МІНІСТЕРСТВО ОСВІТИ І НАУКИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра систем штучного інтелекту

Лабораторна робота №3

З курсу “Дискретна математика ”

Виконав:
ст.гр. КН-110
Синюк Павло

Викладач:
Мельникова Н. І.

Львів – 2018

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.

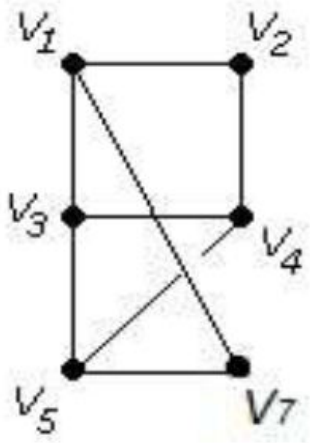
Граф, який не має ребер називається *пустим графом*, *нуль-графом*. Вершина графа, яка не інцедентна до жодного ребра, називається *ізольованою*. Вершина графа, яка інцедентна тільки до одного ребра, називається *звисаючою*.

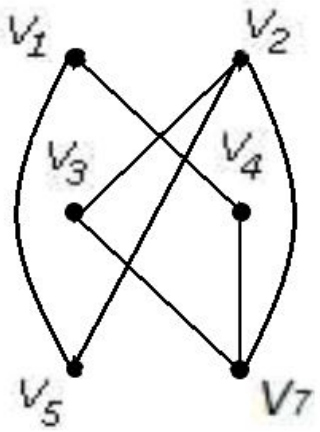
Варіант №10

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

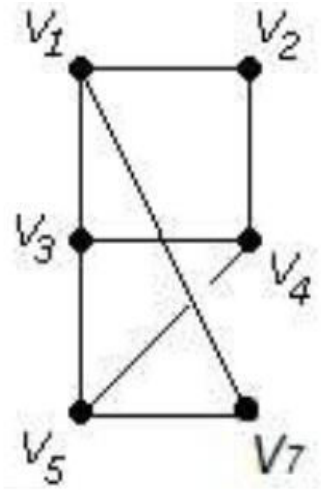
1. Виконати наступні операції над графами:

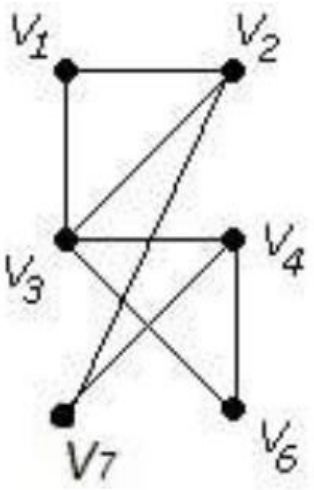
1) знайти доповнення до першого графу,

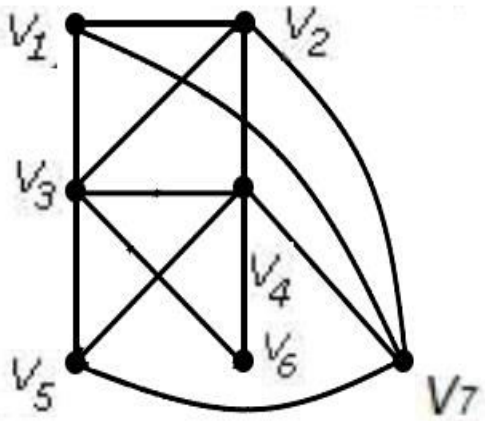




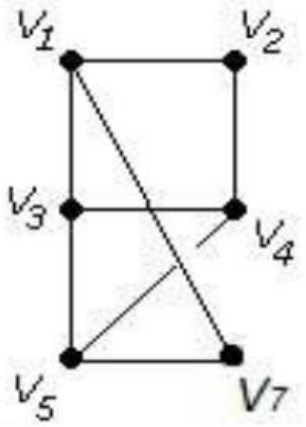
2) об’єднання графів,

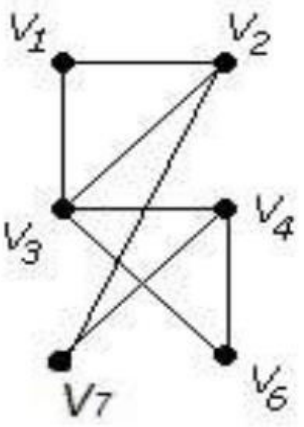


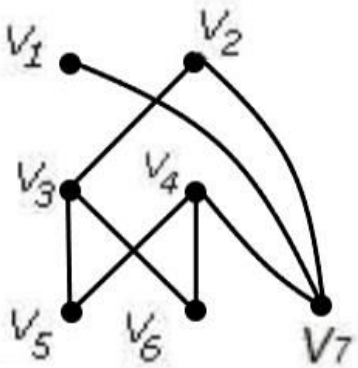




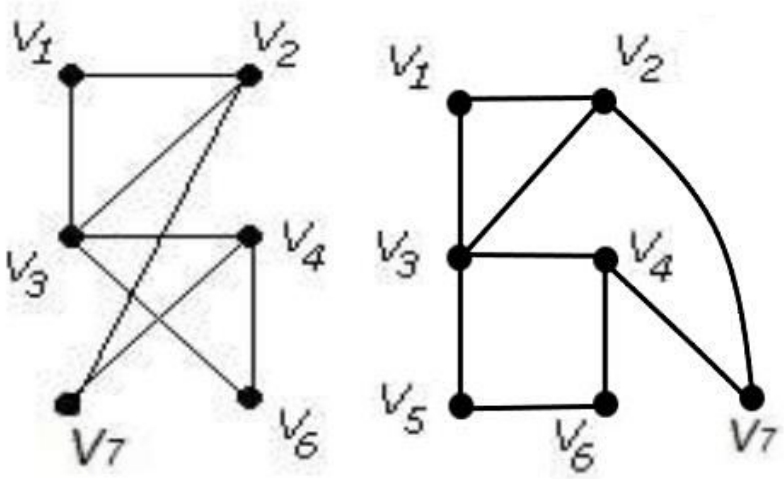
3) кільцеву суму G1 та G2 (G1+G2),



