

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТУ “ЛЬВІВСЬКА
ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Розрахункова робота

з дисципліни

«Дискретна математика»

Виконала:

студентка групи КН-112

Тимчишин Марта

Викладач:

Мельникова Н.І

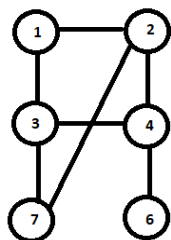
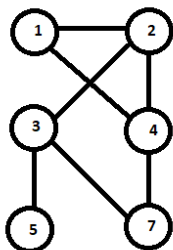
Львів-2019 р.

Варіант 14

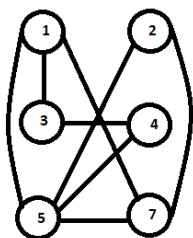
Завдання 1

Розв'язати на графах наступні задачі:

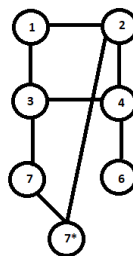
- 1) знайти доповнення до першого графу
- 2) об'єднання графів
- 3) кільцеву суму $G1$ та $G2$
- 4) розщепити вершину у другому графі
- 5) виділити підграф A , що складається з 3-х вершин в $G1$ і знайти стягнення A в $G1$ ($G1 \setminus A$)
- 6) добуток графів



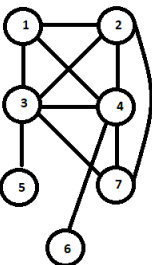
1.



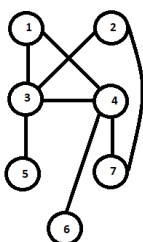
4.



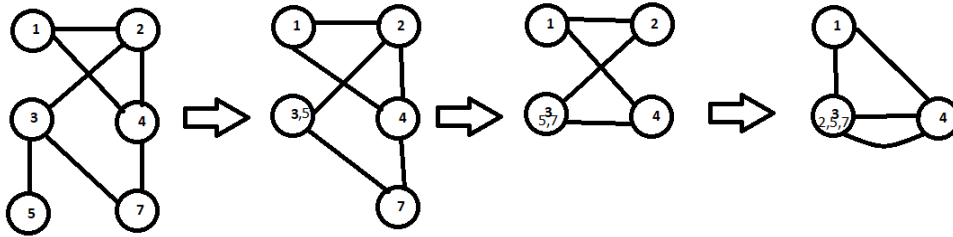
2.



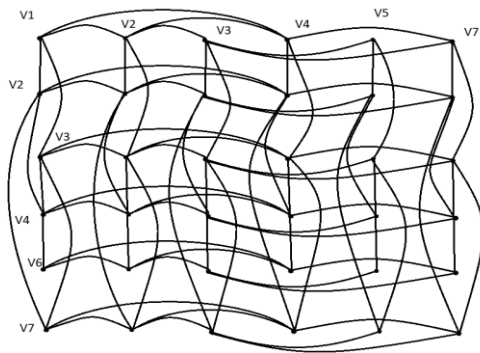
3.



5.

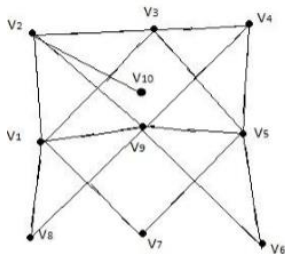


6.



Завдання 2

Скласти таблицю суміжності для графа



	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
V1	0	1	1	0	0	0	1	1	1	0
V2	1	0	1	0	0	0	0	0	1	1
V3	1	1	0	1	1	0	0	0	0	0
V4	0	0	1	0	1	0	0	0	1	0
V5	0	0	1	1	0	1	1	0	1	0
V6	0	0	0	0	1	0	0	0	1	0
V7	1	0	0	0	1	0	0	0	0	0
V8	1	0	0	0	0	0	0	0	1	0
V9	1	1	0	1	1	1	0	1	0	0
V10	0	1	0	0	0	0	0	0	0	0

Завдання 3

Для графа з другого завдання знайти діаметр

Розв'язок :

