Кафедра систем штучного інтелекту

Звіт

Розрахункова робота

З дисципліни

«Дискретна математика»

Виконав:

Студент групи КН-112

Бенчарський Максим

Викладач:

Мельникова Н. І.

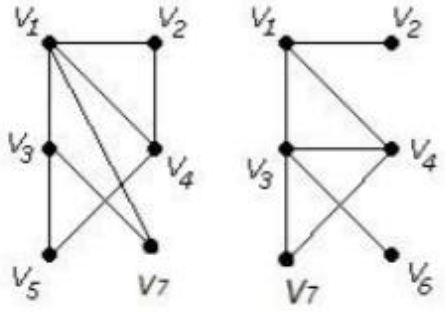
Львів-2019 р.

Варіант 9

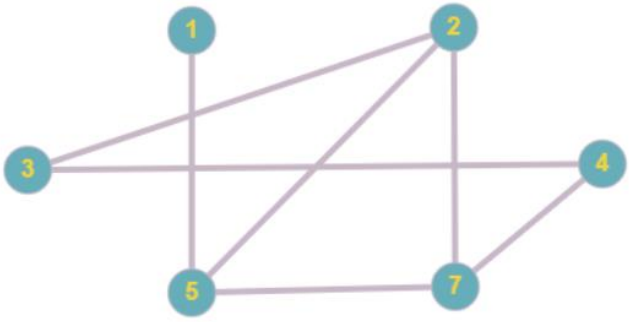
Завдання № 1

Виконати наступні операції над графами: 1) знайти доповнення до першого графу, 2) об'єднання графів, 3) кільцеву сумму G1 та G2 (G1+G2), 4) розмножити вершину у другому графі, 5) виділити підграф А - що скадається з 3-х вершин в G1 6) добуток графів.

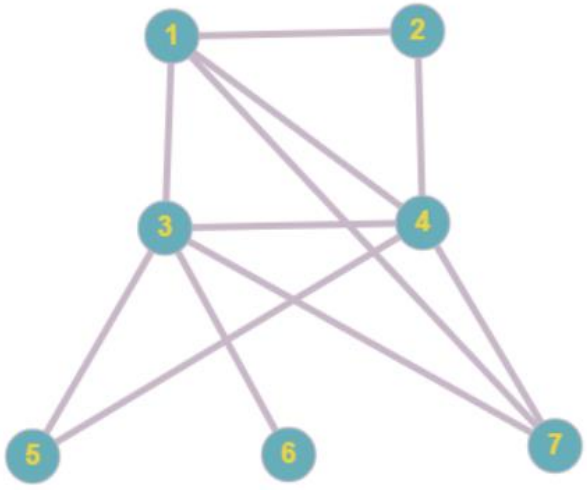
9)

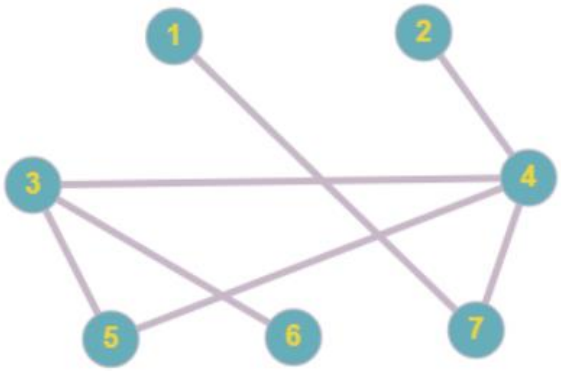


1)

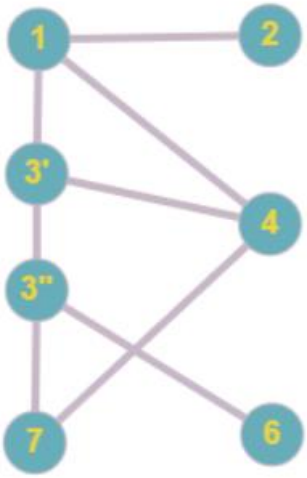


2)

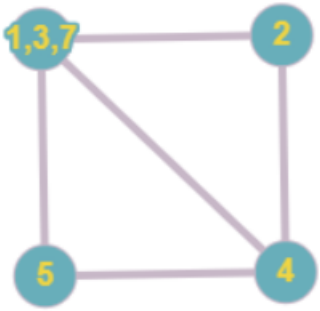




3)

