МІНІСТЕРСТВО ОСВІТИ І НАУКИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра систем штучного інтелекту

Лабораторна робота №4

З курсу “Дискретна математика ”

Виконала:
ст.гр. КН-110
Ямнюк Аліна
Викладач:
Мельникова Н.І.

Львів – 2018

Лабораторна робота № 4.

Тема: Основні операції над графами. Знаходження остовамінімальної ваги за алгоритмом Прима-Краскала.

Мета роботи: набуття практичних вмінь та навичок з використанняалгоритмів Прима і Краскала.

ТЕОРЕТИЧНІ ВІДОМОСТІ ТА ПРИКЛАДИ РОЗВ'ЯЗАННЯ ЗАДАЧ

Графом G називається пара множин (V, E), де V – множина вершин v, а E – множина упорядкованих або неупорядкованих пар $e = \{v', v''\}$, що $v' \in V$ & $v'' \in V$. Неорієнтований та орієнтований граф (орграф) відрізняються упорядкованістю пар $e \in E$.

Кратними ребрами називають ребра, що зв'язують одні і ті ж множини. Ребро, що входить у ту ж вершину, звідки й виходить, називається петлею

Мультиграф – граф, що має кратні ребра. Псевдографи мають петлі, а прості графи не мають ні кратних ребер, ні петель.

Будь-яке ребро є інцидентним вершинам (v', v''), які воно з'єднує. Дві вершини називають суміжними, якщо вони належать до спільного ребра й несуміжними у протилежному випадку. Степенем вершини v називається кількість інцидентних їй ребер.

Граф, що не має ребер, називається пустим. Граф, що не має вершин – нульграфом. Вершина графа, що не інцидентна до жодного з ребер є ізольованою. Вершина із одиничним степенем є листком.

Граф $G' = (V', E')$ є підграфом графа $G = (V, E)$, якщо $V' \subseteq V$ і $E = \{(v', v'') \mid v' \in V \,\&\, v'' \in V \,\&\, (v', v'') \in E\}$. G' також називається кістяковим підграфом, якщо виконано умову $V = V$ & $E \subseteq E$.

Над графами можна виконувати деякі операції.

1. Вилученням ребра $e \in E$ можна отримати новий граф $G' = (V, E \backslash \{e\})$
2. Доповненням графа $G = (V, E)$, можна отримати новий граф $G' = (V', E')$, де $E = \{(v1, v2) | (v1, v2) \notin E\}$;
3. Об'єднанням графів $G1 = (V1, E1)$ та $G2 = (V2, E2)$ є новий граф $G = (V, E)$, в якому $V = V1 \cup V2$, $E = E1 \cup E2$;
4. Кільцевою сумою графів $G1 = (V1, E1)$ та $G2 = (V2, E2)$ є граф $G = (V, E)$, в якому $V = V1 \cup V2$, а $E = E1 \Delta E2$;
5. Розщепленням вершин є операція, за якої на місці однієї вершини з'являється дві, інцидентні одна до одної. Нові вершини довільно успадковують інцидентність зі старими вершинами.
6. Стягненням ребра ($\boldsymbol{a}$, $\boldsymbol{b}$) є операція, при якій це ребро зникає, і замість точок a та b виникає c, що успадковує їхню інцидентність.
7. Добутком графів $G1 = (V1, E1)$ та $G2 = (V2, E2)$ є граф $G = G1 \times G2 = (V, E)$, в якого $V = V1 \times V2$, а вершини $(v1', v2')$ та $(v1'', v2'')$ суміжні тільки якщо $(v1' = v1'')$ & $(v2', v2'') \in E2$ або $(v2' = v2'')$ & $(v1', v1'') \in E1$.

Матрицею суміжності $R = [rij]$ графа $G = (V, E)$ є квадратна матриця порядку $|V|$, елементи rij якої визначаються за формулою $rij = \{\ 1, (vi, vj) \in E\ 0, (vi, vj) \notin E$.

