МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота №4

з дисципліни

«Дискретна математика»

Виконав:

студент групи КН-109

Яворський Володимир

Викладач:

Мельникова Н.І.

Львів – 2018 р.

Лабораторна робота № 4

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.

Варіант №14

Завдання 1

1. *Виконати наступні операції над графами:*

1) знайти доповнення до першого графу,

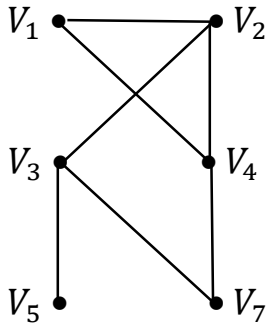
2) об’єднання графів,

3) кільцеву суму G1 та G2 (G1+G2),

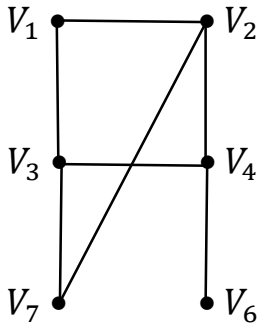
4) розщепити вершину у другому графі,

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

6) добуток графів.

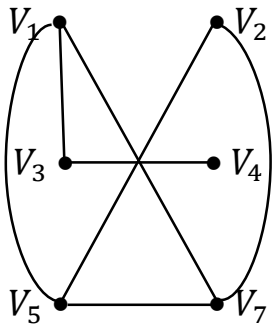


G_1

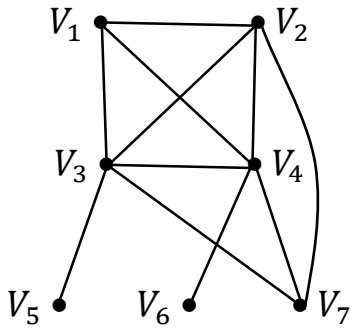


G_2

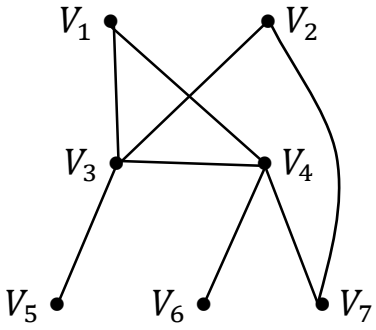
1) доповнення до першого графу



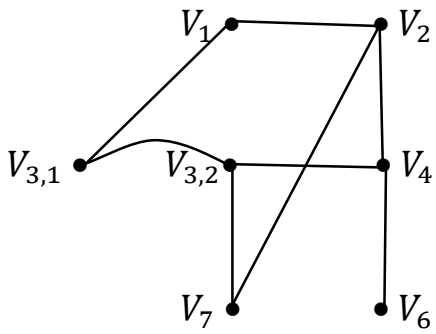
2) об’єднання графів



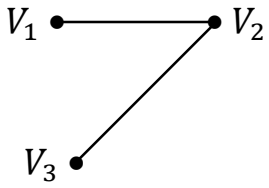
3) кільцева сума $\boldsymbol{G_1}$ та $\boldsymbol{G_2}$ ($\boldsymbol{G_1 + G_2}$)

