UMKC

School of Computing and Engineering

COMP-SCI 5592

Design Analysis and Algorithms

Spring 2018

Final Project

Emergency vehicle Dispatch System

Submitted By,

Onica Sai Prasanna Lakshmi Rayineedi (16231918)

SreeLakshmi Nandanamudi(16244172)

Sudheesha Reddy Musku(16241536)

Sowmya Yalamanchili (16246716)

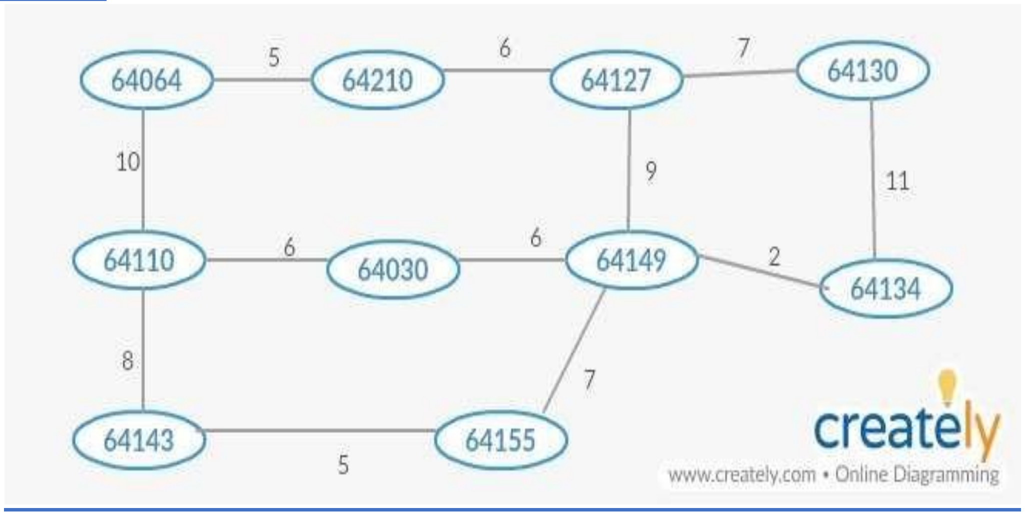
Overview Implemented a web application using java servlets, in which a vehicle will be dispatched as per the user's request. The system would choose closest available vehicle and display the vehicle distance at which it is available. The shortest path calculation is computed using Dijkstra's algorithm.

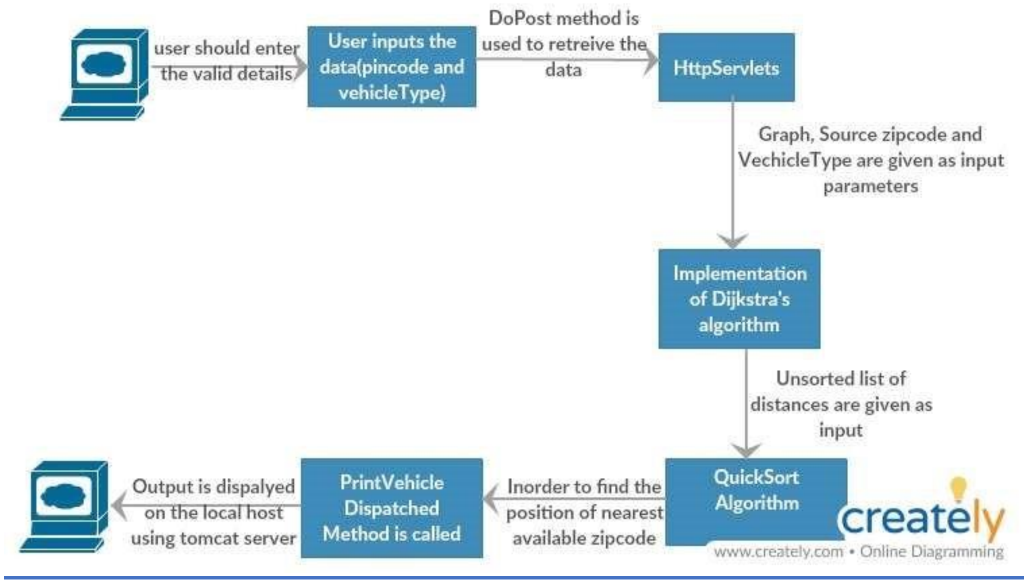
Assumptions

Following are the assumptions made while trying to calculate the distance

- All the vehicles travel at same speed irrespective of type of vehicle.
- There is no obstacle in the path of vehicle when the vehicle is in motion.
- Distance between zip codes is equal to difference between them.
- The distance is zero within that zip code.
- Once the emergency is attended, the vehicle will be relocated to the same zip code from where it was dispatched.

