МІНІСТЕРСТВО ОСВІТИ І НАУКИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра систем штучного інтелекту



Лабораторна робота №4

З курсу «Дискретна математика»

Виконав:  
ст. гр. КН-110

Марій Павло

Львів – 2018

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

**Мета:** Набуття практичних вмінь та навичок з використання алгоритмів Прима та Краскала.

**Варіант 2**

**Практична частина.**

1. Виконати наступні операції над графами:

1) Знайти доповнення до першого графу.

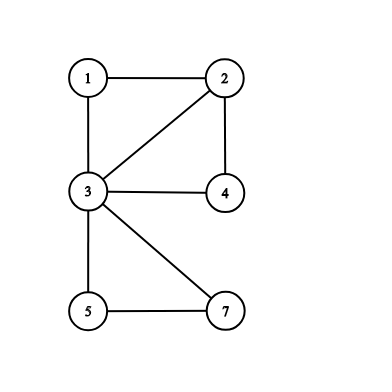
2) Об’єднання графів.

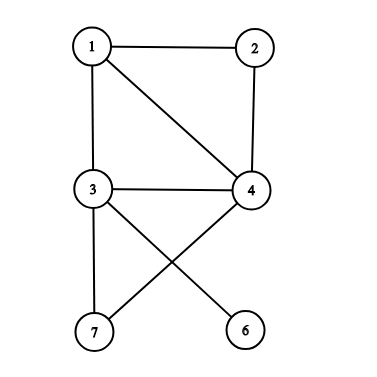
3) Кільцеву суму G1 та G2 (G1+G2).

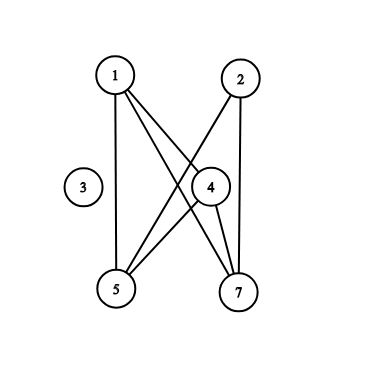
4) Розщепити вершину у другому графі.

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

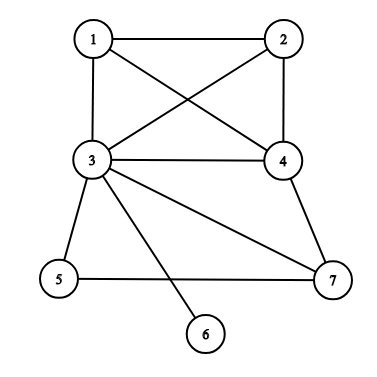
6) Добуток графів.



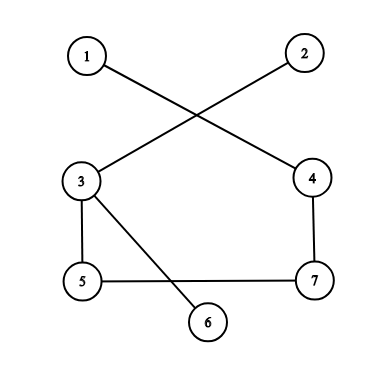
  
1) Доповнення до 1 графу



2) Об’єднання графів.

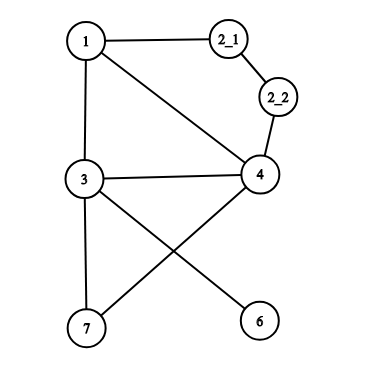


3) Кільцева суму G1 та G2 (G1+G2).



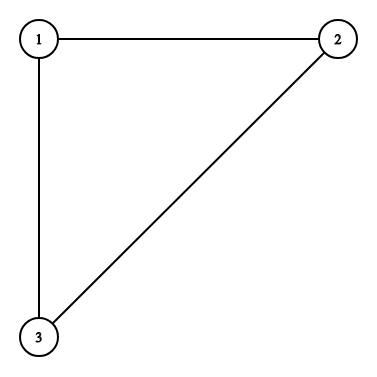
4) Розщеплення вершини у другому графі.

