МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ

“ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Розрахункова робота

з дисципліни

«Дискретна математика»

Виконала:

студентка групи КН-112

Яцунда Софія

Викладач:

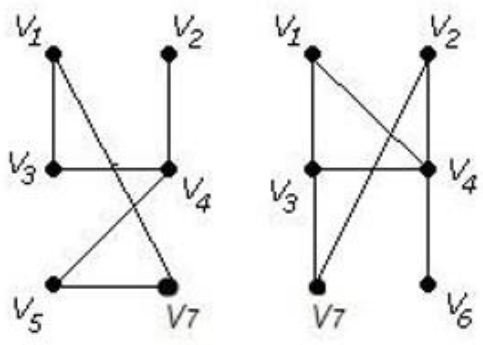
Мельникова Н.І.

Львів – 2019р.

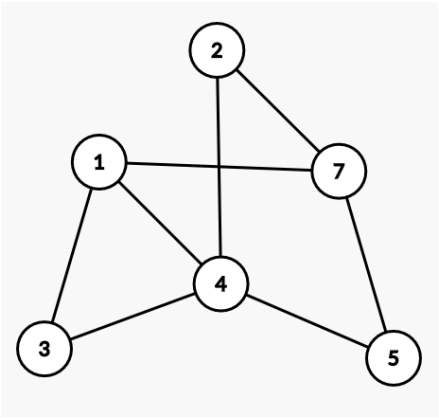
Варіант 17

Завдання № 1

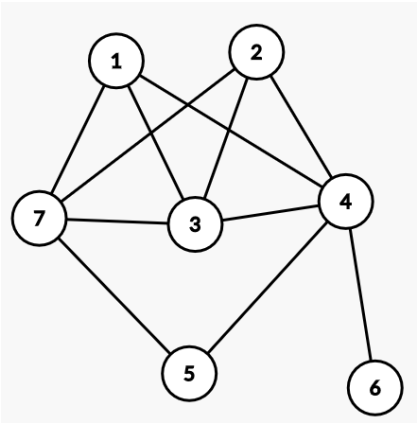
Виконати наступні операції над графами:



