МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра систем штучного інтелекту

Лабораторна робота №4

З дисципліни

«Дискретна математика»

Виконав :

Студент КН-113

Сайкевич В.А.

Викладач:

Мельникова Н.І.

Львів – 2019

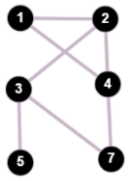
Тема : «Основні операції над графами. Знаходження остова мінімальної ваги за алгоритмом Прима-Краскала»

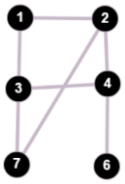
Мета : Набуття практичних вмінь та навичок з використання алгоритмів Прима і Краскала.

Варіант №14

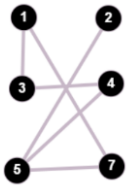
Завдання 1

1. Виконати наступні операції над графами:

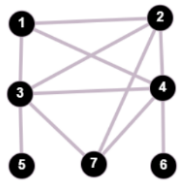




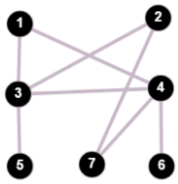
1) знайти доповнення до першого графу



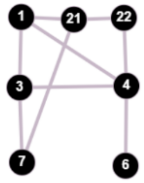
2) об’єднання графів



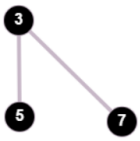
3) кільцеву суму G1 та G2 (G1+G2)

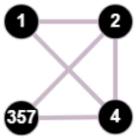


4) розщепити вершину у другому графі

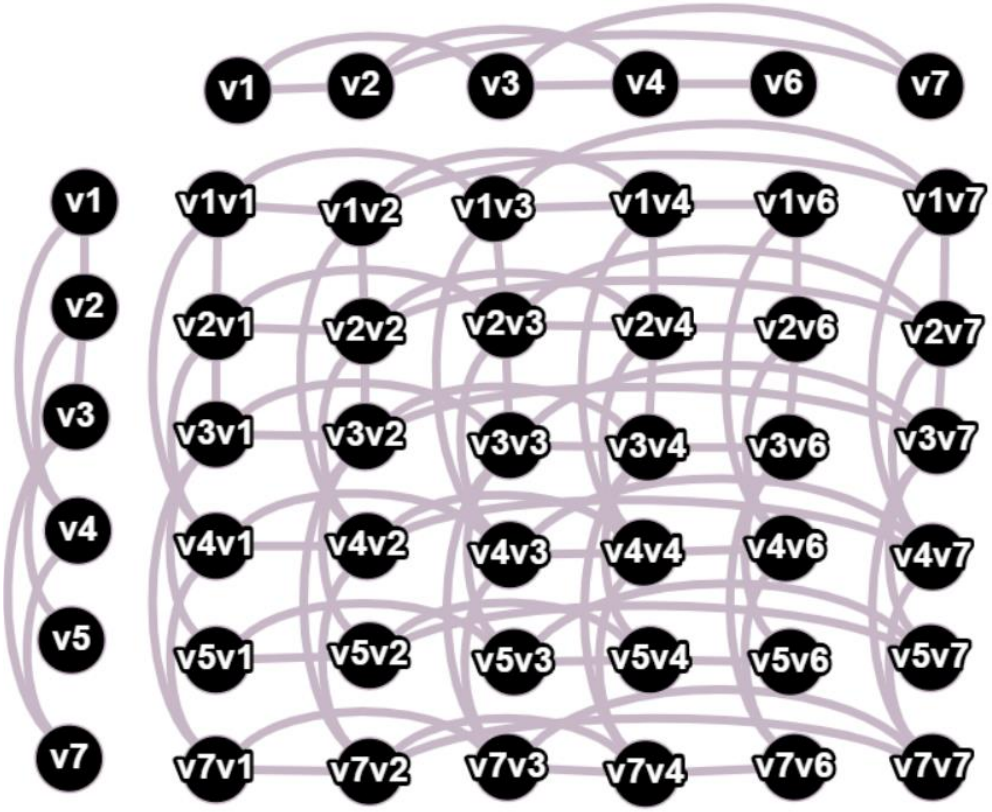


5) виділити підграф A, що складається з 3-х вершин в G1 і знайти стягнення A в G1 (G1\ A)

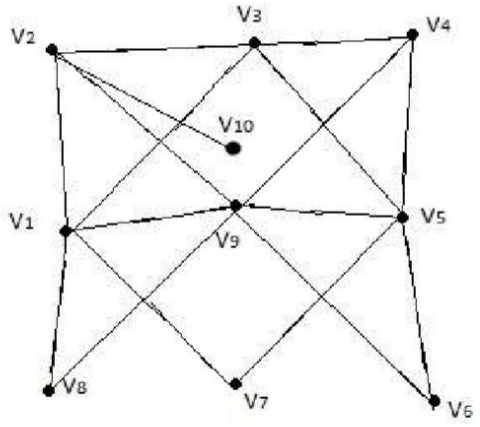
A) 