Розщепимо другу вершину:

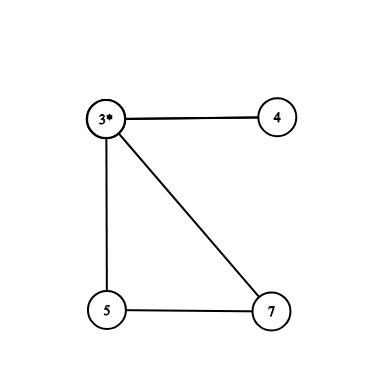


5) Виділяємо підграф A, що складається з 3-х вершин в G1 і знаходимо стягнення A в G1 (G1\A).

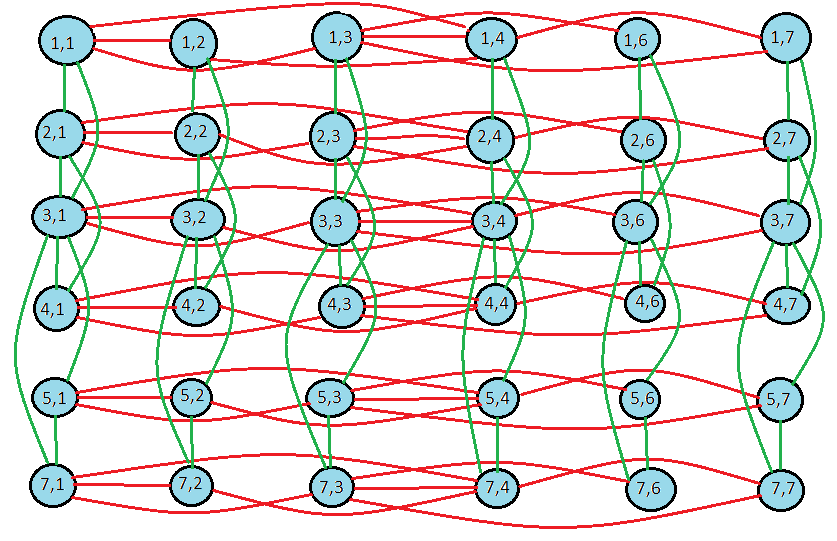
Підграф А:



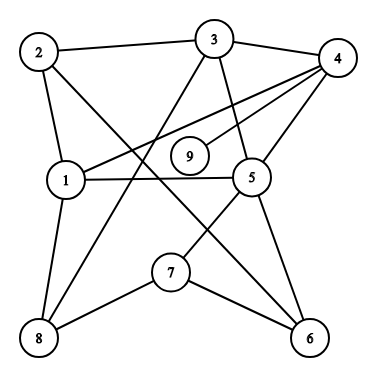
Стягнення:



6) Добуток графів.



2. Знайти таблицю суміжності та діаметр графа.

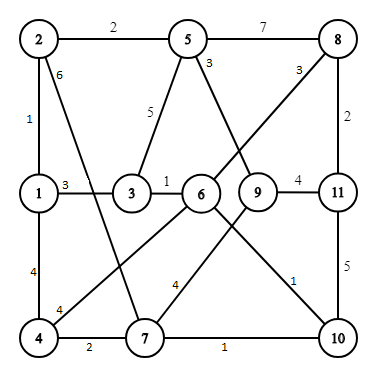


Таблиця суміжності

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 2 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 3 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 4 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 5 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 6 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 8 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 9 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

Діаметр – максимально можлива довжина між двома вершинами графа = 3.

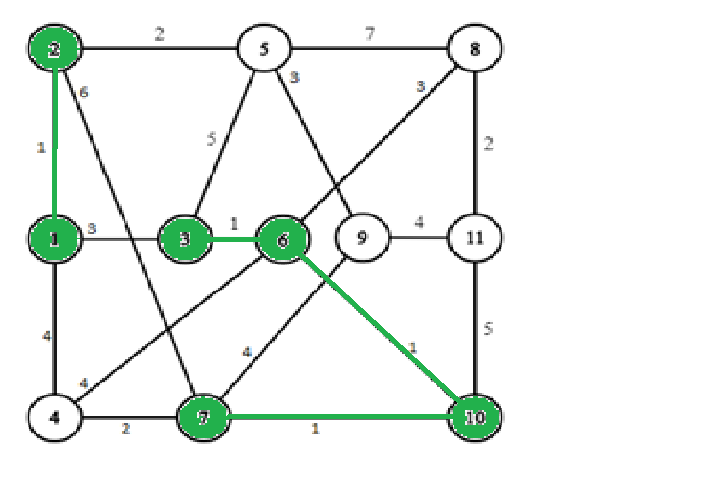
3. Знайти двома методами (Краскала і Прима) мінімальне остове дерево графа.