Flow Chart and Graph




user should enter the valid details
User inputs the data(pincode and vehicleType)
DoPost method is used to retreive the data
HttpServlets
Graph, Source zipcode and VechicleType are given as input parameters
Implementation of Dijkstra's algorithm
Unsorted list of distances are given as input
QuickSort Algorithm
Inorder to find the position of nearest available zipcode
PrintVehicle Dispatched Method is called
Output is dispalyed on the local host using tomcat server
creately
www.creately.com • Online Diagramming

Data Passing

Passing of input data to graph and listing out the vehicle availability have been discussed below.

Graph

Each zip code is considered as the node of the graph. So, we consider V=10(number of nodes) for our algorithm. The input to the graph is given as a multi-dimensional array in which each value indicates the distance from one zip code to the other.

This graph along with source location and the type of emergency vehicle required are given as the input parameters to the Dijkstra's algorithm which calculates the minimum distance between the zip codes.

Vehicle Availability

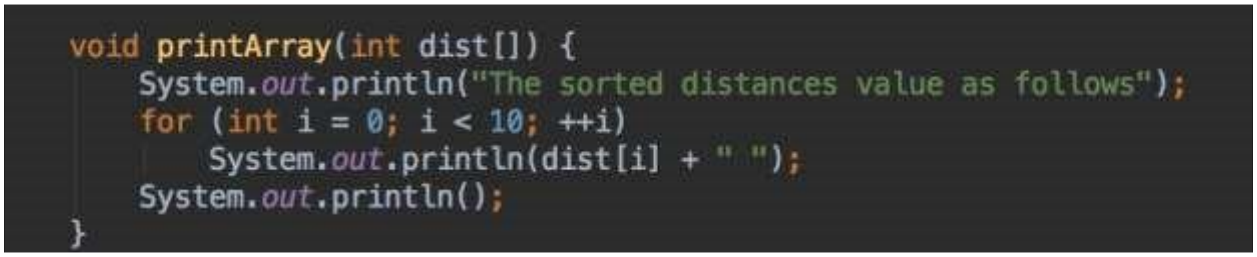
The availability for all the three types of vehicles at each node have been given as an Array List. I1, i2, i3….i10 has been considered as the list which indicates the availability of each vehicle at the zip code.

Dijkstra's Algorithm

Dijkstra's algorithm follows the concept of greedy algorithm which is used to find out the single-source shortest path for a given graph.

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

When the user enters a particular zip code shortest distance from that source to all other zip codes have been calculated by Dijkstra's algorithm. It calculates all the minimum possible distances between the nodes and that distances have been printed using the Print Array method.



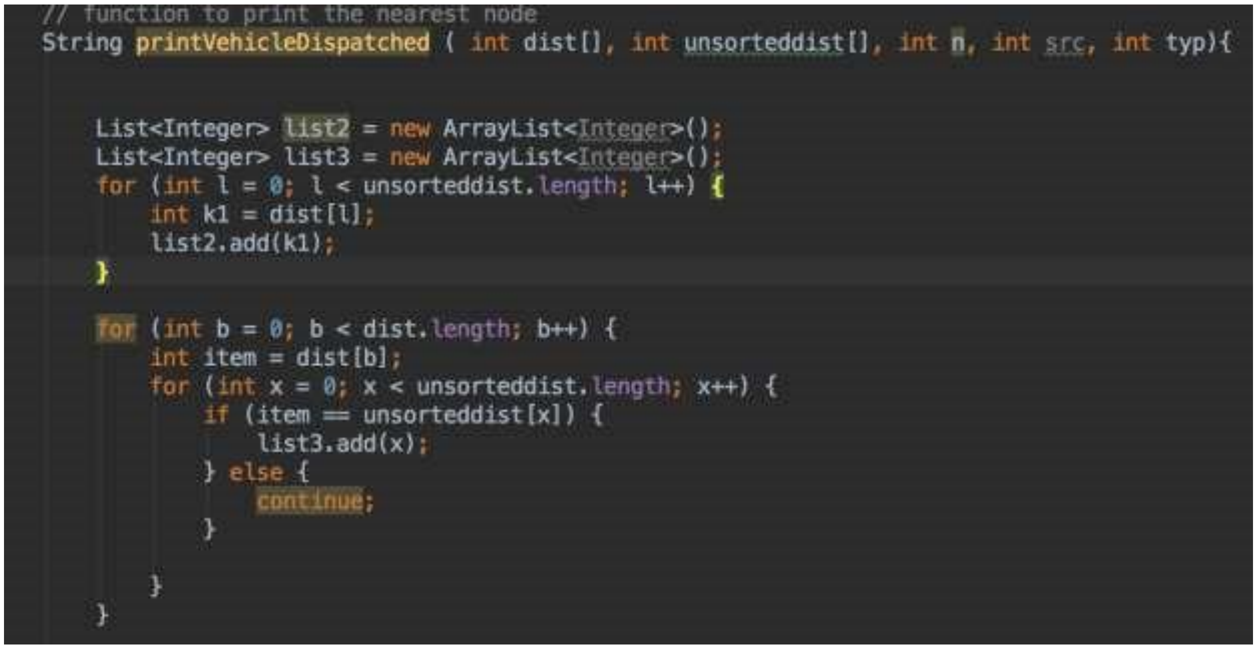
```
void printArray(int dist[]) {
    System.out.println("The sorted distances value as follows");
    for (int i = 0; i < 10; ++i)
        System.out.println(dist[i] + " ");
    System.out.println();
}
```