Діаметр = 3

Завдання 4

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

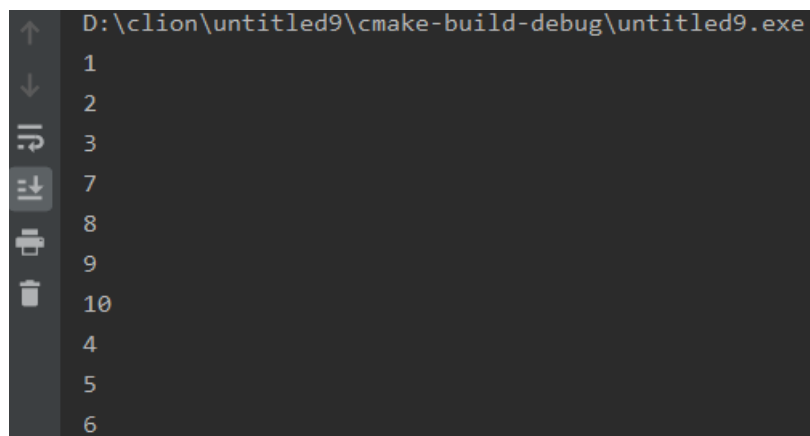
Вершина	Номер	Черга
V1	1	V1
V2	2	V1V2
V3	3	V1V2V3
V7	4	V1V2V3V7
V8	5	V1V2V3V7V8
V9	6	V1V2V3V7V8V9
-	-	V2V3V7V8V9
V10	7	V2V3V7V8V9V10
-	-	V3V7V8V9V10
V4	8	V3V7V8V9V10V4
V5	9	V3V7V8V9V10V4V5
-	-	V7V8V9V10V4V5
-	-	V8V9V10V4V5
-	-	V9V10V4V5
V6	10	V9V10V4V5V6
-	-	V10V4V5V6
-	-	V4V5V6
-	-	V5V6
-	-	V6
-	-	Пуста множина

```

#include <iostream>
#include <queue> // черга
using namespace std;
int main()
{
    queue<int> Queue;
    int mas[10][10] = {{0,1,1,0,0,0,1,1,1,0},
                       {1,0,1,0,0,0,0,0,1,1},
                       {1,1,0,1,1,0,0,0,0,0},
                       {0,0,1,0,1,0,0,0,1,0},
                       {0,0,1,1,0,1,1,0,1,0},
                       {0,0,0,0,1,0,0,0,1,0},
                       {1,0,0,0,1,0,0,0,0,0},
                       {1,0,0,0,0,0,0,0,1,0},
                       {1,1,0,1,1,1,0,1,0,0},
                       {0,1,0,0,0,0,0,0,0,0}};

    int nodes[10]; // вершини графа
    for (int i = 0; i < 10; i++)
        nodes[i] = 0; // іпочатково всі вершини = 0
    Queue.push(0); // поміщаємо в чергу першу вершину
    while (!Queue.empty())
    { // поки очерга не пуста
        int node = Queue.front(); // витягаємо вершину
        Queue.pop();
        nodes[node] = 2; // помічаємо її як відвідану
        for (int j = 0; j < 10; j++)
        { // перевіряємо всі суміжні для неї вершини
            if (mas[node][j] == 1 && nodes[j] == 0)
            { // якщо вершина суміжна і не виявлена ще
                Queue.push(j); // додаємо її в чергу
                nodes[j] = 1; // помічаємо як відмічену
            }
        }
        cout << node + 1 << endl; // виводимо номер вершини
    }
    cin.get();
    return 0;
}