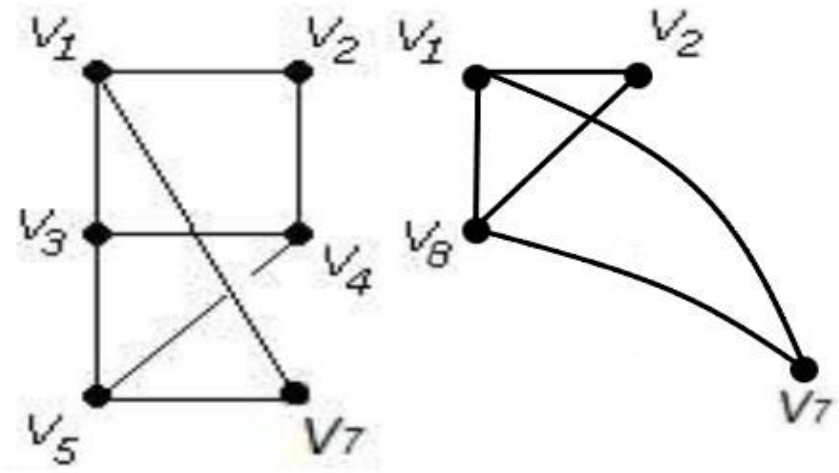




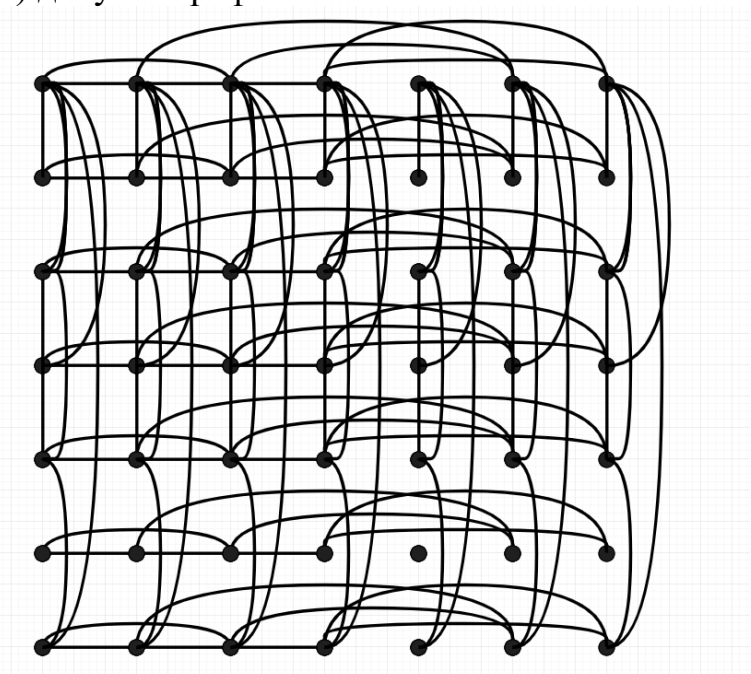
4) розщепити вершину у другому графі,



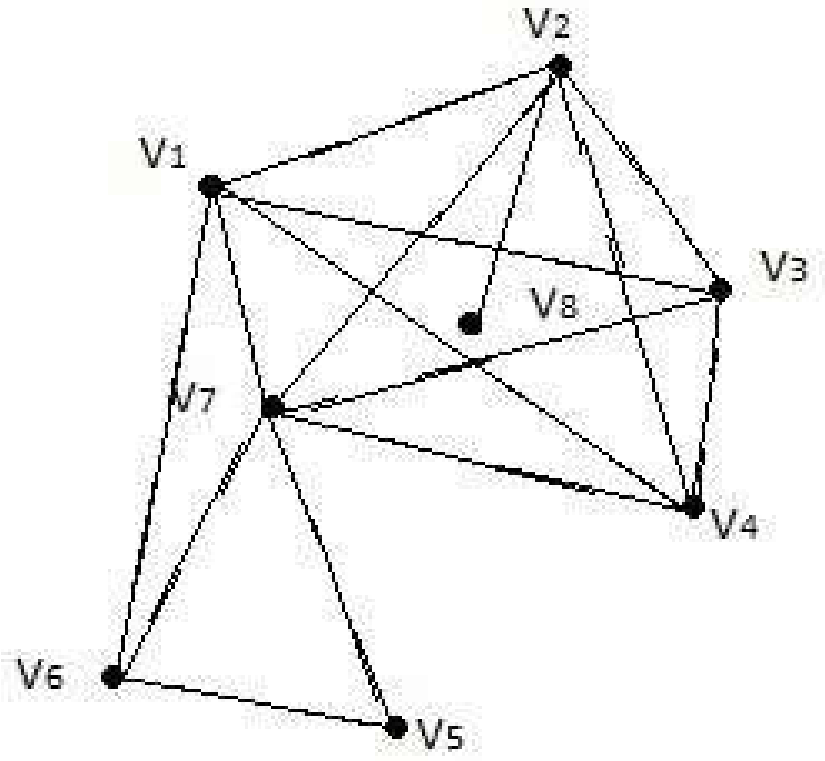
5) виділити підграф А, що складається з 3-х вершин в G1 і знайти стягнення А в G1 (G1\ A),



6) добуток графів.



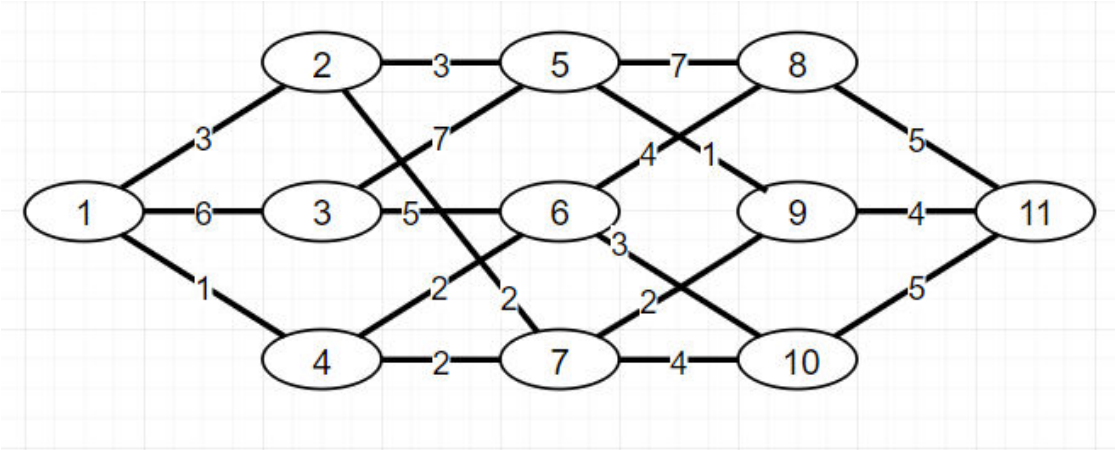
2. Знайти таблицю суміжності та діаметр графа.