Quick Sort

All the distances which are calculated will be in the unsorted array and that array will be given as input to the Quick Sort which displays all the distances in order (increasing order).

PrintVehicleDispatch

Emergency vehicle is searched based on sorted distances (from Quick Sort). To find the value of the node from which we get the shortest available distance comparisons have been made between the sorted and unsorted arrays which is implemented through PrintVehicleDispatched method.



```java
// function to print the nearest node
String printVehicleDispatched ( int dist[], int unsorteddist[], int n, int src, int typ){


    List<Integer> list2 = new ArrayList<Integer>();
    List<Integer> list3 = new ArrayList<Integer>();
    for (int l = 0; l < unsorteddist.length; l++) {
        int k1 = dist[l];
        list2.add(k1);
    }

    for (int b = 0; b < dist.length; b++) {
        int item = dist[b];
        for (int x = 0; x < unsorteddist.length; x++) {
            if (item == unsorteddist[x]) {
                list3.add(x);
            } else {
                continue;
            }

        }
    }
```

After, the availability is found the vehicle will be dispatched from that node and the count for the availability will be decreased by 1.

Application flow

- When the user performs a request through web application, variables such as zip code entered, and type of the vehicle required have been obtained by the request object present in doPost method which is the main method in our java file using java servlets.
- Depending on the vehicle type selected by the user the type variable will be assigned accordingly

  If vehicletype=fire, type=0
  If vehicletype=police, type=1
  If vehicletype=ambulance, type=2

