МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота №4

з дисципліни «Дискретна математика»

Виконав:
студент групи КН-115
Сирватка Максим
Викладач:
Мельникова Н.І.

Львів – 2019 р.

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

Графом (точніше **неорієнтованим графом**) називається пара (V, E) .Елементи множини V називають **вершинами** граф G , а елементи з E - його **ребрами**. Аналогічно визначаються **орієнтовані** графи (або **орграфи**), у яких E складається з упорядкованих пар. Ребра неорієнтованих графів позначатимемо у круглих дужках, а орієнтованих у квадратних.

Якщо в E містяться однакові пари, то такі ребра називають **кратними**. Ребра виду (v, v) називають **петлями**. Якщо граф не містить петель та кратних ребер, то його називають **простим**. Далі, якщо не буде сказано інакше, ми розглядаємо лише прості графи. У таких графах сукупність невпорядкованих пар E стає множиною, яку називають множиною **ребер**.

Як правило, в графі $G =(V, E)$ ми будемо нумерувати елементи множини вершин V . У такому випадку одержимо **граф із поміченими вершинами**. Якщо ребро (v, w) належить множині ребер E , то вершини v та w називають **суміжними**. Також говорять, що вершини v та w **інцидентні** ребру (v, w) та навпаки. Два ребра **суміжні**, якщо вони мають спільну вершину.

Степенем вершини v називається кількість ребер, інцидентних цій вершині (позначається $d(v)$). Вершина степеня 0 називається **ізольованою**, а вершина степеня 1 - **висячою**.

Оскільки кожне ребро є інцидентних двом вершинам, то в суму степенів вершин графа кожне ребро входить двічі. Тому справедливе твердження: **сума степенів вершин графа G дорівнює подвоєній кількості його ребер.**

Граф $G =(V, E)$ називається **дводольним**, якщо існує таке розбиття множини його вершин V на дві підмножини V_1 та V_2 так, що кінці будь-якого ребра графа належать різним часткам. Дводольний граф називається **повним дводольним**, якщо будь-які дві його вершини, що належать різним часткам, є суміжними. Повний дводольний граф, частки якого V_1 і V_2 складаються відповідно з n та m вершин, позначають $K_{n,m}$.

Маршрутом або **шляхом**, що з'єднує вершини v та v' називається скінченна послідовність $v, e_i, v_k, e_{i+1}, v_{k+1}, \ldots, e_{i+p}, v'$ вершин та ребер, у якій будь-які два сусідні елементи є інцидентними один одному. Оскільки ми розглядаємо лише прості графи, то в маршрутах ми будемо указувати лише вершини. Отже, **маршрутом** у графі $G =(V, E)$ назвемо таку послідовність вершин $v, v_k, v_{k+1}, \ldots, v_{k+p}, v'$, що всі

пари $(v,v_k),(v_k,v_{k+1}),\ldots,(v_{k+p-1},v_{k+p}),(v_{k+p},v)$ належать множині ребер E. Число ребер у маршруті називають його **довжиною**. Маршрут називають **замкненим**, якщо його кінці співпадають (тобто $v=v$). Маршрут, в якому всі ребра попарно різні називають **ланцюгом**, а маршрут, в якому всі вершини попарно різні (крім, можливо, кінців маршруту) **простим ланцюгом**. Замкнений ланцюг називається **циклом**, а замкнений простий ланцюг **простим циклом**. Простий цикл довжини 3 називається **трикутником**, довжини 4 - **чотирикутником** і т.д.

Граф називається **зв'язним**, якщо будь-які дві його вершини можуть бути з'єднані деяким маршрутом.

Введемо деякі операції на графі $G=(V,E)$:

Доповненням цього графа називається граф $\overline{G}=(V,V^2-E)$, вершини якого ті ж самі, що і у G, та будь-які дві вершини графа $\overline{G}$ суміжні тоді і тільки тоді, коли вони несуміжні у G.

Операція вилучення вершини v із графа G полягає у вилученні з множини V елемента v, а з множини E усіх ребер, інцидентних вершині v. Такий граф позначають $G-v$.

