МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТУ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра систем штучного інтелекту

Звіт

Лабораторна робота №4

з дисципліни
«Дискретна математика»

Виконала:
Студентка групи КН-113
Білинська Віолетта
Викладач:
Мельникова Н.І

Львів-2019р.

ТЕМА РОБОТИ

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

Граф - це сукупність точок (вершин) і ліній (ребер), що їх з'єднують.
Інцидентним певним вершинам називають таке ребро, яке з'єднує ці певні дві вершини, які називаються суміжними.
Нуль-графом називають граф, який складається лише із ізольованих вершин.
Порожнім є граф, який не має жодних ребер.
Дводольним називається граф, множина вершин якого може бути розбита на дві підмножини так, що кожне ребро графа має одну вершину з першої підмножини і одну з другої.
Повним графом називають граф, граф, у якому будь-яка пара вершин з'єднана ребрами. Граф називатимемо зв'язним, якщо будь-яка його вершина зв'язна.
Мультиграф - це граф, який має кратні ребра. Псевдограф - це граф, який має петлі.
Простий граф - це граф, який не має кратних ребер та петель.
Степінь вершини - це число ребер, яким належить ця вершина. Якщо степінь вершини дорівнює 1, то вершина називається кінцевою вершиною графа. Якщо степінь вершини дорівнює 0, то вершини називається ізольованою.
Цикл - це шлях, в якому збігаються початкова та кінцева вершини. Якщо цикл через кожну вершину проходить лише один раз, то такий шлях називається простим.
Довжиною шляху (циклу) називається кількість ребер цього шляху (циклу).
Операції над графами поділяються на бінарні та унарні.
До бінарних належать такі операції як об'єднання, перетин, кільцева сума, сполучення, добуток.
До унарних операцій належать доповнення графа, видалення вершини, видалення графа, стягування вершин, замикання вершин, розмноження вершини, дублювання вершини, розмноження ребра.
Існують такі матриці представлення графа: матриця суміжності і матриця інцидентності. Матриця суміжності - це квадратна матриця Мij, рядкам і стовпцям якої відповідають вершини графа, а відношення інцидентності визначають матрицею Мij, що має m рядків і n стовпців. Стовпці матриці відповідають ребрам, рядки — вершинам.
Кістякове (остове) дерево зв'язаного неорієнтованого графа — ациклічний зв'язний підграф цього графа, який містить всі його вершини.

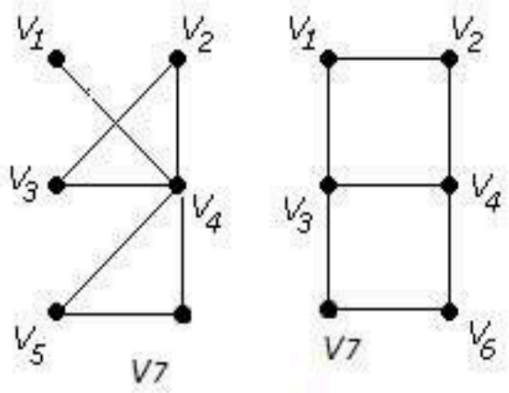
ЗАГАЛЬНЕ ЗАВДАННЯ

Варіант №3.

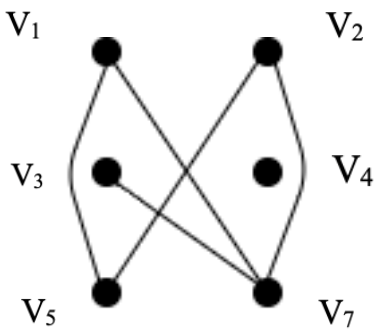
Завдання № 1.

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.

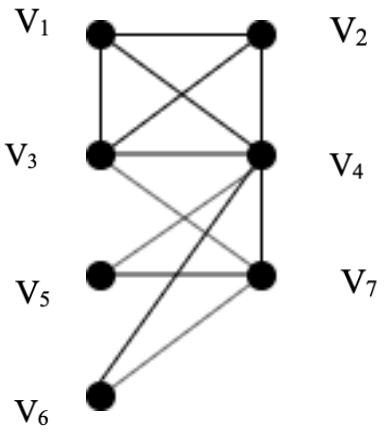
3



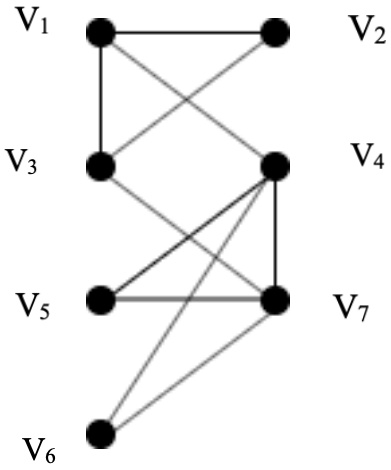
1) Доповнення до першого графу:



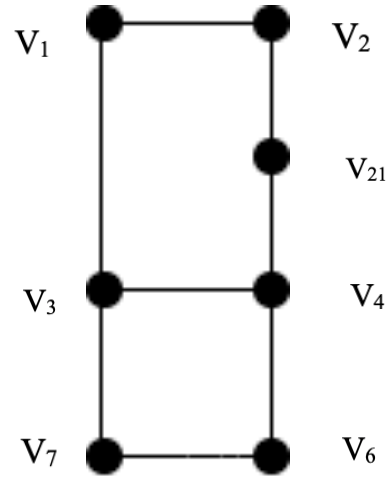
2) Об’єднання графів:



3) Кільцева сума графів:

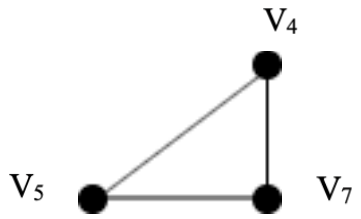


4) Розщепити вершину у другому графі:

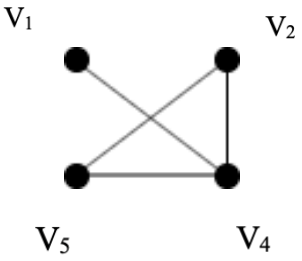


5) Виділити підграф A, що складається з 3-х вершин в G1 і знайти стягнення A в G1 (G1\ A):

Спочтаку виділимо підграф A, який складається з вершин V_4 , V_5 , V_7



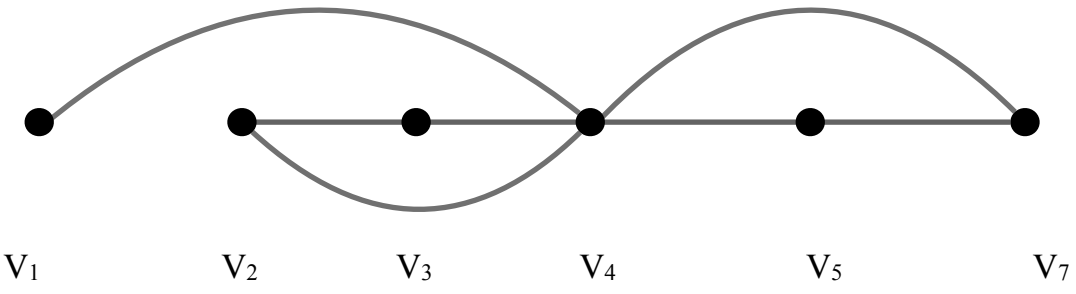
Отже, стягуємо ребро (V_4 , V_5) та ребро (V_4, V_7) .



