Project Report

on

Emergency Vehicle Dispatching System

Design and Analysis of Algorithms
CS5592
Spring 2018

Submitted By,

Md Mainul Islam Mamun
Student ID: 16212167

11 May 2018
University of Missouri- Kansas City

INDEX

Emergency Vehicle Dispatching System

1. Abstract

In this project I tried to design an emergency vehicle dispatching system in a virtual region in Kansas. I have used 30 vehicles (3 types) at 10 different zip codes' locations. I have designed a model that can assign a specific emergency vehicle based on a request as the vehicle can reach the spot at the earliest possible time with minimum cost. I have tested the model by using 6 random requests and found that the model woks well with the algorithm complexities $O(Rn^2)+O(n+mlogm)$ and $\Omega(Rn)+O(n+mlogm)$.

2. Introduction

I have designed a model for emergency vehicle dispatching system. There were three different types of Vehicles: Vehicle Type 1 is Ambulance, Vehicle Type 2 is Fire Truck, and Vehicle Type 3 is Police Car. There is an important assumptions that every request only needs one emergency vehicle at a time. I have generated 30 vehicles ID by using 10 Zip codes randomly (Table 1). After that I have marked out the neighboring nodes (Zip codes) and calculated the distances among the nodes only who are the neighboring nodes (Table 3). Moreover, I have generated a list of requests (in our case the total requests are 6) (Table 03) for serving with any specific emergency vehicle. I have coded a model by using java (EmergencyAssigning.java) that calls another algorithm (Dijkstra's Algorithm) in order to run our model properly to find out the exact vehicle from a suitable location that takes the minimum time with the minimum cost to reach the spot of the requester in order to serve as soon as possible.

3. Background

We know that Dijkstra's Algorithm can be used to find the single source shortest path. There are other algorithms those could also be used here in this Project, e.g. Prim's Algorithm for Minimum Spanning Tree, A*, Depth First Search etc. But in my Project I have used the Dijkstra's Algorithm to find the single source shortest path with minimum cost.

3.1 Dijkstra's Algorithm

E.W. Dijkstra solved the problem to find the shortest path from a source in a graph to a destination which is known as Dijkstra's Algorithm. It turns out that one can find the shortest paths from a given point to *all* points in a graph in the same time.

Dijkstra's algorithm uses two sets of vertices: S and V- S where S→ the set of vertices whose shortest paths from the source have already been determined and V- S→ the remaining vertices. Other parameters are d (array of best estimates of shortest path to each vertex) and pi (an array of predecessors for each vertex)

The working principle of the Algorithm is mentioned below:

1. Initialize **d** and **pi**
2. Set **S** to empty
3. While there are still vertices in **V-S**

i.Sort the vertices in **V-S** according to the current best estimate of their distance from the source
ii.Add **u**, the closest vertex in **V-S**, to **S**
iii.Relax all the vertices still in **V-S** connected to **u**

What the Relaxation does is to process updates the costs of all the vertices, **v**, connected to a vertex, **u**, if we could improve the best estimate of the shortest path to **v** by including **(u,v)** in the path to **v**.

[References (https://www.cs.auckland.ac.nz/software/AlgAnim/dijkstra.html)]

4. Model and Discussion

4.1 Virtual Geographical Locations

I have used 10 random Zip codes to represent 10 different locations which is considered as a connected graph. The Zip codes/ different locations and their connectivities along with their weights are shown in Fig. 01. We can consider it as a connected Graph, G.

4.2 Adjacency Matrix

After that I have created an Adjacency matrix from that graph, G in order to find the adjacency with the respective weights which is shown in Fig. 03.

4.3 Number of Vehicles

I have created a list of three different types of 30 emergency vehicles with different IDs at the 10 locations/ Zip codes (Table 1).

4.4 Distance Table

After that I have created a distance table (Table 03) (the distances between Zip codes that are neighboring to each other).

4.5 Number of Requests

Finally, I have generated 6 different requests randomly to verify our model (Table 02).

