IMPERIAL COLLEGE LONDON, DEPARTMENT OF MATHEMATICS

Scientific Computing (M3SC) Project 1

Omar Haque

February 20, 2017

1 MAIN SOLUTION

My solution strategy will focus on following the 5 step algorithm given in the question in its entirety, without focusing on major changes for speedup. However, I will at the same time not store all the values as the cars at each node are updated at each iteration.

```
1  # imports
2  import Dijkstra as dijk
3  import misc
4  import numpy as np
5  import csv
6  # this import is needed for the last question
7  from solution_accident_occurs import max_index_tracker_no30
8
9
10 # ------------------------------------------------------------
11 # --------------    FUNCTIONS USED     ---------------------
12 # ------------------------------------------------------------
13
14 def next_node(path):
15     """ Returns the next index (after the node itself) in the path.
16         If the path contains only one node, returns the node itself.
17     """
18     if len(path) == 1:
19         return path[0]
```

```
    else:
        return path[1]


def update_weight_matrix(epsilon, c, original_weight_matrix, noNodes=58):
    """
    This function updates the weight matrix according to step 5 of the
    Project. Note the added fix - the weight matrix is not changed if
    the original entry was 0.



    :param epsilon: given in question
    :param c: the vector containing number of cars at each node
    :param original_weight_matrix: the weight matrix given by RomeEdges
    :param noNodes: number of nodes in the system
    :return: the updated weight matrix
    """
    new_weight_matrix = np.zeros((noNodes, noNodes))
    for i in range(noNodes):
        for j in range(noNodes):
            if original_weight_matrix[i, j] != float(0):
                new_weight_matrix[i, j] = original_weight_matrix[i, j] + \
                                          (epsilon * (float(c[i]) +
                                                      float(c[j]))) / float(2)
    return new_weight_matrix

def extract_data():
    """
    This function opens the RomeVertices and RomeEdges files, and creates
    global variables RomeX, RomeY, RomeA, RomeB and RomeV. These are variables
    used to create the original weight matrix.

    """
    global RomeX, RomeY, RomeA, RomeB, RomeV
    RomeX = np.empty(0, dtype=float)
    RomeY = np.empty(0, dtype=float)
    with open('./data/RomeVertices', 'r') as file:
        AAA = csv.reader(file)
        for row in AAA:
            RomeX = np.concatenate((RomeX, [float(row[1])]))
            RomeY = np.concatenate((RomeY, [float(row[2])]))
    file.close()
    RomeA = np.empty(0, dtype=int)
```

```
64      RomeB = np.empty(0, dtype=int)
65      RomeV = np.empty(0, dtype=float)
66      with open('./data/RomeEdges2', 'r') as file:
67          AAA = csv.reader(file)
68          for row in AAA:
69              RomeA = np.concatenate((RomeA, [int(row[0])]))
70              RomeB = np.concatenate((RomeB, [int(row[1])]))
71              RomeV = np.concatenate((RomeV, [float(row[2])]))
72      file.close()
```