МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА“

Кафедра систем штучного інтелекту

Лабораторна робота №4

З дисципліни

«Дискретна математика»

Виконала:

Студентка групи КН-115

Рокицька Анастасія

Викладач:

Мельникова Н.І.

Львів – 2019р.

Тема: Основні операції над графами. Знаходження остова мінімальної ваги за алгоритмом Пріма-Краскала.
Мета роботи: набуття практичних вмінь та навичок з використання алгоритмів Пріма і Краскала.

Теоретичні відомості:

Теорія графів дає простий, доступний і потужний інструмент побудови моделей прикладних задач, є ефективним засобом формалізації сучасних інженерних і наукових задач у різних областях знань.

Графом G називається пара множин (V, E), де V – множина вершин, перенумерованих числами 1, 2, ..., $n = \upsilon$; $V = \{\upsilon\}$, E – множина упорядкованих або неупорядкованих пар $e = (v', v'')$, $v' \in V$, $v'' \in V$, називаних дугами або ребрами, $E = \{e\}$. При цьому не має примусового значення, як вершини розташовані в просторі або площині і які конфігурації мають ребра.

Неорієнтованим графом G називається граф у якого ребра не мають напрямку. Такі ребра описуються неупорядкованою парою (v', v''). *Орієнтований граф (орграф)* – це граф ребра якого мають напрямок та можуть бути описані упорядкованою парою (v', v''). Упорядковане ребро називають *дугою*. Граф є *змішаним*, якщо наряду з орієнтованімі ребрами (дугами) є також і неорієнтовані. При розв'язку задач змішаний граф зводиться до орграфа.

Кратними (паралельними) називаються ребра, які зв'язують одні і ті ж вершини. Якщо ребро виходить та й входить у дну і ту саму вершину, то таке ребро називається *петлею*.

Мультиграф – граф, який має кратні ребра. *Псевдограф* – граф, який має петлі. *Простий граф* – граф, який не має кратних ребер та петель.

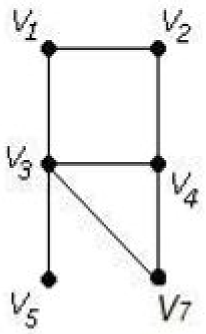
Варіант 13

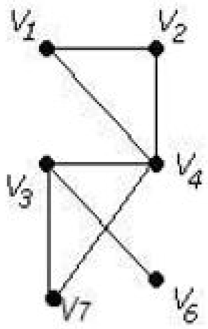
Додаток 1:

Завдання № 1. Розв'язати на графах наступні задачі:

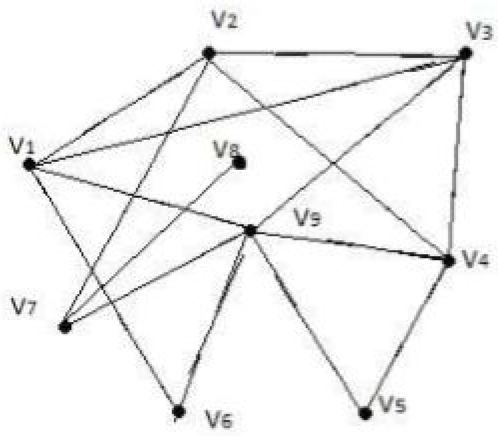
1. Виконати наступні операції над графами:

1) знайти доповнення до першого графу,
2) об'єднання графів,
3) кільцеву суму 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), 6) добуток графів.

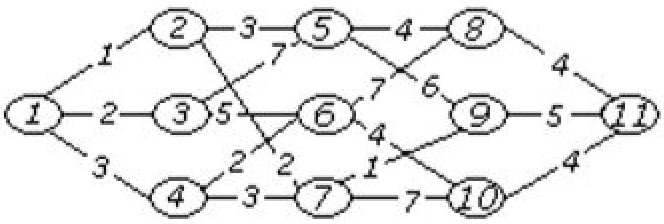




2. Знайти таблицю суміжності та діаметр графа.



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

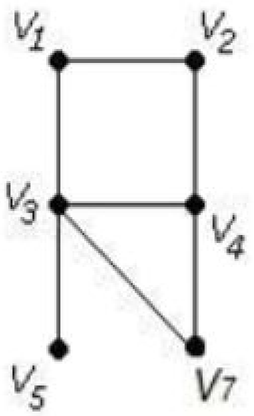


Розв'язання:

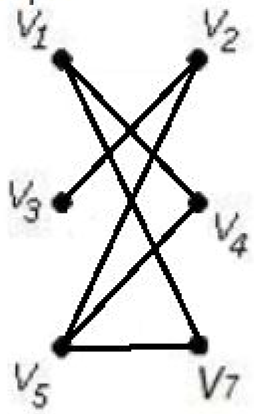
1.

1.

Граф:

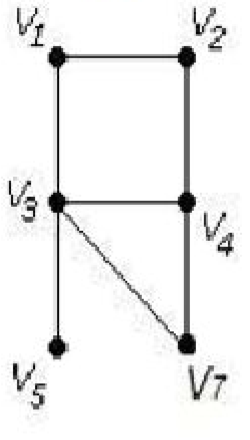


Операція доповнення до графа:

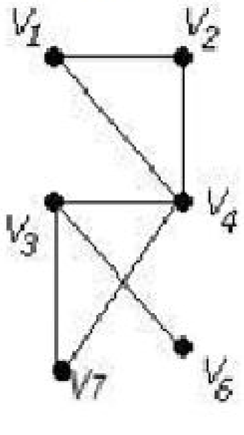


2.

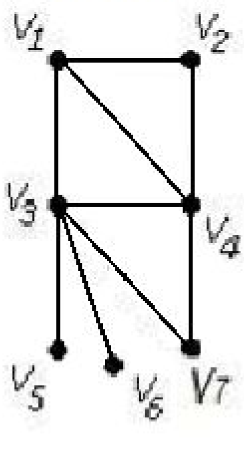
G1



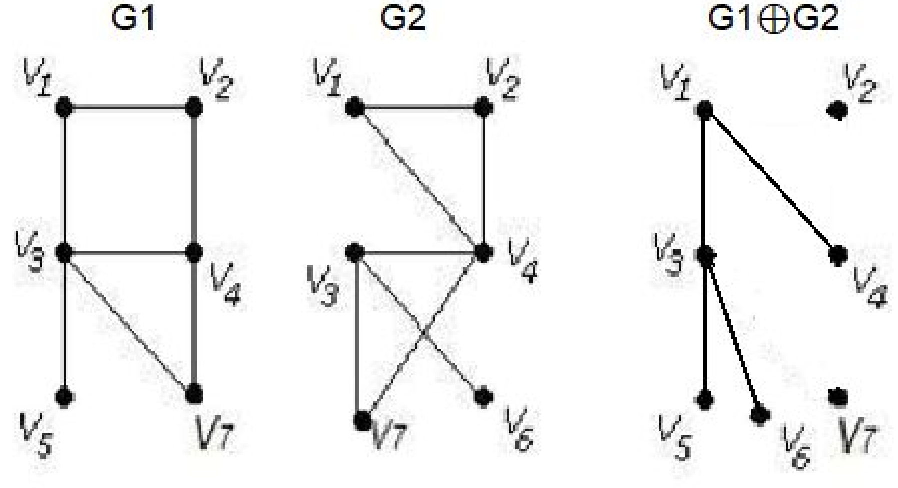
G2



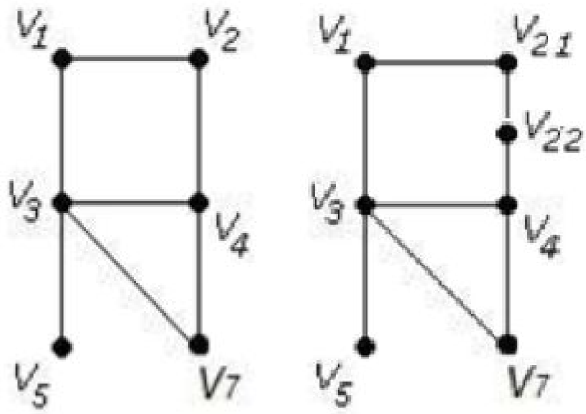
G1∪G2



3.



