МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота №5

з дисципліни

«Дискретна математика»

Виконав:

студент групи КН-109

Яворський Володимир

Викладач:

Мельникова Н.І.

Львів – 2018 р.

Лабораторна робота № 5

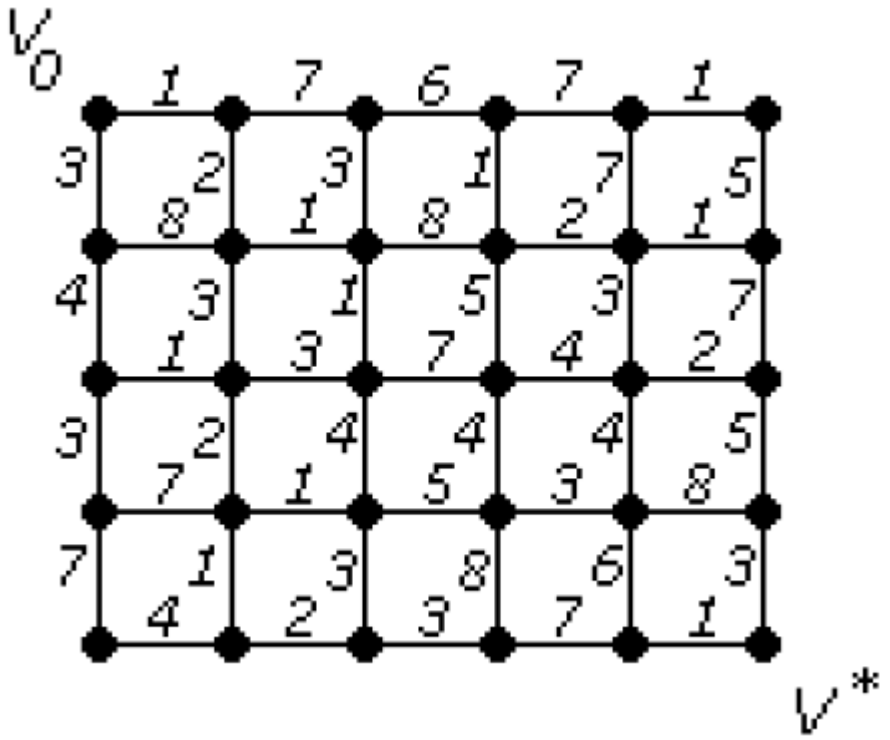
Тема: Знаходження найкоротшого маршруту за алгоритмом Дейкстри. Плоскі планарні графи

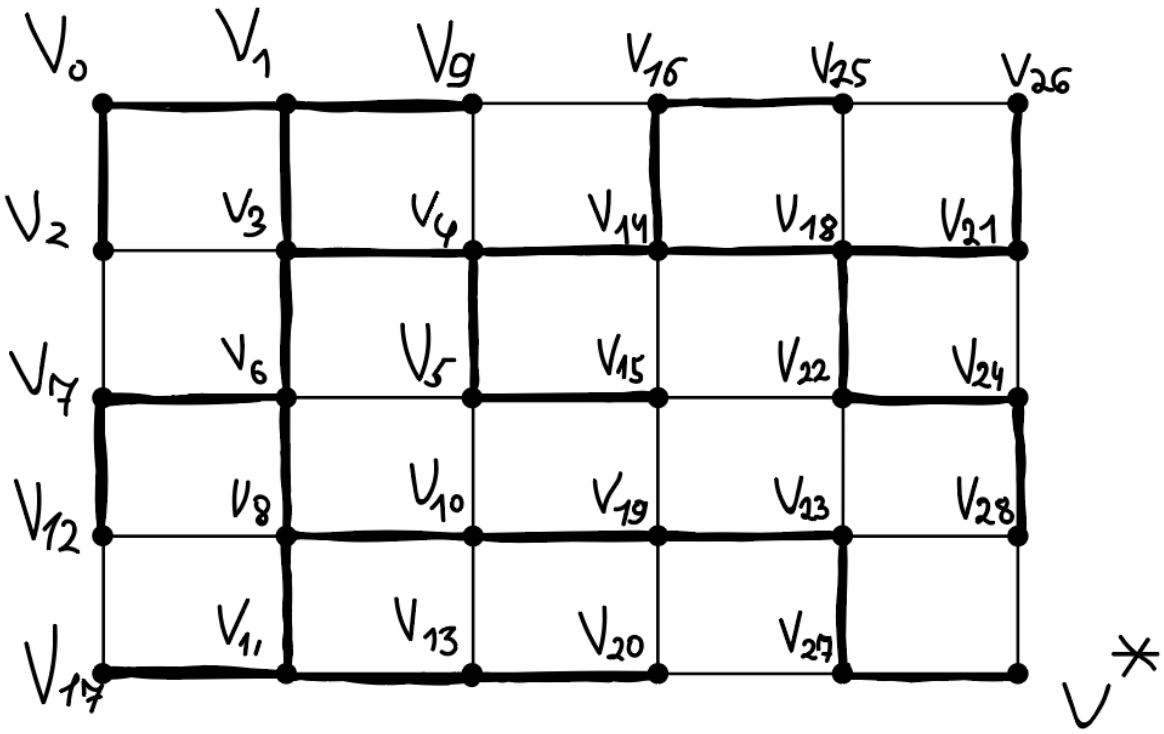
Мета роботи: набуття практичних вмінь та навичок з використання алгоритму Дейкстри.

Варіант №14

Завдання 1

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



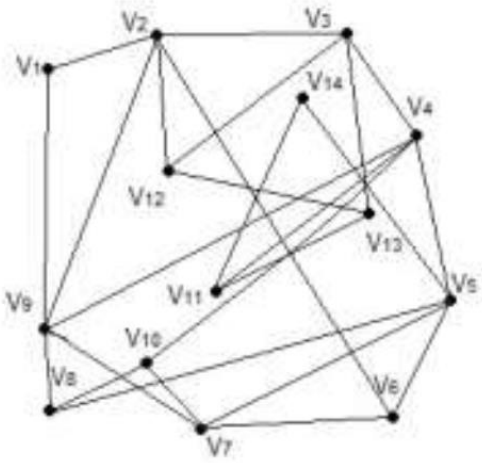


Запишемо найближчі вершини V_1, V_2, V_3, … у порядку їх появи

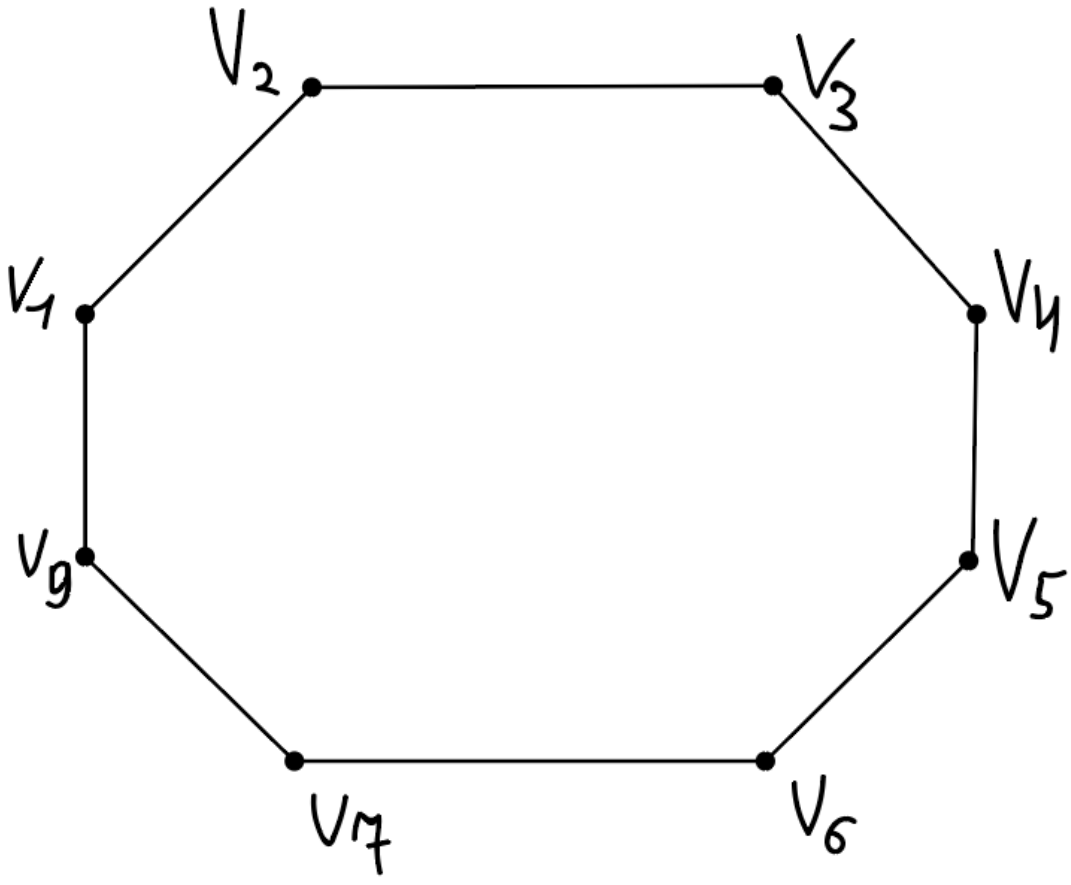
$l(v_1)$ = 1	$l(v_2)$ = 3	$l(v_3)$ = 3
$l(v_4)$ = 4	$l(v_5)$ = 5	$l(v_6)$ = 6
$l(v_7)$ = 7	$l(v_8)$ = 8	$l(v_9)$ = 8
$l(v_{10})$ = 9	$l(v_{11})$ = 9	$l(v_{12})$ = 10
$l(v_{13})$ = 11	$l(v_{14})$ = 12	$l(v_{15})$ = 12
$l(v_{16})$ = 13	$l(v_{17})$ = 13	$l(v_{18})$ = 14
$l(v_{19})$ = 14	$l(v_{20})$ = 14	$l(v_{21})$ = 15
$l(v_{22})$ = 17	$l(v_{23})$ = 17	$l(v_{24})$ = 19
$l(v_{25})$ = 19	$l(v_{26})$ = 20	$l(v_{27})$ = 23
$l(v_{28})$ = 24		$\boldsymbol{l(v^*)}$ = **24**

Наш шуканий шлях : [v_0, v_1, v_3, v_6, v_8, v_{10}, v_{19}, v_{23}, v_{27}, v^*].

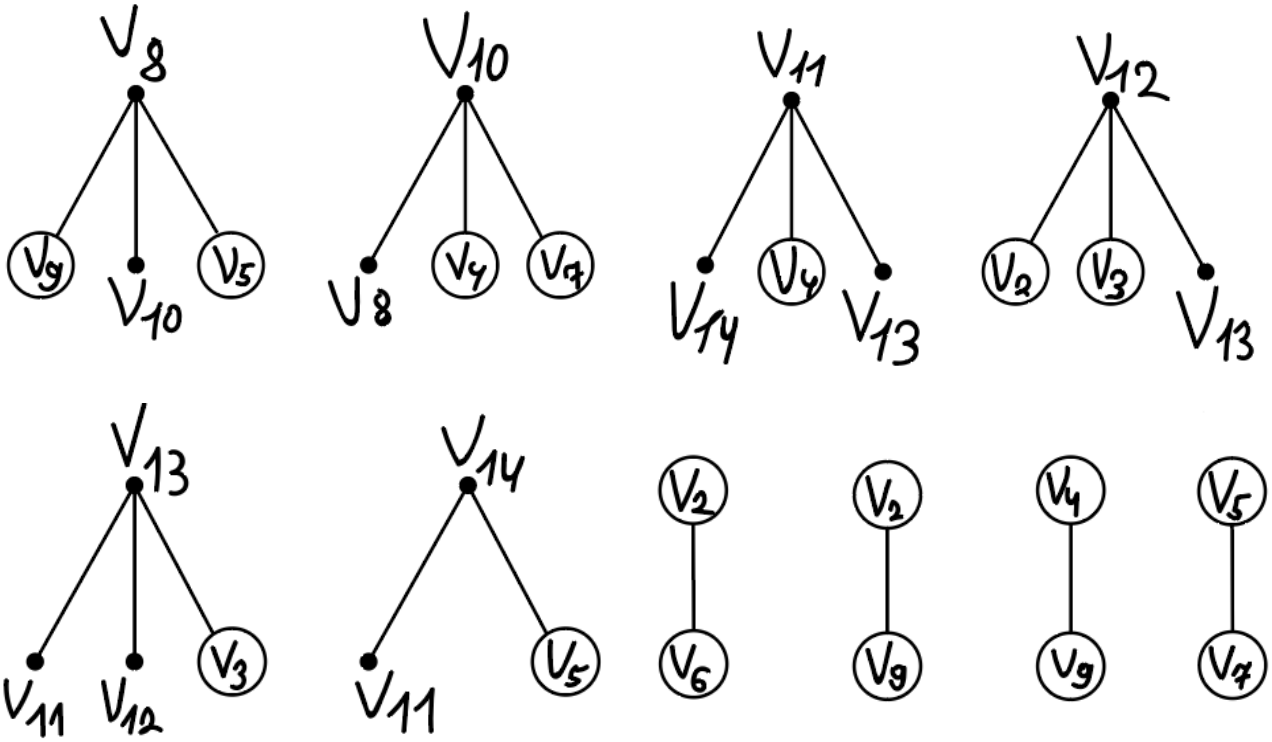
2. За допомогою γ -алгоритма зробити укладку графа у площині, або довести що вона неможлива.



Довільно вибираємо цикл [$v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_9$]



Виписуємо всі сегменти відносно циклу (контактні вершини обведені)



По черзі вкладаємо ланцюжки у наш цикл:

v_9 - v_8 - v_5;

v_2 - v_{12} - v_3;

v_{12} - v_{13} - v_3;

v_4 - v_{11} - v_{13};

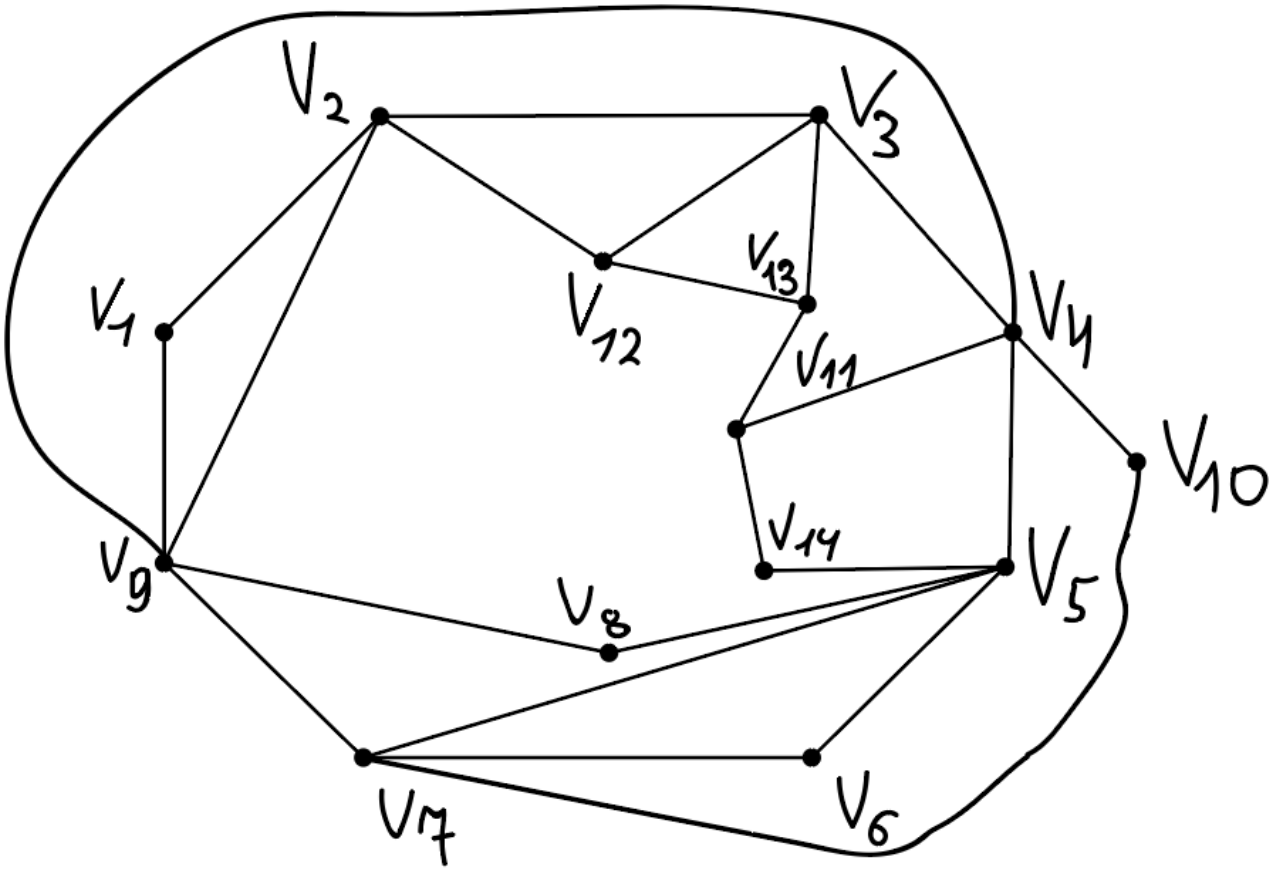
v_{11} - v_{14} - v_5;

v_4 - v_{10} - v_7;

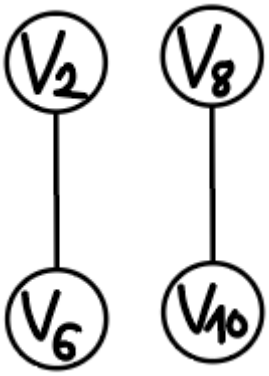
v_2 - v_9;

v_4 - v_9;

v_5 - v_7;



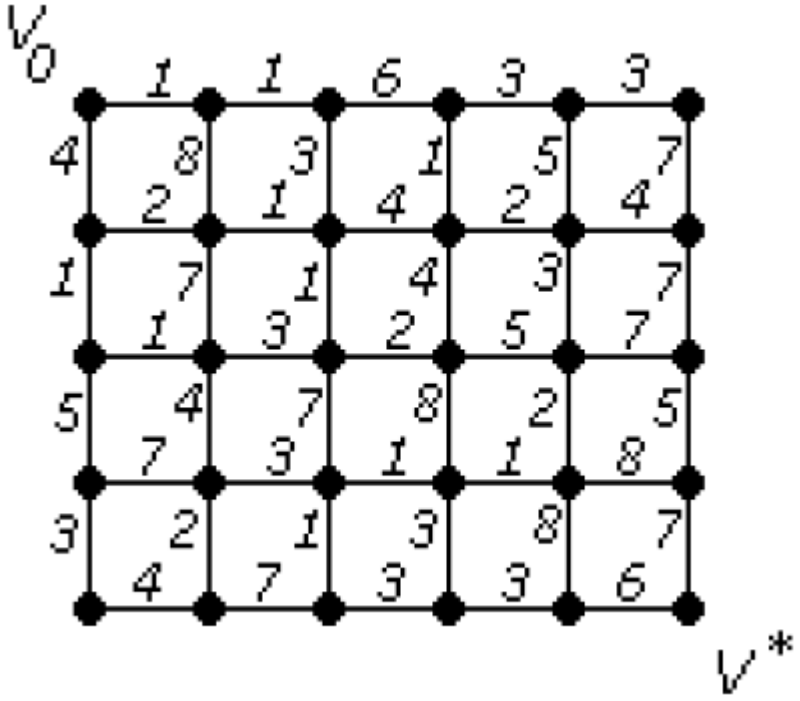
У нас залишились сегменти



Додати їх до нашого графу ми не можемо, адже вони належать різним секторам, і намалювати ребра без перетину не вийде. Отже, наш граф ***не є планарним***.

Завдання 2

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



Код програми:

```
#include <stdio.h>

//size of our matrix
#define N 30

int main()
{
   int DISTANCE[N];
   int VISITED[N];

   int GRAPH[N][N] = {
   //   1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
  /*1*/ { 0, 1, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  /*2*/ { 0, 0, 1, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  /*3*/ { 0, 0, 0, 6, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  /*4*/ { 0, 0, 0, 0, 3, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  /*5*/ { 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  /*6*/ { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  /*7*/ { 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  /*8*/ { 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
```

```
/*9*/ { 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
/*10*/{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
/*11*/{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
/*12*/{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
/*13*/{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
/*14*/{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0 },
/*15*/{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0 },
/*16*/{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0 },
/*17*/{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0 },
/*18*/{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0 },
/*19*/{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 3, 0, 0, 0 },
/*20*/{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 2, 0, 0 },
/*21*/{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0 },
/*22*/{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 3 },
/*23*/{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 8 },
/*24*/{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7 },
/*25*/{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0 },
/*26*/{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7 },
/*27*/{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0 },
/*28*/{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3 },
/*29*/{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6 },
/*30*/{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }
 };

 for (int i = 0; i < N; i++)
 {
    DISTANCE[i] = 999;
    VISITED[i] = 0;
 }
 DISTANCE[0] = 0;

 int min, k, a;

 do
 {
    k = 999;
    min = 999;

    for (int i = 0; i < N; i++)
    {
       if ((VISITED[i] == 0) && (DISTANCE[i] < min))
       {
          min = DISTANCE[i];
          k = i;
       }
    }

    if (k != 999)
    {
       for (int i = 0; i < N; i++)
```

```
        {
            if (GRAPH[k][i] > 0)
            {
                a = min + GRAPH[k][i];
                if (a < DISTANCE[i])
                {
                    DISTANCE[i] = a;
                }
            }
        }
        VISITED[k] = 1;
    }
  } while (k < 999);

  printf("\nThe shortest ways:\n\n");
  for (int i = 0; i < N; i++)
  {
     printf("l(V0-V%d) = %d\n", i, DISTANCE[i]);
  }
  printf("\n");

  return 0;
}
```

Результат виконання програми: