#include <iostream>  
#include <vector>  
#include <queue>  
#include <climits>  
using namespace std;  
struct Edge {  
 int to;  
 int weight;  
};  
struct Node {  
 int distance;  
 vector<int> path;  
};  
struct TP {  
 int first;  
 int second;  
};  
struct Compare {  
 bool operator()(const TP& a, const TP& b) const {  
 return a.first > b.first;  
 }  
};  
void dijkstra(vector<vector<Edge>>& graph, int start) {  
 int n = graph.size();  
 vector<Node> nodes(n, {INT\_MAX, {}});  
 nodes[start].distance = 0;  
 nodes[start].path.push\_back(start);  
  
 priority\_queue<TP,vector<TP>,Compare> pq;  
 pq.push({0, start});  
  
 while (!pq.empty()) {  
 int curr\_dist = pq.top().first;  
 int curr\_node = pq.top().second;  
 pq.pop();  
  
 if (curr\_dist > nodes[curr\_node].distance) continue;  
  
 for (auto& edge : graph[curr\_node]) {  
 int next\_node = edge.to;  
 int weight = edge.weight;  
 int new\_dist = curr\_dist + weight;  
  
 if (new\_dist < nodes[next\_node].distance) {  
 nodes[next\_node].distance = new\_dist;  
 nodes[next\_node].path = nodes[curr\_node].path;  
 nodes[next\_node].path.push\_back(next\_node);  
 pq.push({new\_dist, next\_node});  
 }  
 }  
 }  
  
 // Output shortest paths  
 for (int i = 1; i < n; ++i) {  
 if (nodes[i].distance == INT\_MAX)  
 cout << start << "->" << i << ":-1" << endl;  
 else {  
 cout << start << "->" << i << ":" << nodes[i].distance << "----[";  
 for (int j = 0; j < nodes[i].path.size(); ++j) {  
 if (j != 0) cout << " ";  
 cout << nodes[i].path[j];  
 }  
 cout << "]" << endl;  
 }  
 }  
}  
  
int main() {  
 int N, M, S;  
 cin >> N >> M >> S;  
  
 vector<vector<Edge>> graph;  
 graph.resize(N);  
  
 for (int i = 0; i < M; ++i) {  
 int u, v, w;  
 cin >> u >> v >> w;  
 graph[u].push\_back({v, w});  
 }  
  
 dijkstra(graph, S);  
  
 return 0;  
}