4.6 Our Model Algorithm

The idea behind our model is as follows:

I. Read the two files and stores as variables (Now we have vehicle and request information)

II. Covert the distance table to Adjacency matrix

III. For each request do the following:

a. Find the single source shortest path from that request

b. Now we will have list of shortest distance to each node from the request node

c. Find the node with minimum cost

d. Check the required vehicle type is available in that Zip code or not

e. If available show and stop the rest requests

else go for the next minimum and continue

Algorithm:

```
for (int k = 0; k < NumberofRequests; k++)
    int [] distances = SP.dijkstra (Graph, RequestLocation [k] % 10;
    int MinimumDistance = Integer.MAX_VALUE;
    int TempAssign = -1;
    for (int i = 0; i < distance.length; i++)
        if (distance [i] < MinimumDistance)
            for (int j = 0; j < NumberofVehicles; j++)
                if (((VehicleLocation [j] % 10) == i) && (VehicleAssigned [j] == false)
                    && RequestType [k] == VehicleType [j]))
                      TempAssign = j;
                      MinimumDistance = distance [i];
                      break;
                end if
            end for
        end if
    end for
end for
```

4.7 Model Outcomes

The Table 04 shows finally the outcomes of our model and we found that our model performs better with the complexities $O(Rn^2)+O(n+mlogm)$ and $\Omega(Rn)+O(n+mlogm)$ in the worst and best cases repectively, where R→ Number of Requests, n→ Number of Nodes and m→ number of edges.

4.8 Manual Verification

I did check our model manually with the Dijkstra's Algorithm with one request from the Node B (Zip code 66200) and found the single source shortest path which matches exactly with the model output. (Fig. 02). Table 5 shows the each steps as it works.

5. Results and Conclusions

We have observed during the execution of the 6 requests that our model is performing better to find the shortest path for the specific vehicle to reach the spot with minimum cost. Our model complexities are $O(Rn^2) + O(n+mlogm)$ and $\Omega(Rn)$. We got the assigned specific emergency vehicles IDs that can be assigned to the requester for the best performance regarding time and cost which has been shown in Table 04 (the outcomes of our designed model).

6. GitHub Links

7. Acknowledgements

Professor Mohammad Amin H Kuhail, TA- Aishvwarya Natarajan Lyer and Kaushik Ayinala.

8. References

(https://www.cs.auckland.ac.nz/software/AlgAnim/dijkstra.html)

