



University of Missouri-Kansas City

**Design and Analysis of Algorithms**  
(COMP-SCI 5592)

SPRING SEMESTER PROJECT

**EMERGENCY VEHICLE DISPATCHING  
SYSTEM**

by

**Lavanya Cheldhi (16242313)**

**Anand Sanjay Reddy Gundapureddy (16241776)**

**Harish Kolla (16231181)**

## Table of Contents

- Problem Statement
- Proposed Approach
- Assumptions
- Implementation Languages and Platforms
- About Dijkstra Algorithm
- BIG-O Complexity of Dijkstra Algorithm
- Illustrating Diagram
- Screenshots
- References

## Problem Statement

The Problem Statement here states that we must implement an algorithm such that it sends an Emergency Vehicle required to the zip code from which the request has occurred, and the requests must be processed one after the other.

## Proposed Approach

According to the given issue articulation, we should outline an effective constant framework. In this way, we have chosen to plan a framework where a few postal districts are taken as nodes(vertices) of a chart and the separation (in miles) between these hubs as the edges of the diagram. We have dispensed the three kinds of Emergency Vehicles i.e.; Ambulance, Fire Truck and Police to the postal divisions with the end goal that utilizing the Dijkstra's Algorithm we can locate the most brief way and separation between the hubs. We have additionally executed different codes so as to check the accessibility of the Emergency Vehicle required at the postal district where the vehicle is required, if the vehicle is available then it is designated to that specific postal district else the Dijkstra's Algorithm works and the postal division which is nearest and has the Emergency Vehicle required is recovered and assigned. This is a short depiction of what we executed to take care of the issue.

## Assumptions

To implement the Emergency Vehicle System, we have made several assumptions and they are:

- The vehicle once dispatched from a zip code will not return to it.
- The amount of time taken is dependent on distance only. So, the closest vehicle will always be the first to reach the destination, hence dispatched first.
- Vehicle Id's are not in order. There is no need that the vehicle with id no.1 will be dispatched before the vehicle with id no.2.
- A zip code can have any number of neighbors.
- The unit of distance is in miles. The time for the vehicle is in miles per minutes.
- There is no constraint on the number of vehicles to be present at the nodes. It can be of any value or none. Our chosen values are the number of fire truck 10, police 5 and ambulance 9.
- **The zip codes:** The base idea to tackle the problem is to find the closest nodes and check if the requested emergency vehicle is available at the selected node, if not then move to the 2nd closest node and so on until the requested vehicle id dispatched.

- **The Vehicle:** The number of the available emergency vehicle are limited, this might result in different nodes dispatching a different vehicle in consecutive requests. The vehicle once dispatched from a node is no more available at that node. Hence decrementing the number of emergency vehicles available.
- **The Idea:** The algorithm used to solve the problem is Dijkstra's algorithm. The base idea of Dijkstra is to find the shortest path to every other node present in the graph. Using this idea to dispatch the vehicles, a list of availability is assigned to every node (from here on the zip codes will be referred to as nodes). With this list being fixed the node closest to the requesting node with a vehicle available will dispatch it.

## Implementation Languages and Platforms

- Java
- Eclipse IDE

## About Dijkstra Algorithm

Dijkstra algorithm is used to find the shortest path between nodes in a graph, there are various real-life applications build on this Dijkstra algorithm for example in our day to day life we use navigation maps to reach our destination in less time this was build using Dijkstra algorithm. It was developed by a computer scientist named Edsger W Dijkstra in 1956. This algorithm is used to find the shortest path between nodes, where it selects a node as a starting point and selects the neighboring node which has the edge with low weight and continues until it reaches the specified destination node. This is the most efficient way to find shortest path between nodes.

