**Trần Hoàng Triển - MSSV:46.01.104.196**

**Cài đặt bài toán Tìm đường đi ngắn nhất (Sử dụng thuật toán Dijkstra)**

**Dữ liệu vào:**

- Dòng 1: Nhập 3 số N u v

‘N’: số đỉnh; u,v là đường đi ngắn nhất đi từ đỉnh u đến v

- N dòng tiếp theo: nhập ma trận trọng số của đồ thị

A screenshot of a computer

Description automatically generated with low confidence

**Dữ liêu ra :**

- Nếu có đường đi thì xuất:

* Tổng độ dài đường đi
* *Đường đi từ v ngược về u theo mẫu: v<- ... <-u*

- Nếu không có đường đi xuất “NO”

**A picture containing text, font, white, calligraphy

Description automatically generated**

**Miêu tả cách xử lý dữ liệu như sau:**

**1.** Sau khi nhập: 6 1 6

- `n` là số đỉnh của đồ thị.

- `s` là đỉnh xuất phát.

- `t` là đỉnh kết thúc.

- Ma trận **`Matrix`** chứa các trọng số của các cạnh trong đồ thị. Nếu có cạnh nối từ đỉnh `i` đến đỉnh `j` thì `Matrix[i][j]` được gán giá trị tương ứng, nếu không có cạnh nối thì `Matrix[i][j]` được gán giá trị `VOCUNG` (giá trị rất lớn).

**2.** Hàm `Dijkstra()` thực hiện thuật toán Dijkstra để tìm đường đi ngắn nhất từ đỉnh `s` đến đỉnh `t`. Các bước chính của thuật toán như sau:

**Khởi tạo:**

- Gán nhãn tạm thời (khoảng cách) cho tất cả các đỉnh trong đồ thị, ban đầu đặt giá trị của khoảng cách là vô cùng (VOCUNG) trừ đỉnh xuất phát s, giá trị khoảng cách đến chính nó là 0.

- Đánh dấu tất cả các đỉnh chưa được gán nhãn (chuaxet) là FALSE.

- Đặt truoc[t] = 0 (không có đỉnh trước t trong đường đi ban đầu).

**Bước lặp:**

- Chọn đỉnh u chưa được gán nhãn (chuaxet) sao cho khoảng cách d[u] là nhỏ nhất.

- Gán nhãn (chuaxet[u] = TRUE) cho đỉnh u đã chọn.

- Nếu đỉnh kết thúc t chưa được gán nhãn (chuaxet[t] = FALSE), thực hiện các bước sau:

- Cập nhật lại khoảng cách và đỉnh trước của các đỉnh chưa được gán nhãn theo công thức: d[v] = d[u] + Matrix[u][v], truoc[v] = u, nếu d[v] > d[u] + Matrix[u][v].

- Lặp lại bước lặp cho đến khi tất cả các đỉnh đều được gán nhãn.

**Kết quả:**

- Khi chuaxet[t] = TRUE, đường đi ngắn nhất đã được tìm thấy.

- In ra màn hình khoảng cách ngắn nhất từ đỉnh xuất phát đến đỉnh kết thúc (d[t]).

- In ra màn hình chuỗi các đỉnh trên đường đi ngắn nhất, bắt đầu từ đỉnh xuất phát và kết thúc tại đỉnh kết thúc. Chuỗi này được xây dựng bằng cách lặp qua các đỉnh trước đó ghi nhận trong mảng truớc[].

**3.** Hàm `Result()` in ra đường đi ngắn nhất từ đỉnh `s` đến đỉnh `t`. Các bước chính của hàm như sau:

- In ra khoảng cách ngắn nhất từ `s` đến `t`.

- In ra đỉnh `t`.

- Dùng mảng `truoc` để truy vết từ `t` về `s` và in ra

**Để thực thi chương trình trên, bạn có thể làm theo các bước sau:**

1. Mở trình biên dịch C++ (như Dev-C++, Visual Studio,).

2. Tạo một tệp tin mới và sao chép đoạn mã vào tệp tin đó.

3. Lưu tệp tin với phần mở rộng `.cpp` (ví dụ: `**Dijkstra.cpp**`).

4. Tiến hành chạy chương trình và nhập dữ liệu từ bàn phím ‘n’: số đỉnh;

s,t là đường đi ngắn nhất đi từ đỉnh s đến t .

- n dòng tiếp theo: nhập ma trận trọng số của đồ thị

5. Trình biên dịch sẽ thực thi chương trình và hiển thị kết quả.

**Source code:**

#include<iostream>

#include<fstream>

using namespace std;

#define MAX 50

#define TRUE 1

#define FALSE 0

#define VOCUNG 10000000

int n;//số đỉnh của đồ thị.

int s;//đỉnh đầu.

int t;//đỉnh cuối

char chon;

int truoc[MAX];//mảng đánh dấu đường đi.

int d[MAX];//mảng đánh dấu khoảng cách.

int Matrix[MAX][MAX];//ma trận trọng số

int chuaxet[MAX];//mảng đánh dấu đỉnh đã được gán nhãn.

void Init(void) {

cin >> n;

cin >> s >> t;//nhập đỉnh đầu và đỉnh cuối của đồ thị.

//nhập ma trận của đồ thị.

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n; j++) {

cin >> Matrix[i][j];

if (Matrix[i][j] == 0) Matrix[i][j] = VOCUNG;

}

}

}

void Result(void) {

cout << d[t] << endl;

cout << t << "<-";// in đỉnh cuối

int i = truoc[t];

while (i != s) {

cout << i << "<-";//in ra kết quả

i = truoc[i];

}

cout << s;//in đỉnh đầu

}

void Dijkstra(void) {

int u, minp;

//khởi tạo nhãn tạm thời cho các đỉnh.

for (int v = 1; v <= n; v++) {

d[v] = Matrix[s][v];

truoc[v] = s;

chuaxet[v] = FALSE;

}

truoc[s] = 0;

d[s] = 0;

chuaxet[s] = TRUE;

//bươc lặp

while (!chuaxet[t]) {

minp = VOCUNG;

//tìm đỉnh u sao cho d[u] là nhỏ nhất

for (int v = 1; v <= n; v++) {

if ((!chuaxet[v]) && (minp > d[v])) {

u = v;

minp = d[v];

}

}

chuaxet[u] = TRUE;// u la dinh co nhan tam thoi nho nhat

if (!chuaxet[t]) {

//gán nhãn lại cho các đỉnh.

for (int v = 1; v <= n; v++) {

if ((!chuaxet[v]) && (d[u] + Matrix[u][v] < d[v])) {

d[v] = d[u] + Matrix[u][v];

truoc[v] = u;

}

}

}

}

}

int main(void) {

Init();

Dijkstra();

Result();

return 0;

}