МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота № 4

з дисципліни

«Дискретна математика»

Виконала:

студентка групи КН-114

Кемська Юлія

Викладач:

Мельникова Н.І.

Львів – 2019

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 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.

Варіант -9

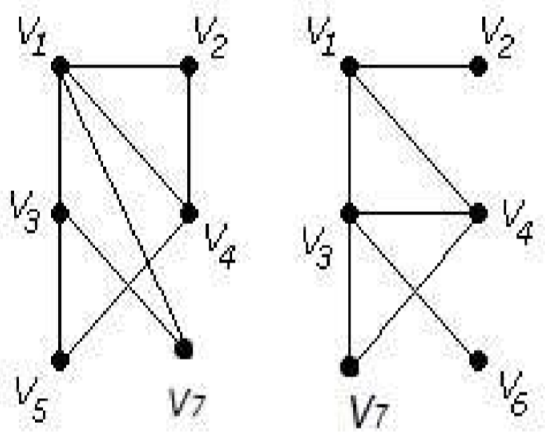
ІНДИВІДУАЛЬНІ ЗАВДАННЯ

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

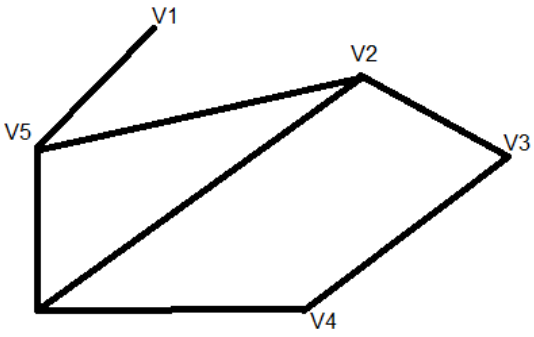
1. Виконати наступні операції над графами:

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 А, що складається з 3-х вершин в G1 і знайти стягнення А в G1 (G1\ А), 6) добуток графів.

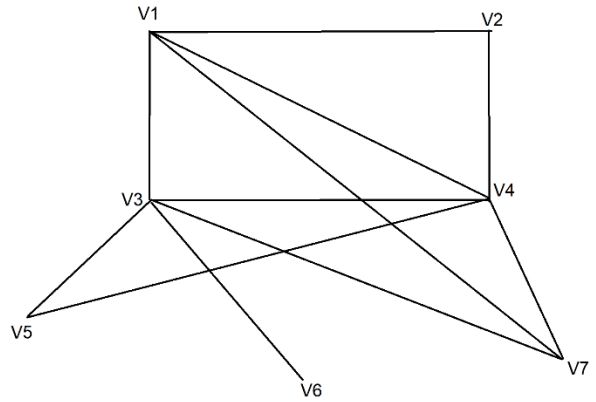
9



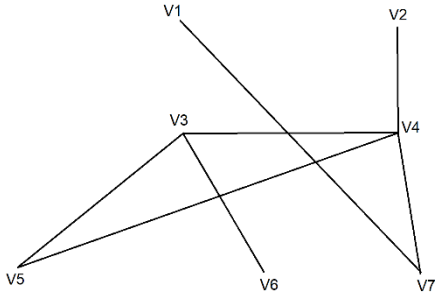
1)



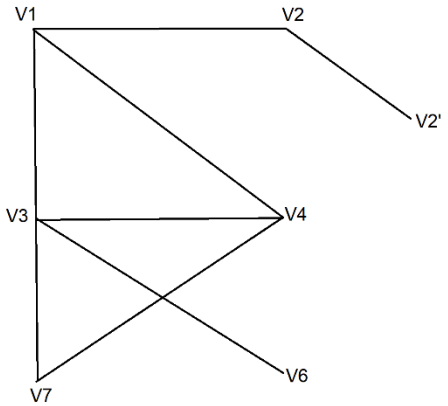
2)



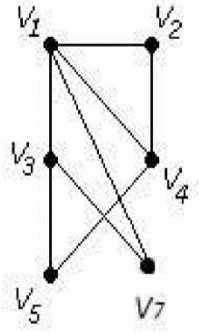
3)

