МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота №4

з дисципліни

«Дискретна математика»

Виконав: студент групи
КН-113

ПІ студента: Сидорук Михайло
Викладач: Мельникова Н.І.

Львів 2019

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 12

Завдання № 1.

Розв'язати на графах наступні задачі:

1. Виконати наступні операції над графами:

1) знайти доповнення до першого графу,

2) об’єднання графів,

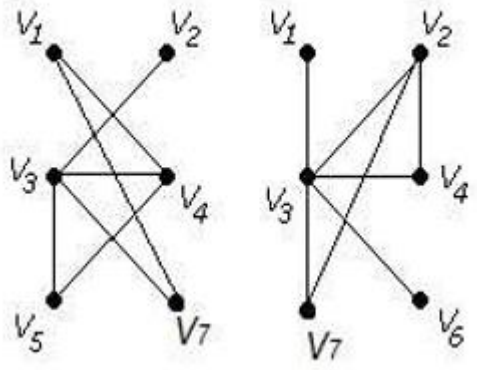
3) кільцеву суму G1 та G2 (G1+G2),

4) розщепити вершину у другому графі,

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

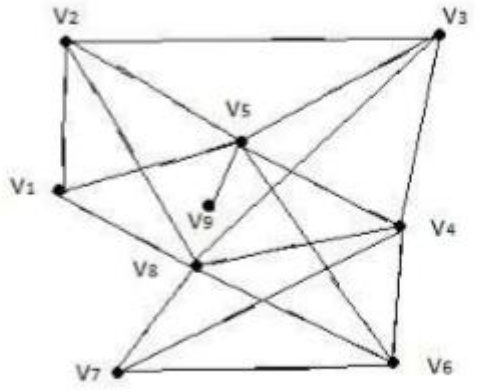
6) добуток графів.

12



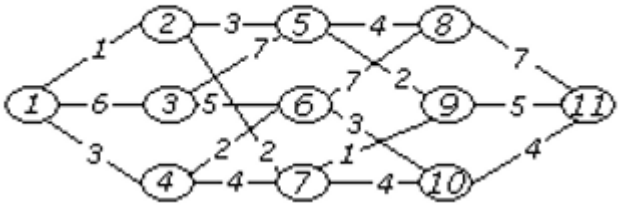
2. Знайти таблицю суміжності та діаметр графа.

12



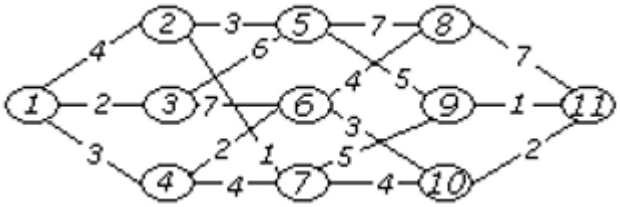
3. Знайти двома методами (Краскала і Прима) мінімальне остове дерево графа.

12



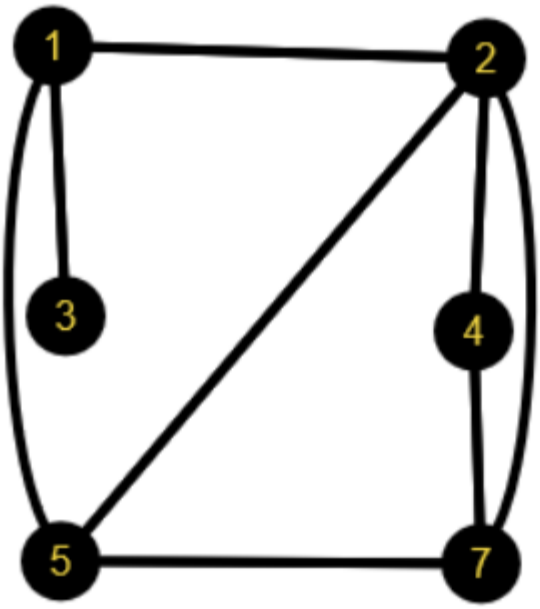
Завдання №2.

Написати програму, яка реалізує алгоритм знаходження остового дерева мінімальної ваги згідно свого варіанту.

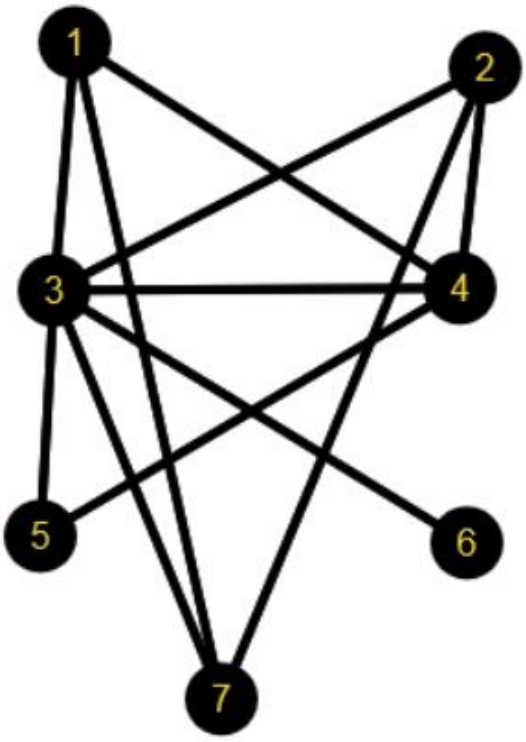


Розв’язок завдання 1

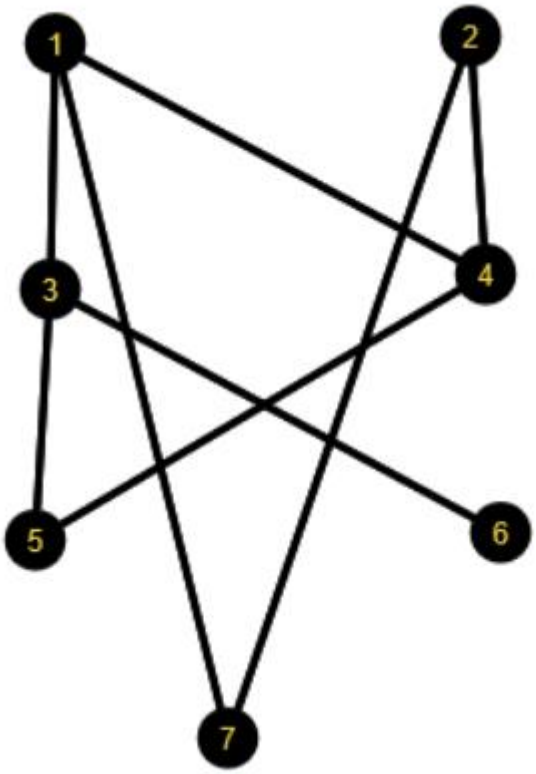
1.1



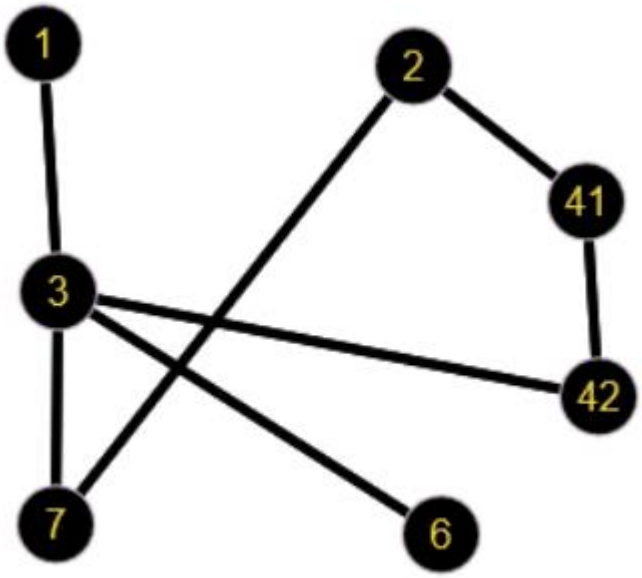
1.2



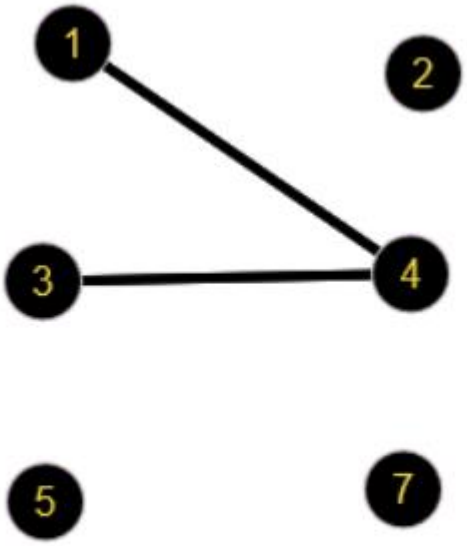
1.3



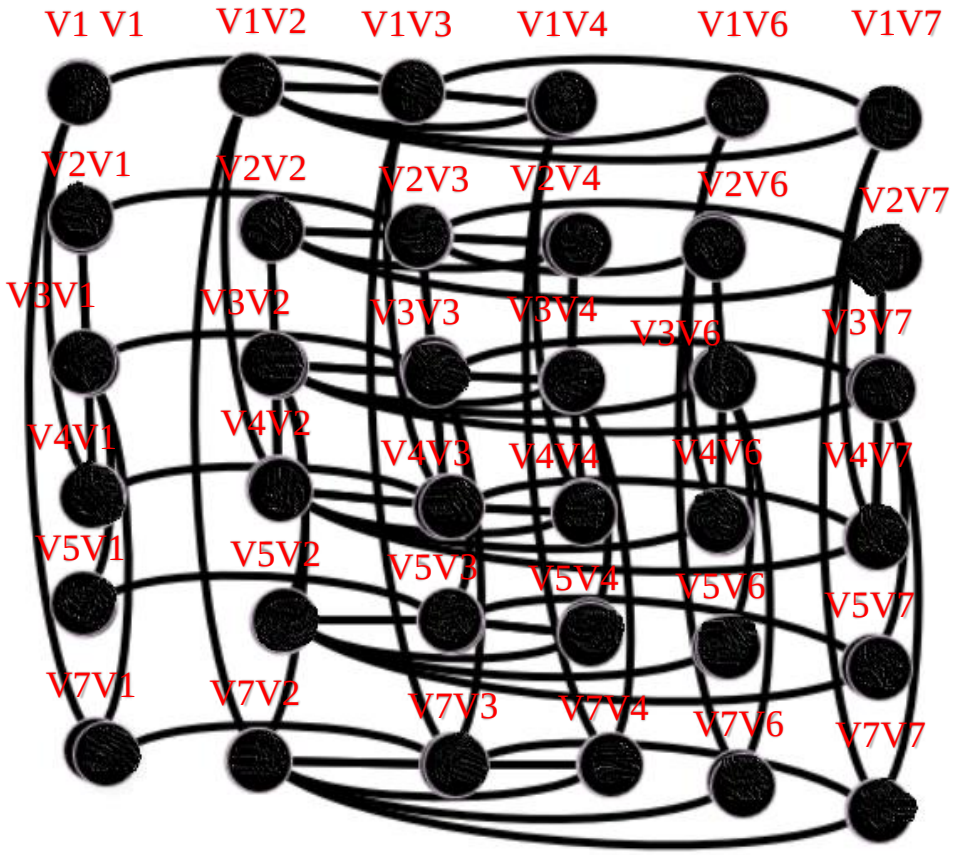
1.4



1.5



1.6



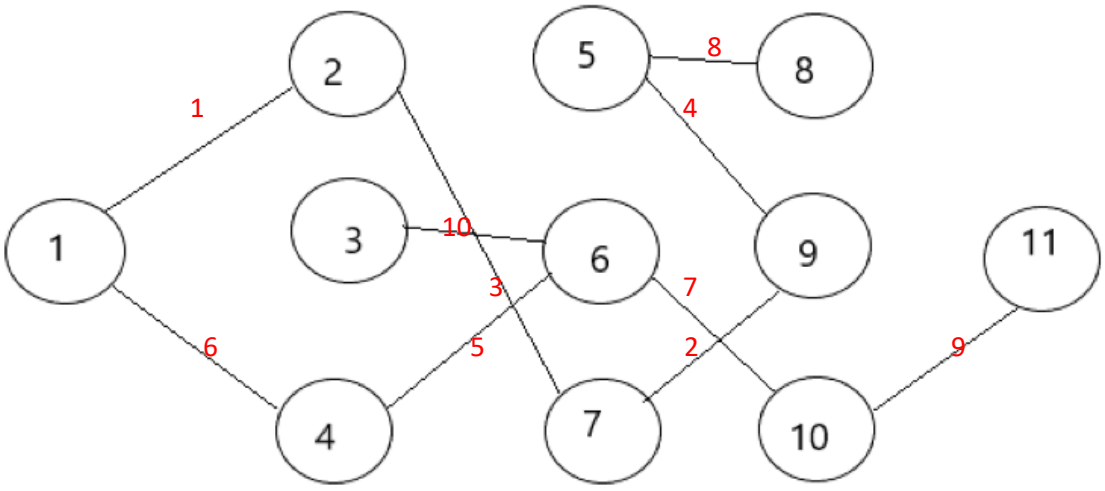
2.1

0	1	0	0	1	0	0	1	0
1	0	1	1	1	0	0	1	0
0	1	0	1	1	0	0	1	0
0	1	1	0	1	1	1	1	0
1	1	1	1	0	1	0	0	1
0	0	0	1	1	0	1	1	0
0	0	0	1	0	1	0	1	0
1	1	1	1	0	1	1	0	0
0	0	0	0	1	0	0	0	0

2.2 Діаметр графа – 4 (9->5->4->7)

3.1 Метод Краскала

Спочатку сортуємо ребра графа по їх вазі по зростанню. Починаєм включати ребра, слідкуючи, щоб не утворювався цикл. Якщо включивши ребро ми утворимо цикл то не включаємо його. В кінці ми утворимо такий граф:



3.2 Метод Прима

Сортуємо ребра графа по їх вазі по зростанню. Починаємо включати ребра, взявши будь-яку вершину графа і включивши найменше ребро до неї. Наступне ребро включаємо до графа по такому ж принципу, слідкуємо за тим, щоб не утворився цикл. Беремо за початок вершину 1 і включаємо до дерева такі ребра:

{1, 2}, {2, 7}, {7, 9}, {5, 9}, {1, 4}, {4, 6}, {6, 10}, {5, 8}, {10, 11}, {3, 6} (див. малюнок у завданні 3.1.)

Розв’язок завдання 2

```
#include "pch.h"
#include <iostream>
```

```
#include <stdio.h>
using namespace std;
struct rebro {

	int leng;
	int v1;
	int v2;
	bool in = false;
};
struct mas {

	int arr[11] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
	int c = 0;
};

int in(rebro *reb, int n) {
	setlocale(LC_ALL, "Ukrainian");
	for (int i = 0; i < n; i++)
	{
		cout << "Введіть довжину " << i + 1 << " ребра: ";
		cin >> reb[i].leng;
		cout << "Введіть першу суміжну вершину з " << i + 1 << " ребром: ";
		cin >> reb[i].v1;
		cout << "Введіть другу суміжну вершину з " << i + 1 << " ребром: ";
		cin >> reb[i].v2;
		cout << endl;
	}
	return 0;
}
int main() {

	setlocale(LC_ALL, "Ukrainian");
	int n = 0, x = 100, y = 100;

	cout << "Введіть кількість ребер у графі: ";
	cin >> n;
	int z;
	cout << "Введіть кількість вершин у графі: ";
	cin >> z;
	cout << endl;

	rebro *reb = new rebro[n];
	mas inn[5];

	in(reb, n);

	for (int i = 0; i < n - 1; i++)
	{
		for (int j = 0; j < n - 1; j++)
		{
			if (reb[j].leng > reb[j + 1].leng) { swap(reb[j].leng, reb[j + 1].leng);
swap(reb[j].v1, reb[j + 1].v1); swap(reb[j].v2, reb[j + 1].v2); }
		}
	}

	/*for (int i = 0; i < n; i++)
		cout << reb[i].leng << " " << reb[i].v1 << " " << reb[i].v2 << endl;*/


	int c = -1;
	for (int i = 0; i < n; i++)
```

```
	{
		for (int j = 0; j < 5; j++)
		{
			for (int k = 0; k < z; k++)
			{
				if (reb[i].v1 == inn[j].arr[k]) { x = j; goto point0;; }
			}
		}
	point0:;

		for (int j = 0; j < 5; j++)
		{
			for (int k = 0; k <z; k++)
			{
				if (reb[i].v2 == inn[j].arr[k]) { y = j; goto point1; }
			}
		}
	point1:;
		if (x != y && x == 100) { inn[y].arr[inn[y].c] = reb[i].v1; inn[y].c++; }

		if (x != y && y == 100) { inn[x].arr[inn[x].c] = reb[i].v2; inn[x].c++; }

		if (x != y && x != 100 && y != 100) {
			if (x < y) {
				for (int l = 0; l < inn[y].c; l++)
				{
					inn[x].arr[inn[x].c+l] = inn[y].arr[l];
					inn[y].arr[l] = 0;
				}
				inn[x].c += inn[y].c;
				inn[y].c = 0;
			}

			if (y < x) {
				for (int l = 0; l < inn[x].c; l++)
				{
					inn[y].arr[inn[y].c+l] = inn[x].arr[l];
					inn[x].arr[l] = 0;
				}
				inn[y].c += inn[x].c;
				inn[x].c = 0;
			}
		}
		if (x == 100 && y == 100) { c++;  inn[c].arr[inn[c].c] = reb[i].v1;
inn[c].arr[inn[c].c + 1] = reb[i].v2; inn[c].c += 2; }

		reb[i].in = true;

		if (x == y && x != 100) { reb[i].in = false; }

		x = 100; y = 100;

		/*for (int j = 0; j < 5; j++)
		{
			for (int k = 0; k < 11; k++)
				cout << inn[j].arr[k] << " ";
			cout << endl;
		}
		cout << endl;*/
	}
```

```
	cout << "Щоб побудувати остове дерево мінімальної ваги, ми повинні включити в нього такі ребра: " << endl;

	int s = 0;;

	for (int i = 0; i < n; i++)
	{
		if (reb[i].in == true) { cout << "Ребро" << ", що сполучає вершини " << reb[i].v1 << " " << reb[i].v2 << endl; s += reb[i].leng; }
	}
	cout << "Остове дерево мінімальної ваги для даного графа: " << s;
		return 0;}
```

Результат виконання програми:

Введіть кількість ребер у графі: 18

Введіть кількість вершин у графі: 11

Введіть довжину 1 ребра: 4

Введіть першу суміжну вершину з 1 ребром: 1

Введіть другу суміжну вершину з 1 ребром: 2

Введіть довжину 2 ребра: 2

Введіть першу суміжну вершину з 2 ребром: 1

Введіть другу суміжну вершину з 2 ребром: 3

Введіть довжину 3 ребра: 3

Введіть першу суміжну вершину з 3 ребром: 1

Введіть другу суміжну вершину з 3 ребром: 4

Введіть довжину 4 ребра: 3

Введіть першу суміжну вершину з 4 ребром: 2

Введіть другу суміжну вершину з 4 ребром: 5

Введіть довжину 5 ребра: 7

Введіть першу суміжну вершину з 5 ребром: 3

Введіть другу суміжну вершину з 5 ребром: 6

Введіть довжину 6 ребра: 4

Введіть першу суміжну вершину з 6 ребром: 4

Введіть другу суміжну вершину з 6 ребром: 7

Введіть довжину 7 ребра: 7

Введіть першу суміжну вершину з 7 ребром: 5

Введіть другу суміжну вершину з 7 ребром: 8

Введіть довжину 8 ребра: 4

Введіть першу суміжну вершину з 8 ребром: 7

Введіть другу суміжну вершину з 8 ребром: 10

Введіть довжину 9 ребра: 7

Введіть першу суміжну вершину з 9 ребром: 8

Введіть другу суміжну вершину з 9 ребром: 11

Введіть довжину 10 ребра: 1

Введіть першу суміжну вершину з 10 ребром: 9

Введіть другу суміжну вершину з 10 ребром: 11

Введіть довжину 11 ребра: 2

Введіть першу суміжну вершину з 11 ребром: 10

Введіть другу суміжну вершину з 11 ребром: 11

Введіть довжину 12 ребра: 1

Введіть першу суміжну вершину з 12 ребром: 2

Введіть другу суміжну вершину з 12 ребром: 7

Введіть довжину 13 ребра: 5

Введіть першу суміжну вершину з 13 ребром: 5

Введіть другу суміжну вершину з 13 ребром: 9

Введіть довжину 14 ребра: 3

Введіть першу суміжну вершину з 14 ребром: 6

Введіть другу суміжну вершину з 14 ребром: 10

Введіть довжину 15 ребра: 6

Введіть першу суміжну вершину з 15 ребром: 3

Введіть другу суміжну вершину з 15 ребром: 5

Введіть довжину 16 ребра: 4

Введіть першу суміжну вершину з 16 ребром: 6

Введіть другу суміжну вершину з 16 ребром: 8

Введіть довжину 17 ребра: 2

Введіть першу суміжну вершину з 17 ребром: 4

Введіть другу суміжну вершину з 17 ребром: 6

Введіть довжину 18 ребра: 5

Введіть першу суміжну вершину з 18 ребром: 7

Введіть другу суміжну вершину з 18 ребром: 9

Щоб побудувати остове дерево мінімальної ваги, ми повинні включити в нього такі ребра:

Ребро, що сполучає вершини 9 11

Ребро, що сполучає вершини 2 7

Ребро, що сполучає вершини 1 3

Ребро, що сполучає вершини 10 11

Ребро, що сполучає вершини 4 6

Ребро, що сполучає вершини 1 4

Ребро, що сполучає вершини 2 5

Ребро, що сполучає вершини 6 10

Ребро, що сполучає вершини 1 2

Ребро, що сполучає вершини 6 8

Остове дерево мінімальної ваги для даного графа: 25

Висновок: Отже, я набув практичних вмінь і навичок з використання алгоритмів Прима та Краскала.