G1\A) 

6) добуток графів



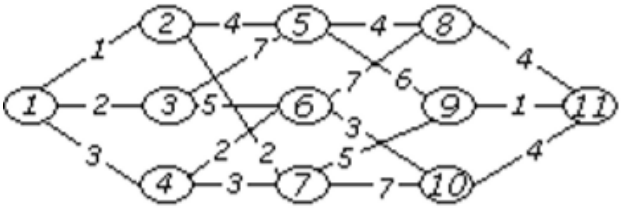
2. Знайти таблицю суміжності та діаметр графа



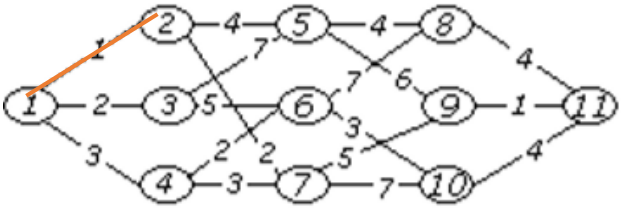
	1	2	3	4	5	6	7	8	9	10
1	_	1	1	0	0	0	1	1	1	0
2	1	_	1	0	0	0	0	0	1	1
3	1	1	_	1	1	0	0	0	0	0
4	0	0	1	_	1	0	0	0	1	0
5	0	0	1	1	_	1	1	0	1	0
6	0	0	0	0	1	_	0	0	1	0
7	1	0	0	0	1	0	_	0	0	0
8	1	0	0	0	0	0	0	_	1	0
9	1	1	0	1	1	1	0	1	_	0
10	0	1	0	0	0	0	0	0	0	_

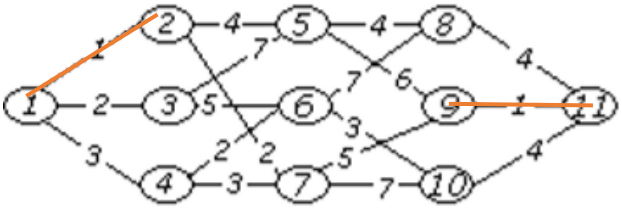
Діаметр графа=3

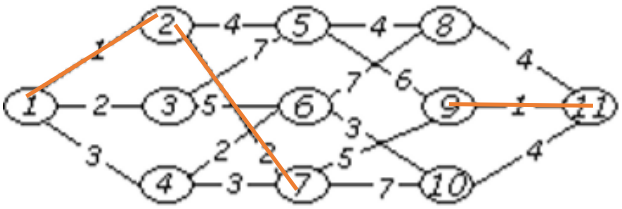
3. Знайти двома методами (Краскала і Прима) мінімальне остове дерево графа.

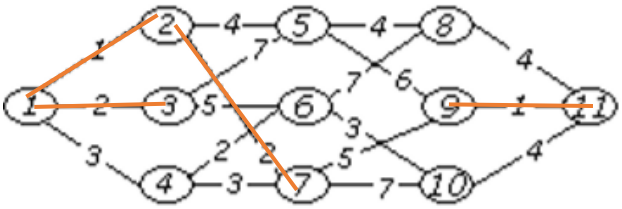


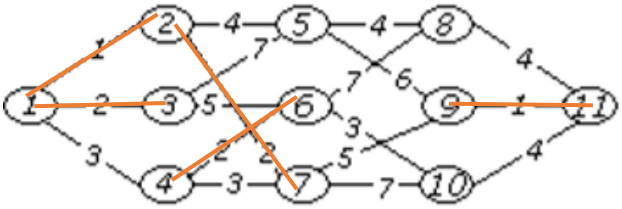
Краскала:

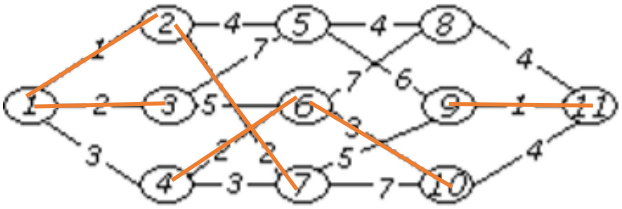


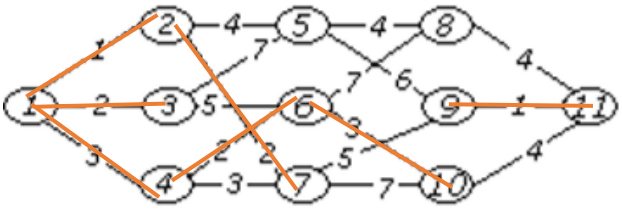


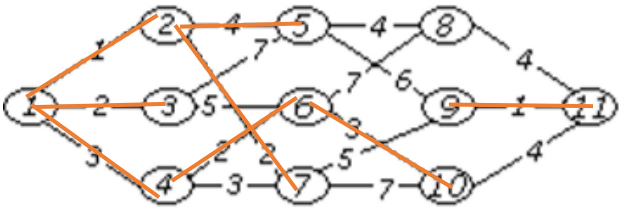
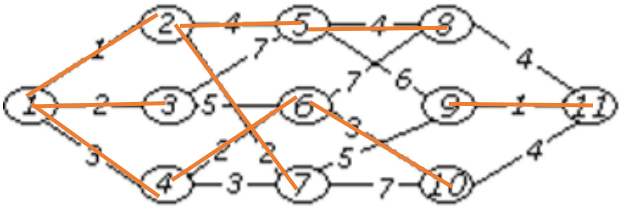
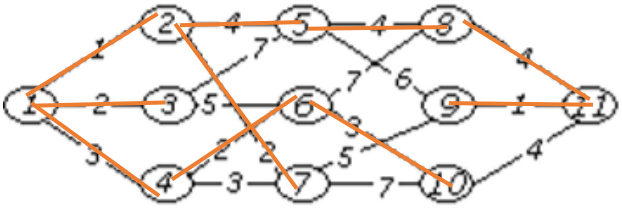




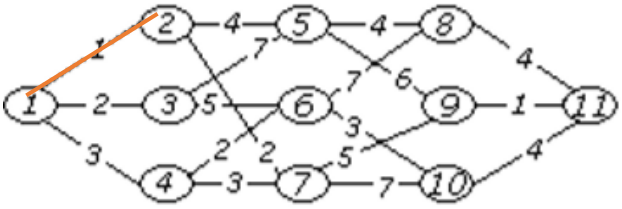
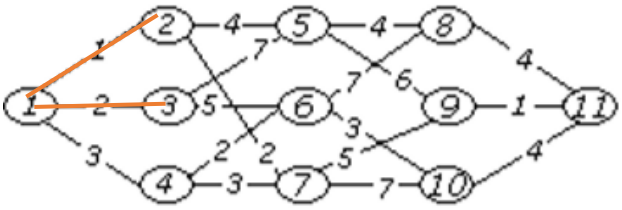
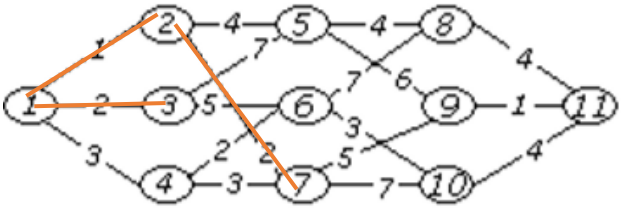
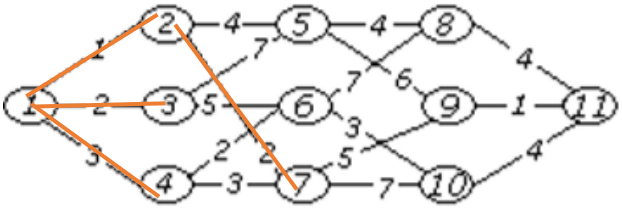
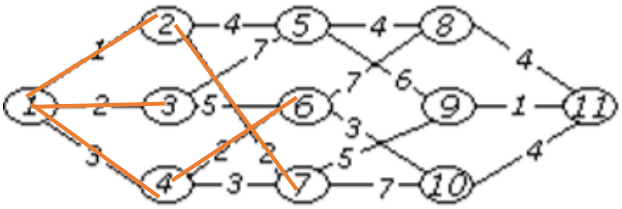


