

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

Кафедра систем штучного інтелекту

Лабораторна робота №4
із дисципліни
«Дискретна математика»

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

Викладач:
Мельникова Н.І.

Львів – 2019 р.

Тема роботи: Основні операції над графами. Знаходження остова мінімальної ваги за алгоритмом Прима-Краскала

Мета роботи: Набуття практичних вмінь та навичок з використання алгоритмів Прима і Краскала.

Варіант – 10

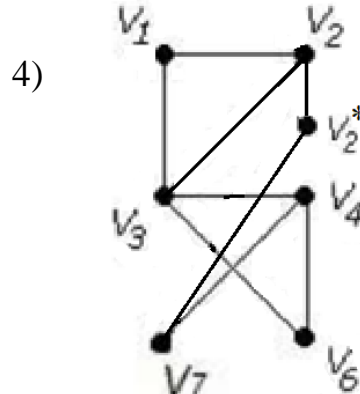
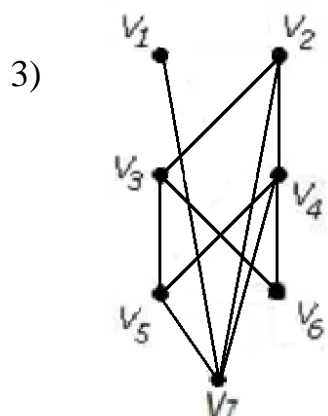
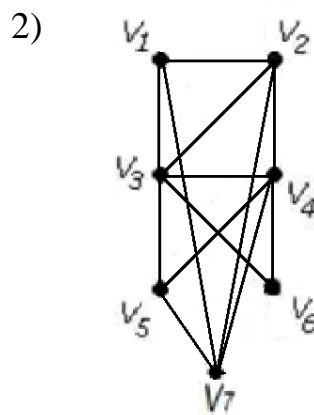
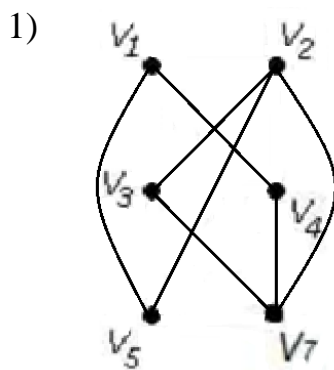
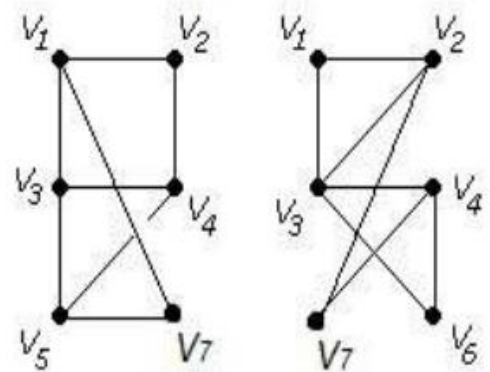
Завдання 1

Розв'язати на графах наступні задачі:

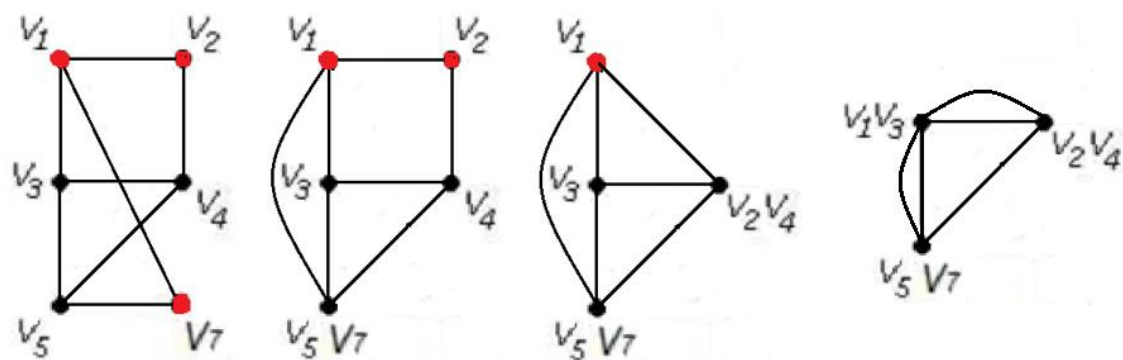
1. Виконати наступні операції над графами:

- 1) знайти доповнення до першого графу.
- 2) об'єднання графів.
- 3) кільцеву суму G_1 та G_2 ($G_1 + G_2$)
- 4) розщепити вершину у другому графі.
- 5) виділити підграф A , що складається з 3-х вершин в G_1 і знайти стягнення A в G_1 ($G_1 \setminus A$).
- 6) добуток графів.

