ДОДАТОК

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота №4

з дисципліни
«Дискретна математика»

Виконала:
студент групи КН-114
Ярка Ірина
Викладач:
Мельникова Н.І.

Львів – 2019 р.

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 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 одну 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.

Будь яке ребро e *інцедентно* двом вершинам (v', v''), які воно з'єднує. У свою чергу вершини (v', v'') інцендентні до ребра e .

Дві вершини (v', v'') називають *суміжними*, якщо вони належать до одного й того самого ребра e.

Степенем вершини графа G називається число інцидентних їй ребер.

Граф, який не має ребер називається *пустим графом*, *нуль-графом*. Вершина графа, яка не інцедентна до жодного ребра, називається *ізольованою*. Вершина графа, яка інцедентна тільки до одного ребра, називається *звисаючою*.

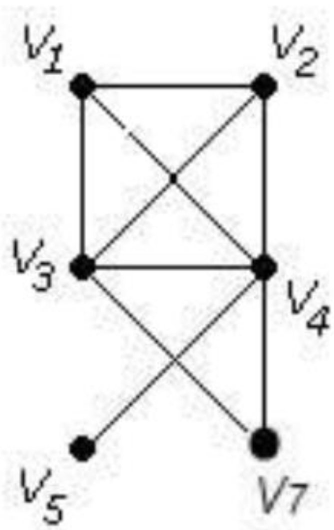
Частина $G' = (V', E')$ графа $G = (V, E)$ називається *підграфом* графа G , якщо $V' \subseteq V$ і E' складається з тих і тільки тих ребер $e = (v', v'')$, у яких обидві кінцеві вершини $v', v'' \in V'$. Частина $G' = (V' , E')$ називається *суграфом* або *остовим підграфом* графа G , якщо виконано умови: $V' = V$, $E' \subseteq E$.

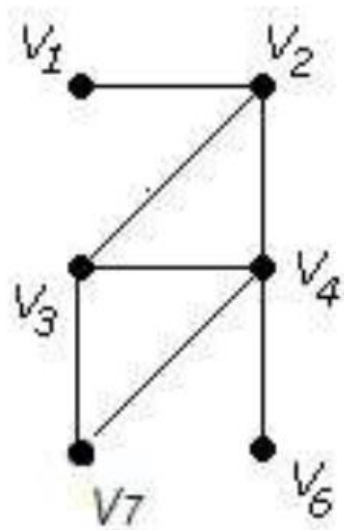
Варіант 15

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

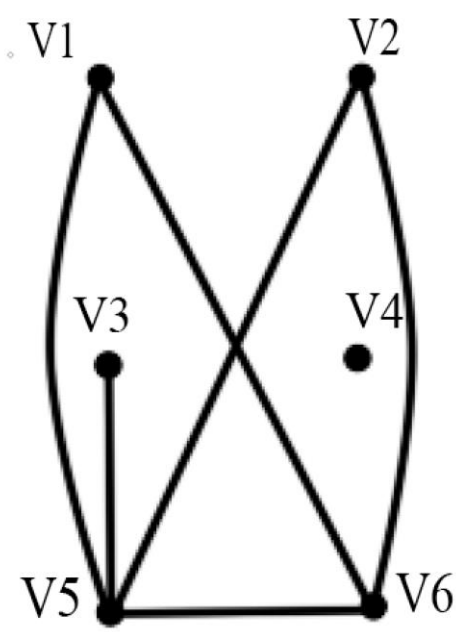
1. Виконати наступні операції над графами:

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.