Операція вилучення ребра e із графа G полягає у вилученні з множини E елемента e. Такий граф позначають $G-e$.

Операція підрозбиття ребра $e=(v,w)$ у графі G полягає у вилученні з множини E елемента (v,w), додавання до множини V нової вершини u та додавання до множини E нових ребер (v,u) та (u,w).

Граф $G_1=(V_1,E_1)$ називається **підграфом** графа $G=(V,E)$ і позначається $G_1\subseteq G$, якщо $V_1\subseteq V$ та $E_1\subseteq E$.

Якщо виконується $E_1\neq E$,то G_1 називають **власним** підграфом графа G і позначають $G_1\subset G$. Іншими словами, якщо виконується $G_1\subset G$, то граф G_1 є результатом виконання скінченої кількості операцій вилучення ребер та/або вершин з графа G.

Таблицею (матрицею) суміжності називається квадратна матриця порядку n (n – число вершин графа), елементи якої рівні:

1 - якщо існує дуга між вершинами;

0 - в іншому випадку.

Матриця суміжності повністю **визначає структуру графа.**

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**

1. Алгоритм Прима

Алгоритм Прима — жадібний алгоритм побудови мінімального кістякового дерева зваженого зв'язного неорієнтованого графа.

Побудова починається з дерева, що включає в себе **одну (довільну) вершину.** Протягом роботи алгоритму дерево розростається, поки не охопить **всі вершини** вихідного графа. На кожному кроці алгоритму до поточного дереву приєднується **найлегше з ребер, що з'єднують вершину з побудованого дерева і вершину, що не належить дереву.**

Алгоритм:

1. Спочатку ребра сортують за зростанням ваги.
2. Додають найменше ребро в дерево.
3. Зі списку ребер із найменшою вагою вибирають таке нове ребро, щоб одна з його вершин належала дереву, а інша — ні.
4. Це ребро додають у дерево і знову переходять до кроку 3.
5. Робота закінчується, коли всі вершини будуть у дереві.

2. Алгоритм Краскала

Алгоритм Краскала — алгоритм побудови мінімального кістякового дерева зваженого неорієнтовного графа.

Візьмемо зважений зв'язний граф G=(V, E), де V — множина вершин, E — множина ребер, для кожного з яких задано вагу. Тоді ациклічна множина ребер, що поєднують усі вершини графа і чия загальна вага мінімальна, називається ***мінімальним кістяковим деревом*.**

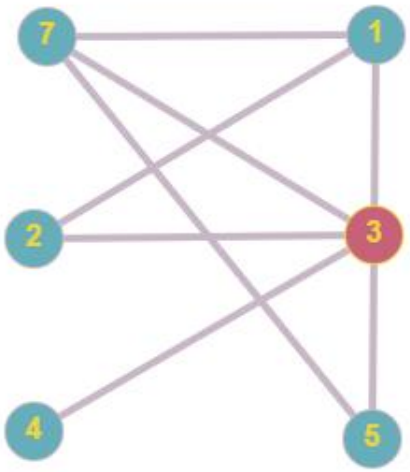
Алгоритм Крускала починається з побудови **виродженого лісу, що містить V дерев, кожне з яких складається з однієї вершини**. Далі виконуються **операції об'єднання двох дерев**, для чого використовуються найкоротші можливі ребра, поки не утвориться єдине дерево. Це дерево і буде **мінімальним кістяковим деревом.**

