МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних наук та інформаційних технологій
Кафедра систем штучного інтелекту

Лабораторна робота №4

з курсу “Дискретна математика ”

Виконав:
ст. гр. КН-110
Петров Кирил

Викладач:
Мельникова Н.І.

Львів – 2018

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 ТА ПРИКЛАДИ РОЗВ'ЯЗАННЯ ЗАДАЧ

Теорія графів дає простий, доступний і потужний інструмент побудови моделей прикладних задач, є ефективним засобом формалізації сучасних інженерних і наукових задач у різних областях знань.

Графом G називається пара множин (V, E), де V − множина вершин, перенумерованих числами 1, 2, ..., $n = \upsilon$; $V = \{\upsilon\}$, E − множина упорядкованих або неупорядкованих пар $e = (v', v'')$, $v' \in V$, $v'' \in V$, називаних дугами або ребрами, $E = \{e\}$. При цьому не має примусового значення, як вершини розташовані в просторі або площині і які конфігурації мають ребра.

Неорієнтованим графом G називається граф у якого ребра не мають напрямку. Такі ребра описуються неупорядкованою парою (v', v''). *Орієнтований граф (орграф)* – це граф ребра якого мають напрямок та можуть бути описані упорядкованою парою (v', v''). Упорядковане ребро називають *дугою*. Граф є *змішаним*, якщо наряду з орієнтованими ребрами (дугами) є також і неорієнтовані. При розв'язку задач змішаний граф зводиться до орграфа.

Кратними (паралельними) називаються ребра, які зв'язують одні і ті ж вершини. Якщо ребро виходить та й входить у дну і ту саму вершину, то таке ребро називається *петлею.*

Мультиграф – граф, який має кратні ребра. *Псевдограф* – граф, який має петлі. *Простий граф* – граф, який не має кратних ребер та петель.

Будь яке ребро e *інцедентно* двом вершинам (v', v''), які воно з'єднує. У свою чергу вершини (v', v'') інцендентні до ребра e . Дві вершини (v', v'') називають *суміжними*, якщо вони належать до одного й того самого ребра e , і *несуміжні* у протилежному випадку. Два *ребра називають суміжними*, якщо вони мають спільну вершину. Відношення суміжності як для вершин, так і для ребер є симетричним відношенням. *Степенем вершини* графа G називається число інцидентних їй ребер.

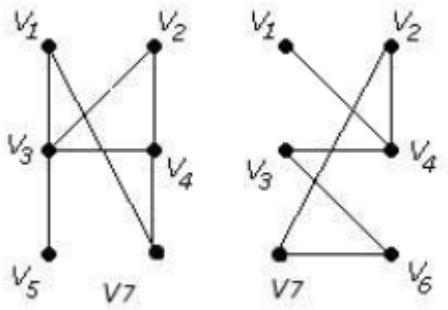
Граф, який не має ребер називається *пустим графом*, *нуль-графом*. Вершина графа, яка не інцедентна до жодного ребра, називається *ізольованою*. Вершина графа, яка інцедентна тільки до одного ребра, називається *звисаючою.*

Частина $G' = (V', E')$ графа $G = (V, E)$ називається *підграфом* графа G , якщо $V' \subseteq V$ і E' складається з тих і тільки тих ребер $e = (v', v'')$, у яких обидві кінцеві вершини $v', v'' \in V'$. Частина

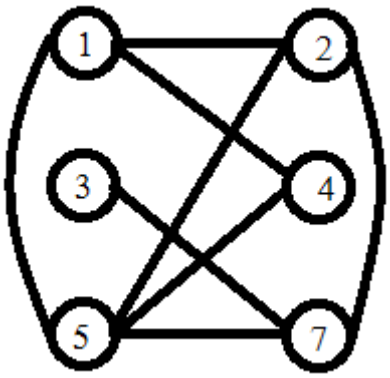
$G' = (V', E')$ називається *суграфом* або *остовим підграфом* графа G, якщо виконано умови: $V' = V$, $E' \subseteq E$.

Варіант 4

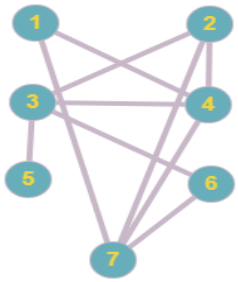
Завдання № 1. Розв'язати на графах наступні задачі:
1. Виконати наступні операції над графами:
1) знайти доповнення до першого графу,
2) об’єднання графів,
3) кільцеву суму G1 та G2 (G1+G2),
4) розщепити вершину у другому графі,
5) виділити підграф А, що складається з 3-х вершин в G1 і знайти стягнення А в G1 (G1\ A),
6) добуток графів.



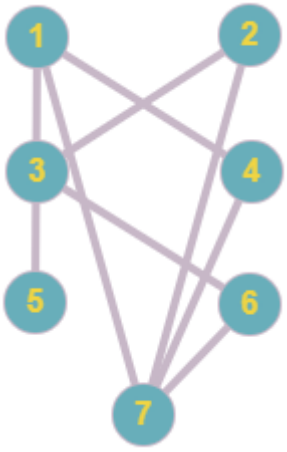
1) Доповнення до першого графу:



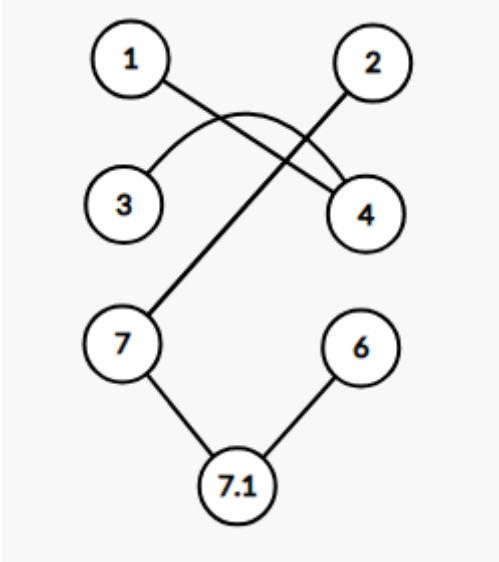
2) Об’єднання:



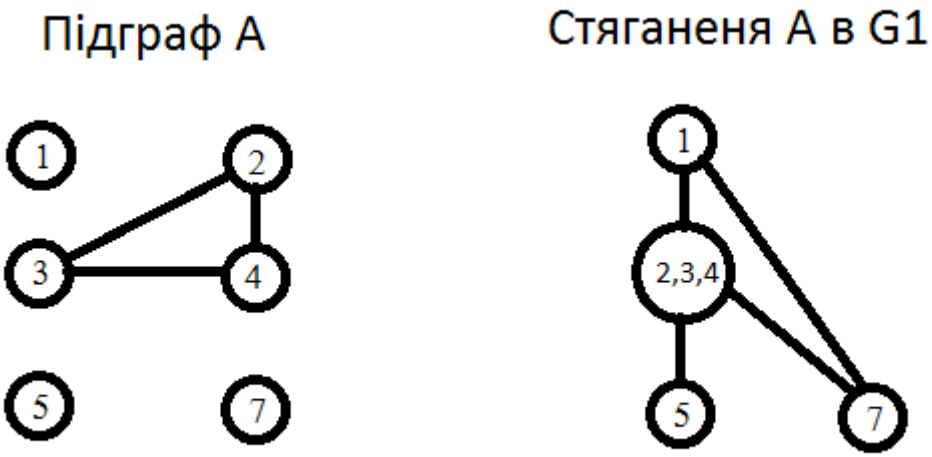
3) Кільцева сума :



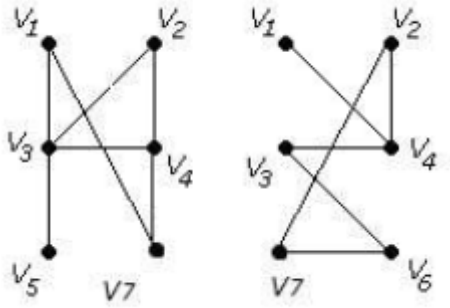
4) Розщеплення 7 вершини у 2 графі:

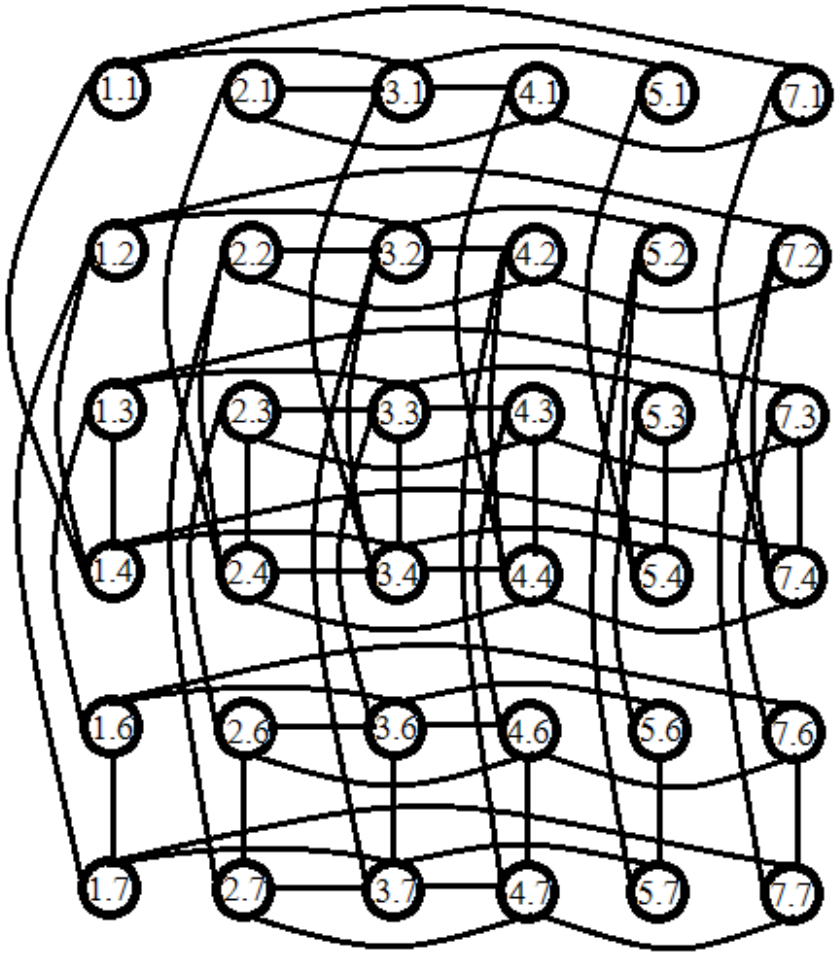


5) Виділити підграф А, що складається з трьох вершин в G1:



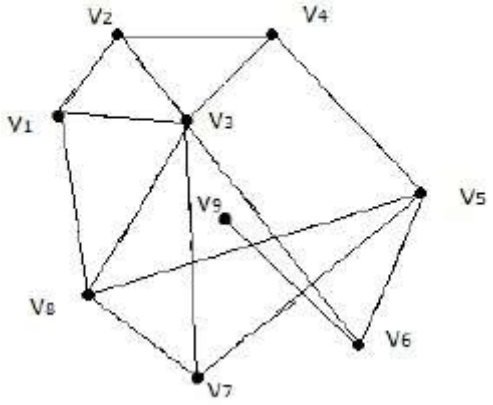
6) Добуток





2. Знайти таблицю суміжності та діаметр графа.

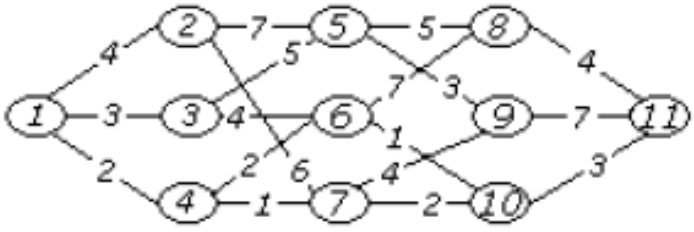
4



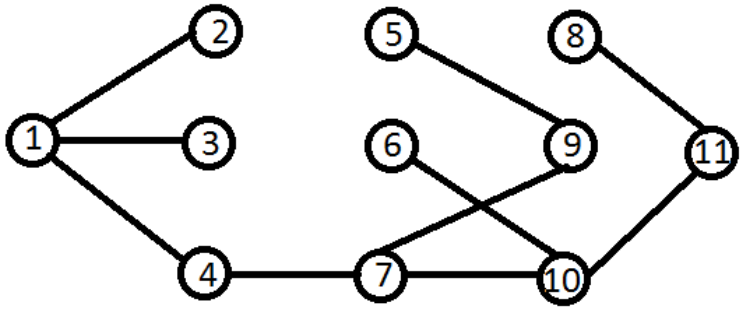
Діаметр=3(V1-V3-V6-V9)

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

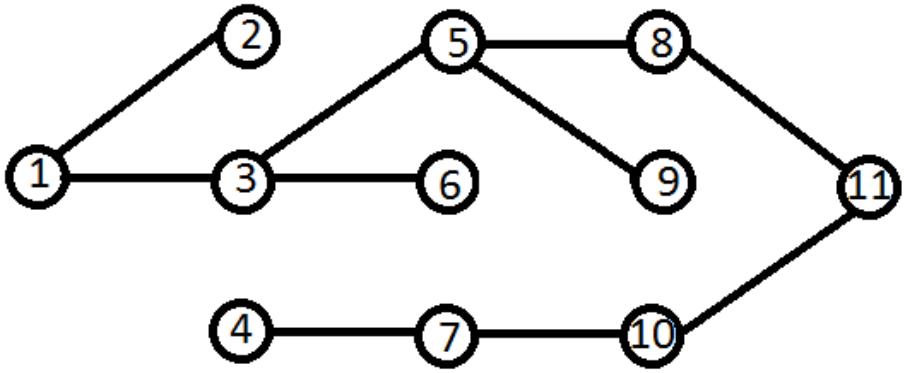
3. Знайти двома методами (Краскала і Прима) мінімальне остове дерево графа.



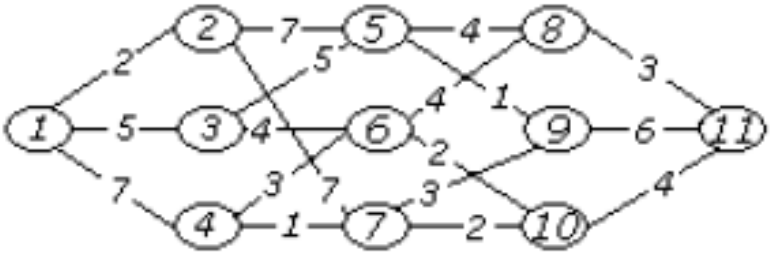
Краскала



Прима



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




```c
#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()
{
```

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

removeRepeated(11, A);

  for (int i = 1; i <= 7; i++)
   {
    printf("\nEdges with weight: %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);
            }
        }
    }
}
```

```
 //Check sorted edges and add to tree


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

printf("\n\nNew Tree: ");
// 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);
            }
```

```
                    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;
                    }
                }
            }
        }
    }
}
```

```
Edges with weight: 1: 4-7 5-9
Edges with weight: 2: 1-2 6-10 7-10
Edges with weight: 3: 4-6 7-9 8-11
Edges with weight: 4: 3-6 5-8 6-8 10-11
Edges with weight: 5: 1-3 3-5
Edges with weight: 6: 9-11
Edges with weight: 7: 1-4 2-5 2-7

New Tree: 4-7 5-9 1-2 6-10 7-10 7-9 8-11 3-6 5-8 1-3
```

Висновок: в результаті проведеної роботи ми набули практичних вмінь та навичок з використання алгоритмів Пріма і Краскала, також ознайомились з основами теорії Графів.