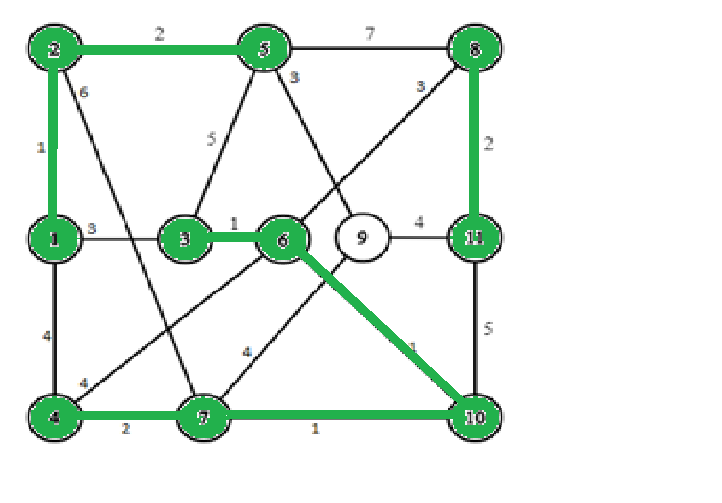
**Алгоритм Краскала**:

1) Беремо ребра найменшої вартості: (1-2) = 1; (3-6) = 1; (7-10) = 1;

(6-10) = 1; і перевіряємо, чи не утворюють вони циклу.



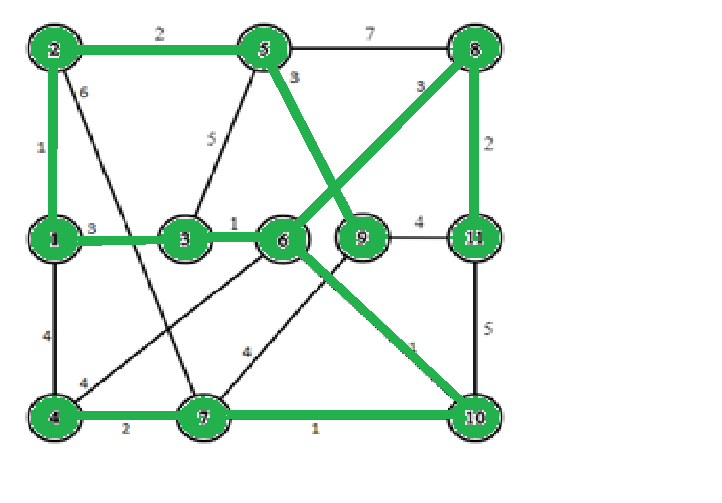
2) Беремо наступні ребра найменшої вартості і при включенні їх в остове дерево перевіряємо, чи не утворюється цикл. (2-5) = 2; (8-11) = 2; (4-7) = 2;



Цикл не утворено.

3) Беремо ребра з вагою 3 і перевіряємо на утворення циклу. (5-9) = 3;

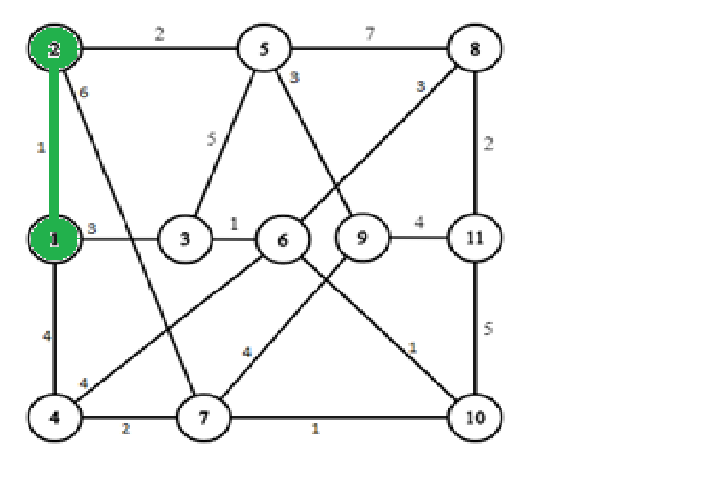
(1-3) = 3; (6-8) = 3;



Циклу немає, всі вершини включені в остове дерево. Кістякове дерево побудоване.

