Міністерство освіти і науки України

Національному університеті "Львівська Політехніка"

Кафедра систем штучного інтелекту

Лабораторна робота № 4

з дисципліни

Виконав:

студент групи КН-114

Сиротюк Владислав

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 ТА ПРИКЛАДИ РОЗВ'ЯЗАННЯ ЗАДАЧ

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 = υ ; V = {υ}, 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.

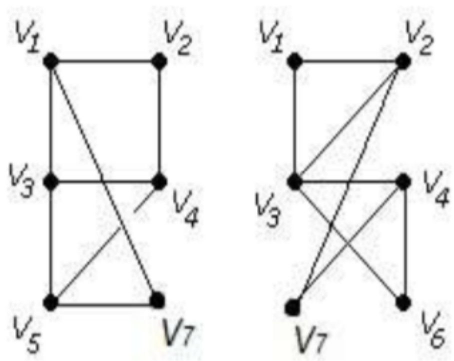
ІНДИВІДУАЛЬНІ ЗАВДАННЯ

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

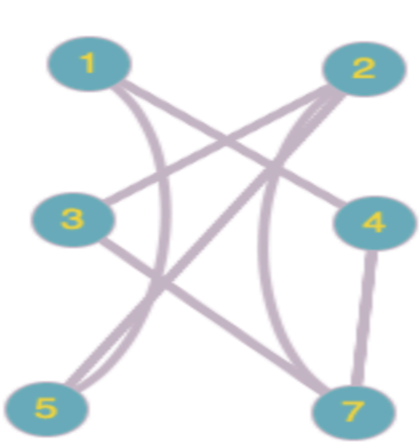
1. Виконати наступні операції над графами:

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

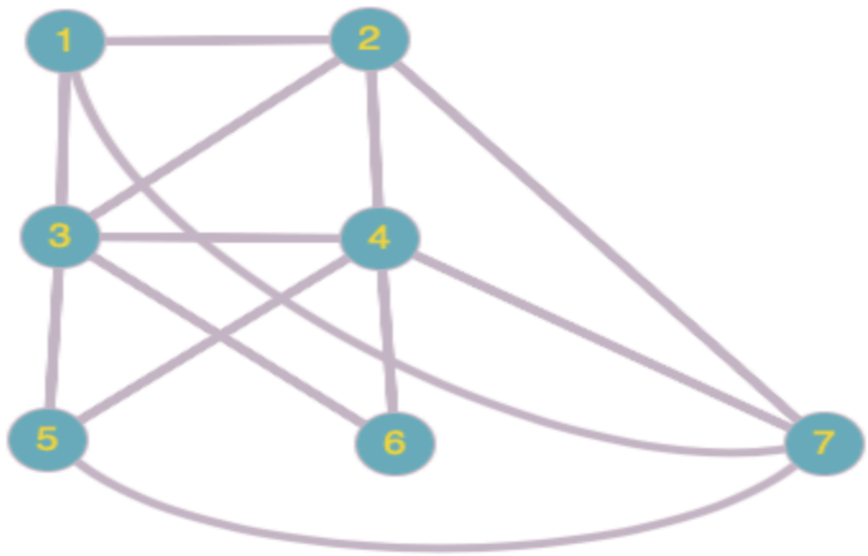
10



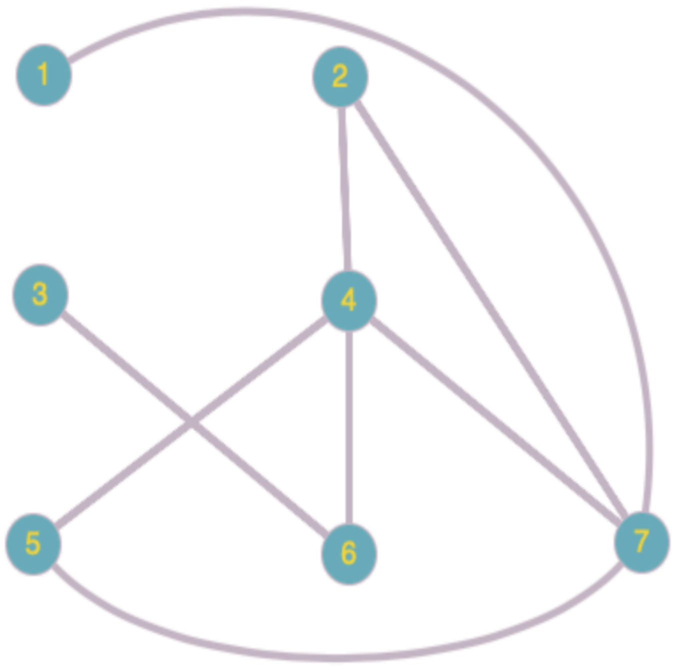
1)



