**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”**

**Кафедра систем  штучного інтелекту**

**Лабораторна робота 5**

<<Дискретна математика>>

Виконав:

                                                                                   студент групи КН-114

                                                                                        Микицький Назар

                                                                                                      Викладач:

                                                                                               Мельникова Н.І

Тема: Знаходження найкоротшого маршруту за алгоритмом Дейкстри. Плоскі планарні графи  
Мета роботи: набуття практичних вмінь та навичок з використання алгоритму Дейкстри

**ТЕОРЕТИЧНІ ВІДОМОСТІ ТА ПРИКЛАДИ РОЗВ’ЯЗАННЯ ЗАДАЧ**Задача знаходження найкоротшого шляху з одним джерелом полягає у  
знаходженні найкоротших(мається на увазі найоптимальніших за  
вагою) шляхів від деякої вершини(джерела) до всіх вершин графа *G* .  
Для розв’язку цієї задачі використовується «жадібний» алгоритм, який  
називається алгоритмом Дейкстри.  
«Жадібними» називаються алгоритми, які на кожному кроцівибирають оптимальний із можливих варіантів.кроку s+2. Алгоритм γукладання графа *G* являє собою процес  
послідовного приєднання до деякого укладеного підграфа *G*~ графа *G*нового ланцюга, обидва кінці якого належать *G* . При цьому в якості  
початкового плоского графа *G*~ вибирається будь-який простий цикл  
графа *G* . Процес продовжується доти, поки не буде побудовано  
плоский граф, ізоморфний графові *G* , або приєднання деякого  
ланцюга виявиться неможливим. В останньому випадку граф *G* не є  
планарным.Нехай побудоване деяке укладання підграфа *G*~ графа *G* .  
*Сегментом S відносно G* будемо називати підграф графа G  
одного з наступних виглядів:- ребро *e*∈*E* , *e* =(*u*, *v*), таке, що *e*∉*E*~ , *u*,*v* ∈*V*~,  
*G*~ =(*V*~, *E*~)- зв'язний компонент графа *G* – *G*~ , доповнений всіма  
ребрами графа *G* , інцидентними вершинам узятого компонента,  
і кінцями цих ребер.Вершину v сегмента S відносно *G* будемо називати  
*контактною*, якщо *v* ∈*V*.*Припустимою гранню для сегмента S* відносно *G*~ називається  
грань Г графа *G* , що містить усі контактні вершини сегмента S. Через  
Г(S) будемо позначати множину припустимих граней для S.  
Назвемо α*-ланцюгом* простий ланцюг L сегмента S, що містить  
дві різні контактні вершини і не містить інших контактних вершин.  
Тепер формально опишемо алгоритм γ.  
0. Виберемо деякий простий цикл С графа *G* і укладемо  
його на площині; покладемо *G*~ =*G* .  
1. Знайдемо грані графа *G*~ і сегменти відносно *G*~ . Якщо  
множина сегментів порожня, то перейдемо до пункту 7.  
2. Для кожного сегмента S визначимо множину Г(S).  
3. Якщо існує сегмент S, для якого Г(S)=∅, то граф G не  
планарний. Кінець. Інакше перейдемо до п. 4.  
4. Якщо існує сегмент S, для якого мається єдина  
припустима грань Г, то перейдемо до п. 6. Інакше до п. 5.5. Для деякого сегмента S Г(S)>1. У цьому випадку  
вибираємо довільну припустиму грань Г.

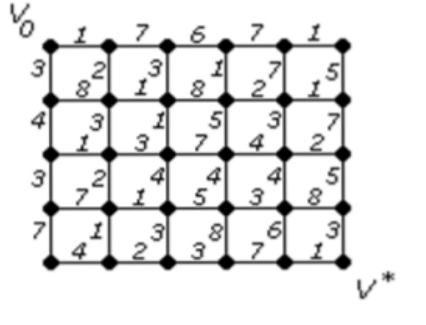
|  |  |
| --- | --- |
| 6.  замінимо | Розмістимо довільний α- ланцюг L∈S у грань Г; |

*G*~ на *G*~ Υ*L* і перейдемо до п. 1.  
7. Побудовано укладання *G*~ графа G на площині. Кінець.

Кроком алгоритму γбудемо вважати приєднання до *G* α- ланцюга L.

**Варіант 14**

1. За допомогою алгоритму Дейкстра знайти найкоротший шлях  
   у графі поміж парою вершин *V*0 і *V*\* .



