**Міністерство освіти і науки України**

**Національному університеті "Львівська**

**Політехніка"**

**Кафедра систем штучного інтелекту**

**Лабораторна робота № 4**

з дисципліни



**Виконав:**

студентка групи КН-114

Церковник Оксана **Викладач:**

Мельникова Н.І.

Львів - 2019р.

**Лабораторна робота № 4.**

**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:**

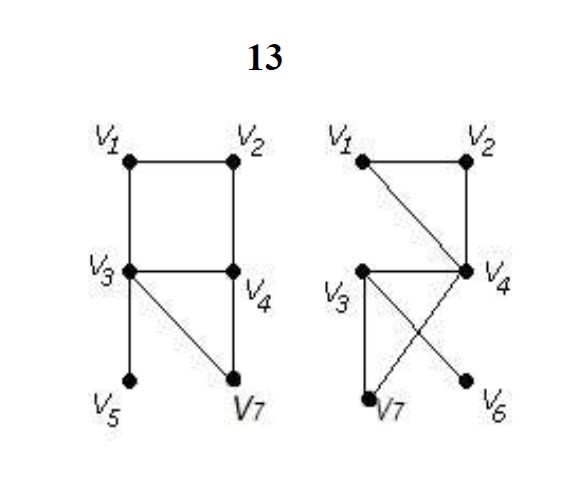
Теорія графів дає простий, доступний і потужний інструмент побудови моделей прикладних задач, є ефективним засобом формалізації сучасних інженерних і наукових задач у різних областях знань.

*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 орієнтованими 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*.

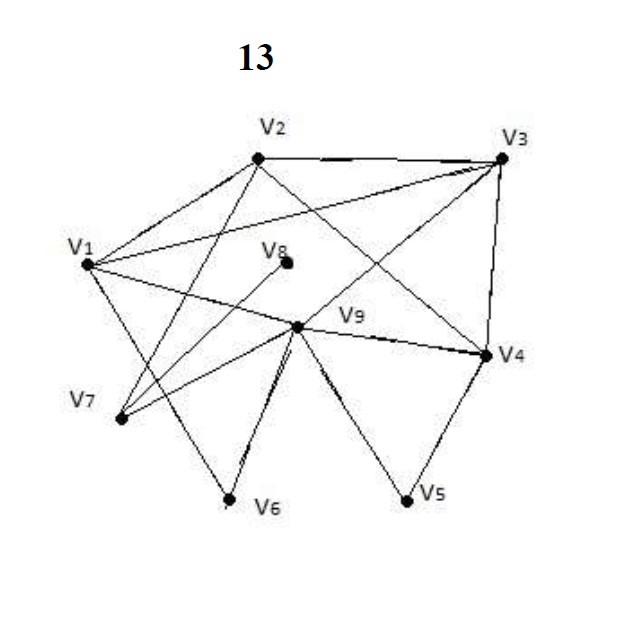
*Мультиграф* – граф, який має кратні ребра. *Псевдограф* – граф, який має петлі. *Простий граф* – граф, який не має кратних ребер та петель.

**Завдання**

**A screenshot of a cell phone

Description automatically generated**

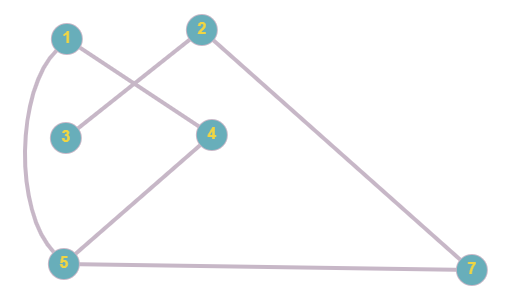
A close up of a map

Description automatically generated

**Розв'язок**

1.

1) доповнення до першого



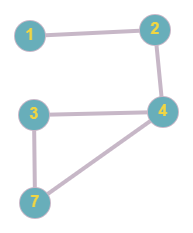
2) об’єднання 2 графів

A picture containing skiing, accessory

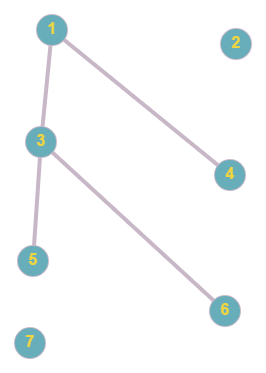
Description automatically generated

3) кільцева сума

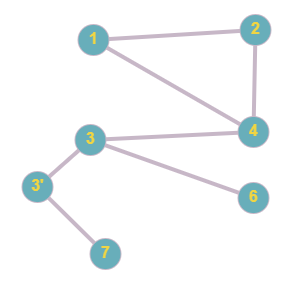
∩



⊕



4)розщеплена вершина 3 у 2 графі



5)

