#include <iostream>

#include <vector>

#include <queue>

#include <climits>

using namespace std;

struct Edge {

int to;

int weight;

};

struct QueueEntry {

int vertex; // đỉnh

int distance; // khoảng cách

bool operator>(const QueueEntry& other) const {

return distance > other.distance;

}

};

vector<int> dijkstra(int V, const vector<vector<Edge>>& graph, const vector<int>& sources) {

priority\_queue<QueueEntry, vector<QueueEntry>, greater<QueueEntry>> pq;

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

for (int src : sources) {

pq.push({src, 0});

distances[src] = 0;

}

while (!pq.empty()) {

QueueEntry current = pq.top();

pq.pop();

int u = current.vertex;

for (const Edge& edge : graph[u]) {

int v = edge.to;

int weight = edge.weight;

if (distances[v] > distances[u] + weight) {

distances[v] = distances[u] + weight;

pq.push({v, distances[v]});

}

}

}

return distances;

}

int main() {

int V = 5; // Số lượng đỉnh

vector<vector<Edge>> graph(V);

// Thêm cạnh vào đồ thị

graph[0].push\_back({1, 2});

graph[0].push\_back({3, 6});

graph[1].push\_back({2, 3});

graph[1].push\_back({3, 8});

graph[1].push\_back({4, 5});

graph[2].push\_back({4, 7});

graph[3].push\_back({4, 9});

vector<int> sources = {0}; // Đỉnh nguồn

vector<int> distances = dijkstra(V, graph, sources);

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

cout << "Distance from source to vertex " << i << " is " << distances[i] << endl;

}

return 0;

}