```
#include <iostream>
#include <limits.h>
#include <vector>

using namespace std;

#define INF INT_MAX

void dijkstra(vector<vector<int>>& graph, int start) {
    int num_nodes = graph.size();

    vector<int> distances(num_nodes, INF);
    distances[start] = 0;

    vector<bool> visited(num_nodes, false);

    for (int _ = 0; _ < num_nodes; ++_) {
        int min_distance = INF;
        int min_index = -1;

        for (int i = 0; i < num_nodes; ++i) {
            if (!visited[i] && distances[i] < min_distance) {
                min_distance = distances[i];
                min_index = i;
            }
        }

        visited[min_index] = true;

        for (int j = 0; j < num_nodes; ++j) {
            if (!visited[j] && graph[min_index][j] != 0 &&
distances[min_index] != INF &&
                distances[min_index] + graph[min_index][j] <
distances[j]) {
                distances[j] = distances[min_index] +
graph[min_index][j];
            }
        }
    }

    for (int i = 0; i < num_nodes; ++i) {
        cout << "从节点 " << start << " 到节点 " << i << " 的最短距离为 " << distances[i] << endl;
    }
}

int main() {
    vector<vector<int>> graph = {
        {0, 4, 0, 0, 0, 0, 0, 8, 0},
        {4, 0, 8, 0, 0, 0, 0, 11, 0},
        {0, 8, 0, 7, 0, 4, 0, 0, 2},
        {0, 0, 7, 0, 9, 14, 0, 0, 0},
        {0, 0, 0, 9, 0, 10, 0, 0, 0},
        {0, 0, 4, 14, 10, 0, 2, 0, 0},
        {0, 0, 0, 0, 0, 2, 0, 1, 6},
        {8, 11, 0, 0, 0, 0, 1, 0, 7},
        {0, 0, 2, 0, 0, 0, 6, 7, 0}
    };

    int start_node = 0;
    dijkstra(graph, start_node);

    return 0;
}
```

```
#include <iostream>
#include <vector>
#include <queue>
#include <limits.h>

using namespace std;

struct Edge {
    int target;
    int weight;

    Edge(int t, int w) : target(t), weight(w) {}
};

void dijkstra(vector<vector<Edge>>& graph, int start) {
    int num_nodes = graph.size();
    vector<int> distances(num_nodes, INT_MAX);
    distances[start] = 0;

    priority_queue<pair<int, int>, vector<pair<int, int>>,
greater<pair<int, int>>> minHeap;
    minHeap.push({0, start});

    while (!minHeap.empty()) {
        int current_distance = minHeap.top().first;
        int current_node = minHeap.top().second;
        minHeap.pop();

        for (const Edge& edge : graph[current_node]) {
            int neighbor = edge.target;
            int weight = edge.weight;

            if (current_distance + weight <
distances[neighbor]) {
                distances[neighbor] = current_distance +
weight;
                minHeap.push({distances[neighbor], neighbor});
            }
        }
    }

    for (int i = 0; i < num_nodes; ++i) {
        cout << "从节点 " << start << " 到节点 " << i << " 的最
短距离为 " << distances[i] << endl;
    }
}

int main() {
    vector<vector<Edge>> graph = {
        {{1, 4}, {7, 8}},
        {{0, 4}, {2, 8}, {7, 11}},
        {{1, 8}, {3, 7}, {5, 4}, {8, 2}},
        {{2, 7}, {4, 9}, {5, 14}},
        {{3, 9}, {5, 10}},
        {{2, 4}, {3, 14}, {4, 10}, {6, 2}},
        {{5, 2}, {8, 6}, {7, 1}},
        {{0, 8}, {1, 11}, {6, 1}, {8, 7}},
        {{2, 2}, {6, 6}, {7, 7}}
    };

    int start_node = 0;
    dijkstra(graph, start_node);

    return 0;
}
```