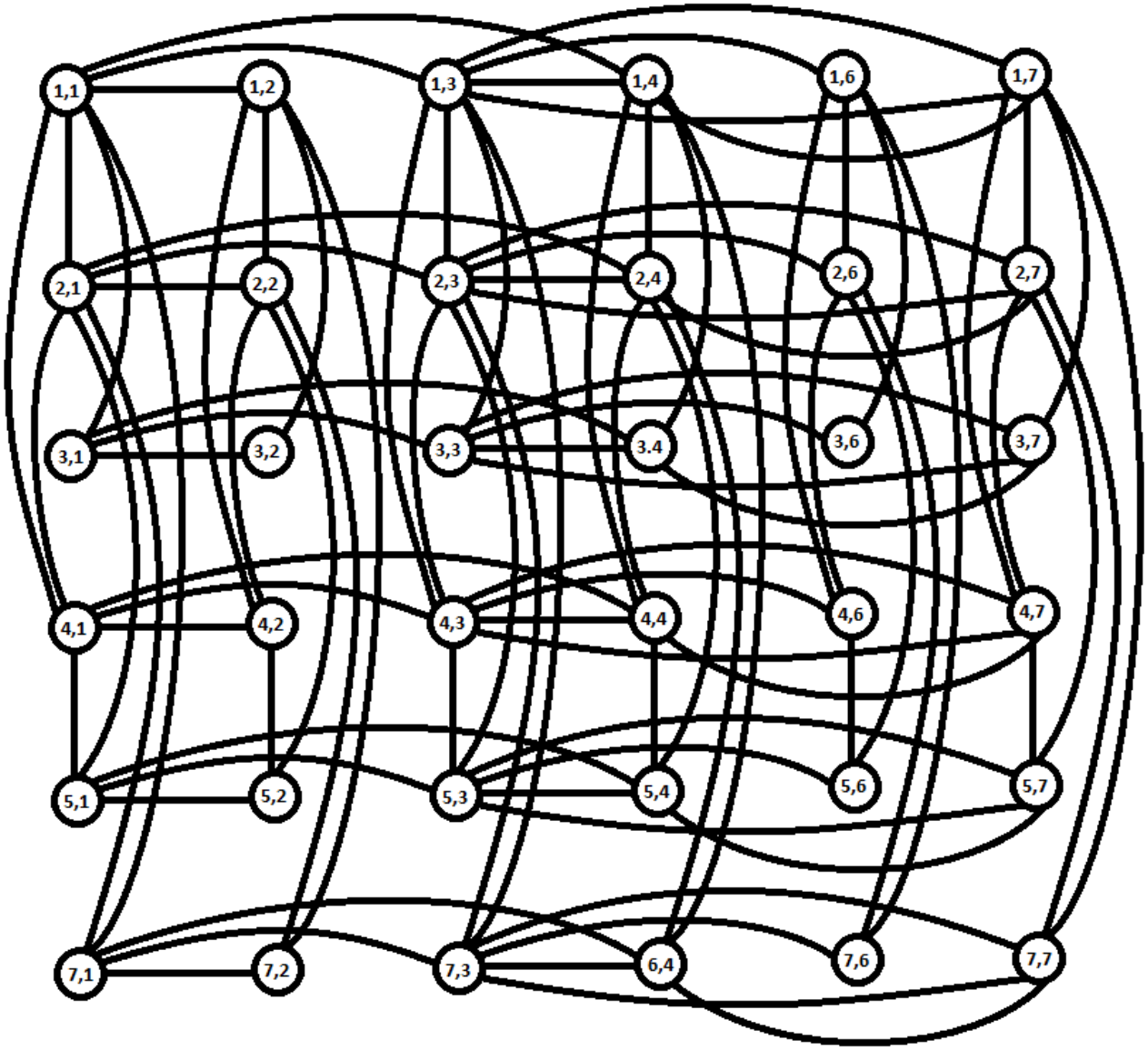


4)



5)

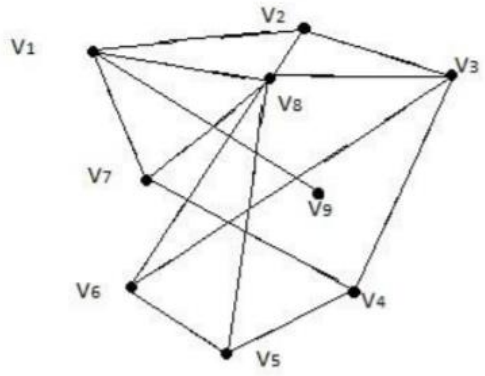
6)



Завдання № 2

Скласти таблицю суміжності для орграфа.

9)



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

Завдання № 3

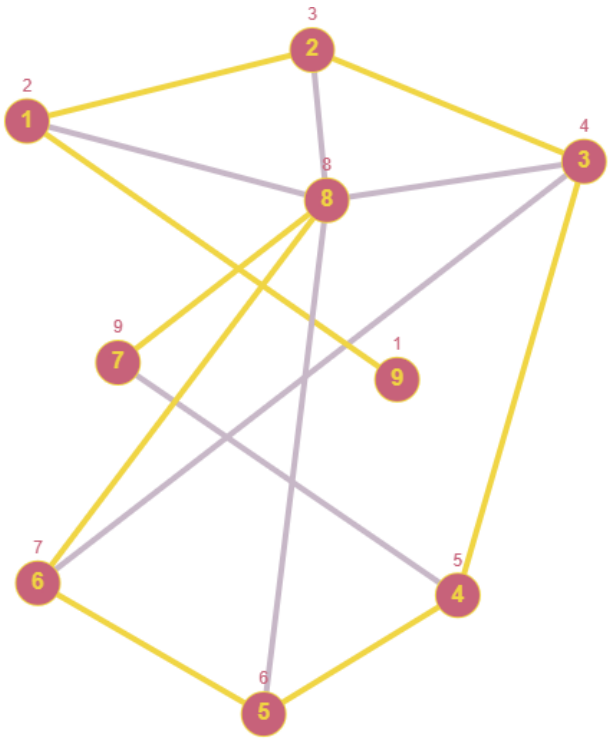
Для графа з другого завдання знайти діаметр.

Діаметр – 3.

Завдання № 4

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

V1	№	Стек
1	1	V1
2	2	V1,V2
3	3	V1,V2,V3
4	4	V1,V2,V3,V4
5	5	V1,V2,V3,V4,V5
6	6	V1,V2,V3,V4,V5,V6
8	7	V1,V2,V3,V4,V5,V6,V8
7	8	V1,V2,V3,V4,V5,V6,V8,V7
-	-	V1,V2,V3,V4,V5,V6,V8
-	-	V1,V2,V3,V4,V5,V6
-	-	V1,V2,V3,V4,V5
-	-	V1,V2,V3,V4
-	-	V1,V2,V3
-	-	V1,V2
-	-	V1
-	-	Ø




```
#include <iostream>
using namespace std;
const int n = 9;
bool* visited = new bool[n];
int graph[n][n] =
        {
                { 0, 1, 0, 0, 0, 0, 1, 1, 1 },
                { 1, 0, 1, 0, 0, 0, 0, 1, 0 },
                { 0, 1, 0, 1, 0, 1, 0, 1, 0 },
                { 0, 0, 1, 0, 1, 0, 1, 0, 0 },
                { 0, 0, 0, 1, 0, 1, 0, 1, 0 },
                { 0, 0, 1, 0, 1, 0, 0, 1, 0 },
                { 1, 0, 0, 1, 0, 0, 0, 1, 0 },
                { 1, 1, 1, 0, 1, 1, 1, 1, 0 },
                { 1, 0, 0, 0, 0, 0, 0, 0, 0 },
        };
void DFS(int st)
```

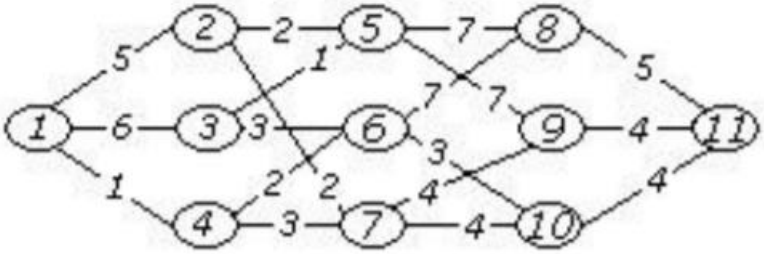
```
{
    int r;
    cout << st + 1 << " ";
    visited[st] = true;
    for (r = 0; r <= n; r++)
        if ((graph[st][r] != 0) && (!visited[r]))
            DFS(r);
}
int main()
{
    int start;
    int i, j;
    cout << "The adjency matrix: " << endl;
    for (i = 0; i < n; i++)
    {
        visited[i] = false;
        for (j = 0; j < n; j++)
            cout << " " << graph[i][j];
        cout << endl;
    }
    cout << "Enter the start point: "; cin >> start;
    bool* vis = new bool[n];
    cout << "\t Crawl order: ";
    DFS(start - 1);
    delete[]visited;
}
```

```
C:\Users\Max\Desktop\lbs\rozrahash
The adjency matrix:
 0 1 0 0 0 0 1 1 1
 1 0 1 0 0 0 0 1 0
 0 1 0 1 0 1 0 1 0
 0 0 1 0 1 0 1 0 0
 0 0 0 1 0 1 0 1 0
 0 0 1 0 1 0 0 1 0
 1 0 0 1 0 0 0 1 0
 1 1 1 0 1 1 1 1 0
 1 0 0 0 0 0 0 0 0
Enter the start point:9
   Crawl order: 9 1 2 3 4 5 6 8 7
Process finished with exit code 0
```

