МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота №4

з дисципліни

«Дискретна математика»

Виконав:
студент групи КН-113
Зварич Адріана
Викладач: Мельникова Н.І.

Львів – 2019 р.

Тема роботи:

Основні операції над графами. Знаходження остова мінімальної ваги за алгоритмом Прима-Краскала

Мета роботи:

Набуття практичних вмінь та навичок з використання алгоритмів Прима 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=*v; V={v}, E*-множина упорядкованих або неупорядкованих пар e = (v ′,v ″) , v ′∈ V , v ″∈ 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 (оргграф)** – 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.

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 , і несуміжні у протилежному випадку. Два ребра називають суміжними, якщо вони мають спільну вершину. Відношення суміжності як для вершин, так і для ребер є симетричним відношенням. **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: $V' = V$, $E' \subseteq E$.

Операції над графами

1.Вилучення ребра
2.Доповнення графа
3.Об'єднання графів
4. Кільцева сума графів
5. Розщеплення вершини графа
6. Стягування ребра
7. Добуток графів

Ексцентриситет вершини графа – відстань до максимально віддаленої від неї вершини. Для графа, для якого не визначена вага його ребер, відстань визначається у вигляді числа ребер.

Радіус графа – мінімальний ексцентриситет вершин.

Діаметр графа – максимальний ексцентриситет вершин.

Діаметром зв'язного графа називається максимально можлива довжина між двома його вершинами.

Алгоритми знаходження найкоротшого кістякового дерева

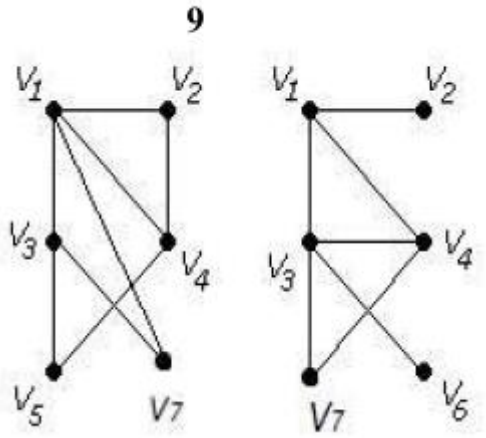
Алгоритм Краскала. Перший етап – підготовчий, для даного графа G упорядковуються ребра $e \in E$ у послідовність $e_1, e_2, \ldots, e_m$, $m=|E|$, у порядку неспадання ваг цих ребер: $w(e_1) \le w(e_2) \le \ldots \le w(e_s) \le \ldots \le w(e_m)$.

Другий етап виконується по кроках $s=1, 2, \ldots, m_0 \le m$ у такий спосіб. На кроках $s=1, 2$ ребра e_1, e_2 з послідовності офарблюються. На кожному наступному кроці s розглядається ребро e_s з послідовності, і воно офарблюється тоді і тільки тоді, коли не утворює циклу з ребрами, пофарбованими на попередніх кроках. У противному випадку ребро e_s умовно викреслюється з графа $G=(V,E)$. Алгоритм закінчує роботу на кроці $s=m_0$, коли пофарбованим виявиться $(n-1)$ по рахунку ребро e_s, $n=|V|$, тому що по необхідності $n-1$ пофарбованих ребер утворюють кістякове дерево n-вершинного графа.

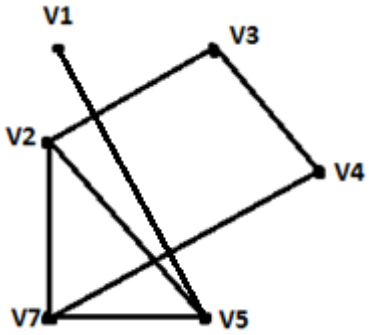
Варіант 9

Завдання №1.

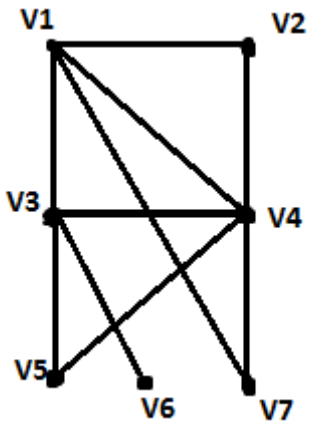
1. Виконати наступні операції над графами:



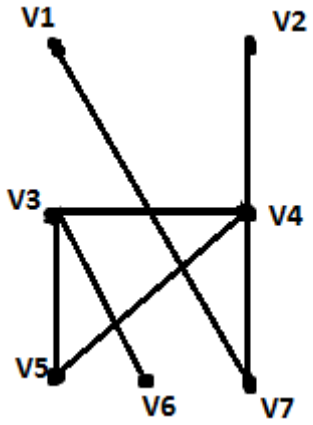
1)знайти доповнення до першого графу:



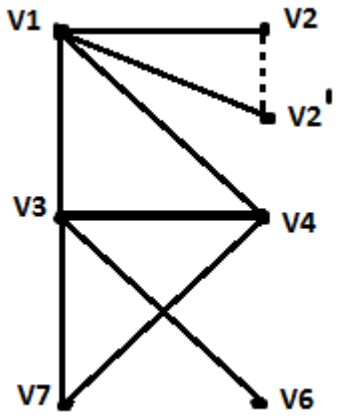
2) об’єднання графів:



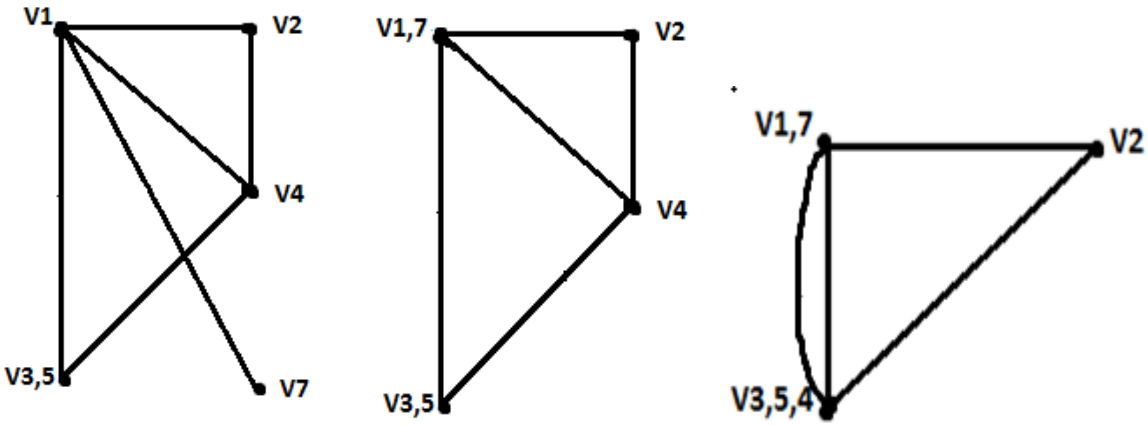
3) кільцеву суму G1 та G2 (G1+G2):



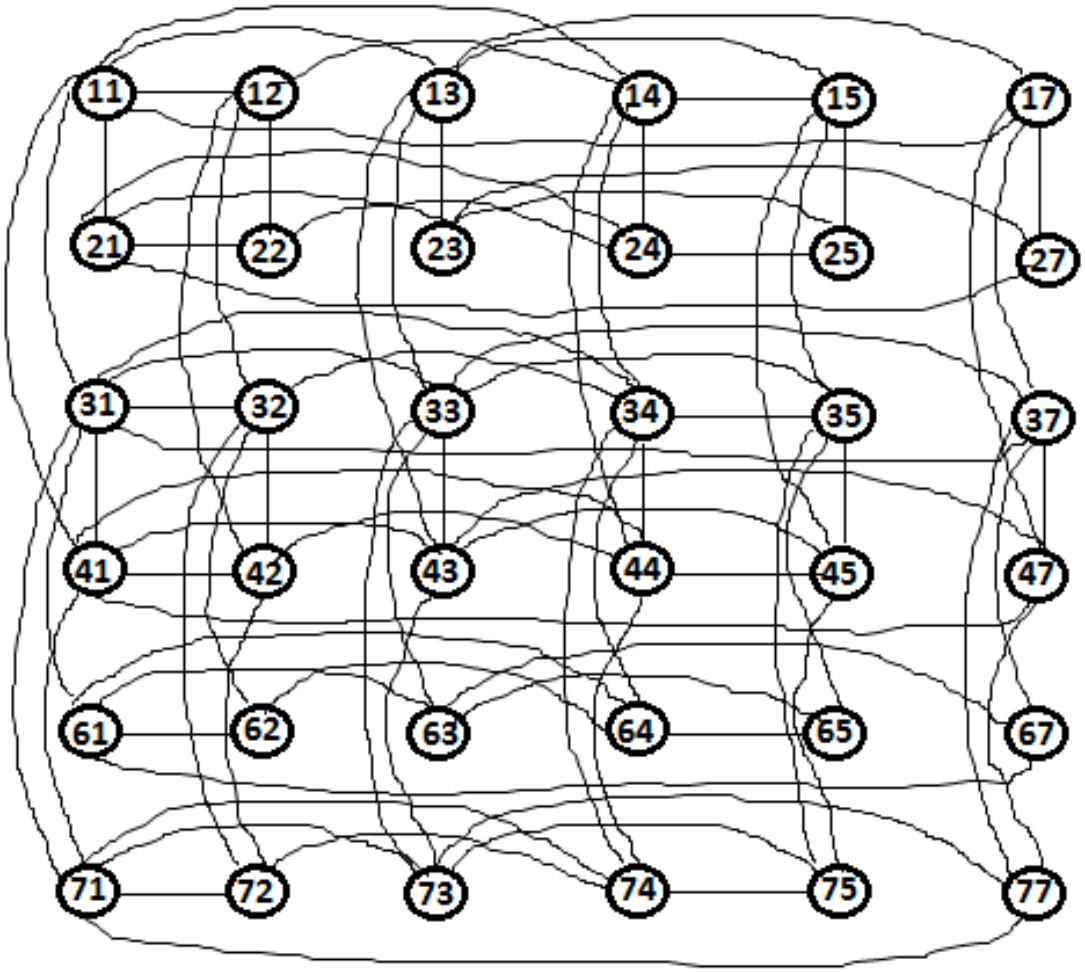
4) розщепити вершину у другому графі:



