int way[30];
    way[0] = versh - 1;
    int k = 1;
    while (end != 0){
        for (int i = 0; i < versh; i++){
                if (matr[end][i] > 0){
                        if (dist[i] == waga - matr[end][i]){
                                waga = dist[i];
                                way[k] = i;
                                end = i;
                                k++;
                        }
                }
        }
    }
    cout << "\nНайменший шлях з V0 в V29: ";
    for (int i = k; i > 0; i--){
        if (i - 1 > 0)
                cout << way[i - 1] << " -> ";
        else
                cout << way[i - 1];
    }
    cout << "\n\nДовжина відстані: " << dist[versh - 1] << endl;
}
```

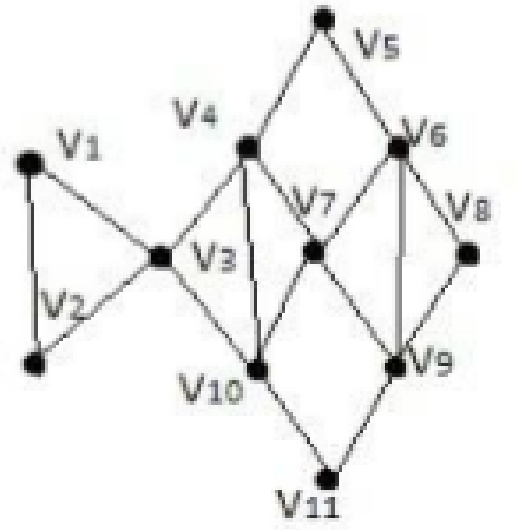
```
Алгоритм Дейкстра

Найменший шлях з V0 в V29: 0 -> 1 -> 2 -> 3 -> 9 -> 10 -> 16 -> 22 -> 28 -> 29

Довжина відстані: 21
```

Завдання № 8

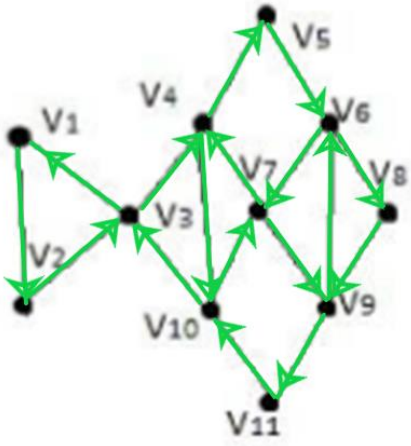
Знайти ейлеровий цикл в ейлеровому графі двома методами:

а) Флері;

б) елементарних циклів.



А) 1→2→3→4→10→7→4→5→6→7→9→6→8→9→11→10→3→1




```
#include <iostream>
#include <string.h>
#include <algorithm>
#include <list>
using namespace std;

class Graph{
	int V;
	list<int>* adj;
public:
	Graph(int V) { this->V = V; adj = new list<int>[V + 1]; }
	~Graph() { delete[] adj; }
	void addEdge(int u, int v) { adj[u].push_back(v);
adj[v].push_back(u); }
	void rmvEdge(int u, int v);
	void printEulerTour();
	void printEulerUtil(int s);
	int DFSCount(int v, bool visited[]);
	bool isValidNextEdge(int u, int v);
};

void Graph::printEulerTour(){
	int u = 1;
	for (int i = 1; i <= V; i++)

		if (adj[i].size() & 1){
			u = i; break;
		}
	printEulerUtil(u);
	cout << endl;
}

void Graph::printEulerUtil(int u){
	list<int>::iterator i;
	for (i = adj[u].begin(); i != adj[u].end(); ++i){
		int v = *i;
```

```
            if (v != -1 && isValidNextEdge(u, v)){
                cout << u << "-" << v << " ";
                rmvEdge(u, v);
                printEulerUtil(v);
            }
        }
}

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

void Graph::rmvEdge(int u, int v){
        list<int>::iterator iv = find(adj[u].begin(), adj[u].end(), v);
        *iv = -1;
        list<int>::iterator iu = find(adj[v].begin(), adj[v].end(), u);
        *iu = -1;
}

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

int main(){
        Graph g1(17);
        g1.addEdge(1, 2);
        g1.addEdge(1, 3);
        g1.addEdge(2, 3);
        g1.addEdge(3, 4);
        g1.addEdge(3, 10);
        g1.addEdge(4, 10);
        g1.addEdge(4, 7);
        g1.addEdge(4, 5);
        g1.addEdge(5, 6);
        g1.addEdge(6, 7);
        g1.addEdge(6, 8);
        g1.addEdge(6, 9);
        g1.addEdge(7, 9);
        g1.addEdge(7, 10);
        g1.addEdge(8, 9);
        g1.addEdge(9, 11);
        g1.addEdge(10, 11);
        g1.printEulerTour();
        return 0;
    }
```

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

Б) 1→3→10→11→9→8→6→9→7→6→5→4→7→10→4→3→2→1

```
#include <iostream>
#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];
    }
    if (any_of(deg.begin() + 1, deg.end(), [](int i) {return i & 1; }))
        cout << "No euler cycle exists\n";
    else {
        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 17
1 2
1 3
2 3
3 4
3 10
4 10
4 5
4 7
10 7
10 11
5 6
7 6
7 9
11 9
6 9
6 8
9 8
1 3 10 11 9 8 6 9 7 6 5 4 7 10 4 3 2 1
```

Завдання №9

Спростити формули (привести їх до скороченої ДНФ).

17. $x\bar{y} \vee \bar{x}\bar{z} \vee yz$

$= (x \wedge \neg y) \vee (\neg x \wedge \neg z) \vee (y \wedge z) =$

$= (x \wedge \neg y) \vee ((\neg x \wedge \neg z) \vee y) \wedge ((\neg x \wedge \neg z) \vee z) =$

$= (x \wedge \neg y) \vee ((\neg x \vee y) \wedge (\neg z \wedge y)) \wedge ((\neg x \vee z) \wedge (\neg z \wedge z)) =$

$= (x \wedge \neg y) \vee ((\neg x \vee y) \wedge (\neg z \wedge y)) \wedge ((\neg x \vee z)) =$

$= (x \wedge \neg y) \vee \neg x \vee y =$

$= ((x \wedge \neg x) \wedge (\neg y \vee \neg x)) =$

$= \neg y \vee \neg x \vee y =$

$= 1$