Design and Analysis of Algorithm

PROJECT

Emergency Vehicle Dispatching System

Submitted By

Abhiram Reddy Nalla/16230799

Arun Kumar Reddy Yeddula/16233006

Avinash Sankarasetty/16233012

Sravani Konujula/16230172

Saidu Dosapati/12533623

Dijkstra's Algorithm:

Dijkstra's algorithm is a way to find out the single-source shortest path problem in graph theory.

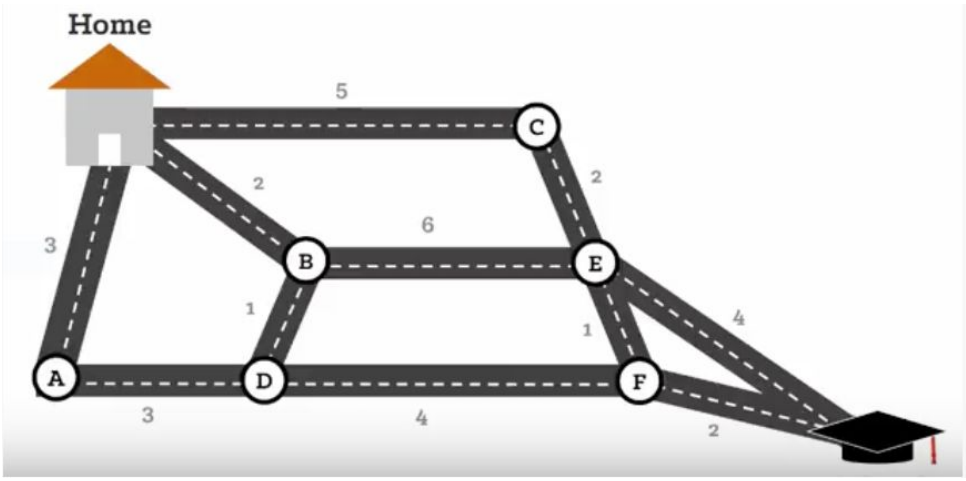
- It can be either directed or undirected.
- It contains all the weighted edges.
- It is a connected graph.

For instance

Find the distance between home and school using Dijkstra's Algorithm?

Solution:-

1) Create a set sptSet (shortest path tree set) that keeps track of vertices included in shortest path tree, i.e., whose minimum distance from source is calculated and finalized. Initially, this set is empty.
2) Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign distance value as 0 for the source vertex so that it is picked first.
3) While sptSet doesn't include all vertices
a) Pick a vertex u which is not there in sptSet and has minimum distance value.
b) Include u to sptSet.
c) Update distance value of all adjacent vertices of u. To update the distance values, iterate through all adjacent vertices. For every adjacent vertex v, if sum of distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.



From the above figure shortest path is

Home -> B -> D -> F ->School

Distance = 2+1+4+2 => 9

Question:- Emergency Vehicle Dispatching System

Abstract:

The main aim of this project is using Dijkstra's Algorithm to find the shortest distance between nodes and respond to requests and allotting emergency vehicles for which they requested.

There are 3 different types of emergency vehicles:

1 (Ambulance),

2 (Fire Truck),

3 (Police car).

Three attributes are to be considered

Zip-codes which are considered as nodes.

Distance – distance between 2 Nodes.

Vehicle type- tells which type it is.

Construction of our graph:

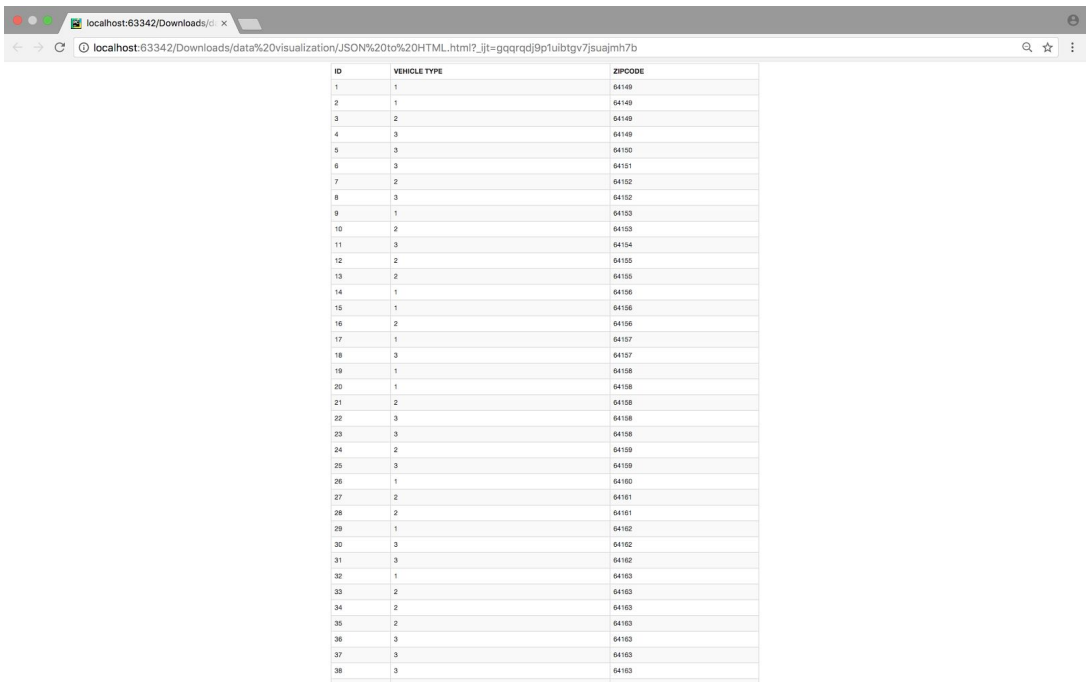
To solve this, we implemented in python using the “json” data

a) For the Vehicles:

json data shows nodes with their vehicles .


```
{
    "vehicles": [{
            "id": 1,
            "type": 1,
            "zipcode": 64149
        },
        {
            "id": 2,
            "type": 1,
            "zipcode": 64149
        }
    ]
}
```