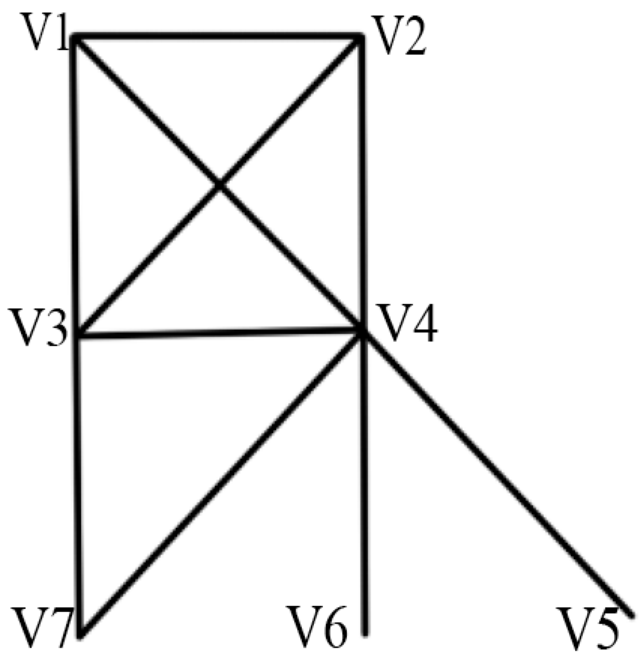




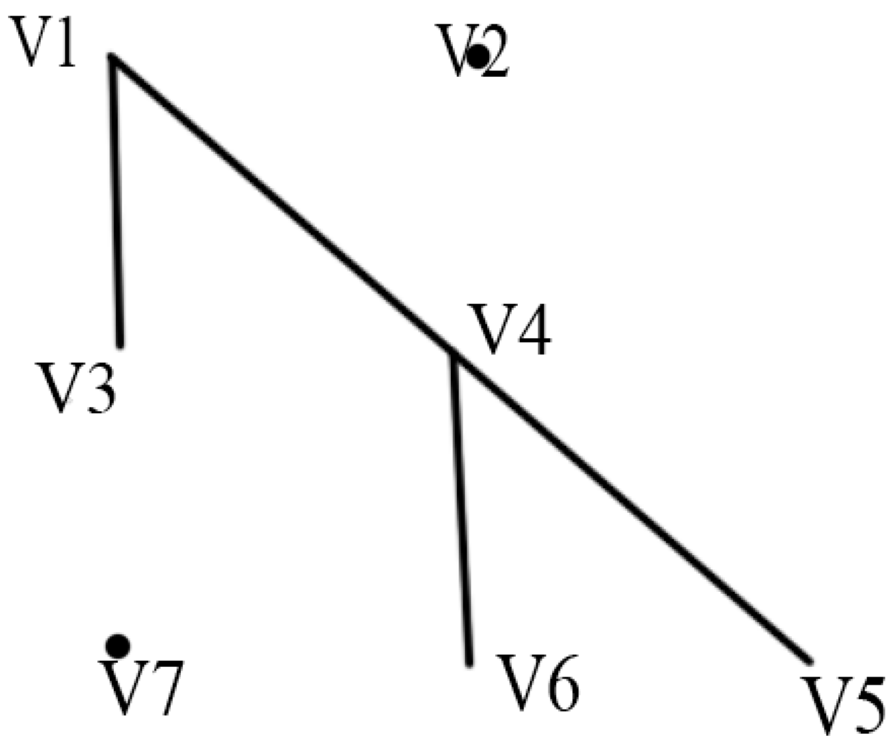
1) Доповнення до першого графу



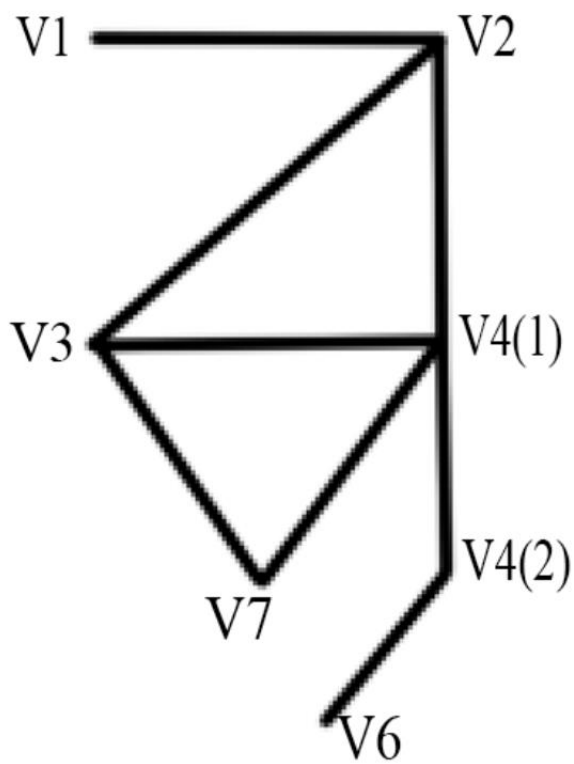
2) Об’єднання графів



