**Міністерство освіти і науки України**

**Національному університеті "Львівська**

**Політехніка"**

**Кафедра систем штучного інтелекту**

**Лабораторна робота № 5**

з дисципліни

**Виконав:**

студентка групи КН-114

Церковник Оксана

**Викладач:**

Мельникова Н.І.

Львів - 2019р.

**Лабораторна робота № 5.**

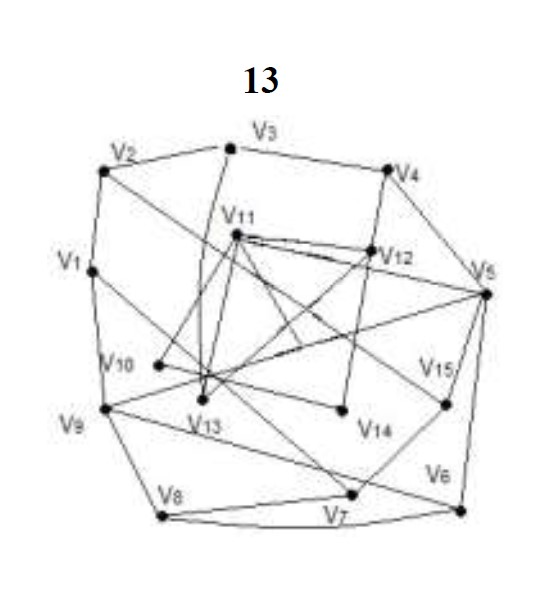
**Тема:** Знаходження найкоротшого маршруту за алгоритмом Дейкстри. Плоскі планарні графи  
**Мета роботи:** набуття практичних вмінь та навичок з використання алгоритму Дейкстри.

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

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 кроці  
вибирають оптимальний із можливих варіантів.  
Задача про найкоротший ланцюг. Алгоритм Дейкстри.  
Дано n-вершинний граф *G* = (*V*, *E*), у якому виділено пару  
вершин *v v* ∈*V* \*  
0 , , і кожне ребро зважене числом *w*(*e*) ≥ 0 . Нехай  
*X*= { *x*} – множина усіх простих ланцюгів, що з'єднують 0 *v* з \* *v* ,  
( ) *x x E V x* , = . Цільова функція min ) ( ) ( → =∑  
∈*e Ex*  
*F x w e* . Потрібно  
знайти найкоротший ланцюг, тобто : 0 *x* ∈*X* ( ) min ( ) 0 *F x F x*  
*x*∈*X*  
=  
Перед описом алгоритму Дейкстри подамо визначення термінів  
“k-а найближча вершина і “дерево найближчих вершин”. Перше з цих  
понять визначається індуктивно так.  
1-й крок індукції. Нехай зафіксовано вершину x0, E1 – множина  
усіх ребер e ∈E, інцидентних v0. Серед ребер e ∈E1 вибираємо ребро  
e(1) =(v0, v1), що має мінімальну вагу, тобто ( (1)) min ( )  
1  
*w e w e*  
*e*∈*E*  
= . Тоді  
v1 називаємо першою найближчою вершиною (НВ), число w(e(1))  
позначаємо l(1) = l(v1) і називаємо відстанню до цієї НВ. Позначимо  
V1={v0, v1} – множину найближчих вершин.  
2-й крок індукції. Позначимо E2 – множину усіх ребер e=(*v*′,*v*′′), e ∈E,  
таких що *v*′∈V1, *v*′′∈(V \V1). Найближчим вершинам v∈V1 приписано  
відстані l(v) до кореня v0 , причому l(v0)=0. Введемо  
позначення: 1 *V* – множина таких вершин *v*′′∈(V \V1), що ∃ ребра виду  
e =(v, *v*′′), де v∈V1. Для всіх ребер e∈E2 знаходимо таке ребро e2=(*v*′,  
v2), що величина l(*v*′)+w(e2) найменша. Тоді v2 називається другою  
найближчою вершиною, а ребра e1, e2 утворюють зростаюче дерево  
для виділених найближчих вершин D2 ={e1, e2}.  
(s+1)-й крок індукції. Нехай у результаті s кроків виділено множину  
найближчих вершин Vs={v0, v1, …, vs} і відповідне їй зростаюче  
дерево Ds={e1, e2, …, es}... Для кожної вершини v∈Vs  
обчислена відстань l(v) від кореня v0 до v; *s V* – множина вершин  
v∈(V\Vs), для яких існують ребра вигляду e =(vr, v), де vr ∈Vs ,  
v∈(V\Vs). На кроці s+1 для кожної вершини vr∈Vs обчислюємо  
відстань до вершини vr : ( 1)( ) ( ) min ( , ) \*  
\*  
*L s v l v w v v r*  
*v V*  
*r r*  
∈ *s*  
+ = + , де min  
береться по всіх ребрах e=(vr, v∗), *v* ∈*V s* \*  
, після чого знаходимо min  
серед величин L(s+1)(vr). Нехай цей min досягнуто для вершин *vr*0і  
відповідної їй *v* ∈*V s* \* , що назвемо vs+1. Тоді вершину vs+1 називаємо  
(s+1)-ю НВ, одержуємо множину Vs+1 =Vs Υ *vs*+1 і зростаюче дерево  
Ds+1 =Ds Υ (*vr*0,*vs*+1). (s+1)-й крок завершується перевіркою: чи є  
чергова НВ vs+1 відзначеною вершиною, що повинна бути за умовою  
задачі зв'язано найкоротшим ланцюгом з вершиною v0 . Якщо так, то  
довжина шуканого ланцюга дорівнює l(vs+1)=l(*vr*0)+w(*vr*0, vs+1); при  
цьому шуканий ланцюг однозначно відновлюється з ребер  
зростаючого дерева Ds+1. У противному випадку випливає перехід до  
кроку s+2.

**Завдання варіанту №13**

**з1.pngA close up of a keyboard

Description automatically generatedз3.png**

**Розв'язання**

**1.**

**A square white text

Description automatically generated**

**l(v1) = 4 l(v16) = 13**

**l(v2) = 5 l(v17) = 14**

**l(v3) = 6 l(v18) = 16**

**l(v4) = 7 l(v19) = 16**

**l(v5) = 7 l(v20) = 16**

**l(v6) = 8 l(v21) = 16**

**l(v7) = 8 l(v22) = 17**

**l(v8) = 8 l(v23) = 17**

**l(v9) = 9 l(v24) = 18**

**l(v10) = 10 l(v25) = 18**

**l(v11) = 10 l(v26) = 18**

**l(v12) = 11 l(v27) = 22**

**l(v13) = 12 l(v28) = 26**

**l(v14) = 11 l(v\*) = 23**

**l(v15) = 11**

Найкоротша відстань від вершини v0 до v\* проходить через вершини { v1, v2, v8, v11, v12, v18, v24, v27} та ця відстань дорівнює 24.

2. Виділяємо цикл v1 v2 v3 v13 v11v10 v14 v12 v4 v5 v8 v9 v8 v7A picture containing snow

Description automatically generated

Ось ребра, які не входять до циклу:

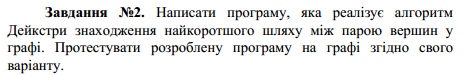
A close up of a device

Description automatically generated

Отже, укладений граф виглядає так:

A screenshot of a video game

Description automatically generated



A close up of a black keyboard

Description automatically generated

**#include<iostream>**

**using namespace std;**

**int n;**

**int i, j, q;**

**int dist[40];**

**bool visited[40];**

**int pred[40];**

**void createGraph(int c[40][40])**

**{**

**int g1, g2;**

**cout << "Enter the number of vertices: ";**

**cin >> n;**

**for (int i = 0; i < n; i++) {**

**for (int j = 0; j < n; j++)**

**{**

**c[i][j] = 0;**

**}**

**}**

**cout << "Enter size of (n\*m) : ";**

**cin >> g1 >> g2;**

**for (i = 0; i < n; i++) {**

**for (j = i + 1; j < n; j++)**

**{**

**if (j == i + 1 || j == i + g1) {**

**cout << "Enter the length from x" << i+1 << " to x" << j+1 << ": ";**

**cin >> c[i][j];**

**}**

**else {**

**c[i][j] = 0;**

**}**

**}**

**}**

**}**

**int minDistance()**

**{**

**int minimum = 10000, minDist;**

**for (int v = 0; v < n; v++)**

**if (visited[v] == false && dist[v] <= minimum)**

**{**

**minimum = dist[v];**

**minDist = v;**

**}**

**return minDist;**

**}**

**void printPath(int j)**

**{**

**if (pred[j] == -1)**

**return;**

**printPath(pred[j]);**

**cout << "X" << j+1 << " -> ";**

**}**

**void dijkstra(int c[40][40])**

**{**

**int start;**

**cout << "Enter the first node : ";**

**cin >> start;**

**for (int i = 0; i < n; i++)**

**{**

**pred[0] = -1;**

**dist[i] = 10000;**

**visited[i] = false;**

**}**

**dist[start-1] = 0;**

**for (int count = 0; count < n - 1; count++)**

**{**

**int u = minDistance();**

**visited[u] = true;**

**for (int v = 0; v < n; v++)**

**if (!visited[v] && c[u][v] &&**

**dist[u] + c[u][v] < dist[v])**

**{**

**pred[v] = u;**

**dist[v] = dist[u] + c[u][v];**

**}**

**}**

**cout << "The least way is: ";**

**cout << dist[29] << endl;**

**cout << "The way is: ";**

**cout << "X1 -> ";**

**printPath(29);**

**cout << "The end!)" << endl;**

**//}**

**}**

**int main()**

**{**

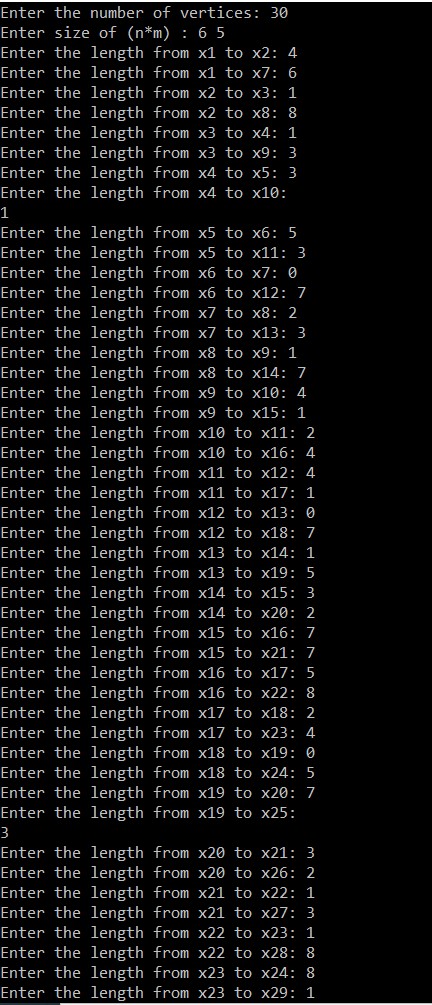
**int c[40][40];**

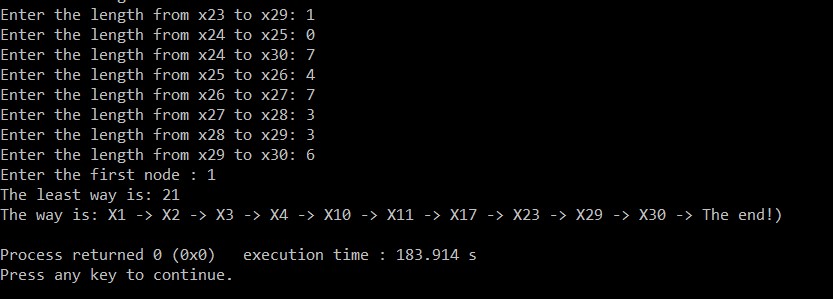
**createGraph(c);**

**dijkstra(c);**

**return 0;**

**}**

****

****

***Висновок:*** під час цієї лабораторної роботи я набула практичних вмінь та навичок з використання алгоритму Дейкстри.