

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”  
ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра систем штучного інтелекту

**Лабораторна робота №5**  
із дисципліни  
«Дискретна математика»

**Виконав:**  
студент групи КН-113  
Калапунь Н.Т.

**Викладач:**  
Мельникова Н.І.

Львів – 2019 р.

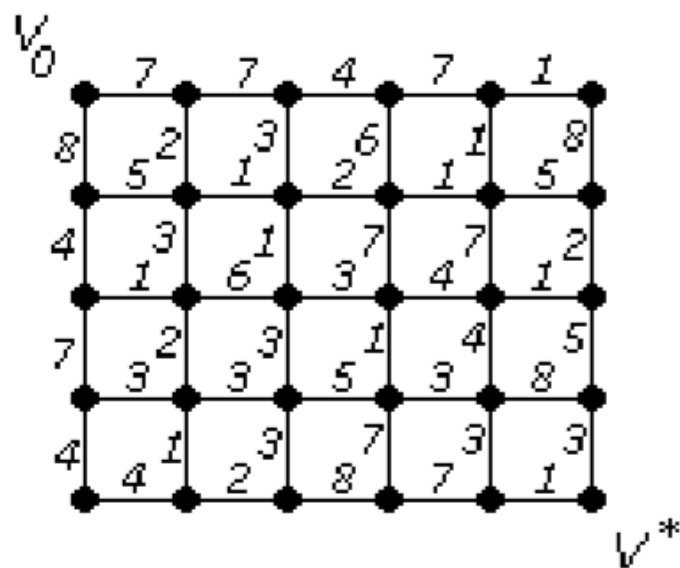
**Тема роботи:** Знаходження найкоротшого маршруту за алгоритмом Дейкстри.  
Плоскі планарні графи.

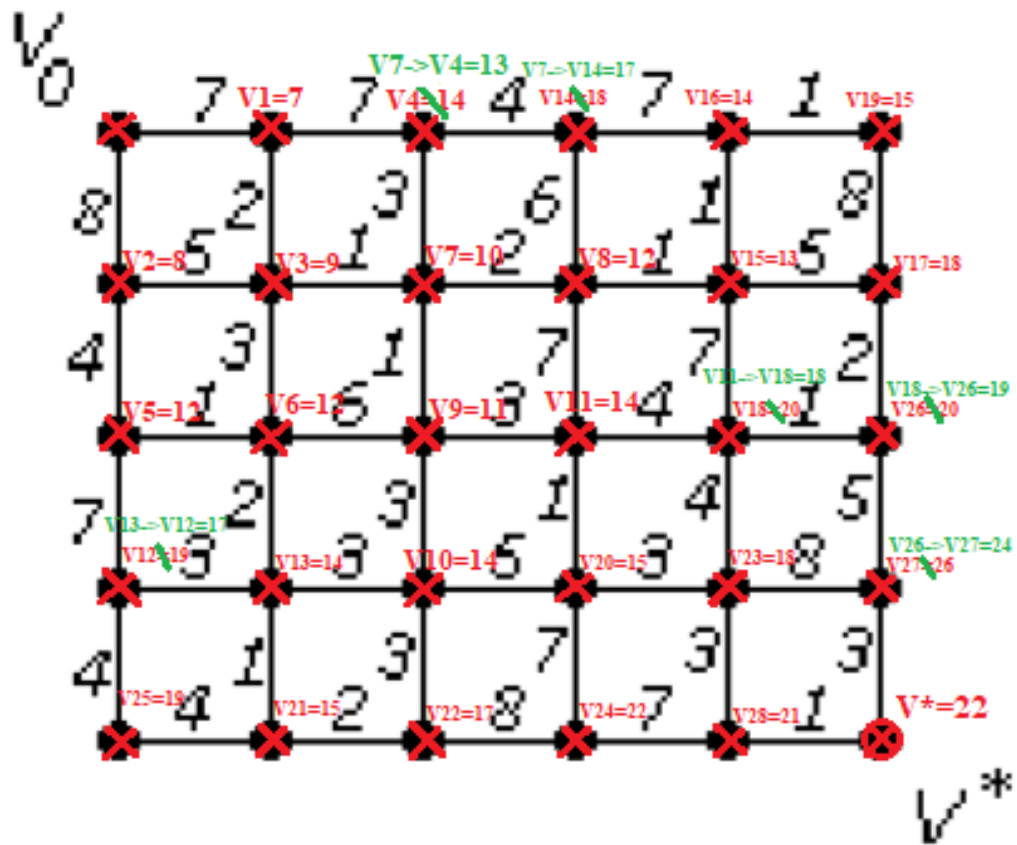
**Мета роботи:** Набуття практичних вмінь та навичок з використання алгоритму Дейкстри.

### Варіант – 10

**Розв'язати на графах наступні 2 задачі:**

1. За допомогою алгоритму Дейкстри знайти найкоротший шлях у графі поміж парою вершин  $V_0$  і  $V^*$ .

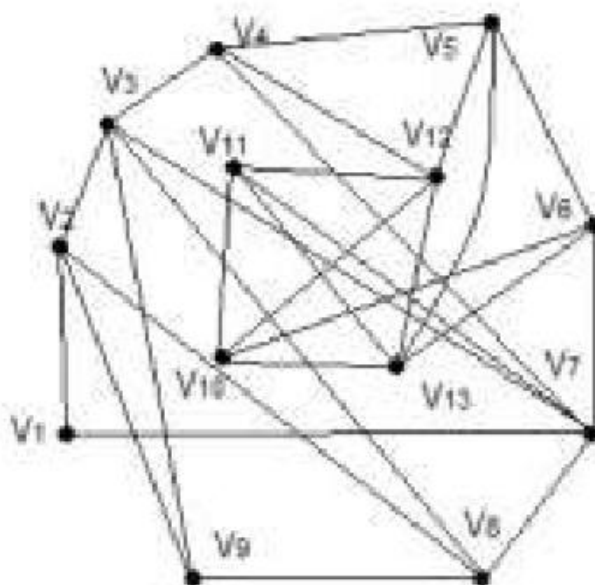




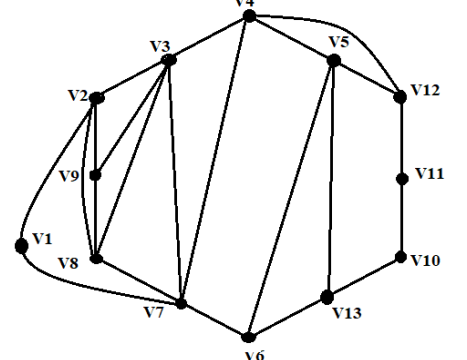
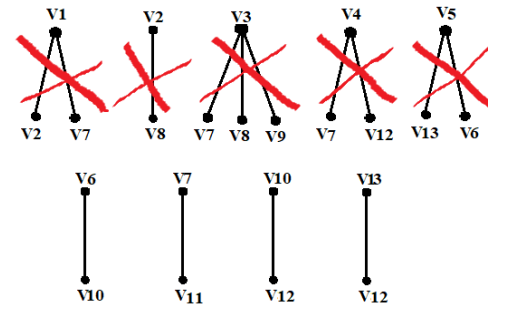
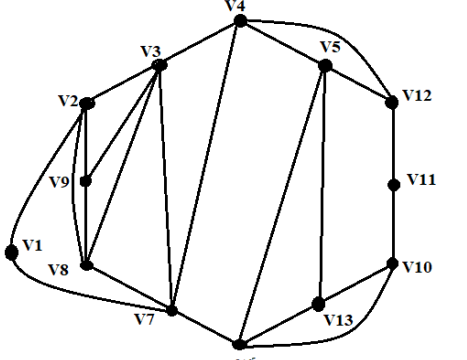
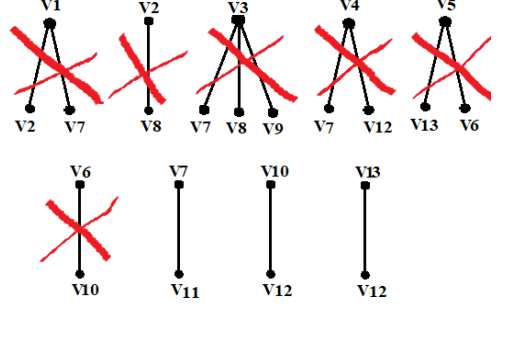
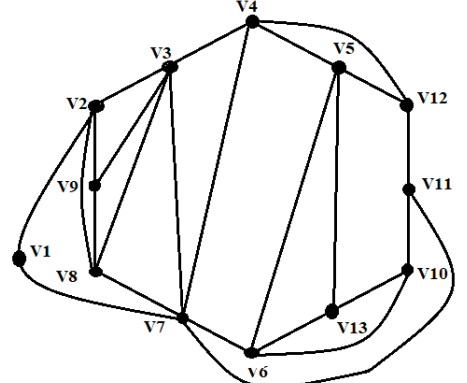
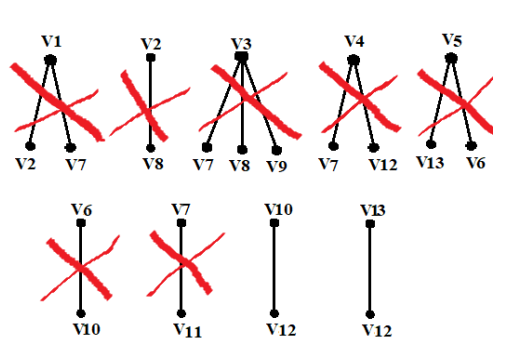
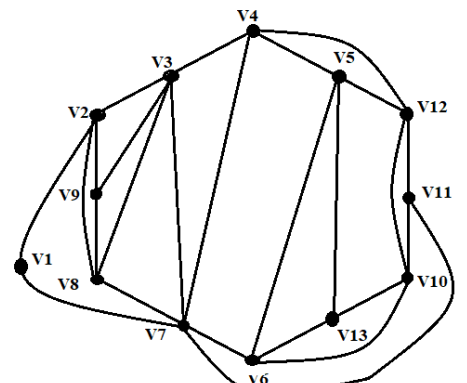
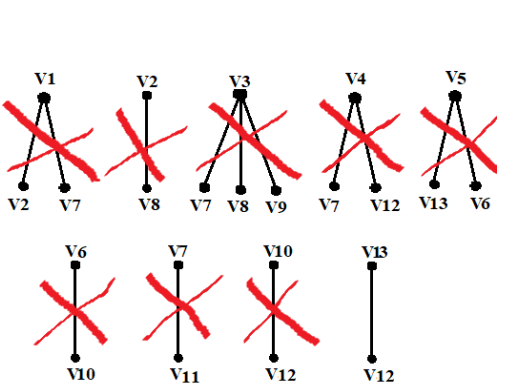
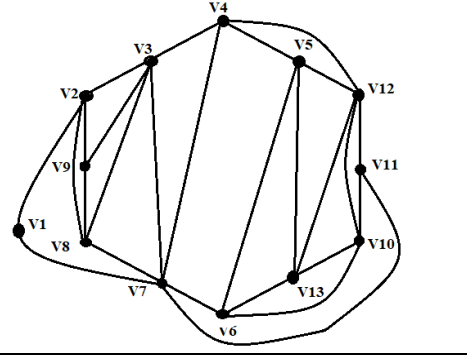
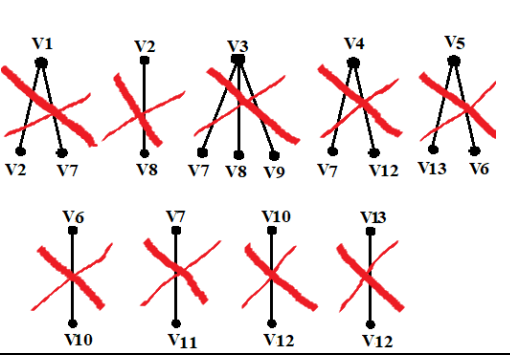
Відстань з  $V_0$  до  $V^* = 22$ ;

Шлях:  $V_0 \rightarrow V_1 \rightarrow V_3 \rightarrow V_7 \rightarrow V_9 \rightarrow V_{11} \rightarrow V_{20} \rightarrow V_{23} \rightarrow V_{28} \rightarrow V^*$

2. За допомогою  $\Upsilon$ -алгоритма зробити укладку графа у площині, або довести що вона неможлива.

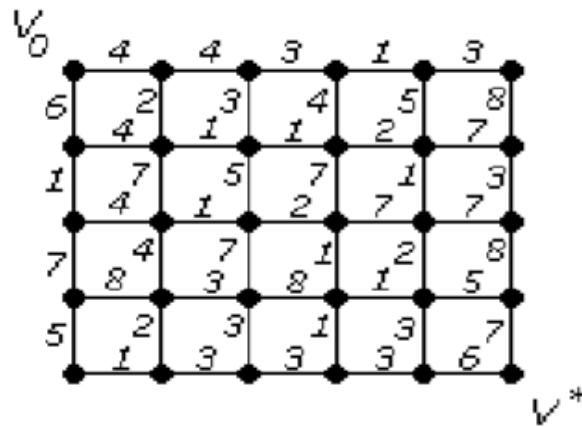


	Граф	Сегменти
Укладаємо сегмент V2-V1-V7		
Укладаємо сегмент V2-V8		
Сегмент V7-V3-V8-V9 має тільки одну грань для укладання, тому обираємо його.		
Укладаємо сегмент V7-V4-V12		

<p>Укладаємо сегмент V13-V5-V6</p>		
<p>Укладаємо сегмент V6-V10</p>		
<p>Укладаємо сегмент V7-V11</p>		
<p>Укладаємо сегмент V10-V12</p>		
<p>Укладаємо сегмент V12-V13</p>		

## Додаток 2

Написати програму, яка реалізує алгоритм Дейкстри знаходження найкоротшого шляху між парою вершин у графі. Протестувати розроблену програму на графі згідно свого варіанту.



## Програмна реалізація:

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <fstream>
#include <locale>
using namespace std;
int main()
{
    ifstream fin("MyFile.txt");
    setlocale(LC_ALL, "Ukrainian");
    int versh, rebra;
    fin >> versh >> rebra;
    const int SIZE = 30;
    int matrix[SIZE][SIZE]; // матриця зв'язків
    int distance[SIZE]; // мінімальна відстань
    int visited[SIZE]; // чи відвідані вершини
    int dis, top, min;
    int begin_index = 0;

    for (int i = 0; i < versh; i++) // Ініціалізація матриці зв'язків
    {
        distance[i] = 99999;
        visited[i] = 0;
        for (int j = 0; j < versh; j++)
        {
            matrix[i][j] = 0;
            matrix[j][i] = 0;
        }
    }
    for (int i = 0; i < rebra; i++) {
```

```

int v1, v2, dis;
fin >> v1 >> v2 >> dis;
matrix[v1 - 1][v2 - 1] = dis;
matrix[v2 - 1][v1 - 1] = dis;
}

/*for (int i = 0; i < versh; i++)    // Вивід матриці
{
    for (int j = 0; j < versh; j++)
    {
        cout << " " << matrix[i][j];
    }
    cout << endl;
} */
cout <<
"\n\n+++++\n\n";

distance[0] = 0;
do {
    top = 99999;
    min = 99999;
    for (int i = 0; i < versh; i++)
    {
        if (visited[i] == 0 && distance[i] < min)
        {
            min = distance[i];
            top = i;
        }
    }
    if (top != 99999)
    {
        for (int i = 0; i < versh; i++)
        {
            if (matrix[top][i] > 0)
            {
                dis = min + matrix[top][i];
                if (dis < distance[i])
                {
                    distance[i] = dis;
                }
            }
        }
        visited[top] = 1;
    }
} while (top < 99999);

int end = versh - 1;
int waight = distance[end];
int way[30];
way[0] = versh - 1;
int k = 1;
while (end != 0)
{
    for (int i = 0; i < versh; i++)
    {
        if (matrix[end][i] > 0)
        {
            if (distance[i] == waight - matrix[end][i])
            {
                waight = distance[i];
                way[k] = i;
                end = i;
                k++;
            }
        }
    }
}
}

```

```

}
cout << "\nНайкоротший шлях з V0 до V29: ";
for (int i = k; i > 0; i--)
{
    if (i - 1 > 0)
        cout << way[i - 1] << " -> ";
    else
        cout << way[i - 1];
}
cout << "\n\nДовжина шляху: " << distance[versh - 1] << endl;
cout <<
"\n\n+++++\n\n";
;
}

```

## Результат роботи програми:

```

+++++
Найкоротший шлях з V0 до V29: 0 -> 1 -> 7 -> 8 -> 9 -> 10 -> 16 -> 22 -> 28 -> 29
Довжина шляху: 22
+++++

```

## Висновок:

На цій лабораторній роботі я навчився Знаходити найкоротший маршрут за алгоритмом Дейкстри. Засвоїв теорію про плоскі планарні графи. Набув практичних вмінь та навичок з використання алгоритму Дейкстри.