#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 distIncrease[MAX], distDecrease[MAX];

bool visited[MAX];

void dijkstra(int s, int dist[], 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] > dist[u] + w) {

dist[v] = dist[u] + 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, distIncrease, true);

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

dijkstra(v, distDecrease, false);

}

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

if (distIncrease[v] != INF && distDecrease[v] != INF)

cout << "Đường đi bitonic ngắn nhất từ " << s << " đến " << v << " là: " << distIncrease[v] + distDecrease[v] << endl;

else

cout << "Không có đường đi bitonic từ " << s << " đến " << v << endl;

}

return 0;

}