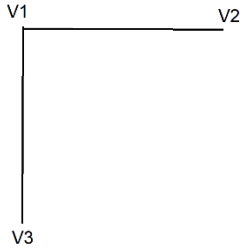


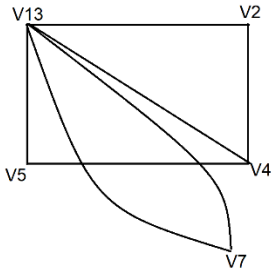
4)

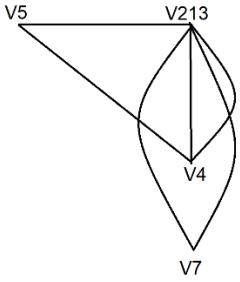


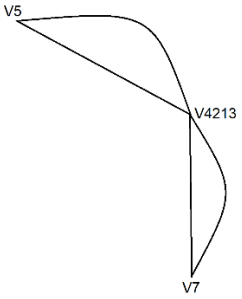
5)


V1
V2
V3
V4
V5
V7


V1
V2
V3

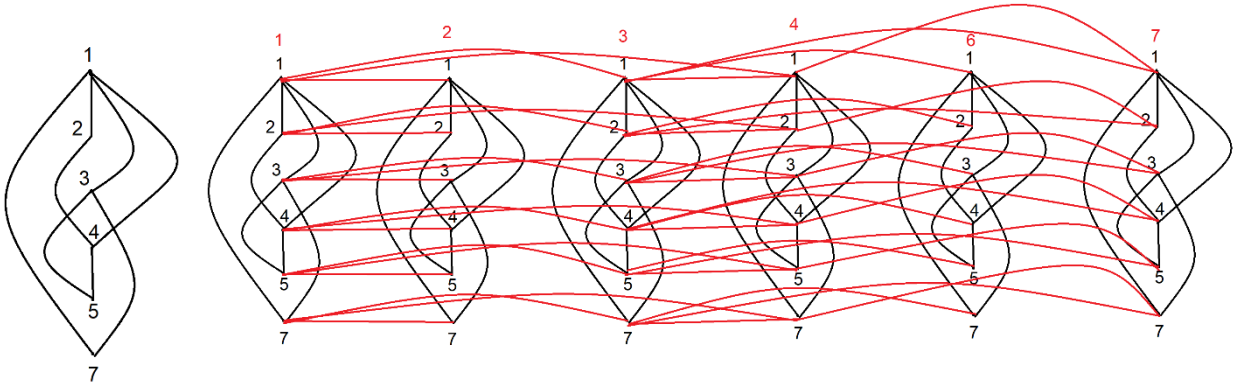

V13
V2
V5
V4
V7


V5
V213
V4
V7


V5
V4213
V7

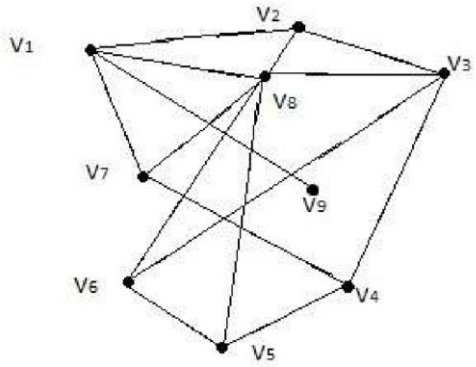
6)


1
2
3
4
6
7


1
2
3
4
5
7
1
2
3
4
6
7

2. Знайти таблицю суміжності та діаметр графа.

