**Виконав:** Коцур Н.О.

**Група:** КН-404

**Кафедра:** САПР

**Дисципліна:** Дискретні моделі в САПР

# Варіант: 6

**Перевірив:** канд. техн. наук , доц. Кривий Р.З.

**Дата:** 16.02.2021

**GitHub:** https://github.com/kotsurnazariy/kotsur\_dusMod/tree/main/1

# ЗВІТ

до лабораторної роботи №1

на тему: АЛГОРИТМ ПОБУДОВИ ДЕРЕВ

# 1.МЕТА РОБОТИ

Метою даної лабораторної роботи є вивчення алгоритмів рішення задач побудови остових дерев.

**2. КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ**

Алгоритм Прима.

Цей алгоритм названий на честь американського математика Роберта Прима (Robert Prim), який відкрив цей алгоритм у 1957 р. Втім, ще в 1930 р. цей алгоритм був відкритий чеським математиком Войтеком Ярніком (Vojtěch Jarník). Крім того, Едгар Дейкстра (Edsger Dijkstra) в 1959 р. також винайшов цей алгоритм, незалежно від них.

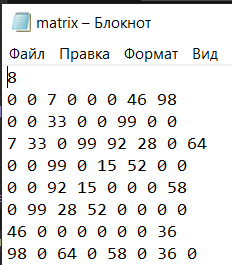
Максимальне остовне дерево

Даний зважений неорієнтований граф з вершинами і ребрами. Потрібно знайти таке піддерево цього графа, яке б з'єднувало всі його вершини, і при цьому мало найбільшу можливу вагою (тобто сумою ваг ребер). Таке піддерево називається максимальним остовим деревом. У природному постановці ця задача звучить наступним чином: є міст, iдля кожної пари відома вартість з'єднання їх дорогою (або відомо, що з'єднати їх не можна). Потрібно з'єднати всі міста так, щоб можна було доїхати з будь-якого міста в інший, а при цьому вартість прокладання доріг була б максимальною.

Сам алгоритм має дуже простий вигляд. Шуканий максимальний кістяк будується поступово, додаванням до нього ребер по одному. Спочатку остов покладається складається з єдиної вершини (її можна вибрати довільно). Потім вибирається ребро максимальної ваги, що виходить з цієї вершини, і додається в максимальне остове дерево. Після цього остов містить уже дві вершини, і тепер шукається і додається ребро максимальної ваги, що має один кінець в одній з двох обраних вершин, а інший - навпаки, у всіх інших, крім цих двох. І так далі, тобто щоразу шукається максимальне по вазі ребро, один кінець якого - вже взята в остов вершина, а інший кінець - ще не взята, і це ребро додається в остов (якщо таких ребер кілька, можна взяти будь-яке). Цей процес повторюється до тих пір, поки остов не стане містити всі вершини (або, що те ж саме, ребро).

# Завдання

Побудувати мінімальне остове дерево для графа за алгоритмом Прими



**Код программи:**

#include <iostream>

#include <cstring>

#include <vector>

#include <fstream>

using namespace std;

#define INF 9999999

int main()

{

ifstream myfile("matrix.txt");

int size;

myfile >> size;

if (myfile.is\_open())

{

vector<vector<int>> matrix(size, vector<int>(size));

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

{

for (int j = 0; j < size; j++)

{

myfile >> matrix[i][j];

}

}

int no\_edge;

vector<int>selected(size);

no\_edge = 0;

selected[0] = true;

int x;

int y;

cout << "Edge" << " : " << "Weight";

cout << endl;

int result\_weight = 0;

while (no\_edge < size - 1) {

int min = INF;

x = 0;

y = 0;

for (int i = 0; i < size; i++) {

if (selected[i]) {

for (int j = 0; j < size; j++) {

if (!selected[j] && matrix[i][j]) {

if (min > matrix[i][j]) {

min = matrix[i][j];

x = i;

y = j;

}

}

}

}

}

cout << x << " - " << y << " : " << matrix[x][y];

result\_weight += matrix[x][y];

cout << endl;

selected[y] = true;

no\_edge++;

}

cout << endl;

cout << "Result weight: " << result\_weight << endl;

}

else

{

cout << "File couldn't be open";

}

return 0;

}

# Код програми

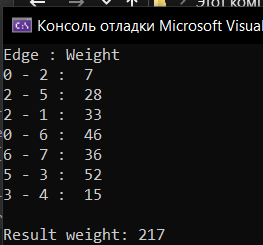


Рис.1. Результат виконання програми

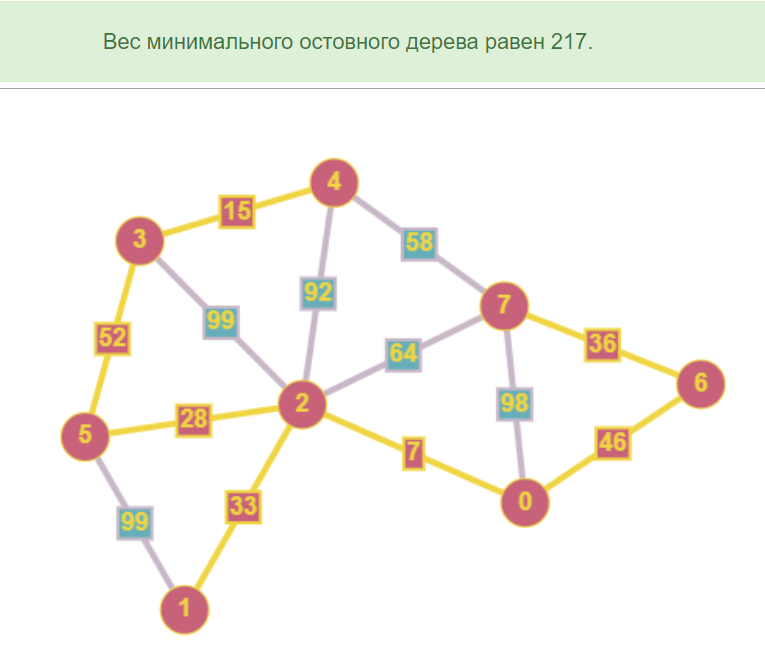


Рис.2. Візуалізація

## Висновок

На лабораторній роботі розглянуто алгоритми для побудови остових дерев для неорієнтовaних графів. В ході роботи мною було розроблено програмну реалізацію алгоритму Прима.