МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра систем штучного інтелекту

Лабораторна робота № 4

З дисципліни: *“Дискретна математика”*

Виконав
студент групи КН 113
Карабін Я. В.

Викладач:
Мельникова Н. І.

Львів – 2019

Тема роботи

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.

Короткі теоретичні відомості

Графом G називається пара множин (V, E), де V − множина вершин, пронумерованих числами $1, 2, \dots, n = v$; $V = \{v\}$, E − множина упорядкованих або неупорядкованих пар $e = (v', v''), v' \in V\ , v'' \in V$, називаних дугами або ребрами, $E\ = \{e\}$. При цьому не має примусового значення, як вершини розташовані в просторі або площині і які конфігурації мають ребра.

Неорієнтованим графом G називається граф у якого ребра не мають напрямку. Такі ребра описуються неупорядкованою парою (v', v''). *Орієнтований граф (орграф)* – це граф ребра якого мають напрямок та можуть бути описані упорядкованою парою (v', v''). Упорядковане ребро називають дугою. Граф є *змішаним*, якщо наряду з орієнтованими ребрами (дугами) є також і неорієнтовані. При розв'язку задач змішаний граф зводиться до орграфа.

Кратними (паралельними) називаються ребра, які зв'язують одні і ті ж вершини. Якщо ребро виходить та й входить в одну і ту саму вершину, то таке ребро називається *петлею*.

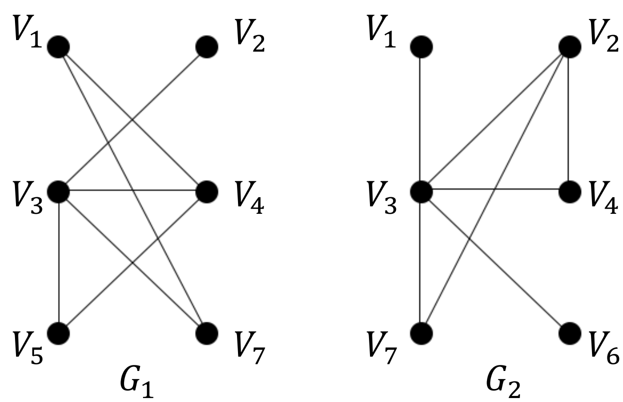
Мультиграф – граф, який має кратні ребра. *Псевдограф* – граф, який має петлі. *Простий граф* – граф, який не має кратних ребер та петель.

Граф, який не має ребер називається *пустим графом, нульграфом*. Вершина графа, яка не інцедентна до жодного ребра, називається *ізольованою*. Вершина графа, яка інцедентна тільки до одного ребра, називається *звисаючою*.

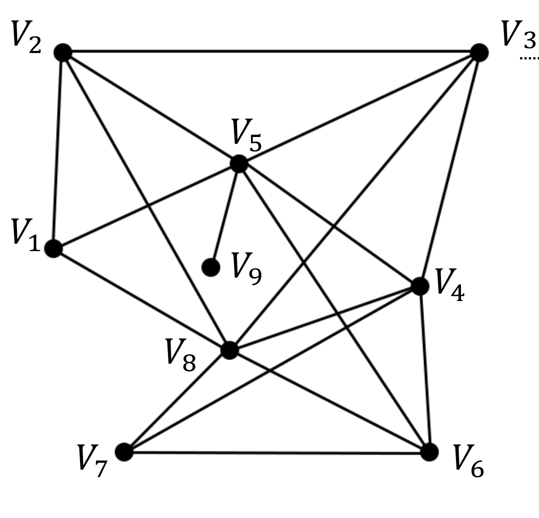
Завдання варіанту №12 з додатку 1

1. Виконати наступні операції над графами:

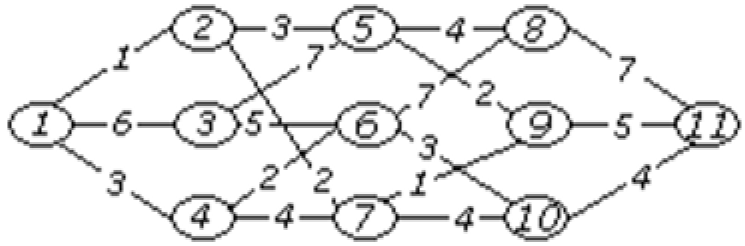
   1) знайти доповнення до першого графу,
   2) об'єднання графів,
   3) кільцеву суму $G1$ та $G2$ ($G1 \oplus G2$),
   4) розщепити вершину у другому графі,
   5) виділити підграф A, що складається з 3-х вершин в $G1$ і знайти стягнення A в $G1$ ($G1 \setminus A$),
   6) добуток графів.