9

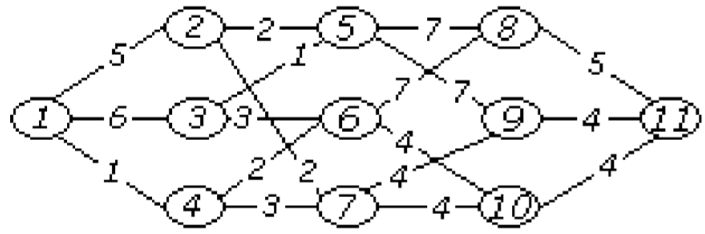


	V1	V2	V3	V4	V5	V6	V7	V8
V1	0	1	0	0	0	0	1	1
V2	1	0	1	0	0	0	0	1
V3	0	1	0	1	0	1	0	1
V4	0	0	1	0	1	0	1	0
V5	0	0	0	1	0	1	0	1
V6	0	0	1	0	1	0	0	1
V7	1	0	0	1	0	0	0	1
V8	1	1	1	0	1	1	1	0

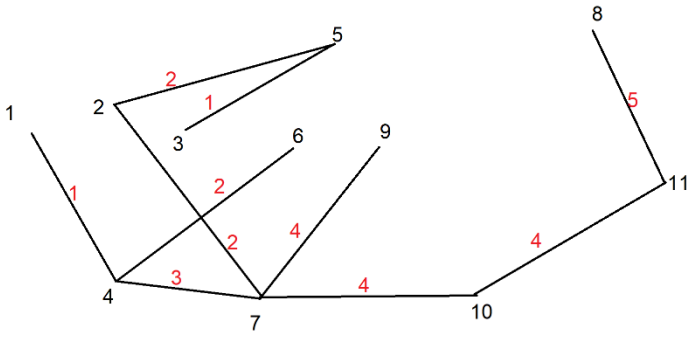
D=3

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

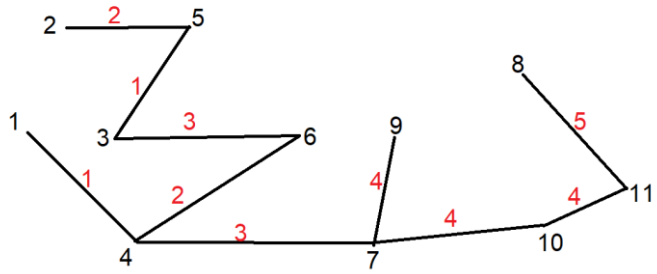
9



Алгоритм Краскала



Алгоритм Прима



Додаток 2:

```
#include <iostream>
using namespace std;
int ways[20], a[20][20],n,vis[20],vers[20],svers = 0, m,val,s,f;

int min_array(int n){
    int min = INT_MAX, nmin = 0;
    for(int i=1;i<=n;i++){
        if(ways[i]<min && vis[i]==0){
            min = ways[i];nmin = i;
        }
    }
    return  nmin;
}
int main() {
    cout<<"Enter amount of vertices and edges of graph\n";
    cin>>n>>m;
    cout<<"Enter number of start vertice,end vertice and edge weight between them\n";
    for(int i=1;i<=n;i++)
        ways[i] = INT_MAX;
    for(int i = 1;i<=m;i++){
        cin>>s>>f>>val;
        a[s][f] = val;
        a[f][s] = val;
    }
    ways[1] = 0;
    int start = 1;
    while(min_array(n)!=0){
        vis[start] = 1;
        for(int i=1;i<=n;i++){
            if(a[start][i]!=0 && vis[i] ==0  )ways[i] = min(ways[i], a[start][i]);
        }
            svers++;
        vers[svers] = start;
        start = min_array(n);
    }
    cout<<"The order of tops after algoritm passing\n";
    for(int i=1;i<=n;i++){
        cout<<vers[i]<<' ';
    }
}

```

Результат:

```
Enter amount of vertices and edges of graph
11 19
Enter number of start vertice,end vertice and edge weight between them
1 2 4
1 3 3
1 4 7
2 5 2
2 7 1
3 5 1
3 6 7
4 7 2
4 6 2
5 8 4
5 9 7
6 8 4
6 10 5
7 9 3
7 4 2
7 10 3
8 11 4
9 11 6
10 11 5
The order of tops after algoritm passing
1 3 5 2 7 4 6 9 10 8 11
Process returned 0 (0x0)   execution time : 778.694 s
Press any key to continue.
```