4. Розщеплення вершини V2

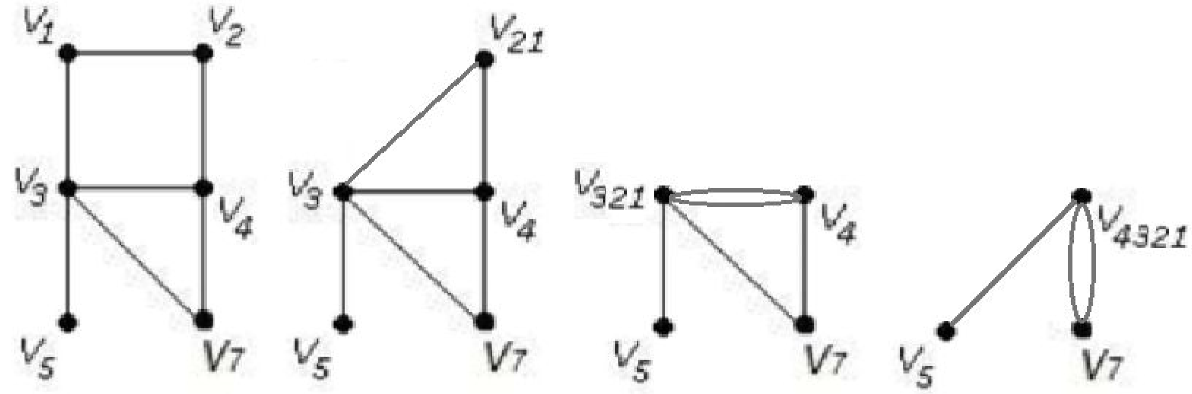


5.

