Design and Analysis of Algorithms

Project Report

Team Members:

Vinuthna Gummadi (16245992)

Sai Sravya Reddy Bokka (16240386)

Project 2 – Emergency Vehicle Dispatching System

Abstract:

This report presents an algorithm and implementation for Emergency Vehicle Dispatching System using Dijkstra's shortest path algorithm, that processes requests one by one. For each request, the algorithm will find the closest available emergency vehicle.

To implement this project, we also made necessary modifications to Dijkstra's shortest path algorithm.

Assumptions:

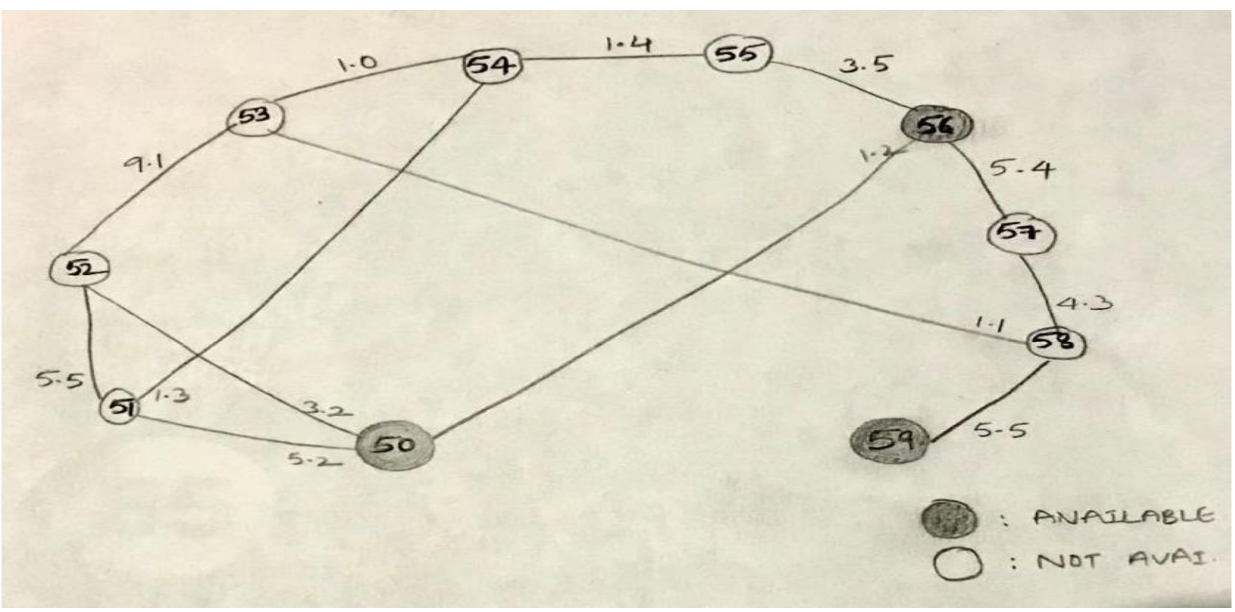
- There are three distinct types of emergency vehicles:

  1. Ambulance
  2. Fire Truck
  3. Police car

- When a vehicle is available in computed shortest path zip code it will reach the requested zip and will not go back to its origin.

Explanation:

Consider the following diagram, containing 10 nodes (Zip codes) and each of which are connected by Edges with the distances as their weights. Shaded nodes represent that a vehicle is available in that Zip code and vice versa.



Idea of the algorithm:

1. Construct a Min Heap with object (vertex, distance and path) as nodes.
2. Iterating through the nodes, extract minimum node, calculate the distance between the other linked nodes and update in the Min Heap.
3. Delete the minimum node and Re-heapify.
4. Continue the steps 2 and 3 until all nodes are visited and return the min node.

Below is the calculation for closest available Ambulance for 64153:

Note: Path and distance between two zip codes are retrieved from a dataset.

v	d[v]	p[v]
50	-	-
51	-	-
52	9.1	53
54	1.0	53
55	-	-
56	-	-
57	-	-
58	1.1	53
59	-	-

→

V	d[v]	p[v]
50	-	-
51	2.3	54
52	9.1	53
54	1.0	53
55	2.4	54
56	-	-
57	-	-
58	1.1	53
59	-	-

→

V	d[v]	p[v]
50	-	-
51	2.3	54
52	9.1	53
54	1.0	53
55	2.4	54
56	-	-
57	5.4	58
58	1.1	53
59	6.6	58

→

V	d[v]	p[v]
50	7.5	51
51	2.3	54
52	7.8	51
54	1.0	53
55	2.4	54
56	-	-
57	5.4	58
58	1.1	53
59	6.6	58

v	d[v]	p[v]
50	7.5	51
51	2.3	54
52	7.8	51
54	1.0	53
55	2.4	54
56	5.9	55
57	5.4	58
58	1.1	53
59	6.6	58

→

V	d[v]	p[v]
50	7.5	51
51	2.3	54
52	7.8	51
54	1.0	53
55	2.4	54
56	5.9	55
57	5.4	58
58	1.1	53
59	6.6	58

→

V	d[v]	p[v]
50	7.1	56
51	2.3	54
52	7.8	51
54	1.0	53
55	2.4	54
56	5.9	55
57	5.4	58
58	1.1	53
59	6.6	58

→

V	d[v]	p[v]
50	7.1	56
51	2.3	54
52	7.8	51
54	1.0	53
55	2.4	54
56	5.9	55
57	5.4	58
58	1.1	53
59	6.6	58

V	d[v]	p[v]
50	7.1	56
51	2.3	54
52	7.8	51
54	1.0	53
55	2.4	54
56	5.9	55
57	5.4	58
58	1.1	53
59	6.6	58

→

V	d[v]	p[v]
50	7.1	56
51	2.3	54
52	7.8	51
54	1.0	53
55	2.4	54
56	5.9	55
57	5.4	58
58	1.1	53
59	6.6	58

Available & Completed
Not Available & Completed
Min Node & To Be Considered

From the calculations above we can conclude that for an ambulance to reach 64153

from 64156 (56→55→54→53) the distance is 5.9

from 64159 (59→58→53) the distance is 6.6

from 64150 (50→56→55→54→53) the distance is 7.5

Out of all these, the minimum is 5.9 and thus algorithm chooses 56→55→54→53 path.

Efficiency of Algorithms:

Best Case: O(1) , if the requested vehicle is available in same zip code.

Average Case\Worst Case: Usually, for a Dijkstra's algorithm the efficiency is the order of n^2, since this algorithm uses Min heap concept, building heap takes O(nodes Log(nodes)) time(which does not have to be considered), inner loop which traces the adjacent edges and calculates the distance are executed in O(nodes+edges) times. After the inner loop, the node is deleted and re heapified which takes O(Log(nodes)) time. So overall time complexity is

O(nodes+edges) * O(Log(nodes))) = O(edges Log(nodes))

This algorithm can be enhanced to use Fibonacci Heap which takes O(1) for updating the heap nodes instead of Min heap which uses O(logn).

The same is verified in the program also. Refer Git Hub URL for implementation and validation.

GIT HUB:

https://github.com/VinuthnaGummadi/EmergencyVehicleDispatchSystem/wiki/Emergency-Vehicle-Dispatch-System---Design-and-Analysis-of-Algorithms-Project-Report

References:

https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

https://en.wikipedia.org/wiki/Binary_heap

https://angular.io/

https://projects.spring.io/spring-boot/

http://www.geeksforgeeks.org/