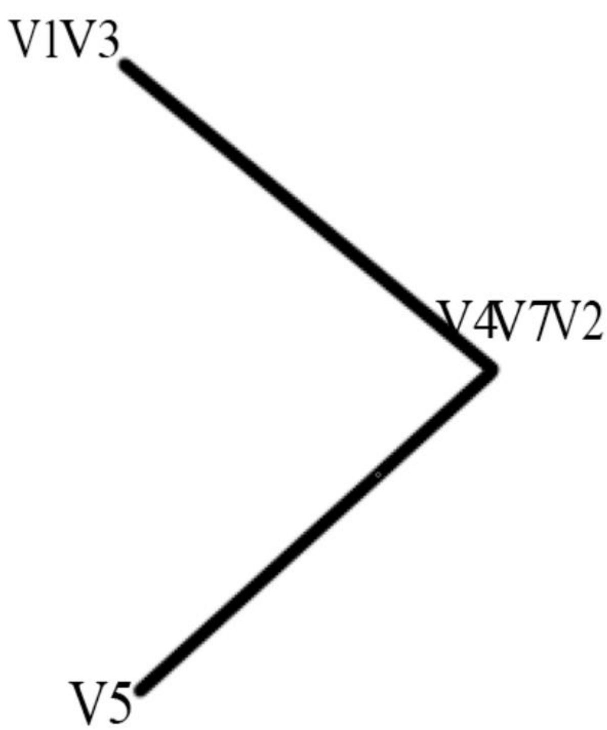
3) Кільцева сума графів



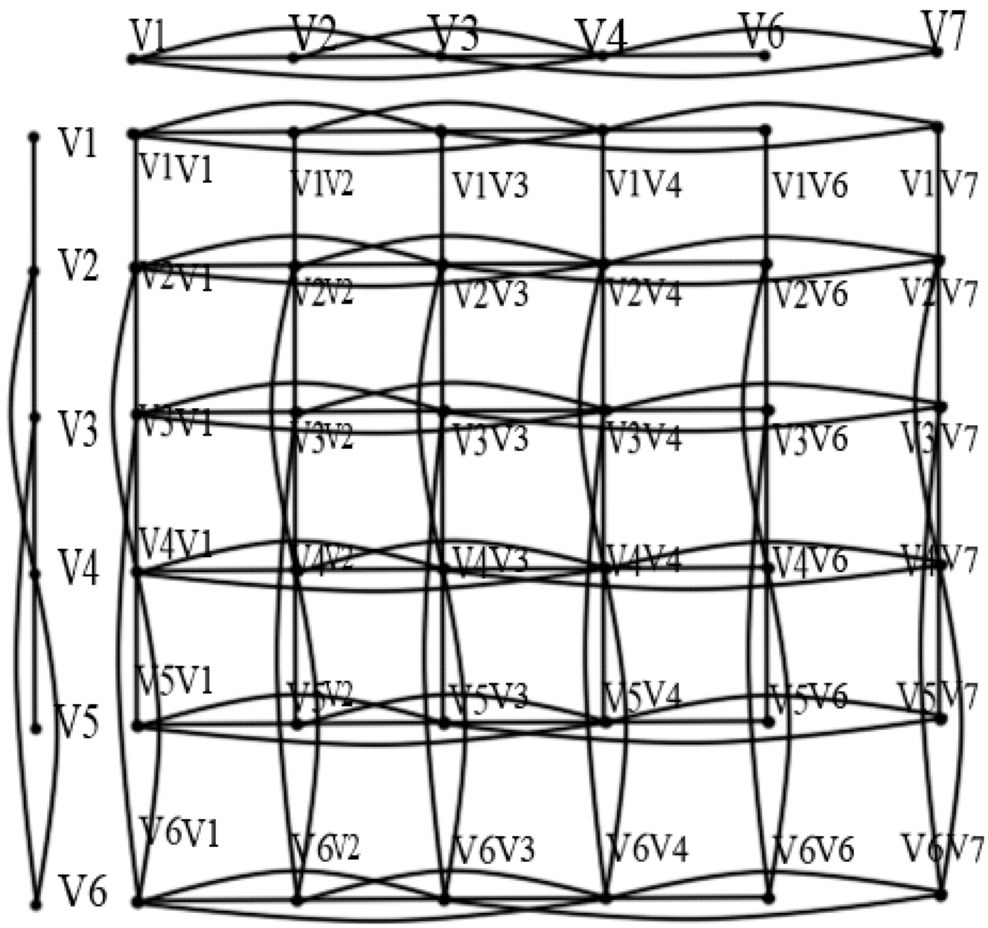
4) Розщеплення вершини V4



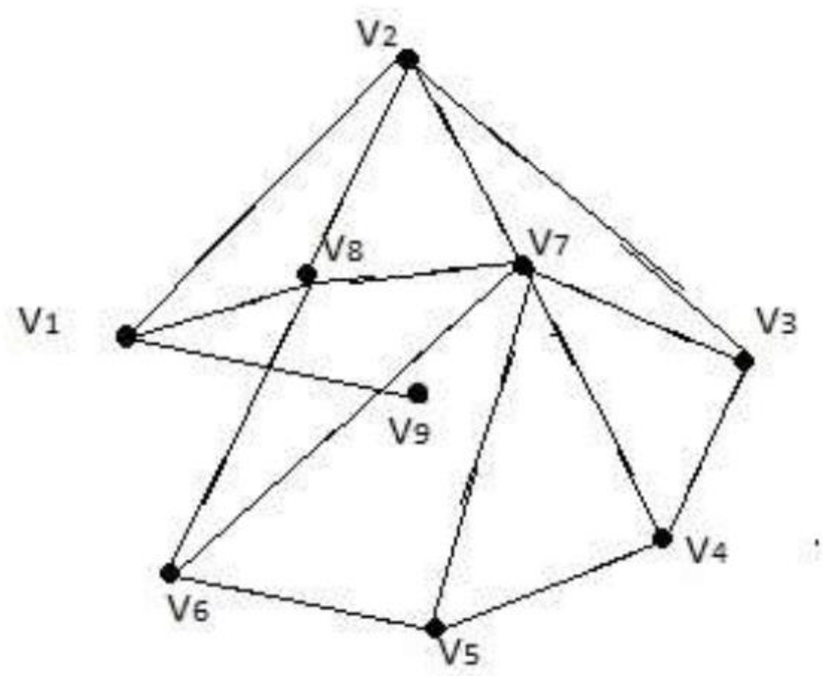
5) Стягнення A в G1
Виділений підграф – V1V4V5;
Стягуємо V4 і V7;
Стягуємо V4V7 і V2;
СтягуємоV3 і V1;



6)Множення графів



2. Знайти таблицю суміжності та діаметр графа.