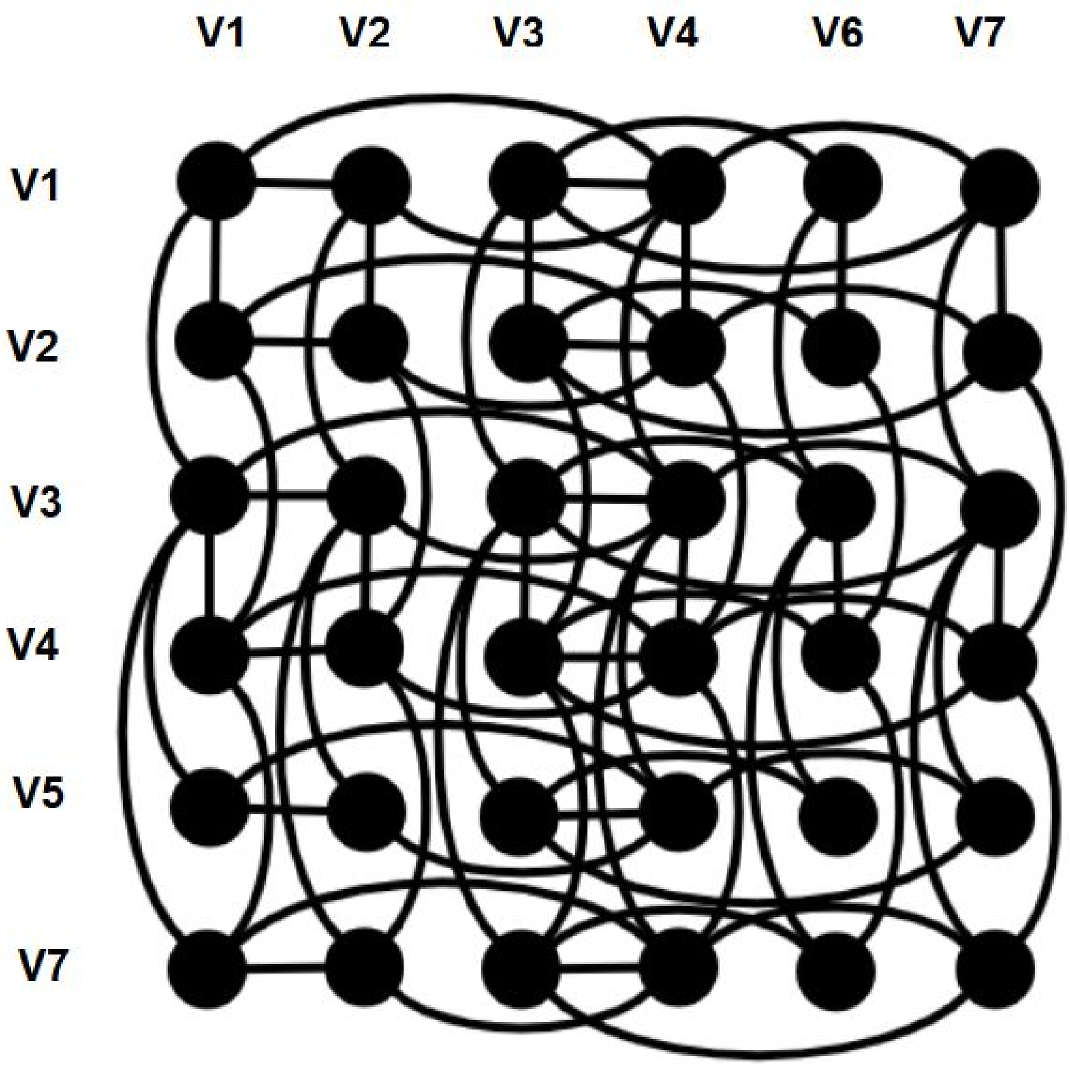
G1: V1={V1,V2,V3,V4,V5,V7}

A : V2={V1,V2,V3}

G1\A={V4321,V5,V7}



6. G1 × G2



2. **Матриця суміжності**

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

D=3 (V4->V2->V7->V8)

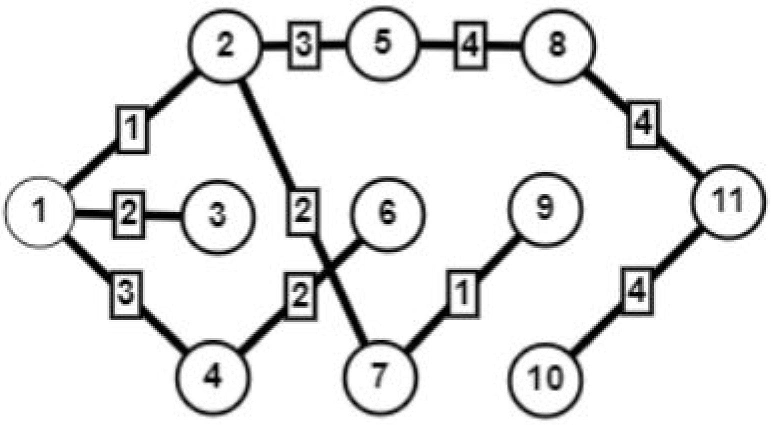
3.

Мінімальне остове дерево:

1. **за методом Прима**

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

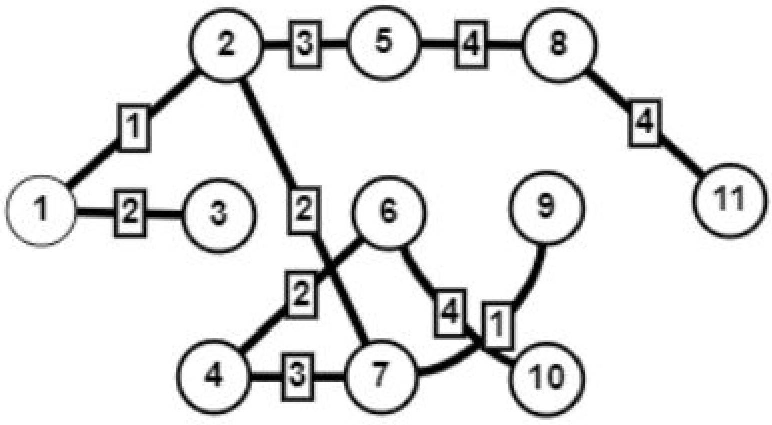
E={(1,2),(1,3),(2,7),(7,9),(1,4),(4,6),(2,5),(5,8),(8,11),(11,10)}



