МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА

Кафедра систем штучного інтелекту

Лабораторна робота №5

з дисципліни «Дискретна математика»

Виконала:

студентка групи КН-115

Попів Христина

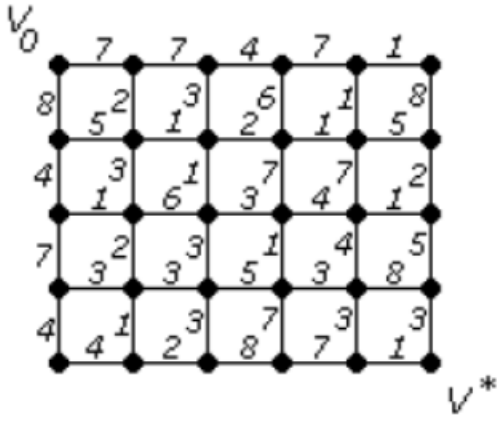
Викладач:

Мельникова Н.І.

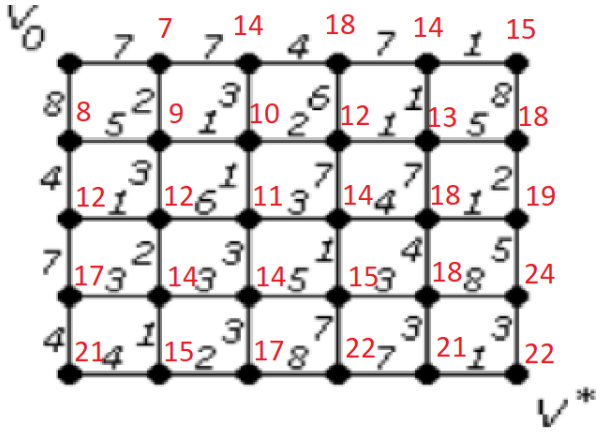
Львів – 2019 р.

Варіант 10

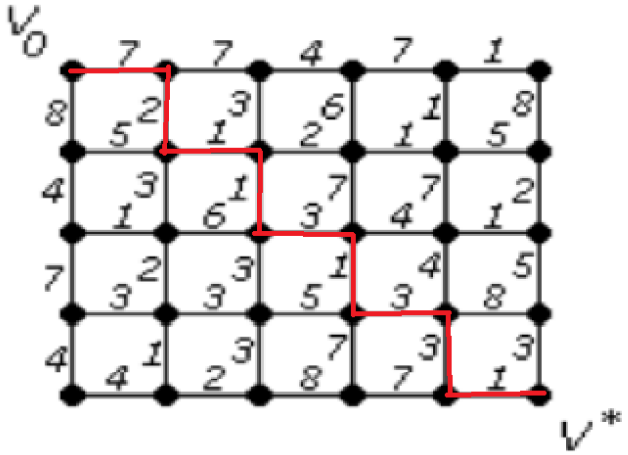
1.За допомогою алгоритму Дейкстра знайти найкоротший шлях у графі поміж парою вершин V0 і V* .



Розв'язання



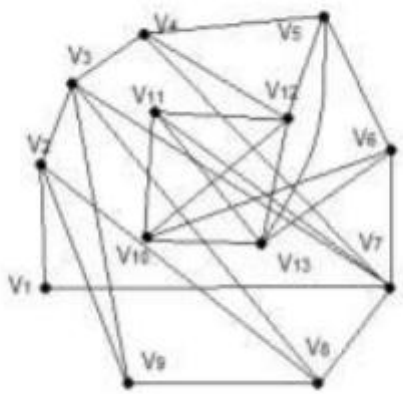
Найкоротший шлях:



Отже, найкоротша відстань від вершини 1 до вершини 30 рівна **22**.

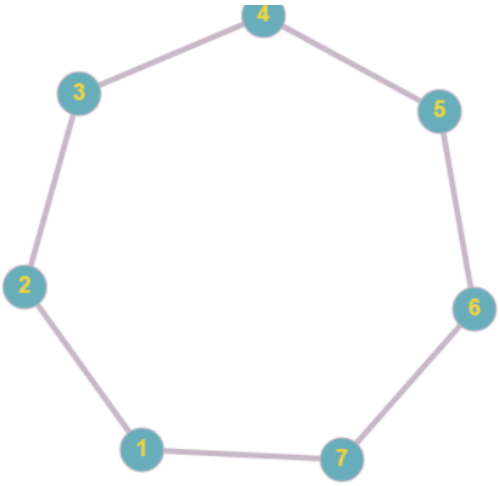
2.За допомогою γ -алгоритма зробити укладку графа у площині, або довести що вона неможлива.

10

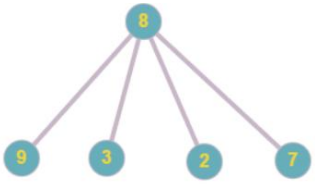


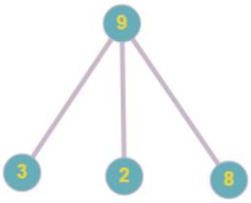
Розв’язання

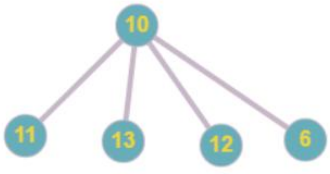
Вибираємо з даного графа простий цикл:

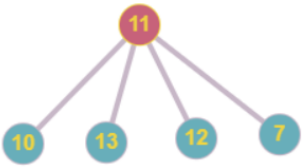


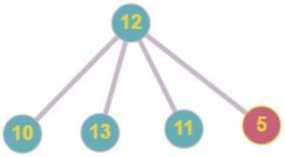
Розбиваємо граф на сегменти:

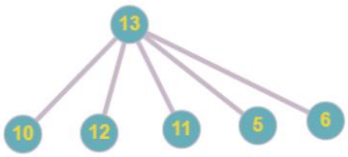


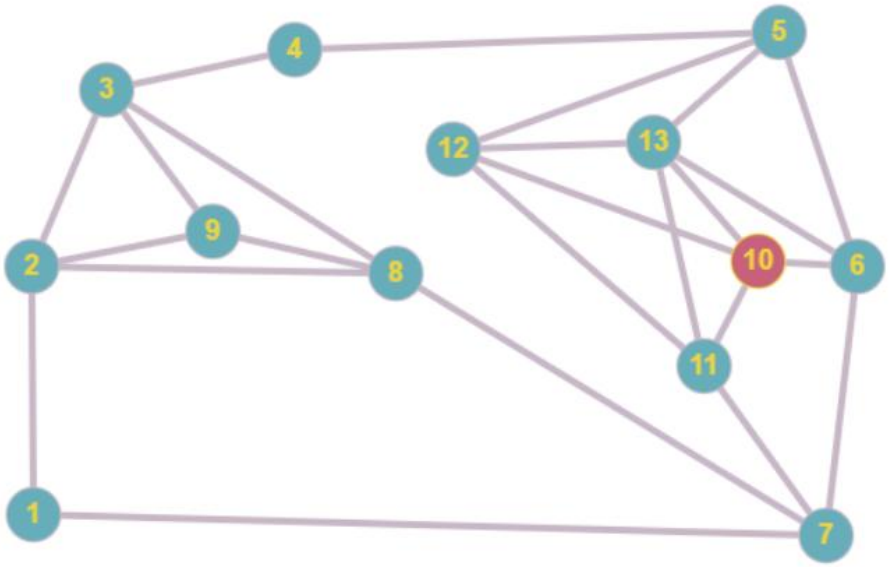






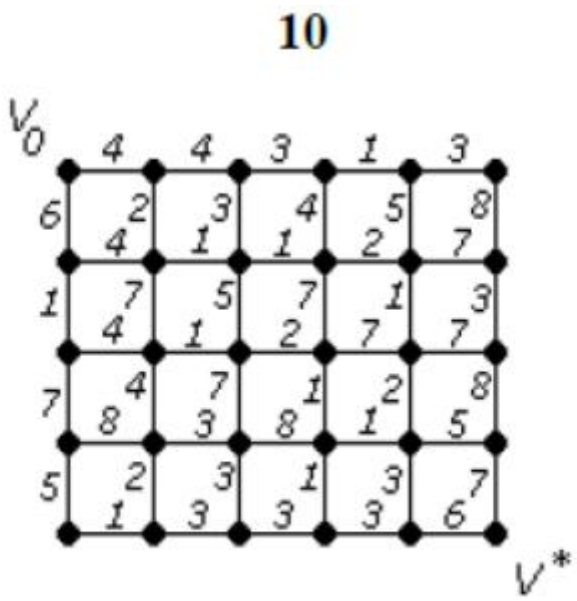






Отже , укладку графа у площині є неможливою , оскільки існує перетен ребер.

Завдання №2.Написати програму, яка реалізує алгоритм Дейкстри знаходження найкоротшого шляху між парою вершин у графі. Протестувати розроблену програму на графі згідно свого варіанту.



Текст програми:

```
#include<iostream>
#include<stdio.h>
using namespace std;
#define INFINITY 9999
#define max 30
void dijkstra(int G[max][max], int n, int startnode);
int main() {
	int G[max][max] = {
{0,4,0,0,0,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
```

```
{4,0,4,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,4,0,3,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,3,0,1,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,1,0,3,0,0,0,0,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,3,0,0,0,0,0,0,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{6,0,0,0,0,0,0,4,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,2,0,0,0,0,4,0,1,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,3,0,0,0,0,1,0,1,0,0,0,0,5,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,4,0,0,0,0,1,0,2,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,5,0,0,0,0,2,0,7,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,8,0,0,0,0,7,0,0,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,1,0,0,0,0,0,0,4,0,0,0,0,7,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,7,0,0,0,0,4,0,1,0,1,0,0,4,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,5,0,0,0,0,1,0,2,0,0,0,0,7,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,7,0,0,0,0,2,0,7,0,0,0,0,1,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,7,0,7,0,0,0,0,2,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,7,0,0,0,0,0,0,8,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,0,0,8,0,0,0,0,5,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,4,0,0,0,0,8,0,3,0,0,0,0,2,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,3,0,8,0,0,0,0,3,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,8,0,1,0,0,0,0,1},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,1,0,5,0,0,0,3},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,0,0,0,0,5,0,0,0,0,7},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,0,0,0,0,0,1,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,1,0,3,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,3,0,3},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,3,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,3},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0}

};
	int n = 30;
	int u = 0;
	dijkstra(G, n, u);
	return 0;
}
void dijkstra(int G[max][max], int n, int startnode) {
	int cost[max][max], distance[max], pred[max];
	int visited[max], count, mindistance, nextnode, i, j;
	for (i = 0; i < n; i++)
		for (j = 0; j < n; j++)
			if (G[i][j] == 0)
				cost[i][j] = INFINITY;
			else
				cost[i][j] = G[i][j];
	for (i = 0; i < n; i++) {
		distance[i] = cost[startnode][i];
		pred[i] = startnode;
		visited[i] = 0;
	}
	distance[startnode] = 0;
	visited[startnode] = 1;
	count = 1;
	while (count < n - 1) {
		mindistance = INFINITY;
		for (i = 0; i < n; i++)
			if (distance[i] < mindistance && !visited[i]) {
				mindistance = distance[i];
				nextnode = i;
			}
		visited[nextnode] = 1;
		for (i = 0; i < n; i++)
			if (!visited[i])
				if (mindistance + cost[nextnode][i] < distance[i]) {
					distance[i] = mindistance + cost[nextnode][i];
					pred[i] = nextnode;
				}
		count++;
```

```
	}
	for (i = 0; i < n; i++)
		if (i != startnode) {
			cout << "\nDistance of node" << i << "=" << distance[i];
			cout << "\nPath=" << i;
			j = i;
			do {
				j = pred[j];
				cout << "<-" << j;
			} while (j != startnode);
		}
}
```

Результат:

```
Distance of node1=4
Path=1<-0
Distance of node2=8
Path=2<-1<-0
Distance of node3=11
Path=3<-2<-1<-0
Distance of node4=12
Path=4<-3<-2<-1<-0
Distance of node5=15
Path=5<-4<-3<-2<-1<-0
Distance of node6=6
Path=6<-0
Distance of node7=6
Path=7<-1<-0
Distance of node8=7
Path=8<-7<-1<-0
Distance of node9=8
Path=9<-8<-7<-1<-0
Distance of node10=10
Path=10<-9<-8<-7<-1<-0
Distance of node11=17
Path=11<-10<-9<-8<-7<-1<-
Distance of node12=7
Path=12<-6<-0
Distance of node13=11
Path=13<-12<-6<-0
Distance of node14=12
Path=14<-8<-7<-1<-0
Distance of node15=11
Path=15<-10<-9<-8<-7<-1<-
Distance of node16=12
Path=16<-13<-12<-6<-0
Distance of node17=19
Path=17<-16<-13<-12<-6<-0
Distance of node18=14
```

```
Distance of node19=15
Path=19<-13<-12<-6<-0
Distance of node20=18
Path=20<-19<-13<-12<-6<-0
Distance of node21=12
Path=21<-15<-10<-9<-8<-7<-1<-0
Distance of node22=13
Path=22<-21<-15<-10<-9<-8<-7<-1<-0
Distance of node23=18
Path=23<-22<-21<-15<-10<-9<-8<-7<-1<-0
Distance of node24=18
Path=24<-25<-19<-13<-12<-6<-0
Distance of node25=17
Path=25<-19<-13<-12<-6<-0
Distance of node26=16
Path=26<-27<-21<-15<-10<-9<-8<-7<-1<-0
Distance of node27=13
Path=27<-21<-15<-10<-9<-8<-7<-1<-0
Distance of node28=16
Path=28<-22<-21<-15<-10<-9<-8<-7<-1<-0
Distance of node29=22
Path=29<-28<-22<-21<-15<-10<-9<-8<-7<-1<-0
```

Висновок: на цій лабораторній роботі я навчився знаходити найкоротший шлях за алгоритмом Дейкстри та укладати граф за допомогою алгоритму γ-укладання графа.