#include <bits/stdc++.h>  
using namespace std;  
  
void dijkstra(int source, vector<vector<pair<int,int>>> &graph, vector<int> &dist) {  
    int V = graph.size();  
    dist.assign(V, INT\_MAX);  
    dist[source] = 0;  
  
    priority\_queue<pair<int,int>, vector<pair<int,int>>, greater<pair<int,int>>> pq;  
    pq.push({0, source});  
  
    while (!pq.empty()) {  
        int u = pq.top().second;  
        int d = pq.top().first;  
        pq.pop();  
  
        if (d > dist[u]) continue;  
  
        for (auto &edge : graph[u]) {  
            int v = edge.first;  
            int w = edge.second;  
  
            if (dist[v] > dist[u] + w) {  
                dist[v] = dist[u] + w;  
                pq.push({dist[v], v});  
            }  
        }  
    }  
}  
  
int main() {  
    int V, E;  
    cout << "Enter number of intersections (vertices): ";  
    cin >> V;  
    cout << "Enter number of roads (edges): ";  
    cin >> E;  
  
    vector<vector<pair<int,int>>> graph;  
    graph.resize(V);    
  
    cout << "Enter edges (u v w):\n";  
    for (int i = 0; i < E; i++) {  
        int u, v, w;  
        cin >> u >> v >> w;  
        graph[u].push\_back({v, w});  
        graph[v].push\_back({u, w});   
    }  
  
    int source;  
    cout << "Enter ambulance start location (source): ";  
    cin >> source;  
  
    int H;  
    cout << "Enter number of hospitals: ";  
    cin >> H;  
    vector<int> hospitals(H);  
    cout << "Enter hospital nodes: ";  
    for (int i = 0; i < H; i++) {  
        cin >> hospitals[i];  
    }  
  
    vector<int> dist;  
    dijkstra(source, graph, dist);  
  
    int minTime = INT\_MAX, nearestHospital = -1;  
    for (int h : hospitals) {  
        if (dist[h] < minTime) {  
            minTime = dist[h];  
            nearestHospital = h;  
        }  
    }  
  
    if (nearestHospital == -1)  
        cout << "No hospital reachable.\n";  
    else  
        cout << "Nearest hospital is at node " << nearestHospital  
             << " with travel time " << minTime << " minutes.\n";  
  
    return 0;  
}

OUTPUT:

Enter number of intersections (vertices): 4  
Enter number of roads (edges): 4  
Enter edges (u v w):  
0 1 1  
1 2 2  
2 3 3  
0 3 10  
Enter ambulance start location (source): 0  
Enter number of hospitals: 2  
Enter hospital nodes: 2 3  
Nearest hospital is at node 2 with travel time 3 minutes.