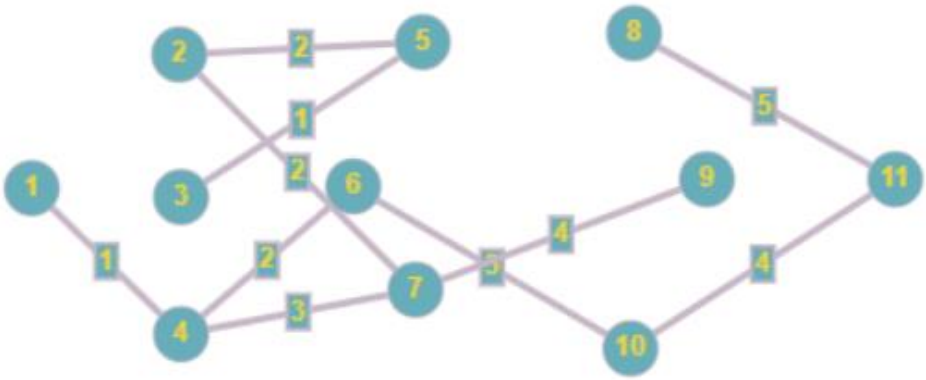
Завдання № 5

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

9)



Краскла:



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

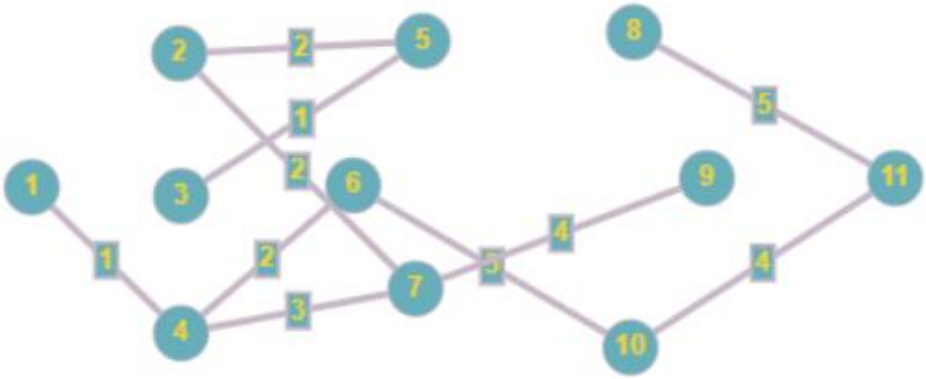
E:{(1,4),(3,5),(4,6),(2,7),(2,5),(4,7),(7,9),(10,11),(7,9),(11,8)}

```
#include <iostream>

using namespace std;

void Probe(int v, int AM[][11], int Values[], int Lines[]);

int main()
{ //матриця суміжності
    int AM[11][11] =
            {   {99,5,6,1,99,99,99,99,99,99,99},
                {5,99,99,99,2,99,2,99,99,99,99},
                {6,99,99,99,1,3,99,99,99,99,99},
                {1,99,99,99,99,2,3,99,99,99,99},
                {99,2,1,99,99,99,99,7,7,99,99},
                {99,99,3,2,99,99,99,7,99,3,99},
                {99,2,99,3,99,99,99,99,4,4,99},
                {99,99,99,99,7,7,99,99,99,99,5},
                {99,99,99,99,7,99,4,99,99,99,4},
                {99,99,99,99,99,3,4,99,99,99,4},
                {99,99,99,99,99,99,99,5,4,4,99}
            };

    //вага ребер
    int Values[]={1,2,3,4,5,6,7}; // "v"
    int value = (sizeof(Values))/4;

    //масив для запису пройдених ребер
    int Lines[11];
    //занулити
    for (int i=0;i<value;i++)
    {
```

```
        Lines[i]=0;
    }
    //виклик функції
    cout << "\n Line\t| Weight"<<endl;
    cout << "--------|---------" << endl;

    for (int weight=0;weight<value;weight++)
    {
        Probe(weight, AM, Values, Lines);
    }

    return 0;
}
void Probe(int v, int AM[][11], int Values[], int Lines[])
{
    int counter1 = 0;
    int counter2 = 0;
    bool flag1, flag2;

    for (int i=0;i<11;i++)
    {
        for (int j=0;j<11;j++)
        {
            if (AM[i][j]==Values[v])
            {
                for (int x=0;x<11;x++)
                {
                    if (Lines[x]!=i)    //якщо нема ще такого ребра
                    {
                        counter1++;
                    }
                    if (Lines[x]!=j)    //якщо нема ще такого ребра
                    {
                        counter2++;
                    }
                }
                if (counter1==11)
                {
                    Lines[i]=i;
                    flag1 = true;
                }
                if (counter2==11)
                {
                    Lines[j]=j;
                    flag2 = true;
                }
                if ((flag1==false)&&(flag2==false))
                {
                    //  cout << "False flags!";

                }
                else {
                    cout << "{" << Lines[i]+1 << ";" << Lines[j]+1 << "}\t| ";
                    cout << v+1 << endl;
                }

            }
            counter1=0;
            counter2=0;
            flag1 = false;
            flag2 = false;
        }
    }
}
```

```
C:\Users\Max\Desktop\lbs\rozrahash

 Line    | Weight
---------|---------
{1;4}    | 1
{3;5}    | 1
{2;5}    | 2
{2;7}    | 2
{4;6}    | 2
{6;10}   | 3
{7;9}    | 4
{9;11}   | 4
{8;11}   | 5

Process finished with exit code 0
```

Прима:



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

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

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

int main()
{

    int v, count = 0, min = 0, k,t;
```

```
    bool check = false;
    v = 11;
    int* tops = new int[v];

    int** rebra = new int* [v - 1];

    for (int i = 0; i < v - 1; i++) {
        rebra[i] = new int[2];
    }

    int matrix[11][11] =
            {   {99,5,6,1,99,99,99,99,99,99,99},
                {5,99,99,99,2,99,2,99,99,99,99},
                {6,99,99,99,1,3,99,99,99,99,99},
                {1,99,99,99,99,2,3,99,99,99,99},
                {99,2,1,99,99,99,99,7,7,99,99},
                {99,99,3,2,99,99,99,7,99,3,99},
                {99,2,99,3,99,99,99,99,4,4,99},
                {99,99,99,99,7,7,99,99,99,99,5},
                {99,99,99,99,7,99,4,99,99,99,4},
                {99,99,99,99,99,3,4,99,99,99,4},
                {99,99,99,99,99,99,99,5,4,4,99}
            };

    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++) {

                    if (tops[m] == a + 1) {
                        check = true;
                    }
                }

                if (check) { check = false; continue; }

                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];

                    k = rebra[count - 1][0] = tops[j]; t = rebra[count - 1][1] = a + 1;
                }
            }
        }

        matrix[k - 1][t - 1] = 0; matrix[t - 1][k - 1] = 0;

        tops[count] = t;
        count++;
        min = 0;
    }
```

```
    for (int j = 0; j < v - 1; j++) {
        cout  << "{" << rebra[j][0] << ";" << rebra[j][1] << "}" ;
        if (j != v-2) cout << " -> ";
    }

    return 0;

}
```

```
C:\Users\Max\Desktop\lbs\rozrahashit\cmake-build-debug\rozrahashit.exe
{1;4} -> {4;6} -> {4;7} -> {7;2} -> {2;5} -> {5;3} -> {6;10} -> {7;9} -> {10;11} -> {11;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:

Найкоротші шдяхи:

1) 1-8-7-4-5-2-6-3-1 d=19
2) 2-6-8-1-5-4-7-3-2 d=19
3) 3-2-6-8-1-5-4-7-3 d=19
4) 4-5-8-1-6-2-3-7-4 d=19
5) 5-4-7-8-1-6-2-3-5 d=19
6) 6-2-5-4-7-8-1-3-6 d=19
7) 7-4-5-2-6-8-1-3-7 d=19
8) 7-4-5-8-1-6-2-3-7 d=19

9)

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

```cpp
#include<iostream>
#include<vector>
using namespace std;
int counter = 0, Inf = 9999;

bool check(vector<int> q, int Node);

int F_Min(vector<int>* q, int arr[][8], int n, int i);

void Find(vector<int>* q, int arr[][8], int n, int pos, vector<int>* qq);

int main()
{
    int n;
    n = 8;
    int arr[][8] = {
            { 0,5,5,3,3,4,4,1 },
            { 5,0,4,3,2,1,4,6 },
            { 5,4,0,4,5,6,5,5 },
            { 3,3,4,0,1,5,1,7 },
            { 3,2,5,1,0,5,5,2 },
            { 4,1,6,5,5,0,7,3 },
            { 4,4,5,1,5,7,0,2 },
            { 1,6,5,7,2,3,2,0 },
    };

    vector<int> q;
    vector<int> qq;
    cout << endl;

    for (int i = 0; i < n; i++) {
        q.clear();
        q.push_back(i);
        Find(&q, arr, n, i, &qq);
    }

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




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


int F_Min(vector<int>* q, int arr[][8], int n, int i) {
    int min = 999;
    for (int j = 0; j < n; j++) {
        if (arr[i][j] < min && arr[i][j] != 0 && check((*q), j))min = arr[i][j];
    }
    return min;
}

void Find(vector<int>* q, int arr[][8], int n, int pos, vector<int>* qq) {
```

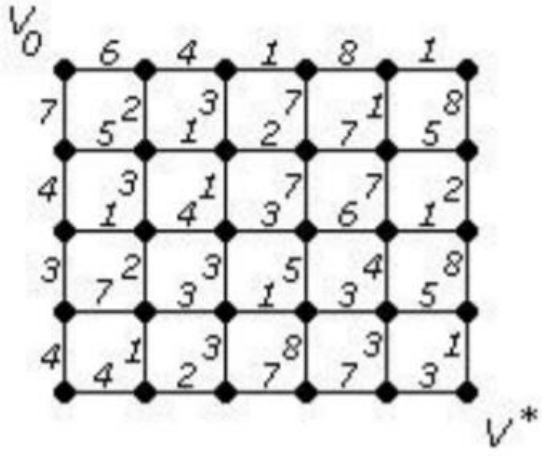
```
    int min;
    for (int i = pos, k = 0; k < 1; i++, k++) {
        min = F_Min(q, arr, n, i);
        for (int j = 0; j < n; j++) {
            if (arr[i][j] == min && check((*q), j)) {
                (*q).push_back(j);
                Find(q, arr, n, j, qq);
            }
        }
        if (q->size() == n) {
            (*q).push_back((*q)[0]);
            counter = 0;

            for (int l = 1; l <= n; l++) {
                counter += arr[(*q)[l - 1]][(*q)[l]];
            }

            if (Inf == counter) {
                for (int b = 0; b <= n; b++) {
                    (*qq).push_back((*q)[b]);
                }
                (*qq).push_back(counter);
            }

            else if (Inf > counter) {
                (*qq).clear();

                for (int b = 0; b <= n; b++) {
                    (*qq).push_back((*q)[b]);

                }
                (*qq).push_back(counter);

                Inf = counter;
            }
            q->pop_back();
        }
    }
    q->pop_back();
}
```

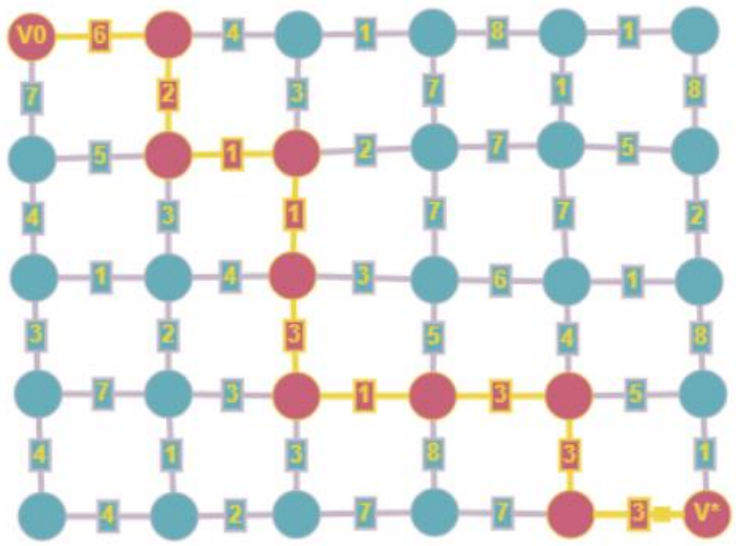
```
C:\Users\Max\Desktop\lbs\rozrahash

1 8 7 4 5 2 6 3 1  {19}
2 6 8 1 5 4 7 3 2  {19}
3 2 6 8 1 5 4 7 3  {19}
4 5 8 1 6 2 3 7 4  {19}
5 4 7 8 1 6 2 3 5  {19}
6 2 5 4 7 8 1 3 6  {19}
7 4 5 2 6 8 1 3 7  {19}
7 4 5 8 1 6 2 3 7  {19}

Process finished with exit code 0
```

Завдання № 7

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

9)





Відстань – 23.