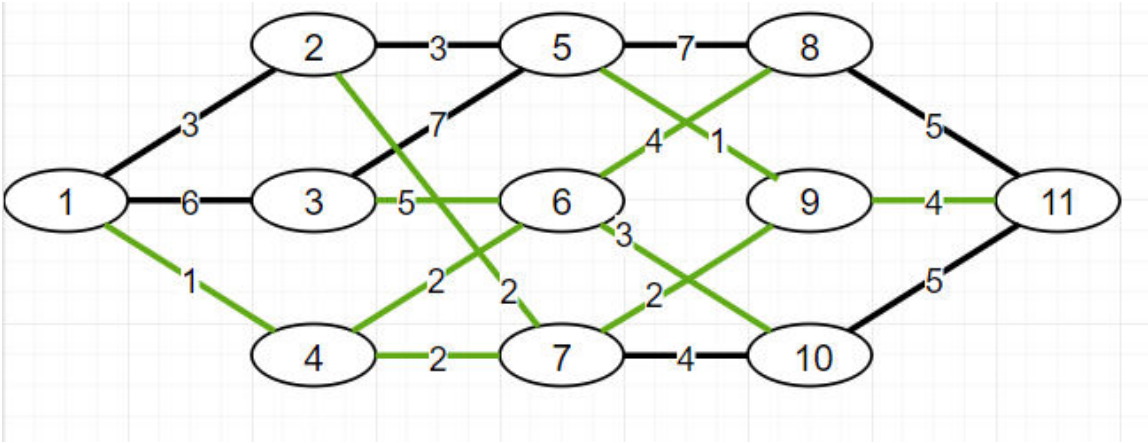
Таблиця суміжності для даного графа виглядатиме так:

	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8
V_1	0	1	1	1	0	1	1	0
V_2	1	0	1	1	0	0	1	1
V_3	1	1	0	1	0	0	1	0
V_4	1	1	1	0	0	0	1	0
V_5	0	0	0	0	0	1	1	0
V_6	1	0	0	0	1	0	1	0
V_7	1	1	1	1	1	1	0	0
V_8	0	1	1	0	1	0	0	0

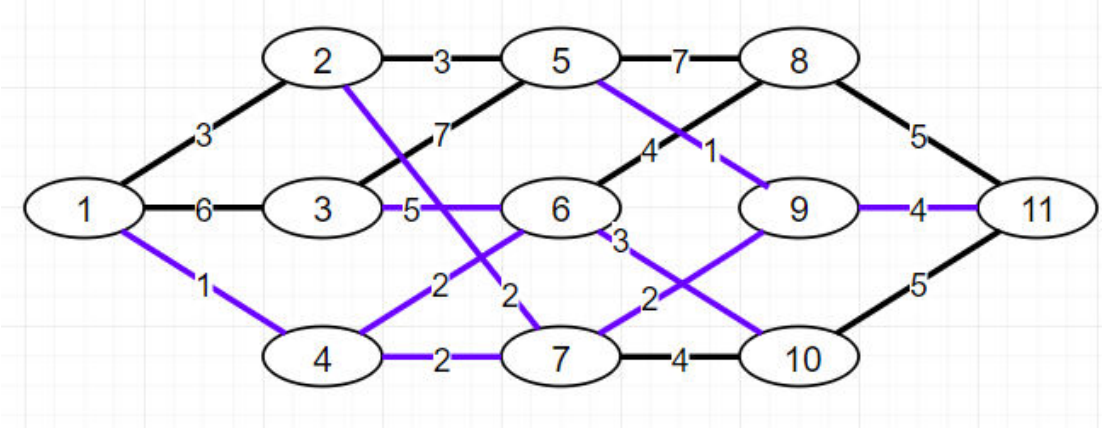
3. Знайти двома методами (Краскала і Прима) мінімальне остове дерево графа.



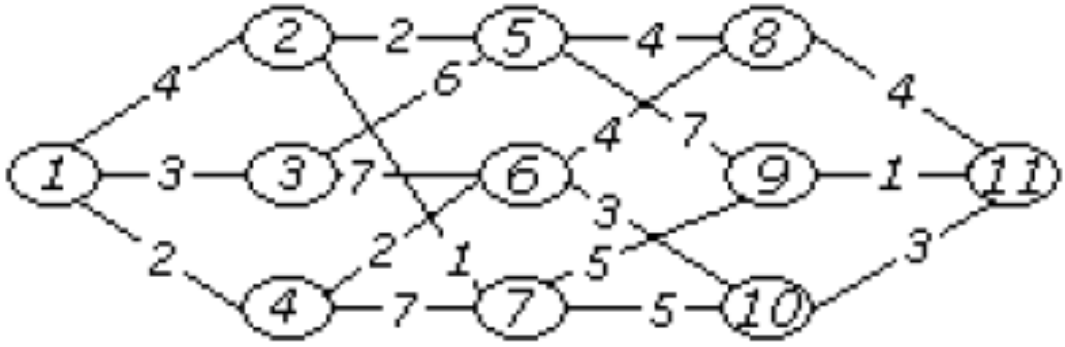
Його остове дерево за методом Крускала виглядатиме так:



Його остове дерево за методом Прима виглядатиме так:



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



Код програми:

```
#include <stdio.h>
#include <cs50.h>
#include <stdlib.h>



int FIND(int n, int array[n][n]);
int CHECK(int n, int array[n][n], int first, int second);
void REMOVE(int n, int array[n][n]);
void ADD(int n, int array[n][n], int first, int second);



int main()
{
   int array[11][11] = {
        { 0, 4, 3, 2, 0, 0, 0, 0, 0, 0, 0 },
        { 4, 0, 0, 0, 2, 0, 1, 0, 0, 0, 0 },
        { 3, 0, 0, 0, 6, 7, 0, 0, 0, 0, 0 },
        { 2, 0, 0, 0, 0, 2, 7, 0, 0, 0, 0 },
        { 0, 2, 6, 0, 0, 0, 0, 4, 7, 0, 0 },
        { 0, 0, 7, 2, 0, 0, 0, 6, 0, 3, 0 },
        { 0, 1, 0, 7, 0, 0, 0, 0, 5, 5, 0 },
        { 0, 0, 0, 0, 4, 4, 0, 0, 0, 0, 4 },
        { 0, 0, 0, 0, 7, 0, 5, 0, 0, 0, 1 },
        { 0, 0, 0, 0, 0, 3, 5, 0, 0, 0, 3 },
        { 0, 0, 0, 0, 0, 0, 0, 4, 1, 3, 0 }
   };
```

```
    REMOVE(11, array);

    printf("Sorting ribs by weight:");

    for (int i = 1; i <= 7; i++){
        printf("\n%d: ", i);
        for (int j = 1; j <= 11; j++){
            for (int k = 1; k <= 11; k++){
                if (array[j - 1][k - 1] == i && array[j-1][k-1] ){
                    printf("%d->%d; ", j, k);
                }
            }
        }
    }

    int array1[11][11];
    FIND(11, array1);

    printf("\nWay: ");

    for (int i = 1; i <= 7; i++){
        for (int j = 1; j <= 11; j++){
            for (int k = 1; k <= 11; k++){
                if (array[j - 1][k - 1] == i && CHECK(11, array1, j, k)){
                    ADD(11, array1, j, k);
                    printf("%d->%d; ", j, k);
                }
            }
        }
    }
    printf("\n\n");

    return 0;
}

int FIND(int n, int array[n][n]){
    for (int i = 0; i < n; i++){
        for (int j = 0; j < n; j++){
            array[i][j] = 0;
        }
    }
    for (int i = 0; i < n; i++){
        array[i][i] = i + 1;
    }
```

```
    return array[n][n];
}



void REMOVE(int n, int array[n][n]){
    for (int i = 0; i < n; i++){
        for (int j = 0; j < n; j++){
            if (j < i){
                array[i][j] = 0;
            }
        }
    }
}



void ADD(int n, int array[n][n], int first, int second){
    int text;
    for (int i = 0; i < n; i++){
        for (int j = 0; j < n; j++){
            if (array[i][j] == second){
                text = i;
            }
        }
    }

    for (int i = 0; i < n; i++){
        for (int j = 0; j < n; j++){
            if (array[i][j] == first){
                for (int k = 0; k < n; k++){
                    if (array[text][k]){
                        array[i][k] = array[text][k];
                        array[text][k] = 0;
                    }
                }
            }
        }
    }
}
```

```
int CHECK(int n, int array[n][n], int first, int second){

    int temp1, temp2;

    for (int i = 0; i < n; i++){
        temp1 = 0;
        temp2 = 0;
        for (int j = 0; j < n; j++){
            if (array[i][j] == first){
                temp1 = 1;
            }
        }

        for (int k = 0; k < n; k++){
            if (array[i][k] == second){
                temp2 = 1;
            }
        }

        if (temp1 == 1 && temp2 == 1){
            return 0;
        }
    }

    return 1;
}
```