Bài tập thực hành

Đường đi ngắn nhất {1}

Bài 1.

#include <bits/stdc++.h>

using namespace std;

const int INF = 1e9;

void printState(int *n*, const *vector*<int>& *dist*, const *vector*<int>& *pre*, const *vector*<int>& *vis*) {

    cout << "dist: ";

    for (int i = 1; i <= n; ++i) {

        if (dist[i] >= INF) cout << i << ":INF ";

        else cout << i << ":" << dist[i] << " ";

    }

    cout << "\npre:  ";

    for (int i = 1; i <= n; ++i) {

        if (pre[i] == -1) cout << i << ":- ";

        else cout << i << ":" << pre[i] << " ";

    }

    cout << "\nvis:  ";

    for (int i = 1; i <= n; ++i) cout << vis[i] << " ";

    cout << "\n-------------------------\n";

}

int main() {

    int n, m, s, t;

    // Ví dụ: 6 7 1 4

    // followed by edges

    if (!(cin >> n >> m >> s >> t)) return 0;

    vector<vector<pair<int,int>>> adj(n+1);

    for (int i = 0; i < m; ++i) {

        int u,v,w; cin >> u >> v >> w;

        adj[u].push\_back({v,w});

        adj[v].push\_back({u,w});

    }

    vector<int> dist(n+1, INF), pre(n+1, -1), vis(n+1, 0);

    dist[s] = 0;

    cout << "Initial state:\n";

    printState(n, dist, pre, vis);

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

        int u = -1;

        int best = INF;

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

            if (!vis[i] && dist[i] < best) { best = dist[i]; u = i; }

        }

        if (u == -1) break;

        vis[u] = 1;

        cout << "Iteration " << iter << ": choose u = " << u << " (dist=" << dist[u] << ")\n";

        for (auto e : adj[u]) {

            int v = e.first, w = e.second;

            if (!vis[v] && dist[v] > dist[u] + w) {

                dist[v] = dist[u] + w;

                pre[v] = u;

            }

        }

        printState(n, dist, pre, vis);

    }

    if (dist[t] >= INF) {

        cout << "Khong co duong di\n";

        return 0;

    }

    cout << "Khoang cach ngan nhat: " << dist[t] << "\n";

    vector<int> path;

    for (int cur = t; cur != -1; cur = pre[cur]) path.push\_back(cur);

    reverse(path.begin(), path.end());

    cout << "Duong di: ";

    for (int x : path) cout << x << " ";

    cout << "\n";

    return 0;

}

Bài 2.

#include <bits/stdc++.h>

using namespace std;

const long long INF = (long long)1e15;

void dijkstra(int *n*, int *src*, const vector<vector<pair<int,int>>>& *adj*,

              vector<long long>& *dist*, vector<int>& *parent*) {

*dist*.assign(*n*+1, INF);

*parent*.assign(*n*+1, -1);

*dist*[*src*] = 0;

    priority\_queue<pair<long long,int>, vector<pair<long long,int>>, greater<pair<long long,int>>> pq;

    pq.push({0, *src*});

    while (!pq.empty()) {

        long long du = pq.top().first;

        int u = pq.top().second;

        pq.pop();

        if (du != *dist*[u]) continue;

        for (*size\_t* i=0;i<*adj*[u].size();++i) {

            int v = *adj*[u][i].first;

            int w = *adj*[u][i].second;

            if (*dist*[v] > *dist*[u] + w) {

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

*parent*[v] = u;

                pq.push({*dist*[v], v});

            }

        }

    }

}

int main() {

*ios*::sync\_with\_stdio(false);

    cin.tie(nullptr);

    int n, m;

    // Dòng đầu: n m s t x

    int s, t, x;

    if (!(cin >> n >> m >> s >> t >> x)) return 0;

    vector<vector<pair<int,int>>> adj(n+1);

    for (int i=0;i<m;i++){

        int u,v,w; cin >> u >> v >> w;

        // vô hướng:

        adj[u].push\_back({v,w});

        adj[v].push\_back({u,w});

    }

    vector<long long> dist\_s, dist\_x;

    vector<int> parent\_s, parent\_x;

    dijkstra(n, s, adj, dist\_s, parent\_s);

    dijkstra(n, x, adj, dist\_x, parent\_x);

    if (dist\_s[x] >= INF || dist\_x[t] >= INF) {

        cout << "Khong co duong di thoa man (qua dinh " << x << ")\n";

        return 0;

    }

    long long total = dist\_s[x] + dist\_x[t];

    cout << total << "\n";