We have 43 vehicles distributed among 19 nodes

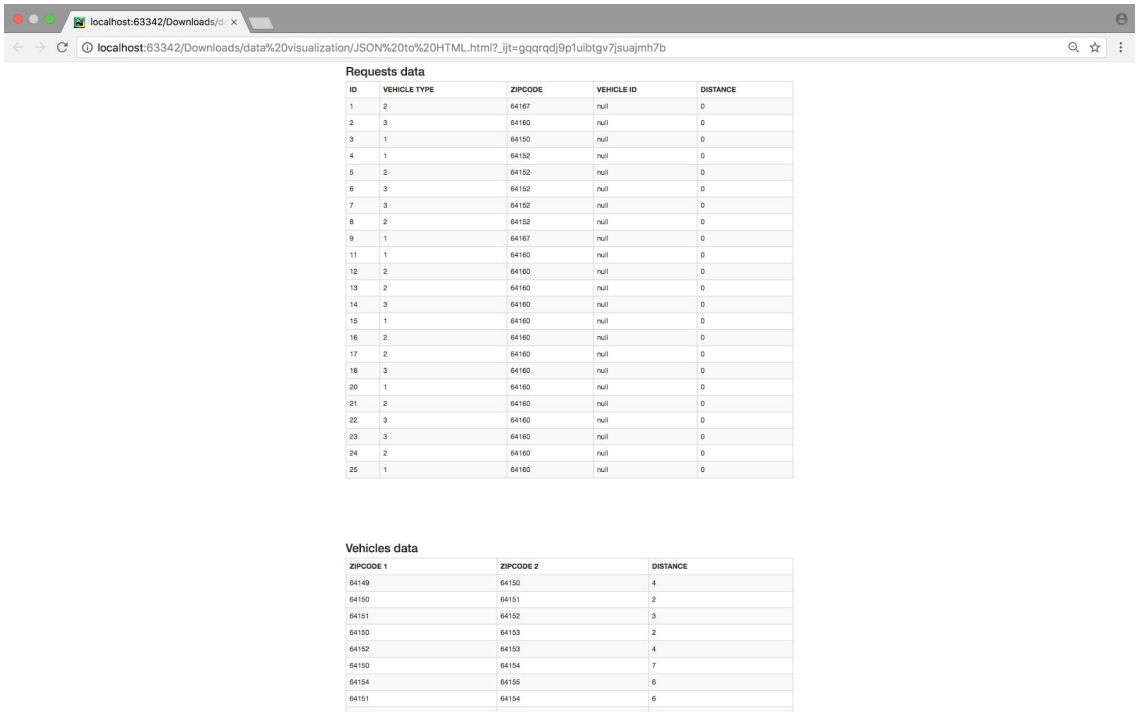


ID	VEHICLE TYPE	ZIPCODE
1	1	64149
2	1	64149
3	2	64149
4	3	64149
5	3	64150
6	3	64151
7	2	64152
8	3	64152
9	1	64153
10	2	64153
11	3	64154
12	2	64155
13	2	64155
14	1	64156
15	1	64156
16	2	64156
17	1	64157
18	3	64157
19	1	64158
20	1	64158
21	2	64158
22	3	64158
23	3	64158
24	2	64159
25	3	64159
26	1	64160
27	2	64161
28	2	64161
29	1	64162
30	3	64162
31	3	64162
32	1	64163
33	2	64163
34	2	64163
35	2	64163
36	3	64163
37	3	64163
38	3	64163

b) For the Requests:

json data show vehicles requested by each node .


```
"requests": [{
            "id": 1,
            "vehicle_type": 2,
            "zipcode": 64167,
            "vehicle_id": null,
            "distance": 0
        },
        {
            "id": 2,
            "vehicle_type": 3,
            "zipcode": 64160,
            "vehicle_id": null,
            "distance": 0
        },
```

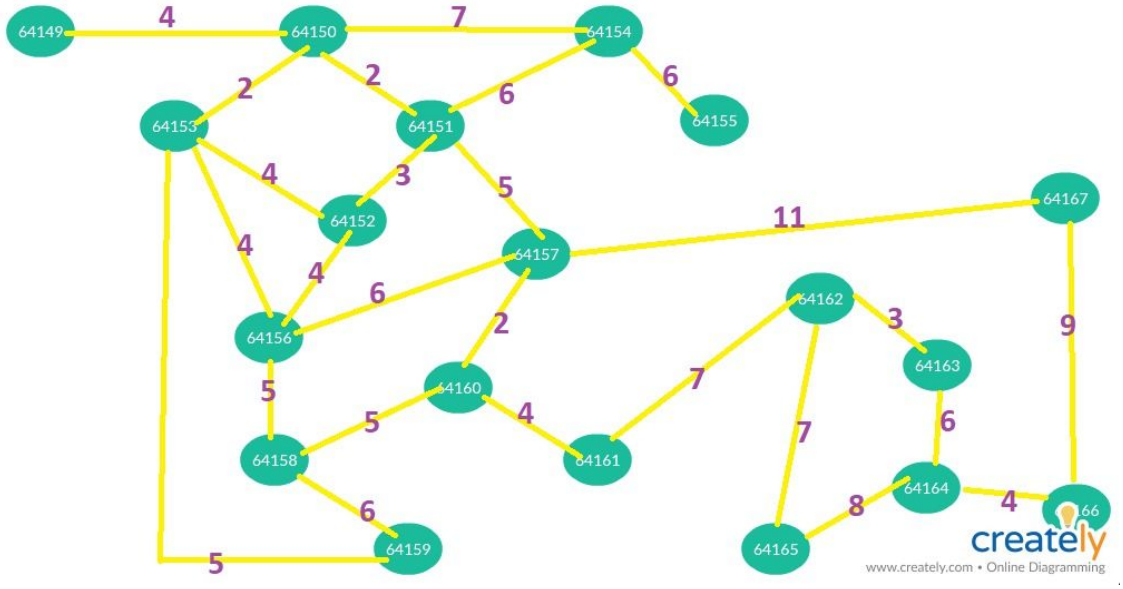
c) Distance between nodes for graph

this json files gives the length of edge between each node


```
"distances": [{
            "zipcode1": 64149,
            "zipcode2": 64150,
            "distance": 4
        },
        {
            "zipcode1": 64150,
            "zipcode2": 64151,
            "distance": 2
        },
```


Graph :

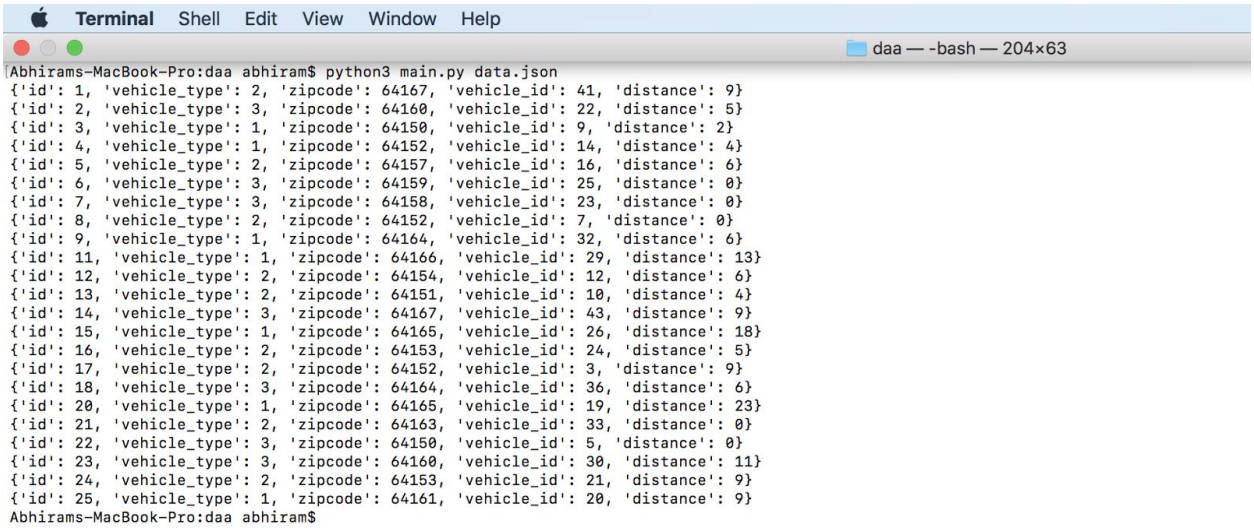
Graph is constructed from the nodes and the distances mentioned in the distance vector of json .



Requests :

Algorithms takes each request first come first serve and find the nearest node and check for availability and move to the next nearest node till it finds the node with requested vehicle availability .

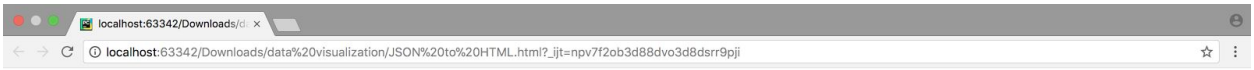
Final Output :



Testing the algorithm with limited requests :

Demo1 :

One vehicle and one request



Vehicles data

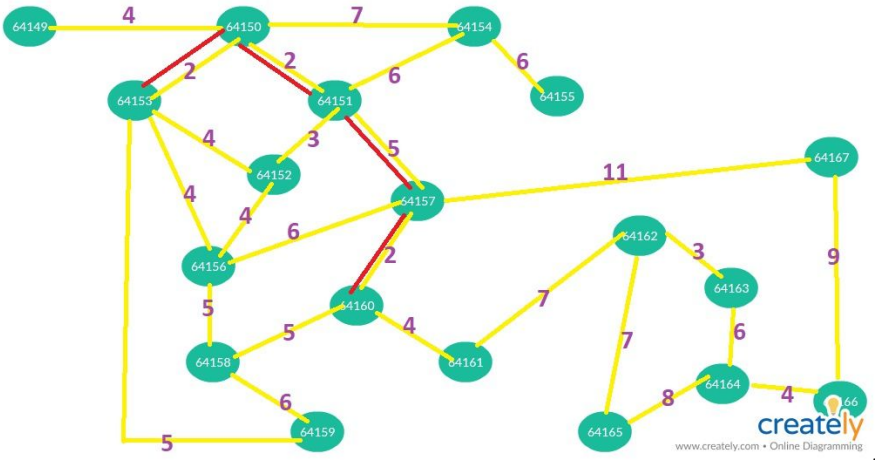
ID	VEHICLE TYPE	ZIPCODE
1	2	64160

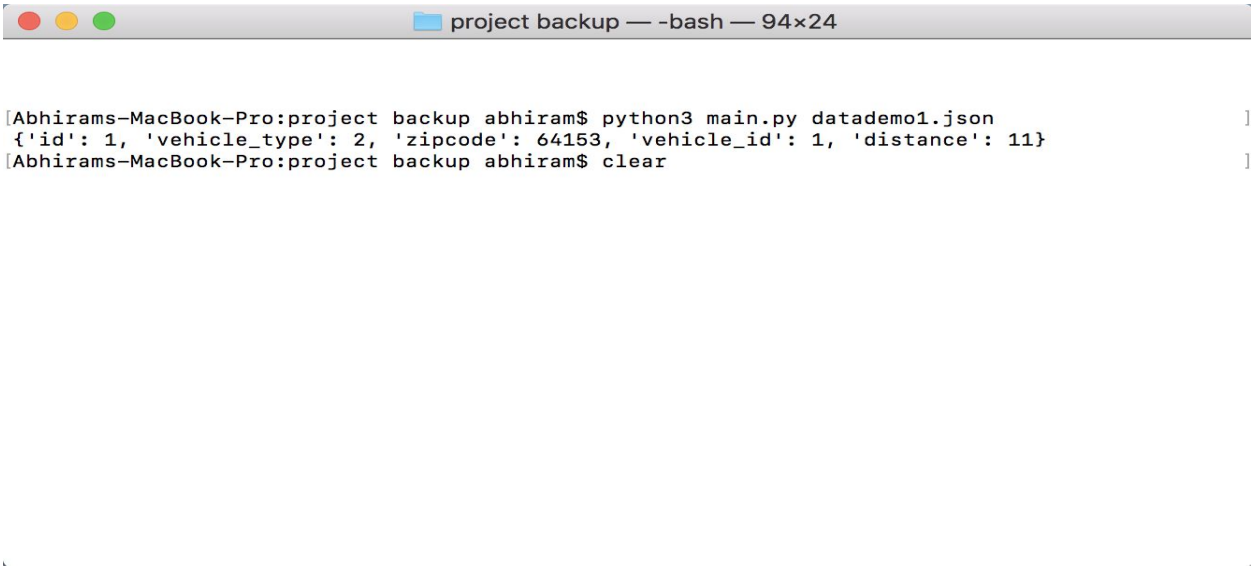
Requests data

ID	VEHICLE TYPE	ZIPCODE	VEHICLE ID	DISTANCE
1	2	64153	null	0

Vehicles data

ZIPCODE 1	ZIPCODE 2	DISTANCE
64149	64150	4
64150	64151	2
64151	64152	3
64150	64153	2
64152	64153	4



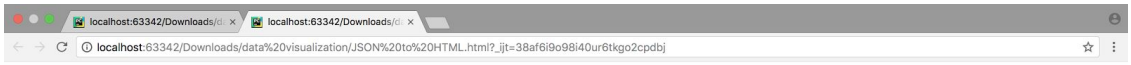


Distance from node 64160 to node 64153 . it has many paths but takes the path with minimum length , 64160 -64157-64151-64150-64153

We can see the algorithm found that algorithm found a shortest path from the requested node and node which has the vehicle requested .

Demo 2 .

One request two vehicles : we have two nodes at 64156 and 64153. we get a request from 64160 .



Vehicles data

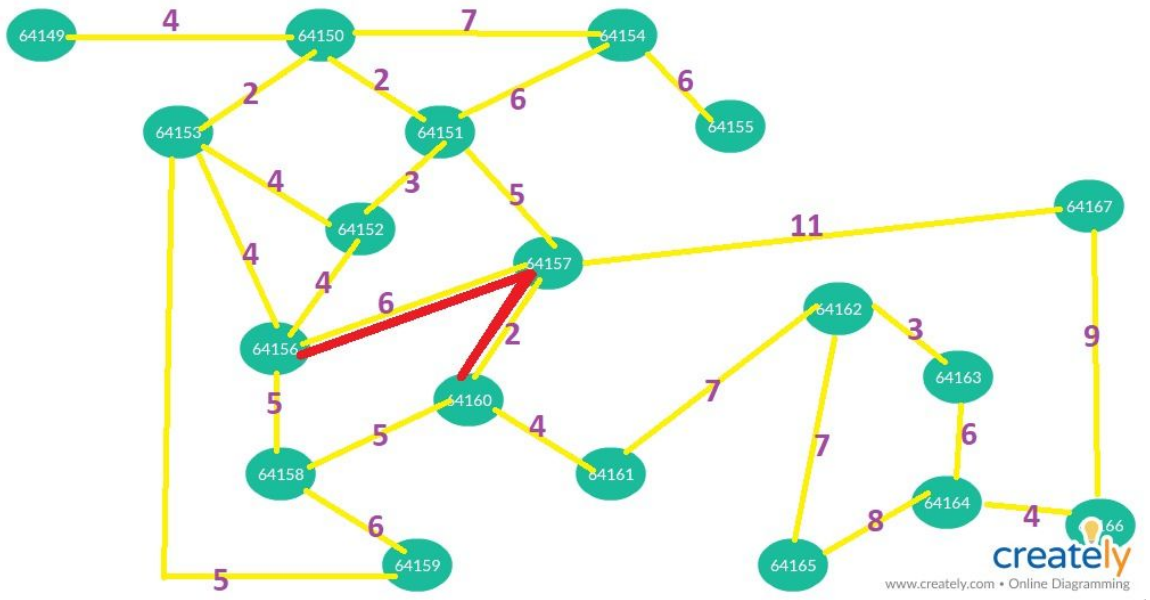
ID	VEHICLE TYPE	ZIPCODE
1	2	64156
2	2	64153

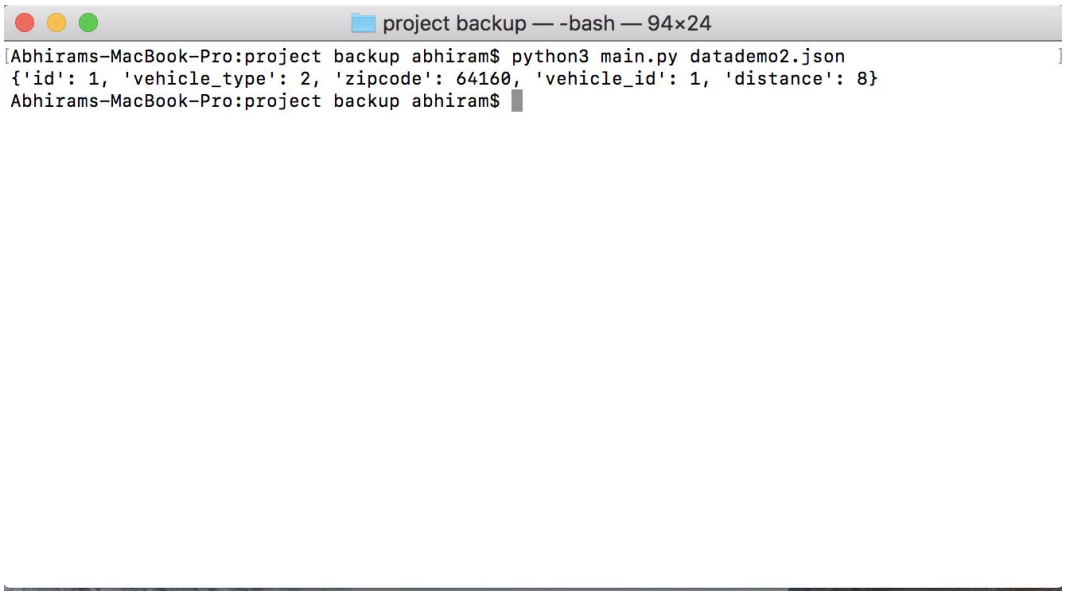
Requests data

ID	VEHICLE TYPE	ZIPCODE	VEHICLE ID	DISTANCE
1	2	64160	null	0

Vehicles data

ZIPCODE 1	ZIPCODE 2	DISTANCE
64149	64150	4
64150	64151	2
64151	64152	3
64150	64153	2





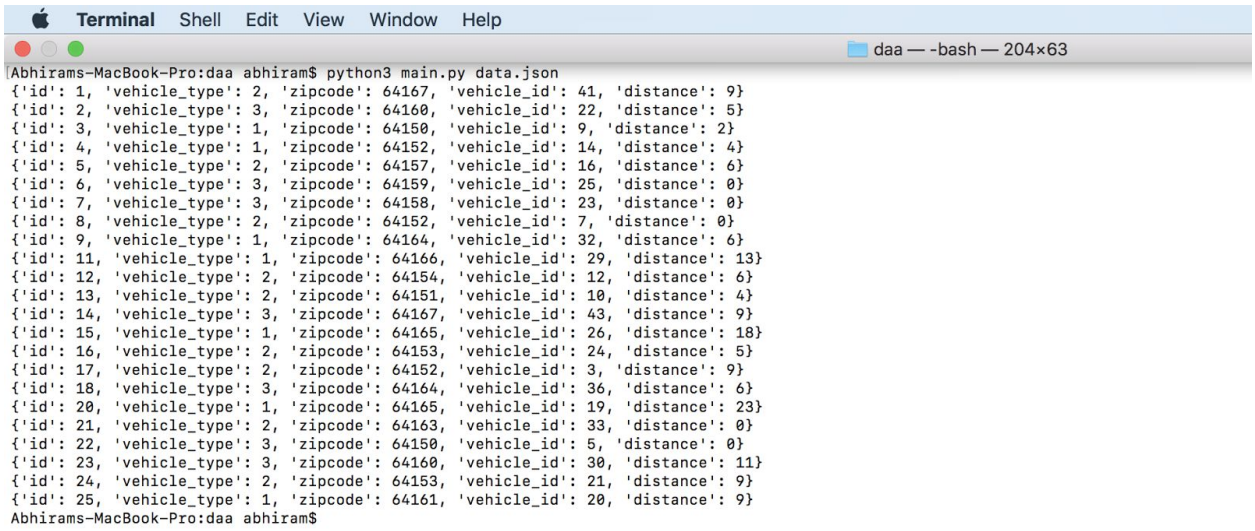
Algorithm had two options to choose between 64153 and 64156 . but it choose the nearest one among both

We can see that algorithm found the nearest node among the two nodes with the vehicle requested .

Conclusion :

Dijkstra's algorithm finds the shortest path between two nodes . Here we found distance between the node from which the request is raised to all other nodes . found shortest distance to all other nodes . then we sorted the nodes in increasing order of distance and check node with availability . the first node with the availability (requested vehicle) is the dispatcher node .

Final output :



Github link : https://github.com/abhiram383/Daaproject

Reference :

http://www.geeksforgeeks.org/greedy-algorithms-set-6-dijkstras-shortest-path-algorithm/

Dijkstras code :
http://www.bogotobogo.com/python/python_Dijkstras_Shortest_Path_Algorithm.php