МІНІСТЕРСТВО ОСВІТИ І НАУКИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра систем штучного інтелекту

Лабораторна робота №5

З курсу “Дискретна математика”

Виконав:
ст .гр. КН-110
Марій Павло

Львів – 2018

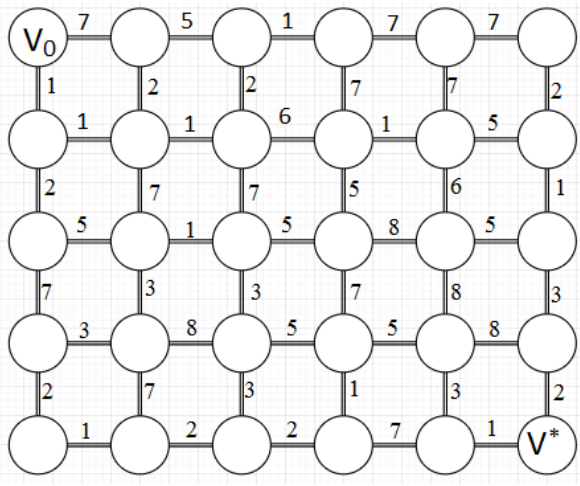
Тема: Знаходження найкоротшого маршруту за алгоритмом Дейкстри. Плоскі планарні графи.

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

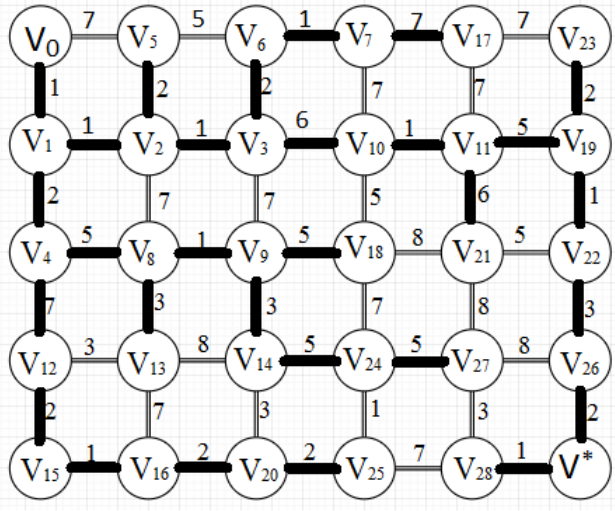
Варіант 2

Завдання 1. Розв'язати на графах наступні 2 задачі:

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



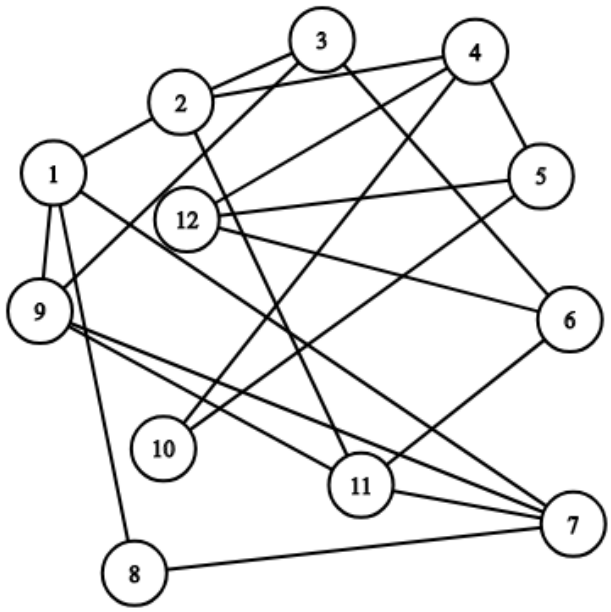
За алгоритмом, починаючи з вершини V_0, вибираємо такі найближчі вершини, щоб довжина ланцюг до них була мінімальною. Результатом буде граф G:



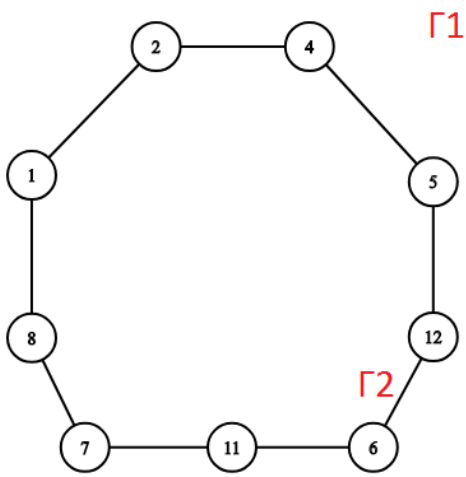
Маршрут до V^*: $V_0 \to V_1 \to V_2 \to V_3 \to V_{10} \to V_{11} \to V_{19} \to V_{22} \to V_{26} \to V^*$

$l(V_0) = 0;$	$l(V_{11}) = 10;$	$l(V_{22}) = 16;$
$l(V_1) = 1;$	$l(V_{12}) = 10;$	$l(V_{23}) = 17;$
$l(V_2) = 2;$	$l(V_{13}) = 11;$	$l(V_{24}) = 17;$
$l(V_3) = 3;$	$l(V_{14}) = 12;$	$l(V_{25}) = 17;$
$l(V_4) = 3;$	$l(V_{15}) = 12;$	$l(V_{26}) = 19;$
$l(V_5) = 4;$	$l(V_{16}) = 13;$	$l(V^*) = 21;$
$l(V_6) = 5;$	$l(V_{17}) = 13;$	$l(V_{27}) = 22;$
$l(V_7) = 6;$	$l(V_{18}) = 14;$	$l(V_{28}) = 22;$
$l(V_8) = 8;$	$l(V_{19}) = 15;$	
$l(V_9) = 9;$	$l(V_{20}) = 15;$	
$l(V_{10}) = 9;$	$l(V_{21}) = 16;$	

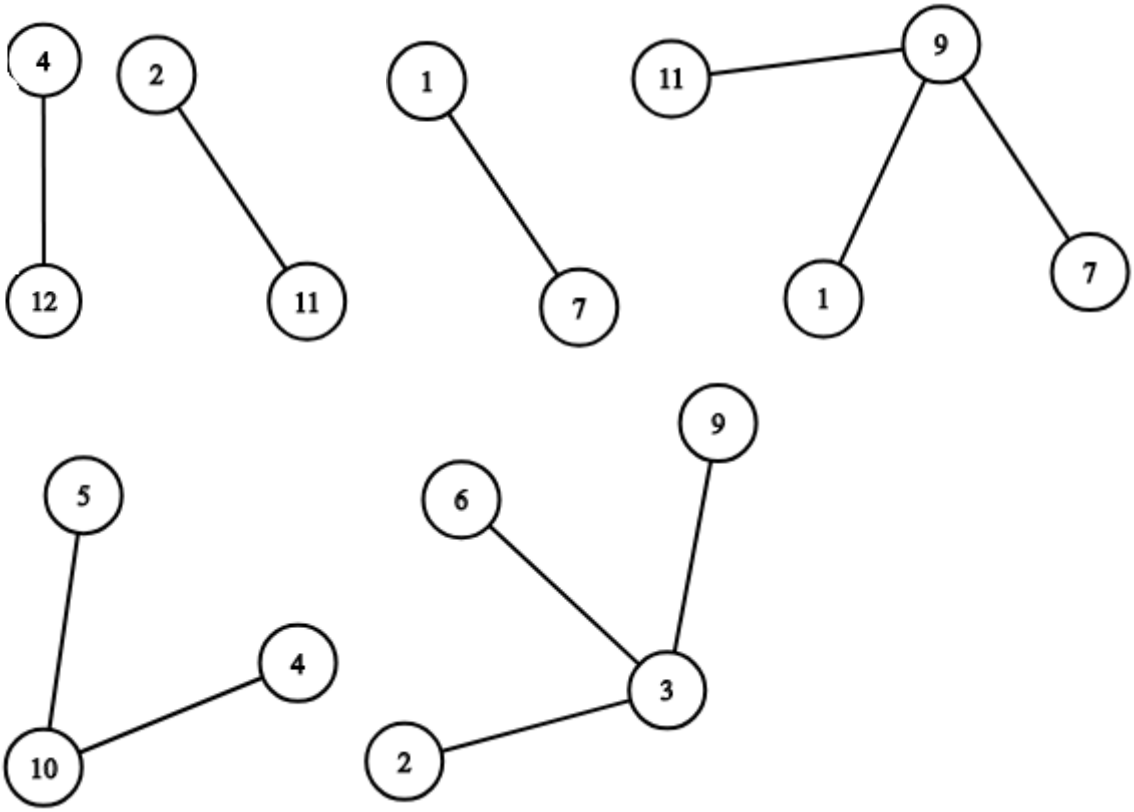
2. За допомогою у-алгоритма зробити укладку графа у площині, або довести, що вона неможлива.



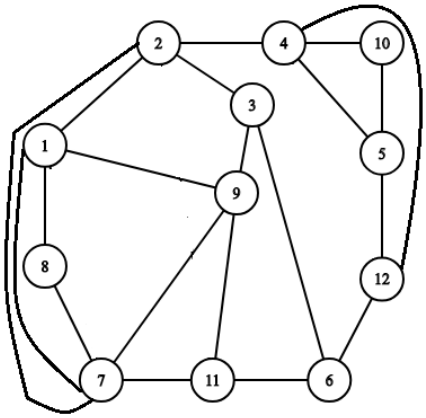
Перший крок: ініціалізація алгоритму: вибираємо будь-який простий цикл в G, нехай це буде {1,2,4,5,12,6,11,7,8,1} і укладаємо його на площині. Утворюється дві грані: Γ_1 – зовнішня та Γ_2 – внутрішня.



Виділяємо сегменти:



Далі вибираємо сектори і вписуємо їх в цикл:



Укладка проведена успішно.

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>
#include <stdlib.h>
#define SIZE 30                // Кількість вершин, розмір матриці

int Array[SIZE][SIZE];         // Масив матриці графа
int distance[SIZE];            // Масив мінімальних відстаней
int vertex[SIZE];              // Масив вершин

// Заповнення матриці нулями
void zeroArray()
{
   for(int i=0; i<SIZE; i++)
      for(int j=0; j<SIZE; j++)
         Array[i][j]=0;
}

// Запис дуг в матрицю
void enterEdges()
{
   printf("Enter edges:\n");
   int r, c, n;
   for(int i=0; i<49; i++)
   {
      scanf("%d %d %d", &r, &c, &n);
      Array[r-1][c-1]=n;
      Array[c-1][r-1]=n;
   }
}
```

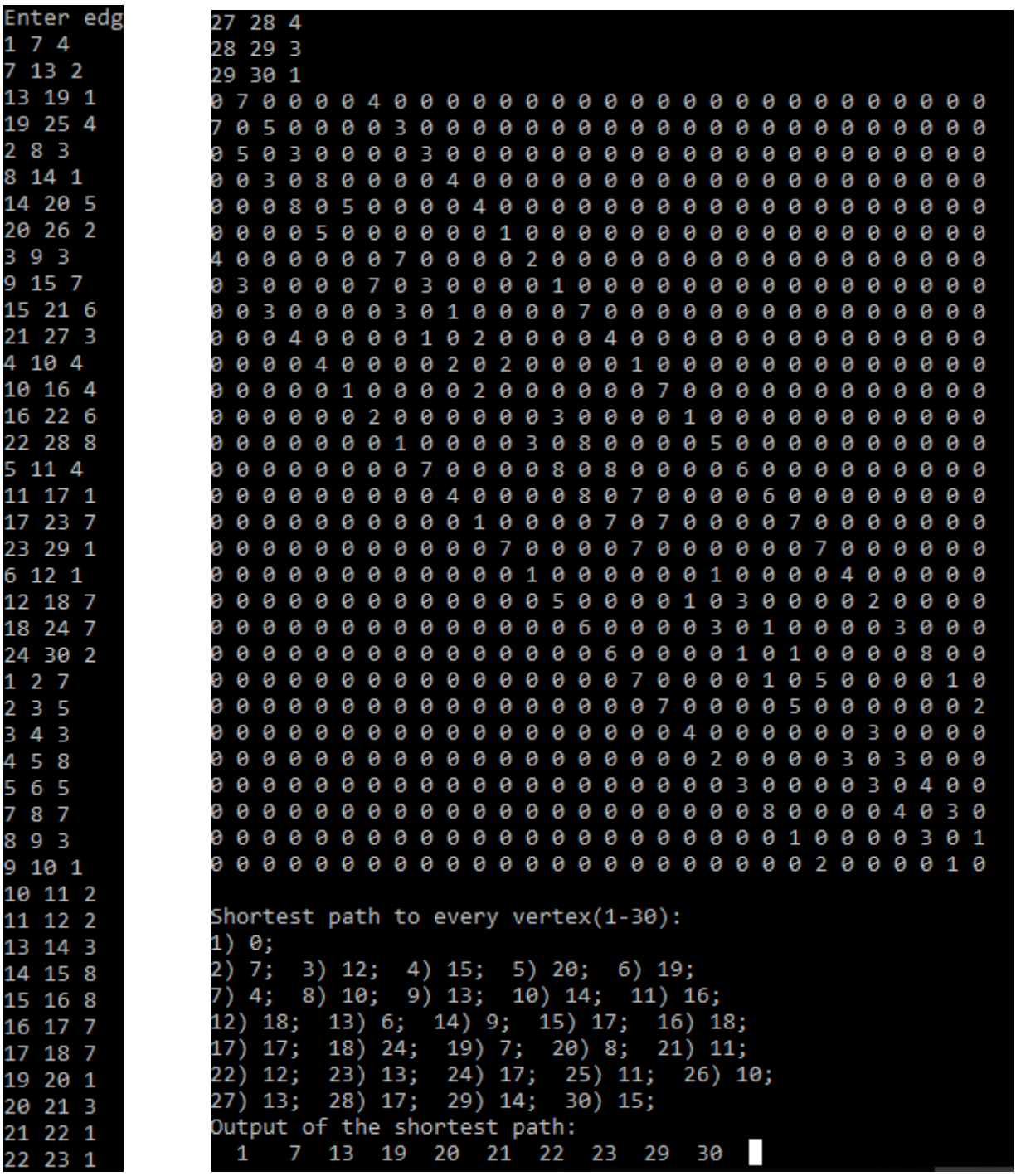
```
// Вивід матриці графа
void printArray()
{
   for(int i=0; i<SIZE; i++)
   {
      for(int j=0; j<SIZE; j++)
      {
         printf("%d ", Array[i][j]);
      }
      printf("\n");
   }
}

// Ініціалізація масивів відстаней та вершин
void initArray()
{
   for(int i=0; i<SIZE; i++)
   {
      distance[i]=10000;        // Всі відстані поки що є невизначеними, тому 10000
      vertex[i]=1;              // Всі вершини є необійденими, мають значення 1
   }
   distance[0]= 0;              // Відстань до першої вершини 0
}

// Вивід найкоротших відстаней до вершин 1-30
void printDistance()
{
   printf("\nShortest path to every vertex(1-30): \n");
      for(int i=0; i<SIZE; i++)
      {
         printf("%d) %d;  ", i+1, distance[i]);
         if((i+1)%5==0 && i!=0)
          printf("\n");
```

```
        }
}



// Головна функція
int main(void)
{
    int temp;                   // Тимчасова змінна для запису матриці графа
    int minindex;               // Змінна для ітерацій циклу основного алгоритму
    int min;

    zeroArray();                // Заповнення матриці нулями

    enterEdges();               // Ввід ребер та їхньої ваги

    printArray();               // Вивід матриці

    initArray();                // Ініціалізація масивів відстаней та вершин



    // Основний алгоритм
    do
    {
        minindex=10000;
        min=10000;
        for (int i=0; i<SIZE; i++)
        {
            if ((vertex[i]==1) && (distance[i]<min))        // Якщо вершину ще не обійшли і вага менша за
min
            {
                min=distance[i];                        // Встановлюємо змінній min мінімальне значення
                minindex=i;                                                             // Визначаємо
позицію мінімальної відстані
            }
```

```
    }

    if(min!=10000)                          // Якщо попередня умова виконалась
    {
        for(int i=0; i<SIZE; i++)
        {
            if(Array[minindex][i]>0)
            {
                temp=min+Array[minindex][i];
                if(temp<distance[i])
                {
                    distance[i]=temp;                   // Відстань записуємо в масив відстаней
                }
            }
        }
        vertex[minindex]=0;                         // Помічаємо вершину пройденою
    }
}
while(minindex < 10000);                            // Допоки не знайдемо всіх відстаней

printDistance();                                    // Вивід найкоротших відстаней

        // Відновлення шляху
int ver[SIZE];                                      // Масив відвіданих вершин
int end = 29;                                       // Індекс кінцевої вершини 30-1=29
ver[0] = end + 1;                                   // Перший елемент -- остання вершина
int k = 1;                                          // Індекс попередньої вершини
int weight = distance[end]; // вес конечной вершины

while (end > 0) // Поки не дійшли до початкової вершини
{
    for(int i=0; i<SIZE; i++) // Переглядаємо всі вершини
        if (Array[end][i] != 0)   // Якщо зв'язок є
        {
```

```
            int temp = weight - Array[end][i]; // Визначаємо вагу шляху з попередньої вершини
            if (temp == distance[i]) // Якщо вага співпала з вирахуваною
            {                        // Значить, з цієї вершини був здійснений перехід
               weight = temp;       // Зберігаємо нову вагу
               end = i;             // Зберігаємо попередню вершину
               ver[k] = i + 1;      // І записуємо її в масив
               k++;
            }
         }
   }

   // Вивід шляху (початкова вершина опинилась в кінці масиву з k елементів)
   printf("\nOutput of the shortest path:\n");
   for(int i = k-1; i>=0; i--)
      printf("%3d ", ver[i]);

   scanf("%d", &temp);
   return 0;
}
```

Результат:



Висновок: Я набув практичних вмінь та навичок з використання алгоритму Дейкстри.