```

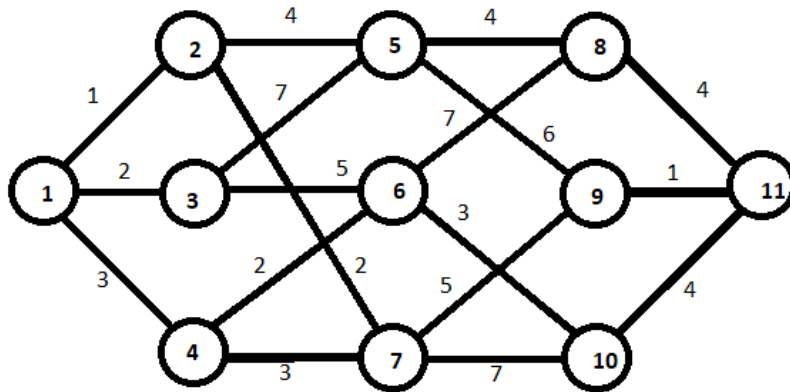


D:\clion\untitled9\cmake-build-debug\untitled9.exe

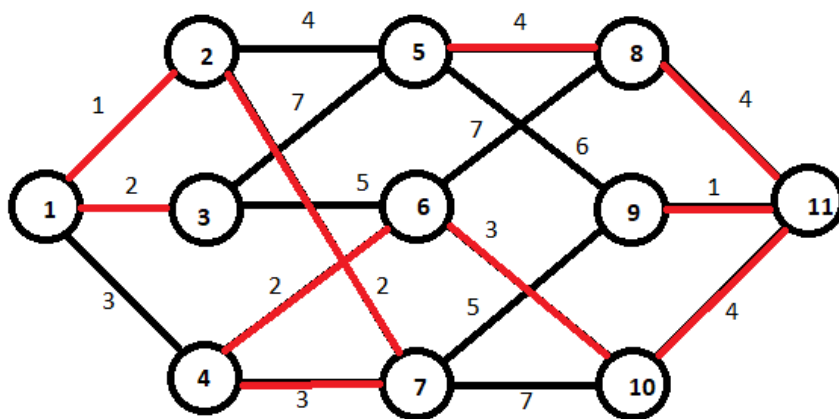
1
2
3
7
8
9
10
4
5
6

Завдання 5

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

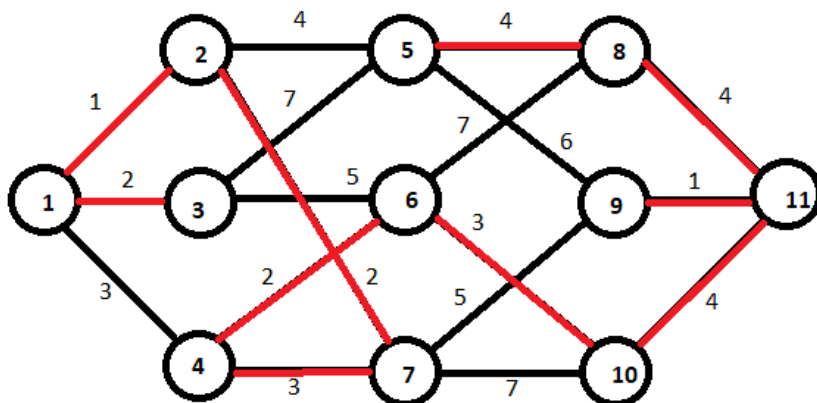


Краскала



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

Прима



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

Краскала

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

int create(int n, int A[11][11]) {
    for (int i = 0; i < 11; i++) {
        for (int j = 0; j < 11; j++) {
            A[i][j] = 0;
        }
    }
    for (int i = 0; i < 11; i++) {
        A[i][i] = i+1;
    }
    return A[11][11];
}

void dubl(int n, int A[11][11]) {
    for (int i = 0; i < 11; i++) {
        for (int j = 0; j < 11; j++) {
            if (j < i) {
                A[i][j] = 0;
            }
        }
    }
}

int no(int n, int A[11][11], int t, int r) {
    int tm, tm2;
    for (int i = 0; i < 11; i++) {
        tm = tm2 = 0;
        for (int j = 0; j < 11; j++) {
            if (A[i][j] == t) {
                tm = 1;
            }
        }
        for (int f = 0; f < 11; f++) {
            if (A[i][f] == r) {
                tm2 = 1;
            }
        }
        if (tm && tm2) {
            return 0;
        }
    }
    return 1;
}

void add(int n, int A[11][11], int t, int r) {
    int scn;
    for (int i = 0; i < 11; i++) {
        for (int j = 0; j < 11; j++) {
            if (A[i][j] == r) {
                scn = i;
            }
        }
    }
    for (int i = 0; i < 11; i++) {
        for (int j = 0; j < 11; j++) {
            if (A[i][j] == t) {
```

```

        for (int k = 0; k < 11; k++) {
            A[i][k] = A[scn][k];
            A[scn][k] = 0;
        }
    }
}

int main() {
    int MS[11][11]{
        {0,1,2,3,0,0,0,0,0,0,0},
        {1,0,0,0,4,0,2,0,0,0,0},
        {2,0,0,0,7,5,0,0,0,0,0},
        {3,0,0,0,0,2,3,0,0,0,0},
        {0,4,7,0,0,0,0,4,6,0,0},
        {0,0,5,2,0,0,0,7,0,3,0},
        {0,2,0,3,0,0,0,0,5,7,0},
        {0,0,0,0,4,7,0,0,0,0,4},
        {0,0,0,0,6,0,5,0,0,0,1},
        {0,0,0,0,0,3,7,0,0,0,4},
        {0,0,0,0,0,0,0,4,1,4,0}
    };
    dubl(11, MS);
    for (int i = 1; i <= 7; i++) {
        cout << endl<<"Edges with weight: " << i << " ";
        for (int j = 1; j <= 11; j++) {
            for (int k = 1; k <= 11; k++) {
                if (MS[j - 1][k - 1] == i) {
                    cout << " " << j << "," << k << " ";
                }
            }
        }
    }