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

#include <iostream>
using namespace std;
const int SIZE = 30;
int main()
{
    int Matrix[][30] = {
            { 0,6,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }, /*1*/
            { 6,0,4,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
            { 0,4,0,1,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }, /*3*/
            { 0,0,1,0,8,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
            { 0,0,0,8,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }, /*5*/
            { 0,0,0,0,1,0,0,0,0,0,0,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
            { 7,0,0,0,0,0,0,5,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }, /*7*/
            { 0,2,0,0,0,0,5,0,1,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
            { 0,0,3,0,0,0,0,1,0,2,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 }, /*9*/
            { 0,0,0,7,0,0,0,0,2,0,7,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
            { 0,0,0,0,1,0,0,0,0,7,0,5,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0 }, /*11*/
            { 0,0,0,0,0,8,0,0,0,0,5,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0 },
            { 0,0,0,0,0,0,4,0,0,0,0,0,0,1,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0 }, /*13*/
            { 0,0,0,0,0,0,0,3,0,0,0,0,1,0,4,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0 },
            { 0,0,0,0,0,0,0,0,1,0,0,0,0,4,0,3,0,0,0,0,3,0,0,0,0,0,0,0,0,0 }, /*15*/
            { 0,0,0,0,0,0,0,0,0,7,0,0,0,0,3,0,6,0,0,0,0,5,0,0,0,0,0,0,0,0 },
            { 0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,6,0,1,0,0,0,0,4,0,0,0,0,0,0,0 }, /*17*/
            { 0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,1,0,0,0,0,0,0,8,0,0,0,0,0,0 },
            { 0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,0,0,7,0,0,0,0,4,0,0,0,0,0 }, /*19*/
            { 0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,7,0,3,0,0,0,0,1,0,0,0,0 },
            { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,3,0,1,0,0,0,0,3,0,0,0 }, /*21*/
            { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,0,0,0,0,1,0,3,0,0,0,0,8,0,0 },
            { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,0,0,0,0,3,0,5,0,0,0,0,3,0 }, /*23*/
            { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,0,0,0,0,5,0,0,0,0,0,0,1 },
            { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,0,0,0,0,0,0,4,0,0,0,0 }, /*25*/
            { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,4,0,2,0,0,0 },
            { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,2,0,7,0,0 }, /*27*/
            { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,0,0,0,0,7,0,7,0 },
            { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,7,0,3 }, /*29*/
            { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,3,0 }
    };
    cout << "The adjency matrix.\n" << endl;
    for (int i =0;i<SIZE; i++)
    {
```

```
        for (int j=0;j<SIZE; j++)
        {
            cout << Matrix[i][j] << " ";
        }
        cout << endl;
    }
    int dis[SIZE]; // відстань
    int visited[SIZE]; //відвідані вершини
    int minindex, min;
    int startpoint = 0;
    int split = SIZE/3;
    for (int i = 0; i<SIZE; i++)
    {
        dis[i] = 10000; //відстань до інших вершин
        visited[i] = 1; //позначити як невідвідано
    }
    dis[startpoint] = 0;
    do {
        minindex = 10000;
        min = 10000;
        for (int i = 0; i<SIZE; i++)
        {
            if ((visited[i] == 1) && (dis[i]<min))
            {
                min = dis[i];
                minindex = i;
            }
        }
        if (minindex != 10000)
        {
            for (int i = 0; i<SIZE; i++)
            {
                if (Matrix[minindex][i] > 0)
                {
                    int point = min + Matrix[minindex][i]; //додати знайдену мін вагу
до існуючої ваги вершини
                    if (point < dis[i]) // порівняти з потоною вагою
                    {
                        dis[i] = point;
                    }
                }
            }
            visited[minindex] = 0;
        }
    } while (minindex < 10000);
// Восстановление пути
    int endy;
    endy = 30;
    int end = endy-1;
    int seen[SIZE]; // масив відвіданих вершин
    seen[0] = end + 1; // початковий элемент - кінцева вершина
    int weight = dis[end]; // вага кінцева вершини
    int k = 1; // індекс попередньої
    while (split!=0) // пока не початок
    {
        for (int i = 0; i<SIZE; i++)
        {
            if ((Matrix[end][i] != 0)&&(Matrix[end][i] != seen[k])) // якщо вершини
суміжні
            {
                int point = weight - Matrix[end][i];
                if (point == dis[i]) // якщо вага не співпадає
                {
                    weight = point;
```

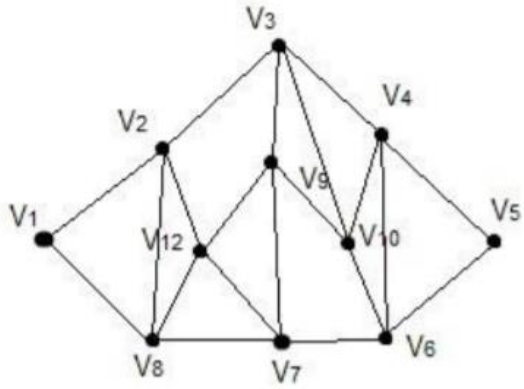
```
                    end = i; // зберігаємо попередню вершину
                    seen[k] = i + 1; // і записуємо її в масив
                    k++;
                }
            }
        }
        split--;
    }
//Шлях
    cout<<"\nThe shortest way from start point to the end\n";
    for (int i = k-1; i >= 0; i--)
    {
        if (i) cout << seen[i] << " > ";
        else cout << seen[i];
    }
    cout << "\n Total weight = " << dis[SIZE-1] << endl;
    return 0;
}
```

```
0 6 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9
6 0 4 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 4 0 1 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 8 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 8 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
7 0 0 0 0 0 0 0 5 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 2 0 0 0 0 5 0 1 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 3 0 0 0 0 1 0 2 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 7 0 0 0 0 2 0 7 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 7 0 5 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 8 0 0 0 0 5 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 4 0 0 0 0 0 0 1 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 3 0 0 0 0 1 0 4 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 4 0 3 0 0 0 0 3 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 7 0 0 0 0 3 0 6 0 0 0 0 5 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 6 0 1 0 0 0 0 4 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 1 0 0 0 0 0 0 8 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 7 0 0 0 0 4 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 7 0 3 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 3 0 1 0 0 0 0 3 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 1 0 3 0 0 0 0 8 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 3 0 5 0 0 0 0 3 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 0 0 0 0 5 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 0 0 0 0 0 0 4 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 4 0 2 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 2 0 7 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 0 0 0 0 7 0 7 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 0 0 0 7 0 3
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 3 0

