**Design and Analysis of Algorithms**

**Final project report**

**PROJECT 2:**

**Emergency Vehicle Dispatching System**

**Ahlam Alghamdi**

**Cameron L'Ecuyer**

**Fahad Alsubaie**

**Faiz Alotaiby**

**Implementation:**

The Dijkstra part of the project is written in Python 2, while the data generation is written in Python 3. Python 2 seems to be able to run all the code as it currently stands

**Assumption:**

We have followed the technical requirements that have mentioned in the project description. Moreover, we use undirected graph, and we follow the strategy of first request first serve.

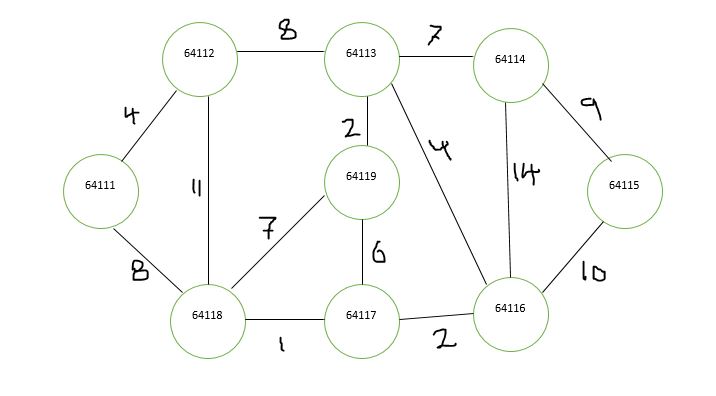
**Illustrating the Solution:**

We use Dijkstra algorithm to solve this problem. So, we have two files main and Dijkstra. Therefore, Dijkstra algorithm will be called during the main file.

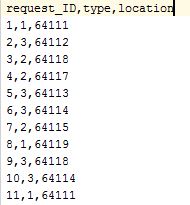
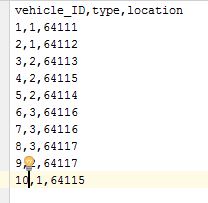
First, our algorithm will read data from CSV files and prepare data to create undirected graph. After the graph is created, an available vehicle that is in the same location of a request and the type is identical will be assigned to that request. In this case, we do not need to apply Dijkstra algorithm. Otherwise, find an available vehicle that is in the nearest location of a request and has the same type of a requested vehicle. Then, it will be assigned for the request. In this case, the Dijkstra algorithm will be called to find the shortest path from location of the request to location of an available vehicle. Then, this vehicle will not be available for any request later since it is assigned for a previous request.

In fact, we generate numerous data, but we use a small set of data to evaluate correctness of our solution. In this data set, we generate several cases to be sure that the solution is correct and accurate. The cases are: if an available and appropriate vehicle is existing in the same location, no need to apply Dijkstra. If an available and appropriate vehicle is not existing in the same location, apply Dijkstra to find the shortest location that has an available vehicle. If an available and appropriate vehicle is existing in the same location, but it assigned to a previous request, apply Dijkstra to find the shortest location that has an available vehicle. If an available and appropriate vehicle is existing only in the location that will be in the farthest location.

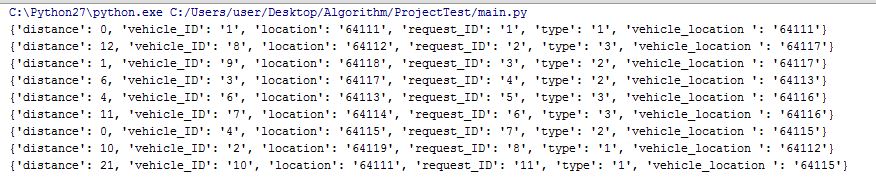
**Graph:**



**Requests and vehicles data:**

****

**Result:**



Indeed, the result is as expected. We solve it manually and compare the two results. We find the same results.

**Time Complexity:**

Let assume that K is number of requests, N is number of vehicles, V is number of vertices, and E is number of edges. The Time Complexity is O(K\*N\*(ElogV)). The Dijkstra uses min heap in implementation, so its performance by using heap is better than using matrix.

The Time Complexity of the graph generation is O(V^5) in worst case, where V is the number of vertices, but behaves closer to O(V^4) in most cases. O(V^3) is created just from doing matrix multiplication to determine if the graph is fully connected.

**GitHub:**

https://github.com/camlecuyer/DesignAnalysisProject

**References:**

<https://stackoverflow.com>

https://www.geeksforgeeks.org/greedy-algorithms-set-7-dijkstras-algorithm-for-adjacency-list-representation/