    int B[11][11];
    create(11, B);
    cout << endl << "New Tree: ";
    for (int i = 1; i <= 7; i++) {
        for (int j = 1; j <= 11; j++) {
            for (int k = 1; k <= 11; k++) {
                if (MS[j - 1][k - 1] == i && no(11, B, j, k)) {
                    add(11, B, j, k);
                    cout << " " << j << "," << k << " ";
                }
            }
        }
    }
    cout << endl;
    return 0;
}

```

```

Edges with weight: 1      1,2  9,11
Edges with weight: 2      1,3  2,7  4,6
Edges with weight: 3      1,4  4,7  6,10
Edges with weight: 4      2,5  5,8  8,11  10,11
Edges with weight: 5      3,6  7,9
Edges with weight: 6      5,9
Edges with weight: 7      3,5  6,8  7,10
New Tree:  1,2  9,11  1,3  2,7  4,6  1,4  4,7  6,10  2,5  5,8  8,11  10,11  3,6  7,9  5,9  3,5  6,8  7,10

```


Прима

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

int main()
{
    int v, count = 0, min = 0, k, t;
    bool check = false;
    cout << "The number of vertices of the graph : ";
    cin >> v;
    int* tops = new int[v];
    //для матриці ваг
    int** matrix = new int* [v];
    for (int i = 0; i < v; i++) {
        matrix[i] = new int[v];
    }
    //для ребер
    int** rebra = new int* [v - 1];

    for (int i = 0; i < v - 1; i++) {
        rebra[i] = new int[2];
    }
    //ввід матриці ваг
    for (int i = 0; i < v; i++) {
        for (int j = 0; j < v; j++) {
            cin >> matrix[i][j];
        }
    }
    //беремо вершину один як початкову

    tops[count] = 1;
    count++;

    for (int i = 0; count < v; i++) {
        for (int j = 0; j < count; j++) {
            for (int a = 0; a < v; a++) {
                for (int m = 0; m < count; m++) {
                    //перевірка чи вершина вже була пройде, якщо так, то переходимо до
наступної ітерації
                    if (tops[m] == a + 1) {
                        check = true;
                    }
                }

                if (check) { check = false; continue; }
                //шукаємо мінімальний елемент в рядку, спочатку береться просто перший
ненульовий елемент, а потім вже шукаємо відносно нього, це в наступному ifі
                if (min == 0 && matrix[tops[j] - 1][a] > 0) {
                    min = matrix[tops[j] - 1][a];
                    k = rebra[count - 1][0] = tops[j]; t = rebra[count - 1][1] = a + 1;
                    continue;
                }

                if (matrix[tops[j] - 1][a] > 0 && matrix[tops[j] - 1][a] < min) {
                    min = matrix[tops[j] - 1][a];
                    //записуємо ребро
                    k = rebra[count - 1][0] = tops[j]; t = rebra[count - 1][1] = a + 1;
                }
            }
        }
    }
}
```

```

    }
    //Обнулюємо ребро, бо вже не можемо по ньому проходити і маємо опускати в
    подальшому пошуку
    matrix[k - 1][t - 1] = 0; matrix[t - 1][k - 1] = 0;
    //Додаємо знайдену вершину в масив вершин
    tops[count] = t;
    count++;
    min = 0;
}
//output
//Виводимо наш масив вершин, який вийшов, які послідовно додавались
cout << "V: { ";

for (int j = 0; j < v; j++) {
    cout << tops[j] << ", ";
}

cout << "}";
//і виводимо ребра
cout << endl << "E:{ ";

for (int j = 0; j < v - 1; j++) {
    cout << "(" << rebra[j][0] << ", " << rebra[j][1] << "), ";
}
cout << "}";
return 0;
}

