#include <stdio.h>

#include <limits.h>

#define V 9 // 假设有9个顶点

#define INF INT\_MAX // 表示无穷大

// 图的邻接矩阵表示

typedef struct {

int vertex[V][V];

int numVertices;

} Graph;

// 初始化图

void initializeGraph(Graph\* graph, int numVertices) {

graph->numVertices = numVertices;

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

for (int j = 0; j < numVertices; j++) {

graph->vertex[i][j] = (i == j) ? 0 : INF;

}

}

}

// 打印解决方案

void printSolution(int dist[], int numVertices) {

printf("Vertex \t\t Distance from Source\n");

for (int i = 0; i < numVertices; i++)

printf("%d \t\t %d\n", i, dist[i]);

}

// 找到未访问的最小距离顶点

int minDistance(int dist[], bool sptSet[]) {

int min = INF, min\_index;

for (int v = 0; v < V; v++)

if (sptSet[v] == false && dist[v] <= min)

min = dist[v], min\_index = v;

return min\_index;

}

// Dijkstra算法

void dijkstra(Graph\* graph, int src) {

int dist[V]; // 存储从源点到每个顶点的最短距离

bool sptSet[V]; // 标记顶点是否已包括在最短路径树中

// 初始化所有距离为无穷大，源点距离为0

for (int i = 0; i < graph->numVertices; i++)

dist[i] = INF, sptSet[i] = false;

dist[src] = 0; // 源点距离为0

// 找到最短路径的顶点

for (int count = 0; count < graph->numVertices - 1; count++) {

// 找到未处理的顶点中距离最小的顶点

int u = minDistance(dist, sptSet);

// 标记该顶点为已处理

sptSet[u] = true;

// 更新与该顶点相邻的顶点的距离

for (int v = 0; v < graph->numVertices; v++)

if (!sptSet[v] && graph->vertex[u][v] != INF && dist[u] != INF && dist[u] + graph->vertex[u][v] < dist[v])

dist[v] = dist[u] + graph->vertex[u][v];

}

// 打印结果

printSolution(dist, graph->numVertices);

}

int main() {

Graph graph;

initializeGraph(&graph, V);

// 填充图的邻接矩阵（这里只是示例，你可以根据需要修改）

graph.vertex[0][1] = 5;

graph.vertex[0][5] = 6;

graph.vertex[1][2] = 3;

graph.vertex[1][3] = INF; // 表示顶点1和顶点3之间没有直接连接

graph.vertex[1][4] = 1;

graph.vertex[2][3] = 2;

graph.vertex[2][4] = INF;

graph.vertex[3][4] = 4;

graph.vertex[4][5] = 2;

graph.vertex[4][6] = 1;

graph.vertex[5][6] = 3;

graph.vertex[5][7] = INF;

graph.vertex[6][7] = 1;

graph.vertex[6][8] = 6;

graph.vertex[7][8] = 0; // 这里假设顶点7和顶点8之间的距离为0，实际上这可能不是一个好的选择，因为它可能会破坏图的性质

}

// 从顶点0开始