5) виділити підграф A, що складається з 3-х вершин в G1 і знайти стягнення A в G1:

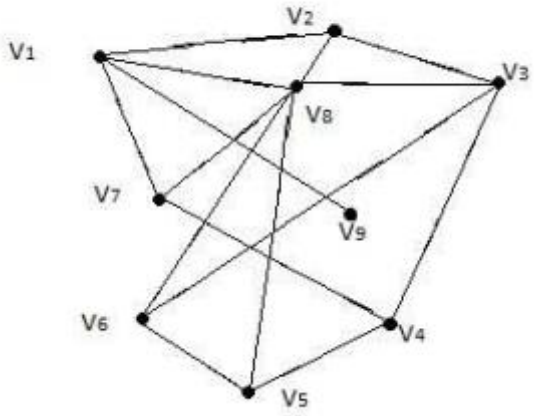


6) добуток графів:



2. Знайти таблицю суміжності та діаметр графа.

9

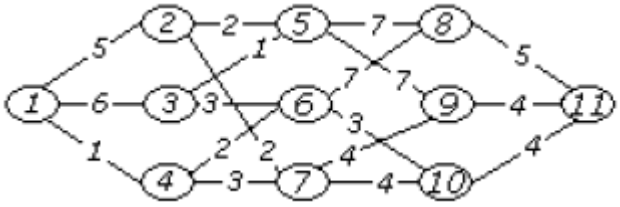


Таблиця суміжності виглядатиме так:

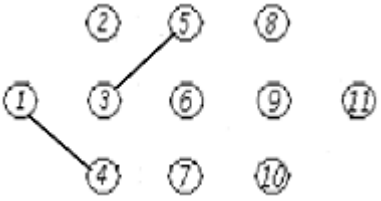
	V1	V2	V3	V4	V5	V6	V7	V8	V9
V1	0	1	0	0	0	0	1	1	1
V2	1	0	1	0	0	0	0	1	0
V3	0	1	0	1	0	1	0	1	0
V4	0	0	1	0	1	0	1	0	0
V5	0	0	0	1	0	1	0	1	0
V6	0	0	1	0	1	0	0	1	0
V7	1	0	0	1	0	0	0	1	0
V8	1	1	1	0	1	1	1	0	0
V9	1	0	0	0	0	0	0	0	0

