МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ

“ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота № 4

з дисципліни

«Дискретна математика»

Виконав:

студент групи КН-112

Калітовський Роман

Викладач:

Мельникова Н.І.

Львів – 2019 р.

ЛАБОРАТОРНА РОБОТА № 4

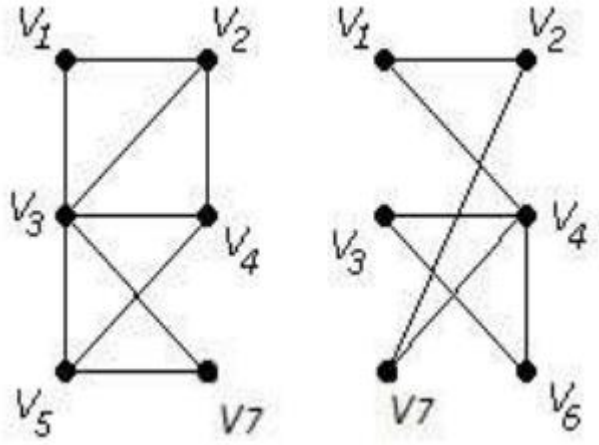
Тема: Основні операції над графами. Знаходження остова мінімальної ваги за алгоритмом Пріма-Краскала.

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

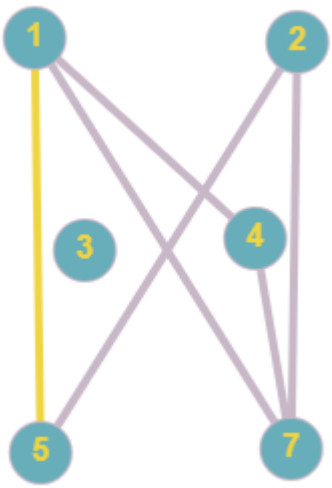
Варіант № 6

Завдання № 1. Розв'язати на графах наступні задачі:

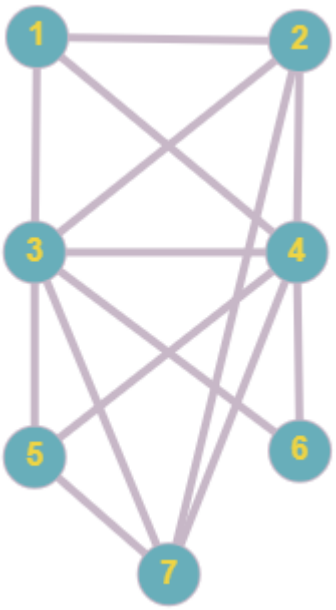
1. Виконати наступні операції над графами:



