МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА

Кафедра систем штучного інтелекту

Лабораторна робота №4

з дисципліни «Дискретна математика»

Виконала:

студентка групи КН-115

Попів Христина

Викладач:

Мельникова Н.І.

Львів – 2019 р.

Варіант 10

Завдання № 1. Виконати наступні операції над графами:

1) знайти доповнення до першого графу,

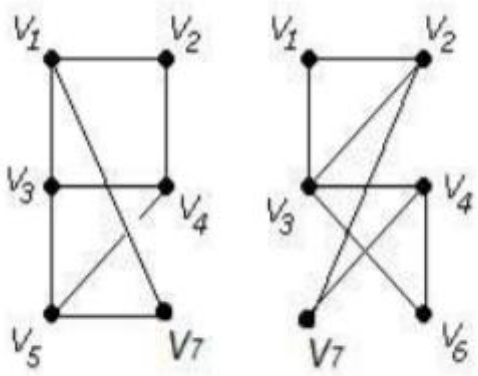
2) об’єднання графів

3) кільцеву суму G1 та G2 (G1+G2),

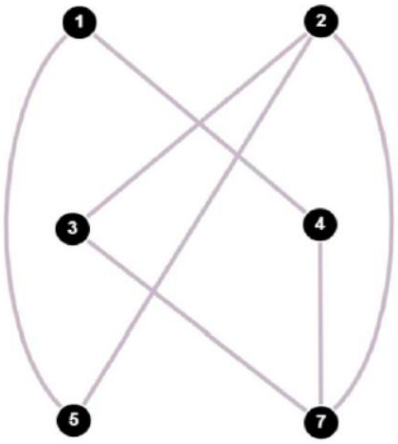
4) розщепити вершину у другому графі,

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

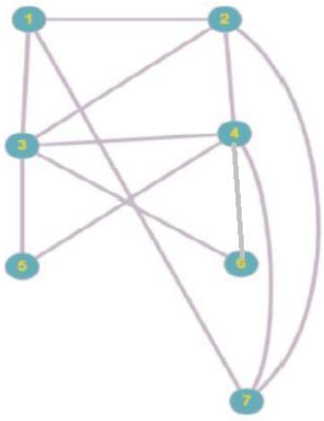
6) добуток графів.



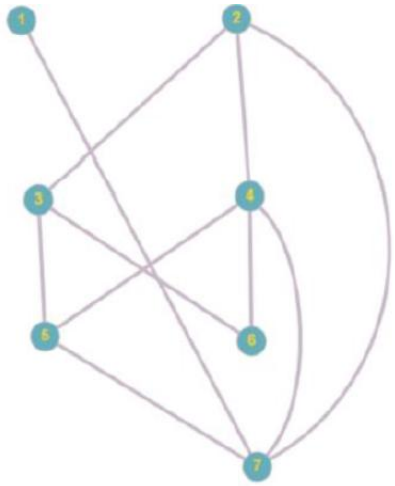
Доповнення до першого графу:



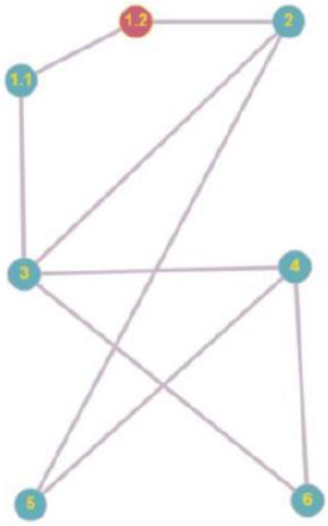
Обє’єднання графів:



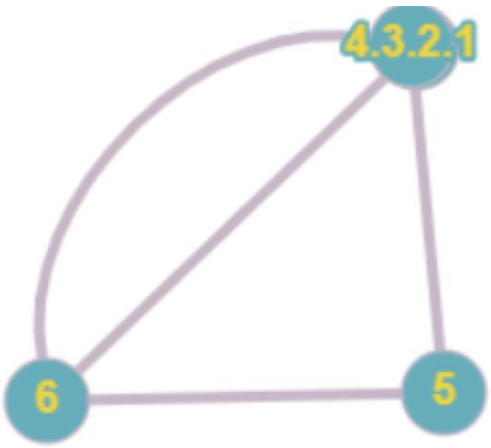
Кільцева суму G1 та G2 (G1+G2):