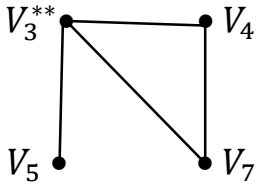


4) розщеплення вершини $\boldsymbol{V_3}$ у другому графі

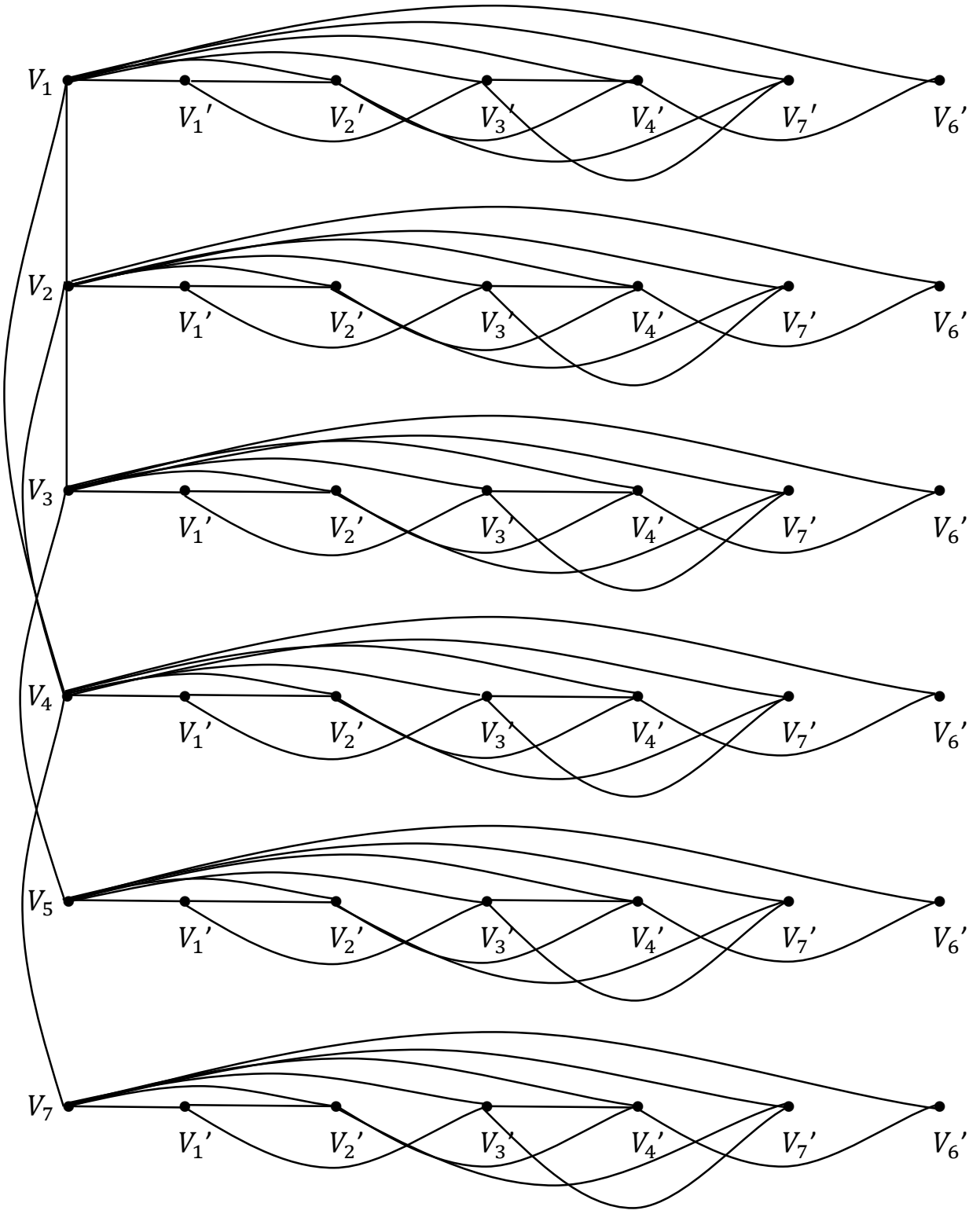


5) підграф **A**, що складається з 3-х вершин в $\boldsymbol{G_1}$ ($\boldsymbol{V_1}, \boldsymbol{V_2}, \boldsymbol{V_3}$) і стягнення **A** в $\boldsymbol{G_1}$ ($\boldsymbol{G_1}$ **\ A**)