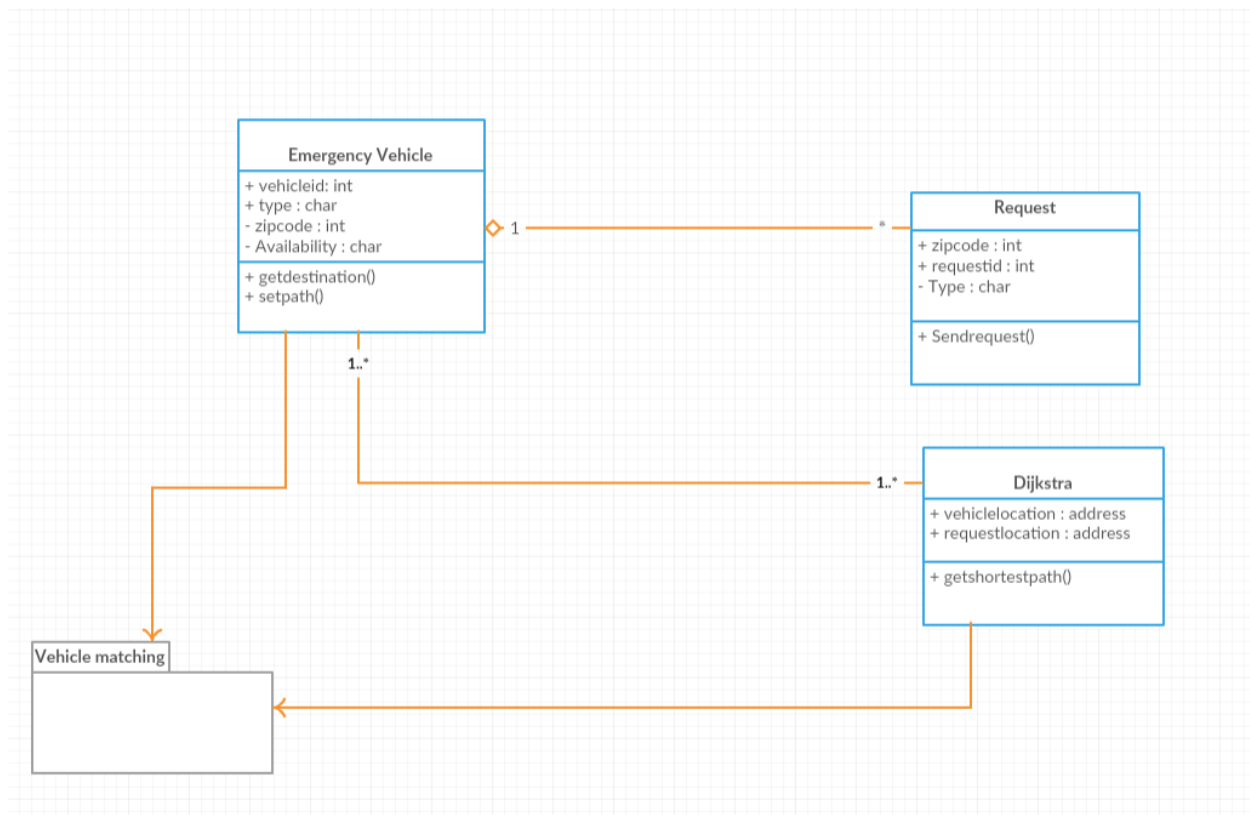
## BIG-O Complexity

- The efficiency of the Dijkstra's algorithm is  $O(V^2)$  where  $V$  is the number of edges.
- The efficiency of the algorithm with the added functions will also be close to  $O(V^2+n^2)$  as there are extra number of iterations. But the number of iterations  $n$  is also based on number of vertices  $V$  so it is  $\sim O(V^2)$
- It can be improved using priority queue, when implemented with binary heap and hash map. The big  $O$  is reduced to  $O(E \log V)$