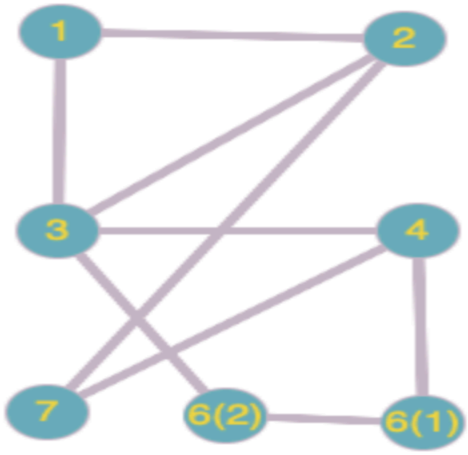
2)



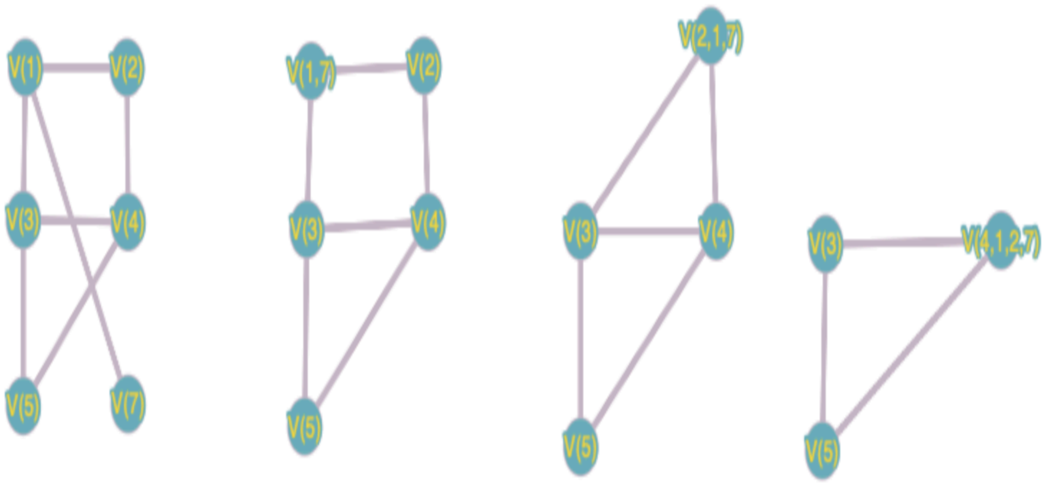
3)