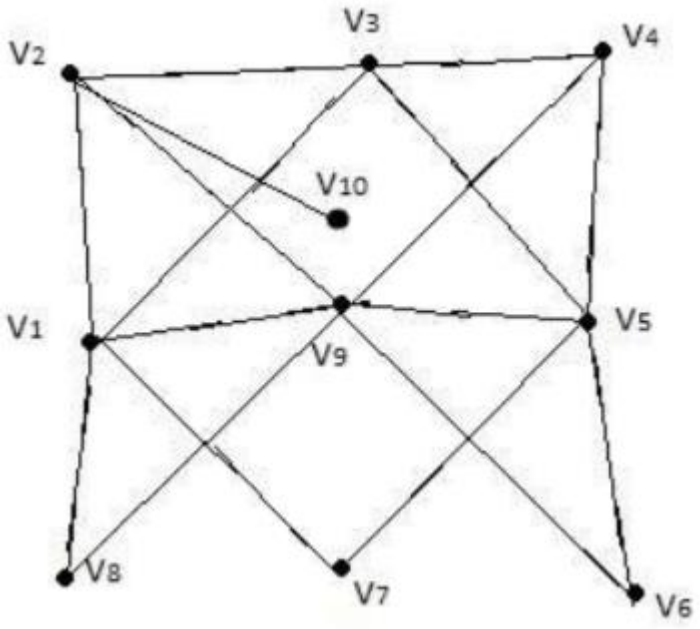




6) добуток графів



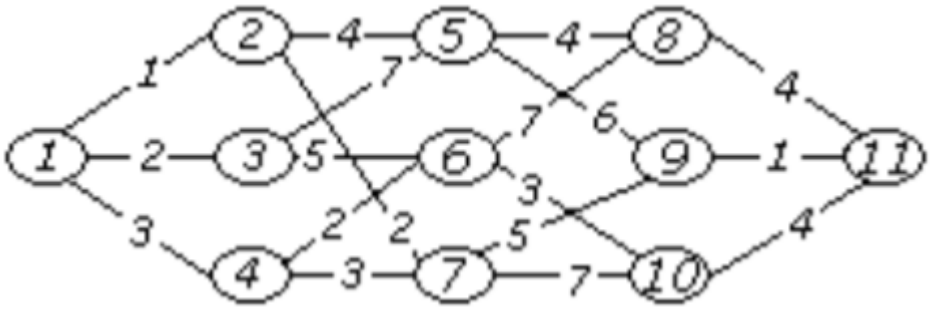
2. Знайти таблицю суміжності та діаметр графа.



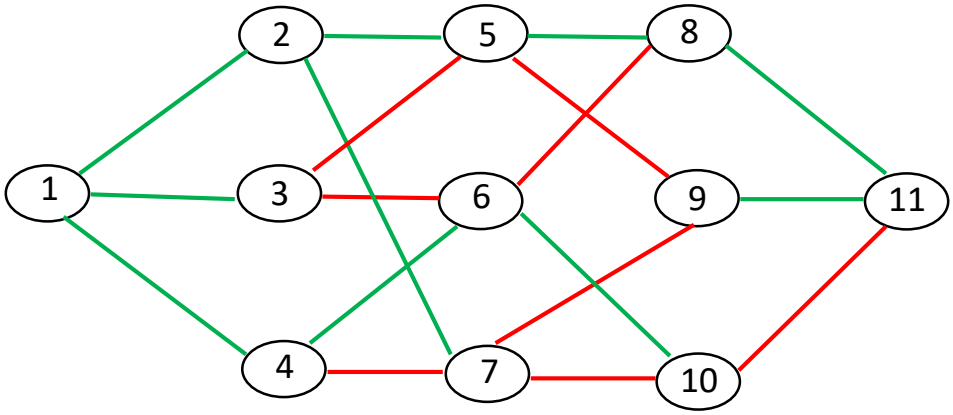
	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	V_{10}
V_1	0	**1**	**1**	0	0	0	**1**	**1**	**1**	0
V_2	**1**	0	**1**	0	0	0	0	0	**1**	**1**
V_3	**1**	**1**	0	**1**	**1**	0	0	0	0	0
V_4	0	0	**1**	0	**1**	0	0	0	**1**	0
V_5	0	0	**1**	**1**	0	**1**	0	0	**1**	0
V_6	0	0	0	0	**1**	0	0	0	**1**	0
V_7	**1**	0	0	0	**1**	0	0	0	0	0
V_8	**1**	0	0	0	0	0	0	0	**1**	0
V_9	**1**	**1**	0	**1**	**1**	**1**	0	**1**	0	0
V_{10}	0	**1**	0	0	0	0	0	0	0	0

d = 3 (найдовша відстань – від V_8 до V_{10}, 8-1-2-10)

