Báo cáo Phân tích thiết kế giải thuật

**Đề bài:** Thuật toán tìm đường đi ngắn nhất giữa mọi cặp đỉnh Floyd Warshall

**Họ và tên:** Vũ Tuấn Sơn

**Lớp:** Tin học – k50

1. **Nêu bài toán**

Với thuật toán Dijsktra hay Bellman đã giải quyết vấn đề tìm đường đi ngắn nhất từ 1 đỉnh tới n đỉnh còn lại trong đồ thị. Bài toán đặt ra ở đây là tìm đường đi ngắn nhất giữa mọi cặp đỉnh trong đồ thị (có thể có trọng số âm nhưng không có chu trình âm) sau một lần chạy thuật toán

1. **Mô tả chi tiết thuật toán**

* Cho đồ thị có hướng, có trọng số G = (V, E) với n đỉnh và m cạnh, không có chu trình âm.
* Gọi d(u, v) là khoảng cách từ u tới v
* Từ ma trận trọng số ta tính lại các d[u, v] thành độ dài đường đi ngắn nhất từ u tới v
* Với mọi đỉnh k của đồ thị được xét theo thứ tự từ 1 tới n , xét mọi cặp đỉnh u, v. Cực tiểu hóa theo công thức:

d[u, v] := min(d[u, v], d[u, k] + d[k, v])

* Công thức đó thể hiện rằng nếu đường đi từ u tới v đang có lại dài hơn đường đi từ u tới k cộng với đường đi từ k tới v thì ta bỏ đường đi từ u tới v hiện tại và coi đường đi từ u tới v là nối của u tới k và k tới v
* Giả mã

for k := 1 to n do

for u := 1 to n do

for v := 1 to n do

d[u, v] := min(d[u, v], d[u, k] + d[k, v]);

1. **Độ phức tạp của thuật toán**

Ở đây ta sử dụng 3 vòng for để duyệt nên độ phức tạp của thuật toán là O(n3)

1. **Ví dụ minh họa**

* *Ví dụ 1*

2

3

1

1

1

5

1

2

4

* Qua từng bước ta thu được kết quả như sau:

0 1 2 3 4 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0  0  1  2  3  4  5 | 1 | 0 | 2 | 0 | 0 |
| 1 | 0 | 5 | 2 | 0 | 4 |
| 0 | 5 | 0 | 1 | 0 | 1 |
| 2 | 2 | 1 | 0 | 3 | 0 |
| 0 | 0 | 0 | 3 | 0 | 1 |
| 0 | 4 | 1 | 0 | 1 | 0 |

d(0)[u, v]

0 1 2 3 4 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0  0  1  2  3  4  5 | 1 | 6 | 2 | 0 | 5 |
| 1 | 0 | 5 | 2 | 0 | 4 |
| 6 | 5 | 0 | 1 | 0 | 1 |
| 2 | 2 | 1 | 0 | 3 | 6 |
| 0 | 0 | 0 | 3 | 0 | 1 |
| 5 | 4 | 1 | 6 | 1 | 0 |

d(1)[u, v]

* Tiếp tục cho đến khi kết thúc ta thu được kết quả là đường đi ngắn nhất giữa mọi cặp đỉnh

0 1 2 3 4 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0  0  1  2  3  4  5 | 1 | 6 | 2 | 0 | 5 |
| 1 | 0 | 5 | 2 | 0 | 4 |
| 6 | 5 | 0 | 1 | 0 | 1 |
| 2 | 2 | 1 | 0 | 3 | 6 |
| 0 | 0 | 0 | 3 | 0 | 1 |
| 5 | 4 | 1 | 6 | 1 | 0 |

* *Ví dụ 2*
* Qua từng bước ta thu được kết quả như sau:

0 1 2 3 4 5 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0  0  1  2  3  4  5  6 | 3 | 6 | 0 | 0 | 0 | 0 |
| 3 | 0 | 2 | 4 | 0 | 0 | 0 |
| 6 | 2 | 0 | 1 | 4 | 2 | 0 |
| 0 | 4 | 1 | 0 | 2 | 0 | 4 |
| 0 | 0 | 4 | 2 | 0 | 2 | 1 |
| 0 | 0 | 2 | 0 | 2 | 0 | 1 |
| 0 | 0 | 0 | 4 | 1 | 1 | 0 |

d(0)[u, v]

0 1 2 3 4 5 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0  0  1  2  3  4  5  6 | 3 | 5 | 7 | 0 | 0 | 0 |
| 3 | 0 | 2 | 4 | 0 | 0 | 0 |
| 5 | 2 | 0 | 1 | 4 | 2 | 0 |
| 7 | 4 | 1 | 0 | 2 | 0 | 4 |
| 0 | 0 | 4 | 2 | 0 | 2 | 1 |
| 0 | 0 | 2 | 0 | 2 | 0 | 1 |
| 0 | 0 | 0 | 4 | 1 | 1 | 0 |

d(1)[u, v]

* Tiếp tục cho đến khi kết thúc ta thu được kết quả là đường đi ngắn nhất giữa mọi cặp đỉnh

0 1 2 3 4 5 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0  0  1  2  3  4  5  6 | 3 | 5 | 6 | 8 | 7 | 8 |
| 3 | 0 | 2 | 3 | 5 | 4 | 5 |
| 5 | 2 | 0 | 1 | 3 | 2 | 3 |
| 6 | 3 | 1 | 0 | 2 | 3 | 3 |
| 8 | 5 | 3 | 2 | 0 | 2 | 1 |
| 7 | 4 | 2 | 3 | 2 | 0 | 1 |
| 8 | 5 | 3 | 3 | 1 | 1 | 0 |

1. **Mã nguồn C++**

#include <iostream>

#include <conio.h>

#include <fstream>

using namespace std;

#define n 7

void file(int b[][n])

{

ifstream f;

f.open("input.txt", ios::out | ios::in);

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

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

f >> b[i][j];

f.close();

}

void floyd(int b[][n])

{

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

{

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

{

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

{

if ((b[i][k] \* b[k][j] != 0) && (i != j))

{

if (b[i][k] + b[k][j] < b[i][j] || b[i][j] == 0)

{

b[i][j] = b[i][k] + b[k][j];

}

}

}

}

}

cout << "Ket qua:" << endl;

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

{

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

{

cout << b[i][j] << " ";

}

cout << endl;

}

}

int main()

{

int b[n][n];

file(b);

floyd(b);

\_getch();

}