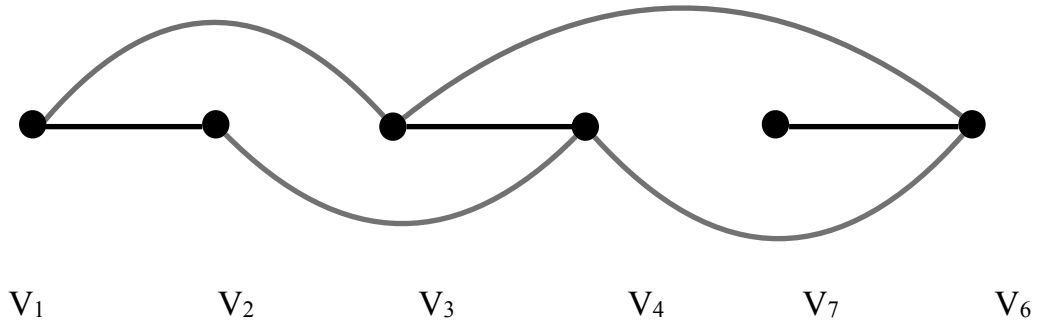
6) Добуток графів:

Добуток матиме 36 вершин, оскільки кожен з графів містить по 6 вершин.

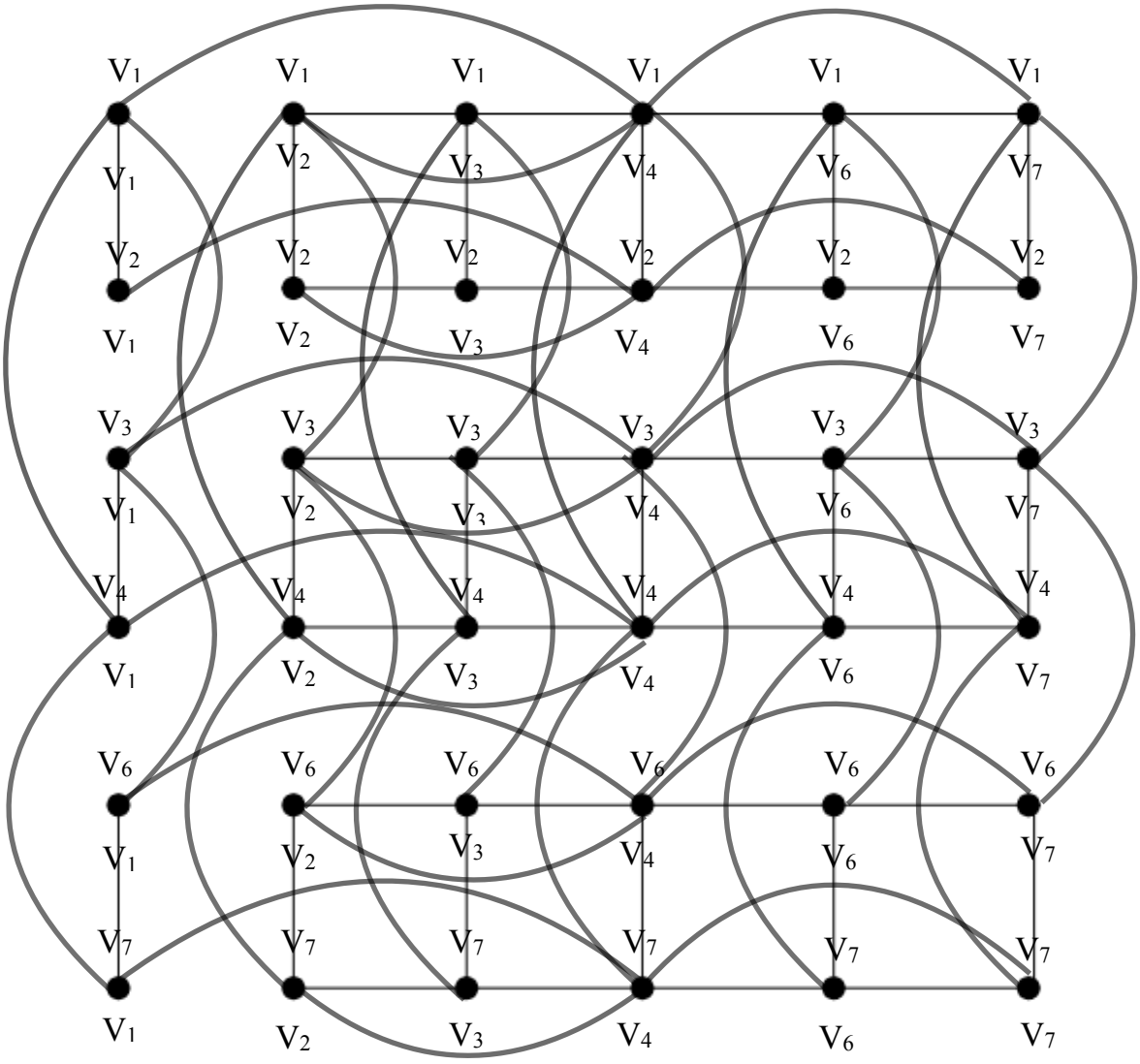
G1:



G2:



Отже, добуток графів G1 і G2 буде виглядати так:



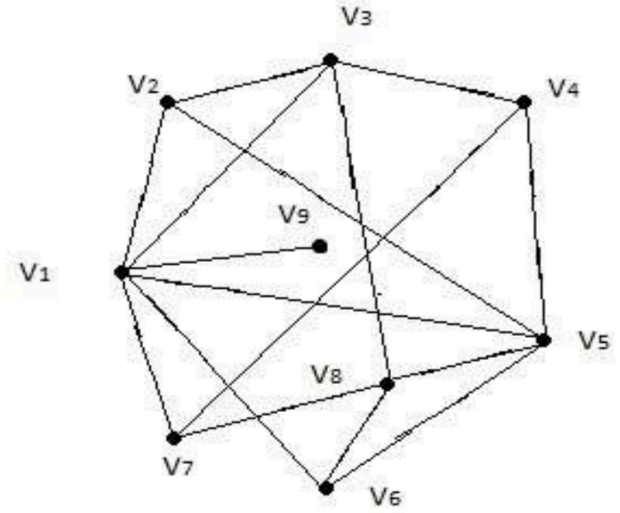
$G = G_1 \times G_2$

$V(G) = V(G_1) \times V(G_2)$
$E(G) = E(G_1 \times G_2)$

V(G) = { (1, 1), (1, 2), (1, 3), (1, 4), (1, 7), (1, 6), (2, 1), (2, 2), (2, 3), (2, 4), (2, 7), (2, 6), (3, 1), (3, 2), (3, 3), (3, 4), (3, 7), (3, 6), (4, 1), (4, 2), (4, 3), (4, 4), (4, 7), (4, 6), (5, 1), (5, 2), (5, 3), (5, 4), (5, 7), (5, 6), (6, 1), (6, 2), (6, 3), (6, 4), (6, 7), (6, 6), (7, 1), (7, 2), (7, 3), (7, 4), (7, 7), (7, 6) }

Завдання № 2.

