#include <iostream>

#include <vector>

#include <queue>

#include <algorithm>

using namespace std;

const int MAX = 1e4 + 5;

const int INF = 1e9;

typedef pair<int, int> pii;

vector<pii> graph[MAX];

int dist[MAX];

bool visited[MAX];

void dijkstra(int s, bool increasing) {

fill(dist, dist + MAX, INF);

memset(visited, false, sizeof(visited));

priority\_queue<pii, vector<pii>, greater<pii>> pq;

pq.push({0, s});

dist[s] = 0;

while (!pq.empty()) {

int u = pq.top().second;

pq.pop();

if (visited[u]) continue;

visited[u] = true;

for (auto &edge : graph[u]) {

int v = edge.first;

int w = edge.second;

if ((increasing && w >= dist[u]) || (!increasing && w <= dist[u])) {

if (dist[v] > w) {

dist[v] = w;

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

}

}

}

}

}

int main() {

int n, m, s;

cin >> n >> m >> s;

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

int u, v, w;

cin >> u >> v >> w;

graph[u].push\_back({v, w});

graph[v].push\_back({u, w});

}

dijkstra(s, true);

dijkstra(s, false);

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

if (dist[i] == INF) cout << "Không có đường đi đến " << i << endl;

else cout << "Khoảng cách từ " << s << " đến " << i << " là: " << dist[i] << endl;

}

return 0;

}