2. Знайти таблицю суміжності та діаметр графа.



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



Розв'язки

1.

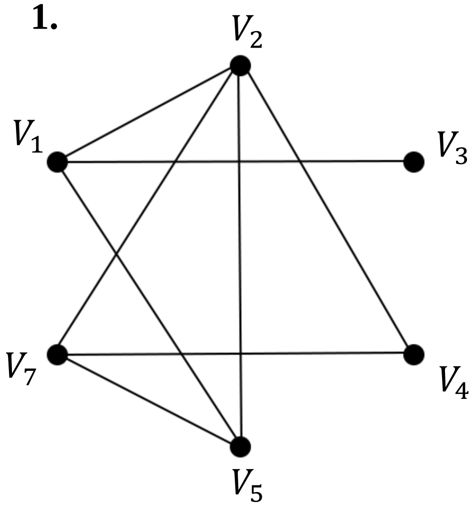


Рис. 1) $\overline{G_1}$

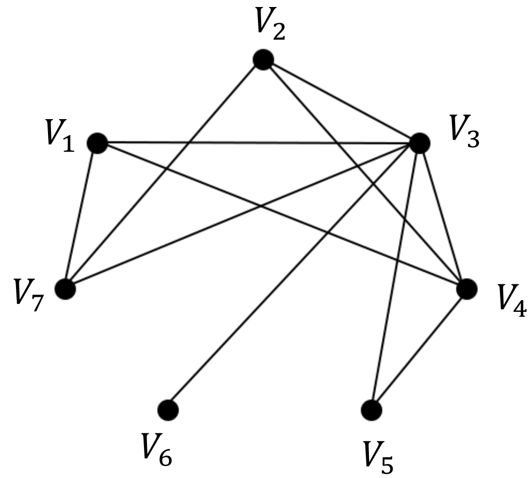


Рис. 2) $G_1 \cup G_2$

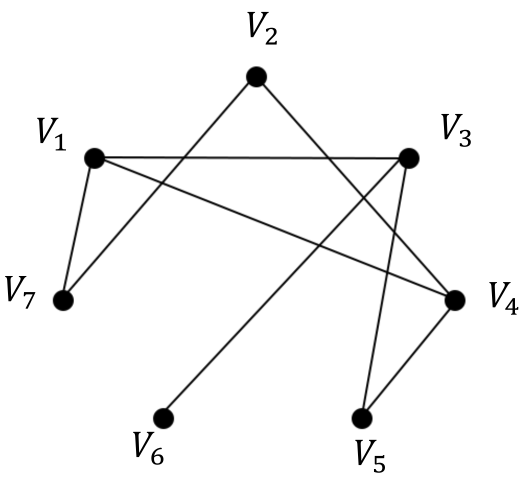


Рис. 3) $G_1 \oplus G_2$

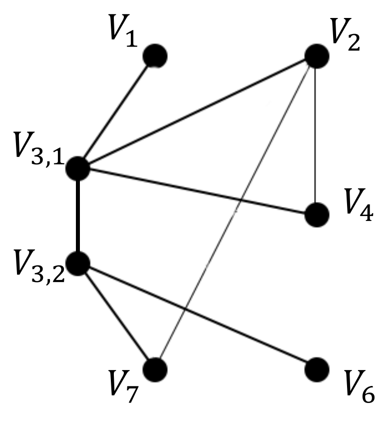


Рис. 4) Розщеплення вершини V_3

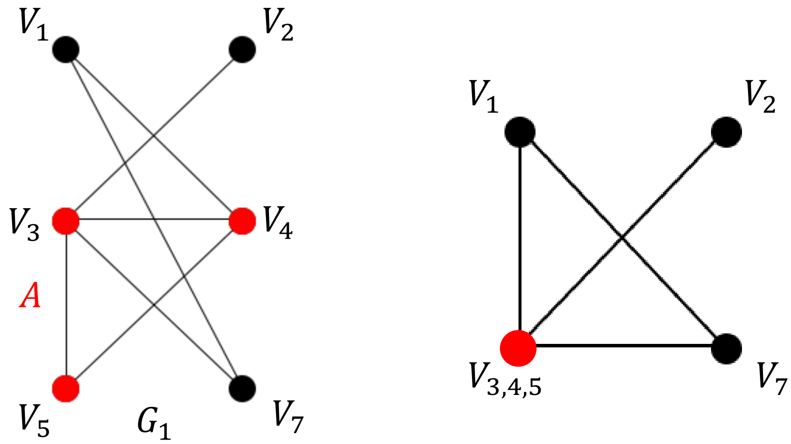


Рис. 5) Стягнення підграфа A в G_1

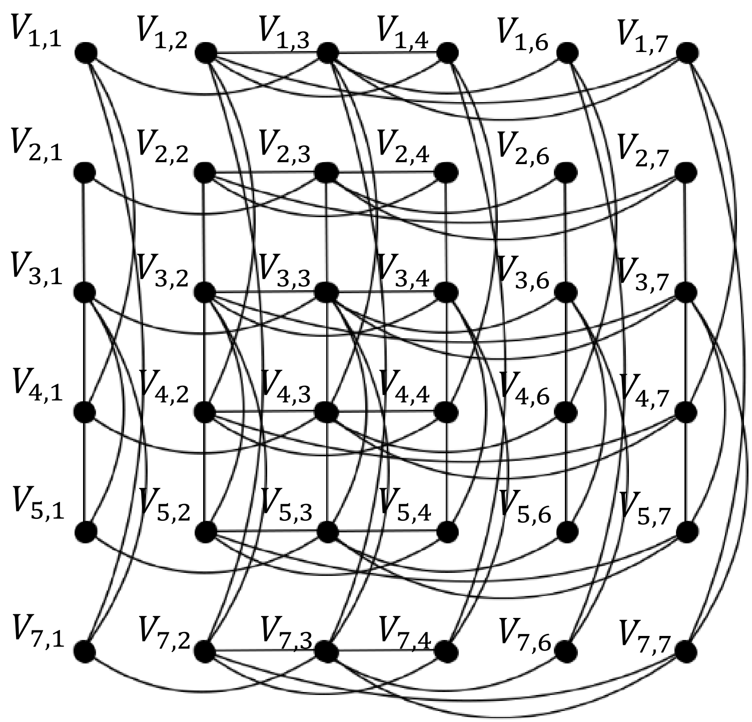