9. Figures and Tables

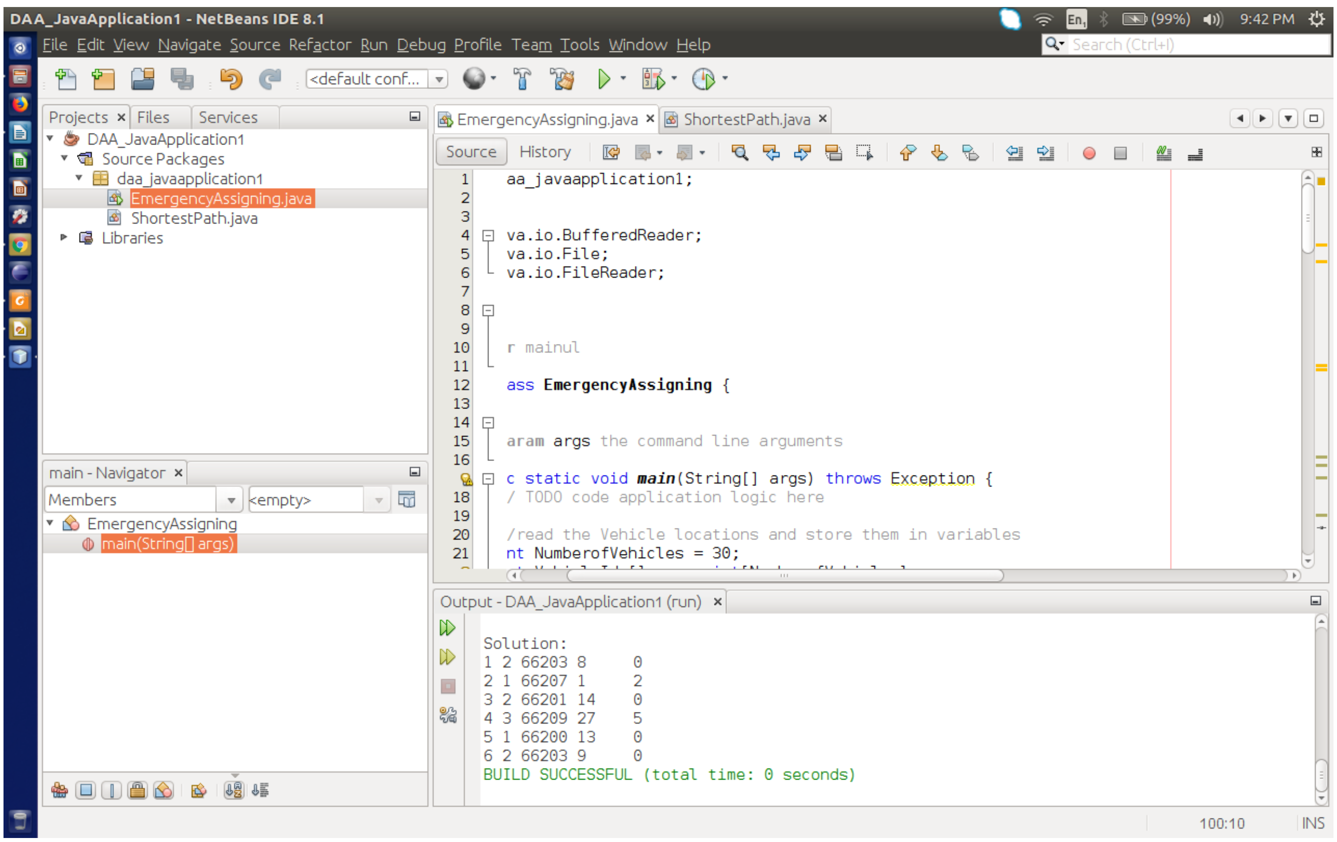

Figure: Screen shot of the Output

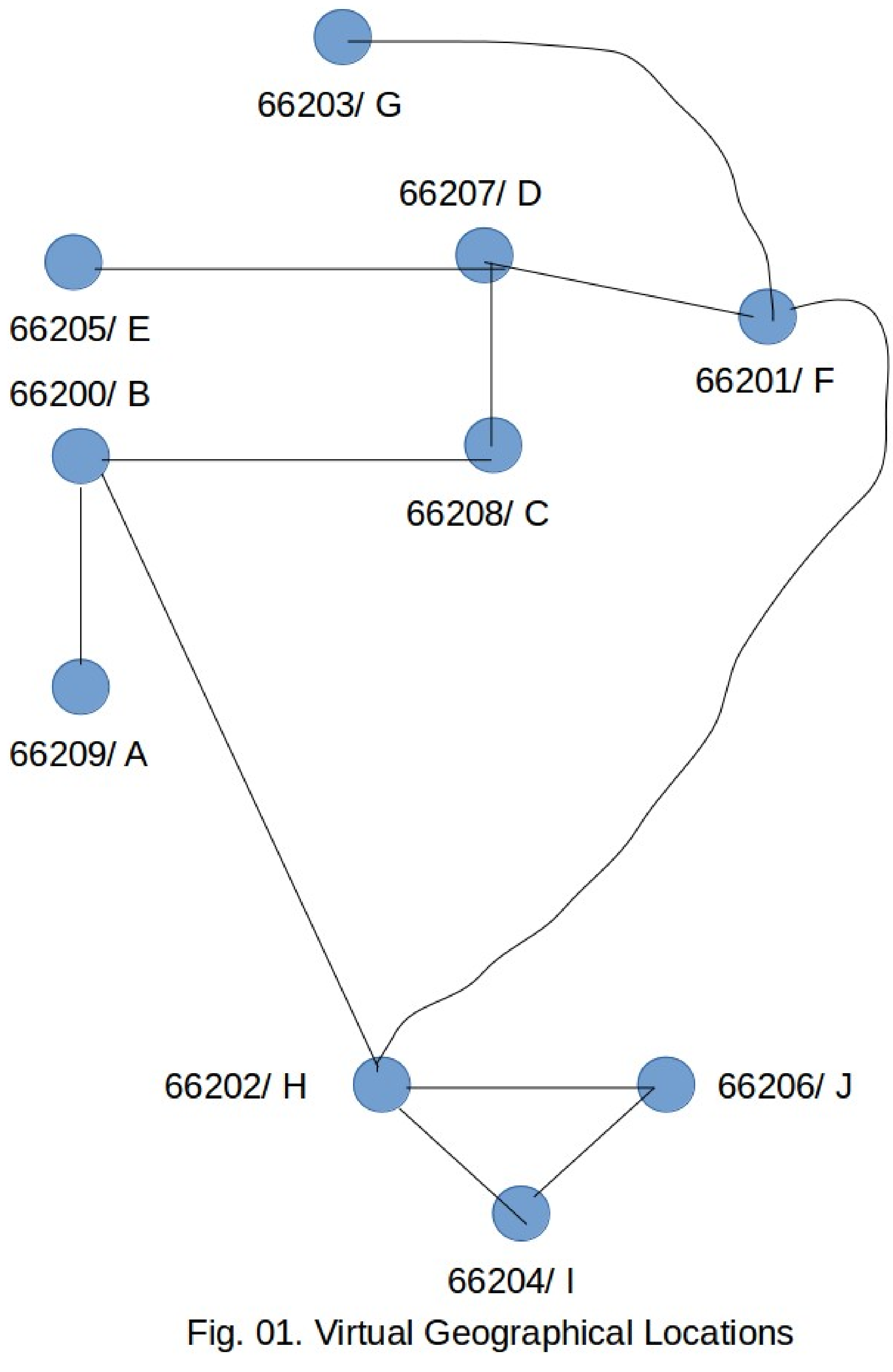

Fig. 01. Virtual Geographical Locations

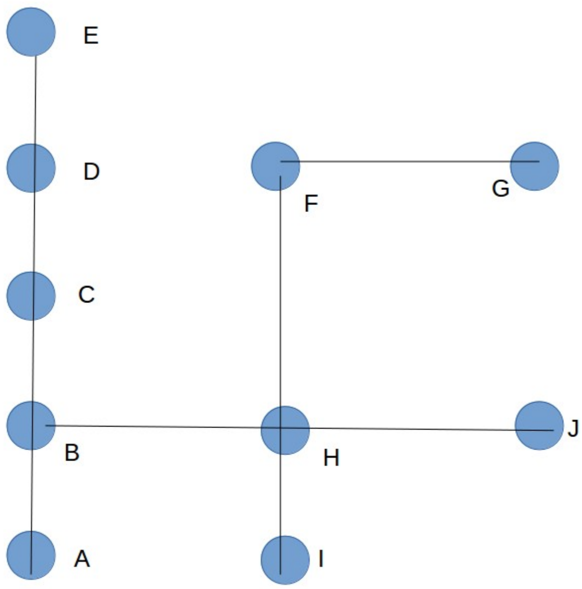

Fig. 02. Single- source shortest path from Node B

{0,0,2,0,0,0,0,0,4,1},
{0,0,5,5,0,0,0,2,0,0},
{2,5,0,0,2,0,3,0,0,0},
{0,5,0,0,0,0,0,0,0,0},
{0,0,2,0,0,0,2,0,0,0},
{0,0,0,0,0,0,0,3,0,0},
{0,0,3,0,2,0,0,0,0,0},
{0,2,0,0,0,3,0,0,2,0},
{4,0,0,0,0,0,0,2,0,0},
{1,0,0,0,0,0,0,0,0,0}

Fig. 03. Adjacency Matrix of my virtual Geographical Locations Graph

Table: 01 Number of Emergency Vehicles at 10 different locations/ Zip codes

IDs	Vehicle Type	Zip Codes
1	1	66201
2	1	66202
3	2	66206
4	3	66203
5	1	66209
6	3	66208
7	1	66209
8	2	66203
9	2	66203
10	2	66202
11	3	66206
12	1	66202
13	1	66200
14	2	66201
15	3	66206
16	2	66207
17	2	66206
18	2	66209
19	3	66201
20	3	66206
21	2	66201
22	2	66201
23	1	66202
24	1	66206
25	2	66205
26	2	66208
27	3	66204

28	1	66201
