**Міністерство освіти і науки України**

**Національний університет  
“Львівська політехніка”**

**Кафедра систем штучного інтелекту**

**Лабораторна робота № 4**

з дисципліни

«Дискретна математика»

**Виконав**:

студент групи КН-115  
Поставка Маркіян

**Викладач:**

Мельникова Н. І.

Львів – 2019р.

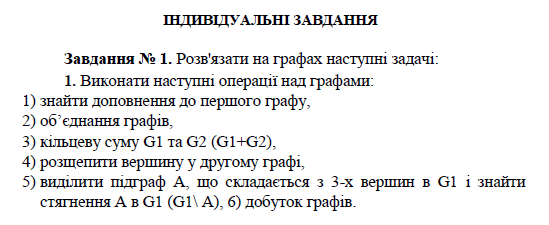
**Тема**: “Основні операції над графами. Знаходження остова мінімальної ваги за алгоритмом Прима-Краскала”

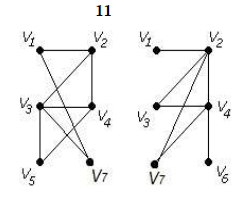
**Мета роботи:** набуття практичних вмінь та навичок з використання алгоритмів Прима і Краскала.

**Додаток 1:**

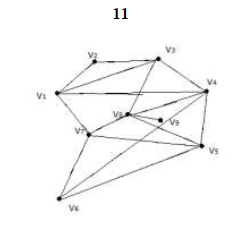
Варіант 11.

Постановка задачі:

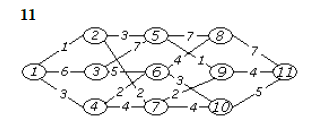






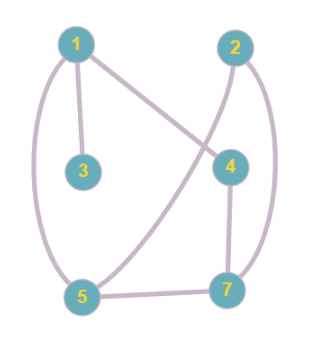




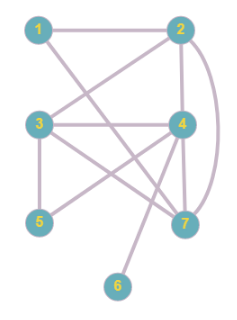


Рішення:

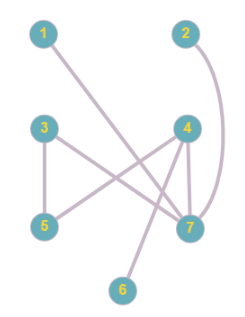
**1.  
 1)** Доповнення до першого графа



**2)** Об’єднання графів



**3)** Кільцева сума графів



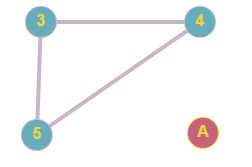
**4)** Розщеплення вершини у другому графі

Я розщепив вершину 7:

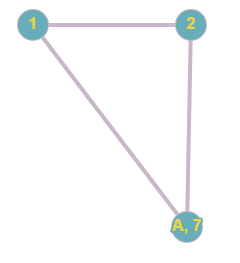


**5)** Виділити підграф А з 3х вершин у першому графі і знайти його стягнення до цього графа

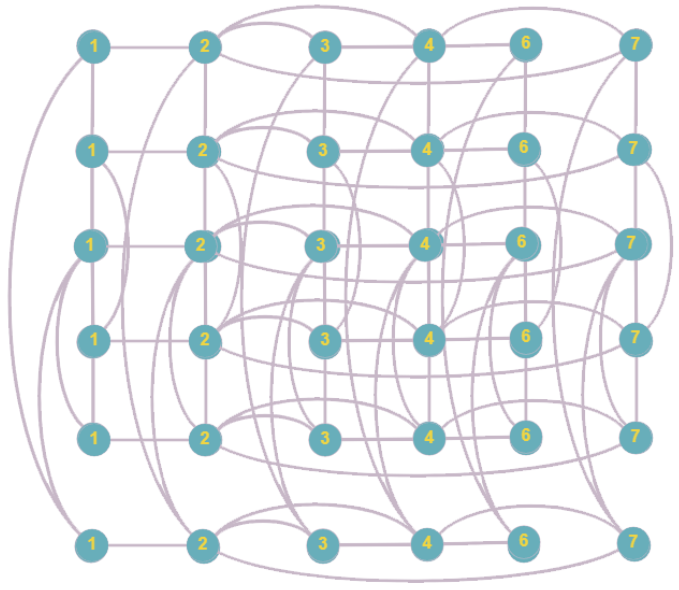
Стягнемо вершини 3, 4, 5 в А.



А стягнемо до першого графа в вершину 7.



**6)** Добуток графів



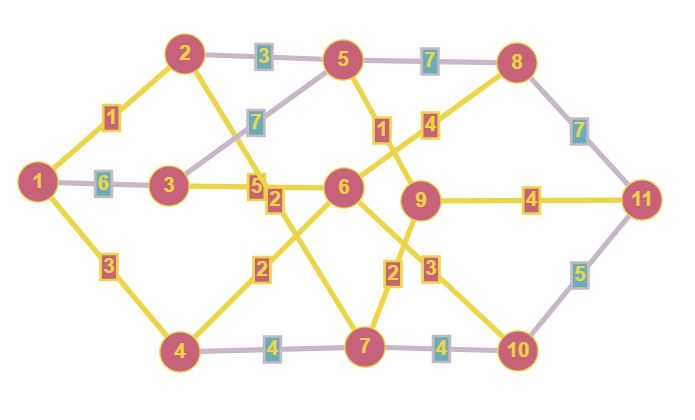
**2.**

Таблиця суміжності:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Вершини | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 2 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 4 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 5 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 6 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 7 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 8 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

Діаметр графа: 3

**3.**

****

Краскала:

1. Вибираємо будь-яку вершину і знаходимо найкоротше ребро, що з’єднує її з іншою вершиною. Помічаємо вершину як додану до графу.
2. Послідовно проходимося по всіх вершинах і для кожної знаходимо найменше ребро з сусідньою, записуючи їх доданими до дерева.
3. Робимо це поки всі вершини не будуть пройдені.
4. Отриманий дерево і буде кістяковим деревом.

Прима:

1. Сортуємо ребра по зростанню.
2. Беремо найменше ребро і помічаємо вершини як додані до графа.
3. Перевіряємо суміжні вершини циклом по зростанню ваг ребер з ними на приналежність до графа. Якщо одна вершина належить графу а інша ні, то додаємо її.
4. Отриманий дерево і буде кістяковим деревом.

**Додаток 2:**

Постановка задачі:





Код програми:

// Laba4.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include <iostream>

#include <conio.h>

using namespace std;

struct graph

{

int a;

int b;

int w;

bool ck\_a;

bool ck\_b;

graph()

{

}

graph(int a, int b, int w) : a{ a }, b{ b }, w{ w }

{

this->ck\_a = 0;

this->ck\_b = 0;

}

};

//Initialisation of graph

void initgraph(graph \*g, int nodes, int sides)

{

g[0] = { 1,2,4 };

g[1] = { 2,5,2 };

g[2] = { 5,8,7 };

g[3] = { 8,11,7 };

g[4] = { 11,10,3 };

g[5] = { 10,7,5 };

g[6] = { 7,4,4 };

g[7] = { 4,1,2 };

g[8] = { 1,3,3 };

g[9] = { 3,5,6 };

g[10] = { 5,9,5 };

g[11] = { 9,11,1 };

g[12] = { 9,7,4 };

g[13] = { 2,7,1 };

g[14] = { 3,6,7 };

g[15] = { 6,4,2 };

g[16] = { 6,10,3 };

g[17] = { 6,8,4 };

}

void sort(graph \*g, int sides)

{

graph t;

for (int j = 0; j < sides - 1; j++)

{

for (int i = 0; i < sides - 1; i++)

{

if (g[i + 1].w < g[i].w)

{

t = g[i];

g[i] = g[i + 1];

g[i + 1] = t;

}

}

}

for (int i = 0; i < sides; i++)

{

cout << i<< "\t" << g[i].a << "\t" << g[i].b << "\t" << g[i].w << "\t" << endl;

}

cout << endl;

}

void MOD\_P(graph \*g, int nodes, int sides)

{

sort(g, sides);

int \*ck\_node = new int[nodes]; // Масив перевірки на приналежність

int ck\_i = 0; // Кількість записаних

for (int i = 0; i < nodes; i++)

{

ck\_node[i] = 0;

}

cout << "Min ost tree" << endl;

cout << "A\t" << "B\t" << "Weight\t" << endl;

cout << endl;

cout << g[0].a << "\t" << g[0].b << "\t" << g[0].w << "\t" << endl;

g[0].ck\_a = 1; g[0].ck\_b = 1;

ck\_node[ck\_i] = g[0].a;

ck\_i++;

ck\_node[ck\_i] = g[0].b;

ck\_i++;

int active\_i = 0; // Значення яке провіряється

while (ck\_i < nodes)