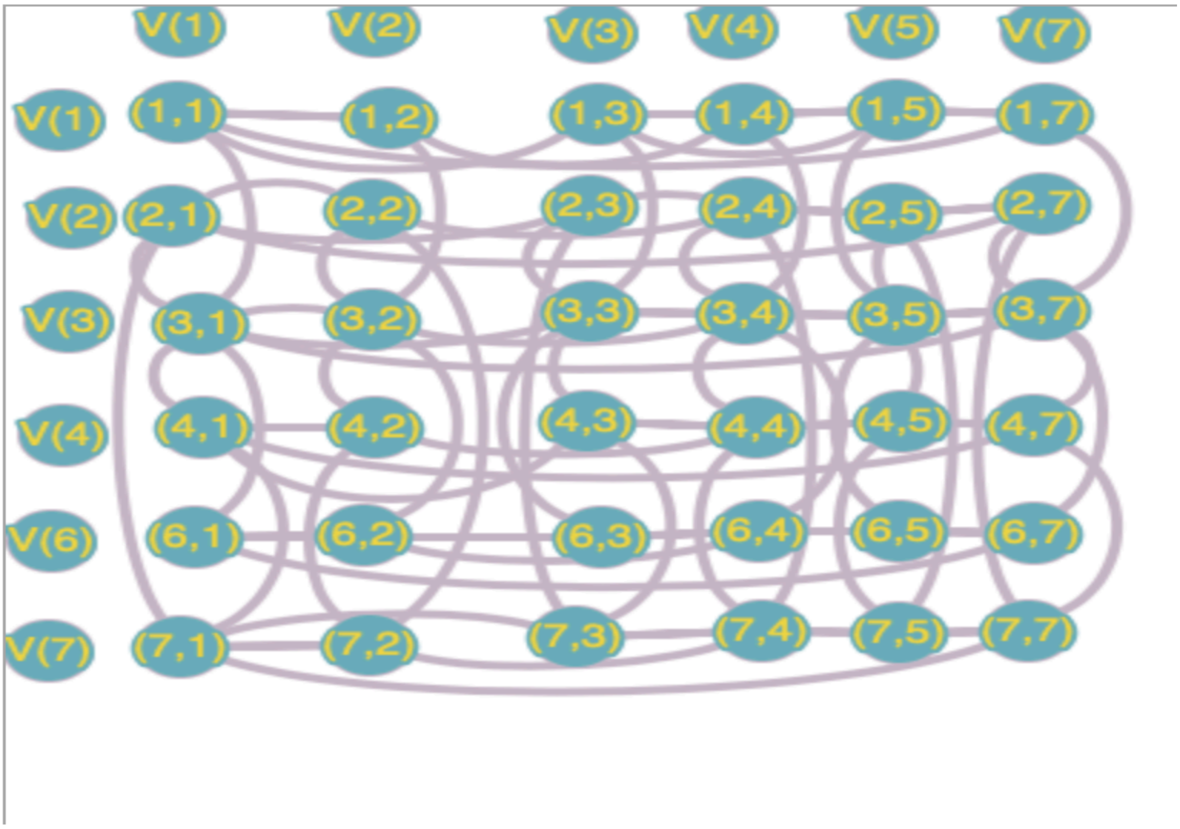
4)


1
2
3
4
7
6(2)
6(1)

5)

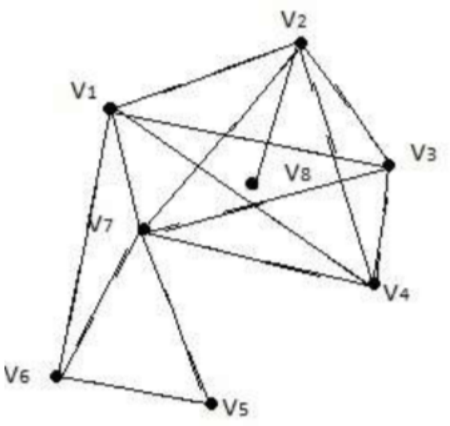

V(1)
V(2)
V(3)
V(4)
V(5)
V(7)
V(1,7)
V(2)
V(3)
V(4)
V(5)
V(2,1,7)
V(3)
V(4)
V(5)
V(3)
V(4,1,2,7)
V(5)

б)



2. Знайти таблицю суміжності та діаметр графа.

10



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

Діаметр графа = 3

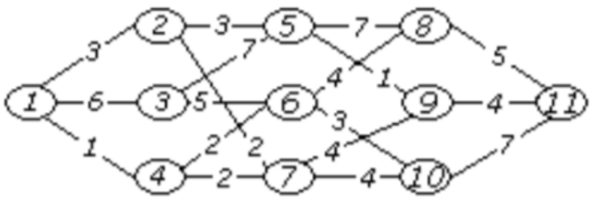
Відстань між V8 і V5 є найдовшою.

3)

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

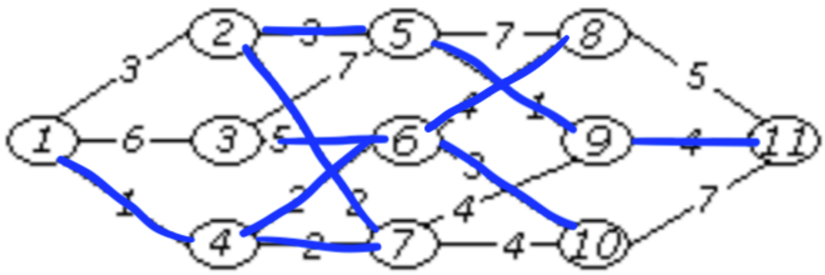
1

10



Алгоритм Краскала:

10



1)(1;4)-1;

(5;9)-1

2)(4;7)-2

(7;2)-2

(4;6)-2

3)(2;5)-3

(6;10)-3

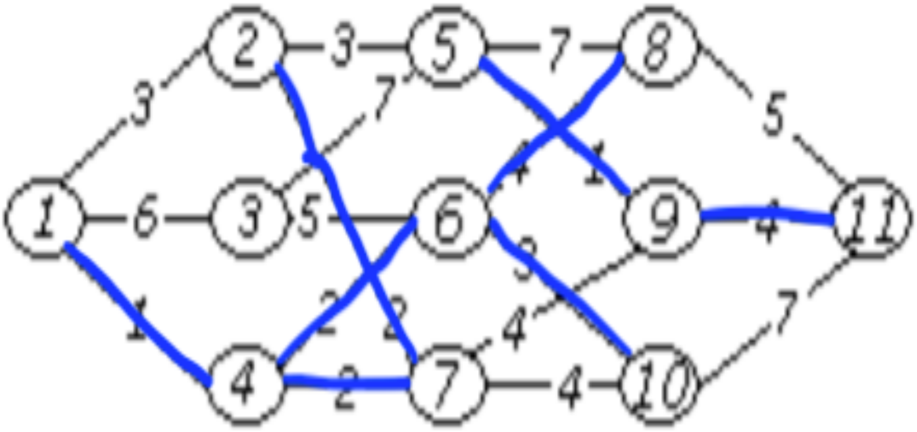
4)(9;11)-4

(6;8)-4

5)(3;6)-5

Алгоритм Прима:

10

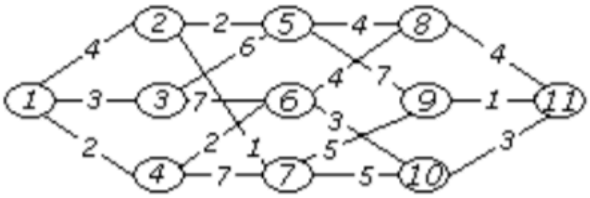


1) (1;4) - 1
2) (4;7) – 2
3) (2;7) – 2
4) (4;6) – 2
5) (6;10) – 3
6) (6;8) – 4
7) (5;9) - 1
8) (9;11) - 4

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

Варіант № 10

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




```
#include <iostream>
#include<algorithm>
#include <vector>
using namespace std;

int main()
{
  int n,m,weight,x,y;
  cout << "kil versun:";cin >> n;
  cout << "kil reber:";cin >> m;
  vector < pair < int, pair<int, int> > > g;

  for (int i = 0; i < m; i++) {
    cout << "Rebro[" << i << "] = "<<endl;
    cout << "ver1:";cin >> x;
    cout << "ver2:";cin >> y;
    cout << "weight:";cin >> weight;
    g.push_back({weight, {--x,--y}});
  }


  int cost = 0;
  vector < pair<int, int> > res;
  sort(g.begin(), g.end());
  vector<int> tree_id(n);


  for (int i = 0; i < n; ++i)
    tree_id[i] = i;
  for (int i = 0; i < m; ++i)
  {
    int a = g[i].second.first, b = g[i].second.second, l = g[i].first;
    if (tree_id[a] != tree_id[b])    {
```

```
      cost += l;
      res.push_back(make_pair(a, b));
      int old_id = tree_id[b], new_id = tree_id[a];
      for (int j = 0; j < n; ++j)
        if (tree_id[j] == old_id)
          tree_id[j] = new_id;
    }
  }
  for (auto index : res) {
    cout << index.first << " - " << index.second << endl;;
  }
    return 0;
}
```

Висновок: на цій лабораторній роботі я набув практичних вмінь та навичок з використання алгоритмів Прима і Краскала.