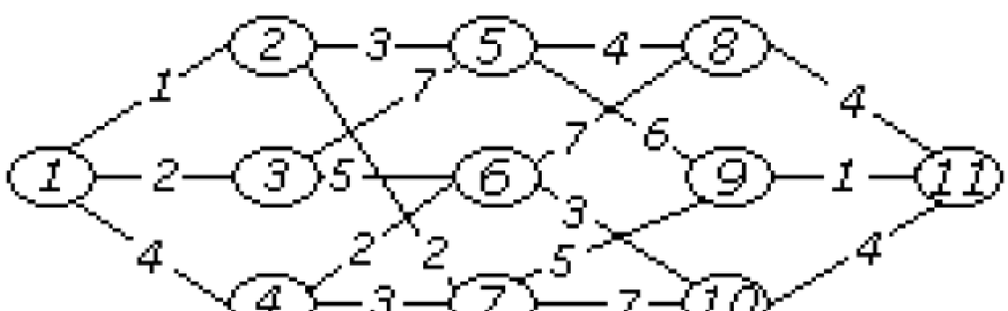


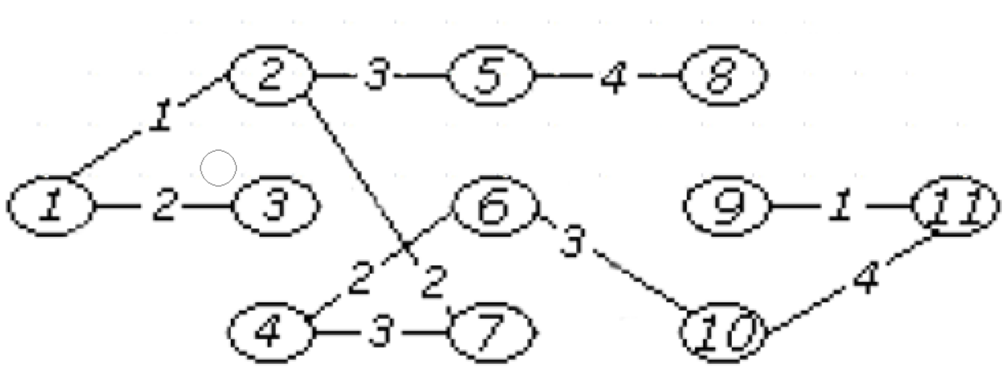
Матриця суміжності

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

Діаметр матриці = 4
4 -> 3 -> 2 -> 1 -> 9

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





Завдання №2. Написати програму, яка реалізує алгоритм знаходження остового дерева мінімальної ваги згідно свого варіанту.

Скріншот коду програми:

```cpp
#include <cstdio>
#include <iostream>
#include <climits>

using namespace std;

int minKey(int key[], bool mstSet[], int V){
    int min = INT_MAX, min_index = 0;
    for (int v = 0; v < V; v++)
        if (!mstSet[v] && key[v] < min)
            min = key[v], min_index = v;
    return min_index;
}
int main()
{
    int x, y, w, vert, e;
    cout << "Enter the number of vertices and edges: "<< endl;
    cin >> vert >> e;
    int graph[vert][vert];
    for(int i = 0; i < vert; i++){
        for(int j = 0; j < vert; j++){
            graph[i][j] = 0;
        }
    }
    cout <<"V1 V2 Weight" << endl;
    for(int i = 0; i < e; i++){
        cin>>x>>y>>w;
        graph[x-1][y-1] = w;
        graph[y-1][x-1] = w;
    }
    int key[vert], parent[vert];
    bool mstSet[vert];
    for (int i = 0; i < vert; i++)
        key[i] = INT_MAX, mstSet[i] = false;
    key[0] = 0;
    parent[0] = -1;
    for (int count = 0; count < vert - 1; count++){
        int u = minKey(key, mstSet, vert);
        mstSet[u] = true;
        for (int v = 0; v < vert; v++)
            if (graph[u][v] && !mstSet[v] && graph[u][v] < key[v])
                parent[v] = u, key[v] = graph[u][v];
    }
    cout<<"Minimum spanning tree:\n";
    for (int i = 1; i < vert; i++)
        printf("%d - %d\n", parent[i]+1, i+1);
    return 0;
}
```

Скріншот результату виконання:

```
Enter the number of vertices and edges:
11 18
V1 V2 Weight
1 2 1
1 3 2
1 4 4
2 5 3
2 7 2
3 6 5
3 5 7
4 6 2
7 10 7
6 10 3
7 9 5
9 11 1
10 11 4
8 11 4
5 9 6
5 8 4
6 8 7
4 7 3
Minimum spanning tree:
1 - 2
1 - 3
7 - 4
2 - 5
4 - 6
2 - 7
5 - 8
11 - 9
6 - 10
10 - 11
```

Висновок

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