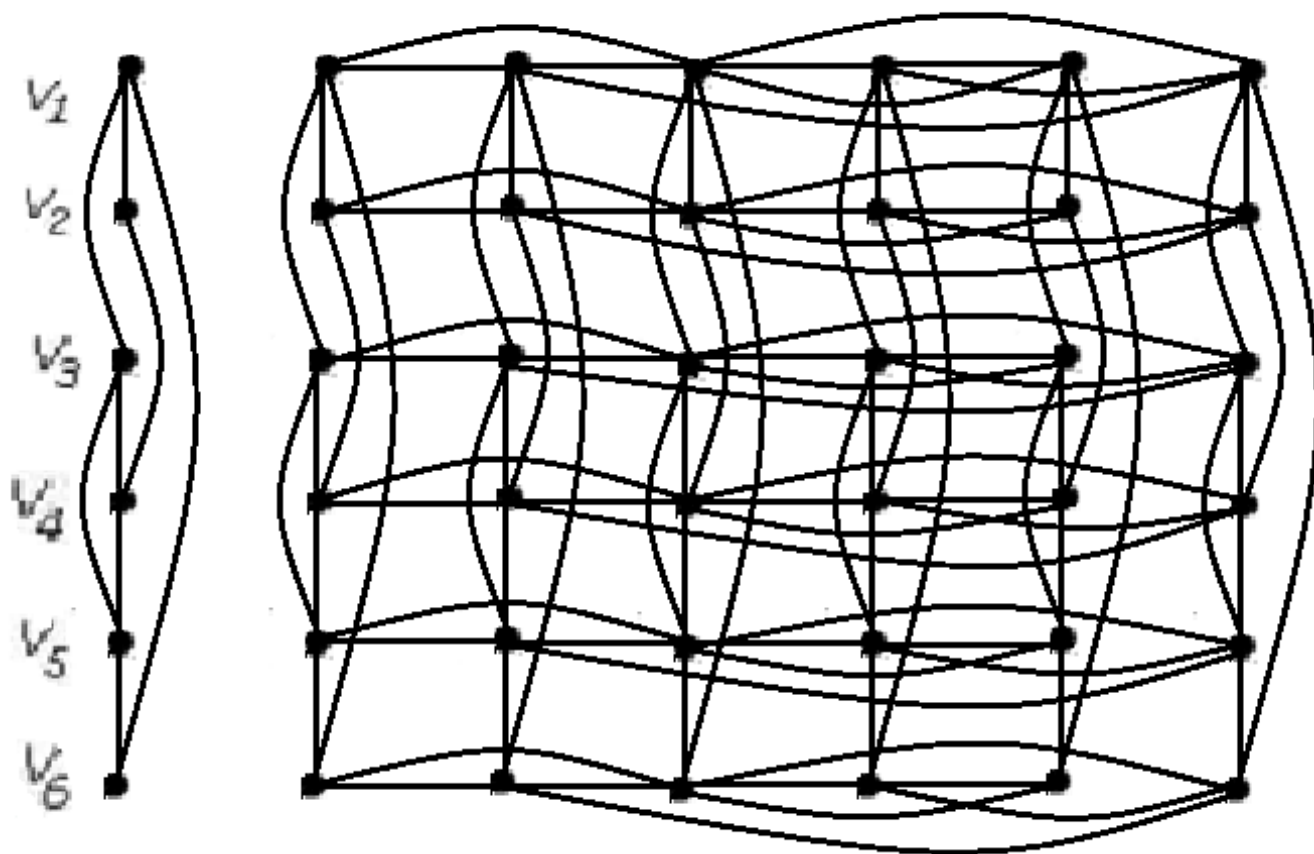
10



5)



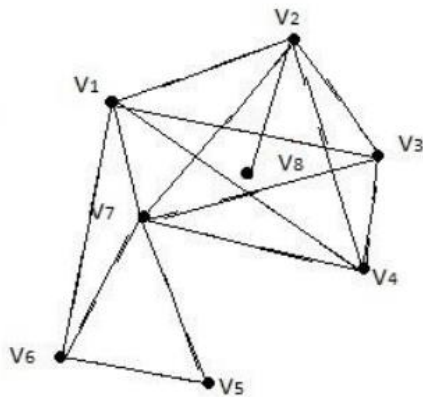
6)



Завдання 2

Знайти таблицю суміжності та діаметр графа.