Рис. 6) Добуток графів G_1 і G_2

2.

	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9
V_1	0	1	0	0	1	0	0	1	0
V_2	1	0	1	0	1	0	0	1	0
V_3	0	1	0	1	1	0	0	1	0
V_4	0	0	1	0	1	1	1	1	0
V_5	1	1	1	1	0	1	0	0	1
V_6	0	0	0	1	1	0	1	1	0
V_7	0	0	0	1	0	1	0	1	0
V_8	1	1	1	1	0	1	1	0	0
V_9	0	0	0	0	1	0	0	0	0

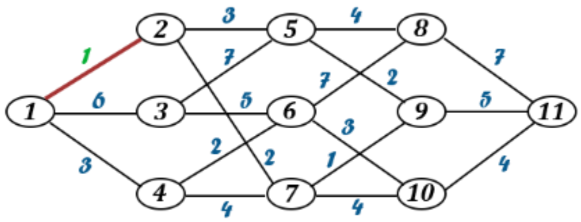
Таблиця суміжності графа

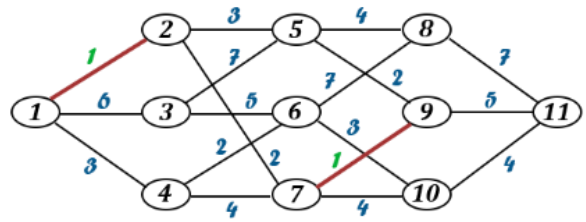
Використовуючи таблицю суміжності та алгоритм Флойда, запишемо найкоротший шлях між будь-якими двома точками.

