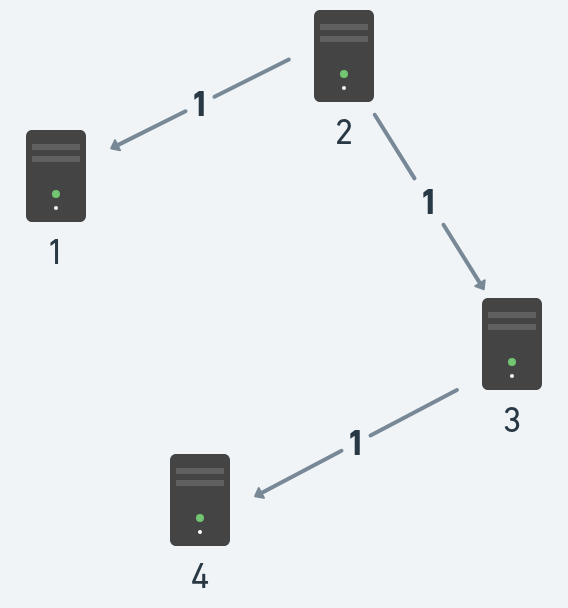
**Solve the questions in C/C++/ Go lang only No other Language**

Wavelabs’ Lab setup consists of a network of n nodes, labeled from 1 to n.As a Network Engineer, you are given times, a list of travel times as directed edges times[i] = (ui, vi, wi), where ui is the source node, vi is the target node, and wi is the time it takes for a signal to travel from source to target.

We will send a signal from a given node k. Return *the* ***minimum*** *time it takes for all the* n*nodes to receive the signal*. If it is impossible for all the n nodes to receive the signal, return -1.

**Example:**



**Input:** times = [[2,1,1],[2,3,1],[3,4,1]], n = 4, k = 2

**Output:** 2

**Constraints:**

* 1 <= k <= n <= 100
* 1 <= times.length <= 6000
* times[i].length == 3
* 1 <= ui, vi<= n
* ui != vi
* 0 <= wi<= 100
* All the pairs (ui, vi) are **unique**. (i.e., no multiple edges.)

**Solve the questions in C/C++/ Go lang only No other Language**

**#include <stdio.h>**

**#include <stdbool.h>**

**#include <limits.h>**

**#define MAX\_NODES 100**

**#define MAX\_EDGES 6000**

**// Structure to represent an edge**

**struct Edge {**

**int source, target, weight;**

**};**

**// Function to find the minimum distance vertex from the set of vertices**

**int minDistance(int dist[], bool visited[], int n) {**

**int min = INT\_MAX, min\_index;**

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

**if (!visited[v] && dist[v] <= min) {**

**min = dist[v];**

**min\_index = v;**

**}**

**}**

**return min\_index;**

**}**

**// Dijkstra's algorithm to find the shortest paths from source to all vertices**

**int dijkstra(int graph[MAX\_NODES][MAX\_NODES], int n, int k) {**

**int dist[MAX\_NODES];**

**bool visited[MAX\_NODES];**

**// Initialize all distances as INFINITE and visited[] as false**

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

**dist[i] = INT\_MAX;**

**visited[i] = false;**

**}**

**// Distance of source vertex from itself is always 0**

**dist[k] = 0;**

**// Find shortest path for all vertices**

**for (int count = 1; count <= n - 1; count++) {**

**int u = minDistance(dist, visited, n);**

**visited[u] = true;**

**// Update dist[] value of adjacent vertices of the picked vertex**

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

**if (!visited[v] && graph[u][v] && dist[u] != INT\_MAX && dist[u] + graph[u][v] < dist[v]) {**

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

**}**

**}**

**}**

**// Find the maximum distance to any node**

**int max\_time = 0;**

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

**if (dist[i] == INT\_MAX)**

**return -1; // If any node is not reachable, return -1**

**if (dist[i] > max\_time)**

**max\_time = dist[i];**

**}**

**return max\_time;**

**}**

**int main() {**

**int n, k, num\_edges;**

**printf("Enter the number of nodes: ");**

**scanf("%d", &n);**

**printf("Enter the source node: ");**

**scanf("%d", &k);**

**printf("Enter the number of edges: ");**

**scanf("%d", &num\_edges);**

**// Input the edges**

**int graph[MAX\_NODES][MAX\_NODES] = {0};**

**for (int i = 0; i < num\_edges; i++) {**

**int source, target, weight;**

**printf("Enter source, target, and weight for edge %d: ", i + 1);**

**scanf("%d %d %d", &source, &target, &weight);**

**graph[source][target] = weight;**

**}**

**// Find the minimum time it takes for all nodes to receive the signal**

**int min\_time = dijkstra(graph, n, k);**

**printf("Minimum time to reach all nodes: %d\n", min\_time);**

**return 0;**

**}**