The shortest way from start point to the end
1 > 2 > 8 > 9 > 15 > 21 > 22 > 23 > 24 > 30
 Total weight = 23
```

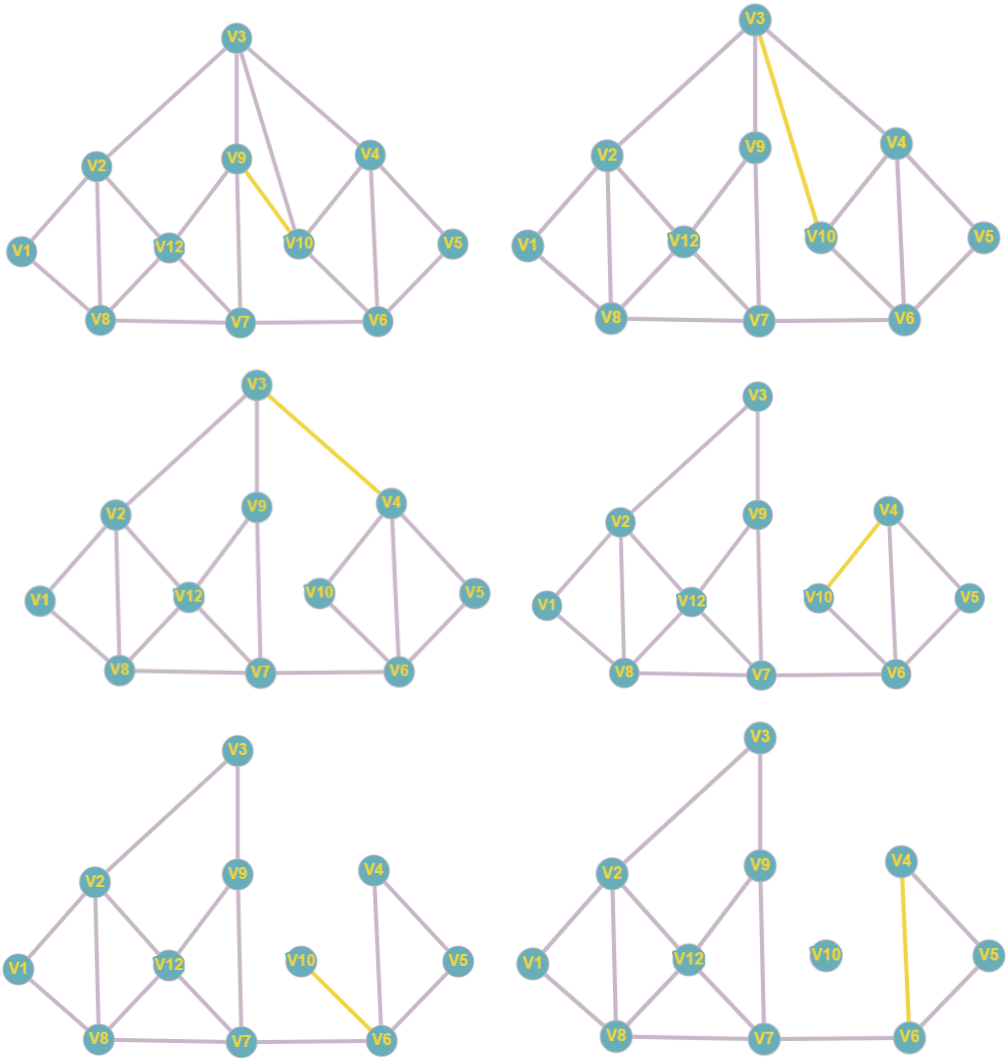
Завдання № 8

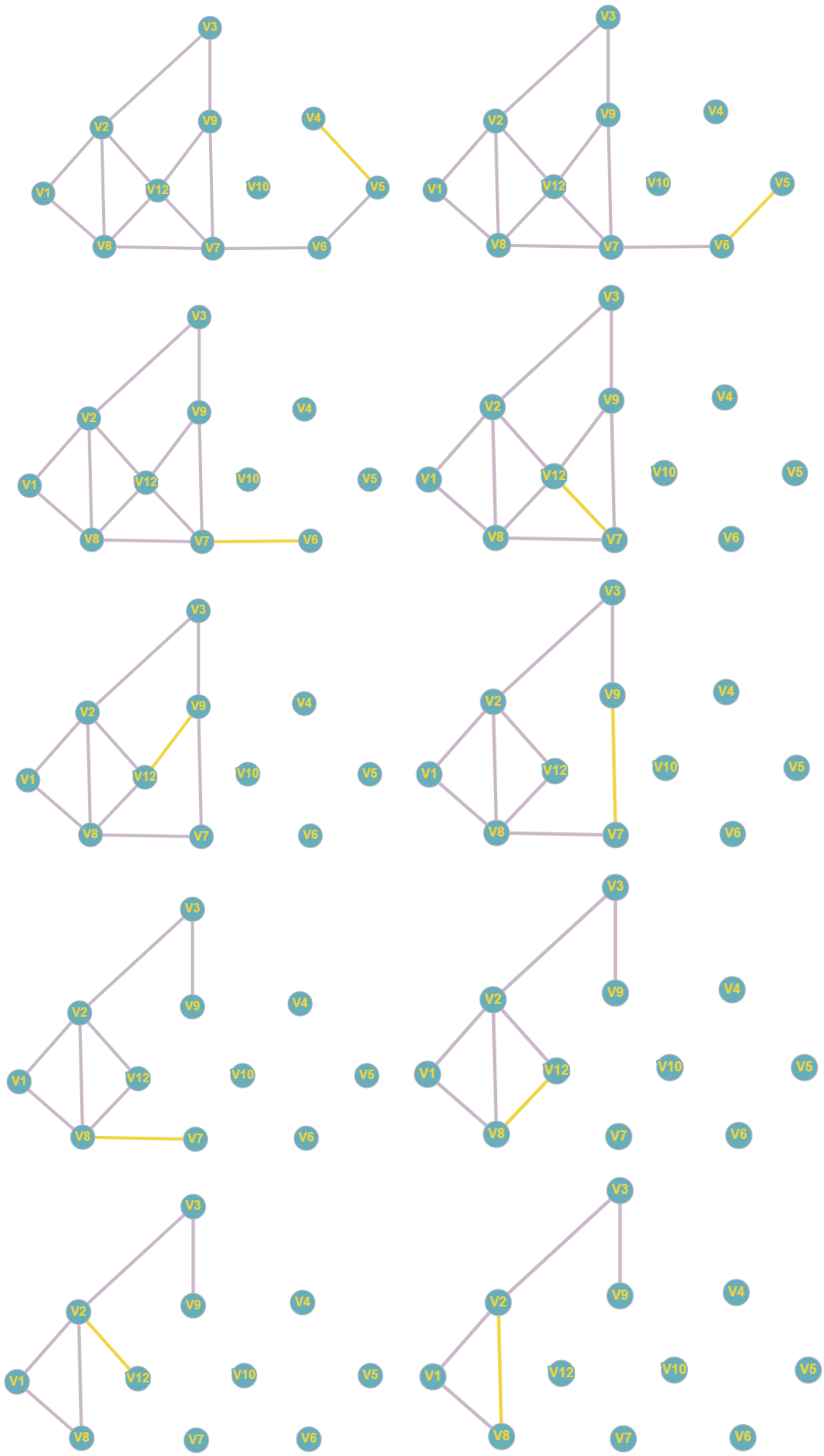
Знайти ейлеровий цикл в ейлеровому графі двома методами: а) Флері; б) елементарних циклів.