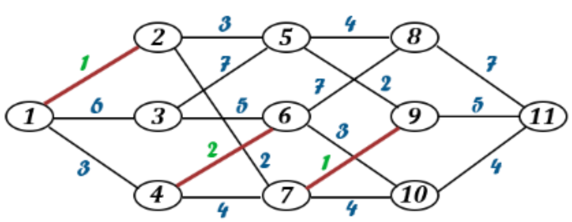
	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9
V_1	0	1	2	2	1	2	2	1	2
V_2	1	0	1	2	1	2	2	1	2
V_3	2	1	0	1	1	2	2	1	2
V_4	2	2	1	0	1	1	1	1	2
V_5	1	1	1	1	0	1	2	2	1
V_6	2	2	2	1	1	0	1	1	2
V_7	2	2	2	1	2	1	0	1	3
V_8	1	1	1	1	2	1	1	0	3
V_9	2	2	2	2	1	2	3	3	0

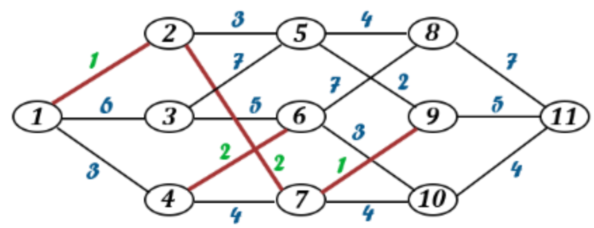
Як бачимо, найбільше число в таблиці це - 3, отже діаметр даного графа дорівнює 3.

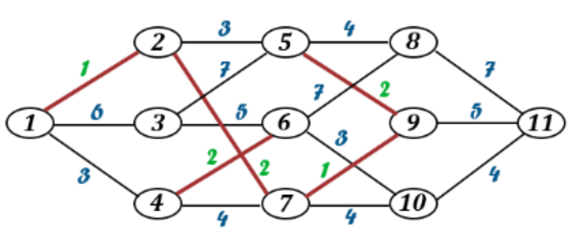
3. Знайдемо мінімальне остове дерево за допомогою алгоритму Красакла, поступово шукаючи ребра з мінімальною вагою, з урахуванням ациклічності:

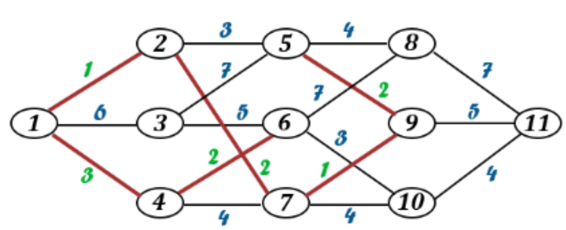


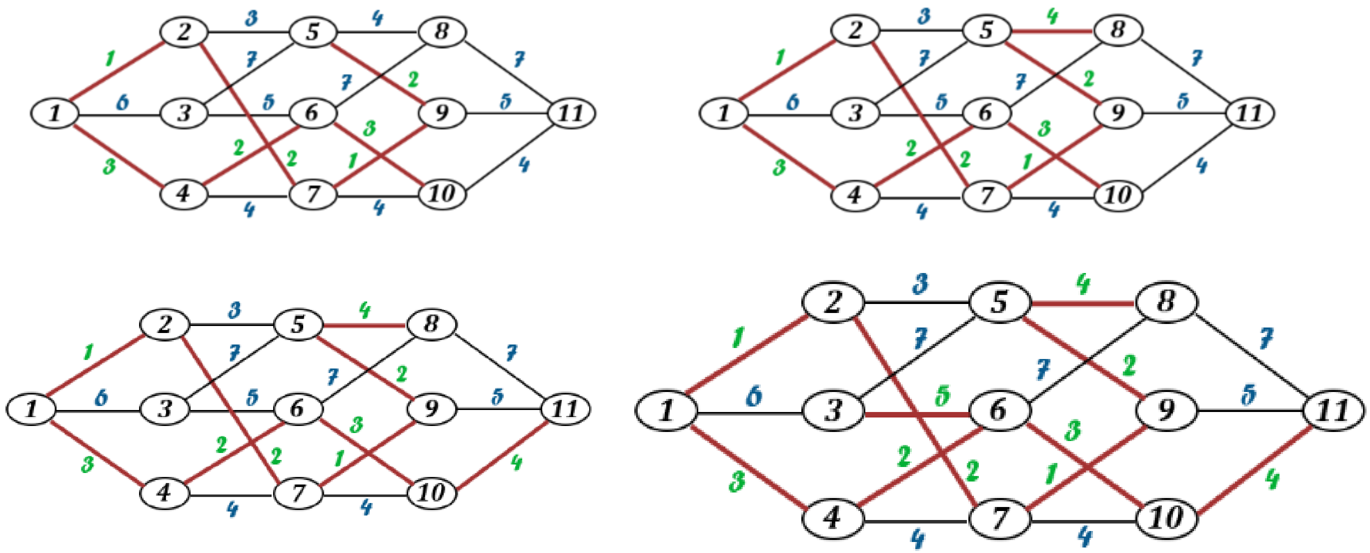






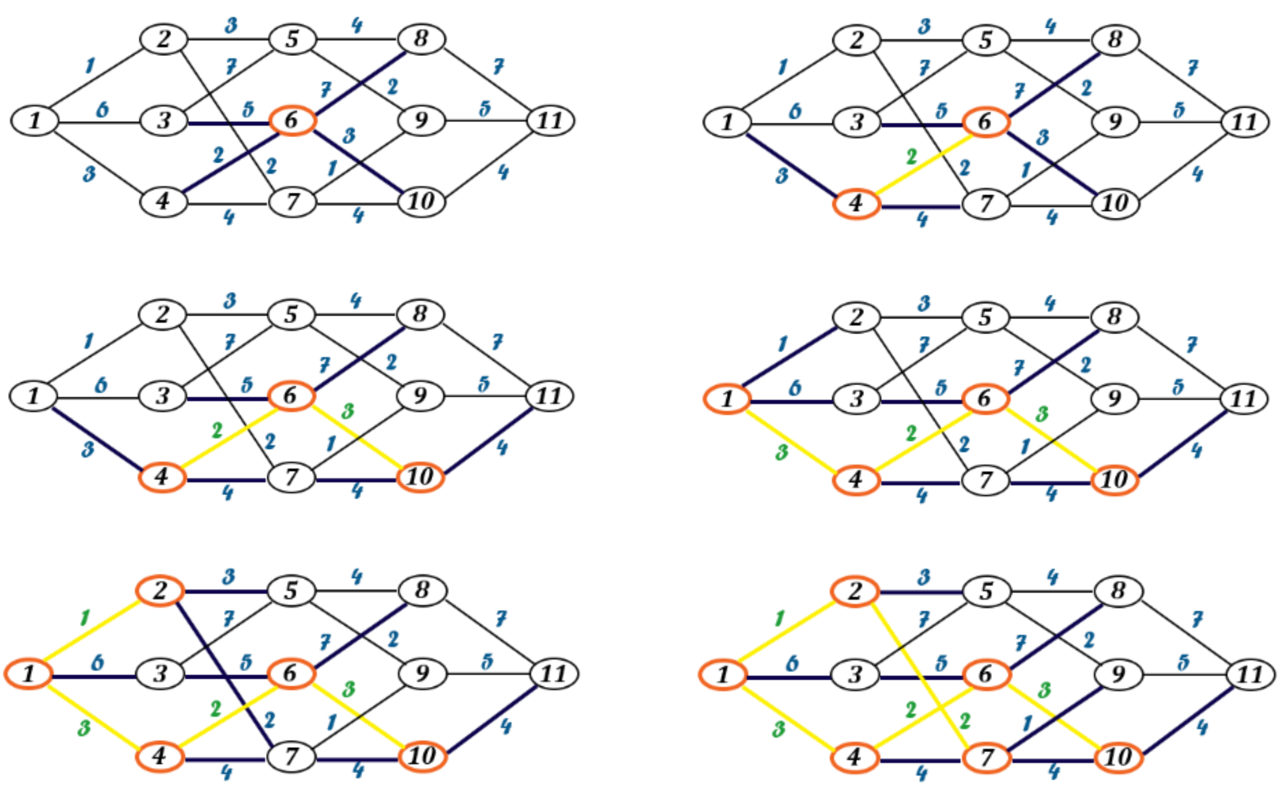


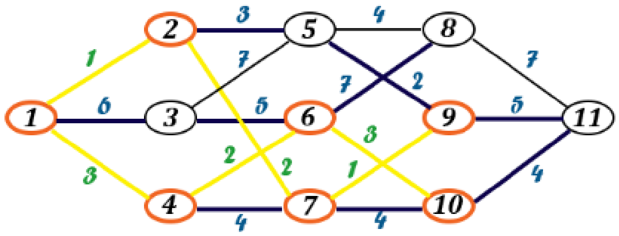


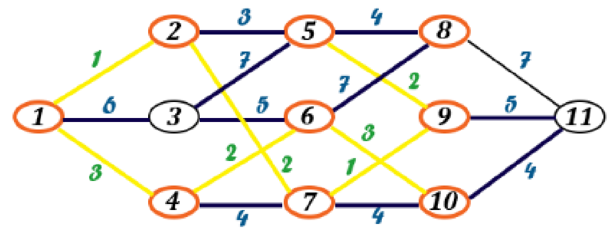


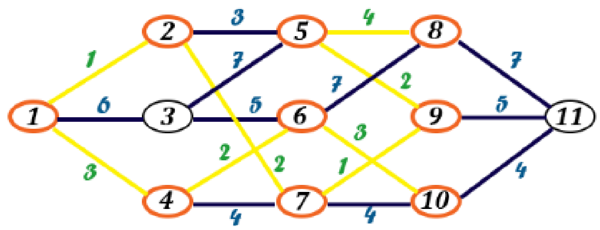
Тепер знайдемо мінімальне кістякове дерево використовуючи алгоритм Прима. Тобто будемо шукати мінімальне за вагою ребро інцидентне тільки до пройдених вершин. Для цього виберемо довільну початкову вершину (наприклад вершину 6) і поступово побудуємо остове дерево за цим алгоритмом.

