#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

struct Edge {

int src, dest, weight;

};

struct Graph {

int V, E;

struct Edge\* edge;

};

struct Graph\* createGraph(int V, int E) {

struct Graph\* graph = (struct Graph\*)malloc(sizeof(struct Graph));

graph->V = V;

graph->E = E;

graph->edge = (struct Edge\*)malloc(E \* sizeof(struct Edge));

return graph;

}

void relax(int dist[], int parent[], struct Edge edge) {

int u = edge.src;

int v = edge.dest;

int weight = edge.weight;

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

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

parent[v] = u;

}}

void printPath(int parent[], int v) {

if (v < 0)

return;

printPath(parent, parent[v]);

printf("%d -> ", v);

}

void bellmanFord(struct Graph\* graph, int src) {

int V = graph->V;

int E = graph->E;

int dist[V];

int parent[V];

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

dist[i] = INT\_MAX;

parent[i] = -1;

}

dist[src] = 0;

for (int i = 1; i < V; i++) {

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

relax(dist, parent, graph->edge[j]);

}

}

printf("Shortest paths from source vertex %d:\n", src);

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

if (i != src) {

printf("To vertex %d, Distance: %d, Path: %d -> ", i, dist[i], src);

printPath(parent, i);

printf("\n");

} }}

int main() {

int V = 5;

int E = 8;

struct Graph\* graph = createGraph(V, E);

graph->edge[0].src = 0;

graph->edge[0].dest = 1;

graph->edge[0].weight = -1;

graph->edge[1].src = 0;

graph->edge[1].dest = 2;

graph->edge[1].weight = 4;

graph->edge[2].src = 1;

graph->edge[2].dest = 2;

graph->edge[2].weight = 3;

graph->edge[3].src = 1;

graph->edge[3].dest = 3;

graph->edge[3].weight = 2;

graph->edge[4].src = 1;

graph->edge[4].dest = 4;

graph->edge[4].weight = 2;

graph->edge[5].src = 3;

graph->edge[5].dest = 2;

graph->edge[5].weight = 5;

graph->edge[6].src = 3;

graph->edge[6].dest = 1;

graph->edge[6].weight = 1;

graph->edge[7].src = 4;

graph->edge[7].dest = 3;

graph->edge[7].weight = -3;

int src = 0;

bellmanFord(graph, src);

return 0;

}