**ЛАБОРАТОРНА РОБОТА №5**

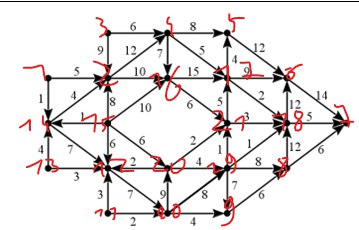
## Пошук найкоротшого шляху у графах

**Мета:**

* навчитися шукати найкоротший шлях у графах на основі методу Дейкстри та Флойда

**Задача 1,2.**

**Варіант 18, завдання/граф 18.**

1. **Створити програму, що реалізує метод алгоритму Дейкстри. Вихідний граф задається у вигляді матриці ваг, яка вводиться по рядках за допомогою консолі.**
2. **Знайти найкоротший шлях від кожної до кожної вершини за алгоритмом Флойда.**
3. **Програму реалізувати для графа згідно варіанту.**
4. **Програма має працювати для будь-якого графа.**

**2)Лістинг програми;**

**//Алгоритм Дейкстри**

#include <iostream>

#include <iomanip>

#include <Windows.h>

const int INF = 1e9;

using namespace std;

void Show\_result\_of\_dijkstra(int\* cur\_distence, int kil\_top);

int Find\_min\_distance(int\* cur\_distence, bool\* proven\_peaks, int kil\_top);

void dijkstra(int\*\* graph, int given\_vertex, int kil\_top);

int main()

{

setlocale(LC\_ALL, "ukr");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cout << "Введіть кількість вершин: ";

int kil\_top;

cin >> kil\_top;

cout << "Задайте матрицю: " << endl;

int\*\* graph = new int\* [kil\_top];

for (int i = 0; i < kil\_top; i++) {

graph[i] = new int[kil\_top];

for (int j = 0; j < kil\_top; j++) {

cin >> graph[i][j];

}

}

cout << endl;

int top\_that\_chose\_user;

cout << "Введіть номер вершини з якої хочете розпочати алгоритм: "; cin >> top\_that\_chose\_user;

top\_that\_chose\_user--;

dijkstra(graph, top\_that\_chose\_user, kil\_top);

return 0;

}

void dijkstra(int\*\* graph, int given\_vertex, int kil\_top)

{

int\* cur\_distence = new int[kil\_top];

bool\* proven\_peaks = new bool[kil\_top];

for (int i = 0; i < kil\_top; i++)

{

cur\_distence[i] = INF;

proven\_peaks[i] = false;

}

cur\_distence[given\_vertex] = 0;

for (int K = 0; K < kil\_top - 1; K++)

{

int u = Find\_min\_distance(cur\_distence, proven\_peaks, kil\_top);

proven\_peaks[u] = true;

for (int i = 0; i < kil\_top; i++)

{

if (!proven\_peaks[i] && graph[u][i] && cur\_distence[u] != INF && cur\_distence[u] + graph[u][i] < cur\_distence[i])

{

cur\_distence[i] = cur\_distence[u] + graph[u][i];

}

}

}

Show\_result\_of\_dijkstra(cur\_distence, kil\_top);

}

int Find\_min\_distance(int \* cur\_distence, bool \* proven\_peaks, int kil\_top)

{

int min = INF, cur\_min;

for (int i = 0; i < kil\_top; i++)

{

if (proven\_peaks[i] == false && cur\_distence[i] <= min)

{

min = cur\_distence[i];

cur\_min = i;

}

}

return cur\_min;

}

void Show\_result\_of\_dijkstra(int \* cur\_distence, int kil\_top)

{

cout << "Відстані до вершини: " << endl;

cout << endl;

for (int i = 0; i < kil\_top; i++)

{

cout << setw(4) << i + 1;

}

cout << endl;

for (int i = 0; i < kil\_top; i++)

{

if (cur\_distence[i] == INF)

{

cout << setw(4) << "INF";

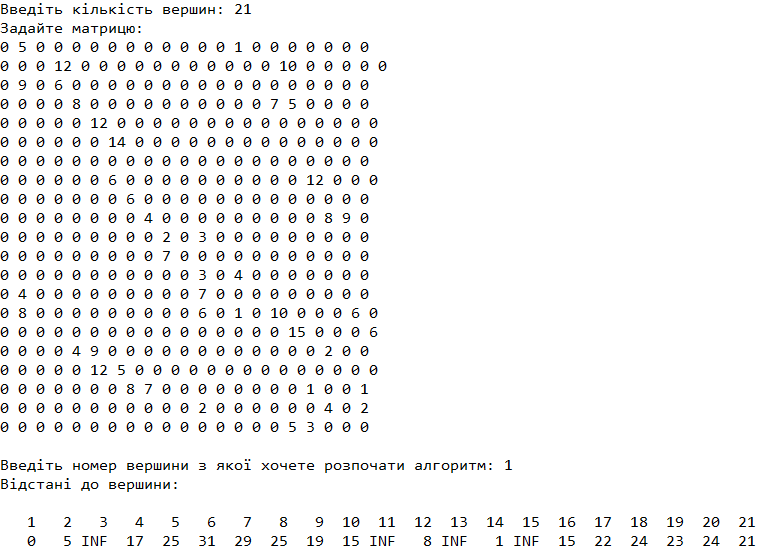
}

else cout << setw(4) << cur\_distence[i];

}

}

**5)Результати по Дейкстри**

****

**3)Лістинг програми;**

**//Алгоритм Флойда**

#include <iostream>

#include <iomanip>

#include <Windows.h>

#include <fstream>

using namespace std;

const int INF = 1e9;

const int MAXN = 100;

void floyd(int n, int weights[][MAXN]);

int main() {

setlocale(LC\_ALL, "ukr");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int n, weights[MAXN][MAXN];

cout << "Введіть кількість вершин графу: ";

cin >> n;

cout << "Введіть матрицю вагів графу (якщо між двома вершинами немає ребра, введіть -1):" << endl;

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

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

cin >> weights[i][j];

if (weights[i][j] == -1) {

weights[i][j] = INF;

}

}

}

floyd(n, weights);

}

void floyd(int n, int weights[][MAXN])

{

int dist[MAXN][MAXN];

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

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

dist[i][j] = weights[i][j];

}

}

for (int k = 0; k < n; k++) {

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

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

if (dist[i][k] < INF && dist[k][j] < INF) {

dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j]);

}

}

}

}

cout << "Матриця найкоротших шляхів:" << endl;

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

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

if (dist[i][j] == INF) {

cout << setw(4) << "INF";

}

else {

cout << setw(4) << dist[i][j];

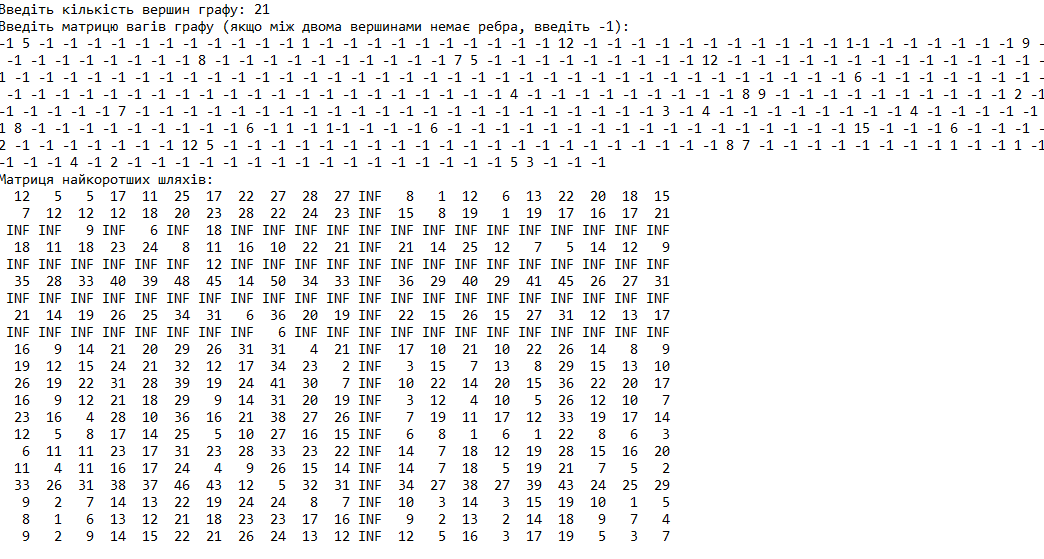
}

}

cout << endl;

}

}

**5)Результати по Флойда**

**Відповіді на запитання:**

1. **Дайте визначення шляху, маршрута, ланцюга, контура.**

**Шлях** — ланцюг, всі ребра якого орієнтовані в напряму руху від початкової до кінцевої вершини ланцюга. **Обходу** мостів відповідає послідовність ребер графа задачі, у якій два сусідніх ребра ‘мають загальну вершину, тобто маршрут. **Ланцюг** (в теорії графів; нім. Kreis, рос. цепь) — це послідовність виду Q = x0u1x1u2x2... xlul, де ребра u0, u1, ..., ul різні і ребро ui з'єднує (в будь-якому напрямі) вершини xi−1 та xi (i = 1, 2, ..., l) графу L = (X, U, P). **Контур** - це замкнений шлях у графі, який не має повторюваних вершин, за винятком початкової та кінцевої вершин. Іншими словами, **контур** - це цикл у графі, який не має повторюваних ребер, крім початкового та кі

1. **Який граф називається зваженим?**

**Зважений граф** — граф, кожному ребру якого поставлено у відповідність деяке значення (вага ребра). Зв'язність. Дві вершини в графі зв'язні, якщо існує (простий) ланцюг, що їх з'єднує. **Зв'язний граф** — граф, в якому всі вершини зв'язані.

1. **Як визначається довжина шляху графа?**

Довжина шляху — **кількість дуг шляху (або сума довжин його дуг, якщо останні задані)**. Так для шляху v1, v2, …, vn довжина дорівнює n-1. Доповнення графа — граф над тою самою множиною вершин, що і початковий, але вершини з'єднані ребрами тоді і тільки тоді, коли в початковому графі ребра немає. Дуга — орієнтоване ребро.

1. **Що називається правильною нумерацією вершин графа?**

**Правильна нумерація вершин графа** - це така нумерація, при якій кожній вершині графа призначений унікальний номер або мітка, що не перетинаються з номерами інших вершин. Для графів без вагованих ребер, зазвичай **нумерація вершин не має значення**, але для деяких алгоритмів пошуку шляхів в графах або для збереження графів в пам'яті комп'ютера **правильна нумерація може бути важливою**.

**Висновок:** навчився шукати найкоротший шлях у графах на основі методу Дейкстри та Флойда.