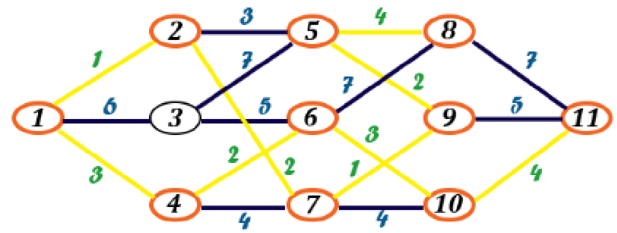
Нехай жовтим ребром ми позначимо ті ребра, які були вже пройдені на даному кроці, а темно-синім, ті, до яких ми можемо тепер потрапити.

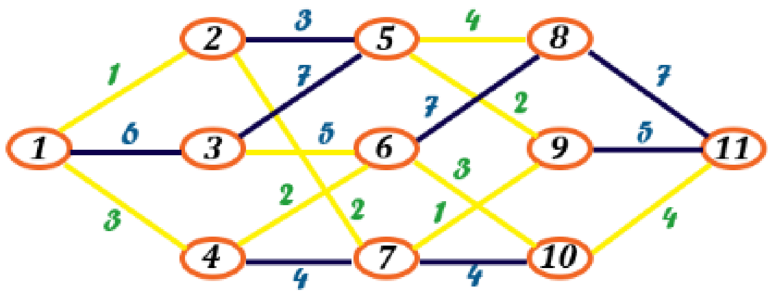










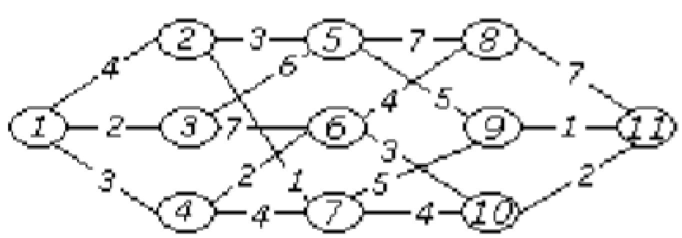


Додаток 2

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

Варіант № 12

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



Представимо даний граф у текстовому файлі “*Graph*”, вказавши кількість вершин, номери з’єднаних ребер та їх ваги відповідно.