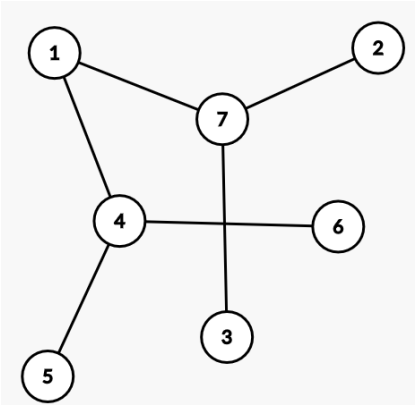
1)знайти доповнення до першого графу,



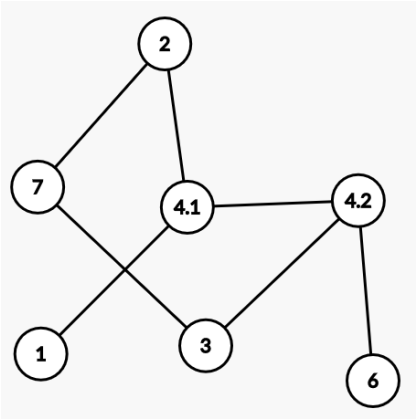
2) об'єднання графів,



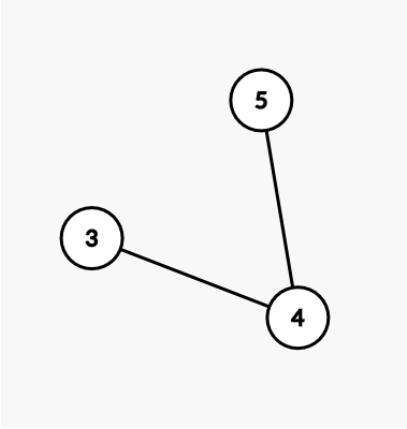
3) кільцеву суму G1 та G2 (G1+G2),



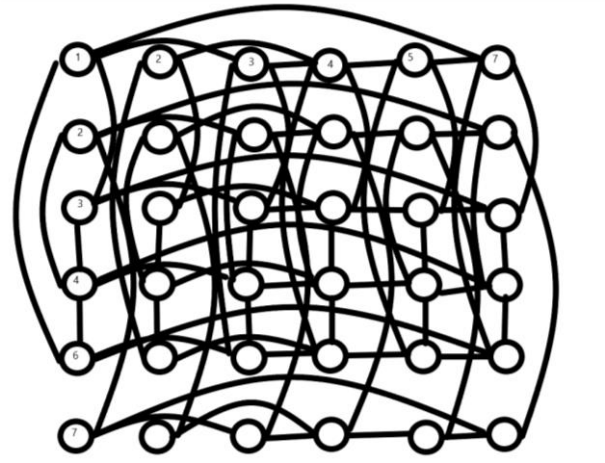
4) розмножити вершину у другому графі,



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

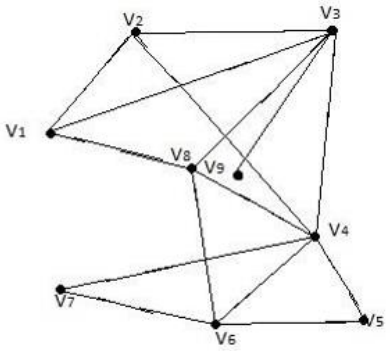


6) добуток графів.



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**	**0**	**1**	**0**	**0**	**0**	**0**	**0**

Завдання № 3

Для графа з другого завдання знайти діаметр.

Діаметр дорівнює 6;

Завдання № 4

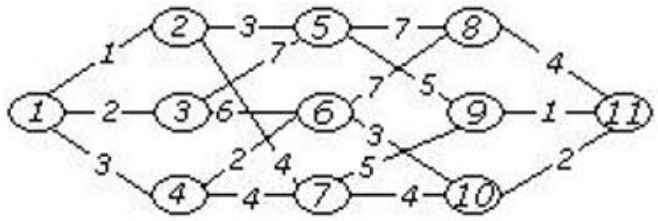
Для графа з другого завдання виконати обхід дерева вглиб (варіант закінчується на непарне число) або вшир (закінчується на парне число).

V1	**1**	**V1**
V2	**2**	**V1V2**
V3	**3**	**V1V2V3**
V4	**4**	**V1V2V3V4**
V5	**5**	**V1V2V3V4V5**
V6	**6**	**V1V2V3V4V5V6**
V7	**7**	**V1V2V3V4V5V6V7**
-	**-**	**V1V2V3V4V5V6**
V8	**8**	**V1V2V3V4V5V6V8**

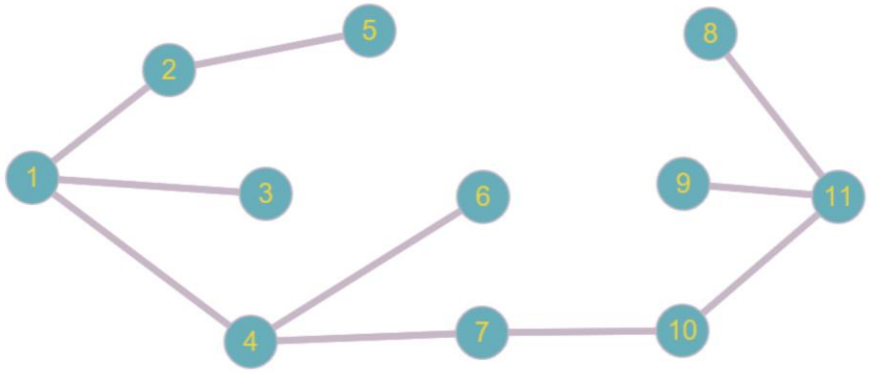
-	-	**V1V2V3V4V5V6**
-	-	**V1V2V3V4V5**
-	-	**V1V2V3V4**
-	-	**V1V2V3**
V9	**9**	**V1V2V3V9**
-	-	**V1V2V3**
-	-	**V1V2**
-	-	**V1**
-	-	¤