```

```

D:\clion\untitled9\cmake-build-debug\untitled9.exe
The number of vertices of the graph :11
0 1 2 3 0 0 0 0 0 0 0
1 0 0 0 4 0 2 0 0 0 0
2 0 0 0 7 5 0 0 0 0 0
3 0 0 0 0 2 3 0 0 0 0
0 4 7 0 0 0 0 4 6 0 0
0 0 5 2 0 0 0 7 0 3 0
0 2 0 3 0 0 0 0 5 7 0
0 0 0 0 4 7 0 0 0 0 4
0 0 0 0 6 0 5 0 0 0 1
0 0 0 0 0 3 7 0 0 0 4
0 0 0 0 0 0 0 4 1 4 0
V: { 1, 2, 3, 7, 4, 6, 10, 5, 11, 9, 8, }
E:{ (1,2),(1,3),(2,7),(1,4),(4,6),(6,10),(2,5),(10,11),(11,9),(5,8),}
Process finished with exit code 0

```

Завдання 6

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

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

1)

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

Починаємо шлях з вершини 1. Закреслюємо перший рядок та у першому стовпці шукаємо вершину, до якої шлях є мінімальний. Таких декілька, тому довільним чином обираємо вершину 3.

2)

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

Тепер закреслюємо третій рядок і шукаємо знову вершину, до якої шлях є мінімальний, вже у третьому стовпці. Це вершина 7. Обводимо елемент 7 3.

3)

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

Закреслюємо сьомий рядок і шукаємо вершину до якої шлях мінімальний у сьомому стовпці. Це вершина 8. Обводимо елемент 8 7.

4)

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

Закреслюємо восьмий рядок і шукаємо вершину, до якої шлях мінімальний, у восьмому стовпці. Це вершина 4. Обводимо елемент 4 8.

5)

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

Закреслюємо четвертий рядок і шукаємо вершину, до якої шлях мінімальний, у четвертому стовпці. Це вершина 2. Обводимо елемент 2 4.

6)

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

Закреслюємо другий рядок і шукаємо вершину, до якої шлях мінімальний, у другому стовпці. Це вершина 6. Обводимо елемент 6 2.

7)

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

Закреслюємо шостий рядок і шукаємо вершину, до якої шлях мінімальний, у шостому стовпці. Це вершина 5. Обводимо елемент 5 6.

8)

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

Ми пройшлися по всіх вершинах, але нам ще потрібно повернутись у початкову. Дивимось яка відстань від 5 до 1, та обводимо елемент 1 5.

Шлях : 1-3-7-8-4-2-6-5-1

Програма :

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

int counter =0, Miminal_way=9999;

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

int minimum(vector<int>* V, int** arr, int n,int i) {
    int min = 999;
    for (int j = 0; j < n; j++) {
        if (arr[i][j] < min && arr[i][j] != 0 && check((*V), j))min = arr[i][j];
    }
    return min;
}

void vhlub(vector<int>* V, int** arr, int n, int pos, vector<int> * Vcon) {
    int min;
    for (int i = pos, k = 0; k < 1; i++, k++) {
        min = minimum(V, arr, n, i);
        for (int j = 0; j < n; j++)
            if (arr[i][j] == min && check((*V), j)) {
                (*V).push_back(j);
                vhlub(V, arr, n, j,Vcon);
            }
        //вид результату
        if (V->size() == n) {
            (*V).push_back((*V)[0]);
            counter = 0;
            for (int l = 1; l <= n; l++)
                counter += arr[(*V)[l - 1]][(*V)[l]];
            for (auto it = (*V).begin(); it != (*V).end(); it++)
                cout << *it + 1 << " ";
            cout <<" ("<<counter<<")"<< endl;
            if (Miminal_way == counter) {
                for (int op = 0; op <= n; op++)
                    (*Vcon).push_back((*V)[op]);
                (*Vcon).push_back(counter);
            }
            else if (Miminal_way > counter) {
                (*Vcon).clear();
                for (int op = 0; op <= n; op++)
                    (*Vcon).push_back((*V)[op]);
                (*Vcon).push_back(counter);
                Miminal_way = counter;
            }
            V->pop_back();
        }
    }
    V->pop_back();
}
```

```

}

int main()
{
    int n;
    cout << "Enter number of nodes: ";
    cin >> n;
    int** arr = new int* [n];
    for (int i = 0; i < n; i++)
        arr[i] = new int[n];

    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            cin >> arr[i][j];
    vector<int> V;
    vector<int> Vcon;

    cout << endl;
    for (int i = 0; i < n; i++) {
        V.clear();
        V.push_back(i);
        vhlub(&V, arr, n, i, &Vcon);
    }

    cout << endl << endl << endl << "rez" << endl << endl;
    for (int i = 1; i <= Vcon.size(); i++) {

        if (i != 0 && i%(n+2) == 0 )
            cout << " (" << Vcon[i-1] << ")" << endl;
        else
            cout << Vcon[i-1] + 1 << " ";
    }
}

```

