#include <stdio.h>

#include <limits.h>

#define MAX\_NODES 101

#define MAX\_EDGES 6001

typedef struct {

int u, v, w;

} Edge;

void dijkstra(int n, int k, int times[][3], int num\_edges) {

int graph[MAX\_NODES][MAX\_NODES];

int distances[MAX\_NODES];

int visited[MAX\_NODES] = {0};

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

for (int j = 1; j <= n; j++) {

graph[i][j] = INT\_MAX;

}

}

for (int i = 0; i < num\_edges; i++) {

int u = times[i][0];

int v = times[i][1];

int w = times[i][2];

graph[u][v] = w;

}

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

distances[i] = INT\_MAX;

}

distances[k] = 0;

for (int count = 0; count < n-1; count++) {

int min\_distance = INT\_MAX;

int min\_index;

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

if (!visited[i] && distances[i] <= min\_distance) {

min\_distance = distances[i];

min\_index = i;

}

}

visited[min\_index] = 1;

for (int v = 1; v <= n; v++) {

if (!visited[v] && graph[min\_index][v] != INT\_MAX && distances[min\_index] != INT\_MAX

&& distances[min\_index] + graph[min\_index][v] < distances[v]) {

distances[v] = distances[min\_index] + graph[min\_index][v];

}

}

}

int max\_time = 0;

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

if (distances[i] == INT\_MAX) {

printf("-1");

return;

}

if (distances[i] > max\_time) {

max\_time = distances[i];

}

}

printf("%d", max\_time);

}

int main() {

int times[MAX\_EDGES][3] = {{2,1,1},{2,3,1},{3,4,1}};

int n = 4;

int k = 2;

int num\_edges = 3;

dijkstra(n, k, times, num\_edges);

return 0;

}