**Dijkstra's shortest path**

#include <bits/stdc++.h>

using namespace std;

vector<int> dijkstra(vector<vector<int>>& graph, int V, int edges, int S) {

// Create adjacency list

unordered\_map<int, list<pair<int, int>>> adj;

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

int u = graph[i][0];

int v = graph[i][1];

int wt = graph[i][2];

adj[u].push\_back({v, wt});

adj[v].push\_back({u, wt});

}

vector<int> dist(V, INT\_MAX);

set<pair<int, int>> pq;

dist[S] = 0;

pq.insert({0, S});

while (!pq.empty()) {

auto it = \*(pq.begin());

int nodeDist = it.first;

int u = it.second;

pq.erase(it);

for (auto neighbor : adj[u]) {

int edgeWt = neighbor.second;

int adjNode = neighbor.first;

if (nodeDist + edgeWt < dist[adjNode]) {

if (dist[adjNode] != INT\_MAX) {

pq.erase({dist[adjNode], adjNode});

}

dist[adjNode] = nodeDist + edgeWt;

pq.insert({dist[adjNode], adjNode});

}

}

}

return dist;

}