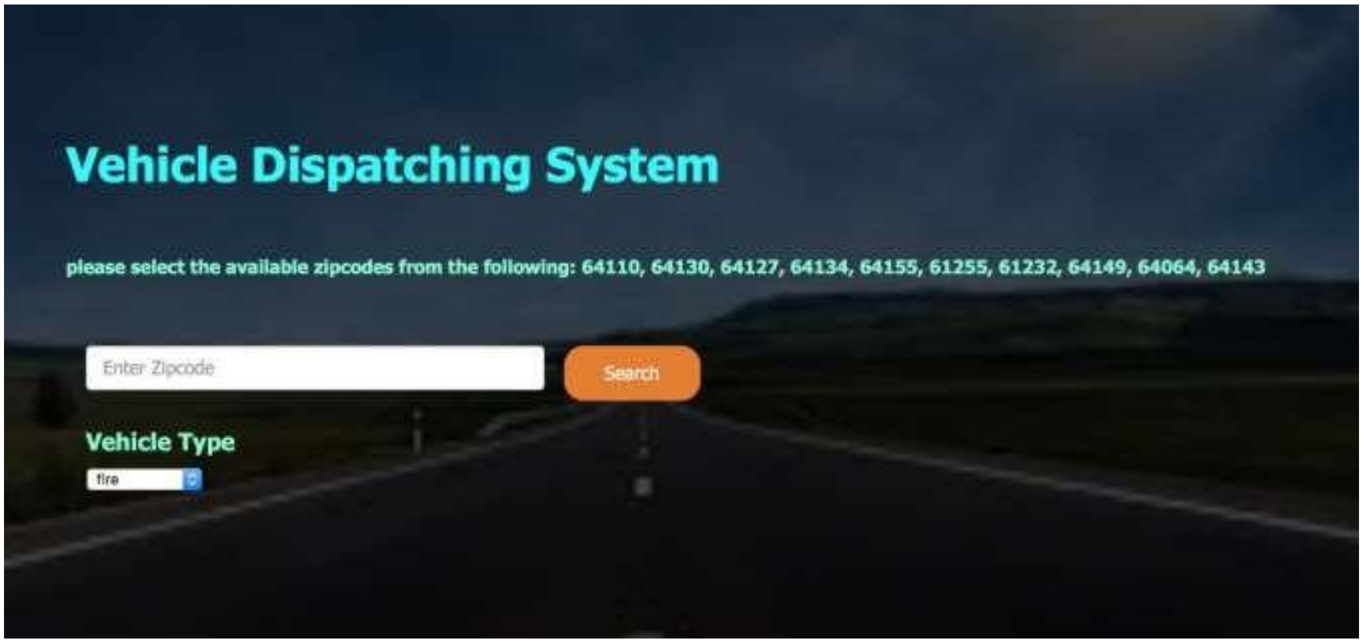
- Each zipcode used in the algorithm is considered as node for convenience. The distance between different nodes is given as an input to a multidimensional graph.
- At each node availability of the vehicles have been given through an Array-List called VehicleAvailable.
- The zipcode value entered by the user is passed as an input parameter to the switch case zipConversion which in turn returns the zip value which is same as the node number.
- The graph and node value for the zipcode(source) has been then given to the Shortestdiszipcode where Dijkstra's algorithm is implemented in which it calculates the distance between the source zipcode to all other zipcodes.
- The output from this algorithm will be the distances between source node to all other nodes and that will be displayed in unsorted order.
- So, this unsorted array is given as input to the QuickSort algorithm where it takes one element as pivot element and gives the sorted array as the final output.
- Print Array is the method used to display all the distances obtained from the Quicksort.
- Both sorted and unsorted arrays are given as input parameters to the printVehicleDispatch method along with the source zipcode and the type of vehicle required.
- By iterating through the length of sorted list and comparing each element of the sorted list to respective element in the unsorted list the position of required element is found.

- All the indexes of sorted elements from the unsorted list have been added to an array and by iterating through that index list we are checking the availability of the required element at that particular zipcode.
- If required vehicle is available at the zipcode which means that vehicleavailable(type)>0 then the vehicle have been dispatched from that destination to the source node and the vehicle availability of that particular type will be decreased by 1 or else if required vehicle is not available at the zipcode then the loop continues to find out a required vehicle from the next nearest zipcode possible.

Overall complexity of the Algorithms

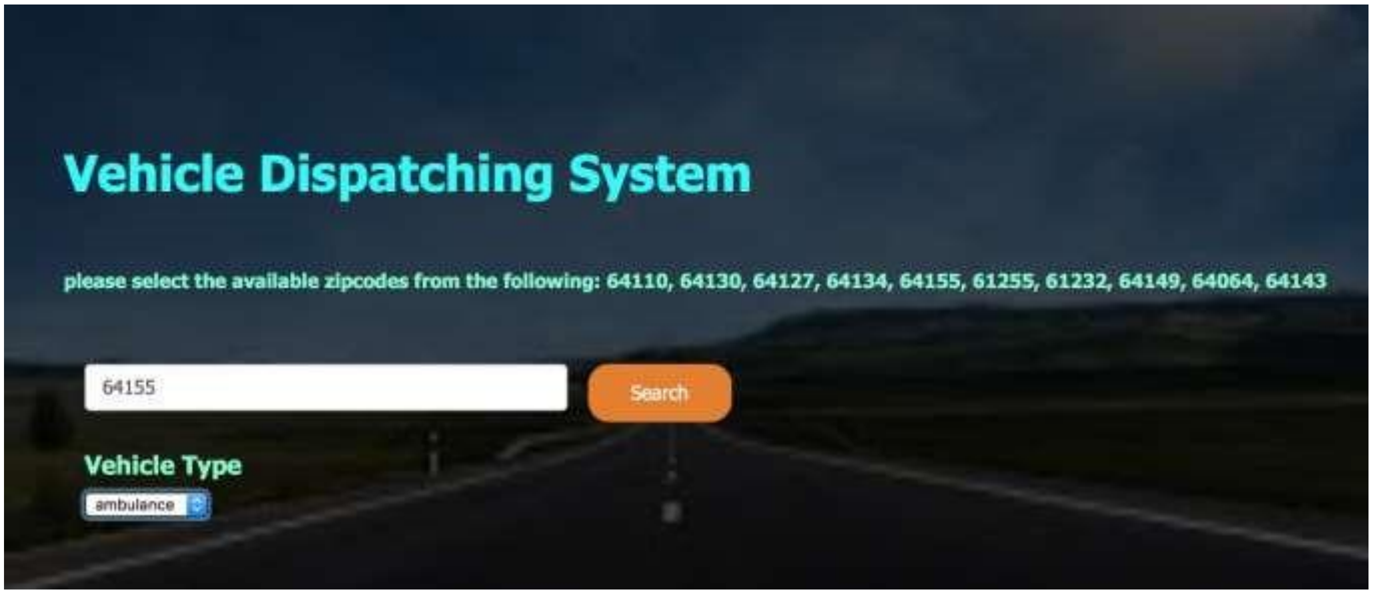
Output Screenshots

Web Application which will get displayed after executing the program using Tomcat Server on Localhost


Vehicle Dispatching System

please select the available zipcodes from the following: 64110, 64130, 64127, 64134, 64155, 61255, 61232, 64149, 64064, 64143

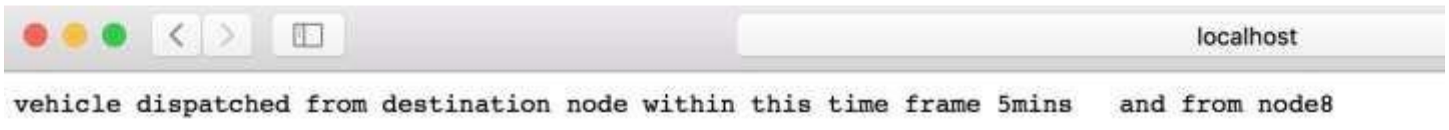
Entering the details of Zipcode and Vehicle Type


Vehicle Dispatching System

please select the available zipcodes from the following: 64110, 64130, 64127, 64134, 64155, 61255, 61232, 64149, 64064, 64143

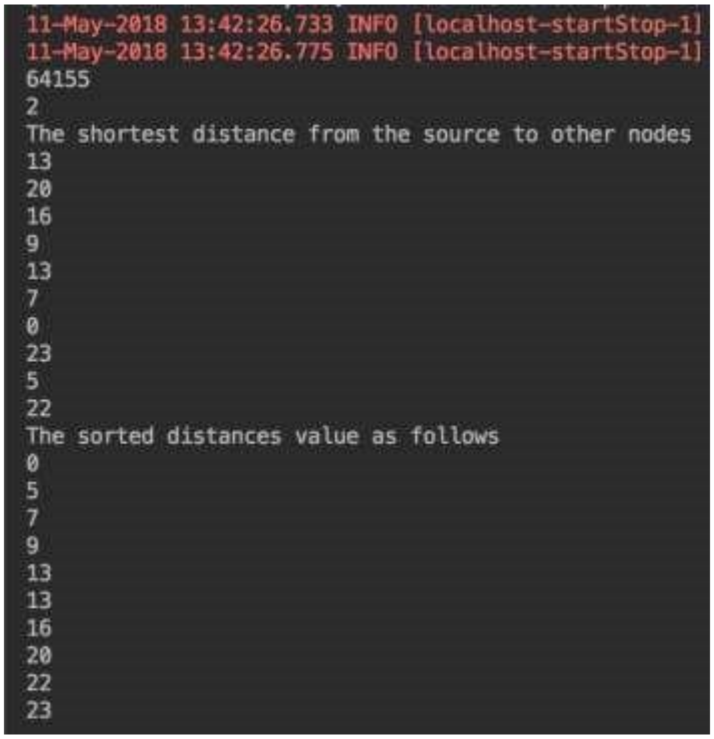
64155

Web Output



```
vehicle dispatched from destination node within this time frame 5mins   and from node8
```

Console Output



```
11-May-2018 13:42:26.733 INFO [localhost-startStop-1]
11-May-2018 13:42:26.775 INFO [localhost-startStop-1]
64155
2
The shortest distance from the source to other nodes
13
20
16
9
13
7
0
23
5
22
The sorted distances value as follows
0
5
7
9
13
13
16
20
22
23
```

Possible Enhancements

- All the Graph related data can be stored in a database and can be given as an input to the program
- Multithreading concepts can be used to handle multiple requests

Complexity of the Algorithm

As we used Quick Sort in this Algorithm Time complexity of the Algorithm is O (VlogV+ E) where E – Number of Edges and V- Number of Vertices

Github Link:

https://github.com/Sreelakshmi-N/DAA-Project

References:

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