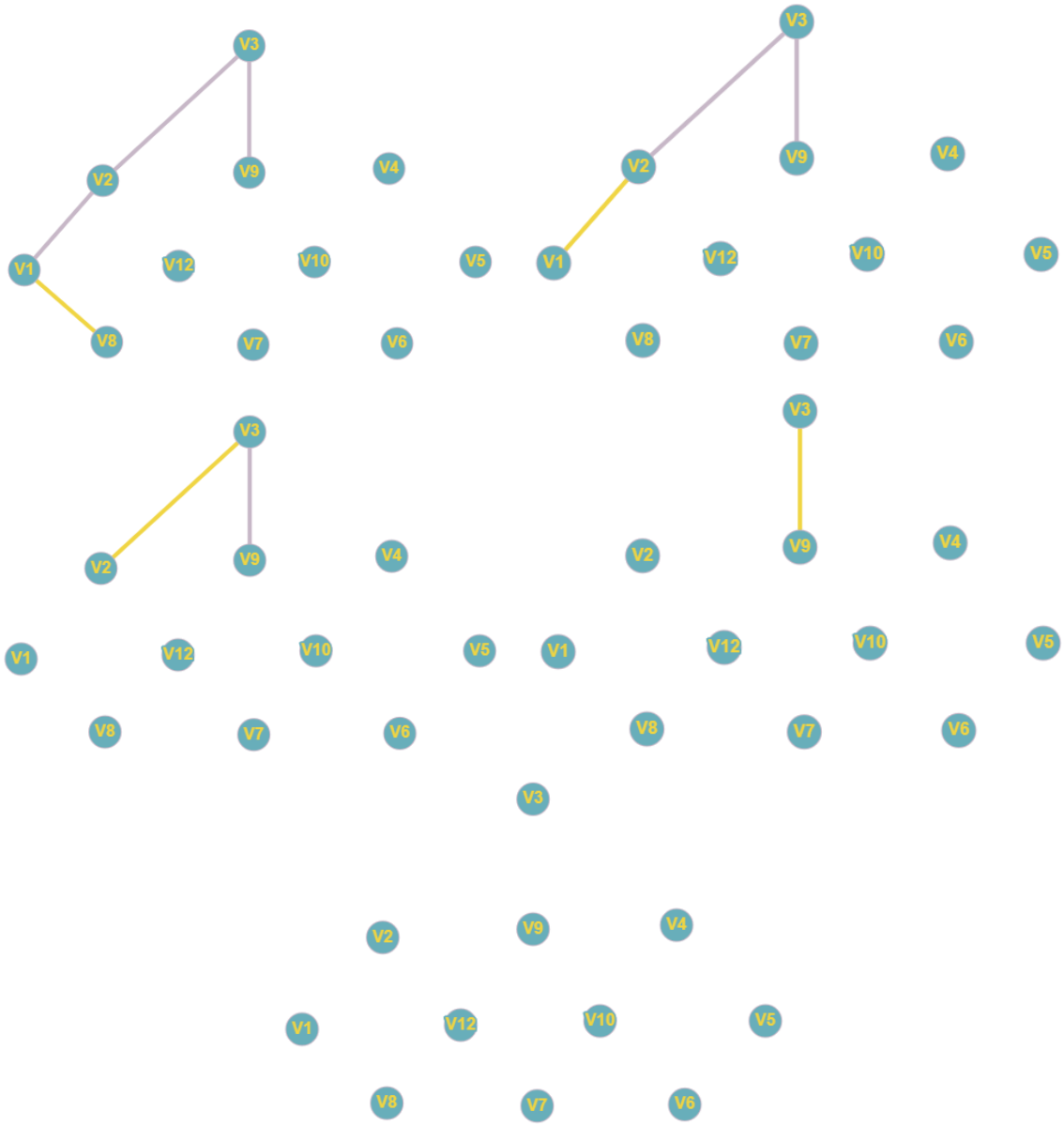
9)



а)




V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V12



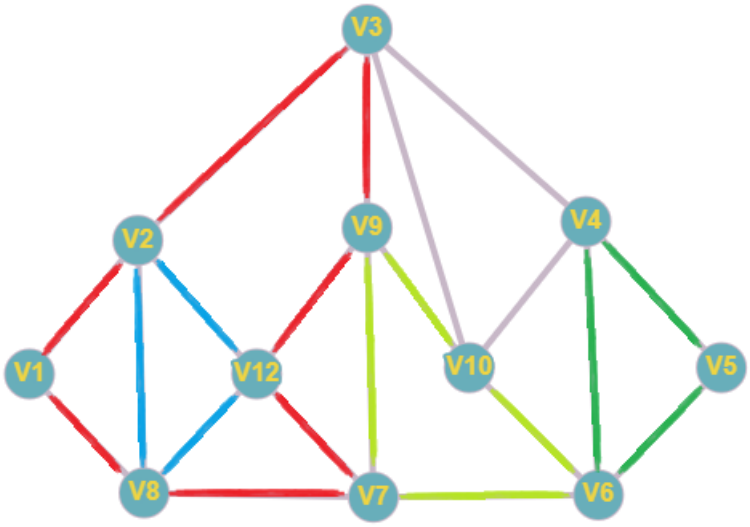
V9-V10-V3-V4-V10-V6-V4-V5-V6-V7-V12-V9-V7-V8-V12-V2-V8-V1-V2-V3-V9.