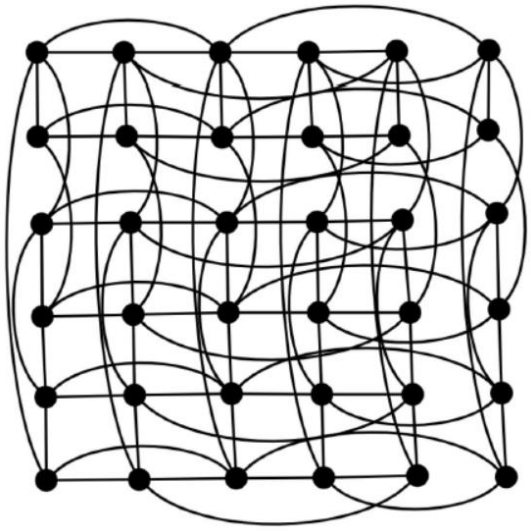
Розщепити вершину у другому графі:



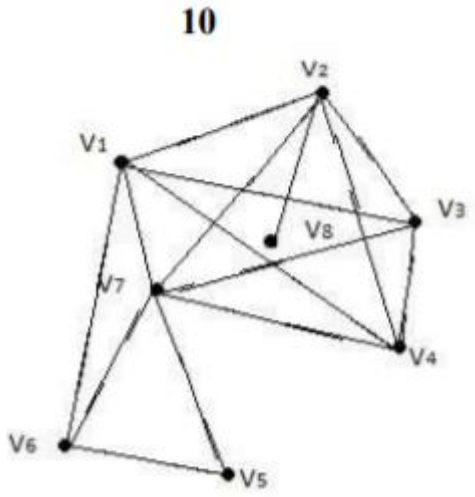
Виділити підграф A, що складається з 3-х вершин в G1 і знайти стягнення A в G1 (G1\ A):



Добуток графів:



2. Знайти таблицю суміжності та діаметр графа.



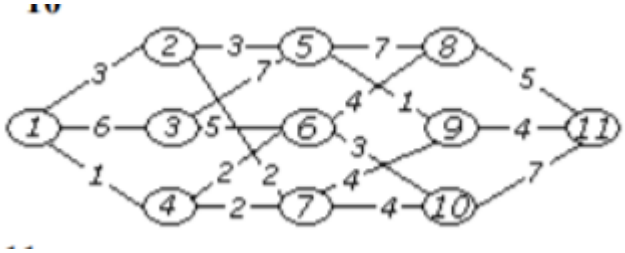
Розв’язання

Таблиця суміжності:

	v1	v2	v3	v4	v5	v6	v7	v8
v1	0	1	1	1	0	1	1	0
v2	1	0	1	1	0	0	1	1
v3	1	1	0	1	0	0	1	0
v4	1	1	1	0	0	0	1	0
v5	0	0	0	0	0	1	1	0
v6	1	0	0	0	1	0	1	0
v7	1	1	1	1	1	1	0	0
v8	0	1	0	0	0	0	0	0

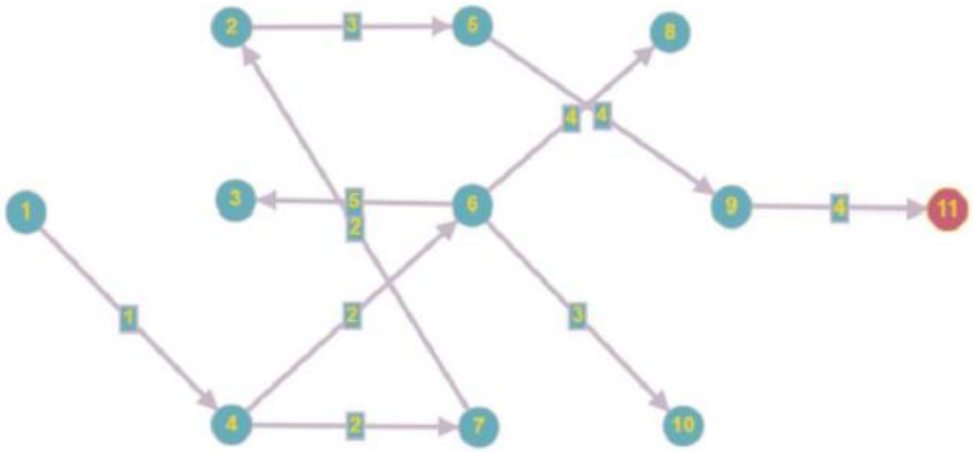
Діаметр графа: 3.

