**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”**

**Кафедра систем штучного інтелекту**

**Лабораторна робота**

з дисципліни

«Дискретна математика»

**Виконав:**

студент групи КН-110

Пиріг Богдан

**Викладач:**

Мельникова Н.І.

Львів – 2018 р.

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

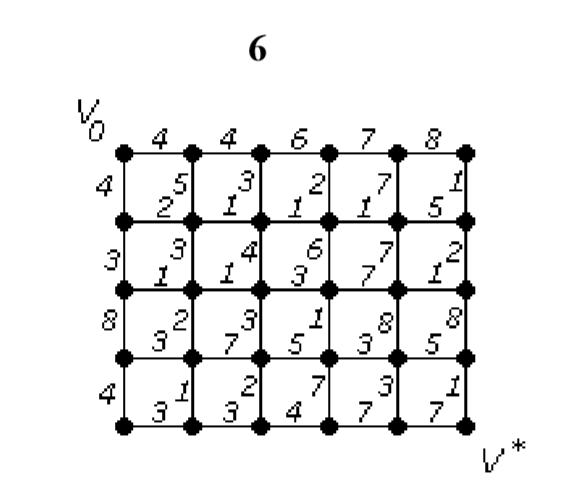
**Мета роботи:** набуття практичних вмінь та навичок з використання алгоритму Дейкстри.

**Варіант – 6.**

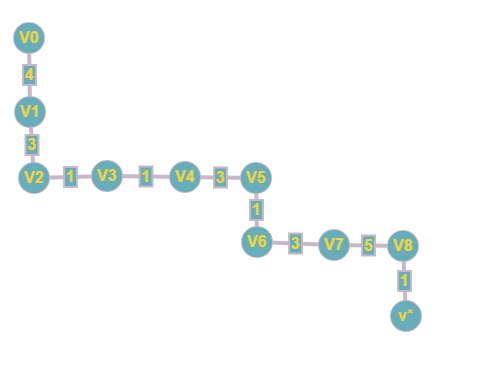
**Завдання з додатку 1:**

Завдання № 1.

Розв'язати на графах наступні 2 задачі: 1. За допомогою алгоритму Дейкстра знайти найкоротший шлях у графі поміж парою вершин V0 і V\* .

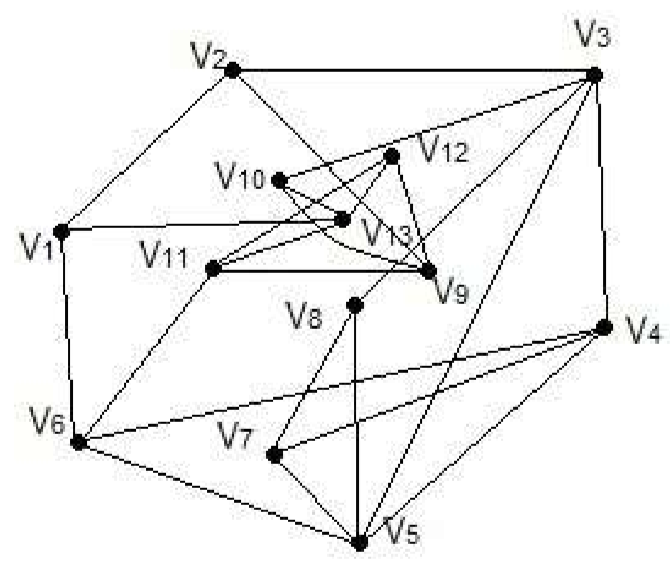


Розв’язання:

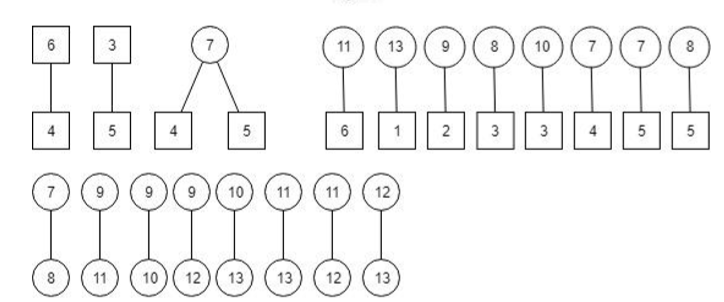


l(V1) = 4; l(V2) = 7; l(V3) = 8; l(V4) = 9; l(V5) = 12; l(V6) = 13; l(V7) = 16; l(V8) = 21; l (V\*) = 22;

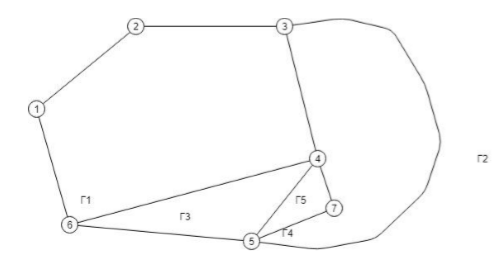
2. За допомогою γ-алгоритма зробити укладку графа у площині, або довести що вона неможлива.

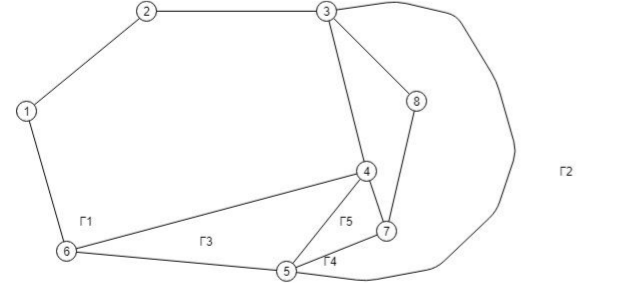


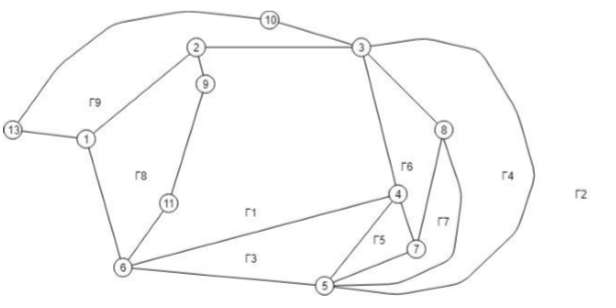
1. Вибрав простий цикл через вершини 1-2-3-4-5-6. Та виділив всі можливі сегменти.



Ребра 4-6 та 5-3 можуть бути розміщені у двох гранях тому вибираю одне з них(6-4) і вношу в середину циклу. Тепер ребро 5-3 можу бути тільки в одній грані, тому проводжу його в грані 2.







1. Оскільки всі сегменти можуть бути в двох гранях ,я вибираю будь-яке з них. Проводжу ребра 5-7-4 та 7-8-3(воно може знаходитись лише в одній грані).
2. Тепер вношу 5-8. Оскільки 6-11-9-2 та 1-13-10-3 можуть знаходитись у двох гранях, одну вношу всередину, тоді інше може знаходитись лише в одній грані, тому вписую його ззовні, у грань 2.
3. Сегменти, що сполучаються з вершиною 12 не можуть буди занесені в граф, оскільки відбудеться перетин ребер, що суперечить завданню, а отже граф не є планарним.

**Завдання з додатку 2:**

**Код:**

#include <stdio.h>  
#include <stdlib.h>  
#define M 100500  
  
typedef struct  
{  
    int n1, n2, w;  
} edge;  
  
int main(void)  
{  
    int size;  
    int N;  
    int end = -1;  
    int minind;  
    int min;  
    int temp;  
  
    printf("Enter numbers of nodes: ");  
    scanf("%i", &N);  
    printf("Enter numbers of edges: ");  
    scanf("%i", &size);  
  
    int\* nodes = (int\*) malloc(N \* sizeof(int));  
    edge\* edges = (edge\*) malloc(size \* sizeof(edge));  
    int\* d = (int\*) malloc(N \* sizeof(int));  
    int\* ver = (int\*) malloc( N \* sizeof(int));  
  
    for(int i = 0; i < N; i++)  
        nodes[i] = 0;  
  
    for(int i = 0; i < size; i++)  
    {  
        printf("Enter edge start end weight: ");  
        scanf("%i %i %i", &edges[i].n1, &edges[i].n2, &edges[i].w);  
    }  
  
    printf("Enter end of way: ");  
  
    while(end < 1 || end > N)  
        scanf("%i", &end);  
  
    end--;  
    for(int i = 0; i < N; i++)  
        d[i] = M;  
  
        d[0] = 0;  
    do  
    {  
        minind = M;  
        min = M;  
  
        for(int i = 0; i < N; i++)  
        {  
            if(nodes[i] == 0 && d[i] < min)  
            {  
             min = d[i];  
             minind = i;  
            }  
        }  
  
        if(minind != M)  
        {  
            for(int i = 0; i < size; i++)  
            {  
                if(edges[i].n1 == minind)  
                {  
                    temp = min + edges[i].w;  
  
                    if(temp < d[edges[i].n2])  
                        d[edges[i].n2] = temp;  
                }  
            }  
            nodes[minind] = 1;  
        }  
    }while(minind < M);  
  
    printf("Distance:\n");  
  
    for(int i = 0; i < N; i++)  
        printf("1 - %i = %i\n", i+1, d[i]);  
  
    int weight = d[end];  
    int k = 1;  
  
    ver[0] = end + 1;  
  
    while(end != 0)  
    {  
        for(int i = 0; i < size; i++)  
        {  
            if(edges[i].n2 == end)  
            {  
                temp = weight - edges[i].w;  
                if(temp == d[edges[i].n1])  
                {  
                    weight = temp;  
                    end = edges[i].n1;  
                    ver[k] = edges[i].n1 + 1;  
                    k++;  
                }  
            }  
        }  
    }  
  
    printf("The shortest way:\n");  
  
    for(int i = k-1; i >= 0; i--)  
        printf("%i ", ver[i]);  
  
    printf("\n");  
}

