```
Guest@vietcodes (/):/$ ls
```


Thuật toán Floyd - Tìm đường đi ngắn nhất giữa mọi cặp đỉnh

Thuật toán Floyd - Tìm đường đi ngắn nhất giữa mọi cặp đỉnh

Contents

Floyd hay còn gọi là Floyd-Warshall là thuật toán để tìm đường đi ngắn nhất giữa mọi cặp đỉnh. Floyd hoạt động được trên đồ thị có hướng, có thể có trọng số âm, tuy nhiên không có chu trình âm. Ngoài ra, Floyd còn có thể được dùng để phát hiện chu trình âm.

Tìm đường đi ngắn nhất

Gọi C là ma trận kề của đồ thị và $D[u][v]$ là độ dài đường đi ngắn nhất từ u đến v.

Ban đầu ta khởi tạo D như sau:

- $D[u][u] = 0$ với mọi đỉnh u
- $D[u][v] = C[u][v]$ nếu có cạnh nối từ u đến v, ngược lại $D[u][v] = \infty$

Sau khi đã khởi tạo, ta tính D như sau:

```
for k:=1 to n do
    for i:=1 to n do
        for j:=1 to n do
            if D[i][j] > D[i][k] + D[j][k] then
                D[i][j] = D[i][k] + D[j][k];
```

Ý nghĩa của đoạn code trên rất đơn giản, lần lượt duyệt tất cả các cặp (k, i, j), nếu đường hiện tại từ i đến j dài hơn đường đi từ i qua k rồi từ k qua j thì ta cập nhật lại $D[i][j]$ bằng đường đi qua trung gian k này.

Lưu ý là ta phải duyệt k trước i và j thì thuật toán mới chạy đúng.

Truy vết đường đi

Để truy vết đường đi, ta thêm một mảng T với ý nghĩa $T[u][v]$ là đỉnh kề với đỉnh u trên đường đi ngắn nhất từ u đến v.

Khởi tạo T như sau:

- $T[u][v] = v$ nếu có cạnh nối từ u đến v

Sau đó ta kết hợp tính T trong quá trình tính D:

```
for k:=1 to n do
    for i:=1 to n do
        for j:=1 to n do
            if D[i][j] > D[i][k] + D[j][k] then
                D[i][j] = D[i][k] + D[j][k];
                T[i][j] = T[i][k];
```

Đoạn code sau trả về danh sách các đỉnh trên đường đi ngắn nhất từ u đến v:

```
vector<int> trace(int u, int v) {
    vector<int> path;
    do {
        path.push_back[u];
        u = T[u][v];
    } while (pack.back() != v);
    return path;
}
```

Cài đặt

Phần sau là code giải bài FLOYD (http://vn.spoj.com/problems/FLOYD/) sử dụng thuật toán Floyd.

```
#include <iostream>
#include <vector>
using namespace std;

#define rep(i,a,b) for (int i=a; i<=b; i++)
#define N 101
int C[N][N];
int T[N][N];

vector<int> trace(int u, int v) {
    vector<int> path;
    do {
        path.push_back(u);
        u = T[u][v];
    } while (path.back() != v);
    return path;
}

int main() {
    ios::sync_with_stdio(false); cin.tie(0);
    rep(i,1,N) rep(j,1,N) C[i][j] = 1e9;
    rep(i,1,N) C[i][i] = 0;

    int n, m, q;
    cin >> n >> m >> q;
    rep(_,1,m) {
        int u, v, w;
        cin >> u >> v >> w;
        C[u][v] = C[v][u] = w;
        T[u][v] = v;
        T[v][u] = u;
    }

    rep(k,1,n) rep(i,1,n) rep(j,1,n) {
        if (C[i][j] > C[i][k] + C[k][j]) {
            C[i][j] = C[i][k] + C[k][j];
            T[i][j] = T[i][k];
        }
    }

    rep(_,1,q) {
        int k,u,v;
        cin >> k >> u >> v;
        if (k) {
            auto path = trace(u,v);
            cout << path.size() << ' ';
            for (int u: path) cout << u << ' ';
        } else cout << C[u][v];
        cout << '\n';
    }
}
```

Phát hiện chu trình âm

Floyd có thể dùng để phát hiện chu trình âm. Sau khi chạy thuật toán, nếu $D[u][u] < 0$ thì có chu trình âm đi qua đỉnh u

```
Loading...
```