10)



	V ₁	V ₂	V ₃	V ₄	V ₅	V ₆	V ₇	V ₈
V ₁	0	1	1	1	0	1	1	0
V ₂	1	0	1	1	0	0	1	1
V ₃	1	1	0	1	0	0	1	0
V ₄	1	1	1	0	0	0	1	0
V ₅	0	0	0	0	0	1	1	0
V ₆	1	0	0	0	1	0	1	0
V ₇	1	1	1	1	1	1	0	0
V ₈	0	1	0	0	0	0	0	0

	V ₁	V ₂	V ₃	V ₄	V ₅	V ₆	V ₇	V ₈
V ₁	-	1	1	1	2	1	1	2
V ₂	1	-	1	1	2	2	1	1
V ₃	1	1	-	1	2	2	1	2
V ₄	1	1	1	-	2	2	1	2
V ₅	2	2	2	2	-	1	1	3
V ₆	1	2	2	2	1	-	1	3
V ₇	1	1	1	1	1	1	-	2
V ₈	2	1	2	2	3	3	2	-

Діаметр -- 3

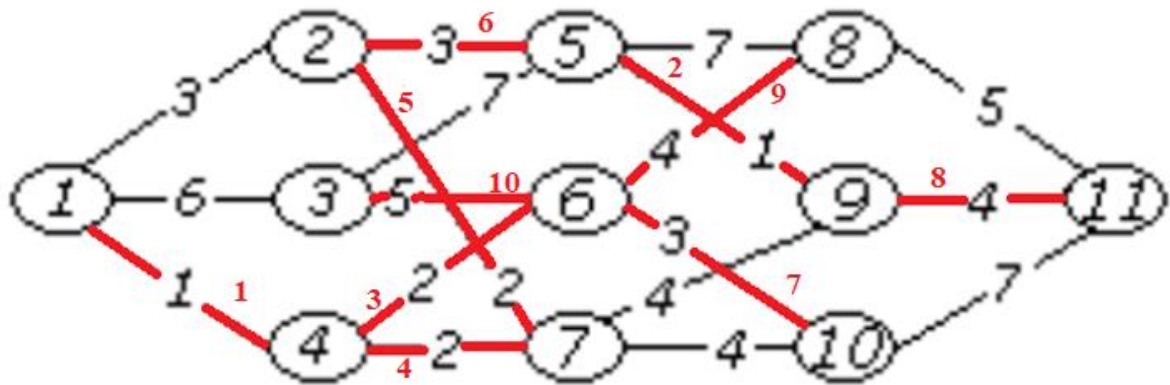
Завдання 3

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

10



Алгоритм Караскала:

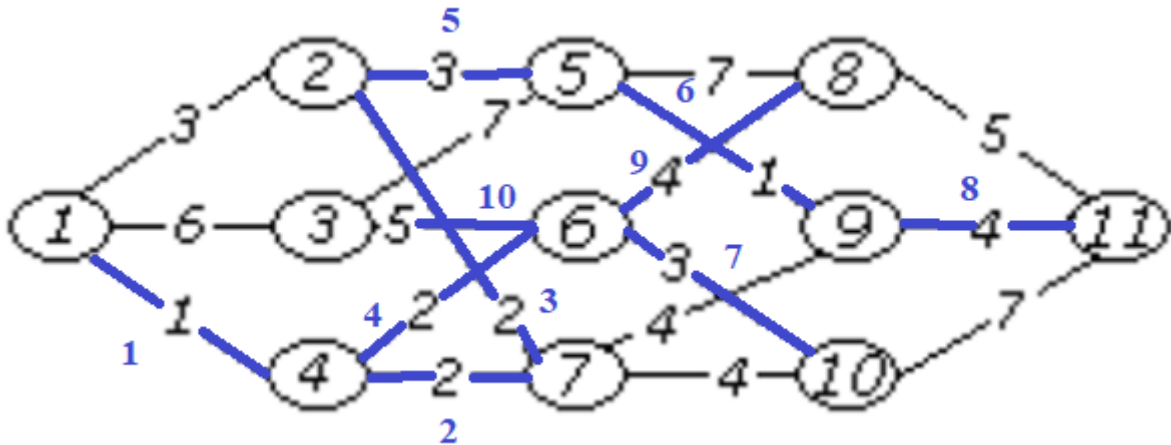


$V = \{1, 4, 5, 9, 6, 7, 2, 10, 11, 8, 3\}$

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

Довжина: 27

Алгоритм прима:



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

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

Довжина: 27

Додаток 2

Написати програму, яка реалізує алгоритм знаходження остового дерева мінімальної ваги згідно свого варіанту.