{

for (int i = 1; i < sides; i++) // Записуємо якшо наша вершина входить в провірені

{

for (int j = 0; j < ck\_i; j++)

{

if (g[i].a == ck\_node[j])

{

g[i].ck\_a = 1;

break;

}

}

for (int j = 0; j < ck\_i; j++)

{

if (g[i].b == ck\_node[j])

{

g[i].ck\_b = 1;

break;

}

}

}

for (int i = 1; i < sides; i++) // рахуємо кількість суміжних вершин

{

if ( (g[i].ck\_a == 1 && g[i].ck\_b == 0) || (g[i].ck\_a == 0 && g[i].ck\_b == 1) )

{

if (g[i].ck\_a == 1)

{

cout << g[i].a << "\t" << g[i].b << "\t" << g[i].w << "\t" << endl;

g[i].ck\_b = 1;

ck\_node[ck\_i] = g[i].b;

ck\_i++;

break;

}

if (g[i].ck\_b == 1)

{

cout << g[i].a << "\t" << g[i].b << "\t" << g[i].w << "\t" << endl;

g[i].ck\_a = 1;

ck\_node[ck\_i] = g[i].a;

ck\_i++;

break;

}

}

}

}

delete[] ck\_node;

}

int main()

{

int nodes = 11;

int sides = 18;

graph \*g = new graph[sides];

cout << "Initialized graph" << endl;

cout <<"Side\t" << "A\t" << "B\t" << "Weight\t" << endl;

cout << endl;

initgraph(g, nodes, sides);

MOD\_P(g, nodes, sides);

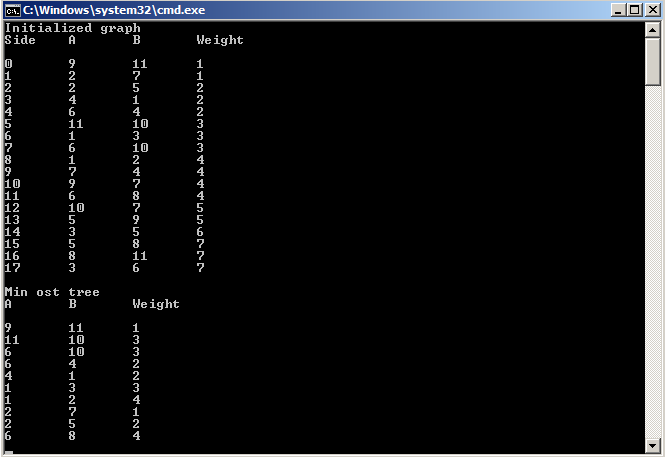
delete[] g;

\_getch();

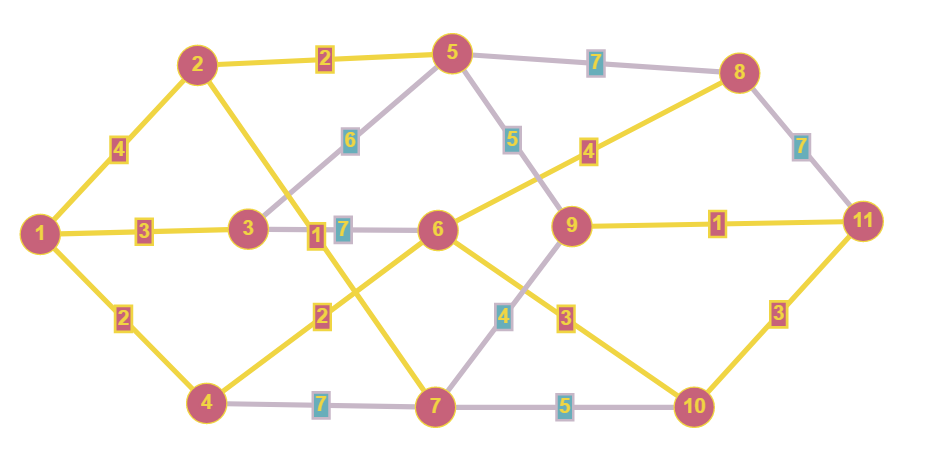
return 0;

}

Результат роботи програми:



Візуальний вигляд остового дерева з цього завдання.



**Висновок:** Янабув практичних вмінь та навичок з використання алгоритмів Прима і Краскала.