Завдання № 5

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



Метод 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++) {
```

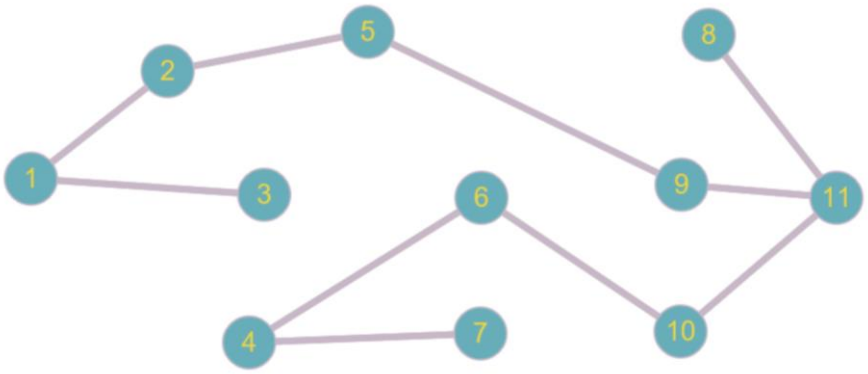
```
        for (int j = 0; j < 11; j++) {
            if (A[i][j] = r) {
                scn = i;
            }
        }
    }
    for (int i = 0; i < 11; i++) {
        for (int j = 0; j < 11; j++) {
            if (A[i][j] == t) {
                for (int k = 0; k < 11; k++) {
                    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},
    {1,0,0,0,3,0,4,0,0,0,0},
    {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},
    {1,4,0,4,0,0,0,0,5,4,0},
    {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},
    {0,0,0,0,0,0,0,0,4,1,2,0}
    };
    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

C:\Users\софія яцунда\source\repos\dm4\Debug\dm4.exe (процесс 8820) завершает работу с кодом 0.
Чтобы закрыть это окно, нажмите любую клавишу…
```

Метод Прима




```cpp
#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 вар
    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 вар
    for (int i = 0; i < v; i++) {
        for (int j = 0; j < v; j++) {
            cin >> matrix[i][j];
```

```cpp
        }
    }
    //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 шукаєм відносно
                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;
                }
```

```cpp

                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 << "}";
```

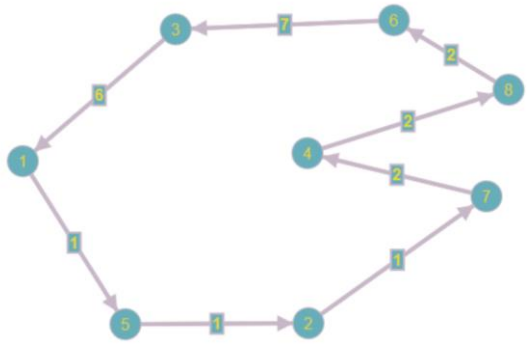
```
    //і виводимо ребра
    cout << endl << "E:{ ";

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

```
The number of vertices of the graph : 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
1 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
V: { 1, 2, 3, 4, 6, 5, 10, 11, 9, 7, 8, }
E:{ (1,2),(1,3),(1,4),(4,6),(2,5),(6,10),(10,11),(11,9),(2,7),(11,8),}
C:\Users\софія яцунда\source\repos\ConsoleApplication4\Debug\ConsoleApplication4.exe (процесс 128) завершает работу с ко
дом 0.
Чтобы закрыть это окно, нажмите любую клавишу…
```

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	∞	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	∞