```
#include<iostream>
using namespace std;
void Search(int v, int G[][12], int N)
{
    int i;
    for (i = 0; i < N; i++) {
        if (G[v][i])
        {
            G[v][i] = G[i][v] = 0;
            Search(i, G, N);
        }
    }
    cout << v + 1 << " => ";
}
int main()
{
    int N = 0;
    N = 12;
```

```
    int G[][12] = {
            { 0,1,0,0,0,0,0,1,0,0,0,0 },
            { 1,0,1,0,0,0,0,1,0,0,0,1 },
            { 0,1,0,1,0,0,0,0,1,1,0,0 },
            { 0,0,1,0,1,1,0,0,0,1,0,0 },
            { 0,0,0,1,0,1,0,0,0,0,0,0 },
            { 0,0,0,1,1,1,0,0,0,1,0,0 },
            { 0,0,0,0,0,1,0,1,1,0,0,1 },
            { 1,1,0,0,0,0,1,0,0,0,0,1 },
            { 0,0,1,0,0,0,1,0,1,0,0,1 },
            { 0,0,1,1,0,1,0,0,1,0,0,0 },
            { 0,0,0,0,0,0,0,0,0,0,0,0 },
            { 0,1,0,0,0,0,1,1,1,0,0,0 },
    };
    int count, p, q, sum;
    count = 1;
    for (p = 0; p < N; p++){
        sum = 0;
        for (q = 0; q < N; q++){
            sum += G[p][q];
        }
        if (sum % 2) count = 0;
    }
    cout << endl;
    if (count)
        Search(0, G, N);
    else
        cout << "Something's wrong :/";
    cout << endl;
    return 0;
}
```

```
C:\Users\Max\Desktop\lbs\rozrahashit\cmake-build-debug\rozrahashit.exe

1 => 8 => 12 => 7 => 8 => 2 => 12 => 9 => 9 => 10 => 6 => 6 => 7 => 9 => 3 => 10 => 4 => 6 => 5 => 4 => 3 => 2 => 1

Process finished with exit code 0
```

б)



Об’єднавши елементарні цикли отримаємо ейлеровий цикл: V7-V12-V9-V3-V2-V1-V8-V12-V2-V8-V7-V9-V10-V3-V4-V10-V6-V4-V5-V6-V7.

```
#include <iostream>
#include <vector>
using namespace std;
vector<int> Vcon;
int Inf = 999;
bool check(vector<int> V, int pork) {
    for (auto i = V.begin(); i != V.end(); i++) {
        if (*i == pork)return false;
    }
    return true;
}
void Find(vector<int>* V, int arr[][12], int n, int pos, int start_pork) {
    for (int i = pos, k = 0; k < 1; i++, k++) {
        for (int j = 0; j < n; j++)
            if (arr[i][j] == 1 && check((*V), j)) {
                if (j == start_pork && (*V).size() > 2) {
                    if (Inf > V->size()) {
                        Vcon.clear();
                        Vcon.push_back(start_pork + 1);
                        for (auto it = (*V).begin(); it != (*V).end();
                             it++)
                            Vcon.push_back(*it + 1);
                        Vcon.push_back(start_pork + 1);
                        Inf = V->size();
                        break;
                    }
                }
                else {
                    (*V).push_back(j);
                    Find(V, arr, n, j, start_pork);
                }
            }
    }
    if (V->size() != 0)
        V->pop_back();
}
int main() {
    int n;
    n = 12;
    int arr[][12] = {
            { 0,1,0,0,0,0,0,1,0,0,0,0 },
            { 1,0,1,0,0,0,0,1,0,0,0,1 },
            { 0,1,0,1,0,0,0,0,1,1,0,0 },
            { 0,0,1,0,1,1,0,0,0,1,0,0 },
            { 0,0,0,1,0,1,0,0,0,0,0,0 },
            { 0,0,0,1,1,1,0,0,0,1,0,0 },
            { 0,0,0,0,0,1,0,1,1,0,0,1 },
            { 1,1,0,0,0,0,1,0,0,0,0,1 },
            { 0,0,1,0,0,0,1,0,1,0,0,1 },
            { 0,0,1,1,0,1,0,0,1,0,0,0 },
            { 0,0,0,0,0,0,0,0,0,0,0,0 },
            { 0,1,0,0,0,0,1,1,1,0,0,0 },
    };
    vector<int> V;
    vector<int> WAS;
    cout << endl;
    int count, p, q, sum;
    count = 1;
```

```
    for (p = 0; p < n; p++)
    {
        sum = 0;
        for (q = 0; q < n; q++)
        {
            sum += arr[p][q];
        }
        if (sum % 2) count = 0;
    }
    cout << endl;
    if (count) {
        for (int j = 0; j < n; j++) {
            Inf = 999;
            Find(&V, arr, n, j, j);
            for (int i = 1; i <= Vcon.size(); i++) {
                cout << Vcon[i - 1] << "";
                if (i < Vcon.size())
                    cout<<" -> ";
            }
            cout << endl;
            Vcon.clear();
        }
    }
    else
        cout << "You messed up...\n";
    cout << endl;
    return 0;
}
```

```
C:\Users\Max\Desktop\lbs\rozrahash

1 -> 2 -> 12 -> 8 -> 1
2 -> 1 -> 8 -> 12 -> 2
3 -> 2 -> 12 -> 9 -> 3
4 -> 3 -> 10 -> 6 -> 4
5 -> 4 -> 10 -> 6 -> 5
6 -> 4 -> 3 -> 10 -> 6
7 -> 6 -> 10 -> 9 -> 7
8 -> 1 -> 2 -> 12 -> 8
9 -> 3 -> 2 -> 12 -> 9
10 -> 3 -> 4 -> 6 -> 10

12 -> 2 -> 1 -> 8 -> 12


Process finished with exit code 0
```

Завдання №9

Спростити формули (привести їх до скороченої ДНФ).

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

(X→Y)(Y→Z)→(X→Z)

(XVY)(Y→Z)→(X→Z)

(XVY)(YVZ)→(X→Z)

(XYVXZVYYVYZ)→(X→Z)

(XYVXZVFVYZ)→(X→Z)

(XYVXZVYZ)→(X→Z)

(XYVYZ)→(X→Z)

(XYVYZ)→(XVZ)

XYVYZVXVZ

(XYYZ)VXVZ

((XVY)YZ)VXVZ

((XVY)YZ)VXVZ

((XVY)YZ)VXVZ

((XVY)(YVZ))VXVZ

XYVXZVYYVYZVXVZ

XYVXZVFVYZVXVZ

XYVXZVYZVXVZ

YZVXYVXVZ

XVYVZVZ

XVYVT

T