    // Truy vết s -> x

    vector<int> path1;

    for (int cur = x; cur != -1; cur = parent\_s[cur]) path1.push\_back(cur);

    reverse(path1.begin(), path1.end()); // s ... x

    // Truy vết x -> t bằng parent\_x (lưu: parent\_x xây từ x)

    vector<int> path2;

    for (int cur = t; cur != -1; cur = parent\_x[cur]) path2.push\_back(cur);

    // path2: t ... x  -> cần đảo để thành x ... t

    reverse(path2.begin(), path2.end()); // x ... t

    // Ghép: path1 (s..x) + path2 (x..t) nhưng tránh lặp x

    vector<int> full = path1;

    for (*size\_t* i = 1; i < path2.size(); ++i) full.push\_back(path2[i]);

    for (int v : full) cout << v << " ";

    cout << "\n";

    return 0;

}

Bài 3.

#include <bits/stdc++.h>

using namespace std;

const long long INF = (long long)1e15;

vector<int> reconstructPath(int *u*, int *v*, const vector<vector<int>>& *next*) {

    vector<int> path;

    if (*next*[*u*][*v*] == -1) return path;

    int cur = *u*;

    path.push\_back(cur);

    while (cur != *v*) {

        cur = *next*[cur][*v*];

        path.push\_back(cur);

        if ((int)path.size() > 1000000) break; // bảo vệ tránh loop

    }

    return path;

}

int main() {

*ios*::sync\_with\_stdio(false);

    cin.tie(nullptr);

    int n, m;

    // Dòng đầu: n m

    if (!(cin >> n >> m)) return 0;

    vector<vector<long long>> dist(n+1, vector<long long>(n+1, INF));

    vector<vector<int>> nxt(n+1, vector<int>(n+1, -1));

    for (int i=1;i<=n;i++){

        dist[i][i] = 0;

        nxt[i][i] = i;

    }

    for (int i=0;i<m;i++){

        int u,v,w; cin >> u >> v >> w;

        // nếu có nhiều cạnh giữa u-v, giữ cạnh nhỏ nhất

        if (w < dist[u][v]) {

            dist[u][v] = w;

            dist[v][u] = w; // nếu vô hướng; nếu có hướng thì bỏ dòng này

            nxt[u][v] = v;

            nxt[v][u] = u;

        }

    }

    // Floyd-Warshall

    for (int k=1;k<=n;k++){

        for (int i=1;i<=n;i++){

            if (dist[i][k] >= INF) continue;

            for (int j=1;j<=n;j++){

                if (dist[k][j] >= INF) continue;

                if (dist[i][k] + dist[k][j] < dist[i][j]) {

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

                    nxt[i][j] = nxt[i][k];

                }

            }

        }

    }

    // In ma trận khoảng cách (n dòng, n cột)

    for (int i=1;i<=n;i++){

        for (int j=1;j<=n;j++){

            if (dist[i][j] >= INF/2) cout << "INF";

            else cout << dist[i][j];

            if (j < n) cout << " ";

        }

        cout << "\n";

    }

    return 0;

}

Bài 4.

#include <bits/stdc++.h>

using namespace std;

const long long INF = (long long)1e15;

int main(){

    ios::sync\_with\_stdio(false);

    cin.tie(nullptr);

    int n, m, s, t;

    if (!(cin >> n >> m >> s >> t)) return 0;

    vector<vector<pair<int,int>>> adj(n+1);

    for (int i=0;i<m;i++){

        int u,v,w; cin >> u >> v >> w;

        adj[u].push\_back({v,w});

        adj[v].push\_back({u,w}); // vô hướng; nếu có hướng thì bỏ dòng này

    }

    vector<long long> dist(n+1, INF);

    vector<int> parent(n+1, -1);

    dist[s] = 0;

    priority\_queue<pair<long long,int>, vector<pair<long long,int>>, greater<pair<long long,int>>> pq;

    pq.push({0, s});

    while (!pq.empty()){

        long long du = pq.top().first;

        int u = pq.top().second;

        pq.pop();

        if (du != dist[u]) continue;

        for (size\_t i=0;i<adj[u].size();++i){

            int v = adj[u][i].first;

            int w = adj[u][i].second;

            if (dist[v] > dist[u] + w) {

                dist[v] = dist[u] + w;

                parent[v] = u;

                pq.push({dist[v], v});

            }

        }

    }

    if (dist[t] >= INF) {

        cout << "Khong co duong di\n";

        return 0;

    }

    cout << dist[t] << "\n";

    vector<int> path;

    for (int cur = t; cur != -1; cur = parent[cur]) path.push\_back(cur);

    reverse(path.begin(), path.end());

    for (int v : path) cout << v << " ";

    cout << "\n";

    return 0;

}