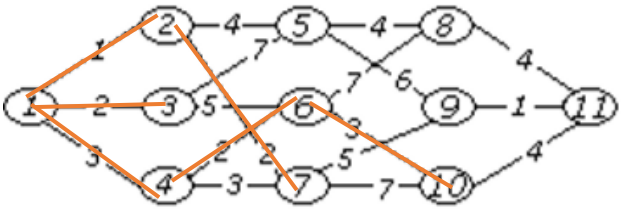
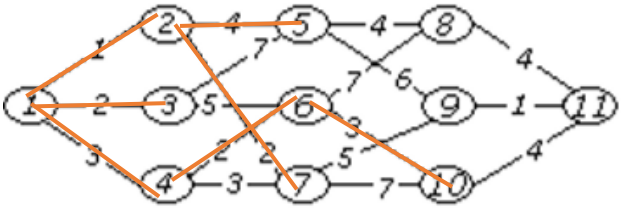
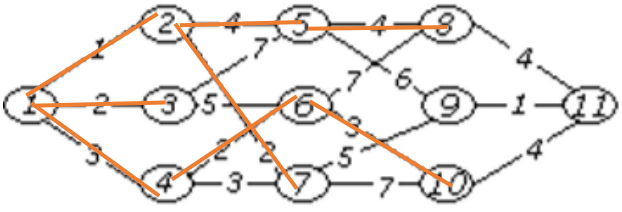
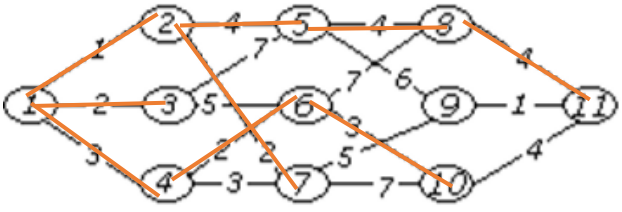
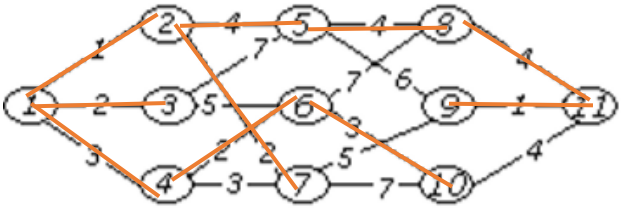




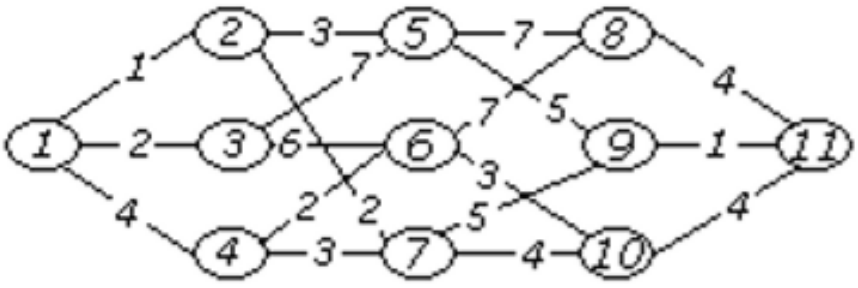
Прима:

Завдання 2

Завдання : Написати програму, яка реалізує алгоритм знаходження остового дерева мінімальної ваги за алгоритмом Краскала. Етапи розв'язання задачі виводити на екран. Протестувати розроблену програму на наступному графі:



Код

1) Прима

```cpp
#include <iostream>
#include <math.h>
#include <iomanip>

using namespace std;

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

void main()
{
    int size;
    cout << "Enter amount of vertices ";
    cin >> size;
    int** adjrcencyarr = new int* [size];
    bool* vertex = new bool[size];
    cout << "Enter adjacency matrix \n";
    cout << "  ";
```

```cpp
    for (int i = 0; i < size; i++) { cout << setw(3) << i + 1; adjrcencyarr[i] = new int[size]; vertex[i] = 0; }
    cout << endl;
    for (int i = 0; i < size; i++) {
        cout << setw(3) << i + 1;
        for (int j = 0; j < i; j++) { cout << setw(3) << adjrcencyarr[j][i]; }
        cout << "  -";
        for (int j = i + 1; j < size; j++) { cin >> adjrcencyarr[i][j]; }
    }
    cout << endl;
    vertex[0] = 1;
    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 (adjrcencyarr[i][j] < min&& adjrcencyarr[i][j]!=0 && (vertex[i] && !vertex[j])) {
                    min = adjrcencyarr[i][j];
                    minI = i, minJ = j;
                }
            }
        }
        vertex[minI] = 1, vertex[minJ] = 1;
        cout << n + 1 << ")conected " << minI+1 << "-" << minJ+1 << endl;
    }
}
```

Результати :

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

2) Краскала

```cpp
#include <iostream>
#include <iomanip>

 using namespace std;

int find(int i,int* vertex)
{
    while (vertex[i] != i)
        i = vertex[i];
    return i;
}

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];
        }
    }
}

void main()
{
    int size;
```

```cpp
    cout << "Enter amount of vertices ";
    cin >> size;
    int** adjacencyarr = new int* [size];
    int* vertex = new int[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] = i; }
    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 && (find(i, vertex) != find(j, vertex))) {
                    min = adjacencyarr[i][j];
                    minI = i, minJ = j;
                }
            }
        }
        vertex[find(minI, vertex)] = find(minJ, vertex);
        cout << n + 1 << ")conected " << minI+1 << "-" << minJ+1 << endl;
    }
}
```

Результати :

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

Висновок:

Ми набули практичних вмінь та навичок з використання алгоритмів Прима і Краскала.