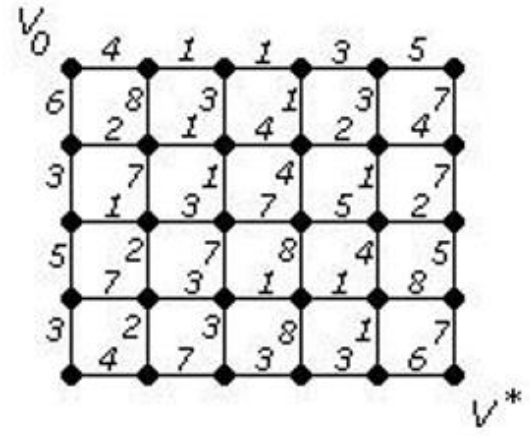
```cpp
#include <iostream>


using namespace std;

bool visited[8];


int graph[8][8] = {
        {0, 6, 6, 6, 1, 3, 1, 3},
        {6, 0, 5, 5, 1, 6, 1, 5},
        {6, 5, 0, 7, 7, 7, 7, 5},
        {6, 5, 7, 0, 6, 5, 1, 2},
        {1, 1, 7, 6, 0, 6, 6, 6},
        {3, 6, 7, 5, 6, 0, 1, 2},
        {1, 1, 7, 1, 6, 1, 0, 2},
        {3, 5, 5, 2, 6, 2, 2, 0}


};

void dfs(int index) {

    visited[index] = true;
    cout << index + 1;
```

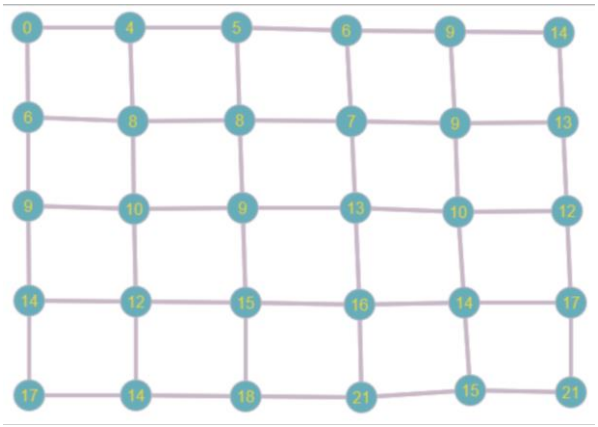
```cpp
    for (int j = 0; j < 9; j++) {
        if (!visited[j] && graph[index][j] == 1) {
            dfs(j);
        }
    }

}

int main() {
    for (bool& i : visited) {
        i = false;
    }
    dfs(0);

    return 0;
}
```

```
1527496
C:\Users\софія яцунда\source\repos\ConsoleApplication4\Debug\ConsoleApplication4.exe (процесс 10944) завершает работу с
кодом 0.
```

Завдання № 7

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



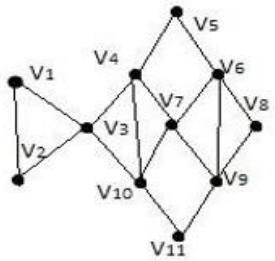


```cpp
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
using namespace std;
#define SIZE 30
int main()
{
    int a[SIZE][SIZE];
    int d[SIZE];
    int v[SIZE];
    int temp, minindex, min;
    int begin_index = 0;
    system("chcp 1251");
    system("cls");

    for (int i = 0; i < SIZE; i++)
    {
        a[i][i] = 0;
        for (int j = i + 1; j < SIZE; j++) {
            cout << "Enter the weight of edges " << i + 1 << " - " << j + 1<<" : ";
            cin >> temp;
            a[i][j] = temp;
            a[j][i] = temp;
        }
    }

    for (int i = 0; i < SIZE; i++)
    {
        d[i] = 10000;
        v[i] = 1;
    }
    d[begin_index] = 0;
    // Algorithm
    do {
        minindex = 10000;
        min = 10000;
        for (int i = 0; i < SIZE; i++){
            if ((v[i] == 1) && (d[i] < min)){
                min = d[i];
                minindex = i;
            }
        }

        if (minindex != 10000)
        {
            for (int i = 0; i < SIZE; i++)
            {
                if (a[minindex][i] > 0)
                {
                    temp = min + a[minindex][i];
                    if (temp < d[i])
                    {
                        d[i] = temp;
                    }
                }
            }
            v[minindex] = 0;
        }
    } while (minindex < 10000);

    cout << "The minimum way to the tops: ";
    for (int i = 0; i < SIZE; i++)
        cout<< d[i]<<"; ";


    int ver[SIZE];
    int end = 29;
    ver[0] = end + 1;
    int k = 1;
    int weight = d[end];

    while (end != begin_index)
    {
        for (int i = 0; i < SIZE; 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;
    cout<< "The minimum way: ";
    for (int i = k - 1; i >= 0; i--)
        cout<< ver[i]<<"-";
    getchar(); getchar();
    return 0;
}
```

```
Enter the weight of edges 1 - 2 : 4
Enter the weight of edges 1 - 3 : 0
Enter the weight of edges 1 - 4 : 0
Enter the weight of edges 1 - 5 : 0
Enter the weight of edges 1 - 6 : 0
Enter the weight of edges 1 - 7 : 6
Enter the weight of edges 1 - 8 : 0
Enter the weight of edges 1 - 9 : 0
Enter the weight of edges 1 - 10 : 0
Enter the weight of edges 1 - 11 : 0
Enter the weight of edges 1 - 12 : 0
Enter the weight of edges 1 - 13 : 0
Enter the weight of edges 1 - 14 : 0
Enter the weight of edges 1 - 15 : 0
Enter the weight of edges 1 - 16 : 0
Enter the weight of edges 1 - 17 : 0
Enter the weight of edges 1 - 18 : 0
Enter the weight of edges 1 - 19 : 0
Enter the weight of edges 1 - 20 : 0
Enter the weight of edges 1 - 21 : 0
Enter the weight of edges 1 - 22 : 0
Enter the weight of edges 1 - 23 : 0
Enter the weight of edges 1 - 24 : 0
Enter the weight of edges 1 - 25 : 0
Enter the weight of edges 1 - 26 : 0
Enter the weight of edges 1 - 27 : 0
Enter the weight of edges 1 - 28 : 0
Enter the weight of edges 1 - 29 : 0
Enter the weight of edges 1 - 30 : 0
Enter the weight of edges 2 - 3 : 1
Enter the weight of edges 23 - 28 : 0
Enter the weight of edges 23 - 29 : 1
Enter the weight of edges 23 - 30 : 0
Enter the weight of edges 24 - 25 : 0
Enter the weight of edges 24 - 26 : 0
Enter the weight of edges 24 - 27 : 0
Enter the weight of edges 24 - 28 : 0
Enter the weight of edges 24 - 29 : 0
Enter the weight of edges 24 - 30 : 7
Enter the weight of edges 25 - 26 : 4
Enter the weight of edges 25 - 27 : 0
Enter the weight of edges 25 - 28 : 0
Enter the weight of edges 25 - 29 : 0
Enter the weight of edges 25 - 30 : 0
Enter the weight of edges 26 - 27 : 7
Enter the weight of edges 26 - 28 : 0
Enter the weight of edges 26 - 29 : 0
Enter the weight of edges 26 - 30 : 0
Enter the weight of edges 27 - 28 : 3
Enter the weight of edges 27 - 29 : 0
Enter the weight of edges 27 - 30 : 0
Enter the weight of edges 28 - 29 : 3
Enter the weight of edges 28 - 30 : 0
Enter the weight of edges 29 - 30 : 6
The minimum way to the tops: 0; 4; 5; 6; 9; 14; 6; 8; 8; 7; 9; 13; 9; 10; 9; 11; 10; 12; 14; 12; 15; 15; 14; 17; 17; 14;
 18; 18; 15; 21;
The minimum way: 1-2-3-4-10-11-17-23-29-30-
```

Завдання № 8

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



Алгоритм Флері

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

Елементарні цикли

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

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

Завдання №9

Спростити формули (привести їх до скороченої ДНФ).

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

Оскільки жоден кон’юнкт не міститься в іншому, та будь-які два доданки відрізняються в, як мінімум, двох позиція, то можна стверджувати, що дана функція є ДНФ і спростити її неможливо.