Варіант № 10

За алгоритмом Краскала знайти мінімальне остове дерево графа. Етапи розв'язання задачі виводити на екран. Протестувати розроблену програму на наступному графі:



Програмна реалізація:

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

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

Graf graf[30];
int *versh = new int[vershyny];
void CreateGraf() {
    cout << "Введіть кількість вершин: ";
    cin >> vershyny;
    cout << "Введіть кількість ребер: ";
    cin >> rebra;
    int v = 0;
    for (int i = 0; i < rebra; i++)
    {
        cout << "Введіть вершину 1: ";
        cin >> graf[i].v1;
```

```

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

        cout << "Введіть вагу: ";
        cin >> graf[i].weight;
    }
}

void Sort() {
    for (int i = 0; i < rebra;i++) {
        for (int j = 0; j < rebra;j++)
        {
            Graf test;
            if (i == j)
            {
                continue;
            }
            else if (graf[i].weight < graf[j].weight)
            {
                test = graf[i];
                graf[i] = graf[j];
                graf[j] = test;
            }
        }
    }
}

void FillArray(int** arr, int ROWS, int COLLS) //функція заповнення масиву
{
    for (int i = 0;i < ROWS;i++)
    {
        for (int j = 0; j < COLLS;j++)
            arr[i][j] = 0;
    }
}

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

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

    CreateGraf();
    Sort();

    int ROWS = 5;
    int COLLS = 11;
    int** points = new int* [ROWS];    //створюємо динамічний двовірний масив
    for (int i = 0;i < ROWS; i++)
    {
        points[i] = new int[COLLS];
    }
    FillArray(points, 5, 11);
    PrintArray(points, 5, 11);
    int weight = graf[0].weight;
    points[0][0] = graf[0].v1;
    points[0][1] = graf[0].v2;
    cout << endl;
    PrintArray(points, 5, 11);
}

```

```

cout << endl << endl;
string v1 = std::to_string(graf[0].v1);
string v2 = std::to_string(graf[0].v2);
string result= v1 + "-" + v2 + "; ";

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

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

    while (points[i][j] != 0)
    {
        if (points[i][j] == graf[gr].v1)
        {
            v1exist = true;
            break;
        }
        else if (points[i][j + 1] == 0 || j + 1 == vershyny)
        {
            if (points[i][j] != graf[gr].v1)
            {
                v1exist = false;
                break;
            }
        }
        else
        {
            j++;
        }
    }
    int k = 0;
    while (points[i][k] != 0)
    {
        if (points[i][k] == graf[gr].v2)
        {
            v2exist = true;
            k++;
            break;
        }
        else if (points[i][k + 1] == 0 || k + 1 == vershyny)
        {
            if (points[i][k] != graf[gr].v2)
            {
                v2exist = false;
                k++;
                break;
            }
        }
        else
        {
            k++;
        }
    }
    if (v1exist == true && v2exist == true)
    {
        continue;
    }
}

```



```

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

                        while (points[l][g] != 0)
                        {
                            switch (i)
                            {
                                case 0: points[i][size1] =
                                case 1: points[i][size2] =
                                case 2: points[i][size3] =
                                case 3: points[i][size4] =
                                case 4: points[i][size5] =

                            }
                            points[l][g] = 0;

                            g++;
                        }
                    }
                    switch (l)
                    {
                        case 0: size1=0; break;
                        case 1: size2=0; break;
                        case 2: size3=0; break;
                        case 3: size4=0; break;
                        case 4: size5=0; break;
                    }
                    i = 0;
                    l = 6;
                    break;
                }
            }
            else
            {
                g++;
            }
        }
    }
}

```

points[l][g]; size1++; break;
 points[l][g]; size2++; break;
 points[l][g]; size3++; break;
 points[l][g]; size4++; break;
 points[l][g]; size5++; break;

```

    }
    }
}

else if (v2exist == true && v1exist == false)
{
    for (int l = 1; l <= 5; l++)
    {
        int g = 0;
        if (points[l][g] == 0)
        {
            switch (i)
            {
                case 0: points[i][size1] = graf[gr].v1; size1++; break;
                case 1: points[i][size2] = graf[gr].v1; size2++; break;
                case 2: points[i][size3] = graf[gr].v1; size3++; break;
                case 3: points[i][size4] = graf[gr].v1; size4++; break;
                case 4: points[i][size5] = graf[gr].v1; size5++; break;
            }
            i = 0;
            break;
        }
        while (points[l][g] != 0)
        {
            if (points[l][g] == graf[gr].v1)
            {
                g = 0;
                while (points[l][g] != 0)
                {
                    switch (i)
                    {
                        case 0: points[i][size1] =
                                points[l][g]; size1++; break;
                        case 1: points[i][size2] =
                                points[l][g]; size2++; break;
                        case 2: points[i][size3] =
                                points[l][g]; size3++; break;
                        case 3: points[i][size4] =
                                points[l][g]; size4++; break;
                        case 4: points[i][size5] =
                                points[l][g]; size5++; break;
                    }
                    points[l][g] = 0;
                    g++;
                }
                switch (l)
                {
                    case 0: size1=0; break;
                    case 1: size2=0; break;
                    case 2: size3=0; break;
                    case 3: size4=0; break;
                    case 4: size5=0; break;
                }
                i = 0;
                l = 6;
                break;
            }
            else
            {
                g++;
            }
        }
    }
}

```

```

    }

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

    cout << graf[gr].v1 << "-" << graf[gr].v2 << endl;
    PrintArray(points, 5, 11);

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

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

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

}

```

Результати виконання програми:

```

Введіть кількість вершин: 11
Введіть кількість ребер: 18
Введіть вершину 1: 4
Введіть вершину 2: 1
Введіть вагу: 7
Введіть вершину 1: 4
Введіть вершину 2: 7
Введіть вагу: 2
Введіть вершину 1: 7
Введіть вершину 2: 10
Введіть вагу: 3
Введіть вершину 1: 10
Введіть вершину 2: 11
Введіть вагу: 5
Введіть вершину 1: 11
Введіть вершину 2: 8
Введіть вагу: 4
Введіть вершину 1: 8
Введіть вершину 2: 5
Введіть вагу: 4
Введіть вершину 1: 5
Введіть вершину 2: 2
Введіть вагу: 2
Введіть вершину 1: 2
Введіть вершину 2: 1
Введіть вагу: 4
Введіть вершину 1: 1
Введіть вершину 2: 3
Введіть вагу: 3
Введіть вершину 1: 3
Введіть вершину 2: 6
Введіть вагу: 7
Введіть вершину 1: 9
Введіть вершину 2: 11
Введіть вагу: 6
Введіть вершину 1: 3
Введіть вершину 2: 5
Введіть вагу: 1
Введіть вершину 1: 6
Введіть вершину 2: 8
Введіть вагу: 4
Введіть вершину 1: 4
Введіть вершину 2: 6
Введіть вагу: 2
Введіть вершину 1: 7
Введіть вершину 2: 9
Введіть вагу: 3
Введіть вершину 1: 2
Введіть вершину 2: 7
Введіть вагу: 1
Введіть вершину 1: 5
Введіть вершину 2: 9
Введіть вагу: 7
Введіть вершину 1: 6
Введіть вершину 2: 10
Введіть вагу: 5
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
3 5 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
2-7
3 5 0 0 0 0 0 0 0 0 0
2 7 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
4-7
3 5 0 0 0 0 0 0 0 0 0
2 7 4 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
4-6
3 5 0 0 0 0 0 0 0 0 0
2 7 4 6 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
5-2
3 5 2 7 4 6 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
1-3
3 5 2 7 4 6 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
7-9
3 5 2 7 4 6 1 9 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
7-10
3 5 2 7 4 6 1 9 10 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
6-8
3 5 2 7 4 6 1 9 10 8 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0

```

```
11-8
3 5 2 7 4 6 1 9 10 8 11
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0

+++++

Мінімальне остове дерево графа складається з ребер: 3-5; 2-7; 4-7; 4-6; 5-2; 1-3; 7-9; 7-10; 6-8; 11-8;
Вага мінімального остового дерева графа = 25
```

Висновок: На цій лабораторній роботі я набув практичних вмінь та навичок при побудові матриць бінарних відношень та визначені їх типів.