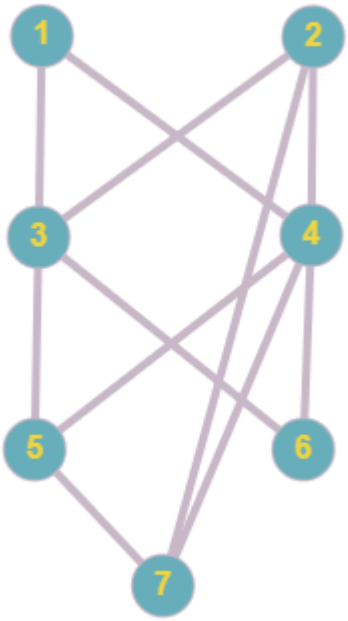
1) знайти доповнення до першого графу,



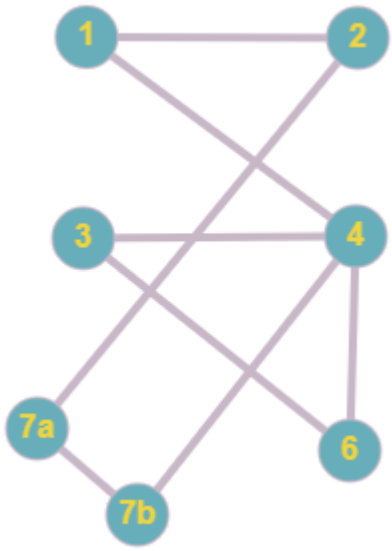
2) об’єднання графів,



3) кільцеву суму G1 та G2 (G1+G2),

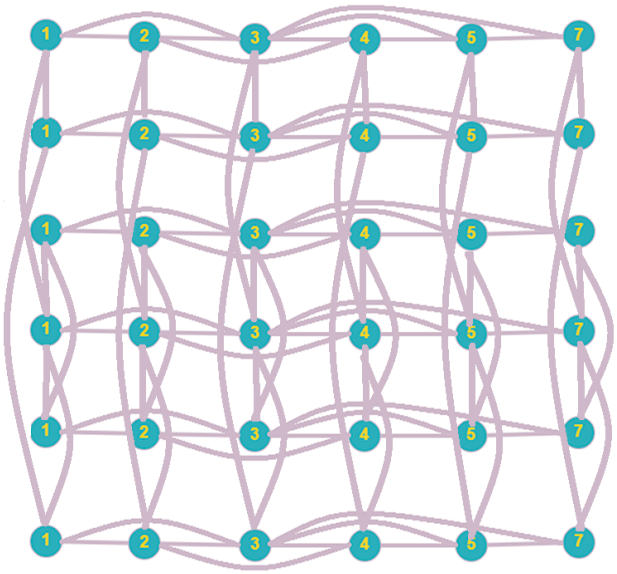


4) розщепити вершину у другому графі,

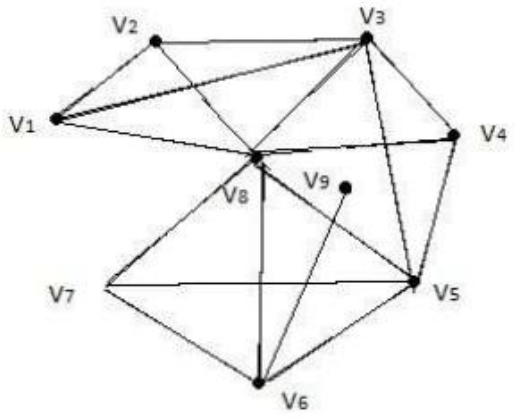


5) виділити підграф A, що складається з 3-х вершин в G1 і знайти стягнення A в G1 (G1\ A),

6) добуток графів.

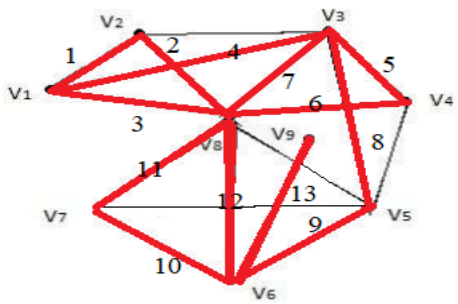


2. Знайти таблицю суміжності та діаметр графа.



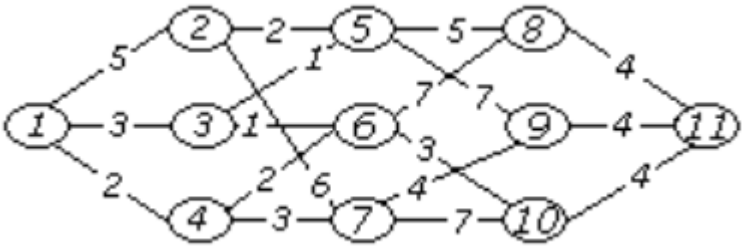
Матриця суміжності:

	1	2	3	4	5	6	7	8	9
1	0	1	1	0	0	0	0	1	0
2	1	0	1	0	0	0	0	1	0
3	1	1	0	1	1	0	0	1	0
4	0	0	1	0	1	0	0	1	0
5	0	0	1	1	0	1	1	1	0
6	0	0	0	0	1	0	1	1	1
7	0	0	0	0	1	1	0	1	0
8	1	1	1	1	1	1	1	0	0
9	0	0	0	0	0	1	0	0	0



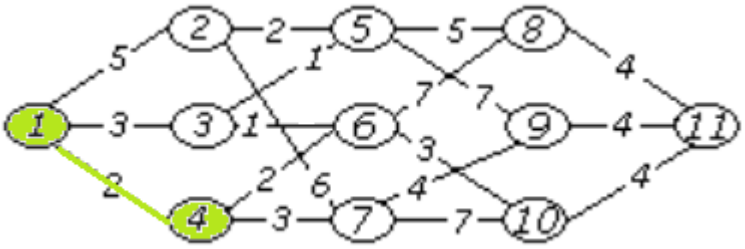
Діаметр графа = 13

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

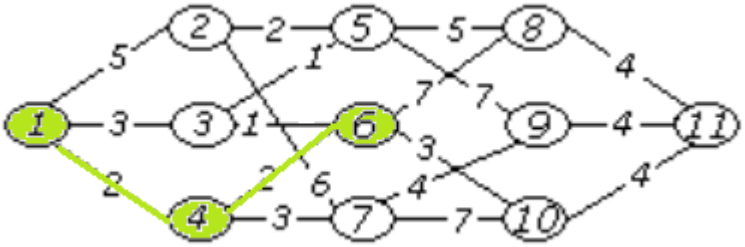


метод Прима:

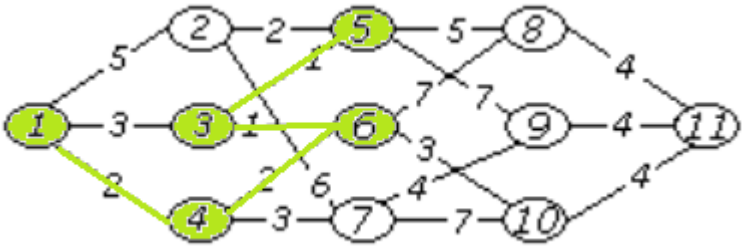
1



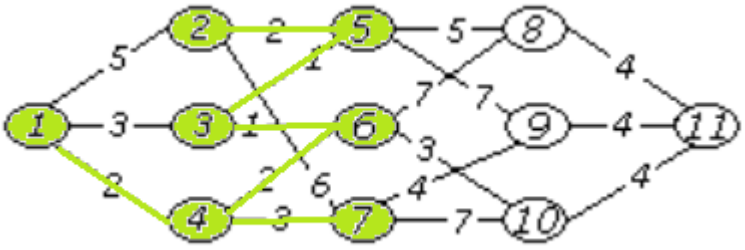
2



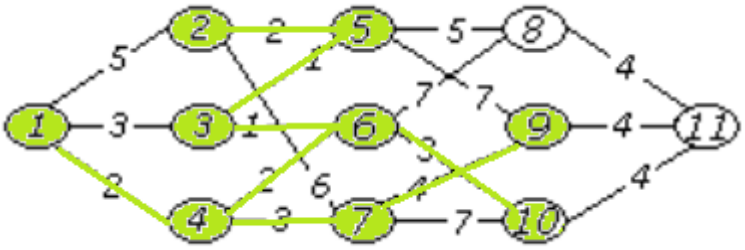
3,4



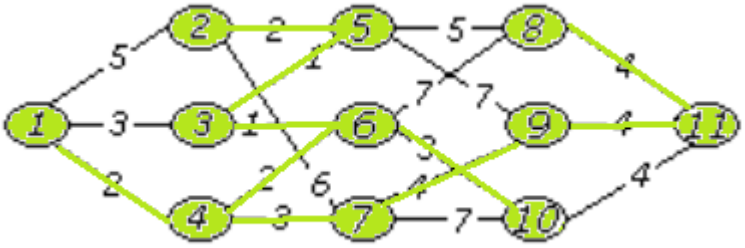
5,6



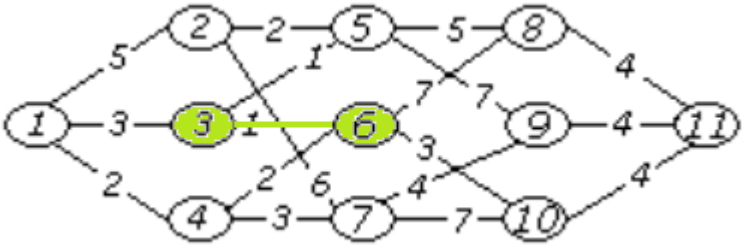
7,8



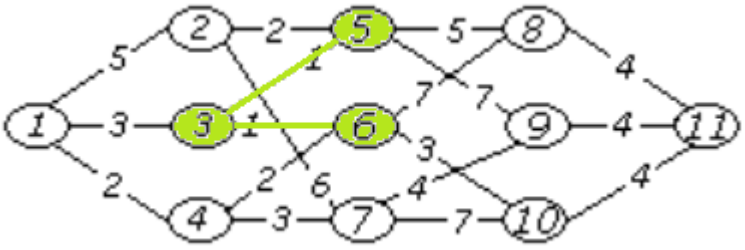
9,10



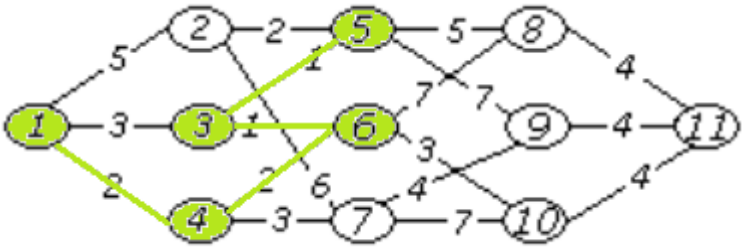
Алгоритм завершено, мінімальне остове дерево знайдено.
Метод краскала:
1



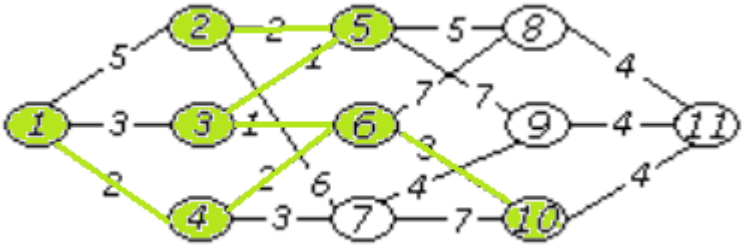
2


1
2
3
4
5
6
7
8
9
10
11

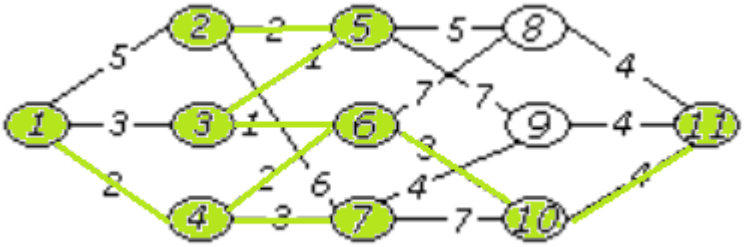
3,4


1
2
3
4
5
6
7
8
9
10
11

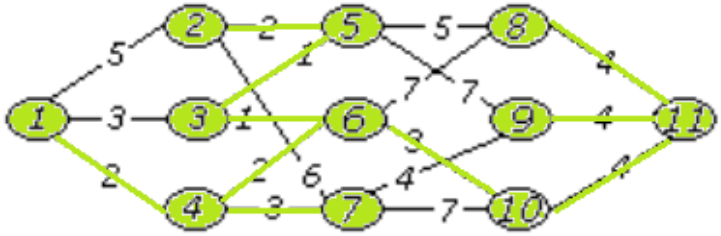
5,6


1
2
3
4
5
6
7
8
9
10
11

7,8


1
2
3
4
5
6
7
8
9
10
11

9,10

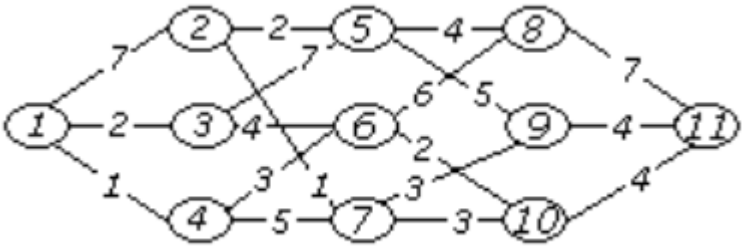


Алгоритм завершено, мінімальне остове дерево знайдено.

Завдання №2.

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

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



Матриця суміжності графа:

0	7	2	1	0	0	0	0	0	0	0
7	0	0	0	2	0	1	0	0	0	0
2	0	0	0	7	4	0	0	0	0	0
1	0	0	0	0	3	5	0	0	0	0
0	2	7	0	0	0	0	4	5	0	0
0	0	4	3	0	0	0	6	0	2	0
0	1	0	5	0	0	0	0	3	3	0
0	0	0	0	4	6	0	0	0	0	7
0	0	0	0	5	0	3	0	0	0	4
0	0	0	0	0	2	3	0	0	0	4
0	0	0	0	0	0	0	7	4	4	0

Код основної прогами:

```
#include <iostream>
#include <clocale>
#include "vector.h"
```

```
using namespace std;
int main()
{
	int numofvertices;
	setlocale(LC_ALL, "ru");
	do
	{
		cout << "Введіть кількість вершин графа:\n";
		numofvertices = 0;
		cin >> numofvertices;
		if (numofvertices < 2)
		{
			cout << "Кількість вершин графа повинна бути більше 2!\n";
		}
	} while (numofvertices < 2);
	cout << "Чудово!\nТепер введіть матрицю суміжності графа.\nВвід виконуйте так: вводьте елементи рядка, розділяючи їх проблом, а для переходу на наступний рядок натискайте Enter.\nЯкщо між вершинами є ребро, вводьте його вагу (вважається що вага - ціле число), якщо ребра немає, вводьте 0.\n";
	int** adjacencymatrix = new int* [numofvertices] {};
	for (int i = 0; i < numofvertices; i++)
	{
		adjacencymatrix[i] = new int[numofvertices] {};
	}
	for(int i=0;i< numofvertices;i++)
	{
		for (int j = 0; j < numofvertices; j++)
		{
			cin >> adjacencymatrix[i][j];
		}
	}
	for (int i = 0; i < numofvertices; i++)
	{
		for (int j = 0; j < numofvertices; j++)
		{
			if (adjacencymatrix[i][j] != adjacencymatrix[j][i])
			{
				cout << "Введена вами матриця не симетрична.Матриця суміжності графа повинна бути симетрична!\n Перевірте правильність вводу.\n";
				return 1;
			}
			if (i == j && adjacencymatrix[i][j])
			{
				cout << "На головній дагоналі повинні бути нулі. Програма не працює з графами у яких є петлі.";
				return 2;
			}
		}
```

```
	}
	//перевірка симетричності
	newedges* p = addnewedges();
	for (int j = 0; j < numofvertices; j++)
	{
		for (int i = j+1; i < numofvertices; i++)
		{
			if (adjacencymatrix[j][i])
			{
				addegdes(p, adjacencymatrix[j][i], j+1, i+1);
			}
		}
	}
	//додавання ребер в список
	printedges(p);
	newost* po = addnewost();

	while (p->size)
	{
		newedges* min1 = new newedges;
		min1->head = p->head;
		newedges* buf = new newedges;
		buf->head = p->head;
		while (buf->head != nullptr)
		{
			if(buf->head->waight< min1->head->waight)
			{
				min1->head = buf->head;
			}
			buf->head = buf->head->next;
		}

		addegdes(po,p,min1->head);
		delete min1;
		delete buf;
	}
	printost(po);
	for (int i = 0; i < numofvertices; i++)
	{
		delete[] adjacencymatrix[i];
	}
	delete[] adjacencymatrix;
}
```

Код в файлі “vector.h”

```
#pragma once
#include <stdio.h>
```

```
#include <iostream>
using namespace std;
struct edges
{
	int waight;//вага ребра
	int firstvertex;//номер першої вершини
	int secondvertex;//номер другої вершини
	edges* next; //Вказівник на наступний елемент в списку
};
struct newedges
{
	int size; // Розмір списку
	edges* head; //Вказівник на перший елемент
	edges* tail; //Вказівник на останній елемент
};
newedges* addnewedges()
{
	newedges* n = new newedges;
	n->size = 0;
	n->head = nullptr;
	n->tail = nullptr;
	return n;
}
void addegdes(newedges* l,const int waight, const int firstvertex, const int secondvertex)
{
	edges* edge = new edges;
	l->size++;
	edge->waight = waight;
	edge->firstvertex = firstvertex;
	edge->secondvertex = secondvertex;
	if (l->head == nullptr)
	{
		l->head = edge;
		l->head->next = nullptr;
		l->tail = edge;
	}
	else
	{
		l->tail->next = edge;
		l->tail = edge;
		edge->next = nullptr;
	}
}
void printedges(newedges* l)
{
	if (l->size == 0)
	{
		 cout << "\nВивід списку ребер графа:\nСписок порожній.\n";
```

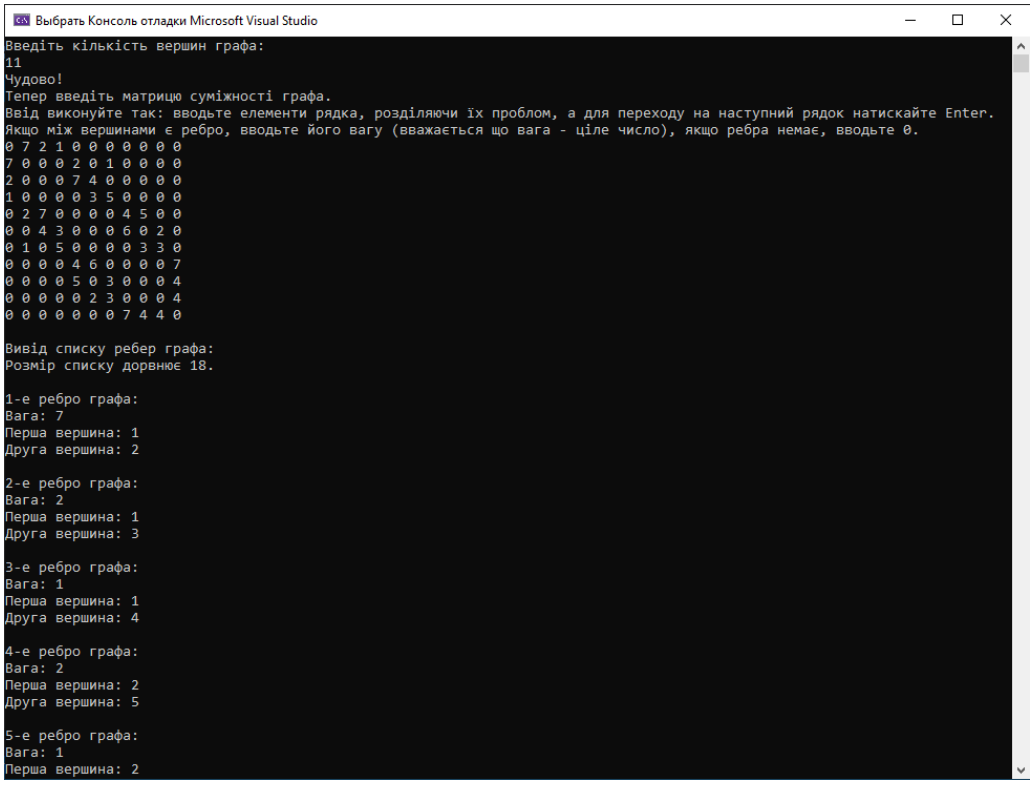
```
	}
	else
	{
		newedges* buf = new newedges;
		buf->head = l->head;
		cout << "\nВивід списку ребер графа:\nРозмір списку дорвнює " << l->size
<< "." << endl;
		int i = 1;
		while (l->head != nullptr)
		{
			cout <<endl<< i << "-е ребро графа: \n" <<"Вага: " <<l->head-
>waight <<"\nПерша вершина: "<< l->head->firstvertex <<"\nДруга вершина: "<< l->head-
>secondvertex << endl;
			l->head = l->head->next;
			i++;
		}
		l->head = buf->head;
		delete buf;
	}
}
void deleteedge(newedges* l, edges* edges1)
{
	edges* n;
	newedges* buf = new newedges;
	buf->head = l->head;

	for (int i = 0; i < l->size; i++)
	{
		if (buf->head == edges1) {
			n = buf->head;
			l->head = buf->head->next;


			delete n;
			break;
		}
		else {
			if (buf->head->next == edges1)
			{

				n = buf->head->next;
				buf->head->next = buf->head->next->next;


				delete n;
				break;
			}
		}
```

```
                buf->head=buf->head->next;
        }

        delete buf;
        l->size--;

}
struct ost
{
        int waight;//вага ребра
        int firstvertex;//номер першої вершини
        int secondvertex;//номер другої вершини
        ost* next; //Вказівник на наступний елемент в списку
};
struct newost
{
        int size; // Розмір списку
        ost* head; //Вказівник на перший елемент
        ost* tail; //Вказівник на останній елемент
};
newost* addnewost()
{
        newost* n = new newost;
        n->size = 0;
        n->head = nullptr;
        n->tail = nullptr;
        return n;
}
void addegdes(newost* thisost, newedges* l,edges* edges1)
{
        newost* buf = new newost;
        buf->head = thisost->head;
        int first = 0;
        int second = 0;
        while (buf->head != nullptr)
        {
                if (buf->head->firstvertex == edges1->firstvertex)
                {
                        first++;
                }
                if ( buf->head->secondvertex == edges1->firstvertex)
                {
                first++;
                }

                if (buf->head->firstvertex == edges1->secondvertex)
                {
                        second++;
```

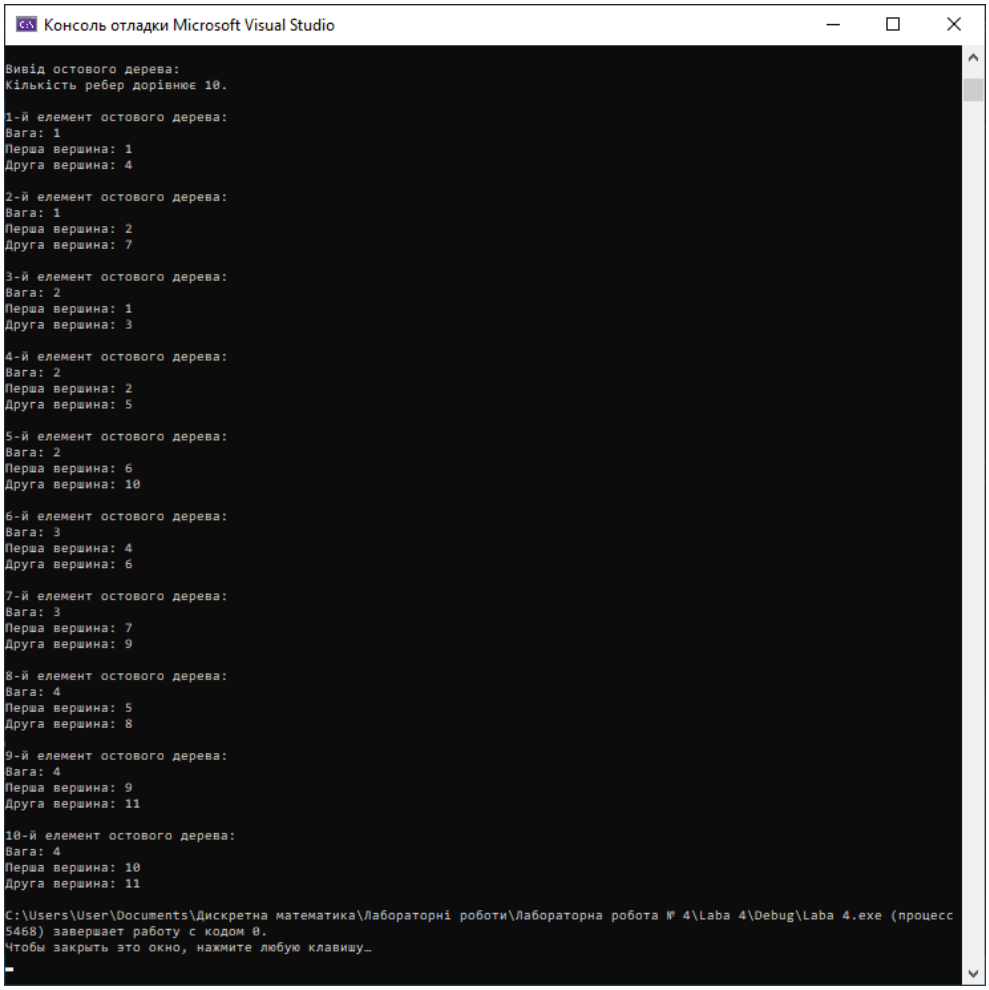
```
                }
                if ( buf->head->secondvertex == edges1->secondvertex)
                {
                second++;
                }
                buf->head = buf->head->next;
        }
        delete buf;

        if (first<2&&second<2)
        {
                ost* ost1 = new ost;
                thisost->size++;
                ost1->waight = edges1->waight;
                ost1->firstvertex = edges1->firstvertex;
                ost1->secondvertex = edges1->secondvertex;

                if (thisost->head == nullptr)
                {
                        thisost->head = ost1;
                        thisost->head->next = nullptr;
                        thisost->tail = ost1;
                }
                else
                {
                        thisost->tail->next = ost1;
                        thisost->tail = ost1;
                        ost1->next = nullptr;
                }

                deleteedge(l,edges1);
        }
        else
        {
                deleteedge(l, edges1);
        }

}
void printost(newost* thisost)
{

        if (thisost->size == 0)
        {
                cout << "\nВивід остового дерева:\nСписок порожній.\n";
        }
        else
        {
                newost* buf = new newost;
```

```
                buf->head = thisost->head;
                cout << "\nВивід остового дерева:\nКількість ребер дорівнює " << thisost->size << "." << endl;
                int i = 1;
                while (thisost->head != nullptr)
                {
                        cout <<endl<< i << "-й елемент остового дерева: \n" << "Вага: " << thisost->head->waight << "\nПерша вершина: " << thisost->head->firstvertex << "\nДруга вершина: " << thisost->head->secondvertex << endl;
                        thisost->head = thisost->head->next;
                        i++;
                }
                thisost->head = buf->head;
                delete buf;
        }
}
```

Приклад виконання програми:



Результат роботи:


Консоль отладки Microsoft Visual Studio
Вивід остового дерева:
Кількість ребер дорівнює 10.
1-й елемент остового дерева:
Вага: 1
Перша вершина: 1
Друга вершина: 4
2-й елемент остового дерева:
Вага: 1
Перша вершина: 2
Друга вершина: 7
3-й елемент остового дерева:
Вага: 2
Перша вершина: 1
Друга вершина: 3
4-й елемент остового дерева:
Вага: 2
Перша вершина: 2
Друга вершина: 5
5-й елемент остового дерева:
Вага: 2
Перша вершина: 6
Друга вершина: 10
6-й елемент остового дерева:
Вага: 3
Перша вершина: 4
Друга вершина: 6
7-й елемент остового дерева:
Вага: 3
Перша вершина: 7
Друга вершина: 9
8-й елемент остового дерева:
Вага: 4
Перша вершина: 5
Друга вершина: 8
9-й елемент остового дерева:
Вага: 4
Перша вершина: 9
Друга вершина: 11
10-й елемент остового дерева:
Вага: 4
Перша вершина: 10
Друга вершина: 11
C:\Users\User\Documents\Дискретна математика\Лабораторні роботи\Лабораторна робота № 4\Laba 4\Debug\Laba 4.exe (процесс
5468) завершает работу с кодом 0.
Чтобы закрыть это окно, нажмите любую клавишу…