2. **за методом Краскала**

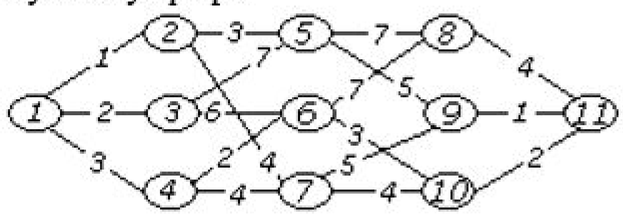
V={1,2,7,9,3,4,6,5,10,8,11}

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



Завдання №2

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



Код програми:

```
#include<iostream>

using namespace std;



void main()
{

	setlocale(LC_ALL , "ukr");

	int a, b, i, j, min, ne = 1, mincost = 0;
```

```
	int visited[15] = { 0 };//компютерне подання множини вершин, 
перевіряємо чи вершина графа вже буладодана, щоб не було циклу
	int path[15] = { 0 }; //шлях
	int num = 0;




	int n = 11;
	int arr[12][12] = { {0,0,0,0,0,0,0,0,0,0,0,0},//матриця суміжності
                            {0,0,1,2,3,0,0,0,0,0,0,0},
                           {0,1,0,0,0,3,0,4,0,0,0,0},
                          {0,2,0,0,0,7,6,0,0,0,0,0},
                          {0,3,0,0,0,0,2,4,0,0,0,0},
                          {0,0,3,7,0,0,0,0,7,5,0,0},
                     {0,0,0,6,2,0,0,0,7,0,3,0},
                     {0,0,4,0,4,0,0,0,0,5,4,0},
                   {0,0,0,0,0,7,7,0,0,0,0,4},
                   {0,0,0,0,0,5,0,5,0,0,0,1},
                     {0,0,0,0,0,0,3,4,0,0,0,2},
                     {0,0,0,0,0,0,0,0,4,1,2,0}
                    };



	int arr2[12][12];
	for (i = 1; i < 12; i++)//та сама матриця суміжності але замість 0 999
		for (j = 0; j < 12; j++) {
			arr2[i][j] = arr[i][j];
			if (arr2[i][j] == 0)
				arr2[i][j] = 999;

		}

	visited[1] = 1;//починаємо з 1 вершини
	cout << "\n\tПошук мінімального остового дерева за методом Прима: 
";
	while (ne < n)
	{
		for (i = 1, min = 999; i <= n; i++) {
```

```
                for (j = 1; j <= n; j++) {
                    if (arr2[i][j] < min) {
                        if (visited[i] != 0)
                        {
                            min = arr2[i][j];
                            a = i;
                            b = j;
                        }
                    }
                }
            }
            if (visited[a] == 0 || visited[b] == 0)
            {
                path[num] = b;

                num++;
                ne++;
                mincost += min;
                visited[b] = 1;
                cout << "\n\t( " << a << ",  " << b << ")";
            }
            arr2[a][b] = arr2[b][a] = 999;

        }


        cout << "\n\n";
        cout << "\tV={";
        cout << 1 << ", ";
        for (int i = 0; i < n - 1; i++)
        {
            cout << path[i];
            if (i < n - 2) cout << ", ";
        }
        cout << "}";
        cout << "\n\tНайкоротший шлях = " << mincost;
        cout << endl << endl;


    }
```

Результат програми:

```
Пошук мінімального остового дерева за методом Прима:
( 1,  2)
( 1,  3)
( 1,  4)
( 4,  6)
( 2,  5)
( 6,  10)
( 10,  11)
( 11,  9)
( 2,  7)
( 11,  8)

V={1, 2, 3, 4, 6, 5, 10, 11, 9, 7, 8}
Найкоротший шлях = 25
```