```

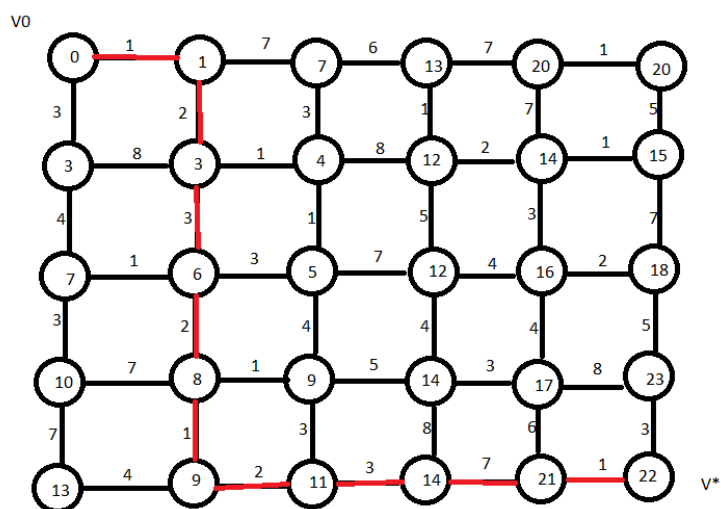
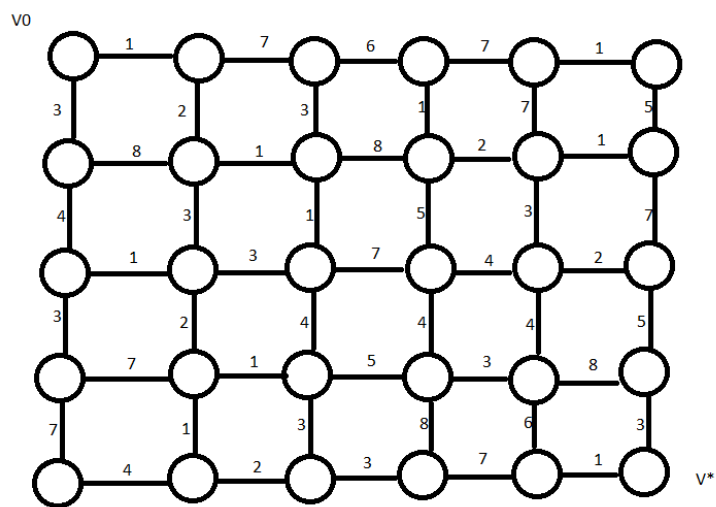
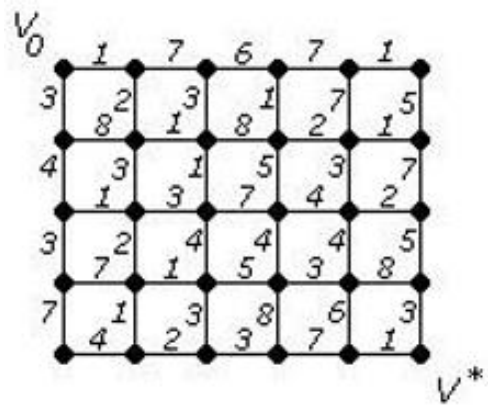
0 1 1 1 1 1 3 1
1 0 5 1 2 1 3 3
1 5 0 2 5 4 1 5
1 1 2 0 5 5 6 1
1 2 5 5 0 1 5 1
1 1 4 5 1 0 5 6
3 3 1 6 5 5 0 1
1 3 5 1 1 6 1 0

1 2 4 8 5 6 3 7 1 (13)
1 2 4 8 7 3 6 5 1 (11)
1 2 6 5 8 4 3 7 1 (11)
1 2 6 5 8 7 3 4 1 (9)
1 3 7 8 4 2 6 5 1 (8)

```

Завдання 7

За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі між парою вершин V_0 і V^* .



Дейкстри :