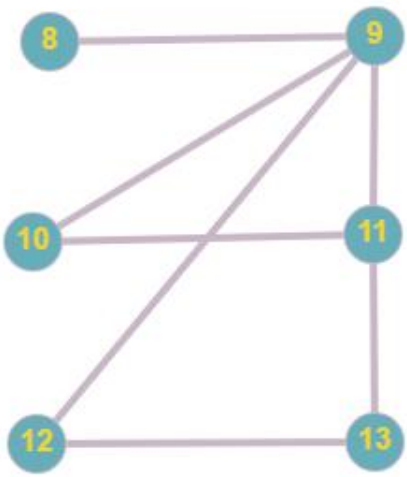
Завдання лабораторної роботи

Варіант 15

1. **Виконати наступні операції над графами:**

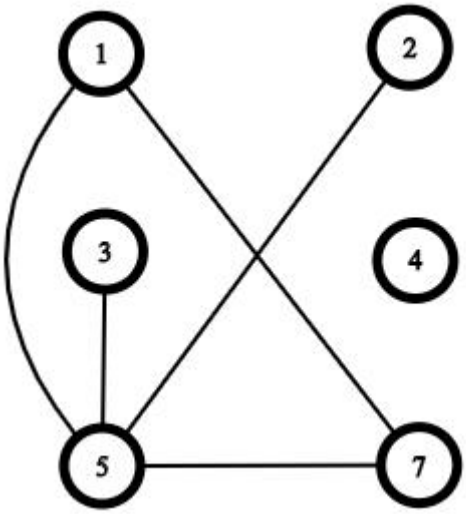
1) знайти доповнення до першого графу;
2) об’єднання графів;
3) кільцеву суму G1 та G2 (G1+G2);
4) розщепити вершину у другому графі;
5) виділити підграф A, що складається з 3-х вершин в G1 і знайти стягнення A в G1 (G1\ A);
6) добуток графів.



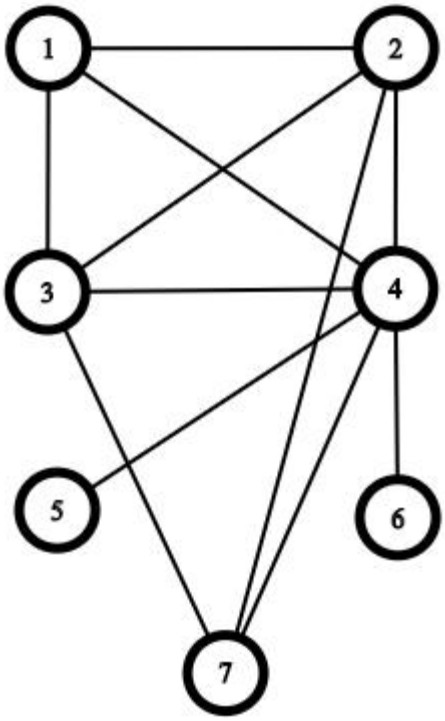


Розв’язання:

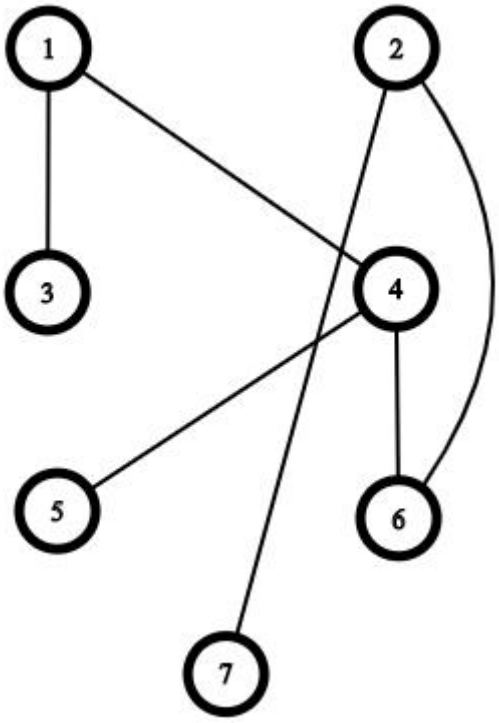
1. Доповнення до першого графу:



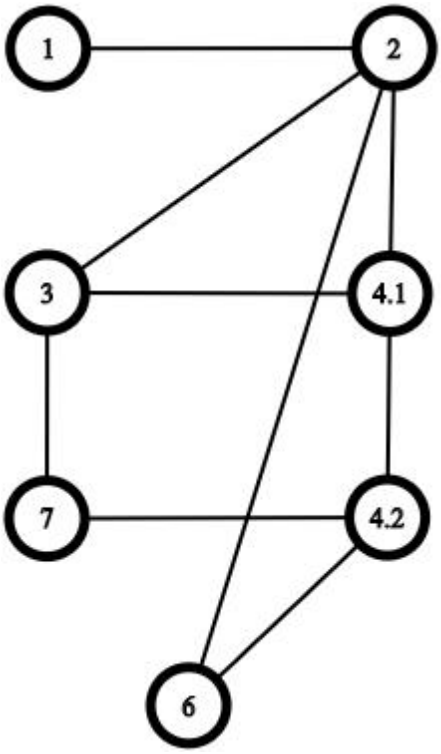
2. Об’єднання графів:



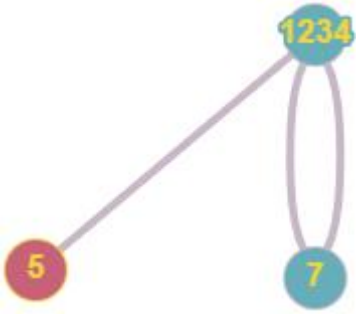
3. Кільцеву суму G1 та G2:



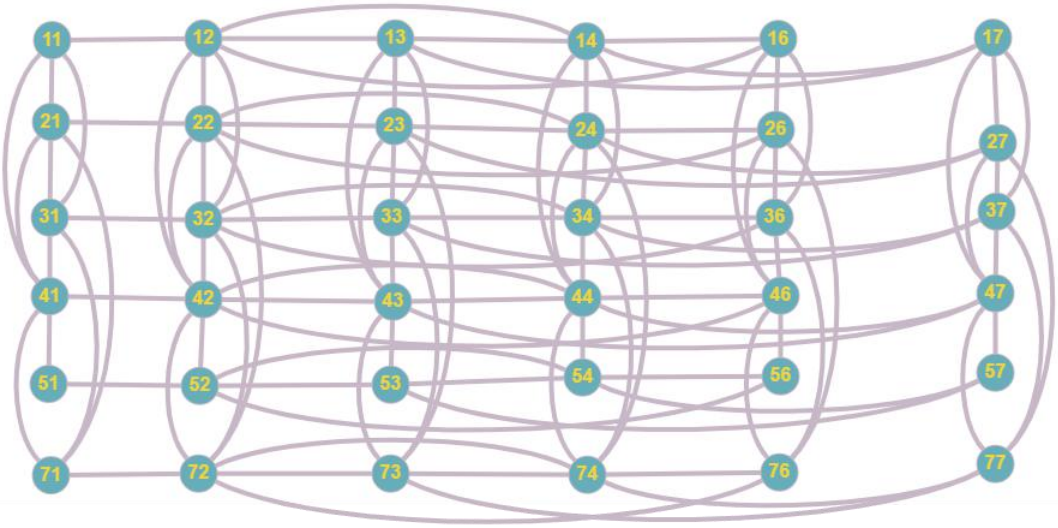
4. Розщепити вершину у другому графі:



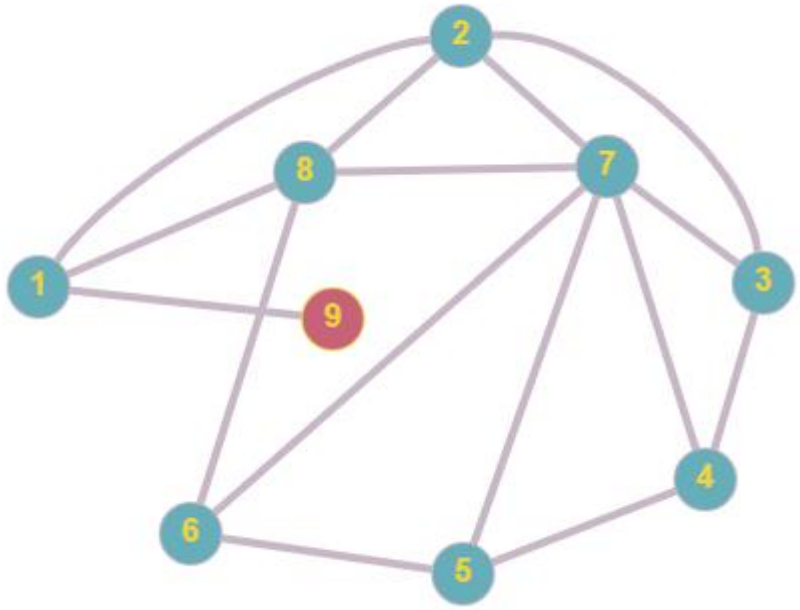
5. Виділити підграф A, що складається з 3-х вершин в G1 і знайти стягнення A в G1:



6. Добуток графів:



2. Знайти таблицю суміжності та діаметр графа.

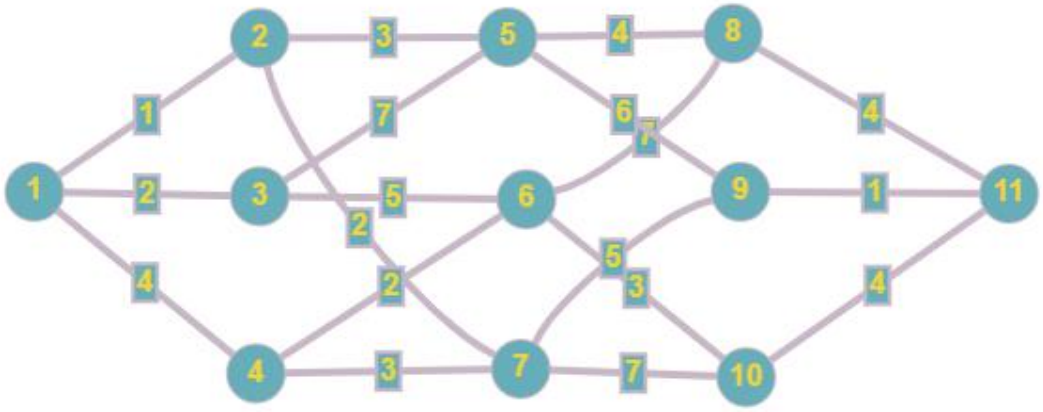


Матриця суміжності:

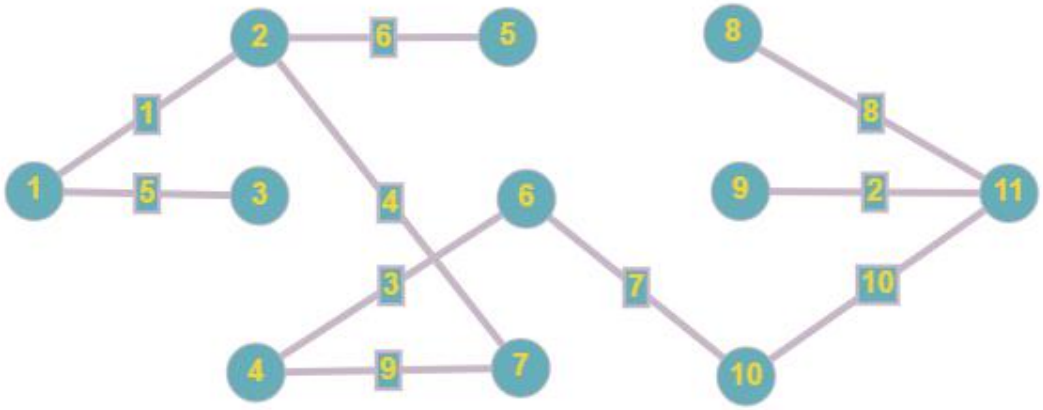
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0	1	0	0	0	0	0	1	1
2	1	0	1	0	0	0	1	1	0
3	0	1	0	1	0	0	1	0	0
4	0	0	1	0	1	0	1	0	0
5	0	0	0	1	0	1	1	0	0
6	0	0	0	0	1	0	1	1	0
7	0	1	1	1	1	1	0	1	0
8	1	1	0	0	0	1	1	0	0
9	1	0	0	0	0	0	0	0	0

Діаметр графа: 4 (4->7->2->1->9)

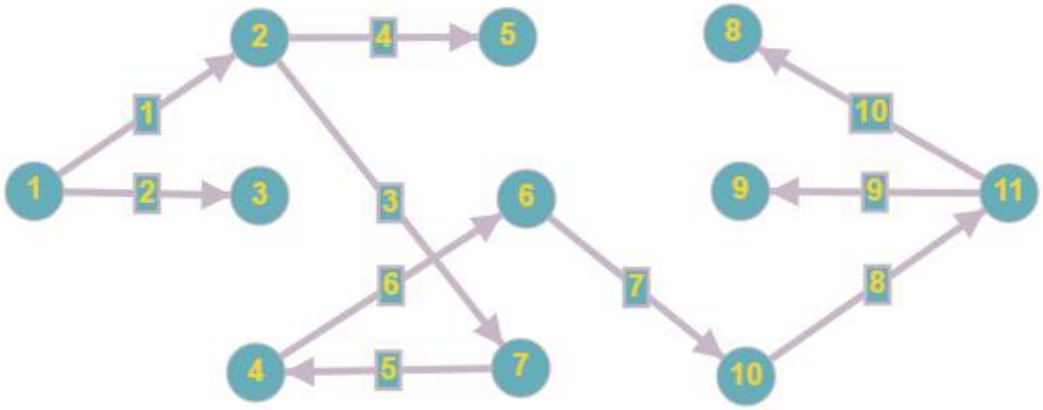
3. Знайти двома методами (Краскала і Прима) мінімальне остове дерево графа.



Метод Краскала (номер на ребрах відповідає послідовності кроків):



Метод Прима(номер на ребрах відповідає послідовності кроків):

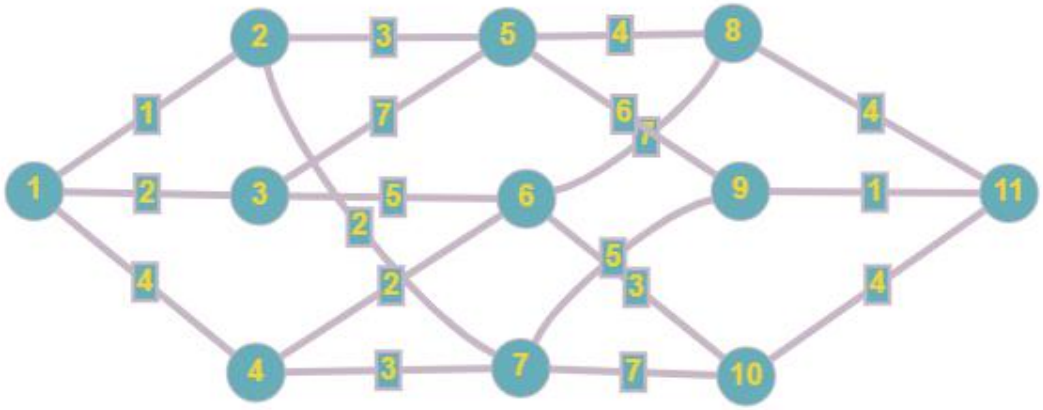


Оскільки дані графи **еквівалентні**, то їхня вага **одинакова** і рівна **25**.

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:



Текст програми:

```
#include <iostream>
#include <Windows.h>
#include <conio.h>

using namespace std;

const int MAX = 2000;

int main()
{
    char s;
    do
    {
        system("cls");
        setlocale(LC_ALL, "Ukr");
        const int n = 11;


        int graph[11][11] =
        { {0, 1, 2, 4, 0, 0, 0, 0, 0, 0, 0},
          {1, 0, 0, 0, 3, 0, 2, 0, 0, 0, 0},
          {2, 0, 0, 0, 7, 6, 0, 0, 0, 0, 0},
          {4, 0, 0, 0, 0, 2, 3, 0, 0, 0, 0},
          {0, 3, 7, 0, 0, 0, 0, 7, 5, 0, 0},
          {0, 0, 6, 2, 0, 0, 0, 7, 0, 3, 0},
          {0, 2, 0, 3, 0, 0, 0, 0, 5, 4, 0},
          {0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 4},
          {0, 0, 0, 0, 5, 0, 5, 0, 0, 0, 1},
          {0, 0, 0, 0, 0, 3, 4, 0, 0, 0, 4},
          {0, 0, 0, 0, 0, 0, 0, 4, 1, 4, 0}, };

        cout << "Матриця суміжності: " << endl;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                cout << graph[i][j] << " ";
            }
            cout << endl;
        }
        cout << endl;
        int start;
        do {
            cout << "Введіть початкову вершину: \n";
            cin >> start;
```

```
        } while (start < 1 || start > 11);
        cout << "\nМінімальне остове дерево: " << endl;
        bool *visited = new bool[n];

        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
            {
                if (graph[i][j] == 0)
                    graph[i][j] = MAX;
            }
            visited[i] = false;
        }
        int v1, v2;
        int min, weight = 0;
        visited[start - 1] = true;
        cout << "E = { ";
        for (int tek = 1; tek < n; tek++)
        {
            min = MAX;
            for (int i = 0; i < n; i++)
            {
                if (visited[i] == true)
                {
                    for (int d = 0; d < n; d++)
                    {
                        if (min > graph[i][d] && !visited[d])
                        {
                            v1 = i;
                            min = graph[i][d];
                            v2 = d;
                        }
                    }
                }
            }
            weight += min;
            visited[v2] = true;
            if (tek != n - 1)
            {
                cout << "(" << v1 + 1 << ',' << v2 + 1 << "),";
            }
            else if (tek == n - 1)
            {
                cout << "(" << v1 + 1 << ',' << v2 + 1 << ") }";
            }
        }
        cout << "\n\nМінімальна вага дерева: " << weight << endl;
        cout << "One more? (y/n)" << endl;
        s = _getch();
    } while (s != 'n');

    return 0;
}
```

Результати виконання програми

```
Матриця суміжності:
0 1 2 4 0 0 0 0 0 0 0
1 0 0 0 3 0 2 0 0 0 0
2 0 0 0 7 6 0 0 0 0 0
4 0 0 0 0 2 3 0 0 0 0
0 3 7 0 0 0 0 7 5 0 0
0 0 6 2 0 0 0 7 0 3 0
0 2 0 3 0 0 0 0 5 4 0
0 0 0 0 7 7 0 0 0 0 4
0 0 0 0 5 0 5 0 0 0 1
0 0 0 0 0 3 4 0 0 0 4
0 0 0 0 0 0 0 4 1 4 0

Введіть початкову вершину:
1

Мінімальне остове дерево:
E = { (1,2),(1,3),(2,7),(2,5),(7,4),(4,6),(6,10),(10,11),(11,9),(11,8) }

Мінімальна вага дерева: 25
One more? (y/n)
```

```
Матриця суміжності:
0 1 2 4 0 0 0 0 0 0 0
1 0 0 0 3 0 2 0 0 0 0
2 0 0 0 7 6 0 0 0 0 0
4 0 0 0 0 2 3 0 0 0 0
0 3 7 0 0 0 0 7 5 0 0
0 0 6 2 0 0 0 7 0 3 0
0 2 0 3 0 0 0 0 5 4 0
0 0 0 0 7 7 0 0 0 0 4
0 0 0 0 5 0 5 0 0 0 1
0 0 0 0 0 3 4 0 0 0 4
0 0 0 0 0 0 0 4 1 4 0

Введіть початкову вершину:
4

Мінімальне остове дерево:
E = { (4,6),(4,7),(7,2),(2,1),(1,3),(2,5),(6,10),(10,11),(11,9),(11,8) }

Мінімальна вага дерева: 25
One more? (y/n)
```

Висновок: на цій лабораторній роботі я навчився будувати графи, виконувати основні операції над графами та знаходити остова мінімальної ваги за алгоритмом Пріма-Краскала.