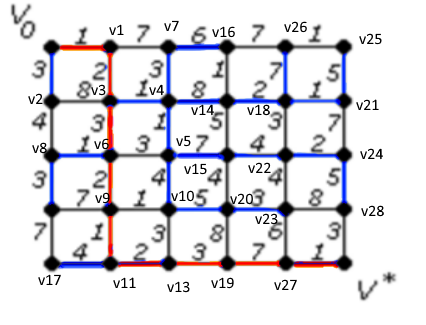
Будем позначати найближчі вершини(v1,v2,v3…) у порядку їх появи l(v1)=1,l(v2)=3,l(v3)=3,l(v4)=4,l(v5)=5,l(v6)=6,l(v7)=7,l(v8)=7,l(v9)=8,l(v10)=9,l(v11)=9

l(v12)=10,l(v13)=11, l(v14)=12, l(v15)=12, l(v16)=13,l(v17)=13, l(v18)=14,l(v19)=14,

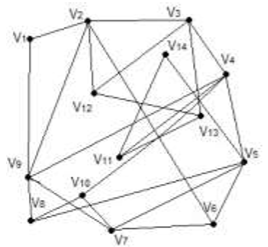
l(v20)=14,l(v21)=15,l(v22)=16,l(v23)=17,l(v24)=18,l(v25)=20,l(v26)=21, l(v27)=21, l(v\*)=22, l(v28)=23

Дерево найближчих вершин виділено синіми лініями і є кістяковим деревом, тому що містить усі вершини графа.

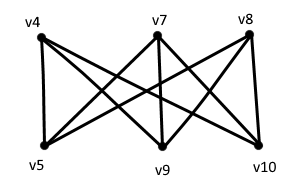
Шуканий найкоротший ланцюг: є (v0,v1,v3,v6,v9,v11,v13,v19,v27,v\*) вага 22.

****

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



Перш ніж робити укладку потрібно перевірити чи вона можлива. Оскільки ми можемо виділити підграф K3,3



То за теоремою Куратовського граф не є планарним а значить його не можна укласти на площину.

**Програмна частина**

**Код програми**

#include<locale>

#include<iostream>

#include <stdio.h>

#include <stdlib.h>

#define SIZE 30

using namespace std;

int main()

{

setlocale(LC\_ALL, "Ukrainian");

int arr[SIZE][SIZE];

int dov[SIZE]; // min vid

int v[SIZE]; // ver

int temp, index, min;

int begin\_index = 0;

for (int i = 0; i < SIZE; i++)

{

for (int j = 0; j < SIZE; j++) {

arr[i][j] = 0;

}

}

arr[0][1] =1;

arr[0][6] =4 ;

arr[1][0] =1 ;

arr[1][2] =1 ;

arr[1][7] =8 ;

arr[2][1] =1;

arr[2][8] =3 ;

arr[2][3] =6 ;

arr[3][2] =6 ;

arr[3][4] =3;

arr[3][9] =1 ;

arr[4][3] =3 ;

arr[4][5] =3 ;

arr[4][10] =5 ;

arr[5][4] =3;

arr[5][11] =7 ;

arr[6][0] =4;

arr[6][7] =2;

arr[6][12] =1;

arr[7][1] =8 ;

arr[7][6] =2;

arr[7][8] =1;

arr[7][13] =7 ;

arr[8][7] =1 ;

arr[8][2] =3;

arr[8][14] =1;

arr[8][9] =4;

arr[9][8] =4;

arr[9][3] =1 ;

arr[9][15] =4;

arr[9][10] =2 ;

arr[10][9] =2;

arr[10][4] =5;

arr[10][16] =3;

arr[10][11] =4 ;

arr[11][10] =4 ;

arr[11][5] =7 ;

arr[11][17] =7 ;

arr[12][6] =1;

arr[12][13] =1 ;

arr[12][18] =5 ;

arr[13][12] =1;

arr[13][7] =7;

arr[13][19]=4;

arr[13][14] =3;

arr[14][13] =3;

arr[14][8] =1;

arr[14][20] =7;

arr[14][15] =2;

arr[15][14] =2;

arr[15][9] = 4;

arr[15][21] =8;

arr[15][16] =5;

arr[16][15] = 5;

arr[16][10] =3 ;

arr[16][22] = 2;

arr[16][17] =7;

arr[17][16] =7;

arr[17][11] =7;

arr[17][23] =5 ;

arr[18][12] =5 ;

arr[18][19] =7 ;

arr[18][24] =3 ;

arr[19][18] =7;

arr[19][13] =4;

arr[19][25] =2;

arr[19][20] =3 ;

arr[20][19] =3 ;

arr[20][14] =7 ;

arr[20][26] =1;

arr[20][21] =1;

arr[21][20] =1;

arr[21][15] =8;

arr[21][27] =8;

arr[21][22] =1 ;

arr[22][21] =1;

arr[22][16] =2;

arr[22][28] =8;

arr[22][23] =8;

arr[23][22] =8;

arr[23][17] =5;

arr[23][29] =7;

arr[24][18] =3;

arr[24][25] =4;

arr[25][24] =4;

arr[25][19] =2;

arr[25][26] = 7;

arr[26][25] = 7;

arr[26][20] =1;

arr[26][27] =3;

arr[27][26] =3;

arr[27][21] =8;

arr[27][28] =3;

arr[28][27] =3;

arr[28][22] =8;

arr[28][29] =6;

arr[29][28] =6;

arr[29][23] =7;

// занул

for (int i = 0; i < SIZE; i++)

{

dov[i] = 10000;

v[i] = 1;

}

dov[begin\_index] = 0;

do {

index = 10000;

min = 10000;

for (int i = 0; i < SIZE; i++)

{

if ((v[i] == 1) && (dov[i] < min))

{

min = dov[i];

index = i;

}

}

if (index != 10000)

{

for (int i = 0; i < SIZE; i++)

{

if (arr[index][i] > 0)

{

temp = min + arr[index][i];

if (temp < dov[i])

{

dov[i] = temp;

}

}

}

v[index] = 0;

}

} while (index < 10000);

//cout << "найкортший шлях до вершин" << endl;

//for (int i = 0; i < SIZE; i++)

//cout<<d[i]<<" ";

// шлях назад

int ver[SIZE]; // відвід верш

int end = 29;

ver[0] = end ;

int k = 1;

int weight = dov[end]; //

while (end != begin\_index)

{

for (int i = 0; i < SIZE; i++)

if (arr[end][i] != 0) // чи є звязок

{

int temp = weight - arr[end][i]; // вага з поперед

if (temp == dov[i]) // якщ співпав

{

weight = temp;

end = i;

ver[k] = i ;

k++;

}

}

}

cout << endl;

cout << "шлях" << endl;

for (int i = k - 1; i >= 0; i--) {

cout << ver[i] << " ";

}

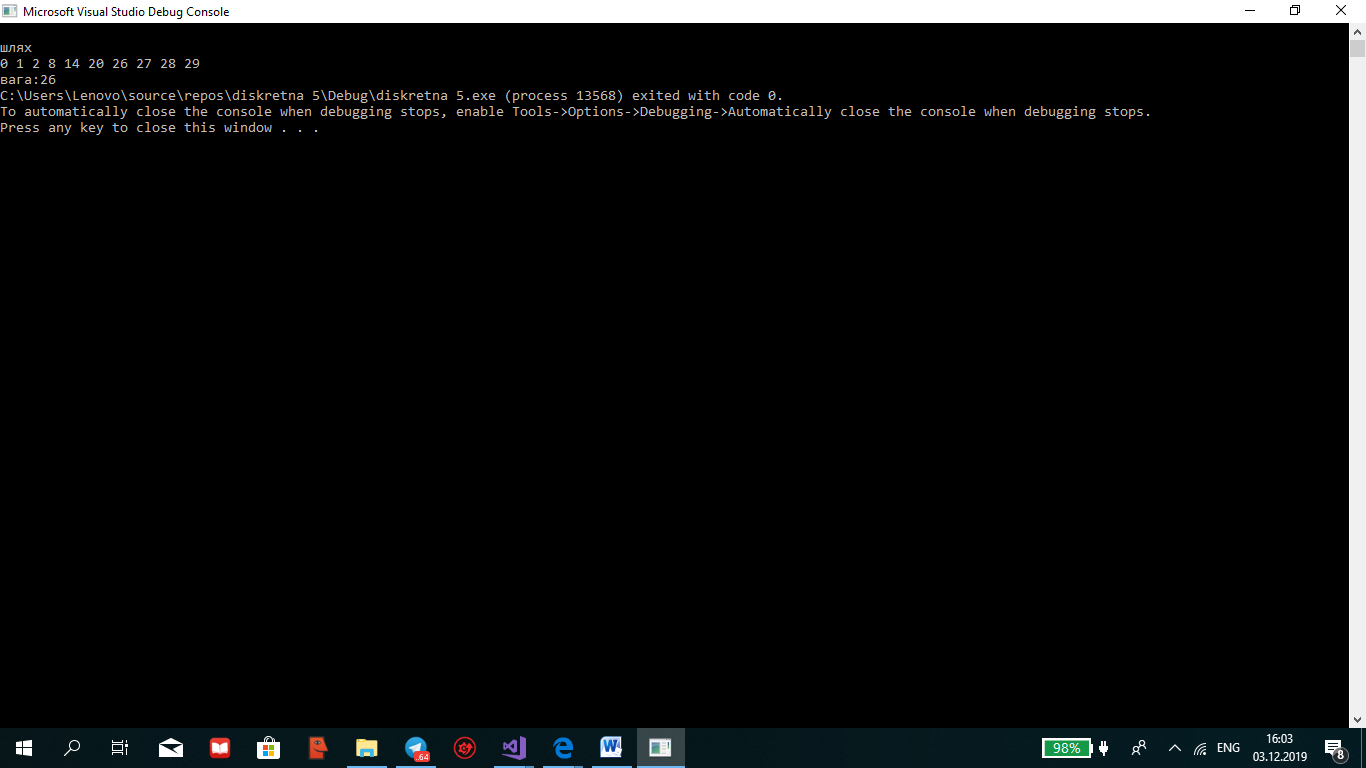
cout << endl;

cout <<"вага:"<< dov[29];

return 0;

}

**Результат роботи програми**

**Висновок:** я набув практичних вмінь та навичок з використання алгоритму