```
#define _CRT_SECURE_NO_WARNINGS
#include<iostream>
using namespace std;
```



```

int main()
{
    int maximalNumber = 10000;
    int vertex ;//к-сть вершин
    cout << "Amount of vertexes: ";
    cin >> vertex;
    int** a = new int* [vertex];
    int* d = new int[vertex];
    int* visited = new int[vertex];
    int temp, minindex, min;
    for (int i = 0; i < vertex; i++)
    {
        a[i] = new int[vertex];
    }
    for (int i = 0; i < vertex; i++)
    {
        for (int j = 0; j < vertex; j++) {
            a[i][j] = 0;
        }
    }
    a[0][1] = 1;
    a[0][6] = 3;
    a[1][0] = 1;
    a[1][2] = 7;
    a[1][7] = 2;
    a[2][1] = 7;
    a[2][8] = 3;
    a[2][3] = 6;
    a[3][2] = 6;
    a[3][4] = 7;
    a[3][9] = 1;
    a[4][3] = 7;
    a[4][5] = 1;
    a[4][10] = 7;
    a[5][4] = 1;
    a[5][11] = 5;
    a[6][0] = 3;
    a[6][7] = 8;
    a[6][12] = 4;
    a[7][1] = 2;
    a[7][6] = 8;
    a[7][8] = 1;
    a[7][13] = 3;
    a[8][7] = 1;
    a[8][2] = 3;
    a[8][14] = 1;
    a[8][9] = 8;
    a[9][8] = 8;
    a[9][3] = 1;
    a[9][15] = 5;
    a[9][10] = 2;
    a[10][9] = 2;
    a[10][4] = 7;
    a[10][16] = 3;
    a[10][11] = 1;
    a[11][10] = 1;
    a[11][5] = 5;
    a[11][17] = 7;
    a[12][6] = 4;
    a[12][13] = 1;
    a[12][18] = 3;
    a[13][12] = 1;

```

```
a[13][7] = 3;
a[13][19] = 2;
a[13][14] = 3;
a[14][13] = 3;
a[14][8] = 1;
a[14][20] = 4;
a[14][15] = 7;
a[15][14] = 7;
a[15][9] = 5;
a[15][21] = 4;
a[15][16] = 4;
a[16][15] = 4;
a[16][10] = 3;
a[16][22] = 4;
a[16][17] = 2;
a[17][16] = 2;
a[17][11] = 7;
a[17][23] = 5;
a[18][12] = 3;
a[18][19] = 7;
a[18][24] = 7;
a[19][18] = 7;
a[19][13] = 2;
a[19][25] = 1;
a[19][20] = 1;
a[20][19] = 1;
a[20][14] = 4;
a[20][26] = 3;
a[20][21] = 5;
a[21][20] = 5;
a[21][15] = 4;
a[21][27] = 8;
a[21][22] = 3;
a[22][21] = 3;
a[22][16] = 4;
a[22][28] = 6;
a[22][23] = 8;
a[23][22] = 8;
a[23][17] = 5;
a[23][29] = 3;
a[24][18] = 7;
a[24][25] = 4;
a[25][24] = 4;
a[25][19] = 1;
a[25][26] = 2;
a[26][25] = 2;
a[26][20] = 3;
a[26][27] = 3;
a[27][26] = 3;
a[27][21] = 8;
a[27][28] = 7;
a[28][27] = 7;
a[28][22] = 6;
a[28][29] = 1;
a[29][28] = 1;
a[29][23] = 3;
for (int i = 0; i < vertex; i++)
{
    for (int j = 0; j < vertex; j++)
        cout << a[i][j] << " ";
    cout << endl;
}
```

```

for (int i = 0; i < vertex; i++)
{
    d[i] = maximalNumber;
    visited[i] = 1;
}
int start, finish;
cout << "From vertex :";
cin >> start;
start--;
cout << "To: ";
cin >> finish;
finish--;
int begin_index = start;
d[begin_index] = 0;
do {
    minindex = maximalNumber;
    min = maximalNumber;
    for (int i = 0; i < vertex; i++)
    {
        if ((visited[i] == 1) && (d[i] < min))
        {
            min = d[i];
            minindex = i;
        }
    }
    if (minindex != maximalNumber)
    {
        for (int i = 0; i < vertex; i++)
        {
            if (a[minindex][i] > 0)
            {
                temp = min + a[minindex][i];
                if (temp < d[i])
                {
                    d[i] = temp;
                }
            }
        }
        visited[minindex] = 0;
    }
} while (minindex < maximalNumber);
cout << "Minimal ways to vertex: " << endl;
for (int i = 0; i < vertex; i++)cout << d[i] << " ";
bool flag = false;
for (int i = 0; i < vertex; i++)if (d[i] != 0 && d[i] != maximalNumber)flag = true;

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

```

```

        ver[k] = i + 1;
        k++;
    }
}
cout << endl << "Print minimal way" << endl;
for (int i = k - 1; i >= 0; i--) cout << ver[i] << " ";
}
else {
    cout << "There isnt such way";
}
return 0;
}

```

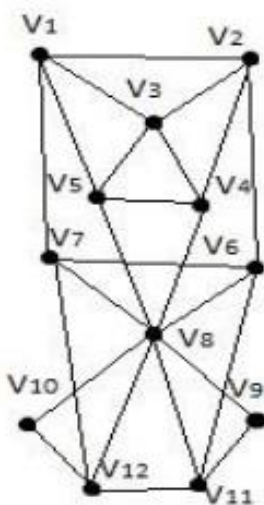
```

From vertex :1
To:30
Minimal ways to vertex:
0 1 7 13 20 20 3 3 4 12 14 15 7 6 5 12 16 18 10 8 9 14 17 23 13 9 11 14 21 22
Print minimal way
1 2 8 14 20 26 27 28 29 30
Process finished with exit code 0

```

Завдання 8

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



Флері

будемо видаляти ребра одне за одним, перевіряючи кожен раз перед цим чи не є це ребро мостом.

Шлях ейлерового циклу: v1 v3 v5 v8 v7 v6 v8 v4 v5 v1 v7 v12 v8 v10 v12 v11 v8 v9 v11 v6 v2 v4 v3 v2 v1. Всі ребра пройдено і ми повернулись у початкову вершину.

Елементарних циклів

будемо виділяти елементарні цикли та об'єднювати їх.

Шлях: v12 v10 v8 v12 v11 v8 v9 v11 v6 v8 v7 v6 v2 v4 v8 v5 v4 v3 v2 v1 v3 v5 v1 v7 v12.

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

#define N 12
#define STACK_SIZE 1000

int G[N][N] =
{
    {0,1,1,0,1,0,1,0,0,0,0,0},
    {1,0,1,1,0,1,0,0,0,0,0,0},
    {1,1,0,1,1,0,0,0,0,0,0,0},
    {0,1,1,0,1,0,0,1,0,0,0,0},
    {1,0,1,1,0,0,0,1,0,0,0,0},
    {0,1,0,0,0,0,1,1,0,0,1,0},
    {1,0,0,0,0,1,0,1,0,0,0,1},
    {0,0,0,1,1,1,1,0,1,1,1,1},
    {0,0,0,0,0,0,0,1,0,0,1,0},
    {0,0,0,0,0,0,0,1,0,0,0,1},
    {0,0,0,0,0,1,0,1,1,0,0,1},
    {0,0,0,0,0,0,1,1,0,1,1,0}
};

int k;
int Stack[STACK_SIZE];

void Search(int v)
{
    int i;
    for(i = 0; i < N; i++)
        if(G[v][i])
        {
            G[v][i] = G[i][v] = 0;
            Search(i);
        }
    Stack[++k] = v;
}

int main()
{
    int T, p, q, s;
```

```

int j, vv;

T = 1;
for(p = 0; p < N; p++)
{
    s = 0;
    for(q = 0; q < N; q++)
    {
        s += G[p][q];
    }
    if(s%2) T = 0;
}
k = -1;
printf("start vertex: "); scanf("%d", &vv);
if(T)
{
    Search (vv);
    for(j = 0; j <= k; j++)
        printf("%d ", Stack[j]);
}
else
    printf("not Eulerian graph\n");
return 0;
}

```

```

untitled10 x
D:\clion\untitled10\cmake-build-debug\untitled10.exe
start vertex: 1
1 5 10 11 9 7 10 8 7 11 6 7 4 3 7 5 6 0 4 2 3 1 2 0 1
Process finished with exit code 0

```

Завдання 9

Спростити формулу (привести до скороченої ДНФ).

$$x\bar{y}z \vee \bar{x}\bar{z} \vee xy$$

z	0	0	1	1
y\ x	0	1	1	0
0	1	0	1	0
1	1	1	1	0

z	0	0	1	1
y\z	0	1	1	0
0	1	0	1	0
1	1	1	1	0

В першій групі незмінними є x та z, вони дорівнювали 0, в другій – також x та z, вони дорівнювали 1, в третій – y та z, y дорівнював 1, а z – 0.

Отже скорочена ДНФ буде виглядати так:

$$\bar{x}\bar{z} \vee xz \vee y\bar{z}$$