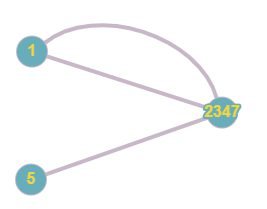
Виділимо підграф 347

A picture containing umbrella

Description automatically generated

A close up of an umbrella

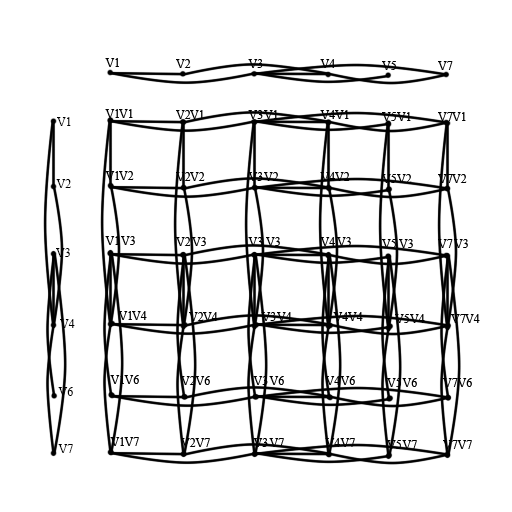
Description automatically generated



A picture containing object

Description automatically generated

6) добуток

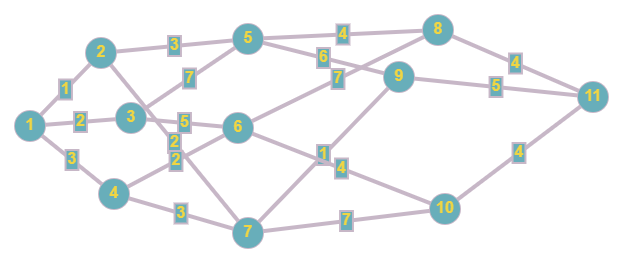
****

**2. Таблиця суміжності:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **1** | **0** | **1** | **1** | **0** | **0** | **0** | **1** | **0** | **0** |
| **2** | **1** | **0** | **1** | **1** | **0** | **0** | **1** | **0** | **0** |
| **3** | **1** | **1** | **0** | **1** | **0** | **0** | **0** | **0** | **1** |
| **4** | **0** | **1** | **1** | **0** | **1** | **0** | **0** | **0** | **1** |
| **5** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** | **1** |
| **6** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **1** |
| **7** | **0** | **1** | **0** | **0** | **0** | **0** | **0** | **1** | **1** |
| **8** | **0** | **1** | **0** | **0** | **0** | **0** | **1** | **0** | **1** |
| **9** | **0** | **0** | **1** | **1** | **1** | **1** | **1** | **0** | **0** |

Діаметр графа - 3, оскільки це максимальний ексцентриситет вершини, тобто відстань до найвіддаленішої від неї вершини.

**3.**



1) метод Краскала

Запишемо пари вершин та відстані між ними:

(1,2) = 1;

(7,9) = 1;

(1,3) = 2;

(4,6) = 2;

(2,7) = 2;

(4,7) = 3;

(2,5) = 3; (6,10) = 4;

(5,6) = 4;

(10,11)=4;

A screenshot of a video game

Description automatically generated

Отже, вага мінімального остового дерева графа дорівнює 26.

2) метод Прима

Запишемо пари вершин та відстань між ними:

(1,2)=1;

(2,7)=2;

(7,9)=1;

(1,3)=2;

(1,4)=3;

(4,6)=2;

(2,5)=3;

(5,8)=4;

(11-10)=4;

(8,11)=4;

A person posing for the camera

Description automatically generated

Отже, вага мінімального остового дерева графа дорівнює 26.

# Завдання з додатку 2

A close up of a piece of paper

Description automatically generated

**Результат**

**#include<iostream>**

**using namespace std;**

**int main()**

**{**

**int a,b,u,v,n,i,j,ne=1;**

**int visited[20]={0},min,mincost=0,cost[20][20];**

**int path[100]={0};**

**int path\_index=0;**

**cout<<"Enter the number of vertices: ";**

**cin>>n;**

**cout<<"Enter the adjacency matrix: " << endl;**

**for(i=1;i<=n;i++){**

**for(j=1;j<=n;j++)**

**{**

**cin>>cost[i][j];**

**if(cost[i][j]==0)**

**cost[i][j]=99;**

**}**

**}**

**visited[1]=1;**

**cout<<endl;;**

**while(ne < n)**

**{**

**for(i=1,min=99;i<=n;i++)**

**for(j=1;j<=n;j++)**

**if(cost[i][j]< min)**

**if(visited[i]!=0)**

**{**

**min=cost[i][j];**

**a=u=i;**

**b=v=j;**

**}**

**if(visited[u]==0 || visited[v]==0)**

**{**

**path[path\_index]=b;**

**path\_index++;**

**ne++;**

**mincost+=min;**

**visited[b]=1;**

**}**

**cost[a][b]=cost[b][a]=99;**

**}**

**cout<<endl;**

**cout<<1<<" -> ";**

**for (int i=0;i<n-1;i++)**

**{**

**cout<<path[i];**

**if (i<n-2) cout<<" -> ";**

**}**

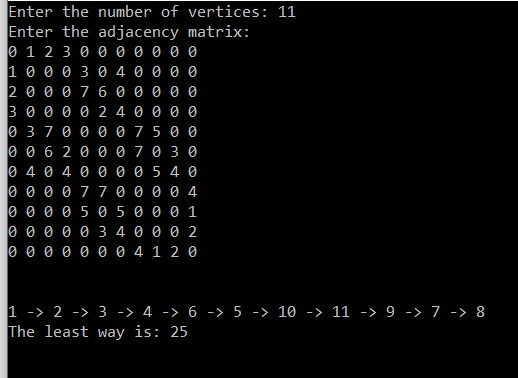
**cout<< endl << "The least way is: "<<mincost;**

**cin.get();**

**cin.get();**

**return 0;**

**}**



**Висновок:** На лабораторній роботі №4, я навчилась виконувати операції над графами, застосовувати алгоритми Краскала і Прима для знаходження мінімального остового дерева.