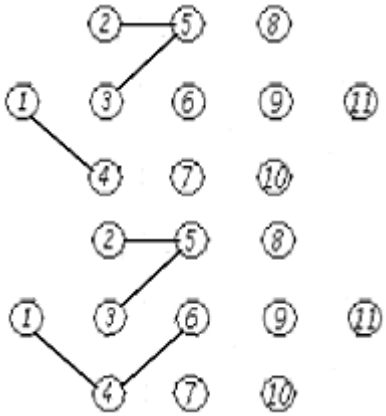
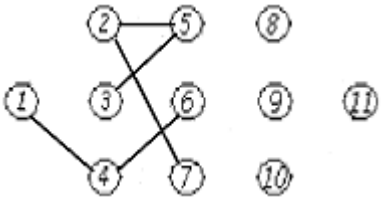
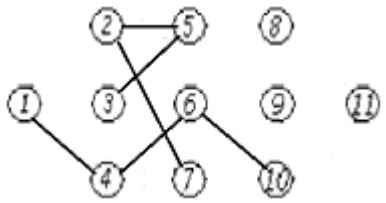
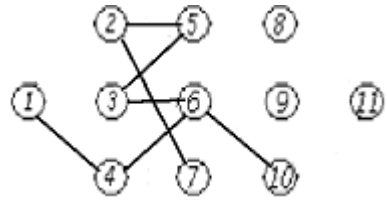
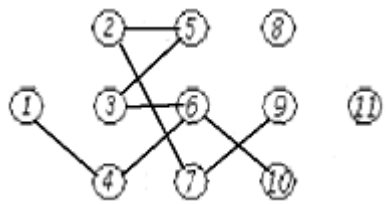
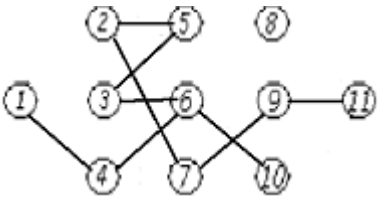
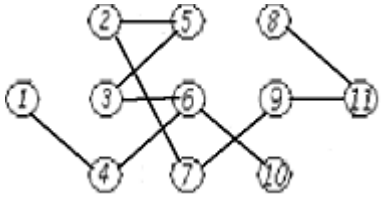
Діаметр графа дорівнює 3.

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

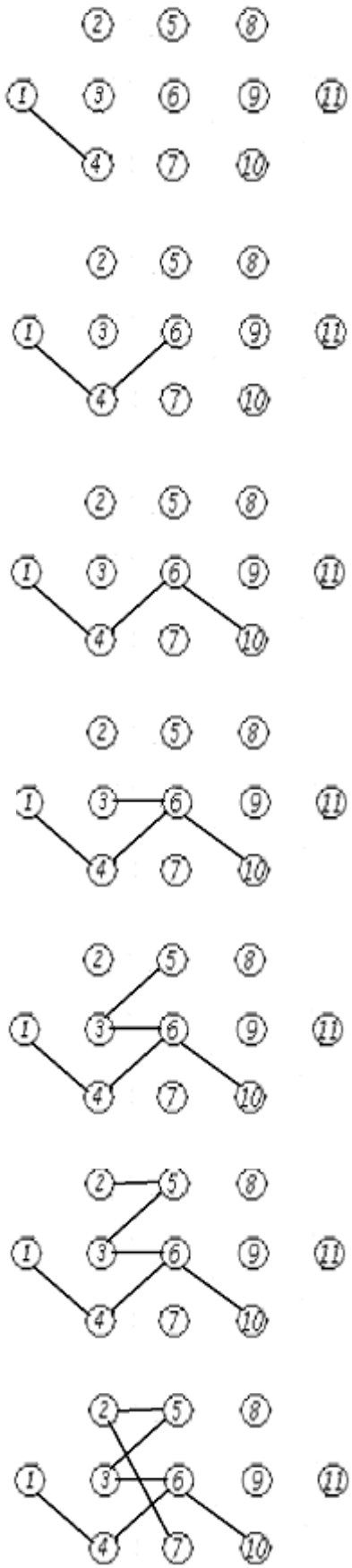


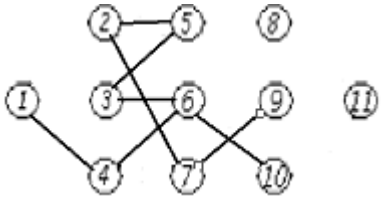
Метод Краскала:
V(G) = {1,4,3,5,2,6,7,10,9,11,8};
E(G) = {(1,4),(3,5),(2,5),(4,6),(2,7),(6,10),(3,6),(7,9),(9,11),(8,11);

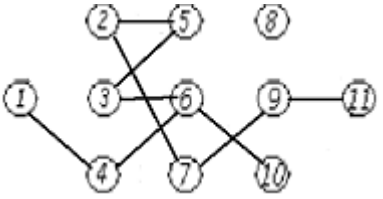


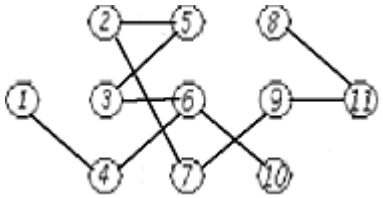








Алгоритм Прима:
V(G) = {1,4,6,10,3,5,2,7,9,11,8};
E(G) = {(1,4),(4,6),(6,10),(6,3),(3,5),(5,2),(2,7),(7,9),(9,11),(11,8)};

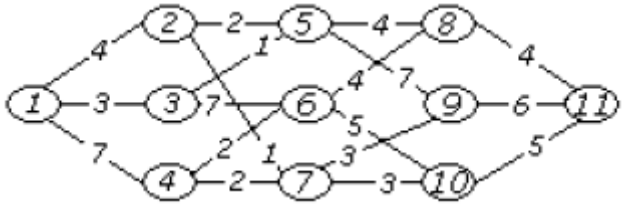








Завдання №2. Написати програму, яка реалізує алгоритм знаходження остового дерева мінімальної ваги згідно свого варіанту. За алгоритмом Прима знайти мінімальне остове дерево графа. Етапи розв'язання задачі виводити на екран. Протестувати розроблену програму на наступному графі:



Програма:

```
#include<stdio.h>
#include<stdlib.h>
#define size 11
int main()
{
int wght =0;
int graph[size][size] = {
{ 0, 4, 3, 7, 0, 0, 0, 0, 0, 0, 0 },
{ 4, 0, 0, 0, 2, 0, 1, 0, 0, 0, 0 },
{ 3, 0, 0, 0, 1, 7, 0, 0, 0, 0, 0 },
{ 7, 0, 0, 0, 0, 2, 2, 0, 0, 0, 0 },
{ 0, 2, 1, 0, 0, 0, 0, 4, 7, 0, 0 },
{ 0, 0, 7, 2, 0, 0, 0, 4, 0, 5, 0 },
```

```
{ 0, 1, 0, 2, 0, 0, 0, 0, 3, 3, 0 },
{ 0, 0, 0, 0, 4, 4, 0, 0, 0, 0, 4 },
{ 0, 0, 0, 0, 7, 0, 3, 0, 0, 0, 6 },
{ 0, 0, 0, 0, 0, 5, 3, 0, 0, 0, 5 },
{ 0, 0, 0, 0, 0, 0, 0, 4, 6, 5, 0 }
};
int visit[size] = {0};
int i, j, p=0, q=0;
int arr[size]={1,2,3,4,5,6,7,8,9,10,11};
int min;
int flag=0;
for(i=0;i<size;i++){
for(j=0;j<size;j++){
if(flag ==0 && graph[i][j]!=0){
flag=1;
p=i;
q=j;
min=graph[p][q];
}
else if(flag == 1 && graph[i][j]<min && graph[i][j]!=0){
p=i;
q=j;
min = graph[i][j];
}
}
}
visit[p]=1;
visit[q]=1;
int flag1=0;
int p1,q1,min1,qq=0;
printf("Ribs included i the minimum span tree \n");
printf("%d-(%d - weight of the rib)-%d ", arr[p], graph[p][q],
arr[q]);
do{
for(i=0;i<size;i++){
for(j=0;j<size;j++){
if(visit[i]==1 && visit[j]==0 && graph[i][j]!=0){
if(flag1==0){
flag1 = 1;
p1 = i;
q1 = j;
```

```
min1=graph[i][j];
}else if(flag1 == 1 && graph[i][j]< min1){
p1 = i;
q1 = j;
min1 = graph[i][j];
}
}
}
}
visit[q1]=1;
flag1=0;
printf("\n%d-(%d - weight of the rib)-%d", arr[p1],
graph[p1][q1], arr[q1]);
qq++;
wght = wght+graph[p1][q1];
}
while(qq < size-2);
printf("\n%d\n", wght);
return 0;
}
```

Результати програми:

```
Ribs included i the minimum span tree
2-(1 - weight of the rib)-7
2-(2 - weight of the rib)-5
5-(1 - weight of the rib)-3
7-(2 - weight of the rib)-4
4-(2 - weight of the rib)-6
3-(3 - weight of the rib)-1
7-(3 - weight of the rib)-9
7-(3 - weight of the rib)-10
5-(4 - weight of the rib)-8
8-(4 - weight of the rib)-11
24

Process returned 0 (0x0)   execution time : 0.009 s
Press any key to continue.
```

Висновки: на цій лабораторній роботі ми набули практичних вмінь та навичок з використання алгоритмів Прима і Краскала.