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



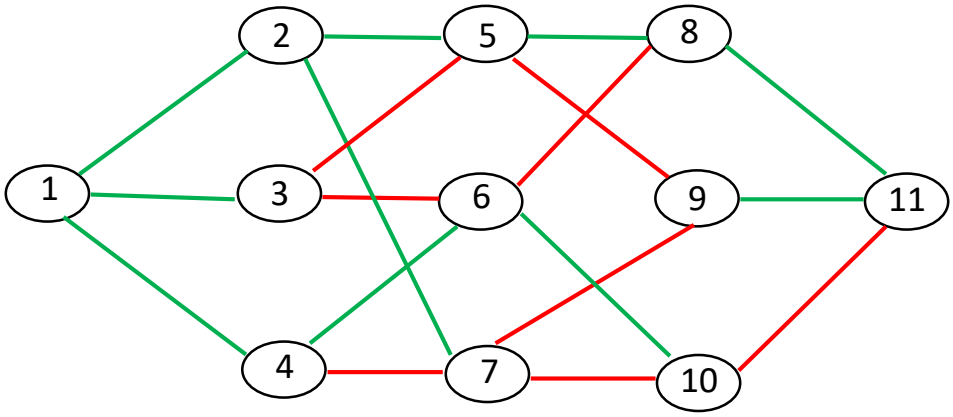
1) Алгоритм Краскала:



Додаємо до дерева ребро з мінімальною довжиною 1: **1-2;**
Додаємо до дерева ребро з довжиною 1: **9-11**;
Додаємо до дерева ребро з довжиною 2: **1-3**;
Додаємо до дерева ребро з довжиною 2: **4-6**;
Додаємо до дерева ребро з довжиною 2: **2-7**;
Додаємо до дерева ребро з довжиною 3: **1-4**;
Додаємо до дерева ребро з довжиною 3: **6-10**;
Ребро **4-7** до дерева не додається, оскільки утвориться цикл;
Додаємо до дерева ребро з довжиною 4: **2-5**;
Додаємо до дерева ребро з довжиною 4: **5-8**;

Додаємо до дерева ребро з довжиною 4: **8-11**;
Ребро **10-11** до дерева не додається, оскільки утвориться цикл;
Ребро **3-6** до дерева не додається, оскільки утвориться цикл;
Ребро **7-9** до дерева не додається, оскільки утвориться цикл;
Ребро **7-10** до дерева не додається, оскільки утвориться цикл;
Ребро **5-9** до дерева не додається, оскільки утвориться цикл;
Ребро **6-8** до дерева не додається, оскільки утвориться цикл;
Ребро **3-5** до дерева не додається, оскільки утвориться цикл;

2) Алгоритм Прима:



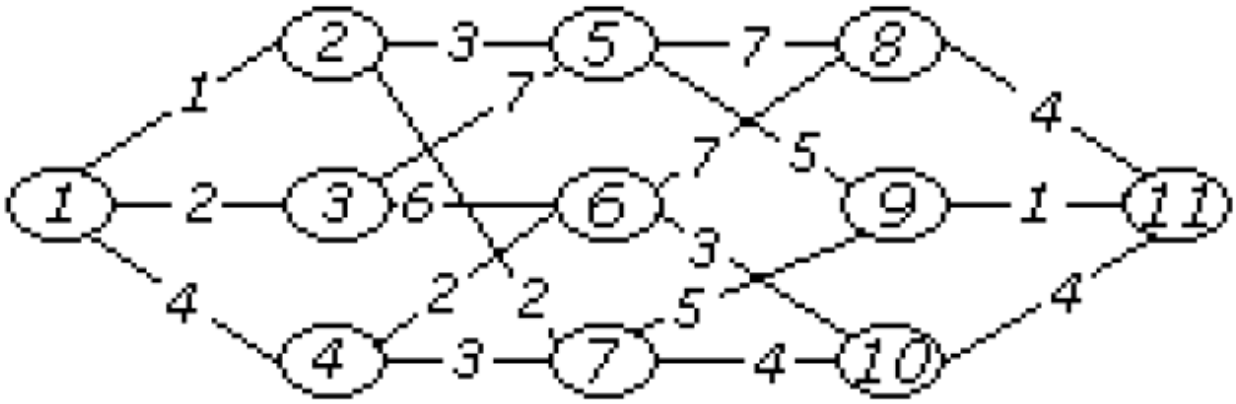
Оберемо вершину довільну вершину 1;
Додаємо до дерева інцидентне ребро **1-2** з найменшою довжиною 1;
Додаємо до дерева інцидентне ребро **1-3** з найменшою довжиною 2;
Додаємо до дерева інцидентне ребро **2-7** з найменшою довжиною 2;
Додаємо до дерева інцидентне ребро **1-4** з найменшою довжиною 3;
Додаємо до дерева інцидентне ребро **4-6** з найменшою довжиною 2;
Ребро **4-7** до дерева не додається, оскільки утвориться цикл;
Додаємо до дерева інцидентне ребро **6-10** з найменшою довжиною 3;
Додаємо до дерева інцидентне ребро **2-5** з найменшою довжиною 4;
Додаємо до дерева інцидентне ребро **5-8** з найменшою довжиною 4;
Додаємо до дерева інцидентне ребро **8-11** з найменшою довжиною 4;
Додаємо до дерева інцидентне ребро **9-11** з найменшою довжиною 1;
Ребро **10-11** до дерева не додається, оскільки утвориться цикл;
Ребро **3-6** до дерева не додається, оскільки утвориться цикл;

Ребро **7-9** до дерева не додається, оскільки утвориться цикл;
Ребро **5-9** до дерева не додається, оскільки утвориться цикл;
Ребро **3-5** до дерева не додається, оскільки утвориться цикл;
Ребро **6-8** до дерева не додається, оскільки утвориться цикл;
Ребро **7-10** до дерева не додається, оскільки утвориться цикл;

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 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()
{
   // 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, 4, 0, 0, 0, 0, 0, 0, 0 },
    /*2*/ { 1, 0, 0, 0, 3, 0, 2, 0, 0, 0, 0 },
    /*3*/ { 2, 0, 0, 0, 7, 6, 0, 0, 0, 0, 0 },
    /*4*/ { 4, 0, 0, 0, 0, 2, 3, 0, 0, 0, 0 },
    /*5*/ { 0, 3, 7, 0, 0, 0, 0, 7, 5, 0, 0 },
    /*6*/ { 0, 0, 6, 2, 0, 0, 0, 7, 0, 3, 0 },
    /*7*/ { 0, 2, 0, 3, 0, 0, 0, 0, 5, 4, 0 },
    /*8*/ { 0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 4 },
    /*9*/ { 0, 0, 0, 0, 5, 0, 5, 0, 0, 0, 1 },
   /*10*/ { 0, 0, 0, 0, 0, 3, 4, 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++)
       {
          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])
{
   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
    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++)
    {
        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 виконання програми:

```
jharvard@appliance (~/Dropbox/hello): ./kruskal

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

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

***Висновок*:** теорія графів є дуже важливою у вивченні дискретної математики. Графи можуть використовуватися у широкому спектрі задач таких, як знаходження мінімального остового дерева, що потрібно наприклад для поштаря, щоб обійти кожен будинок і пройти найменший шлях.