Graph: Блокнот

Файл Редагування Формат Вигляд Довідка

```
11
(1,2) 4
(1,3) 2
(4,1) 3
(2,5) 3
(2,7) 1
(3,5) 6
(3,6) 7
(4,7) 4
(4,6) 2
(5,8) 7
(5,9) 5
(6,8) 4
```

Програмна реалізація:

```
#include <bits/stdc++.h>
using namespace std;
ifstream fin("Graph.txt");
int V;
int parent[100];
int find(int i)
{
	while (parent[i] != i)
		i = parent[i];
	return i;
}

void veretex_union(int i, int j)
{
	int a = find(i);
	int b = find(j);
	parent[a] = b;
}

void kruskalMST(int** Matrix_cost, int V)
{
	int mincost = 0;

	for (int i = 0; i < V; i++)
		parent[i] = i;

	int edge_counter = 0;
	while (edge_counter < V - 1) {
		int mine = INT_MAX, a, b;
		for (int i = 0; i < V; i++) {
			for (int j = 0; j < V; j++) {
				if (find(i) != find(j) && Matrix_cost[i][j] < mine) {
					mine = Matrix_cost[i][j];
```

```
                        a = i;
                        b = j;
                    }
                }
            }

            veretex_union(a, b);
            printf("Edge %d:(%d, %d) cost:%d \n", edge_counter++, a, b, mine);
            mincost += mine;
        }
        printf("\n Minimum cost= %d \n", mincost);
}

int main()
{
        char c;
        int Vertexs_number, vertex1, vertex2, cost;
        if (!fin.is_open()) {
                cout << "File is not found!";
                exit(EXIT_SUCCESS);
        }
        fin >> Vertexs_number;
        V = Vertexs_number;
        int** Matrix_cost = new int* [V];
        for (int i = 0; i < V; i++)
                *(Matrix_cost + i) = new int[V];

        for (int i = 0; i < Vertexs_number; i++) {
                for (int j = 0; j < Vertexs_number; j++) {
                        Matrix_cost[i][j] = INT_MAX;
                }
        }

        while (!fin.eof()) {
                fin >> c >> vertex1 >> c >> vertex2 >> c >> cost;
                if (vertex1 > vertex2) swap(vertex1, vertex2);
                for (int i = 0; i < Vertexs_number; i++) {
                        for (int j = i + 1; j < Vertexs_number; j++) {
                                if (i == vertex1 && j == vertex2) {
                                        Matrix_cost[i][j] = cost;
                                        Matrix_cost[j][i] = cost;
                                        break;
                                }
                        }
                }
        }

        kruskalMST(Matrix_cost, V);

        for (int i = 0; i < V; i++) {
                delete Matrix_cost[i];
        }
        delete Matrix_cost;
        return 0;
}
```

Результат виконання програми

```
"C:\Users\Admin\Desktop\New folder\–шёъЁхЄър\diskretlab4\bin\Debug\diskretlab4.exe"
Edge 1:(2, 7) cost:1
Edge 2:(9, 11) cost:1
Edge 3:(1, 3) cost:2
Edge 4:(4, 6) cost:2
Edge 5:(10, 11) cost:2
Edge 6:(1, 4) cost:3
Edge 7:(2, 5) cost:3
Edge 8:(6, 10) cost:3
Edge 9:(1, 2) cost:4
Edge 10:(6, 8) cost:4

 Minimum cost= 25

Process returned 0 (0x0)   execution time : 0.048 s
Press any key to continue.
```

Результатом виконання програми є мінімальне остове дерево виведене на екран у такий спосіб : Номер ребра:(вершина_1, вершина_2) вага даного ребра.

Після виведення усіх ребер, виводиться також мінімальна вартість кістякового дерева.

Висновки

Ми набули практичних вмінь та навичок з виконання різноманітних дій над графами та використання двох алгоритмів пошуку кістякового дерева: Краскала та Прима. Якщо порівняти остові дерева знайдені за цими алгоритмами, то виявиться, що вони ідентичні. Отже дані алгоритми дають однаковий результат знайдений різними методами.