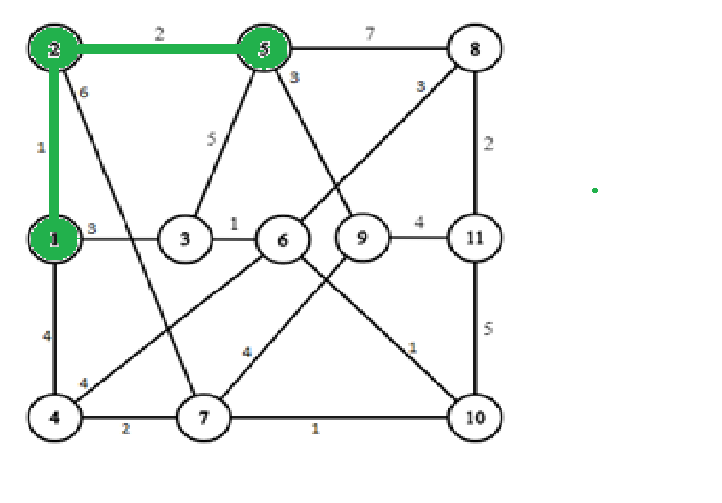
**Алгоритм Прима**:

1) Беремо вершину №1 та шукаємо найменше інцидентне ребро. Це (1-2) = 1.



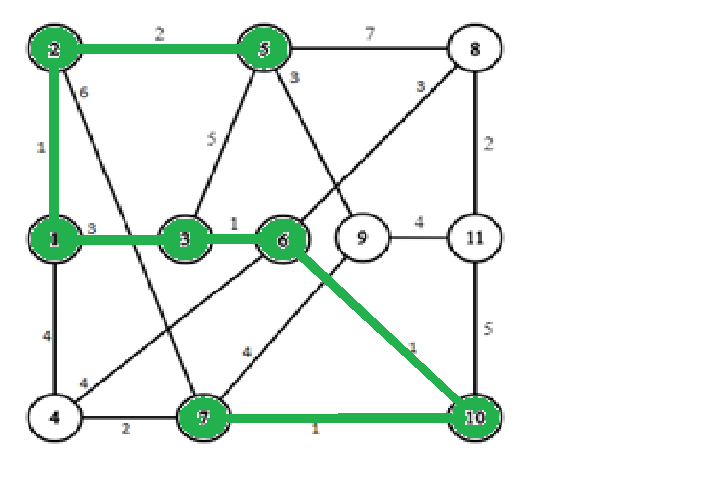
2) Перебираємо всі ребра, інцидентні вершинам, що включені в кістякове дерево, і шукаємо найкоротше, при цьому перевіряємо на ациклічність.

(2-5) = 2;

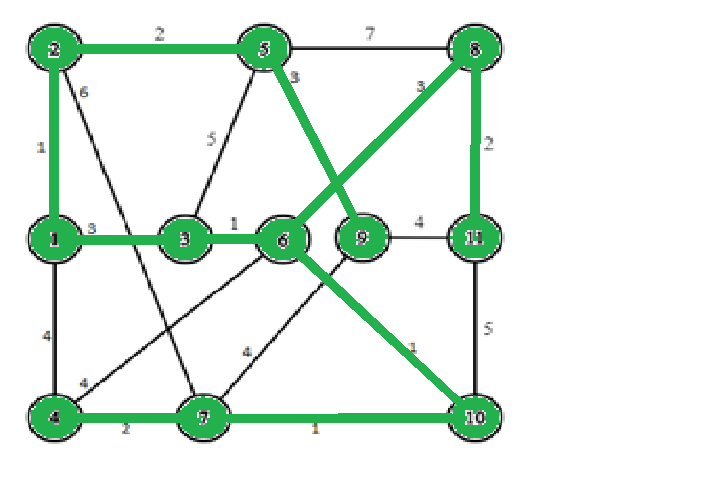


3) Повторюємо другий крок, допоки не буде побудовано кістякове дерево.

(1-3) = 3, (3-6) = 1, (6-10) = 1, (10-7) = 1;



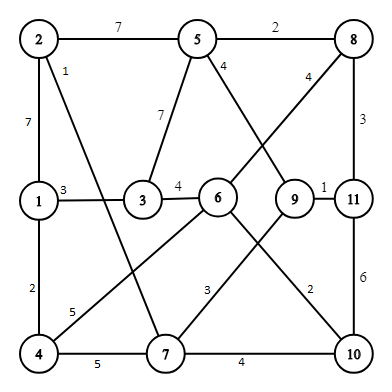
Продовжуємо пошук: (7-4) = 2; (5-9) = 3; (6-8) = 3; (8-11) = 2;



Кістякове дерево побудоване, співпадає з деревом, яке знайдено алгоритмом Краскала.

**Програмна частина**

Написати програму, яка реалізує алгоритм знаходження остового дерева мінімальної ваги за алгоритмом Краскала. Етапи розв’язання задачі виводити на екран. Протестувати розроблену програму на наступному графі:



**Код програми:**

#include <stdio.h>

int makeTrees(int n, int A[n][n]);

void removeRepeated(int n, int A[n][n]);

int areInDifferentTrees(int n, int A[n][n], int first, int second);

void addToTree(int n, int A[n][n], int first, int second);

int main(void)

{

int A[11][11] = {

{ 0, 7, 3, 2, 0, 0, 0, 0, 0, 0, 0 },

{ 7, 0, 0, 0, 7, 0, 1, 0, 0, 0, 0 },

{ 3, 0, 0, 0, 7, 4, 0, 0, 0, 0, 0 },

{ 2, 0, 0, 0, 0, 5, 5, 0, 0, 0, 0 },

{ 0, 7, 7, 0, 0, 0, 0, 2, 4, 0, 0 },

{ 0, 0, 4, 5, 0, 0, 0, 4, 0, 2, 0 },

{ 0, 1, 0, 5, 0, 0, 0, 0, 3, 4, 0 },

{ 0, 0, 0, 0, 2, 4, 0, 0, 0, 0, 3 },

{ 0, 0, 0, 0, 4, 0, 3, 0, 0, 0, 1 },

{ 0, 0, 0, 0, 0, 2, 4, 0, 0, 0, 6 },

{ 0, 0, 0, 0, 0, 0, 0, 3, 1, 6, 0 }

};

removeRepeated(11, A);

printf("\nVerticles sorted 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(A[j-1][k-1]==i)

{

printf("%d<->%d; ", j, k);

}

}

}

}

int B[11][11];

makeTrees(11, B);

printf("\n\nEdges : ");

for(int i=1; i<=7; i++)

{

for(int j=1; j<=11; j++)

{

for(int k=1; k<=11; k++)

{

if(A[j-1][k-1]==i && areInDifferentTrees(11, B, j, k))

{

addToTree(11, B, j, k);

printf("%d-%d; ", j, k);

}

}

}

}

printf("\n\n");

return 0;

}

int makeTrees(int n, int A[n][n])

{

for(int i=0; i<n; i++)

{

for(int j=0; j<n; j++)

{

A[i][j]=0;

}

}

for (int i=0; i<n; i++)

{

A[i][i]=i+1;

}

return A[n][n];

}

void removeRepeated(int n, int A[n][n])

{

for(int i=0; i<n; i++)

{

for(int j=0; j<n; j++)

{

if(j<i)

{

A[i][j]=0;

}

}

}

}

int areInDifferentTrees(int n, int A[n][n], int first, int second)

{

int temp1;

int temp2;

for(int i=0; i<n; i++)

{

temp1 = 0;

temp2 = 0;

for(int j=0; j<n; j++)

{

if(A[i][j]==first)

{

temp1=1;

}

}

for(int k=0; k<n; k++)

{

if(A[i][k]==second)

{

temp2=1;

}

}

if(temp1 && temp2)

{

return 0;

}

}

return 1;

}

void addToTree(int n, int A[n][n], int first, int second)

{

int temp;

for(int i=0; i<n; i++)

{

for(int j=0; j<n; j++)

{

if(A[i][j]==second)

{

temp=i;

}

}

}

for(int i=0; i<n; i++)

{

for(int j=0; j<n; j++)

{

if(A[i][j]==first)

{

for(int k=0; k<n; k++)

{

if(A[temp][k])

{

A[i][k]=A[temp][k];

A[temp][k]=0;

}

}

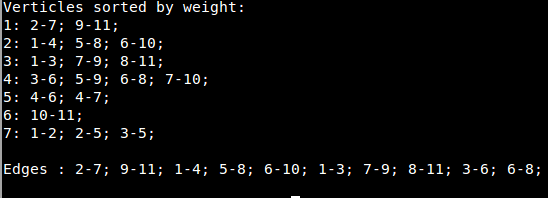
}

}

}

}

**Результат:**

****

**Висновок:** Я набув практичних вмінь та навичок з використання алгоритмів Прима та Краскала.