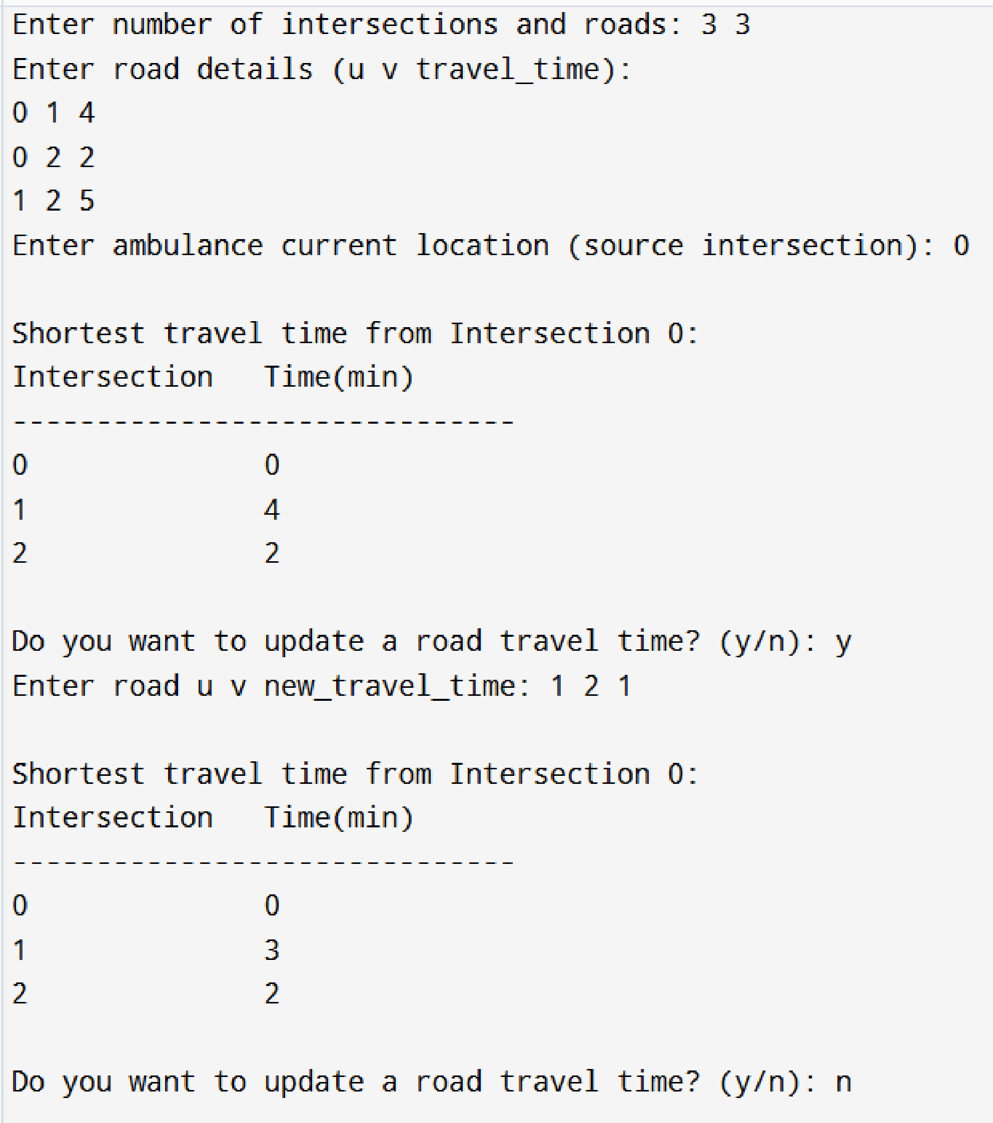
Smart Traffic Management Using Dijkstra’s Algorithm

OUTPUT:



```
Enter number of intersections and roads: 3 3
Enter road details (u v travel_time):
0 1 4
0 2 2
1 2 5
Enter ambulance current location (source intersection): 0

Shortest travel time from Intersection 0:
Intersection   Time(min)
------------------------------
0              0
1              4
2              2

Do you want to update a road travel time? (y/n): y
Enter road u v new_travel_time: 1 2 1

Shortest travel time from Intersection 0:
Intersection   Time(min)
------------------------------
0              0
1              3
2              2

Do you want to update a road travel time? (y/n): n
```

Introduction:
Smart cities require efficient traffic management, especially for **emergency vehicles** like ambulances. The goal is to **minimize travel time** from the ambulance’s current location to hospitals or critical destinations.

Graph Representation:

- **Intersections (nodes):** Each junction in the city.
- **Roads (edges):** Roads connecting intersections, with **weights representing travel time**.
- **Dynamic traffic:** Weights can change in real-time due to congestion.

Algorithm Used:
Dijkstra’s Algorithm is used to find the **shortest path** from a source node to all other nodes

in a weighted graph. It efficiently handles **non-negative edge weights** and works well with dynamic updates.

Steps:

1. Initialize all distances from the source as ∞, except the source itself (0).
2. Use a **priority queue** (min-heap) to repeatedly select the node with the smallest known distance.
3. For each neighbor, **update the distance** if a shorter path is found.
4. Repeat until all nodes are processed.
5. If traffic conditions change, **update edge weights** and rerun Dijkstra to get updated shortest paths.

Time Complexity:

- Using a min-heap: **O(E log V)**
- E = number of roads, V = number of intersections.

Applications:

- Emergency vehicle routing
- Smart traffic systems
- GPS navigation
- Dynamic route optimization