Знайти матрицю суміжності та діаметр графа.



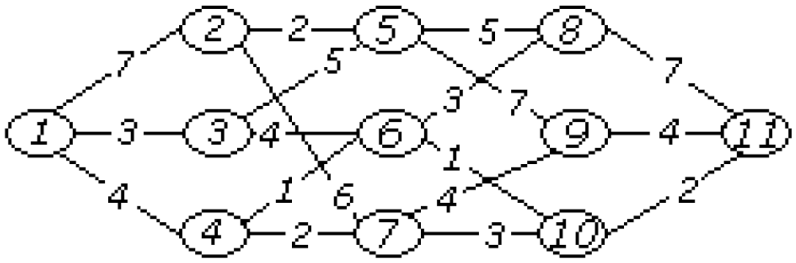
	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9
V_1	0	1	1	0	1	1	1	0	1
V_2	1	0	1	0	1	0	0	0	0
V_3	1	1	0	1	0	0	0	1	0
V_4	0	0	1	0	1	0	1	0	0
V_5	1	1	0	1	0	1	0	1	0
V_6	1	0	0	0	1	0	0	1	0
V_7	1	0	0	1	0	0	0	1	0
V_8	0	0	1	0	1	1	1	0	0
V_9	1	0	0	0	0	0	0	0	0

Діаметр графа дорівнює 3, оскільки цьому дорівнює максимальна відстань найкоротшого шлях між двома вершинами (у цьому випадку V_3 і V_9).

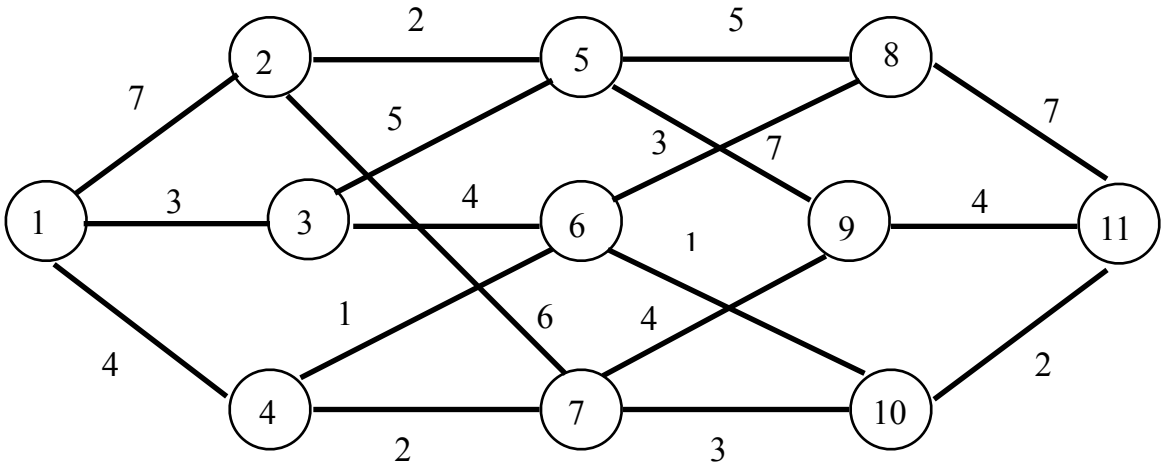
Завдання № 3.

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

3



- Метод Краскала:

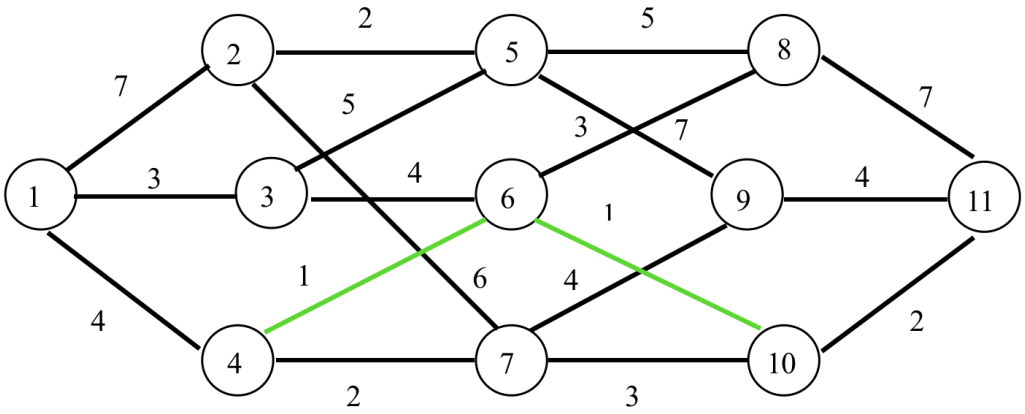


Етап 1: упорядковуємо для даного графа ребра у порядку неспадання ваг цих ребер.

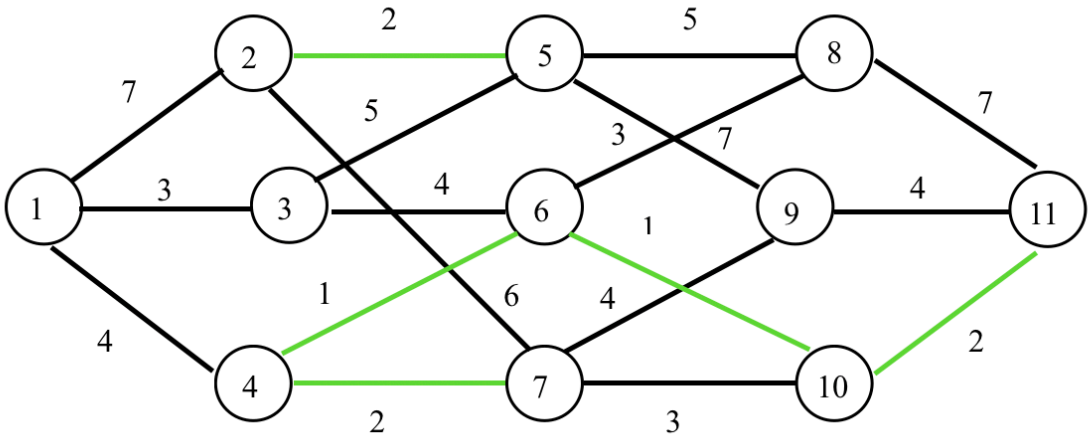
(4, 6) - 1;
(6, 10) - 1;
(2, 5) - 2;
(4, 7) - 2;
(10, 11) - 2;
(1, 3) - 3;
(6, 8) - 3;
(7, 10) - 3;
(1, 4) - 4;
(3, 6) - 4;
(7, 9) - 4;
(9, 11) - 4;
(3, 5) - 5;

(5, 8) - 5;
(2, 7) - 6;
(1, 2) - 7;
(5, 9) - 7;
(8, 11) - 7.

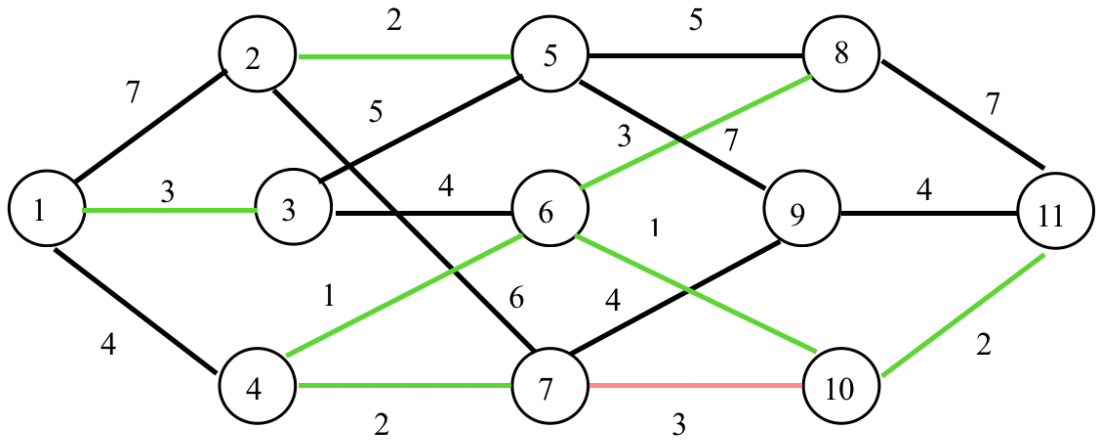
Етап 2: вибираємо ребро графа з найменшою вагою і додаємо до дерева. В даному випадку це ребра (4, 6) і (6, 10) з вагою 1.



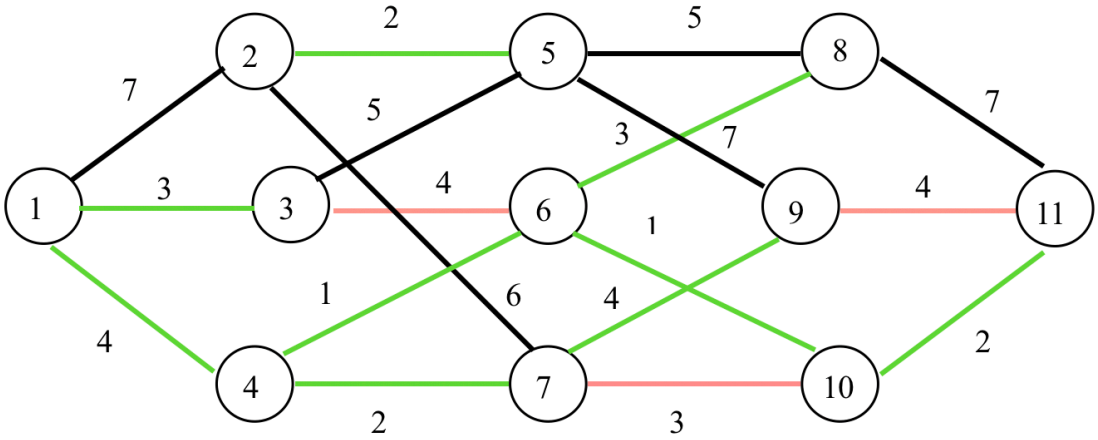
Етап 2:
беремо ребро з наступною найменшою вагою. Це ребра (2, 5), (4, 7), (10, 11) з вагою 2.



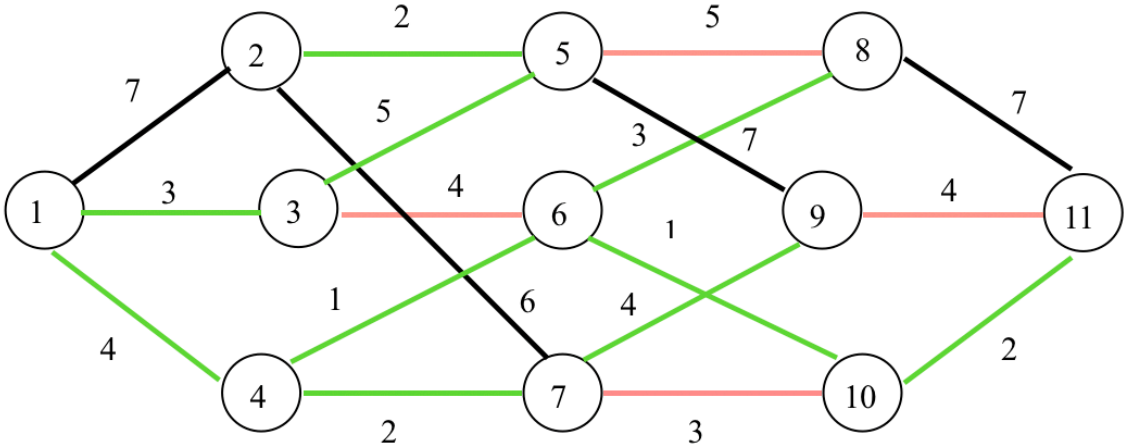
Етап 3: беремо ребра з наступною найменшою вагою - (1, 3), (6, 8), (7, 10) з вагою 2. Але оскільки ребро (7, 10) утворює цикл, то не додаємо його до дерева.



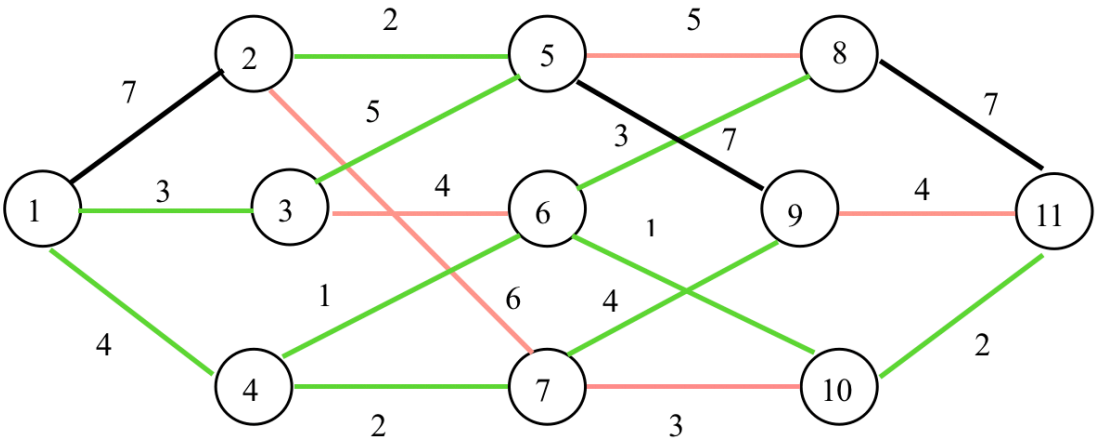
Етап 4: беремо ребра з наступною найменшою вагою - (1, 4), (3, 6), (7, 9), (9, 11) з вагою 4. Відкидаємо ребра (3, 6) і (9, 4), оскільки вони утворять цикли.



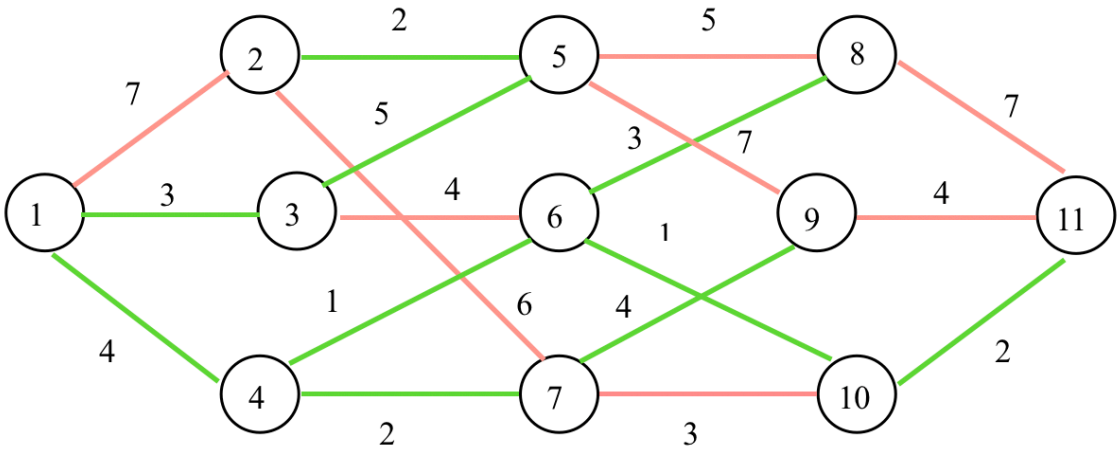
Етап 5: беремо ребра з наступною найменшою вагою - (3, 5) і (5, 8) з вагою 5. (5, 8) відкидаємо через утворення циклу.



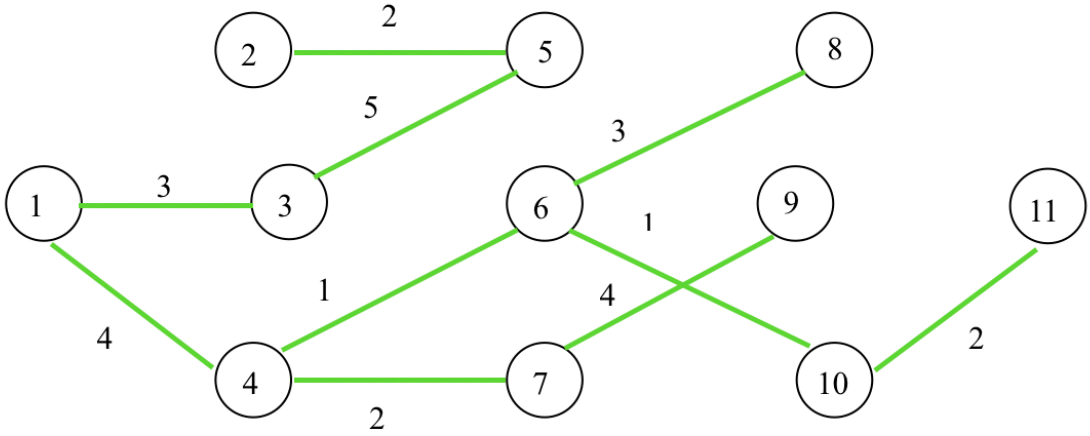
Етап 6: наступне ребро з найменшою вагою - (2, 7) з вагою 6. Не додаємо його, оскільки утворить цикл.



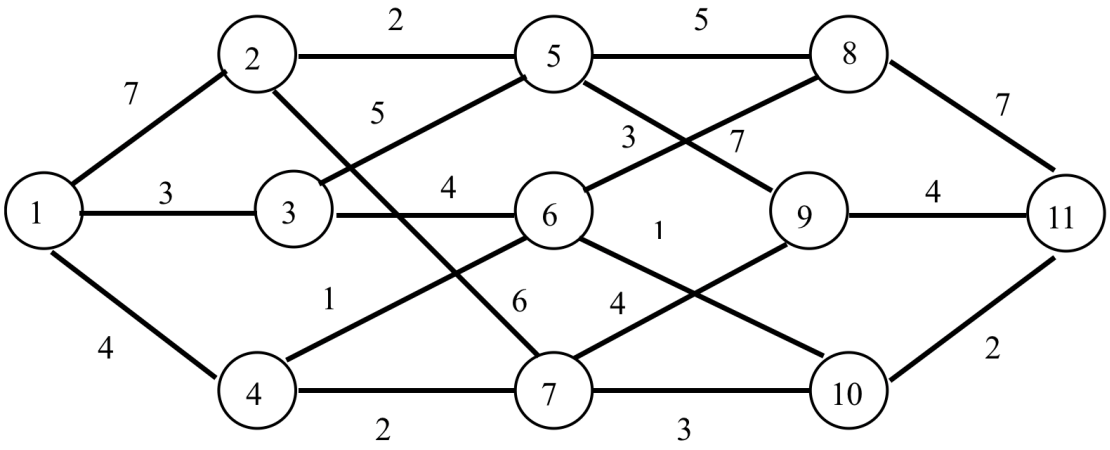
Етап 7: ребра з наступною найменшою вагою - (1, 2), (5, 9), (8, 11) з вагою 7. Відкидаємо їх усіх, оскільки вони утворюють цикли.



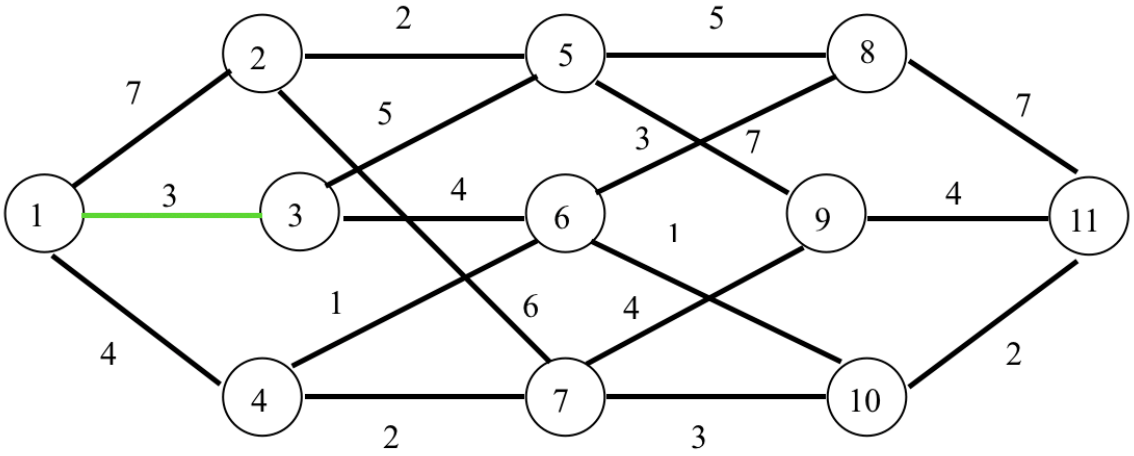
Отже, мінімальне остове дерево даного графа виглядатиме так:



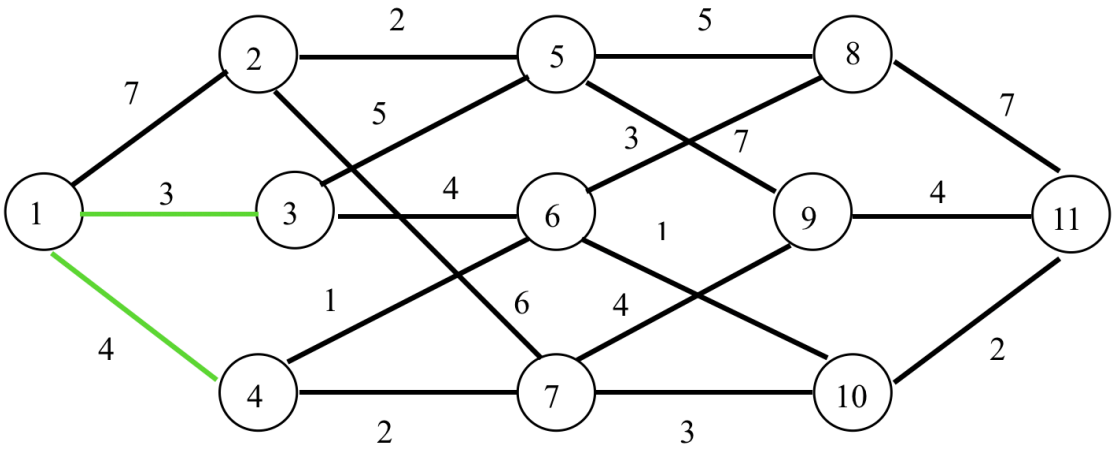
- Метод Прима



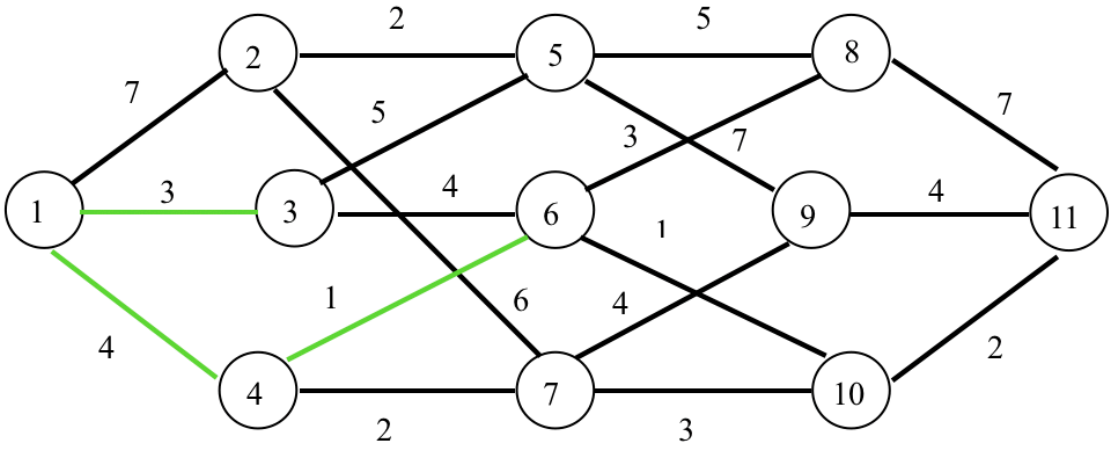
Етап 1: спочатку фіксуємо довільну вершину, з якої розпочнемо пошук мінімального дерева. У даному випадку це вершина 1. Додаємо до кістякового дерева ребро (1, 3).



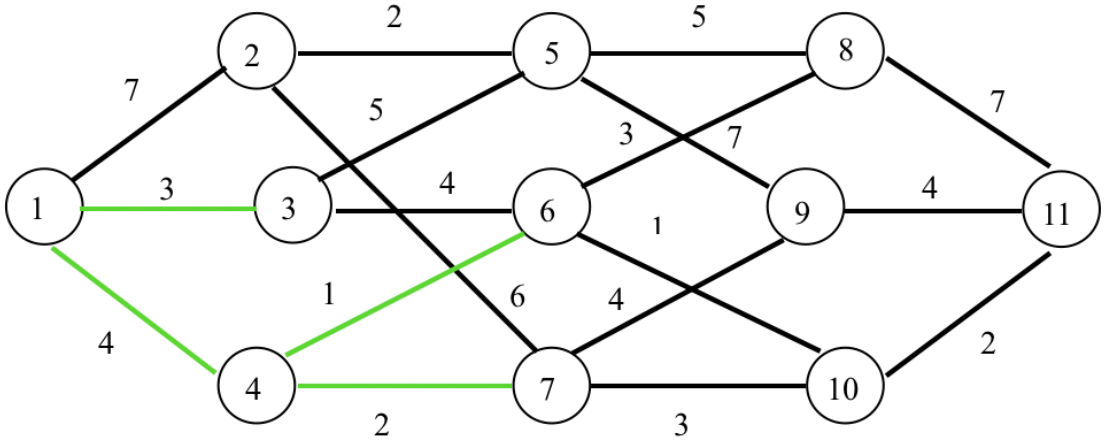
Етап 2: розглядаємо ребра, інцидентні вершинам 1 і 3, і вибираємо те, яке має найменшу вагу. І це (1, 4) з вагою 4. Додаємо його до кістякового дерева.



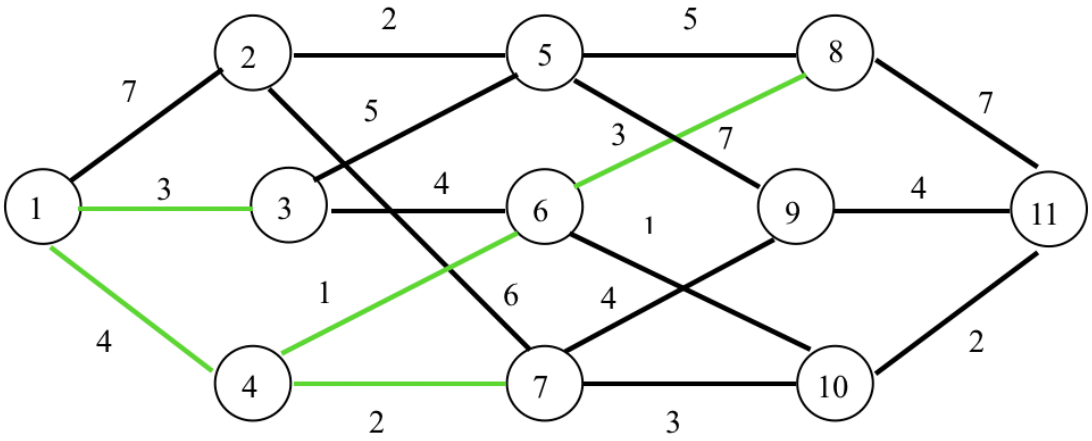
Етап 3: розглядаємо ребра, інцидентні вершинам 1, 3, 4 і вибираємо з найменшою вагою - і це (4, 6) з вагою 1.