29	3	66205
30	2	66203

Table: 02 Request Table

IDs	Vehicle Type	Zip codes
1	2	66203
2	1	66207
3	2	66201
4	3	66209
5	1	66200
6	2	66203

Table: 03 Distance Table

Zip Code 1	Zip Code 2	Distance
66208	66207	2
66208	66200	4
66200	66209	1
66200	66202	2
66202	66204	2
66202	66206	3
66204	66206	2
66202	66201	5
66205	66207	3
66207	66201	2
66201	66203	5

Table: 04 Outcome of our Model

IDs	Vehicle Type	Zip code	Vehicle ID	Distance
1	2	66203	8	0
2	1	66207	1	2
3	2	66201	14	0
4	3	66209	27	5
5	1	66200	13	0
6	2	66203	9	0

Table: 05 Manual Verification of our Model using Dijkstra's Algorithm, step by step (From Source Node, B (Zip code 66202) to all other Nodes

B	A	C	D	E	F	G	H	I	J
B	1, B	4, B	∞	∞	∞	∞	2, B	∞	∞
BA	-	4, B	∞	∞	∞	∞	2, B	∞	∞
BAH	-	4, B	∞	∞	7, H	∞	-	4, H	5, H
BAHC	-	-	6, C	∞	7, H	∞	-	4, H	5, H
BAHCI	-	-	6, C	∞	7, H	∞	-	-	5, H
BAHCIJ	-	-	6, C	∞	7, H	∞	-	-	-
BAHCIJD	-	-	-	9, D	7, H	∞	-	-	-
BAHCIJDF	-	-	-	9, D	-	12, F	-	-	-
BAHCIJDFE	-	-	-	-	-	12, F	-	-	-
BAHCIJDFEG	-	-	-	-	-	-	-	-	-