Діаметром зв’язного графа є максимально можлива довжина між двома його вершинами.

В неорієнтованому графі $G = (V, E)$ маршрутом довжини j є послідовність $M = \{(v1, v2), (v2, v3), \dots , (vj-2, vj-1), (vj-1, vj-2)\}$, що складається з ребер $e \in E$. При цьому кожні два сусідні ребра мають спільну кінцеву вершину. Маршрут називається ланцюгом, якщо всі його ребра різні. Відкритий ланцюг називається шляхом, якщо всі його вершини різні. Замкнений ланцюг називається циклом, якщо всі його вершини, за винятком кінцевих, є різними. Шлях і цикл називаються гамільтоновими, якщо вони проходять через усі вершини графа.

Для пошуку кістякового дерева мінімальної ваги у зв’язному графі можна використовувати алгоритми Прима та Краскала.

1.1Алгоритм Прима

Нехай будуємо дерево $G' = (V',E')$, де $V' = E' = \emptyset$. Над графом $G = (V, E)$ алгоритм Прима буде мати такий порядок дій:

1. Вибрати довільну точку $v \in V$, і додати її у V'.
2. Вибрати перше найлегше ребро $e = (a, b) \in E$, де $a \in V'$, $b \in V$, що не утворює циклів.
   a. Якщо такого ребра не знайдено – завершити роботу, повернувши дерево (V', E').
3. Додати точку b та ребро e у G'.
4. Повернутись до кроку 2.

Якщо граф заданий матрицею суміжності, то ціну цього алгоритму можна оцінити як $O(|V| 2)$. Якщо задати граф списком суміжності, ціною алгоритму буде $O(|E| \log|V|)$.

1.2Алгоритм Краскала

Нехай будуємо дерево $G' = (V',E')$, де $V' = E' = \emptyset$. Над графом $G = (V, E)$ алгоритм Краскала буде мати такий порядкодій:

1. Вибрати найлегше ребро $e = (u, v) \in E$ таке, щоби воно не утворювало циклів
   a. Якщо таких ребер немає – завершити роботу, повернувши дерево (V', E')
2. Додати точки u і v та ребро (u, v) у G'.
3. Повернутись до кроку 1.

Ціна цього алгоритму - $O(|E| \log|E|)$

ІНДИВІДУАЛЬНІ ЗАВДАННЯ

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

1. Виконати наступні операції над графами:

1) знайти доповнення до першого графу,

2) об’єднання графів,

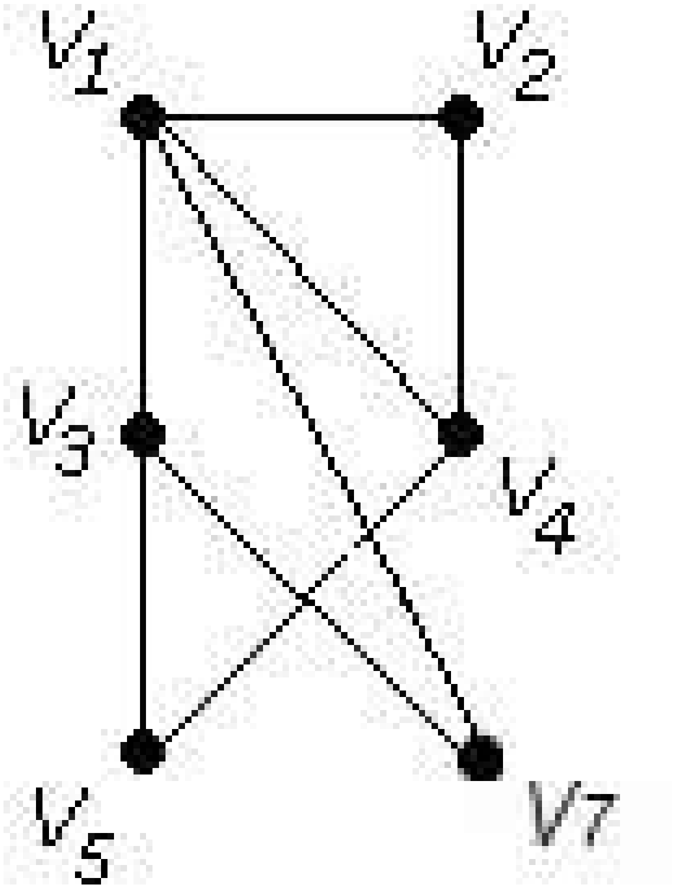
3) кільцеву суму G1 та G2 (G1+G2),

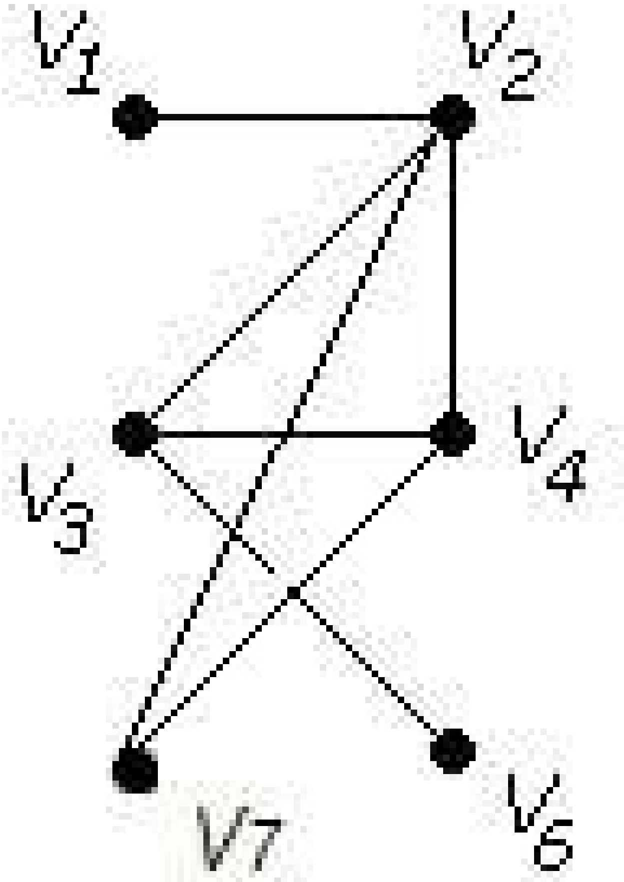
4) розщепити вершину у другому графі,

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

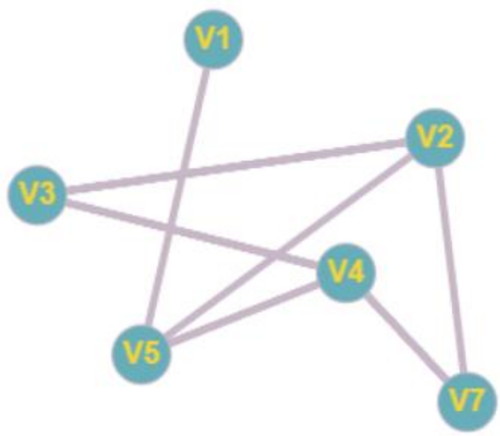
стягнення А в G1 (G1\ A),

6) добуток графів.

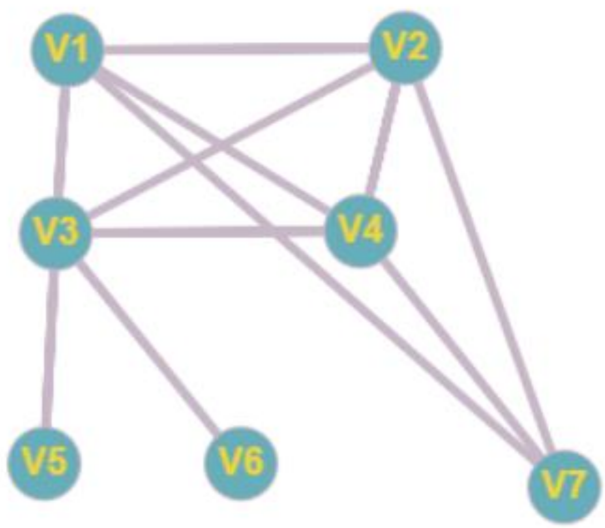




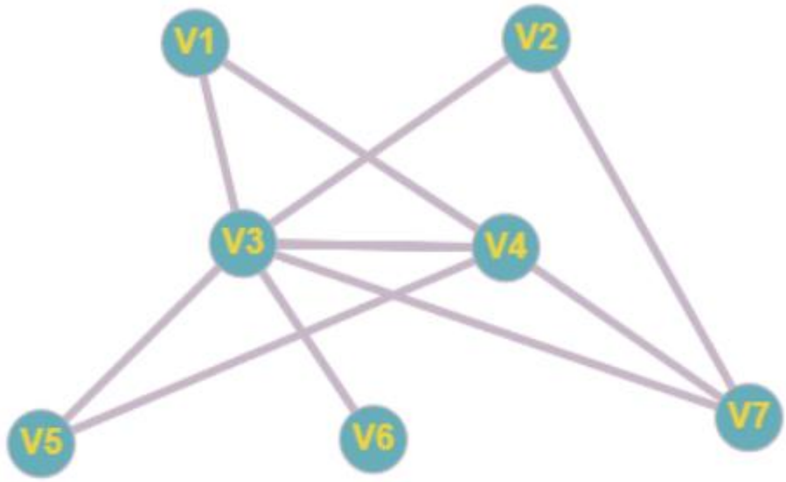
1) знайти доповнення до першого графа



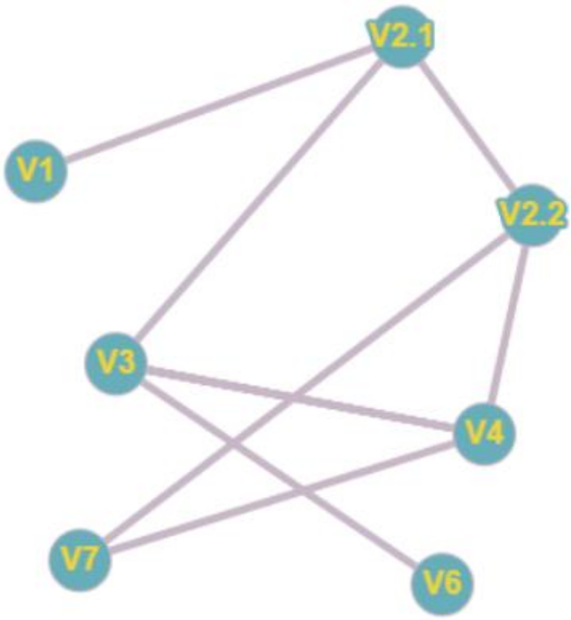
2) об’єднання графів



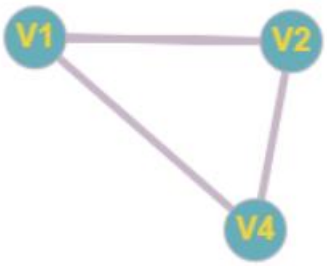
3) кільцеву суму G1 та G2 (G1+G2)

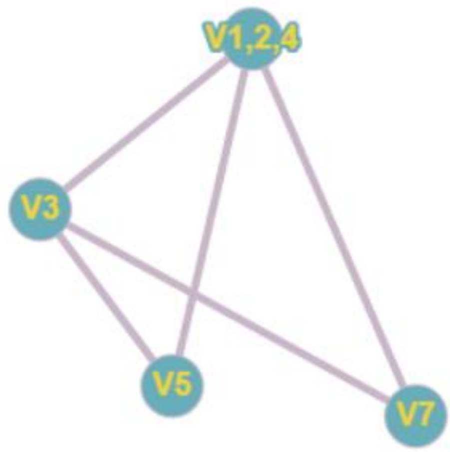


4) розщепити вершину у другому графі

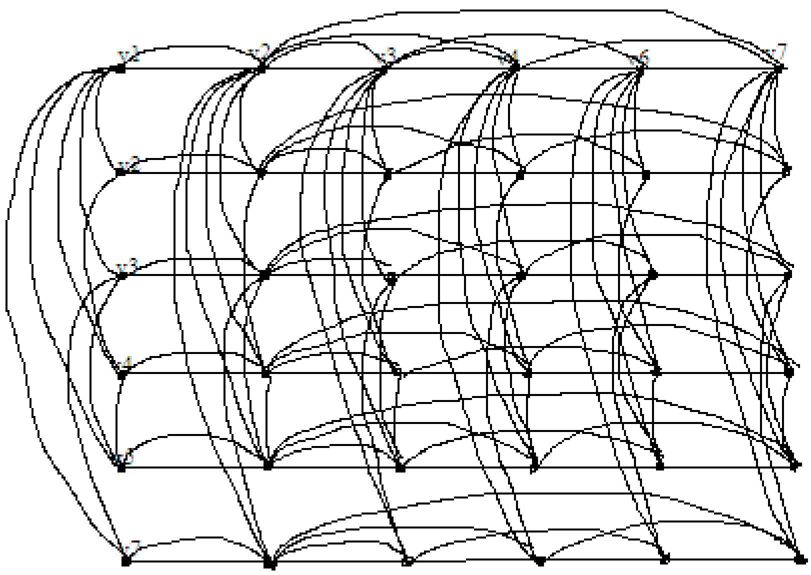


5) виділити підграф A, що складається з 3-х вершин в G1 і знайти стягнення A в G1 (G1\ A)

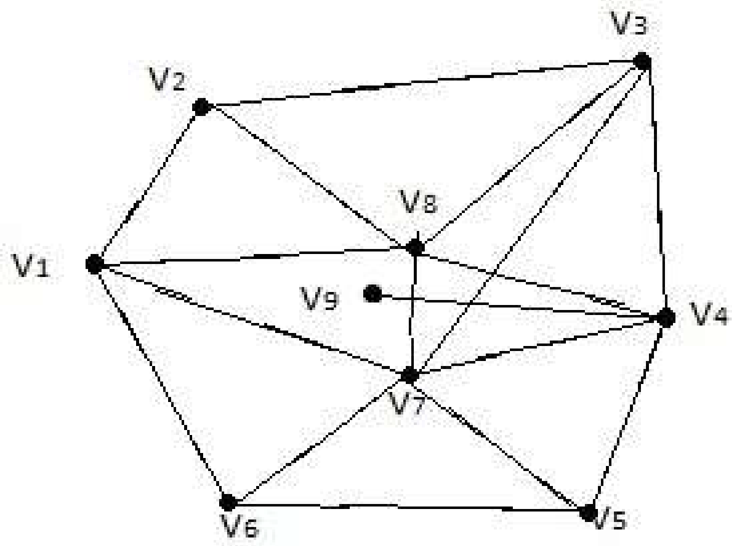




6) добуток графів



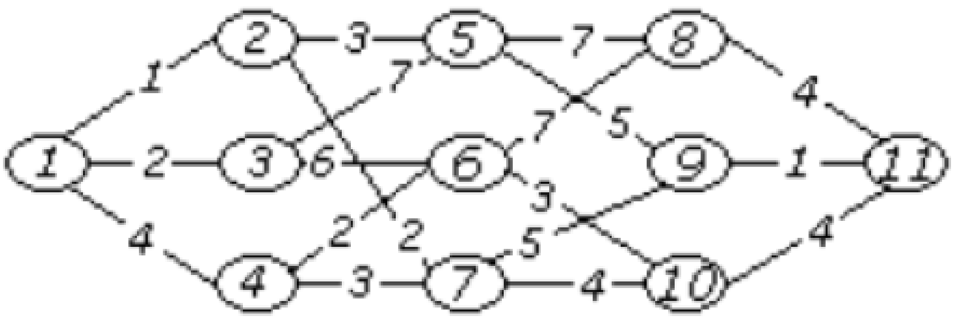
2. Знайти таблицю суміжності та діаметр графа.



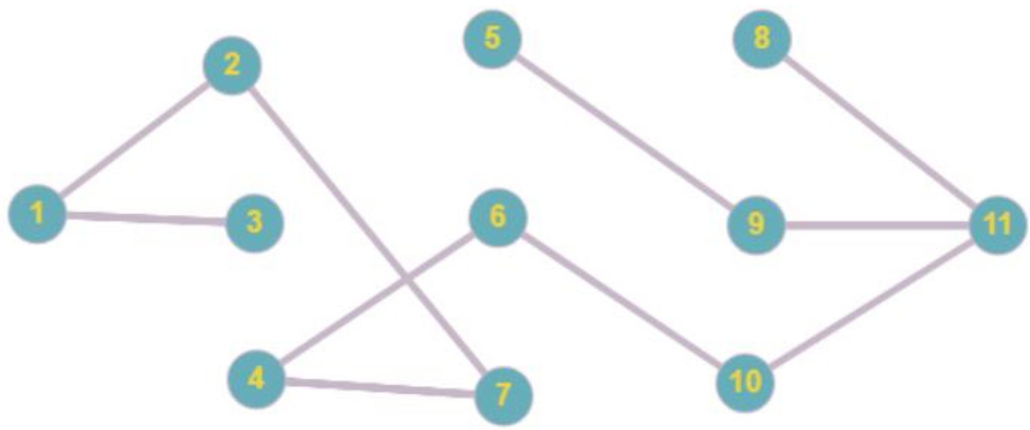
Діаметр = 3.

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

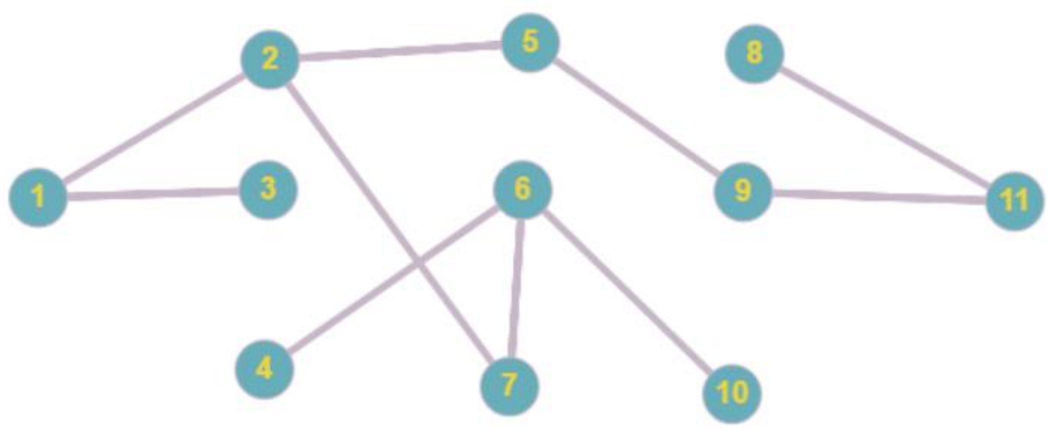
3. Знайти двома методами (Краскала і Прима) мінімальне остоведерево графа.



Метод Прима:

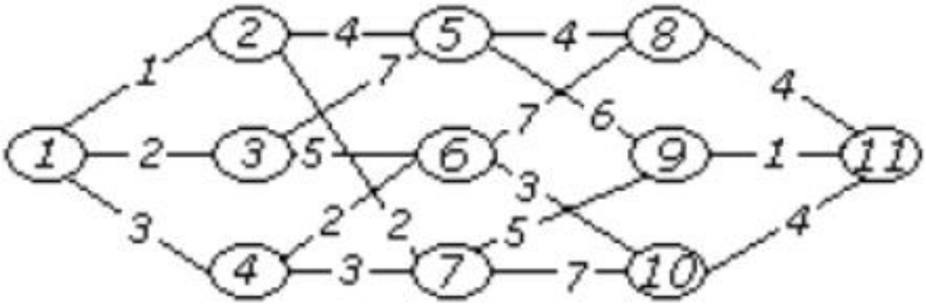


Метод Краскала:



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

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



Код:

```
#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()
{
    // the adjecency matrix of our graph (with weight)
    //          1  2  3  4  5  6  7  8  9  10 11
    int A[11][11] = {
        /*1*/ { 0, 1, 2, 3, 0, 0, 0, 0, 0, 0, 0 },
        /*2*/ { 1, 0, 0, 4, 0, 2, 0, 0, 0, 0, 0 },
        /*3*/ { 2, 0, 0, 0, 7, 5, 0, 0, 0, 0, 0 },
        /*4*/ { 3, 0, 0, 0, 0, 2, 3, 0, 0, 0, 0 },
        /*5*/ { 0, 4, 7, 0, 0, 0, 0, 4, 6, 0, 0 },
        /*6*/ { 0, 0, 5, 2, 0, 0, 0, 7, 0, 3, 0 },
        /*7*/ { 0, 2, 0, 3, 0, 0, 0, 0, 5, 7, 0 },
        /*8*/ { 0, 0, 0, 0, 4, 7, 0, 0, 0, 0, 4 },
        /*9*/ { 0, 0, 0, 0, 6, 0, 5, 0, 0, 0, 1 },
       /*10*/ { 0, 0, 0, 0, 0, 3, 7, 0, 0, 0, 4 },
       /*11*/ { 0, 0, 0, 0, 0, 0, 0, 4, 1, 4, 0 }
    };

    removeRepeated(11, A);

   /* Prints verticles sorted by weight
    */
    printf("\nVerticles sorted by weight:");

    // weight, 7 is max weight
    for (int i = 1; i <= 7; i++)
    {
        printf("\n%d: ", i);
        // first edge
        for (int j = 1; j <= 11; j++)
        {
            // second edge
            for (int k = 1; k <= 11; k++)
            {
```

```c
            {
                if (A[j - 1][k - 1] == i)
                {
                    printf("%d-%d; ", j, k);
                }
            }
        }
    }

   /* Checks sorted vertivles and adds one to our path only if two edges are in different trees
    */

    int B[11][11];
    makeTrees(11, B);

    printf("\n\nOur path: ");
    // weight, 7 is max weight
    for (int i = 1; i <= 7; i++)
    {
        // first edge
        for (int j = 1; j <= 11; j++)
        {
            // second edge
            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])
{
```

```c
    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;

    // line
    for (int i = 0; i < n; i++)
    {
        temp1 = 0;
        temp2 = 0;
        // first element
```

```c
        for (int j = 0; j < n; j++)
        {
            if (A[i][j] == first)
            {
                temp1 = 1;
            }
        }
        // second element
        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 scndLine;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (A[i][j] == second)
            {
                scndLine = i;
            }
        }
    }

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

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

Результат:

```
Verticles sorted by weight:
1: 1-2; 9-11;
2: 1-3; 2-6; 4-6;
3: 1-4; 4-7; 6-10;
4: 2-4; 5-8; 8-11; 10-11;
5: 3-6; 7-9;
6: 5-9;
7: 3-5; 6-8; 7-10;

Our path: 1-2; 9-11; 1-3; 2-6; 4-6; 4-7; 6-10; 5-8; 8-11; 10-11;
```

Висновок: набула практичних навичок з використанням алгоритмів Прима і Краскала.