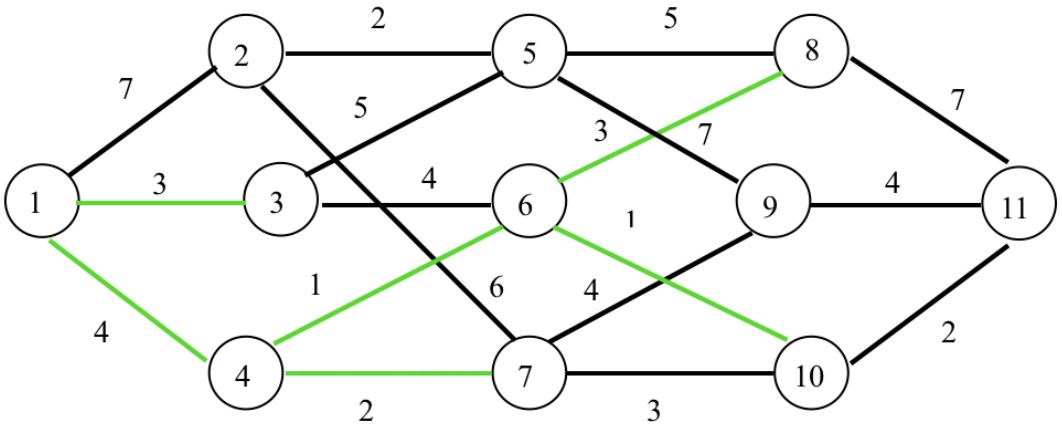
Етап 4: розглядаємо наступні ребра, інцидентні вершинам 1, 3, 4, 7 і вибираємо ребро з найменшою вагою - (4, 7) з вагою 2. Додаємо його до дерева.



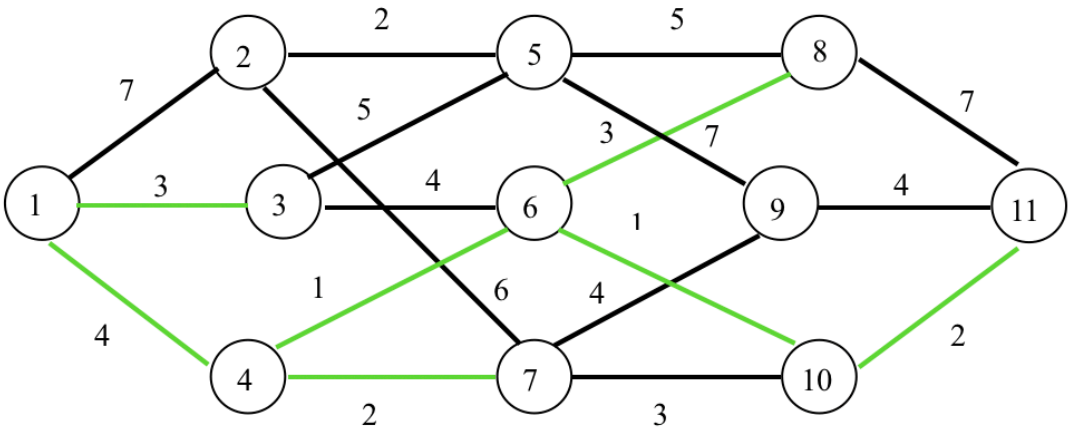
Етап 5: розглядаємо ребра, інцидентні вершинам 1, 3, 4, 6, 7 і вибираємо ребро (6, 8) з вагою 3.



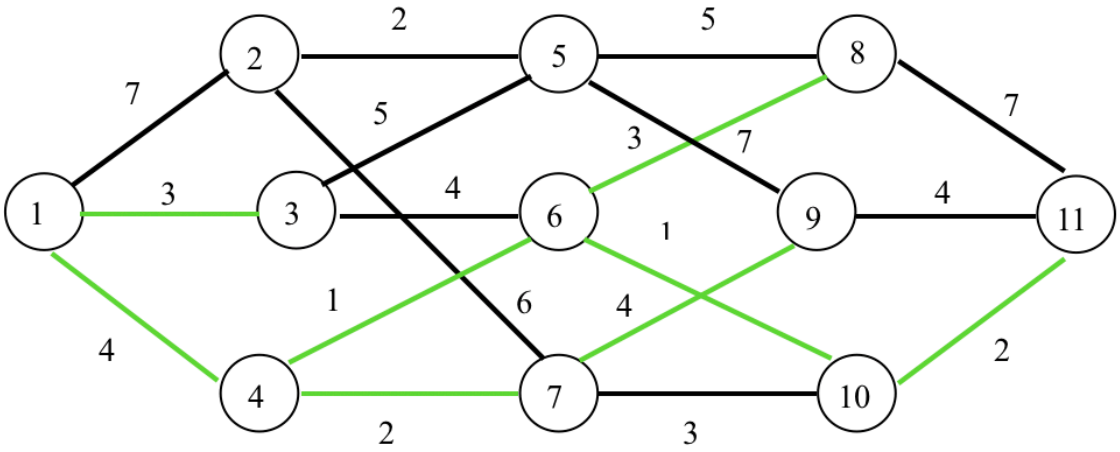
Етап 6: розглядаємо наступні вершини 1, 3, 4, 6, 7, 8 і вибираємо ребро (6, 10) з вагою 4.



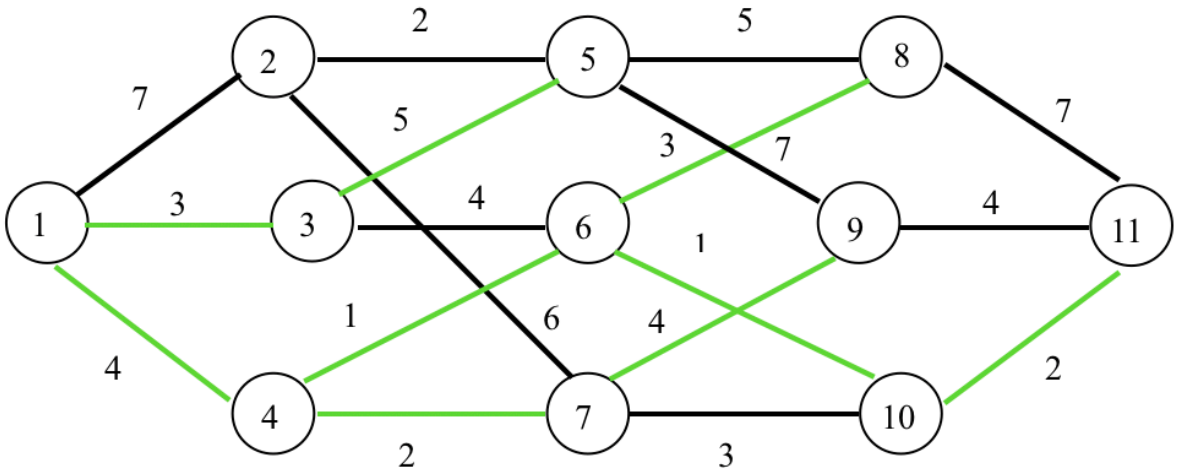
Етап 7: розглядаємо вершини 1, 3, 4, 6, 7, 8, 10 і вибираємо ребро (10, 11) з вагою 2.



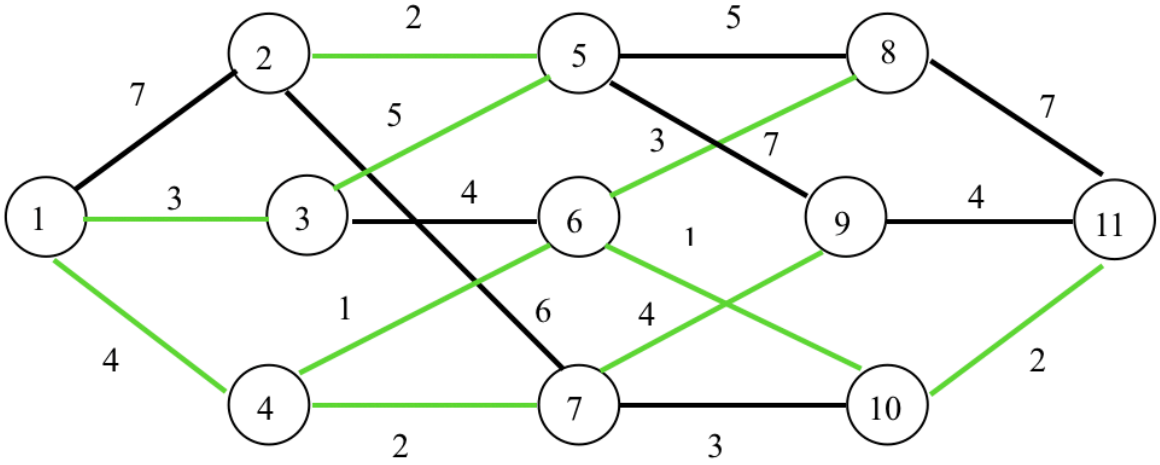
Етап 8: розглядаємо вершини 1, 3, 4, 6, 7, 8, 10 і вибираємо ребро (7, 9) з вагою 4.



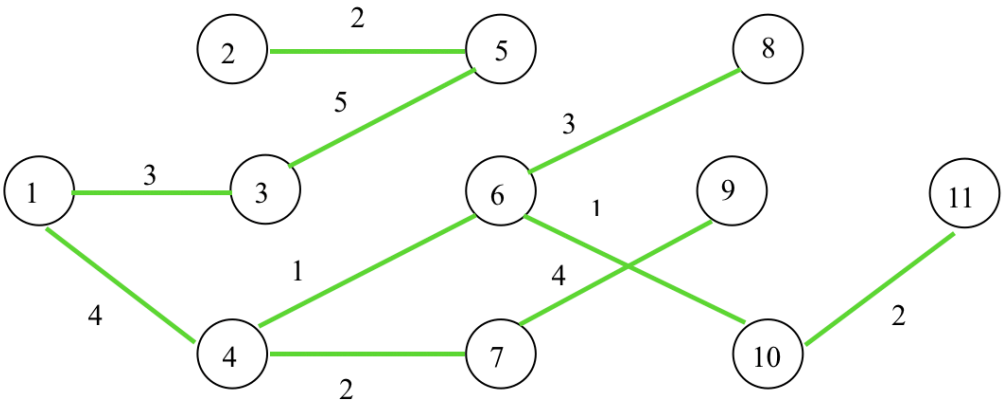
Етап 9: розглядаємо вершини 1, 3, 4, 6, 7, 8, 10 і вибираємо ребро (7, 3) з вагою 3, але оскільки він утворить цикл, то відкидаємо його і беремо наступне (5, 9) з вагою 3, але оскільки він утворить цикл, то відкидаємо і беремо наступне - (3, 6) з вагою 4, яке також утворить цикл, потім беремо (9, 11) з вагою 4, яке також утворить цикл, тому беремо ребро (3, 5) з вагою 5. Додаємо його.



Етап 10: розглядаємо вершини 1, 3, 4, 5, 6, 7, 8, 10 і вибираємо ребро (2, 5) з вагою 2.

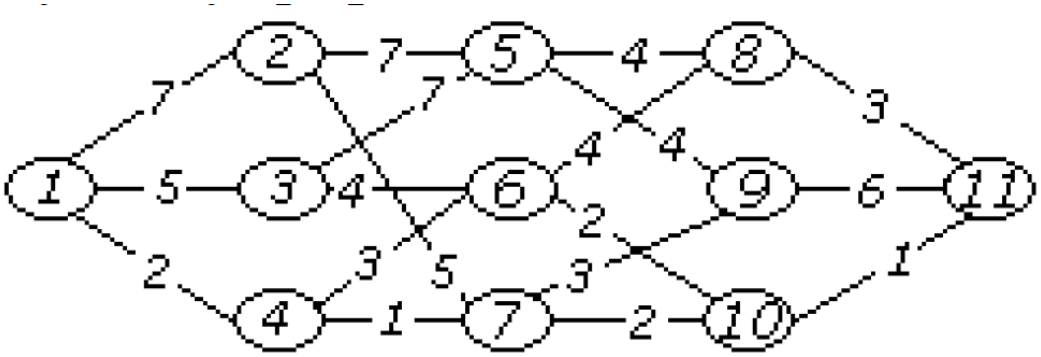


Отже, мінімальне остове дерево за алгоритмом Прима буде виглядати так:



Завдання № 4.

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



Програмна реалізація

```cpp
#include <iostream>

using namespace std;

int main()
{
    int x, y;
    int u, v;
    int number;
    int i, j;
    int ne = 1;
    int scan [10] = {0};
    int minimal, minimal1 = 0;
    int cost [10][10];
    int path [100] = {0};
    int path_index = 0;

    cout <<"Введіть кількість вершин:  ";
    cin >> number ;
    cout <<"Введіть матрицю суміжності \n";


    for(i = 1; i <= number ; i++)
        for(j = 1; j <= number ; j++)
        {
            cin >> cost[i][j];
            if(cost[i][j] == 0)
                cost[i][j] = 999;
        }
    scan [1] = 1;
    cout << endl;

    while(ne < number )
    {
        for (i = 1, minimal = 999; i <= number; i++)
            for (j = 1;j <= number; j++)
                if (cost [i][j] < minimal)
                    if (scan [i] != 0)
                    {
                        minimal = cost [i][j];
                        x = u = i;
                        y = v = j;
                    }
```

```cpp
        for (i = 1, minimal = 999; i <= number; i++)
            for (j = 1;j <= number; j++)
                if (cost [i][j] < minimal)
                    if (scan [i] != 0)
                    {
                        minimal = cost [i][j];
                        x = u = i;
                        y = v = j;
                    }
        if (scan [u] ==0 || scan [v] == 0)
        {
            path[path_index] = y;
            path_index++;
            cout << endl;
            cout << ne++ <<"  "<< x <<"  "<< y << minimal;

            minimal1 += minimal;
            scan [y] = 1;

        }
        cost[x][y] = cost[y][x] = 999;
    }


    cout << endl;
    cout << 1 << " -> ";

    for (int i = 0; i < number-1; i++)
    {
        cout << path[i];
        if (i < number-2) {
            cout << " -> ";
        }
    }
    cout << endl;
    cout << "Minimal cost: " << endl;
    cout << minimal1;
```