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

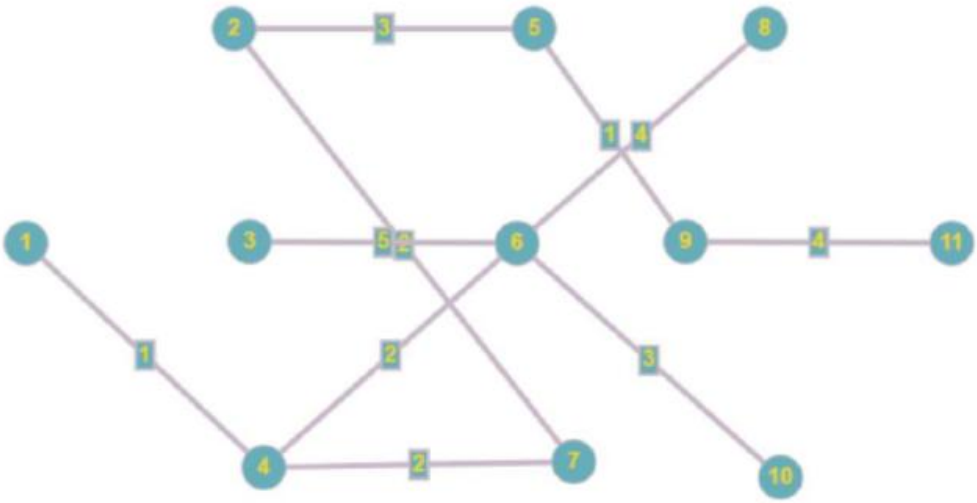


Розв’язання

Прима:



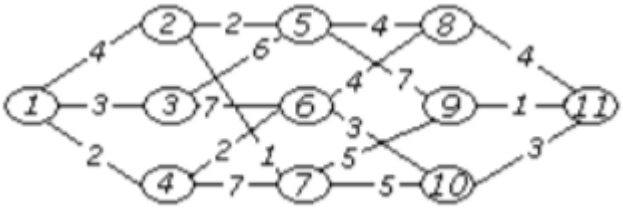
Краскала:



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

Варіант 10

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




```
#include <iostream>
using namespace std;
struct Rib
{
	int v1, v2, weight;
}Graph[100];
struct sort_rib {
	int v1;
	int v2;
	int weight;
}sort;
void Fill_Struct(int number_of_ribs) {
	for (int i = 0; i < number_of_ribs; i++) {
		cout << "Firts point: ";
		cin >> Graph[i].v1;
		cout << "Second point: ";
		cin >> Graph[i].v2;
		int sort;
		if (Graph[i].v1 > Graph[i].v2) {
			sort = Graph[i].v1;
			Graph[i].v1 = Graph[i].v2;
			Graph[i].v2 = sort;
		}
		cout << "The rib [" << Graph[i].v1 << ";" << Graph[i].v2 << "] = ";
		cin >> Graph[i].weight;
		cout << endl;
	}
}
void Sort_Sructure(int number_of_ribs) {
	for (int s = 1; s < number_of_ribs; s++) {
		for (int i = 0; i < number_of_ribs - s; i++) {
			if (Graph[i].weight > Graph[i + 1].weight) {
				sort.v1 = Graph[i].v1;
				sort.v2 = Graph[i].v2;
				sort.weight = Graph[i].weight;
				Graph[i].v1 = Graph[i + 1].v1;
				Graph[i].v2 = Graph[i + 1].v2;
				Graph[i].weight = Graph[i + 1].weight;
				Graph[i + 1].v1 = sort.v1;
				Graph[i + 1].v2 = sort.v2;
				Graph[i + 1].weight = sort.weight;
			}
		}
	}
```

```
}
void Show_Struct(int number_of_ribs) {
	for (int i = 0; i < number_of_ribs; i++) {
		cout << "The rib [" << Graph[i].v1 << ";" << Graph[i].v2 << "] = " <<
			Graph[i].weight << endl;
	}
}
void Algo_Kraskala(int number_of_ribs, int amount_of_points)
{
	int weighttree = 0;
	int* parent = new int[amount_of_points];
	int v1, v2, weight;
	int to_change, changed;
	for (int i = 0; i < amount_of_points; i++)
	{
		parent[i] = i;
	}
	for (int i = 0; i < number_of_ribs; i++)
	{
		v1 = Graph[i].v1;
		v2 = Graph[i].v2;
		weight = Graph[i].weight;
		if (parent[v2] != parent[v1])
		{
			cout << "The rib [" << Graph[i].v1 << ";" << Graph[i].v2 << "] = " <<
				Graph[i].weight << endl;
			weighttree += weight;
			to_change = parent[v1];
			changed = parent[v2];
			for (int j = 0; j < amount_of_points; j++)
			{
				if (parent[j] == changed)
				{
					parent[j] = to_change;
				}
			}
		}
	}
	delete[] parent;
	cout << "The weight of the tree: " << weighttree;
}
int main() {
	cout << "Enter an amount of points" << endl;
	int q;
	cin >> q;
	int amount_of_points = q + 1;
	cout << "Enter a number of ribs" << endl;
	int number_of_ribs;
	cin >> number_of_ribs;
	Fill_Struct(number_of_ribs);
	//Show_Struct(number_of_ribs);
	Sort_Sructure(number_of_ribs);
	cout << "After sorting" << endl;
	Show_Struct(number_of_ribs);
	cout << "Tree" << endl;
	Algo_Kraskala(number_of_ribs, amount_of_points);}
```

Результат виконання: