МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота №5 на тему:

«Знаходження найкоротшого маршруту за алгоритмом Дейкстра. Плоскі планарні графи»

з дисципліни «Дискретна математика»

Виконав:

студент групи КН-113

Добосевич Данило

Викладач:

Мельникова Н. І.

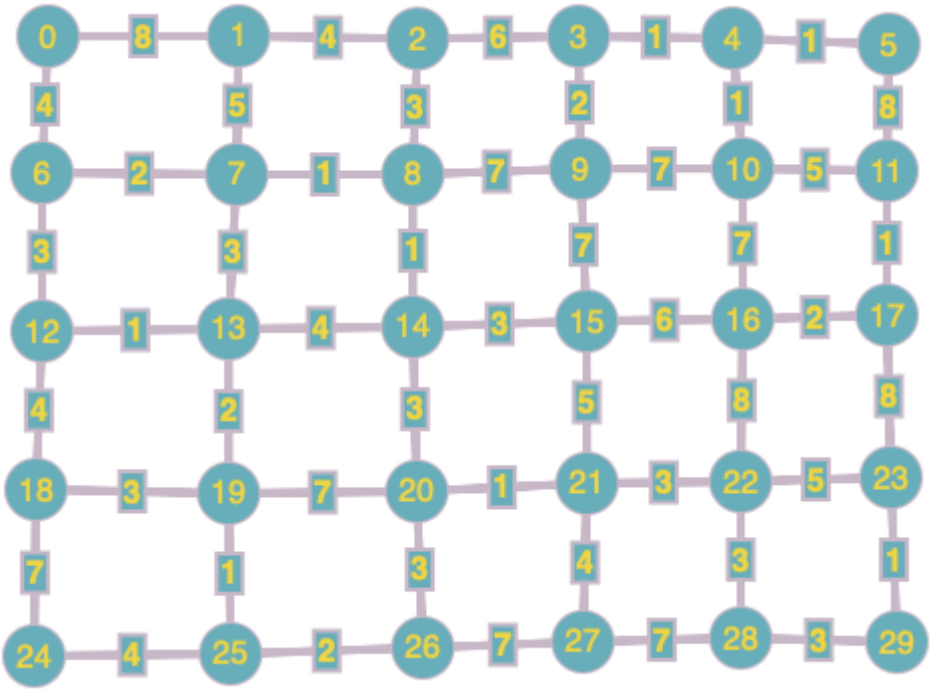
Львів 2019

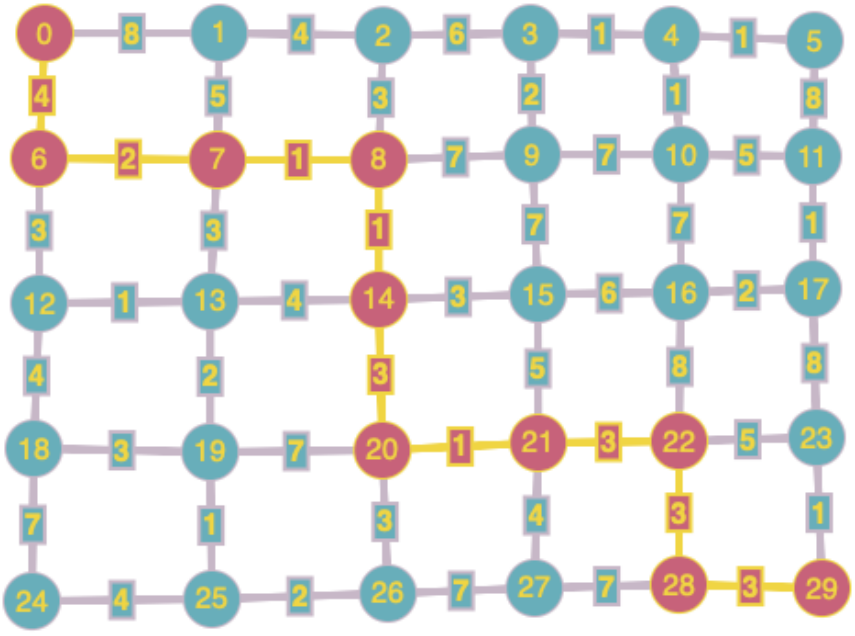
Мета роботи: набуття практичних вмінь та навичок з використання алгоритму Дейкстра.

Варіант 8. 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* .



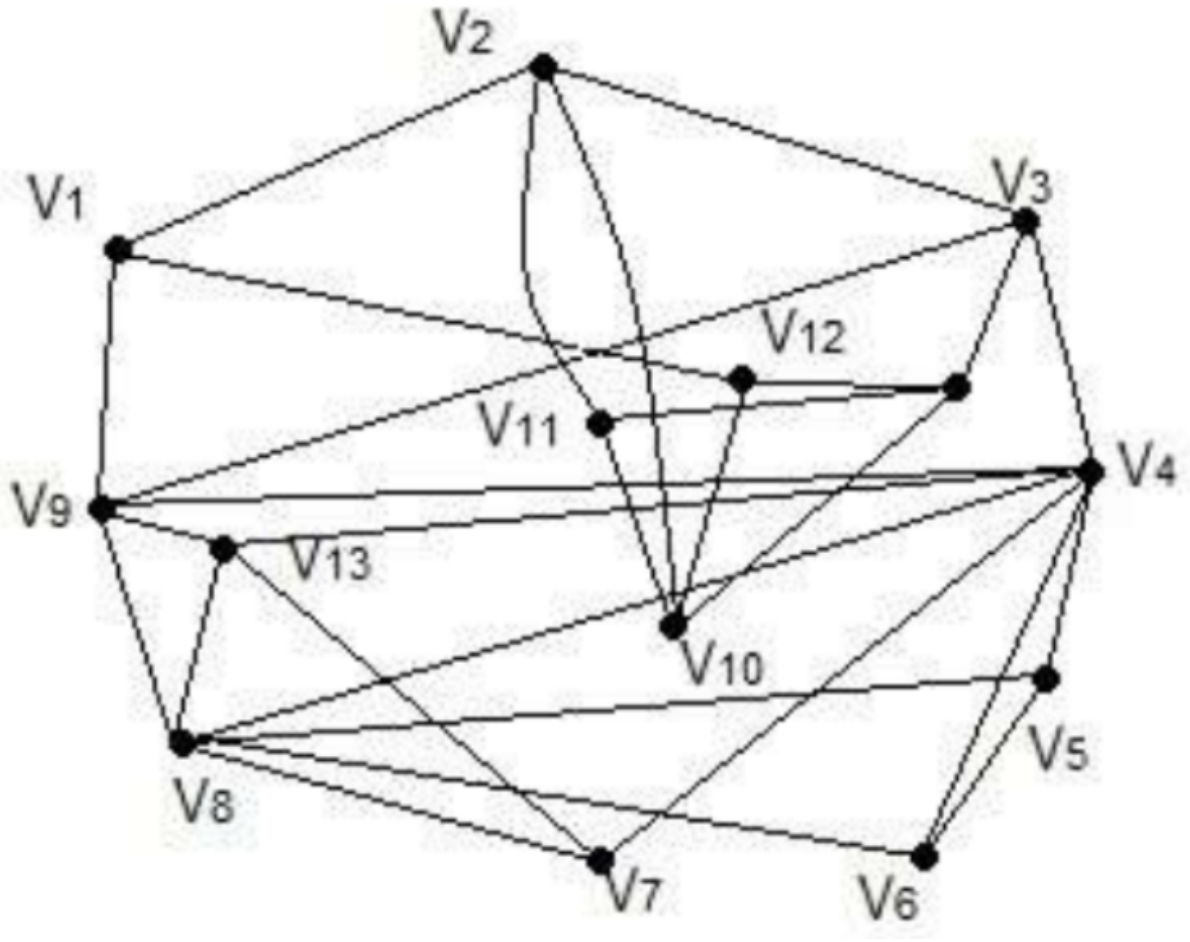


1.Відповідь:

V0->V6 = 4, V0->V7=6, V0->V8=7, V0->V14=8, V0->V20=11, V0->V21=12, V0->V22=15, V0->V28=18, V0->V29(V*)==21

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

8

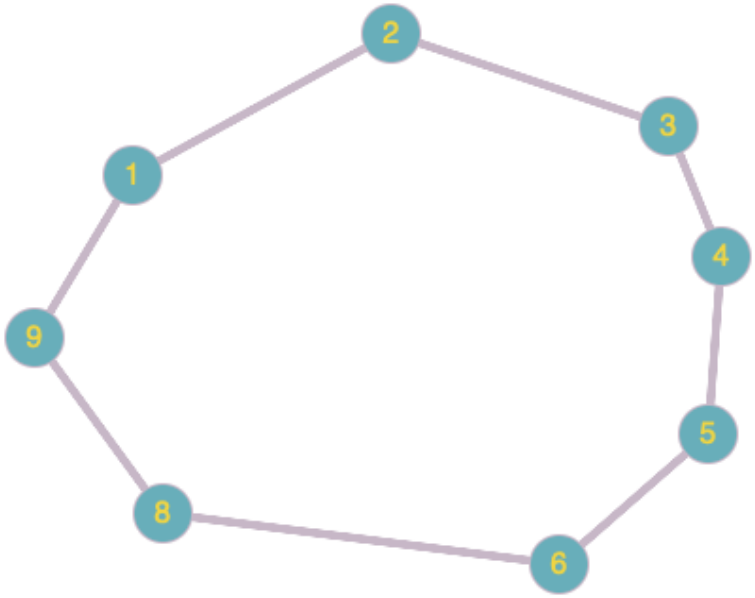


2. Відповідь:

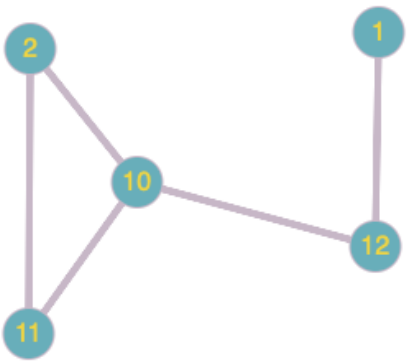
Якщо граф не є планарний, то він повинен містити більше 4 вершин, степінь яких є >3, або більше 5 вершин, степінь яких = 2 - це необхідна умова укладки графа.

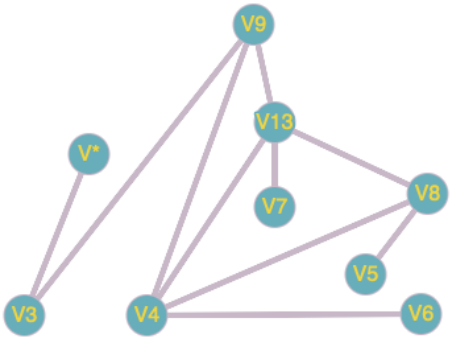
У моєму варіанті ця умова не виконується, тому цей граф є планарним.

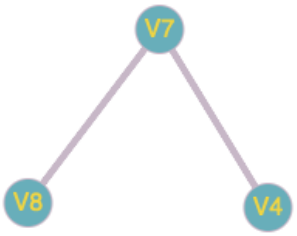
Обираю найбільший цикл графа.



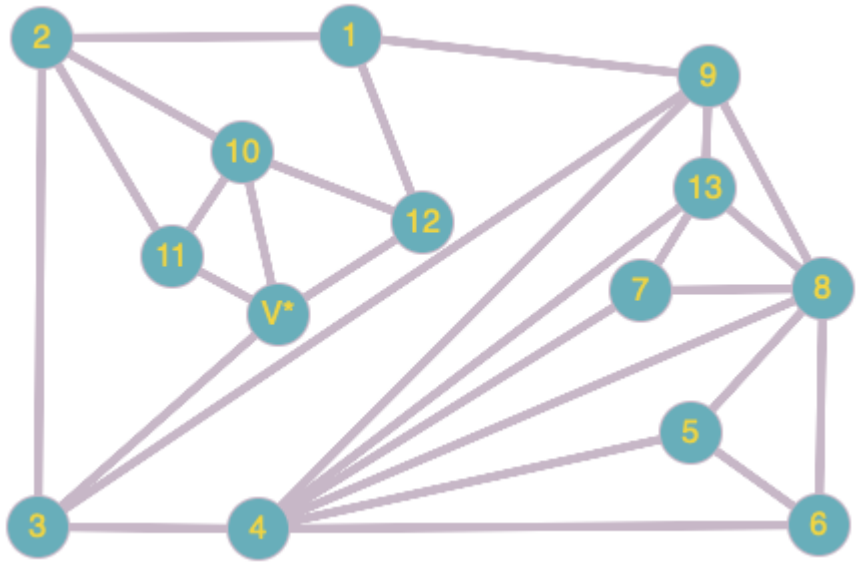
Обираю решту вершин та ребер.







Ось результат:



Варіант 8. 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:

```c
#include <stdio.h>

void print_the_shortest_way(int array [30][4],int node) {
    if(array[node][3]==-1) {
        printf("V%d=%d ",array[node][0],array[node][2]);
    }
    else {
        print_the_shortest_way(array, node: array[node][3]);
        printf("V%d=%d ",array[node][0],array[node][2]);
    }
}

int main(void) {
    int i,k,lenth=0;
    int result [30] [2];
    int nodes [30] [4] = { //{number,times_used,path_value,last_node}
        {1,1,0,-1},{2,0,0,0},{3,0,0,0},{4,0,0,0},{5,0,0,0},{6,0,0,0},
        {7,0,0,0},{8,0,0,0},{9,0,0,0},{10,0,0,0},{11,0,0,0},{12,0,0,0},
        {13,0,0,0},{14,0,0,0},{15,0,0,0},{16,0,0,0},{17,0,0,0},{18,0,0,0},
        {19,0,0,0},{20,0,0,0},{21,0,0,0},{22,0,0,0},{23,0,0,0},{24,0,0,0},
        {25,0,0,0},{26,0,0,0},{27,0,0,0},{28,0,0,0},{29,0,0,0},{30,0,0,0}
    };
    int edges [49] [3] = { //{from,to,weight}
        {1,2,3},{2,3,4},{3,4,8},{4,5,6},{5,6,4},
        {1,7,1},{2,8,2},{3,9,3},{4,10,4},{5,11,5},{6,12,8},
        {7,8,4},{8,9,1},{9,10,1},{10,11,2},{11,12,7},
        {7,13,7},{8,14,7},{9,15,1},{10,16,4},{11,17,1},{12,18,3},
        {13,14,4},{14,15,5},{15,16,2},{16,17,7},{17,18,7},
        {13,19,1},{14,20,7},{15,21,3},{16,22,3},{17,23,2},{18,24,7},
        {19,20,8},{20,21,3},{21,22,1},{22,23,1},{23,24,5},
        {19,25,5},{20,26,3},{21,27,3},{22,28,1},{23,29,2},{24,30,8},
        {25,26,1},{26,27,7},{27,28,3},{28,29,6},{29,30,3}
    };
    //sorting edges
    int temp,pointer;
    do{
        pointer=0;
        for (i = 0; i < 48; i++){
            if (edges[i][2] > edges[i + 1][2]) {
                for (k = 0, temp = 0; k < 3; k++) {
                    temp = edges [i + 1][k];
```

```
                    temp = edges [i + 1][k];
                    edges [i + 1][k] = edges [i][k];
                    edges [i][k] = temp;
                }
                pointer++;
            }
        }
    }while(pointer!=0);

    while(nodes[29][1]==0) {
        for(k=1;k<9;k++) {
            for (i = 0; i < 49; i++) {
                if (nodes [edges[i][0] - 1][1] != nodes [edges[i][1] - 1][1] && edges[i][2]==k) {
                    if (nodes [edges[i][0] - 1][1] == 0 && nodes [edges[i][1] - 1][1] != 0) {
                        nodes [edges[i][0] - 1][1] = nodes [edges[i][1] - 1][1];
                        nodes [edges[i][1] - 1][1] = nodes [edges[i][1] - 1][1] + 1;
                        nodes [edges[i][0] - 1][2] = nodes [edges[i][1] - 1][2] + edges[i][2];
                        result[lenth][0] = nodes[edges[i][0] - 1][0] - 1;
                        result[lenth][1] = nodes[edges[i][0] - 1][2];
                        nodes [edges[i][0] - 1][3] = nodes [edges[i][1] - 1][0]-1;
                        lenth++;
                    }
                    else if (nodes [edges[i][0] - 1][1] != 0 && nodes [edges[i][1] - 1][1] == 0) {
                        nodes [edges[i][1] - 1][1] = nodes [edges[i][0] - 1][1];
                        nodes [edges[i][0] - 1][1] = nodes [edges[i][0] - 1][1] + 1;
                        nodes [edges[i][1] - 1][2] = nodes [edges[i][0] - 1][2] + edges[i][2];
                        result[lenth][0] = nodes[edges[i][1] - 1][0] - 1;
                        result[lenth][1] = nodes[edges[i][1] - 1][2];
                        nodes [edges[i][1] - 1][3] = nodes [edges[i][0] - 1][0]-1;
                        lenth++;
                    }
                }
            }
        }
    }
    //sorting valuable nodes
    do{
        pointer=0;
        for (i = 0; i < lenth; i++){
            if (result[i][1] > result[i + 1][1]) {
                for (k = 0, temp = 0; k < 2; k++) {
                    temp = result [i + 1][k];
                    result[i + 1][k] = result[i][k];
                    result[i][k] = temp;
                }
                pointer++;
            }
        }
    }while(pointer!=0);
    printf("Used edges: \n");
    for(i=1;i<=lenth;i++) {
        printf("V%d = %d\n",result[i][0],result[i][1]);
    }
    printf("the shortest way is: \n");
    print_the_shortest_way(nodes, node: 29);

    return 0;
}
```

Результат роботи:

```
Used edges:
V6 = 1
V1 = 3
V7 = 5
V8 = 6
V2 = 7
V9 = 7
V14 = 7
V12 = 8
V10 = 9
V15 = 9
V20 = 10
V13 = 12
V21 = 12
V26 = 13
V4 = 14
V3 = 15
V27 = 16
V11 = 16
V16 = 16
V19 = 19
V25 = 22
V28 = 22
V17 = 23
V5 = 24
V18 = 27
V23 = 30
V24 = 32
V29 = 38
the shortest way is:
V1=0 V7=1 V8=5 V9=6 V15=7 V16=9 V17=16 V18=23 V24=30 V30=38
Process finished with exit code 0
```

Висновок: Виконуючи цю роботу, я набув практичних вмінь та навичок з використання алгоритму Дейкстра.