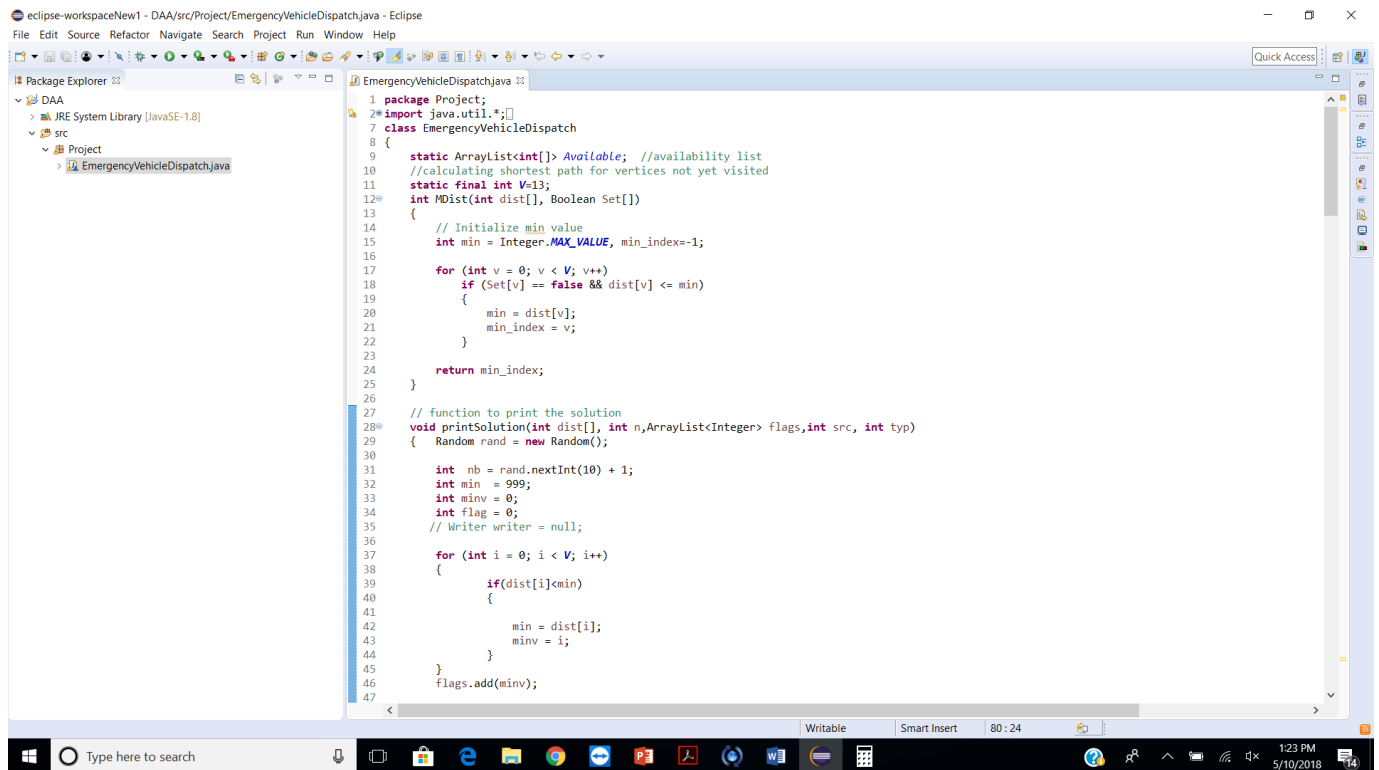
## Illustrating Diagram

### Class diagram:

This class chart portrays the fundamental structure of our view how to illuminate the crisis vehicle dispatching framework. We have three classes they are Emergency Vehicle, Request and Dijkstra we additionally have a vehicle coordinating bundle. We have two connections one total connection between ask for class and crisis vehicle class and second one is affiliation relationship which is between crisis vehicle class and Dijkstra class. Emergency vehicle class and Dijkstra class are having a basic line with vehicle coordinating bundle.



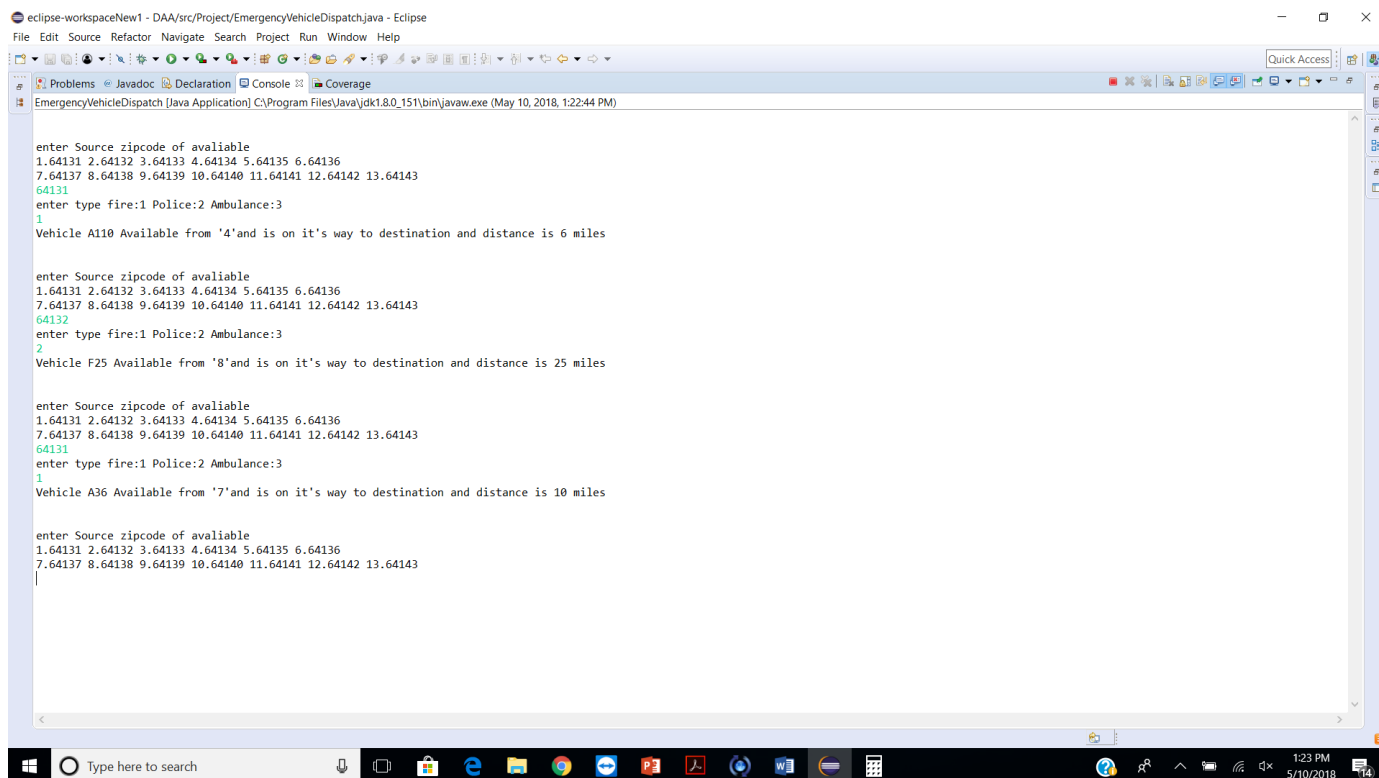
## Screenshots



```

1 package Project;
2 import java.util.*;
7 class EmergencyVehicleDispatch
8 {
9     static ArrayList<int> Available; //availability list
10    //calculating shortest path for vertices not yet visited
11    static final int V=13;
12    int MDist(int dist[], Boolean Set[])
13    {
14        // Initialize min value
15        int min = Integer.MAX_VALUE, min_index=-1;
16
17        for (int v = 0; v < V; v++)
18            if (Set[v] == false && dist[v] <= min)
19            {
20                min = dist[v];
21                min_index = v;
22            }
23
24        return min_index;
25    }
26
27    // function to print the solution
28    void printSolution(int dist[], int n, ArrayList<Integer> flags, int src, int typ)
29    {
30        Random rand = new Random();
31
32        int nb = rand.nextInt(10) + 1;
33        int min = 999;
34        int minv = 0;
35        int flag = 0;
36        // Writer writer = null;
37
38        for (int i = 0; i < V; i++)
39        {
40            if(dist[i]<min)
41            {
42                min = dist[i];
43                minv = i;
44            }
45        }
46        flags.add(minv);
47    }

```



```

enter Source zipcode of available
1.64131 2.64132 3.64133 4.64134 5.64135 6.64136
7.64137 8.64138 9.64139 10.64140 11.64141 12.64142 13.64143
64131
enter type fire:1 Police:2 Ambulance:3
1
Vehicle A110 Available from '4'and is on it's way to destination and distance is 6 miles

enter Source zipcode of available
1.64131 2.64132 3.64133 4.64134 5.64135 6.64136
7.64137 8.64138 9.64139 10.64140 11.64141 12.64142 13.64143
64132
enter type fire:1 Police:2 Ambulance:3
2
Vehicle F25 Available from '8'and is on it's way to destination and distance is 25 miles

enter Source zipcode of available
1.64131 2.64132 3.64133 4.64134 5.64135 6.64136
7.64137 8.64138 9.64139 10.64140 11.64141 12.64142 13.64143
64131
enter type fire:1 Police:2 Ambulance:3
1
Vehicle A36 Available from '7'and is on it's way to destination and distance is 10 miles

enter Source zipcode of available
1.64131 2.64132 3.64133 4.64134 5.64135 6.64136
7.64137 8.64138 9.64139 10.64140 11.64141 12.64142 13.64143

```

## References

- ✓ <http://www.geeksforgeeks.org/greedy-algorithms-set-6-dijkstras-shortest-path-algorithm/>
- ✓ <https://stackoverflow.com/questions/35147544/how-to-read-a-txt-file-into-a-2d-array-java>
- ✓ <https://stackoverflow.com/questions/1994255/how-to-write-console-output-to-a-txt-file>
- ✓ <https://www.youtube.com/watch?v=72BEuCPMgTo>
- ✓ <https://medium.com/@ssaurel/calculate-shortest-paths-in-java-by-implementing-dijkstras-algorithm-5c1db06b6541>
- ✓ [https://rosettacode.org/wiki/Dijkstra%27s\\_algorithm](https://rosettacode.org/wiki/Dijkstra%27s_algorithm)
- ✓ <http://www.vogella.com/tutorials/JavaIO/article.html>
- ✓ [https://www.tutorialspoint.com/java/java\\_files\\_io.htm](https://www.tutorialspoint.com/java/java_files_io.htm)
- ✓ <https://www.programcreek.com/2011/03/java-write-to-a-file-code-example/>

**GitHub Link:** <https://github.com/Lavanyacheldhi/DAA-project>